VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CURRICULUM AND SYLLABI

(2026-2027)

B. Tech. Computer Science and Engineering

VISION STATEMENT OF VELLORE INSTITUTE OF TECHNOLOGY

Transforming life through excellence in education and research.

MISSION STATEMENT OF VELLORE INSTITUTE OF TECHNOLOGY

World class Education: Excellence in education, grounded in ethics and critical thinking, for improvement of life.
Cutting edge Research: An innovation ecosystem to extend knowledge and solve critical problems.
Impactful People: Happy, accountable, caring and effective workforce and students.
Rewarding Co-creations: Active collaboration with national & international industries & universities for productivity and economic development.
Service to Society: Service to the region and world through knowledge and compassion.

VISION STATEMENT OF THE SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

To be a world-renowned centre of education, research and service in computing and allied domains.

MISSION STATEMENT OF THE SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

To offer computing education programs with the goal that the students become technically competent and develop lifelong learning skill.

To undertake path-breaking research that creates new computing technologies and solutions for industry and society at large.

To foster vibrant outreach programs for industry, research organizations, academia and society.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Computer Science and Engineering

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

1. Graduates will be engineering practitioners and leaders, who would help solve industry's technological problems.
2. Graduates will be engineering professionals, innovators or entrepreneurs engaged in technology development, technology deployment, or engineering system implementation in industry.
3. Graduates will function in their profession with social awareness and responsibility.
4. Graduates will interact with their peers in other disciplines in industry and society and contribute to the economic growth of the country.
5. Graduates will be successful in pursuing higher studies in engineering or management.
6. Graduates will pursue career paths in teaching or research.

PROGRAMME OUTCOMES (POs)

PO_01: Having an ability to apply mathematics and science in engineering applications.

PO_02: Having a clear understanding of the subject related concepts and of contemporary issues and apply them to identify, formulate and analyse complex engineering problems.

PO_03: Having an ability to design a component or a product applying all the relevant standards and with realistic constraints, including public health, safety, culture, society and environment

PO_04: Having an ability to design and conduct experiments, as well as to analyse and interpret data, and synthesis of information

PO_05: Having an ability to use techniques, skills, resources and modern engineering and IT tools necessary for engineering practice

PO_06: Having problem solving ability- to assess social issues (societal, health, safety, legal and cultural) and engineering problems

PO_07: Having adaptive thinking and adaptability in relation to environmental context and sustainable development

PO_08: Having a clear understanding of professional and ethical responsibility

PO_09: Having cross cultural competency exhibited by working as a member or in teams

PO_10: Having a good working knowledge of communicating in English – communication with engineering community and society

PO_11: Having a good cognitive load management skills related to project management and finance

PO_12: Having interest and recognise the need for independent and lifelong learning

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Computer Science and Engineering

PROGRAMME SPECIFIC OUTCOMES (PSOs)

1. Apply computing theory, languages and algorithms, as well as mathematical and statistical models, and the principles of optimization to appropriately formulate and use data analysis.

2. Apply the principles and techniques of database design, administration, and implementation to enhance data collection capabilities and decision-support systems. Ability to critique the role of information and analytics in supporting business processes and functions.

3. Invent and use appropriate models of data analysis, assess the quality of input, derive insight from results, and investigate potential issues. Also to organize big data sets into meaningful structures, incorporating data profiling and quality standards.

Category Credit Detail			
Sl.No.	**Description**	**Credits**	**Maximum Credit**
1	NC - Non Credit Course	2	2
2	UCC - University Core Courses	60	60
3	PCC - Programme Core Courses	40	40
4	CON - Concentration	20	20
5	OEC - Open Elective Courses	40	40
Total Credits		162	

Non Credit Course									
sl.no	**Course Code**	**Course Title**	**Course Type**	**Version**	**L**	**T**	**P**	**J**	**Credits**
1	BAENG100	Effective English Communication	Lab Only	1.0	0	0	4	0	2.0

University Core Courses									
sl.no	**Course Code**	**Course Title**	**Course Type**	**Version**	**L**	**T**	**P**	**J**	**Credits**
1	BACHY101	Environmental Sciences	Online Course	1.0	0	0	0	0	2.0
2	BACHY105	Applied Chemistry	Embedded Theory and Lab	1.0	3	0	2	0	4.0
3	BACSE101	Problem Solving using Python	Lab Only	1.0	0	0	4	0	2.0
4	BACSE102	Problem Solving using Java	Lab Only	1.0	0	0	4	0	2.0
5	BACSE191	Basic Multidisciplinary Project	Project	1.0	0	0	0	0	2.0
6	BACSE291	Innovative Design Project	Project	1.0	0	0	0	0	2.0
7	BACSE399	Internship I	Project	1.0	0	0	0	0	1.0
8	BAEEE101	Basic Engineering	Embedded Theory and Lab	1.0	3	0	2	0	4.0
9	BAENG101	Technical English Communication	Embedded Theory and Lab	1.0	3	0	2	0	4.0
10	BAFLC100	Foreign Language	Basket	1.0	0	0	0	0	2.0
11	BAHSM100	Humanities, Social Science and Management	Basket	1.0	0	0	0	0	3.0
12	BAHUM101	India Studies	Online Course	1.0	0	0	0	0	1.0
13	BAMAT101	Multivariable Calculus and Differential Equations	Embedded Theory and Lab	1.0	3	0	2	0	4.0
14	BAMAT207	Probability and Statistics	Embedded Theory and Lab	1.0	3	0	2	0	4.0
15	BAPHY105	Engineering Physics	Embedded Theory and Lab	1.0	3	0	2	0	4.0
16	BASTS101	Qualitative and Quantitative Skills Practice I	Soft Skill	1.0	3	0	0	0	1.0
17	BASTS102	Qualitative and Quantitative Skills Practice II	Soft Skill	1.0	3	0	0	0	1.0

Programme Core Courses									
sl.no	**Course Code**	**Course Title**	**Course Type**	**Version**	**L**	**T**	**P**	**J**	**Credits**
1	BACSE103	Computation Structures	Embedded Theory and Lab	1.0	3	0	2	0	4.0
2	BACSE104	Structured and Object-Oriented Programming	Embedded Theory and Lab	1.0	2	0	4	0	4.0
3	BACSE105	Data Structures and Algorithms	Embedded Theory and Lab	1.0	3	0	2	0	4.0
4	BACSE106	Operating Systems	Embedded Theory and Lab	1.0	3	0	2	0	4.0
5	BACSE201	Models of Computation	Theory Only	1.0	3	1	0	0	4.0
6	BACSE202	Database Systems	Embedded Theory and Lab	1.0	3	0	2	0	4.0
7	BACSE203	Computer Networks	Embedded Theory and Lab	1.0	3	0	2	0	4.0
8	BACSE204	Software Engineering	Embedded Theory and Lab	1.0	3	0	2	0	4.0
9	BACSE205	Fundamentals of Artificial Intelligence and Machine Learning	Embedded Theory and Lab	1.0	3	0	2	0	4.0
10	BAMAT205	Discrete Mathematics and Linear Algebra	Theory Only	1.0	3	1	0	0	4.0

Concentration									
sl.no	**Course Code**	**Course Title**	**Course Type**	**Version**	**L**	**T**	**P**	**J**	**Credits**
1	BABCE001	Systems Engineering	Basket	1.0	0	0	0	0	20.0

Open Elective Courses									
sl.no	**Course Code**	**Course Title**	**Course Type**	**Version**	**L**	**T**	**P**	**J**	**Credits**
1	BAESP201	Spanish Level II	Embedded Theory and Lab	1.0	2	0	2	0	3.0
2	BAFRE201	French for Engineers: Language and Communication Lab	Embedded Theory and Lab	1.0	2	0	2	0	3.0
3	BAGER201	German Level II	Embedded Theory and Lab	1.0	2	0	2	0	3.0
4	BAHIN101	Prathamik Hindi	Theory Only	1.0	3	0	0	0	3.0
5	BAHIN102	Prayojanmulak Hindi	Embedded Theory and Lab	1.0	2	0	2	0	3.0
6	BAJAP201	Japanese Level II	Embedded Theory and Lab	1.0	2	0	2	0	3.0
7	BOPES001	B.Tech. Open Elective Courses	Basket	1.0	0	0	0	0	40.0
8	ULINK100	Undergraduate Lab Integration for Knowledge & Innovation	Basket	1.0	0	0	0	0	3.0

Course code	Course title	L	T	P	J	C
XXXXX	Effective English	0	0	4	0	2
Pre-requisite	Nil	Syllabus version				
		1.0				

Course Objectives

1. To develop fluency in basic communication skills at various academic and Social Contexts.
2. To help students overcome learning difficulties in grammar, vocabulary and pronunciation.

Course Outcomes

Upon completion of the course, learners will be able to:

1. Illustrate listening and reading competence in general and academic contexts
2. Demonstrate sufficient speaking skills in terms of pronunciation and discourse management
3. Apply grammar rules in oral and written communication
4. Employ appropriate vocabulary in diverse situations

Module:1	Listening	5 hours

Listening for General Information, Listening for Specific Information -Listening to formal and informal conversations, listening to talks

Activity: Listening and note-taking

Module:2	Pronunciation	8 hours

Reading Aloud - short and long passages, Listen & Repeat Drills - words with common endings, Tongue Twisters, Mute Words, Commonly Mispronounced Words

Activity: Reading a Passage Aloud

Module:3	Speaking	8 hours

Self-Introduction, Situational Role Play – Formal and Informal Situations, Picture Description - Individual and Group work

Activity: Role Play

Module:4	Grammar	10 hours

Parts of Speech – An overview of the eight parts of speech along with the simple and the progressive tense, Types of Sentences: Affirmative Sentences, Negative Sentences, Interrogative Sentences, and Exclamatory Sentences

Activity: Interactive Worksheets and Pair/Group Activities

Module:5	Vocabulary	8 hours

Synonyms, Antonyms, Homophones, Homonyms, Everyday Vocabulary – Words Describing the Weather, People, and Emotions

Activity: Interactive Worksheets and Pair/Group Activities

Module:6	Reading	10 hours

Activity: Reading Comprehension, Reading Newspaper Articles

Module:7	Writing	11 hours

Activity: Jumbled Sentences, Parts of the Paragraph, Paragraph Writing - Guided and Unguided

	Total Lecture hours:	60 hours

Text Book(s)

1. Chase, B.T. & Johannsen, K.L. (2018). Pathways: Listening, speaking and critical thinking 1. 2nd ed. National Geographic Learning.
2. Douglas, N. & Bohlke, D. (2020). Reading explorer 3. 3rd ed. National Geographic Learning.

3.	Murphy, R. (2024). Essential English grammar. 2nd ed. Cambridge University Press.
Reference Books	
1.	Elbaum, S.N. & Peman, J.P. (2021). Grammar in context: Basic. 7th ed. National Geographic Learning.
2.	Folse, K. (2014). Great writing 1: Great sentences for great paragraphs. 4th ed. National Geographic Learning.
3.	Redman, S. (2017). English vocabulary in use: Pre-intermediate and intermediate. 4th ed. Cambridge University Press.
4.	Singleton, J. (2005). Writers at work: The paragraph. Cambridge University Press.
Mode of Assessment: Continuous assessment / FAT / Oral examination and others	

Recommended by Board of Studies	19-02-2025		
Approved by Academic Council	No. 78	Date	12.06.2025

Course Code	Course Title	L	T	P	C
BACHY101	Environmental sciences	2	0	0	2
Pre-requisite	nil	**Syllabus Version**			
		1			

Course Objectives

To equip themselves with foundational knowledge of environmental systems, sustainability principles, and pollution control methods, to support applications in real-world engineering scenarios.
To develop requisite skills related to sustainable practices, environmental protection, and policy frameworks.
To make students understand and appreciate the unity of life in all its forms, the implications of life style on the environment.

Course Outcomes

Explain key environmental concepts like ecosystems and biodiversity. (CO 1)
Interpret the impact of energy usage, pollution, and waste on environmental quality. (CO 2)
Apply knowledge of environmental laws and sustainable development goals to achieve practical solutions for environmental protection. (CO 3), 4. Evaluate the human impact on climate change and the role of IT and AI in mitigating environmental issues. (CO 4)

Module:1	**Fundamentals of Environment and Ecosystems**	**6 hours**

Definition and scope of environmental science, Earth as a life-support system, Ecosystem definition, Components of the environment (biotic & abiotic), Structure and function of ecosystems, Energy flow in an ecosystem, and ecological succession.

Module:2	**Biodiversity and Environmental Quality**	**6 hours**

Biodiversity definition, levels, significance, and threats. Species interaction, types-extinct, endemic, endangered, and rare species. Conservation strategies and genetically modified crops. Environmental hazards: definition, types, causes, and solutions: Biological (Malaria, COVID-19), chemical (BPA, heavy metals), and disaster management.

Module:3	**Pollution and Its Management Techniques**	**6 hours**

Types of pollution: air, water, soil, noise, thermal- causes, effects, and control methods, water management, its conservation, circular economy and solid waste management techniques.		
Module:4	**Environmental Laws and Sustainable development goals**	**6 hours**
Environmental impact assessment (EIA), Key national and international environmental laws- Environmental protection act (EPA) – Air, Water, forest conservation and wildlife protection acts, Kyoto and Paris Agreement. Introduction to Sustainable development goals (SDGss).		
Module:5	**Climate Change and Emerging Environmental Issues**	**4 hours**
Climate change science and global warming, carbon footprint and mitigation strategies, role of AI/IoT in environmental monitoring.		
Module:6	**Contemporary topics**	**2 hours**
Lecture on green energy technologies		
	Total Lecture Hours:	**30 hours**
Text Book(s)		
G. Tyler Miller and Scott E. Spoolman, "**Environmental Science,** ", Cengage learning, 15th Edition, **2020** Erach Bharucha, "**Textbook of Environmental Studies for Undergraduate students,** ", Orient Blackswan Pvt Ltd, 4th Edition, **2023**		
Reference Books		
Richard T. Wright & Dorothy F. Boorse , "**Environmental Science: Toward a Sustainable Future**", Pearson Education, 13th Edition, **2020** Anubha Kaushik & C.P Kaushik , "**Perspectives in Environmental Studies,**", New Age International Publishers, 7th Edition, **2021** Benny Joseph, "**Environmental Studies**", McGraw Hill Education, 3rd Edition, **2021**		
Mode of Evaluation : Quiz		
Recommended by Board of Studies :	23-05-2025	
Approved by Academic Council : No. 78	12-06-2025	

Course Code	Course Title	L	T	P	C
BACHY105	Applied Chemistry	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

To enable students to learn fundamental concepts in different branches of chemistry
To be well-versed in analytical and computational skills in chemistry
To teach practical skills in chemistry and apply knowledge for societal applications

Course Outcomes

Apply the concepts of thermodynamics and kinetics
Use chemical concepts for the advancement of materials in electrical, magnetic, and energy domains
Demonstrate UV Visible spectroscopic and X- ray diffraction techniques
Discuss molecular orbitals and its energies through computational chemistry
Apply instrumental, wet-chemical, and computational methods to analyze materials, molecules, and ions

Module:1	**Chemical thermodynamics and kinetics**	**9 hours**

Introduction-Terminologies- Zeroth law- First law of thermodynamics – Enthalpy and heat capacity – Applications of first law of thermodynamics – Second law of thermodynamics – Carnot cycle- Third law of thermodynamics – Gibbs Helmholtz equation. Chemical Kinetics – Rate equation – Order of the reaction – Integral rate equations – Half life time of reactions- Molecularity – Collison theory of reaction rate – Effect of temperature and catalysts on reaction rate -Numericals

Module:2	**Energy devices**	**8 hours**

Electrochemical and electrolytic cells – electrode materials with examples (semi-conductors), classifications of batteries - chemistry of Li ion secondary batteries, Fuel cells: H_2-O_2 and solid oxide fuel cell (SOFC); Solar cells - photovoltaic cell (silicon based), photoelectrochemical cells and dye-sensitized cells

Module:3	**Functional materials**	**10 hours**

Electrical materials: Ohm's law, resistivity, classification - metals, semiconductors (intrinsic & extrinsic - Si), insulators, superconductors. Magnetic materials: para, dia, ferro & anti-ferro magnetism, applications in storage devices. Liquid cooling **–** types of coolants, thermal conductivity, and applications. Nanomaterials: introduction, top-down and bottom-up approaches for synthesis - bulk *vs* nano (gold), quantum dots

Module:4	**Spectroscopic and diffraction techniques**	**6 hours**
Fundamental concepts in spectroscopic and diffraction techniques - Principle, instrumentation and applications of UV-Visible and powder XRD (determination of crystallite size, numericals)		
Module:5	**Computational Chemistry Basics to applications**	**10 hours**
Introduction-Quantum mechanical calculations. Molecular orbital theory: atomic orbitals, molecular orbitals. Orbital interactions- chemical bonding. Potential energy curve/surface, electrostatic potentials, geometry optimization, frequency, reaction mechanism, Application of AI to screen semiconductors		
Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

Theodore E. Brown, H Eugene, LeMay Bruce E. Bursten, Catherine Murphy, Patrick Woodward, Matthew E. Stoltzfus, "**Chemistry: The Central Science**", Pearson Publishers, 14th Edition, **2017**
Frank Jensen, "**Introduction to computational chemistry** ", Wiley, 3rd Edition, **2017**

Reference Books

A.R West , "**Solid State Chemistry and its Applications**", Wiley, 2nd Edition, **2014**
Lawrence S. Brown and Thomas Holme , "**Chemistry for engineering students**", Brooks Cole, 4th Edition, **2018**

Indicative Experiments

1. Thermodynamic functions from EMF measurements: Zn – Cu system. Evaluation of properties enthalpy, entropy and free energy	2 hours
2. Determination of reaction rate, order and molecularity - ester hydrolysis (ethylacetate)	2 hours
3. Colorimetric estimation of Ni^{2+} using conventional and smartphone digital-imaging method	2 hours
4. Analysis of iron in carbon steel by potentiometry	2 hours
5. Preparation of ZnO semiconductor and its characterization	2 hours
6. Estimation of sulfate ion in drinking water by the conductivity method	2 hours

7. Build atoms and molecules, visualization of atomic orbitals and hybridized orbitals - calculating the orbital contributions	2 hours
8. Conformational analysis of cyclohexane and ethane molecules and plotting the potential energy profile	2 hours
9. Application of AI to screen semiconductors, materials, drugs - case study	2 hours
10. Size-dependent colour variation of Cu_2O nanoparticles by spectrophotometer	2 hours
11. Laboratory synthesis of hydroxyl apatite and its use for the removal of heavy metals	2 hours
12. Challenging experiment related to evaluation of the state of charge of batteries (SOC)	2 hours
13. Construction of DSSC and its working	2 hours
14. Preparation of Ag metal nanoparticles and their characterization	2 hours
15. Colorimetric estimation of Fe^{2+} using conventional and smartphone digital-imaging method	2 hours
Total Laboratory Hours:	**30 hours**

Text Book(s)

Arthur Vogel and John Mendham, "**Vogel's Textbook of Quantitative Chemical Analysis**", Pearson Education, 6th Edition, **2009**

Mode of Evaluation : Continuous Assessment Test, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment

Recommended by Board of Studies :	23-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACHY105	Applied Chemistry	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

To enable students to learn fundamental concepts in different branches of chemistry
To be well-versed in analytical and computational skills in chemistry
To teach practical skills in chemistry and apply knowledge for societal applications

Course Outcomes

Apply the concepts of thermodynamics and kinetics
Use chemical concepts for the advancement of materials in electrical, magnetic, and energy domains
Demonstrate UV Visible spectroscopic and X- ray diffraction techniques
Discuss molecular orbitals and its energies through computational chemistry
Apply instrumental, wet-chemical, and computational methods to analyze materials, molecules, and ions

Module:1	**Chemical thermodynamics and kinetics**	**9 hours**

Introduction-Terminologies- Zeroth law- First law of thermodynamics – Enthalpy and heat capacity – Applications of first law of thermodynamics – Second law of thermodynamics – Carnot cycle- Third law of thermodynamics – Gibbs Helmholtz equation. Chemical Kinetics – Rate equation – Order of the reaction – Integral rate equations – Half life time of reactions- Molecularity – Collison theory of reaction rate – Effect of temperature and catalysts on reaction rate -Numericals

Module:2	**Energy devices**	**8 hours**

Electrochemical and electrolytic cells – electrode materials with examples (semi-conductors), classifications of batteries - chemistry of Li ion secondary batteries, Fuel cells: H_2-O_2 and solid oxide fuel cell (SOFC); Solar cells - photovoltaic cell (silicon based), photoelectrochemical cells and dye-sensitized cells

Module:3	**Functional materials**	**10 hours**

Electrical materials: Ohm's law, resistivity, classification - metals, semiconductors (intrinsic & extrinsic - Si), insulators, superconductors. Magnetic materials: para, dia, ferro & anti-ferro magnetism, applications in storage devices. Liquid cooling **–** types of coolants, thermal conductivity, and applications. Nanomaterials: introduction, top-down and bottom-up approaches for synthesis - bulk *vs* nano (gold), quantum dots

Module:4	**Spectroscopic and diffraction techniques**	**6 hours**
Fundamental concepts in spectroscopic and diffraction techniques - Principle, instrumentation and applications of UV-Visible and powder XRD (determination of crystallite size, numericals)		
Module:5	**Computational Chemistry Basics to applications**	**10 hours**
Introduction-Quantum mechanical calculations. Molecular orbital theory: atomic orbitals, molecular orbitals. Orbital interactions- chemical bonding. Potential energy curve/surface, electrostatic potentials, geometry optimization, frequency, reaction mechanism, Application of AI to screen semiconductors		
Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

Theodore E. Brown, H Eugene, LeMay Bruce E. Bursten, Catherine Murphy, Patrick Woodward, Matthew E. Stoltzfus, "**Chemistry: The Central Science**", Pearson Publishers, 14th Edition, **2017**
Frank Jensen, "**Introduction to computational chemistry** ", Wiley, 3rd Edition, **2017**

Reference Books

A.R West , "**Solid State Chemistry and its Applications**", Wiley, 2nd Edition, **2014**
Lawrence S. Brown and Thomas Holme , "**Chemistry for engineering students**", Brooks Cole, 4th Edition, **2018**

Indicative Experiments

1. Thermodynamic functions from EMF measurements: Zn – Cu system. Evaluation of properties enthalpy, entropy and free energy	2 hours
2. Determination of reaction rate, order and molecularity - ester hydrolysis (ethylacetate)	2 hours
3. Colorimetric estimation of Ni^{2+} using conventional and smartphone digital-imaging method	2 hours
4. Analysis of iron in carbon steel by potentiometry	2 hours
5. Preparation of ZnO semiconductor and its characterization	2 hours
6. Estimation of sulfate ion in drinking water by the conductivity method	2 hours

7. Build atoms and molecules, visualization of atomic orbitals and hybridized orbitals - calculating the orbital contributions	2 hours
8. Conformational analysis of cyclohexane and ethane molecules and plotting the potential energy profile	2 hours
9. Application of AI to screen semiconductors, materials, drugs - case study	2 hours
10. Size-dependent colour variation of Cu_2O nanoparticles by spectrophotometer	2 hours
11. Laboratory synthesis of hydroxyl apatite and its use for the removal of heavy metals	2 hours
12. Challenging experiment related to evaluation of the state of charge of batteries (SOC)	2 hours
13. Construction of DSSC and its working	2 hours
14. Preparation of Ag metal nanoparticles and their characterization	2 hours
15. Colorimetric estimation of Fe^{2+} using conventional and smartphone digital-imaging method	2 hours
Total Laboratory Hours:	**30 hours**

Text Book(s)

Arthur Vogel and John Mendham, "**Vogel's Textbook of Quantitative Chemical Analysis**", Pearson Education, 6th Edition, **2009**

Mode of Evaluation : Continuous Assessment Test, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment

Recommended by Board of Studies :	23-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE101	Problem Solving Using Python	0	0	4	2
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To impart algorithmic skills through python programming language fundamentals and constructs.
2. To explore python collections for efficient data handling techniques and to learn modular, recursive programming approaches for developing software applications.
3. To understand python libraries for data analysis and manipulations

Course Outcomes

1. Identify appropriate algorithmic approach, data representation, control constructs in developing software solutions for solving real-world problems.
2. Inculcate data handling techniques using python collections and modular programming strategies for developing multi-disciplinary software applications
3. Idealize the importance of python libraries for handling, manipulating and analyzing multi-faceted real world problem domains.

Indicative Experiments

1. **Subsidy Computation** Agricultural subsidies are offered by the Indian government to farmers for improving crop yields and benefiting them. To facilitate the farmers in finding the subsidy amount, develop a Python code to calculate the subsidy based on the crop the farmer grows. The farmers can grow one of these three types of crops: Rice, Wheat, or Maize. The program calculates and displays the type of crop, total subsidy based on the type of crop selected, along with the yield and market rate. Use menu driven approach to implement the same. Each crop has a different formula for calculating the subsidy: Rice: Subsidy= (Yield × Market Rate) +100 Wheat: Subsidy= (Yield × Market Rate) +200 Maize: Subsidy= (Yield × Market Rate) +300 Input Format Press 1/2/3 to compute the Subsidy of Rice/Wheat/Maize Next Line to enter the Market Rate Next Line to enter the Yield Enter either Yes/No .	4 hours

Press 1/2/3 to compute the Subsidy of Rice/Wheat/Maize Next Line to enter the Market Rate Next Line to enter the Yield Enter either Yes/No Output Format Display “Press 1/2/3 to compute the Subsidy of Rice/Wheat/Maize” Display the Type of Crop, Subsidy, Yield and Market Rate Do you want to continue (Yes/No). If Yes . Press 1/2/3 to compute the Subsidy of Rice/Wheat/Maize Display the Type of Crop, Subsidy, Yield and MarketRate Do you want to continue (Yes/No). If No, display ‘Thank you’ **Climate Change Data Analyzer** Scientists track daily temperatures as a string of values separated by spaces (e.g., "30.5 32.1 29.8"). Write a python program that extracts the temperatures, converts them into float values, and finds the highest, lowest, and average temperature. Rules & Constraints: • Input is a string of space-separated float values (e.g., "30.5 31.2"). • All values must be valid floating-point numbers. • The list must have at least two values. • Temperatures must be within a realistic range: -100.0°C to 60.0°C. • The program should round the average to two decimal places. • If the input is invalid (non-numeric, empty, out-of-range), return an error message. Input Format: A single string with space-separated temperature readings (e.g., "25.4 27.8 26.5") Output Format: Highest: Lowest: Average:	
2. **Gaming Leaderboard Username Validator** In an online game, players choose usernames for the leaderboard. To make sure all usernames are clean and fair, they must follow these rules: • The username must start with a letter (A–Z or a–z). • It can contain letters, numbers, and underscores (_) only. • It cannot start with a number.	4 hours

• No other symbols or spaces are allowed. • The length must be between 5 and 15 characters. Write a python program using regular expressions to check if a username is valid or invalid. If invalid, display a message “Invalid Name” Input Format: A single string – the username to validate. Output Format: Print "valid username" if it follows all rules. Otherwise, print "Invalid Name". **Number Game** In Numeria, Alchemists must build an arithmetic expression using given numbers and operations to match a target. Use each number once, with +, -, *, or /. If no exact match, return the expression closest to the target. If multiple expressions give the same best result, return "No single solution" Rules: • Use all numbers once. • Only +, -, *, / allowed. • Division must result in integers only. • Left-to-right evaluation only (no brackets) • All numbers must be positive integers. • Return one valid expression, or "No single solution" if ties exist. Input Format: ((num1, num2, ..., numN), target) Output Format: • A string like "2+3*5", or • "No single solution" • Error message for invalid input	
3. **Drone Delivery** A delivery drone needs to find the shortest path from a warehouse (starting point) to a delivery location (destination) in a grid-based city map. The drone can move up, down, left, or right, but it cannot pass through restricted zones (obstacles). Implement an algorithm (only) using a top-down approach to find the shortest path from the warehouse to the delivery location. Input Format: A 2D grid (N × N) where 0 represents open paths and 1 represents obstacles. The starting position (warehouse) and destination position (delivery location). Top- down Design Approach:	6 hours

If the destination is reached, return the path length. If the current cell is an obstacle or out of bounds, return an invalid path value. Try moving in all four directions (up, down, left, right). Use memoization to store results of already visited paths. Output Format: The minimum number of steps required to reach the destination. The shortest path length or an indication that the delivery is not possible. **Task Completion Time** A group of students is participating in a coding competition, where they need to solve a set of problems as quickly as possible. Each student has a recorded completion time (in minutes) for each problem. The goal is to identify the student who solved their problems in the shortest total time. Implement an algorithm using Divide and Conquer problem solving approach to determine the student with the lowest total completion time. **Input:** A list of students with their total completion times. **Base Case:** If there is only one student, return them as the fastest. **Divide**: Split the student list into two halves. **Conquer**: Recursively find the fastest student in both halves. **Combine**: Compare the two selected students and return the one with the lower total time.	
4. **Rental Computation** Maya is developing a pricing system for a photography studio that rents out different types of cameras and needs to calculate the rental cost based on the equipment type and duration. The system should apply discounts based on the rental cost thresholds: Video Camera: The rental rate is ₹250 per hour. If the total rental cost exceeds ₹1500, a 12% discount is applied. Drone Camera: The rental rate is ₹300 per hour. If the total rental cost exceeds ₹2000, a 15% discount is applied. Write a Python code to compute the rentalCost for VideoCamera, and DroneCamera and apply the respective discount based on the rental duration and the type of camera. Input Format First line contains the duration of usage of Video Camera Second line contains the rental rate of Video camera per hour Third line contains the duration of usage of Drone Camera Fourth line contains the rental rate of Drone camera per hour. Output Format First line prints the total rental cost of Video Camera Second line prints the discount for Video Camera usage.	4 hours

Third line prints the total rental cost with discount for Video Camera. Fourth line prints the total rental cost of Drone Camera Fifth line prints the discount for Drone Camera usage. Sixth line Prints the total rental cost with discount for Drone Camera. **Leader Selection** In the city of Solaria, a group of n scholars sit in a circle. Each scholar has a number from 1 to n. The selection for the next Guardian of Harmony works like this: • Scholar 1 tells Scholar 2 to leave the circle. • Then the next scholar (still in the circle) tells the person next to them to leave. • This continues in a circle — each person removes the one next to them — until only one scholar is left. Determine which scholar will be the last one standing Input Format: A single number n — the number of scholars. Output Format: A number — the scholar who is left at the end.	
5. **Analyze health risk** Analyze health risks based on multiple individuals' BMI (Body Mass Index), blood pressure, and heart rate. Using a loop to collect data for multiple individuals. Calculate BMI using BMI = weight / (height ** 2), use conditional statements to classify BMI as low, medium or high. Low BMI (underweight): Less than 18.5 kg/m² Medium BMI (normal weight): 18.5 - 24.9 kg/m² High BMI (overweight): 25 kg/m² and above A "low" blood pressure is generally considered to be below 90/60 mmHg, while a "medium" or normal blood pressure is between 90/60 mmHg and 120/80 mmHg, otherwise high. For heart rate, a "low" resting rate is under 60 beats per minute (bpm), and a "medium" or normal resting rate is between 60-100 bpm, otherwise high. Additionally, check if blood pressure (systolic/diastolic) and heart rate fall within healthy ranges, categorizing individuals into Low, Moderate, or High health risk levels. Write a python code to display the BMI, blood pressure and heart rate. If two or more health risks are high, then the risk level is high. If any one of the health risk is high, then the risk level is medium. Otherwise, it is under low risk. Finally, display a summary report showing the number of individuals in each risk category. Input Format: First line contains the number of People ‘n’	6 hours

Next line contains the Blood pressure(Systolic/Diastolic) of Person1 in mm/hg
Next line contain heartrate of Person1 in beats per minute(bpm)
.
Next line contains the weight of the Person ‘n’ in Kg
Next line contains the height of the Person ‘n’ in meter
Next line contains the Blood pressure(Systolic/Diastolic) of Person ‘n’ in mm/hg
Next line contain heartrate of Person ‘n’ in beats per minute(bpm)

Output Format
First line Prints the BMI of Person 1(kg/m2)
Second line prints underweight/normalweight/overweight of Person 1
Third line Prints the Blood Pressure of Person 1(mm/hg)
Fourth Line prints Low/Medium/HighBloodPressure of Person 1
Fifth line prints the Heart rate of Person 1(bpm)
Sixth line prints low/normal/high heartrate of Person 1
Seventh line prints low/medium/high risk level of Person 1
.
Next line Prints the BMI of Person ‘n’(kg/m2)
Next line prints underweight/normalweight/overweight of Person ‘n’
Next line Prints the Blood Pressure of Person ‘n’(mm/hg)
Next Line prints Low/Medium/HighBloodPressure of Person ‘n’

Next line prints the Heart rate of Person ‘n’(bpm)
Next line prints low/normal/high heartrate of Person ‘n’
Next line prints low/medium/high risk level of Person ‘n’

Next prints the Summary:
High risk level : number of People
Medium risk level: number of People
Low risk level: number of People

Process DNA sequence

DNA sequences are primarily analyzed by scientists, researchers, and medical professionals in fields like genetics, molecular biology, and medicine, to understand genetic variations, diagnose diseases. Write a Python program to process and analyze the DNA sequences by performing various operations based on nucleotide sequence rules. The program should perform the following on valid DNA Sequences:

- Accept 'n' DNA sequences from the user.
- Verify whether each sequence contains only the valid DNA bases (A, C, G, T). If so, display as Valid otherwise invalid.
- Search for the specific "TATAAA" sequence within the given DNA sequences. If so Display as match/nomatch
- Count and display the occurrences of base 'A' in each DNA sequence.

• Store the unique bases and their frequency in a dictionary for all sequences combined and display them. • Replace a specific substring in the DNA sequences with 'X' (e.g. replacing "AT" in "TATA" results in "TXA"). • Convert DNA to RNA by replacing 'T' with 'U', and store the RNA sequences in a list. • Search for a specific pattern in the DNA sequences and display its position if found. • Compute and display the complementary strand for each DNA sequence, where A pairs with T, and C pairs with G (e.g., "ATCGA" → "TAGCT"). Input Format First line contains number of DNA sequences Next line contains the DNA sequence 1 Next line contains the DNA sequence 2 . Next line contains the DNA sequence n Next line contains the specific sequence Next line contains the base to find its occurrence: Next line contains the substring to replace: Next line contains the string to replace with: Next line contains the string to replace to convert DNA to RNA Next line contains the string to replace with to convert DNA to RNA Next line contains the pattern to display its position Output Format First Line displays DNA sequence 1 valid/invalid Next Line displays DNA sequence 2 valid/invalid . Next Line displays DNA sequence n valid/invalid Next Line displays DNA sequence 1 Match/No Match Next Line displays DNA sequence 2 Match/No Match . Next Line displays DNA sequence n : Match/No Match Next Line displays DNA sequence 1 count Next Line displays DNA sequence 2 count . Next Line displays DNA sequence n count Frequency of Unique bases: {'A': count, 'T': count, 'G': count, 'C': count} Next Line displays DNA sequence 1: New DNA sequence 1 after replacing Next Line displays DNA sequence 2: New DNA sequence 2 after replacing . Next Line displays DNA sequence n: New DNA sequence n after replacing DNA to RNA:['New Sequenc1', 'New Sequenc2',… 'New Sequencn'] Next Line displays the position of the given Pattern in DNA sequence 1:	

Next Line displays the position of the given Pattern in DNA sequence 2: . Next Line displays the position of the given Pattern in DNA sequence n: Display the position of the given Pattern in DNA sequence 1: Display the position of the given Pattern in DNA sequence 2: Display the position of the given Pattern in DNA sequence n: Display the complementary strand of DNA sequence 1: Display the complementary strand of DNA sequence 2: . Display the complementary strand of DNA sequence n:	
6. **Renewable Energy Grid** In the year 2050, the world is powered by solar energy storage units, distributed across three major smart power stations. These stations are: (a) Station 1: Primary source, (b) Station 2: Intermediate (buffer) , (c) Station 3: Final destination. To ensure sustainable and stable energy flow, scientists have developed a precise transfer protocol that moves energy units from Station 1 to Station 3. Transfer Rules (Constraints): • Only one energy unit can be transferred at a time to prevent system overload. • A larger unit can never be placed on top of a smaller unit. • The number of energy units n must be an integer between 1 and 20. • The transfer output must be a sequence of moves: each move is a pair (a, b) meaning "move a unit from Station a to Station b", where a and b are in {1, 2, 3}. • Two consecutive moves must not involve the same pair of stations in opposite directions. For example, (1, 3), (3, 1) back-to-back is not allowed, as it causes unnecessary energy loss and instability. Input Format: An integer n represents the number of energy storage units to be transferred from the primary station to the final station. Output Format: A list of ordered pairs (a,b), where each pair represents a transfer of an energy unit from station a to station b. **Rainfall prediction using Sorting** Write a Python program to assist meteorologists in forecasting rainfall by analyzing relative humidity based on air temperature (23°C to 28°C) and dew point temperature (7°C to 16°C). Use the given formulas to compute specific humidity, saturation point, and relative humidity: specific humidity = 6.11 x 10^{7.5 x dew point} / {237.3 + dew point} saturation point = 6.11 x 10^{7.5 x air temp} / {237.3 + air temp}	4 hours

relative humidity = (specific humidity) / (saturation point) x 100 The chance of rainfall is high when the relative humidity is high and the temperature is dropping. The program should sort the relative humidity values in increasing order and display the highest relative humidity along with its corresponding air temperature and dew point temperature, providing valuable insights for rainfall prediction. Input Format: An integer 'N' representing the number of temperature readings. 'N' lines, each containing: Air Temperature (integer, between 23°C and 28°C) Dew Point Temperature (integer, between 7°C and 16°C) Output Format: Sorted list of relative humidity values in increasing order along with their corresponding air temperature and dew point temperature. Highest relative humidity value with its associated air temperature and dew point temperature. Prediction statement indicating whether the chance of rainfall is High/Low (if humidity is high and temperature is dropping).	
7. **Browser Tracking** Write a Python program to simulate a web browser's back and forward navigation system using stacks, where visiting a new webpage pushes the URL onto the back stack and clears the forward stack. Implement functions to navigate back by popping the top page from the back stack and pushing it onto the forward stack, and to navigate forward by popping the top page from the forward stack and pushing it onto the back stack. The program should allow users to enter URLs, navigate back and forward, and display the current page along with available navigation options dynamically. Input Format: •An integer N representing the number of webpage visits. •N lines, each containing a URL (string) representing a webpage visit. •User can enter the following commands dynamically after initial webpage visits: o BACK: Navigate to the previous page. o FORWARD: Navigate to the next page. o VISIT : Visit a new webpage, clearing the forward history. o EXIT: Terminate the program. Output Format: •Display the current page after each operation. •Show available navigation options (Back and Forward stacks). •Display messages when back or forward navigation is unavailable.	6 hours

•Display Browser session ended when Exit. **Customer Orders Queue Processing** In a smart manufacturing unit, automated machines process customer orders in a First-In-First-Out (FIFO) manner to ensure efficiency. Write a Python program using a queue to simulate an order processing system, where customers place orders for different industrial products (e.g., raw materials, electronic components, machinery parts), and each order is stored along with its order placed time. The system should process the orders in the same sequence they were received, record the order completion time, and remove them from the queue upon completion. Finally, display the status of the order queue before and after processing, along with the time taken to complete each order. Input Format: An integer 'N' representing the number of orders N lines, each containing: Order ID (string) Product Name (string) Order Placed Time (HH:MM 24hr -format) Order Completion Time ((HH:MM format) Output Format: Order queue before processing Processing details, including: Order ID - Product Name - Order Placed Time - Order Completion Time - Time Taken to Process (in minutes) Order queue after processing	
8. **Smart Building** Elena Vasquez, a futuristic architect, is designing modular smart buildings on a blueprint. Each building is represented as a rectangle with: •(x, y) — coordinates of the bottom-left corner •Width and height — horizontal and vertical sizes of the building To avoid overlap during construction, Elena needs a tool that calculates the exact overlapping area between any two buildings. Write a python function (collections framework) that takes two building rectangles as input and returns the area of overlap. If they do not overlap, return 0. Input Format: Each rectangle is given as a list of 4 integers: [x, y, width, height]	6 hours

Output Format:
A single integer — the overlapping area between the two buildings. If no overlap, return 0.
Constraints:
•All coordinates and dimensions are integers.
•Width and height must be positive (greater than 0).
•Input lists must contain exactly 4 integers.
•Coordinates may be negative (to allow buildings in all quadrants).

Analyze Water Quality
Environmental agencies, researchers assess the overall water quality, monitor potential pollution sources, and ensure the water is safe for intended uses like drinking, agriculture, and aquatic life using water pH and dissolved oxygen levels and turbidity. Assist them by developing a Python program using functions and conditional statements to analyse the water quality based on pH, turbidity, and dissolved oxygen levels. The program should take input values for these parameters, check if they are within safe limits, and classify the water as safe or unsafe for consumption.
If pH value is 6.5 and 8.5, Turbidity lesser than 3 NTU(Nephelometric Turbidity Unit) and Dissolved Oxygen between 6.5 and 8 mg/L then the water quality is safe Use conditional statements to give recommendations if the water is safe or unsafe. Allow users to input data for any location and display a summary of safe and unsafe water sources. Locations may include Clean River area, lake near agriculture area, urban storm drain area, Deep Ground water Source etc
Use menu driven approach and functions to compute and display for multiple locations

Input Format
First Line contains the name of the Location 1
Second Line contains the pH value
Third Line contains the Turbidity
Fourth Line contains Dissolved Oxygen
Enter either Yes/No
.
Next Line contains the name of the Location ‘n’
Next Line contains the pH value
Next Line contains the Turbidity
Next Line contains Dissolved Oxygen
Enter either Yes/No

Output Format
Safe/unsafe
Do you want to continue (Yes/No). If Yes
.
Safe/unsafe
Do you want to continue (Yes/No). If No, display ‘Thank you’

9. **Reforestation** In a reforestation project, each tree grows a fixed number of saplings every year. These saplings become mature trees starting from the next year and also begin producing saplings. Implement a python program using functions that calculates the total number of mature trees after a given number of years. Input Format: A tuple of three integers: initial_trees – Number of trees at year 0 saplings_per_tree – Number of saplings each tree produces per year years – Number of years to simulate Output Format: A single integer: total number of mature trees after the given number of years. **Recurring Character** For a given string which is made of uppercase letters (A–Z). Implement a recursive function that finds the first character that appears more than once. The function should return a dictionary with: • The character (key) • A list of two indices: • The index where it first appeared • The index where it reappeared first • If no recurring character is found, or if the input is None or an empty string, return an empty dictionary {}. Rules & Constraints: • Input must be a string containing only uppercase A–Z. • Return the first recurring character based on position. • Ignore later repetitions after the first recurrence. • Use recursion (no loops allowed). • Return an empty dictionary if: • Input is None • Input is an empty string • No character repeats Input Format: A single string: "ABCDAB" Output Format: A dictionary like: {"A": [0, 4]} Or: {} (if no recurring character or invalid input)	6 hours

Tower of Hanoi In robotic warehouse automation, autonomous robotic arms move stacked containers or pallets between storage locations while following strict stacking rules. Similar to the Tower of Hanoi problem, the system must transfer stacked items from a source rack to a destination rack using an intermediate rack, ensuring that larger items are never placed on smaller ones. Write a Python program using a recursive function to simulate this process, where a set number of stacked containers (disks) must be moved while following the First-In-Last-Out (FILO) principle. The program should determine the optimal sequence of moves, ensuring safe and efficient stacking, and display each step of the movement process to track how robotic arms relocate items. Input Format: An integer 'N' representing the number of stacked containers. Output Format: A sequence of steps showing how the containers are moved from the Source Rack to the Destination Rack using the Intermediate Rack.	
10. **Student Attendance Data Analysis using Numpy** Involve as a team to collect attendance dataset from professors. Design a Python program using NumPy to analyse a student attendance dataset. Compute the total attendance per student, identify the student with the highest attendance, calculate the class average attendance per day, and determine the day with the lowest attendance rate. Finally, compute each student's attendance percentage and display students who have less than 75% attendance. Input Format: •An integer N representing the number of students. •An integer D representing the number of days. •A N x D matrix, where each row corresponds to a student, and each column represents a day. o1 indicates present o0 indicates absent Output Format: •Display the original attendance matrix. •Compute and display the total attendance per student. •Identify and display the student with the highest attendance. •Calculate and display the class average attendance per day. •Determine and display the day with the lowest attendance rate. •Compute and display each student's attendance percentage. •Identify and display students who have less than 75% attendance.	4 hours

11. **Student Marks Data Analysis using Pandas** Involve as a team to collect sample archived student marks dataset from professors. Write a Python program using Pandas to analyse students' internal marks, where each student is evaluated based on six internal components and a final component. Create a DataFrame with student names, six internal scores, and final score. Calculate the class average internal score, compute the total marks for each student by summing all components, and determine the student with the highest total marks. Additionally, filter and display students who have scored below 40% indicating the need for improvement. Input Format: An integer 'N' representing the number of students. N lines, each containing: Student Name (String) Six Internal Component Scores (Integers from 0 to 100) Final Component Score (Integer from 0 to 100) Output Format: •Display the original DataFrame with student names, internal scores, and final scores. •Compute and display the class average internal score. •Calculate and display the total marks for each student. •Identify and display the student with the highest total marks. •Filter and display students scoring below 40% of the total (i.e., below 40 out of 100).	4 hours
12. **Restaurant Food Waste Cost** Restaurants throw away a lot of food every day, which affects both costs and the environment. Consider the dataset which contains details of food waste over a month. Write a python program using Pandas to analyze and reduce this waste. Input Format: A CSV file with the following columns: •Date (YYYY-MM-DD) – The day waste was recorded •Food Item – Name of the food item wasted •Category – Type of food (e.g., Vegetables, Dairy, Meat) •Quantity Wasted – Amount wasted (in kg or units) •Cost per Item – Cost of one unit of the food item Perform the following: 1. Clean the Data o Handle missing values in Date, Food Item, Category, Quantity Wasted, or Cost per Item (e.g., fill or remove).	6 hours

2. Calculate Waste Cost o Per Day: Total waste cost for each day o Per Category: Waste cost for each food category 3. Top 5 Most Wasted Items o Find the 5 food items with the highest total quantity wasted o Suggest simple strategies to reduce waste for each (e.g., store better, reuse, order less) 4. Trend & Strategy Analysis o Spot patterns (e.g., which days waste the most) o Recommend ways to reduce waste (e.g., smaller portions, better planning, donation) Output Format: •A cleaned DataFrame (after handling missing data) •Two summary tables: o Total waste cost per day o Total waste cost per category •A ranked list of top 5 wasted food items, with suggestions •Trend insights and recommendations for reducing waste. **Smart Energy Grid** A smart energy grid tracks hourly electricity usage (in kWh) for a residential building over 30 days. The data is stored in a NumPy array. Write a Python program that analyzes the energy usage using NumPy. Input Format: A NumPy array with shape (30, 24): 30 rows = 30 days 24 columns = 24 hours per day Each value shows the electricity used during that hour. Perform the following: 1. Calculate Energy Usage • Daily Consumption: Total energy used per day (output: array of 30 values) • Monthly Consumption: Total energy used in 30 days (output: one float) 2. Find Peak Hours For each day, find hours where usage is above that day's average Output: A dictionary {day_index: [list_of_peak_hours]} 3. Predict Next Day's Usage Use the last 7 days to predict the next day's total energy usage using simple average of daily totals from the last 7 days Output Format: Daily Consumption: NumPy array (30 values) Monthly Consumption: Float Peak Hours: Dictionary of peak hour indices per day Predicted Next-Day Usage: Float (average of last 7 days)	

Total Laboratory Hours:	**60 hours**
Text Book(s)	
John V. Guttag, "**Introduction to computation and programming using Python** ", The MIT Press, 3rd Edition, **2021** Eric Matthes , "**Python Crash Course.** ", No Starch Press, 3rd Edition, **2022**	
Reference Books	
Luciano Ramalho, "**Fluent Python** ", OReilly Media Inc , 2nd Edition, **2022** Charles R. Severance , "**Python for Everybody** ", Shroff Publishers, 1st Edition, **2024**	
Mode of Evaluation : Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE102	Problem Solving using Java	0	0	4	2
Pre-requisite	Nil	**Syllabus Version**			
		1			

Course Objectives

To introduce the fundamental concepts of object-oriented programming using the Java language, including data types, control structures, and basic programming constructs
To enable students to design and implement Java programs utilizing classes, objects, inheritance, polymorphism, and interfaces to build modular and reusable code
To equip students with a comprehensive understanding of Java's exception handling and multithreading capabilities. Java's File Object, input/output (I/O) streams and to equip students with the skills to leverage generic programming and the Java Collection Framework with Databases

Course Outcomes

Develop basic Java programs utilizing fundamental programming constructs, object-oriented principles such as classes, objects, inheritance, and polymorphism
Implement robust and efficient Java applications by applying exception handling techniques, managing concurrent execution using threads, and performing file input/output operations.
Utilize the Java Collection Framework, including Lists and Maps, along with generic programming concepts to effectively manage and manipulate data within Java applications.

Indicative Experiments

1. a. Calculating Interest for Financing Options, Bobby is a financial analyst working for a construction company. He is tasked with evaluating two financing options for funding a new construction project. Each option involves different principal amounts and annual interest rates. Write a program for him that takes the principal amounts and interest rates for both financing options as input and calculates the respective interests for the financial options. Note: Multiply the principal amount by the annual interest rate and then divide by 100 to find the total interest paid b. Three-Digit Number Analysis: Sum of Digits and Parity Check. Alex is teaching a class on number properties and wants to create an exercise for his students. He needs a program that takes a three-digit number as	2 hours

input, calculates the sum of its digits, and determines whether the sum is even or odd. Write a program to assist Alex in this task. c. Generating Prime Numbers Within a Specified Range. Arun is tasked with creating a program that prints prime numbers within a given range. The program should take two integers, start and end, as input, and output the prime numbers between these two values (inclusive). Help Arun to complete the task using a 'for' loop.	
2. a.Calculating Total Distance from Merged Daily Step Counts You are given the number of steps a person takes on two consecutive days. Your task is to merge the step counts from both days and calculate the total distance the person covers, assuming that each step covers a fixed distance of 1 unit. Write a program that takes the step counts for the two days as input, merges them, calculates the total distance covered and outputs the result. b. Identifying and Locating Prime Numbers in a 2D Grid Yashna is creating a program to analyse a 2D grid, identifying prime numbers and printing their coordinates. The input consists of the grid's dimensions and elements. The program's output displays the coordinates of each prime number found in the grid, aiding in the exploration of numerical patterns. Can you assist Yashna in creating the program?	2 hours
3. a. Designing a Rectangle Class to Calculate Area and Perimeter. Arun wants to design a program using a class that calculates and prints both the area and perimeter of a rectangle. Define a class named Rectangle to encapsulate the properties and behaviour related to a rectangle. Include two integer variables length and breadth to represent the dimensions of the rectangle. Calculate and print the area and perimeter of the rectangle. Help Arun to design the program b.Developing an ArrayConcatenator Class for Merging Arrays.Help Roshni to complete the program. Input Format The first line of input consists of an integer N, representing the number of elements of the first Roshni is tasked with developing a program for concatenating two arrays provided by the user. To accomplish this, she wants to create a class named ArrayConcatenator. The class includes a constructor to concatenate the elements of the input arrays. She wants to print the elements of the resulting array. The second line consists of N space-separated integers representing the first array elements. The third line consists of an integer M, representing the number of elements of the second array. The fourth line consists of M space-separated integers, representing the second array elements. Output Format:	2 hours

The output prints the concatenated array of elements separated by space. Refer to the sample output for the formatting specifications. Constraints: In this scenario, the test cases fall under the following constraints 1 ≤ N, M ≤ 10 1 ≤ array elements ≤ 100	
4. a. Calculating Employee Bonuses Using Hierarchical Inheritance ☐ Shasha, an HR manager in a multinational corporation, requires a program to compute bonuses for developers and designers using hierarchical inheritance. Three classes: Employee with a base salary attribute; Developer and Designer extending Employee, each with a bonus percentage, work hours, and methods to calculate and display the final income. Shasha inputs the base salary and work hours for a developer and a designer, and the program outputs their respective final incomes. ☐ Note: For developers, the bonus percentage is 10%, for designers, it is 5% ☐ Formula: Final income = Base salary + (base salary * bonus percentage * work hours / 100) ☐ b.Designing a Fantasy Game Character System.Jessica is tasked with designing a fantasy game character system. The system includes an abstract class named GameCharacter with two abstract methods: attack() and defend().Two subclasses, Warrior and Wizard, extend the GameCharacter class.The program prompts the player to choose a character class (1. Warrior, 2. Wizard) and input their character's strength or magic power. The dynamic calculations involve tripling the strength (strength * 3) for the Warrior's attack and doubling the magic power(power * 2) for the Wizard's attack. Jessica needs your help in completing the program. Help her finish it.	2 hours
5. a. Designing a Washing Machine Control System. Alex and Bob are designing a control system for household appliances, and one of the appliances is a washing machine. You want to create a program to help them that models the washing machine as a motor and calculates its electricity consumption based on its capacity. Define an interface named Motor with the following methods: void run() double consume(double capacity) Create a class called WashingMachine that implements the Motor interface. In the WashingMachine class: Implement the run() method to print "Washing machine is running."	2 hours

Implement a consume() method to print "Washing machine is consuming electricity." Implement the consume(double capacity) method to calculate the electricity consumption (in kWh) of the washing machine based on its capacity. The formula for electricity consumption is (capacity * 0.05). b. Unique Character Extraction.Ashok needs to implement a program using the java.lang package to process a given string. The goal is to create a program that extracts and displays a new string containing only the unique characters from the provided input string. Ashok should efficiently use the java.lang.StringBuilder class and display the output. Help him to complete the program.	
6. a. Validating Student Grade Input with Multi-Catch Exception Handling. Sampad wants to implement a program that takes input for a student's name and grade, validates the input, and then displays the grade for the given student. The grade should be an integer value. The program should validate the grade using the validateGrade() method. The method should throw an IllegalArgumentException if the grade is less than 0 or greater than 100. If the input is invalid due to a non-integer grade, catch the NumberFormatException and print the built-in exception message. If the input is invalid due to an out-of-range grade, catch the IllegalArgumentException and print the built-in exception message. Catch the exceptions using the multi-catch block. Assist Sampad to implement this program. b. Implementing Custom Exceptions for Time Input Validation.Alice is tasked with creating a program that validates user input for time values (hours, minutes, and seconds). The program should check whether the input values are within the valid range. If any value falls outside the valid range, a custom exception, InvalidHourException, InvalidMinuteException, or InvalidSecondException, should be thrown, indicating which part of the time (hours, minutes, or seconds) is invalid. If no exceptions occur, print "Correct Time - " followed by the time in the format "hours:minutes:seconds."	2 hours
7. Synchronizing Producer and Consumer Threads with a Shared Message Buffer.A shared resource, a message buffer, needs to be managed by two threads: a Producer and a Consumer. The Producer thread will generate messages and place them into the buffer. The Consumer thread will retrieve and process messages from the buffer. To ensure proper synchronization and avoid issues like the Consumer trying to read from	2 hours

an empty buffer or the Producer overwriting a message before it's consumed, you need to implement inter-thread communication using wait(), notify(), and synchronized blocks. Create a MessageBuffer class that holds a single message. The Producer thread will put a message into this buffer, and the Consumer thread will take a message from it. Implement the necessary synchronization mechanisms so that: The Consumer waits if the buffer is empty.The Producer waits if the buffer is full (i.e., already contains a message that hasn't been consumed).The Producer notifies the Consumer when a new message is available.The Consumer notifies the Producer after consuming a message, indicating the buffer is now empty.Write a Producer class that generates a sequence of messages (e.g., "Message 1", "Message 2", etc.) and puts them into the MessageBuffer.Write a Consumer class that retrieves and prints the messages from the MessageBuffer.Create a Main class to instantiate the MessageBuffer, a Producer thread, and a Consumer thread, and then start both threads.	
8. Calculating Final Prices with Tax: File Input and Output.Ella is managing her expenses and wants to calculate the final prices after applying a 10% tax to her purchases. She has a list of item prices that she wants to process. Create a file named prices.txt and write the entered item prices to this file. Then, the program should read from prices.txt, apply a 10% tax to each item, write the taxed amounts to a file named tax.txt, and display the taxed amounts. Input Format: The first line of input consists of an integer N, representing the number of items Ella wants to process.The second line consists of N space-separated double values, representing the price of an item.Output Format:The output displays a double value, representing the taxed amounts, each with two decimal places, separated by a space.Refer to the sample output for formatting specifications.Constraints: this scenario, the test cases fall under the following constraints:1 ≤ N ≤ 10.	2 hours
9. Basic Sentiment Analysis: Classifying Sentences from File Input. ☐Mr. Styles is working on a sentiment analysis program to understand the sentiments conveyed in various sentences. he needs your assistance in developing a program that analyzes the sentiment of a given sentence and classifies it as positive, negative, or neutral. ☐positive keywords = happy, good, excellent, positive. ☐ negative keywords = sad, bad, terrible, negative. ☐Anything else is Neutral. ☐Create a file named input.txt, the input is written into the file input.txt. The program then reads the sentence from input.txt and sentiment analysis is performed based on predefined positive and negative keywords. The classified sentiment (positive, negative, or neutral) is written to a new file named output.txt and displayed.	2 hours

10. Converting Speed Units (km/h to m/s) with File Input and Output: Nick is developing a program to convert speed units from kilometers per hour (km/h) to meters per second (m/s). However, he needs assistance in creating a structured and user-friendly program. Create a file named data.txt to enter the speed in km/h as given in the input. The program reads the speed from data.txt and then converts the speed from km/h to m/s using the formula. The converted speed is written to a new file named converted.txt and then displayed.	2 hours
11. Implementing Persistent Bank Account Transactions using Serialization. Sam is managing his bank account and wants to perform various transactions using a program. He wants to deposit and withdraw funds and then save the account details using serialization. Upon restarting the program, he wants to be able to load the serialized data and continue with his banking transactions. Write a program to assist Sam in managing his bank account, create a BankAccount class, and Serialize the BankAccount object to a file named bankAccount.ser after the transactions. Deserialize the BankAccount object from the file when the program restarts. Display the final balance after the transactions in a formatted manner with two decimal places.	2 hours
12. Serializing and Deserializing Savings Data to Determine Savings Category. Alice is managing her expenses and wants to create a simple program to track her savings she wants to categorise her savings as "Poor savings," "Good savings," or "High savings" using a serialization mechanism. Design a program to create an object of the SavingsData class with the provided salary and savings data. Serialize the created object to a file named savings.ser. Deserialize the SavingsData object from the savings.ser file. Display the category determined by the program after deserialization. Note: "Poor savings", if the savings percentage is between 1% (inclusive) and 10%. "Good savings", if the savings percentage is between 10% (inclusive) and 20%. "High savings", if the savings percentage is greater than or equal to 20%. "Invalid input", if none of the above conditions are met.	2 hours
13.	2 hours

Implementing a Generic Record Holder for Academic Data. A university needs a system to manage different types of academic records, such as student grades and course enrollments. To create a flexible and type-safe system, they want to use generic classes.Design a generic class named RecordHolder that can hold any type of academic record. This class should have the following functionalities: A private instance variable to store the record. A constructor that takes an object of the specific record type and initializes the instance variable. A method named getRecord() that returns the stored record. A method named displayRecordInfo() that prints some relevant information about the record. The implementation of this method will depend on the specific type of record being held. Create two concrete classes to represent different types of academic records: GradeRecord: This class should have private instance variables for studentName (String) and grade (Double). It should have a constructor to initialize these variables and a displayRecordInfo() method that prints: "Grade for [studentName]: [grade]". EnrollmentRecord: This class should have private instance variables for courseName (String) and studentId (Integer). It should have a constructor to initialize these variables and a displayRecordInfo() method that prints: "Student [studentId] is enrolled in [courseName]". In your Main class, demonstrate the use of the RecordHolder generic class with both GradeRecord and EnrollmentRecord objects. Create instances of RecordHolder to hold a GradeRecord and an EnrollmentRecord, and then call the getRecord() and displayRecordInfo() methods on these RecordHolder instances.	
14. Student Registration System using ArrayList in Java. Raman is a computer science teacher who is responsible for registering students for his programming class. He wants to streamline the registration process using a simple program. The program should allow him to input the names of students and later retrieve a student's name based on the index entered. Raman has decided to use an ArrayList to store the names.	2 hours

15. Calculating Median Temperature from City Data using HashMap. As a data analyst for a weather forecasting agency, you have a HashMap storing temperature data for various cities. Each city's entry consists of temperature readings. Your task is to find and report the median temperature, providing a more representative measure of the overall temperature patterns over a specific period. Note: To calculate the median, sort the numbers in ascending order. If the count is odd, the median is the middle number. If the count is even, the median is the average of the two middle numbers.	2 hours
16. You are tasked with developing a simple Employee Management System using Java and JDBC. The system should allow users to perform the following operations on employee records stored in a MySQL database Develop a menu-driven program (EmployeeApp.java) that allows users to: 1. Add Employee 2. View Employees 3. Update Employee Salary 4. Delete Employee 5. Exit	2 hours
Total Laboratory Hours:	**60 hours**

Text Book(s)

Marc Loy, Patrick Niemeyer and Daniel Leuck, , "**Learning Java, "**, O Reilly Media, 5th Edition, **2020**
Herbert Schildt and Dr. Danny Coward, " **Java: The Complete Reference**", McGraw Hill, 13th Edition, **2024**

Reference Books

Kathy Sierra, Bert Bates and Trisha Gee,, " **Head First Java**", O Reilly Media, 3rd Edition, **2022**
Dhruti Shah, "**100+ Solutions in Java: A Hands-On Introduction to Programming in Java, "**, BPB Publications, 1st Edition, **2020**

Mode of Evaluation : Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE191	Basic Multidisciplinary Project	0	0	4	2
Pre-requisite	Nil	**Syllabus Version**			
		1			

Course Objectives

Enable students to collaborate effectively in diverse groups by integrating multidisciplinary engineering knowledge to develop practical solutions.
Enhance students' skills in project planning, execution, documentation, and technical communication through hands-on engineering activities.
Encourage ethical and innovative practices to design sustainable solutions for society and environmental challenges.

Course Outcomes

Apply multidisciplinary engineering concepts and techniques to design a sustainable working model.
Utilize appropriate technological tools, effective time management to design and document the prototype as an individual or team member.
Demonstrate the societal impact, and cost-effectiveness of the developed product with ethics.

Module:1	**Project Duration First and Second Semesters**	**20 hours**

Projects can be an experimental setup/model to realize the conceptual prototype, and validate its performance.
Each project team should have 3 to 5 students from at least three different disciplines.
The team must submit both the prototype and a technical report that explains each member's individual contribution.

Total Lecture Hours:	**20 hours**

Text Book(s)

Reference Books

YouTube , "**https://www.youtube.com/watch?v=y4drM0vz3us** ", University of Sydney, 1st Edition, **2020**
YouTube, "**https://uwaterloo.ca/engineering-ideas-clinic/about/engineering-design-days/first-year-design-days** ", Waterloo Engineering, 1st Edition, **2017**
YouTube, "**https://www.youtube.com/watch?v=et6E7hWyS0g**", University of UTAH, 1st Edition, **2015**

YouTube, " **https://www.youtube.com/watch?v=hbNTI1NGQ40**", Johns Hopkins University, 1st Edition, **2017** YouTube, "**https://www.youtube.com/watch?v=lnxl84LBRd0** ", University of Colorado Boulder, 1st Edition, **2017** YouTube, "**https://www.youtube.com/watch?v=uljTK226rsU** ", Olin College of Engineering, 2nd Edition, **2021** YouTube, " **https://www.youtube.com/watch?v=rHAzpXiO1bg**", Olin College of Engineering , 1st Edition, **2019**	
Mode of Evaluation :	
Recommended by Board of Studies :	23-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE291	Innovative Design Project	0	0	4	2
Pre-requisite		Syllabus Version			
		1.0			

Course Objectives

1. To apply engineering design principles to real-world problems.
2. To develop engineering models and prototypes using appropriate tools.
3. To conduct testing, validation, and optimization of designs.
4. To prepare technical reports and deliver professional presentations.

Course Outcomes

1. Apply engineering design methodology to solve practical problems, integrating creativity with technical feasibility.
2. Develop skills in modeling, simulation, prototyping, and testing, while evaluating solutions for performance, safety, cost-effectiveness, and sustainability.
3. Develop strong teamwork and project management skills
4. Communicate engineering solutions effectively through reports and presentations.

Module	One Year Project	120 hours
Engineering design process, encompassing problem identification, research, modeling, prototyping, and testing. It concludes with the development of optimized prototypes, comprehensive technical documentation, and professional presentations, enabling graduates to integrate creativity with technical practicality and effectively communicate engineering solutions. Each project team should be minimum of 2 to 3 students.		
	Total hours:	120 hours

Text Books

1	Jack, Hugh. Engineering design, planning, and management. Academic press, 2021.

Reference Books

1.	Ulrich, Karl, and Steven Eppinger. *EBOOK: Product Design and Development*. McGraw Hill, 2011.

Mode of Evaluation: Reviews and Report Writing

Recommended by Board of Studies			
Approved by Academic Council	**No.**	**Date**	

Course code	Course title	L	T	P	C
BAEEE101	**Basic Engineering**	3	0	2	4
Pre-requisite	NIL	Syllabus version			
		1.0			

Course Objectives

- To introduce fundamental principles of major engineering disciplines.
- To create awareness of interdisciplinary engineering systems and their applications

Course Outcomes

At the end of the course students will be able to

1. Analyse the electrical circuits, electrical motors and the role of power electronics in industrial applications.
2. Acquire foundational knowledge of electronic devices and communication systems.
3. Apply BIS standards to create basic 2D and 3D representations of engineering components.
4. Explain the fundamental principles and applications of manufacturing processes, energy conversion systems, and mechanical automation technologies.
5. Analyse real-world examples where bio-inspired solutions have led to breakthroughs in engineering.

Module:1	Principles of Electrical Engineering: Circuits and Power conversion equipment	12 hours

Electric circuit components, Mesh current analysis, Node voltage analysis, Thevenin's and Superposition theorems, Single phase AC circuits- RL, RC, RLC, Power and Energy Calculations, Power Factor, Basics of Electrical Safety and Earthing.

Introduction to electro mechanical energy conversion, principle of operation of Electrical Machines - DC Motor, Induction motors, BLDC Motor and single phase Transformer, Concepts of Power Electronics and Industrial Applications of Electrical Drives (Qualitative Analysis)

Module:2	Foundations of Electronics and Communication systems: Devices, Circuits, and Systems	8 hours

Characteristics of PN Junction Diode, Zener Diode, BJT and MOSFET. Rectifiers and Voltage Regulators. Introduction to Operational Amplifiers. Electromagnetic Specturm. Elements of Communication Systems. Overview of cellular communication. Fundamentals of Satellite Communication and Radar.

Module:3	Engineering Graphics and CAD	9 hours

Introduction to Engineering Drawing – Importance and scope of engineering graphics, BIS standards for lines, lettering and dimensioning. Orthographic Projections – Principles of projection, first angle and third angle projections. Introduction to Projection of Solids- Isometric Projections and perspective projections. Freehand Sketching – Sketching of simple machine components and objects. Drawing of a simple residential floor plan as per IS SP7, Introduction to CAD.

Module:4	Manufacturing Processs, Energy conversions and Mechanical Automation	9 hours

Basic Metal Casting, Forming, Joining and Cutting Processes - Introduction to CNC Machining. Additive Manufacturing (3D Printing): Principles and Applications.

Fundamentals of Thermal engineering systems, Internal Combustion Engine. Introduction to Power Plants. Introduction to Refrigeration and Air Conditioning. Basics of Fuel Cells and Sustainable Energy Technologies. Introduction to Mechanisms - Four-bar mechanism and its inversions- Hexapod mechanisms. Mechanical Components and Motion Control in mechanical systems. Introduction to Robotics and Automation – Pneumatic and hydraulic actuators.		
Module:5	**Bio-Inspired Design in Engineering**	**5 hours**
Introduction to Biomimicry- Core principles. Case Studies Across Disciplines: Velcro inspired by burrs, self-healing materials based on skin regeneration -Termite mound-inspired ventilation systems in green buildings-Insect-inspired drones and soft robotics-Neural networks based on the human brain -Identify a problem and brainstorm a bio-inspired solution.		
Module: 6	**Contemporary Topics**	**2 hours**
Lectures by Industry or Research experts		
	Total Lecture hours:	**45 hours**

Text Book(s)	
1.	Allan R. Hambley, "Electrical Engineering -Principles & Applications", 2019, 6th Edition, Pearson Education
2	R.L. Boylestad and L. Nashelsky, "Electronic Devices and circuit theory", 11th Edition, person Publications
3	Louis E Frenzel Jr, "Principles of Electronics Communication Systems", 2016, McGraw Hill Eduation.
4	Bhatt N. D., Engineering Drawing, Charotar Publishing House Pvt. Ltd, 2019.
Reference Books	
1.	John bird, "Electrical Circuit Theory and Technology", 4th edition, Newnes Publications, Elsevier, 2010
2.	George Kennedy,Brendan Davis, Srm Prasanna "Electronic Communication Systems", McGraw Hills, 2011
3.	Serope Kalpakjian, and Steven Schmid, Manufacturing Engineering and Technology, 2020, 8th edition, Pearson education.
4.	Nag P. K., Basic & Applied Thermodynamics, 2009, 2nd Edition, McGraw Hill Education.
5.	Rattan S. S, Theory of Machines, Tata McGraw Hill, 2019
6.	Craig, John. J. (2008), Introduction to Robotics: Mechanics and Control, Second Edition, Pearson Education, New Delhi.

List of Lab Experiments (Electrical and Electronics Engineering)

1. Verification of KCL and KVL in an Electrical Circuit
2. Verification of Mesh and Nodal Analysis in an Electrical Circuit
3. Time domain analysis of a single phase series RL Circuit
4. Measurement of Power and power factor in a single phase RL Circuit using 3- volt meter method
5. Speed Control of a PMDC Motor with DC-DC Buck Converter
6. Measurement of Earth Resistance using an Meggar

7. Analysis of a Zener Diode Voltage Regulator 8. Single-Phase Full-Wave Diode Rectifier with Resistive Load 9. Design and Testing of a Light Dimmer Circuit Using a Darlington Pair. 10. Create a 2D profile using basic sketch tools and apply appropriate geometric and dimensional constraints. 11. Develop a simple 3D solid model using features such as extrude, revolve, cut, fillet, chamfer, and shell. 12. Generate detailed 2D engineering drawings of simple parts from 3D solid models, including multiple views, dimensions, and annotations.			
Mode of Evaluation: CAT, Assignment, Quiz, Seminar, Project, FAT			
Recommended by Board of Studies	23-05-2025		
Approved by Academic Council	No. 78	Date	12-06-2025

Course Code	Course Title	L	T	P	C
BAENG101	Technical English Communication	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

To enable learners to decipher the tenets of general and interpersonal communication
To facilitate application of grammar skills in academic and general situations
To help foster learning of productive skills for communicative competence

Course Outcomes

Enhance writing skills for better publication
Identify the basic concepts related to various aspects of communication
Apply grammatical rules in academic and real-life contexts
Demonstrate analytical reading skills in terms of deciphering research articles
Illustrate qualitative writing skills by drafting emails and academic documents

Module:1	**Introduction to Technical Communication**	**6 hours**

The Nature of Technical Communication

Taxonomy of Technical Writing-Scientific Writing, Technical Reporting

Business Communication

5Cs of Technical Writing
Activity: Applying these concepts to answer conceptual questions

Module:2	**Grammar**	**9 hours**

Concord

Tense

Conditionals

Voice

Transitions and Linking words
Activity: Error Detection

Module:3	**Reading**	**10 hours**

Analysing definitions – Types of definitions, Methods for Expanding Definitions

Reading Comprehension – Drawing Inferences from Scientific Texts

Structure of a Research Article Analysing a Research Article **Activity:** Reading a research article		
Module:4	**Writing**	**12 hours**
Research Writing: Structure of a Review Paper Types of Reviews Literature Review Referencing Style - IEEE, APA **Activity:** Writing a literature review		
Module:5	**Interpersonal Communication**	**6 hours**
Cross-cultural Communication Interpersonal Communication Leadership Styles Belbin's Team Roles Negotiation Skills Thomas-Kilmann Modes of Conflict Management Statement of Purpose **Activity:** Case study Analysis		
Module:6	**Contemporary Topics**	**2 hours**
Activity: Guest lectures from Industry and, Research and Development Organizations		
	Total Lecture Hours:	**45 hours**
Text Book(s)		
Lannon, J.M. & Gurak, L.J., "**Technical communication**", Pearson, 7th Edition, **2025** Shoba, K.N. & Sam, P., "**Technical English: A workbook**", Cambridge University Press, **2018**		
Reference Books		
Alred, G.J., Brusaw, C.T., & Oliu, W.E., "**Handbook of technical writing**", Bedford St Martins, 11th Edition, **2015**		

Gross, A., Hamlin, A., Merck, B., Rubio, C., Naas, J., Savage, M., & Desilva, M. , "**Technical writing.** ", Open Oregon Educational Resources, **2017** Laplante, T.A. , "**Technical writing: A practical guide for engineers, scientists, and nontechnical professionals.**", CRC Press, 2nd Edition, **2019** Markel, M. & Selber, S.A. , "**Technical communication.**", Bedford St Martins , 12th Edition, **2018**	
Indicative Experiments	
1. **Vocabulary and Proof Reading** Verbal Traps in Technical Communication: Cliches- Malapropisms- Erroneous Heterographs- The Laziness of "very" Proofreading Symbols Proofreading a Short Passage **Activity**: Interactive Worksheets, Proof Reading	4 hours
2. **Intensive Listening** Listening to Podcasts Viewing Documentaries **Activity**: Listening and Note-taking/ Summarising	8 hours
3. **Speaking** Role Play Storytelling Group Discussion Video Resume Presentations **Activity**: Individual and group presentations	8 hours
4. **Reading** Elements of Fiction-Plot-Character-Setting-Theme-Point of View Appreciation of a text-The Ant and the Grasshopper by Somerset Maugham **Activity**: Story/ Book Review	6 hours
5. **Writing** Blog Writing Transcoding	4 hours

Statement of Purpose Creating a Portfolio **Activity**: Blog/ Portfolio Design	
Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	16-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BAENG101	Technical English Communication	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

To enable learners to decipher the tenets of general and interpersonal communication
To facilitate application of grammar skills in academic and general situations
To help foster learning of productive skills for communicative competence

Course Outcomes

Enhance writing skills for better publication
Identify the basic concepts related to various aspects of communication
Apply grammatical rules in academic and real-life contexts
Demonstrate analytical reading skills in terms of deciphering research articles
Illustrate qualitative writing skills by drafting emails and academic documents

Module:1	**Introduction to Technical Communication**	**6 hours**

The Nature of Technical Communication

Taxonomy of Technical Writing-Scientific Writing, Technical Reporting

Business Communication

5Cs of Technical Writing
Activity: Applying these concepts to answer conceptual questions

Module:2	**Grammar**	**9 hours**

Concord

Tense

Conditionals

Voice

Transitions and Linking words
Activity: Error Detection

Module:3	**Reading**	**10 hours**

Analysing definitions – Types of definitions, Methods for Expanding Definitions

Reading Comprehension – Drawing Inferences from Scientific Texts

Structure of a Research Article Analysing a Research Article **Activity:** Reading a research article		
Module:4	**Writing**	**12 hours**
Research Writing: Structure of a Review Paper Types of Reviews Literature Review Referencing Style - IEEE, APA **Activity:** Writing a literature review		
Module:5	**Interpersonal Communication**	**6 hours**
Cross-cultural Communication Interpersonal Communication Leadership Styles Belbin's Team Roles Negotiation Skills Thomas-Kilmann Modes of Conflict Management Statement of Purpose **Activity:** Case study Analysis		
Module:6	**Contemporary Topics**	**2 hours**
Activity: Guest lectures from Industry and, Research and Development Organizations		
	Total Lecture Hours:	**45 hours**
Text Book(s)		
Lannon, J.M. & Gurak, L.J., "**Technical communication**", Pearson, 7th Edition, **2025** Shoba, K.N. & Sam, P., "**Technical English: A workbook**", Cambridge University Press, **2018**		
Reference Books		
Alred, G.J., Brusaw, C.T., & Oliu, W.E., "**Handbook of technical writing**", Bedford St Martins, 11th Edition, **2015**		

Gross, A., Hamlin, A., Merck, B., Rubio, C., Naas, J., Savage, M., & Desilva, M. , "**Technical writing.** ", Open Oregon Educational Resources, **2017** Laplante, T.A. , "**Technical writing: A practical guide for engineers, scientists, and nontechnical professionals.**", CRC Press, 2nd Edition, **2019** Markel, M. & Selber, S.A. , "**Technical communication.**", Bedford St Martins , 12th Edition, **2018**	
Indicative Experiments	
1. **Vocabulary and Proof Reading** Verbal Traps in Technical Communication: Cliches- Malapropisms- Erroneous Heterographs- The Laziness of "very" Proofreading Symbols Proofreading a Short Passage **Activity**: Interactive Worksheets, Proof Reading	4 hours
2. **Intensive Listening** Listening to Podcasts Viewing Documentaries **Activity**: Listening and Note-taking/ Summarising	8 hours
3. **Speaking** Role Play Storytelling Group Discussion Video Resume Presentations **Activity**: Individual and group presentations	8 hours
4. **Reading** Elements of Fiction-Plot-Character-Setting-Theme-Point of View Appreciation of a text-The Ant and the Grasshopper by Somerset Maugham **Activity**: Story/ Book Review	6 hours
5. **Writing** Blog Writing Transcoding	4 hours

Statement of Purpose Creating a Portfolio **Activity**: Blog/ Portfolio Design	
Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	16-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BAHUM101	**India Studies**	**1**	**0**	**0**	**1**
Pre-requisite	**Nil**	**Syllabus version**			
		1.0			

Course Objectives

1. **To introduce students to the foundations of India's traditional knowledge systems**, cultural heritage, and civilizational values.
2. **To develop an understanding of India's scientific, technological, and philosophical traditions**, and their relevance to contemporary policy frameworks like NEP 2020.
3. **To provide a conceptual and structural understanding of the Indian Constitution**, its principles, governance mechanisms, and contemporary significance.

Course Outcome

On completion of this course the students will be able to:

1. **Explain** the core concepts of India's traditional knowledge systems, culture, civilization, and their historical significance.
2. **Analyse** India's scientific, technological, and philosophical traditions and their influence on the development of the Indian Constitution and governance structures.
3. **Evaluate** the contemporary relevance of Indian knowledge systems and constitutional provisions in addressing modern societal challenges.

Module:1	**Traditional Knowledge, Culture & Civilization**	**3 hours**

Traditional Knowledge: definition, nature, characteristics, scope and importance - Kinds of traditional knowledge: Vedas, Upanishads, Śāstras - Traditional knowledge vs. Indigenous knowledge vs. Western knowledge - Introduction to culture and civilization -Features of Indian culture.

Module:2	Science, Technology & Knowledge Traditions in India	**3 hours**

Scientific and technological developments in Ancient, Medieval, and Modern India - Importance of biodiversity in India - *Arthashastra*, *Sukraniti*, *Kamandaka Niti* - Contemporary need for incorporating Indian knowledge systems (IKS) - Indian Knowledge System in NEP 2020.

Module:3	**Introduction to the Indian Constitution**	**3 hours**

Meaning, nature, and significance of the Constitution - Historical background: Colonial period, freedom struggle & Constituent Assembly - Salient features of the Indian Constitution - Preamble: Philosophy, ideals, and values - Sources of the Indian Constitution.

Module:4	**Rights, Duties & Governance Structure**	**3 hours**

Fundamental Rights - Directive Principles of State Policy - Fundamental Duties - Union Government: President, Prime Minister, Parliament - State Government: Governor, Chief Minister, State Legislature - Federalism.

Module:5	Constitutional Processes	**2 hours**

Amendments to the Constitution - Judicial System - Constitutional bodies - Social justice provisions: Reservations, welfare policies.

Module:6	**Contemporary relevance**	**1 hours**
Total Lecture Hours		**15 hours**

Text Book(s)

1.	Shikha Jain, Parul G. Manjul & Somya Joshi (2020). *Traditional Knowledge Systems and Cultural Heritage*. Aryan Books International.			
2.	Durga Das Basu (2018). *Introduction to the Constitution of India* (23rd ed.). LexisNexis.			
Reference Books				
1.	Bidyut Chakraborty & Rajendra Kumar Pandey (2008). *Indian Government and Politics*. SAGE Publications.			
2.	Amit Jha (2024). *Traditional Knowledge System in India*. Atlantic Publishers & Distributors.			
3.	Granville Austin (1966). *The Indian Constitution: Cornerstone of a Nation*. Oxford University Press.			
Mode of Evaluation: Quiz and FAT.				
Recommended by Board of Studies		03-11-2025		
Approved by Academic Council		No. 80	Date	20-11-2025

Course Code	Multivariable Calculus and Differential Equations	L	T	P	C
BAMAT101		3	0	2	4
Pre-requisite		Syllabus Version			
		1.0			

Course Objectives

1. To provide essential foundational knowledge in engineering mathematics, supporting engineers and scientists in mastering advanced mathematical concepts relevant to their field
2. To enhance problem-solving skills for engineers and scientists by mastering vector calculus principles, including gradient, divergence, curl, and fundamental theorems, for real-world engineering applications.
3. To develop expertise in solving differential equations through various techniques, enabling engineers and scientists to address complex challenges in system modeling and simulations.

Course Outcomes

At the end of the course, the student is able to:

1. evaluate partial derivatives and solve optimization problems involving several variables with or without constraints.
2. solve multiple integrals in cartesian, polar, cylindrical and spherical coordinates.
3. understand gradient, directional derivatives, divergence, curl, Green's, Stokes and Gauss Divergence theorems.
4. utilize different solution methods to obtain the solution for second-order ordinary differential equations.
5. demonstrate proficiency in solving linear and nonlinear first-order partial differential equations.

Module:1	Functions of Several Variables	8 hours

Functions of two variables- Partial derivatives –total differential -Jacobian and its properties , Taylor's expansion for two variables–maxima and minima–constrained maxima and minima Lagrange's multiplier method.

Applications: *Jacobian matrix for feature sensitivity analysis. Constrained optimization in* Cloud computing.

Module:2	Multiple Integrals	9 hours

Evaluation of double integrals–change of order of integration–change of variables between Cartesian and polar co-ordinates, - Evaluation of triple integrals - change of variables between Cartesian and cylindrical and spherical co-ordinates,

Applications: *Change of variables for object recognition. Multiple integrals in Image analysis for density and moments.*

Module:3	Vector Differentiation and Integration	10 hours

Scalar and vector valued functions – gradient, directional derivative, divergence, and curl – scalar and vector potentials. Line, surface and volume integrals - Statement of Green's, Stoke's and Gauss divergence theorems - Verification and evaluation of vector integrals using them,

Applications: *Vector derivatives for Electromagnetic simulation in Computer Aided Design. Vector integrals in Antenna design.*

Module:4	Ordinary Differential Equations	8 hours

Linear second-order ordinary differential equations with constant coefficients – Solutions of ODE by Method of undetermined coefficients – Method of variation of parameters – Solutions of Cauchy-Euler and Cauchy-Legendre differential equations – Applications of linear second-order differential equations.

Applications: *Steady-state response of input signal in Digital Signal Processing, Differential equations for LCR Circuit Simulation.*		
Module:5	**Partial Differential Equations**	**8 hours**
Formation of PDE- solution of first order partial differential equation of the forms: F(p,q)=0, F(z,p,q)=0, F(x,p)=G(y,q) and Clairaut's form - Lagrange's equation: Pp+Qq = R, Solution of a linear first order partial differential equation by separation of variables, Applications: *Lagranges equation for flow modelling, Separation of variables for Analyzing CPU temperature.*		
Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture hours:	**45 hours**

Text Books	
1	Erwin Kreyszig, Advanced Engineering Mathematics, 10th Edition, Wiley India, 2015.
2	B.S. Grewal, Higher Engineering Mathematics, 44th Edition, Khanna Publishers, 2020
Reference Books	
1.	George B. Thomas, D. Weir and J. Hass, Thomas' Calculus, 15th edition, Pearson, 2023.
2.	Dennis G. Zill, Advanced Engineering Mathematics, 6th Edition, Jones & Bartlett Publishers, 2018.
Mode of Evaluation: Quizzes, Assignments, CATs and FAT	
List of Experiments (Indicative)	
1.	Introduction to Calculus through MATLAB – Symbolic Computations
2.	Plotting and visualizing curves and surfaces
3.	Evaluating maxima and minima of functions of several variables
4.	Applying Lagrange multiplier optimization method
5.	Evaluation of volume under a surface by double Integral
6.	Evaluating triple integrals
7	Demonstration of gradient, divergence and curl
8	Evaluating line integrals in vectors
9	Solution of Linear differential equations
10	Solution of Partial differential equations
Total Lab hours:	**30 hours**
Text Book	
1	Ronald L. Lipsman , Jonathan M. Rosenberg, Multivariable Calculus with MATLAB with applications to Geometry and Physics, Springe, 2017.
Reference Book	
1	Dean G. Duffy, Advanced Engineering Mathematics with MATLAB, CRC Press, 2022.
Mode of Evaluation: Assignments and FAT	

Recommended by Board of Studies	14-May-2025		
Approved by Academic Council	**No.**	**Date**	

Course Code	Multivariable Calculus and Differential Equations	L	T	P	C
BAMAT101		3	0	2	4
Pre-requisite		Syllabus Version			
		1.0			

Course Objectives

1. To provide essential foundational knowledge in engineering mathematics, supporting engineers and scientists in mastering advanced mathematical concepts relevant to their field
2. To enhance problem-solving skills for engineers and scientists by mastering vector calculus principles, including gradient, divergence, curl, and fundamental theorems, for real-world engineering applications.
3. To develop expertise in solving differential equations through various techniques, enabling engineers and scientists to address complex challenges in system modeling and simulations.

Course Outcomes

At the end of the course, the student is able to:

1. evaluate partial derivatives and solve optimization problems involving several variables with or without constraints.
2. solve multiple integrals in cartesian, polar, cylindrical and spherical coordinates.
3. understand gradient, directional derivatives, divergence, curl, Green's, Stokes and Gauss Divergence theorems.
4. utilize different solution methods to obtain the solution for second-order ordinary differential equations.
5. demonstrate proficiency in solving linear and nonlinear first-order partial differential equations.

Module:1	Functions of Several Variables	8 hours

Functions of two variables- Partial derivatives –total differential -Jacobian and its properties , Taylor's expansion for two variables–maxima and minima–constrained maxima and minima Lagrange's multiplier method.

Applications: *Jacobian matrix for feature sensitivity analysis. Constrained optimization in* Cloud computing.

Module:2	Multiple Integrals	9 hours

Evaluation of double integrals–change of order of integration–change of variables between Cartesian and polar co-ordinates, - Evaluation of triple integrals - change of variables between Cartesian and cylindrical and spherical co-ordinates,

Applications: *Change of variables for object recognition. Multiple integrals in Image analysis for density and moments.*

Module:3	Vector Differentiation and Integration	10 hours

Scalar and vector valued functions – gradient, directional derivative, divergence, and curl – scalar and vector potentials. Line, surface and volume integrals - Statement of Green's, Stoke's and Gauss divergence theorems - Verification and evaluation of vector integrals using them,

Applications: *Vector derivatives for Electromagnetic simulation in Computer Aided Design. Vector integrals in Antenna design.*

Module:4	Ordinary Differential Equations	8 hours

Linear second-order ordinary differential equations with constant coefficients – Solutions of ODE by Method of undetermined coefficients – Method of variation of parameters – Solutions of Cauchy-Euler and Cauchy-Legendre differential equations – Applications of linear second-order differential equations.

Applications: *Steady-state response of input signal in Digital Signal Processing, Differential equations for LCR Circuit Simulation.*		
Module:5	**Partial Differential Equations**	**8 hours**
Formation of PDE- solution of first order partial differential equation of the forms: F(p,q)=0, F(z,p,q)=0, F(x,p)=G(y,q) and Clairaut's form - Lagrange's equation: Pp+Qq = R, Solution of a linear first order partial differential equation by separation of variables, Applications: *Lagranges equation for flow modelling, Separation of variables for Analyzing CPU temperature.*		
Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture hours:	**45 hours**

Text Books	
1	Erwin Kreyszig, Advanced Engineering Mathematics, 10th Edition, Wiley India, 2015.
2	B.S. Grewal, Higher Engineering Mathematics, 44th Edition, Khanna Publishers, 2020
Reference Books	
1.	George B. Thomas, D. Weir and J. Hass, Thomas' Calculus, 15th edition, Pearson, 2023.
2.	Dennis G. Zill, Advanced Engineering Mathematics, 6th Edition, Jones & Bartlett Publishers, 2018.
Mode of Evaluation: Quizzes, Assignments, CATs and FAT	
List of Experiments (Indicative)	
1.	Introduction to Calculus through MATLAB – Symbolic Computations
2.	Plotting and visualizing curves and surfaces
3.	Evaluating maxima and minima of functions of several variables
4.	Applying Lagrange multiplier optimization method
5.	Evaluation of volume under a surface by double Integral
6.	Evaluating triple integrals
7	Demonstration of gradient, divergence and curl
8	Evaluating line integrals in vectors
9	Solution of Linear differential equations
10	Solution of Partial differential equations
Total Lab hours:	**30 hours**
Text Book	
1	Ronald L. Lipsman , Jonathan M. Rosenberg, Multivariable Calculus with MATLAB with applications to Geometry and Physics, Springe, 2017.
Reference Book	
1	Dean G. Duffy, Advanced Engineering Mathematics with MATLAB, CRC Press, 2022.
Mode of Evaluation: Assignments and FAT	

Recommended by Board of Studies	14-May-2025		
Approved by Academic Council	**No.**	**Date**	

Course Code	Course Title	L	T	P	C
BAMAT207	Probability and Statistics	3	0	2	4
Pre-requisite	BAMAT101	Syllabus version			
		1			

Course Objectives

1. Apply probability and statistical methods to solve engineering problems, including system reliability, risk assessment, and quality control.
2. To develop predictive models and analyze data using probability distributions, correlation, regression, and hypothesis testing for engineering applications.
3. To design experiments and interpret variability using ANOVA and structured experimental designs to optimize engineering systems and innovations.

Course Outcomes

At the end of the course, the students will be able to:

1. Apply probability concepts, including axioms, conditional probability, and Bayes' theorem, to solve real-world problems.
2. Implement probability distributions to model random variables and analyze statistical properties.
3. Utilize hypothesis testing methods to evaluate statistical significance in experiments.
4. Demonstrate correlation and regression techniques for data analysis and predictive modeling.
5. Analyze ANOVA methods and experimental designs to interpret variability in datasets.

Module: 1	Probability and Random Variables	11 hours

Basic Probability- Axioms, probability spaces, conditional probability, Bayes' theorem. Random Variables and Distributions - Discrete and Continuous random variables, probability mass functions, probability density functions. Joint Distributions - Joint, marginal, and conditional distributions. Expectation and Variance – Moments, variance, covariance, statistical independence. Functions of one-dimensional random variables. Spam filtering, password strength estimation, disease prediction, and network packet loss analysis.

Module: 2	Probability Distributions	9 hours

Bernoulli, Binomial, Poisson, Geometric, Negative Binomial, Uniform, Normal, Exponential, Gamma, and Weibull distributions. Central limit theorem, Sampling Distributions – t, F, Chi-square (Concept Only). Modelling software bug occurrences, estimating download failure rates, and analysing system response times.

Module: 3	Testing of Hypothesis	8 hours

Formulating and testing hypotheses, Simple and composite hypotheses, Type I and Type II errors. Parametric Tests – Z-test, t-test and paired sample t-test for single and two population means and F-test for difference of two population variances, Chi-square test for Goodness of fit and independence of attributes. Confidence Intervals - Constructing and interpreting confidence intervals. Testing software performance, comparing algorithm accuracies and validating sensor data quality.

Module: 4	Correlation and Regression	9 hours

Pearson's correlation and Spearman's rank correlation coefficient. Multiple and partial correlation. Simple and Multiple linear regression: model, assumptions, estimation and interpretation of regression coefficients. Model diagnostics and validation. Predicting software defect counts, estimating CPU load from system parameters, and forecasting user traffic in web applications.

Module: 5	Analysis of Variance (ANOVA)	6 hours

One-way and Two-way Analysis of Variance – CRD, RBD and LSD. Comparing classification model accuracies across datasets, evaluating the effect of hyperparameter settings on model performance and analysing query response times across different database indexing strategies.		
Module: 6	**Contemporary topics**	**2 hours**
	Total Lecture hours	**45 hours**
Text Book(s)		
1	Introduction to Probability and Statistics for Engineers and Scientists, Sheldon M. Ross, 6th Edition, Academic Press Inc., ISBN-978-0128243466, 2021.	
2	Richard Arnold Johnson, Irwin Miller, John E. Freund- Probability and Statistics for Engineers, Pearson Education Ltd., 9th Edition, 2018.	
Reference Book(s)		
1	An Introduction to Probability and Statistics, V. K. Rohatgi and A. K. Md. Ehsanes Saleh, John Wiley & Sons, 3rd Edition, 2015.	
2	Applied Statistics and Probability for Engineers" by Douglas C. Montgomery and George C. Runger, 7th Edition, John Wiley & Sons, 2018.	
3.	Michael Baron, Probability and Statistics for Computer Scientists, 3rd Edition, CRC Press / Chapman & Hall, 2019.	
Mode of evaluation: CAT / Quiz / Case study/Study-oriented Project / Final Assessment Test.		
Embedded Lab – Indicative Experiments		
1	Introduction, understanding data types, importing and exporting data sets.	**2 hours**
2	Exploratory Data Analysis - Tabulation and Data Visualization - Create histograms, box plots, and scatter plots to visualize the data distributions and relationships.	**2 hours**
3	Exploratory Data Analysis - Simulation and Descriptive Statistics - Summarize data using central tendency, variability, skewness and kurtosis.	**2 hours**
4	Fitting of Probability Distributions – Discrete and Continuous	**2 hours**
5	Testing of Hypothesis and Confidence Intervals – Z and t-test of hypothesis	**2 hours**
6	Testing of Hypothesis and Confidence Intervals – F and Chi-square test of hypothesis	**2 hours**
7	Simple linear Regression and Correlation Analysis – Modelling, Estimating and Interpreting correlation and regression coefficients.	**4 hours**
8	Multiple Linear Regression and Correlation Analysis – Modelling, Estimating and interpreting correlation and regression coefficients.	**4 hours**
9	Performing One-way ANOVA – CRD - Modelling, Estimation and interpretation of Coefficients.	**4 hours**
10	Performing Two-way ANOVA –RBD & LSD- Modelling, Estimation and interpretation of Coefficients.RBD and LSD.	**4 hours**
	Total Lecture hours	**30 hours**
Text Book(s)		
1	Probability and Statistics with R, Maria Dolores Ugarte, Ana F. Militino, and Alan T. Arnholt, 2nd Edition, Chapman and Hall/CRC, 2016.	
Reference Book(s)		
1	The Book of R: A First Course in Programming and Statistics, Tilman M. Davies, No Starch Press, 1st Edition, 2016.	
2	Applied Statistics: Theory and Problem Solutions with R, Dieter Rasch, Rob Verdooren,	

	and Jürgen Pilz, Wiley, 1st Edition, 2019.		
Mode of evaluation: Lab Continuous Assessment and Lab Final Assessment Test.			
Recommended by Board of Studies	14.05.2025		
Approved by Academic Council	No. 78	Date	12.06.2025

Course Code	Course Title	L	T	P	C
BAMAT207	Probability and Statistics	3	0	2	4
Pre-requisite	BAMAT101	Syllabus version			
		1			

Course Objectives

1. Apply probability and statistical methods to solve engineering problems, including system reliability, risk assessment, and quality control.
2. To develop predictive models and analyze data using probability distributions, correlation, regression, and hypothesis testing for engineering applications.
3. To design experiments and interpret variability using ANOVA and structured experimental designs to optimize engineering systems and innovations.

Course Outcomes

At the end of the course, the students will be able to:

1. Apply probability concepts, including axioms, conditional probability, and Bayes' theorem, to solve real-world problems.
2. Implement probability distributions to model random variables and analyze statistical properties.
3. Utilize hypothesis testing methods to evaluate statistical significance in experiments.
4. Demonstrate correlation and regression techniques for data analysis and predictive modeling.
5. Analyze ANOVA methods and experimental designs to interpret variability in datasets.

Module: 1	Probability and Random Variables	11 hours

Basic Probability- Axioms, probability spaces, conditional probability, Bayes' theorem. Random Variables and Distributions - Discrete and Continuous random variables, probability mass functions, probability density functions. Joint Distributions - Joint, marginal, and conditional distributions. Expectation and Variance – Moments, variance, covariance, statistical independence. Functions of one-dimensional random variables. Spam filtering, password strength estimation, disease prediction, and network packet loss analysis.

Module: 2	Probability Distributions	9 hours

Bernoulli, Binomial, Poisson, Geometric, Negative Binomial, Uniform, Normal, Exponential, Gamma, and Weibull distributions. Central limit theorem, Sampling Distributions – t, F, Chi-square (Concept Only). Modelling software bug occurrences, estimating download failure rates, and analysing system response times.

Module: 3	Testing of Hypothesis	8 hours

Formulating and testing hypotheses, Simple and composite hypotheses, Type I and Type II errors. Parametric Tests – Z-test, t-test and paired sample t-test for single and two population means and F-test for difference of two population variances, Chi-square test for Goodness of fit and independence of attributes. Confidence Intervals - Constructing and interpreting confidence intervals. Testing software performance, comparing algorithm accuracies and validating sensor data quality.

Module: 4	Correlation and Regression	9 hours

Pearson's correlation and Spearman's rank correlation coefficient. Multiple and partial correlation. Simple and Multiple linear regression: model, assumptions, estimation and interpretation of regression coefficients. Model diagnostics and validation. Predicting software defect counts, estimating CPU load from system parameters, and forecasting user traffic in web applications.

Module: 5	Analysis of Variance (ANOVA)	6 hours

One-way and Two-way Analysis of Variance – CRD, RBD and LSD. Comparing classification model accuracies across datasets, evaluating the effect of hyperparameter settings on model performance and analysing query response times across different database indexing strategies.		
Module: 6	**Contemporary topics**	**2 hours**
	Total Lecture hours	**45 hours**
Text Book(s)		
1	Introduction to Probability and Statistics for Engineers and Scientists, Sheldon M. Ross, 6th Edition, Academic Press Inc., ISBN-978-0128243466, 2021.	
2	Richard Arnold Johnson, Irwin Miller, John E. Freund- Probability and Statistics for Engineers, Pearson Education Ltd., 9th Edition, 2018.	
Reference Book(s)		
1	An Introduction to Probability and Statistics, V. K. Rohatgi and A. K. Md. Ehsanes Saleh, John Wiley & Sons, 3rd Edition, 2015.	
2	Applied Statistics and Probability for Engineers" by Douglas C. Montgomery and George C. Runger, 7th Edition, John Wiley & Sons, 2018.	
3.	Michael Baron, Probability and Statistics for Computer Scientists, 3rd Edition, CRC Press / Chapman & Hall, 2019.	
Mode of evaluation: CAT / Quiz / Case study/Study-oriented Project / Final Assessment Test.		
Embedded Lab – Indicative Experiments		
1	Introduction, understanding data types, importing and exporting data sets.	**2 hours**
2	Exploratory Data Analysis - Tabulation and Data Visualization - Create histograms, box plots, and scatter plots to visualize the data distributions and relationships.	**2 hours**
3	Exploratory Data Analysis - Simulation and Descriptive Statistics - Summarize data using central tendency, variability, skewness and kurtosis.	**2 hours**
4	Fitting of Probability Distributions – Discrete and Continuous	**2 hours**
5	Testing of Hypothesis and Confidence Intervals – Z and t-test of hypothesis	**2 hours**
6	Testing of Hypothesis and Confidence Intervals – F and Chi-square test of hypothesis	**2 hours**
7	Simple linear Regression and Correlation Analysis – Modelling, Estimating and Interpreting correlation and regression coefficients.	**4 hours**
8	Multiple Linear Regression and Correlation Analysis – Modelling, Estimating and interpreting correlation and regression coefficients.	**4 hours**
9	Performing One-way ANOVA – CRD - Modelling, Estimation and interpretation of Coefficients.	**4 hours**
10	Performing Two-way ANOVA –RBD & LSD- Modelling, Estimation and interpretation of Coefficients.RBD and LSD.	**4 hours**
	Total Lecture hours	**30 hours**
Text Book(s)		
1	Probability and Statistics with R, Maria Dolores Ugarte, Ana F. Militino, and Alan T. Arnholt, 2nd Edition, Chapman and Hall/CRC, 2016.	
Reference Book(s)		
1	The Book of R: A First Course in Programming and Statistics, Tilman M. Davies, No Starch Press, 1st Edition, 2016.	
2	Applied Statistics: Theory and Problem Solutions with R, Dieter Rasch, Rob Verdooren,	

	and Jürgen Pilz, Wiley, 1st Edition, 2019.		
Mode of evaluation: Lab Continuous Assessment and Lab Final Assessment Test.			
Recommended by Board of Studies	14.05.2025		
Approved by Academic Council	No. 78	Date	12.06.2025

Course Code	Course Title	L	T	P	C
	Engineering Physics	**3**	**0**	**2**	**4**
Pre-requisite		**Syllabus Version**			
		1.0			

Course Objectives

1. Understand the origin and importance of quantum mechanics.
2. To study the principles of quantum mechanics
3. To apply quantum mechanics for the understanding of quantum computing

Course Outcomes

At the end of this course, the student will be able to

1 **Identify** limitations of classical physics through experimental observations.
2 **Apply** matrix algebra and Dirac notation for the understanding of quantum mechanical problems involving linear operators, eigenvalues and eigenvectors.
3 **Solve** the particle in 1-D and 3-D potential box problem using the principles of quantum mechanics.
4 **Apply** fundamental concepts of quantum states and operators to understand the principles of quantum computing.
5 **Demonstrate** quantum and optical phenomena through experiments and quantum operations through simulations.

Module:1	**Wave-Particle Duality**	**8 hours**

Dual nature of radiation – Young's double slit experiment – Blackbody radiation – Planck's hypothesis – de Broglie hypothesis – Wavefunction – Stern-Gerlach experiment.

Module:2	**Mathematical Foundations of Quantum Mechanics**	**10 hours**

Introduction to linear vector space – Basis vectors – Orthonormal sets – Hilbert space - Representation of quantum states using Dirac notation – Inner and outer products – Tensor product of vector spaces – Linear operators – Matrix algebra – Hermitian, Unitary and Projection operators – Eigenvalues and eigenvectors – Pauli matrices – Commutation relations.

Module:3	**Postulates of Quantum Mechanics**	**8 hours**

Observables ⁓ Expectation values ⁓ Measurement in Quantum Mechanics – Sequential Stern-Gerlach Experiment – Time-dependent and time-independent Schrödinger equations.

Module:4	**Applications of Quantum Mechanics**	**7 hours**

Particle in a one-dimensional potential well – Extension to three-dimensional potential wells – Energy eigenvalues and eigenfunctions – Degeneracy of energy levels – Quantum tunneling phenomenon – Tunneling probability

Module:5	**Elements of Quantum Computing**	**10 hours**

Introduction to qubits – Single qubit states and the Bloch Sphere – Two-qubit systems – Quantum entanglement and Bell states – Single-qubit gates: Pauli and Hadamard – Two-qubit gates: CNOT – Generation of entangled states.

Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture hours:	**45 hours**

Text Books

1	D. J. Griffiths, Introduction to Quantum Mechanics, 2018, 3rd Edition, Cambridge University Press, India.
2	A. C. Philips, Introduction to Quantum Mechanics, 2014, Wiley India.

Reference Books

1.	Robert Eisberg and Robert Resnick, Quantum Physics, 2004, 2nd Edition. Wiley India.
2.	Michael A. Nielsen and Isaac L. Chuang, Quantum Computation and Quantum Information, 2010, Cambridge Universities Press, India.
Mode of Evaluation: Quiz, Assignment, CAT and FAT	
List of Challenging and Mini Projects - CHAMP (Indicative)	
1.	Exploring the wave nature of electrons through diffraction.
2.	Determine the fundamental quantum constant through the analysis of semiconductor junction characteristics.
3.	Elemental analysis through photoelectron spectroscopy.
4.	Unveiling stellar composition through quantum simulations of hydrogen spectral lines.
5.	Modeling time dependent behaviour of quantum wave packets.
6.	Designing a solar cell based renewable energy device circuits for electric power generation.
7.	Analyze photodiode I–V characteristics under different light intensities to extract device parameters.
8.	Visualize the uncertainty principle by simulating position-momentum evolution of quantum wave packets.
9.	Simulate the Schrödinger equation to demonstrate quantum superposition and state collapse.
10.	Model tunneling through barriers of different profiles and study dependence on particle energy.
11.	Visualize the Bloch sphere representation of a single qubit through simulation.
12.	Demonstrate the fundamental operations of single-qubit quantum gates via simulation.
13.	Simulating a 2-Qubit Quantum Gate (CNOT or Bell State Creation)
14.	Simulate the superposition state of a single qubit using the Hadamard.
15.	Generate and analyze Bell states using quantum circuit simulation.
Total Laboratory Hours	**30 hours**

Text Book(s)	
1.	Michael A. Nielsen & Isaac L. Chuang, Quantum Computation and Quantum Information, 2010, Cambridge University Press, India
Reference Books	
1.	Aurthur Beiser, Concepts of Modern Physics, 2009, McGraw - Hill, 6th Edition, India
Mode of assessment: Continuous assessment, FAT, Oral examination	

Recommended by Board of Studies	DD-MM-YYYY		
Approved by Academic Council	No. xx	Date	DD-MM-YYYY

Course Code	Course Title	L	T	P	C
	Engineering Physics	**3**	**0**	**2**	**4**
Pre-requisite		**Syllabus Version**			
		1.0			

Course Objectives

1. Understand the origin and importance of quantum mechanics.
2. To study the principles of quantum mechanics
3. To apply quantum mechanics for the understanding of quantum computing

Course Outcomes

At the end of this course, the student will be able to

1 **Identify** limitations of classical physics through experimental observations.
2 **Apply** matrix algebra and Dirac notation for the understanding of quantum mechanical problems involving linear operators, eigenvalues and eigenvectors.
3 **Solve** the particle in 1-D and 3-D potential box problem using the principles of quantum mechanics.
4 **Apply** fundamental concepts of quantum states and operators to understand the principles of quantum computing.
5 **Demonstrate** quantum and optical phenomena through experiments and quantum operations through simulations.

Module:1	**Wave-Particle Duality**	**8 hours**

Dual nature of radiation – Young's double slit experiment – Blackbody radiation – Planck's hypothesis – de Broglie hypothesis – Wavefunction – Stern-Gerlach experiment.

Module:2	**Mathematical Foundations of Quantum Mechanics**	**10 hours**

Introduction to linear vector space – Basis vectors – Orthonormal sets – Hilbert space - Representation of quantum states using Dirac notation – Inner and outer products – Tensor product of vector spaces – Linear operators – Matrix algebra – Hermitian, Unitary and Projection operators – Eigenvalues and eigenvectors – Pauli matrices – Commutation relations.

Module:3	**Postulates of Quantum Mechanics**	**8 hours**

Observables ⁓ Expectation values ⁓ Measurement in Quantum Mechanics – Sequential Stern-Gerlach Experiment – Time-dependent and time-independent Schrödinger equations.

Module:4	**Applications of Quantum Mechanics**	**7 hours**

Particle in a one-dimensional potential well – Extension to three-dimensional potential wells – Energy eigenvalues and eigenfunctions – Degeneracy of energy levels – Quantum tunneling phenomenon – Tunneling probability

Module:5	**Elements of Quantum Computing**	**10 hours**

Introduction to qubits – Single qubit states and the Bloch Sphere – Two-qubit systems – Quantum entanglement and Bell states – Single-qubit gates: Pauli and Hadamard – Two-qubit gates: CNOT – Generation of entangled states.

Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture hours:	**45 hours**

Text Books

1	D. J. Griffiths, Introduction to Quantum Mechanics, 2018, 3rd Edition, Cambridge University Press, India.
2	A. C. Philips, Introduction to Quantum Mechanics, 2014, Wiley India.

Reference Books

1.	Robert Eisberg and Robert Resnick, Quantum Physics, 2004, 2nd Edition. Wiley India.
2.	Michael A. Nielsen and Isaac L. Chuang, Quantum Computation and Quantum Information, 2010, Cambridge Universities Press, India.
Mode of Evaluation: Quiz, Assignment, CAT and FAT	
List of Challenging and Mini Projects - CHAMP (Indicative)	
1.	Exploring the wave nature of electrons through diffraction.
2.	Determine the fundamental quantum constant through the analysis of semiconductor junction characteristics.
3.	Elemental analysis through photoelectron spectroscopy.
4.	Unveiling stellar composition through quantum simulations of hydrogen spectral lines.
5.	Modeling time dependent behaviour of quantum wave packets.
6.	Designing a solar cell based renewable energy device circuits for electric power generation.
7.	Analyze photodiode I–V characteristics under different light intensities to extract device parameters.
8.	Visualize the uncertainty principle by simulating position-momentum evolution of quantum wave packets.
9.	Simulate the Schrödinger equation to demonstrate quantum superposition and state collapse.
10.	Model tunneling through barriers of different profiles and study dependence on particle energy.
11.	Visualize the Bloch sphere representation of a single qubit through simulation.
12.	Demonstrate the fundamental operations of single-qubit quantum gates via simulation.
13.	Simulating a 2-Qubit Quantum Gate (CNOT or Bell State Creation)
14.	Simulate the superposition state of a single qubit using the Hadamard.
15.	Generate and analyze Bell states using quantum circuit simulation.
Total Laboratory Hours	**30 hours**

Text Book(s)	
1.	Michael A. Nielsen & Isaac L. Chuang, Quantum Computation and Quantum Information, 2010, Cambridge University Press, India
Reference Books	
1.	Aurthur Beiser, Concepts of Modern Physics, 2009, McGraw - Hill, 6th Edition, India
Mode of assessment: Continuous assessment, FAT, Oral examination	

Recommended by Board of Studies	DD-MM-YYYY		
Approved by Academic Council	No. xx	Date	DD-MM-YYYY

Course Code	Course Title	L	T	P	C
BASTS101	Qualitative and Quantitative Skills Practice I	3	0	0	1
Pre-requisite	Nil	**Syllabus Version**			
		1			

Course Objectives

1. To enhance students' quantitative, reasoning, verbal, and life skills through problem-solving and critical thinking.

Course Outcomes

1. Apply quantitative and logical reasoning techniques such as speed math, data arrangement, and analytical problem-solving to solve aptitude and reasoning-based problems efficiently.
2. Demonstrate effective life skills, including profile building, decision making, critical thinking, and communication skills for personal and professional growth in placement and career contexts.

Module:1	**Speed Math**	**8 hours**

Addition and Subtraction of Bigger Numbers - Square and square roots-Cubes and cube roots-Vedic Math - Multiplication Shortcuts -Multiplication of 3 and higher digit numbers-Simplifications - Comparing fractions – Calculation Shortcuts - Divisibility rules

Module:2	**Percentages, Simple and Compound Interest**	**6 hours**

Percentage -Increase and Decrease – Dealing with Fractions and Decimals - Simple Interest -Compound Interest - Relation between Simple and Compound Interest

Module:3	**Profit and Loss, Partnership, Time, Speed and Distance**	**10 hours**

Profit and Loss – Partnership- Averages and Weighted Averages- Mixtures and Allegations-Basics- Relative Speed- Average Speed- Problems on Trains -Problems on Boats and Streams- Problem on Races

Module:4	**Data Arrangements, Blood relations, Cryptarithmetic and Attention to detail**	**10 hours**

Linear Arrangement -Circular Arrangement-Multi-dimensional Arrangement- Blood relations - Crypt Arithmetic Type: Addition, Multiplication - Attention to detail

Module:5	**Reading Comprehension and Vocabulary**	**6 hours**

Types of questions - Comprehension Strategies - Synonyms – Antonyms – Analogy - Confusing words

Module:6	Profile Building, Professional Networking and Resume writing	5 hours
Profile building-Learn how to create a strong personal and professional profile - Identify elements of a resume, portfolio, and LinkedIn profile, Importance of professional networking		
	Total Lecture Hours:	**45 hours**
Text Book(s)		
Dr R. S. Aggarwal," **Quantitative Aptitude for Competitive Examinations**", Chand & Company Ltd, ISBN 978-9358705010, Revised Edition, **2024-25** M. Tyra, "**Magical Book on Quicker Maths**", BSC Publishing Co. Private. Ltd, ISBN 978-8190458924,5th Edition, **2024**		
Reference Books		
Stephen R. Covey," **The 7 Habits of Highly Effective People**", Simon & Schuster, ISBN 978-1982137274, 30th Anniversary Edition, **2020** Stella Cottrell, "**Critical Thinking Skills: Effective Analysis, Argument and Reflection**," Bloomsbury Academic (Palgrave Macmillan), ISBN 978-1350322585, 4th Edition, **2023**		
Mode of Evaluation: Continuous Assessment Test, Quiz, Final Assessment Test		
Recommended by the Board of Studies:		
Approved by Academic Council:		

Course Code	Course Title	L	T	P	C
BASTS102	Qualitative and Quantitative Skills Practice II	3	0	0	1
Pre-requisite	Nil	**Syllabus Version**			
		1			

Course Objectives		
1. Equip learners with key life skills and problem-solving, while preparing them for career opportunities through training in quantitative aptitude, logical reasoning, verbal ability and resume building.		
Course Outcomes		
1. Apply time management techniques, write professional emails, and prepare well-structured reports to enhance productivity and communication. 2. Solve quantitative problems involving time and work, probability, and logical reasoning with accuracy and confidence.		
Module:1	**Permutation and Combinations, Probability**	**6 hours**
Fundamentals of Counting Principle, Computation of Permutations – Computation of Combinations – Circular Permutation and Probability		
Module:2	**Time and Work, Logarithms, Progression**	**8 hours**
Work with different efficiencies - Pipes and Cisterns - Work equivalence - Division of wages - Advanced application problems with complexity in calculation - Logarithms, Arithmetic Progression - Geometric Progression and Harmonic Progression		
Module:3	**Logical Reasoning**	**9 hours**
Coding and decoding - Visual Reasoning – Clocks – Calendars - Direction Sense - Cubes		
Module:4	**Number System and Algorithmic Thinking**	**10 hours**
Power Cycle - Reminder Cycle - Applications of number system- Introduction to Algorithmic Thinking and Problem-solving, thinking with numbers, thinking with visuals and finding the shortest path, Branching and repetitive problems		
Module:5	**Verbal Ability**	**7 hours**
Sentence Correction - Subject Verb agreement – Modifiers – Parallelism - Pronoun Antecedent Agreement - Verb Time sequence – Comparisons – Prepositions - Determiners		

Module:6	**Grooming, Impression Management and Personal Interview**	**5 hours**
Grooming- Dressing Etiquette-Impression Management - Building a Positive Personal Brand - Body Language & Non-Verbal Cues -Types of Interviews – HR Interview, Telephonic Interview, Panel Interview		
	Total Lecture Hours:	**45 hours**
Text Book(s)		
R.S. Aggarwal, " **Quantitative Aptitude for Competitive Examinations**", S. Chand Publishing, ISBN 978-8121924624, Revised Edition, **2024-25** John Sonmez, " **Soft Skills: The Software Developer's Life Manual** ", Manning Publications, ISBN 978-1617292392, 1st Edition, 2014		
Reference Books		
R.S. Aggarwal., " **A Modern Approach to Logical Reasoning** ", S. Chand Publishing, ISBN 978-9352833227, 1st Edition, **2018** Richard Paul & Linda Elder, **" Critical Thinking: Tools for Taking Charge of Your Professional and Personal Life** ", Pearson ISBN 978-0132180915, 2nd Edition, **2014**		
Mode of Evaluation: Continuous Assessment Test, Quiz, Final Assessment Test		
Recommended by the Board of Studies:		
Approved by Academic Council:		

Course Code	Course Title	L	T	P	C
BACSE103	Computation Structures	3	0	2	4
Pre-requisite		Syllabus Version			
		1.0			

Course Objectives

1. To Understand and design of digital logic systems ranging from simple combinational circuits to sequential designs.
2. To impart the knowledge of data representation and to understand the implementation of arithmetic algorithms in a typical computer.
3. To introduce and explore the fundamental concepts of microcontrollers and techniques of parallel processing.

Course Outcomes

1. To design and optimize minimal combinational circuits.
2. To analyze and design sequential circuits utilizing various types of flip-flops.
3. To analyze different instruction formats, addressing modes and validate efficient algorithms for fixed-point arithmetic operations.
4. To understand the concept of memory organization and I/O fundamentals
5. To understand the microcontroller architectural designs and parallel computing systems.

Module:1	Digital Systems and Combinational Circuits	7 hours

Introduction to Digital systems: Number system, Data representation: Fixed and Floating point, Logic Gates, Simplification of Boolean function using K-Map, Design of combinational circuit: Adder, Carry look-ahead adder, Encoder and Decoder, Multiplexer and Demultiplexer.

Module:2	Sequential Circuits	9 hours

Flip Flops, Sequential circuit design and Analysis: Finite State Machines-Mealy and Moore, Registers, and Counters: Synchronous and Asynchronous, Timing Sequences.

Module:3	Computer Architectures and Arithmetic Algorithms	10 hours

Introduction to Von Neumann machine and Harvard architecture, Fixed point arithmetic operations: Multiplication (Booths, Modified Booths), Division (restoring and non-restoring). Instruction set architecture, Instruction formats, Instruction types, addressing modes, Instruction cycle, Single cycle Datapath design, Multicycle Datapath design.

Module:4	Memory and I/O Peripherals	8 hours

Memory Organization, Memory Hierarchy, Memory Design, Cache Memory-Mapping Policies, Replacement Algorithms, I/O Fundamentals- programmed I/O and Interrupt driven I/O, Direct Memory Access, Shared Memory Architectures.

Module:5	Advanced Architecture	9 hours

Introduction to Microcontroller, ARM Architecture Versions– Instruction Set – Stacks and Subroutines. Parallel processing Techniques: Instruction Level parallelism, and Thread Level Parallelism.

Module:6	Contemporary Issues	2 hours
	Total Lecture hours:	45 hours

Text Books

1.	M. Morris Mano, Digital Logic and Computer Design, Fifth Edition, Pearson India Education Publishers, 2019
2.	Patterson, David A., and John L. Hennessy, Computer organization and design ARM edition: the hardware software interface, First Edition, Morgan kaufmann, 2016.
Reference Books	
1.	William Stallings, Computer Architecture and Organization-Designing for Performance, Eleventh edition, Pearson Education series, 2022.
2.	Ananth Grama, Anshul Gupta, George Karypis and Vipin Kumar, Introduction to Parallel Computing, 2nd Edition, Pearson, 2008.
Mode of Evaluation: Quiz, Assignment, CAT and FAT	
List of Experiments (Indicative)	
1	Combinational Circuit Design - Full Adder, Full Subtractor, Binary Parallel Adder. Consider there are two counter circuit employed in a food mall one at the front door near the staircase and another beside the elevator, to count the number of customers. At time 't' the number of customers entered through staircase and elevator are counted by the two counter circuits as 2 and 3 respectively. Design a suitable combinational circuit to find the sum of these two-counter output at time 't'.
2	Combinational Circuit Design - Decoder, Encoder. Build a decoder with three input lines but with only six output lines. If the value of the input corresponds to 6 or 7, then all output lines should be asserted to signal an error.
3	Combinational Circuit Design - MUX, DMUX. You are required by your manager to design a 4-bit prime number checker – you want to output 1 if the binary number is primer else 0 (note 0 and 1 are not prime numbers). a. Show the truth table for the primer number checker. b. Design the circuit using single 4x1 MUX and a minimal number of extra AND, OR or NOT gates if needed (i.e., no NAND, NOR, XOR or XNOR, etc.). Show your work including any simplifications done. You must show the logic diagram with all the pins of the mux labeled properly. State any assumptions that you make.
4	Sequential Circuit Design - Flipflop, Shift Register. Use a SIPO shift register to display a binary counter on 8 LEDs. Each second, the counter increments by 1, and the value is sent serially to the register.
5	Sequential Circuit Design - Synchronous and Asynchronous Counters. A bank queuing system has a capacity of 5 customers which serves on first come first served basis. A display unit is used to display the number of customers waiting in the queue. Whenever a customer leaves the queue, the count is reduced by one and the count is increased by one if a customer joins a queue. Two sensors (control signals) are used to sense customers leaving

	and joining the queue respectively. Design a circuit that displays the number of customers waiting in the queue in binary format using LEDs. Binary '1' is represented by LED glow and '0' otherwise.
6	Introduction to Microcontroller -To design and implement a simple calculator
7	Interfacing with GPIO - To interface with GPIO pins by controlling an LED and detecting a button press using a microcontroller.
8	Interfacing Bluetooth module with microcontroller - Connect a Bluetooth module (e.g., HC-05) to the microcontroller and send/receive simple text messages using serial window.
9	Introduction to parallel programming using OpenMP - Write a OpenMP program to find the largest of three numbers and smallest of three numbers using explicit thread identification.
10	OpenMP work sharing constructs - Write a program to perform matrix operations and compare the sequential and parallel execution time.
11	OpenMP synchronization constructs- Demonstrate an example by using Single, Master, Barrier, and critical constructs of OpenMP.
12	Using Performance Analyzer tools find the execution time and memory usage of a parallel program.
	Total hours: 30 hours

Mode of Evaluation:			
Recommended by Board of Studies			
Approved by Academic Council	**No.**	**Date**	

Course Code	Course Title	L	T	P	C
BACSE103	Computation Structures	3	0	2	4
Pre-requisite		Syllabus Version			
		1.0			

Course Objectives

1. To Understand and design of digital logic systems ranging from simple combinational circuits to sequential designs.
2. To impart the knowledge of data representation and to understand the implementation of arithmetic algorithms in a typical computer.
3. To introduce and explore the fundamental concepts of microcontrollers and techniques of parallel processing.

Course Outcomes

1. To design and optimize minimal combinational circuits.
2. To analyze and design sequential circuits utilizing various types of flip-flops.
3. To analyze different instruction formats, addressing modes and validate efficient algorithms for fixed-point arithmetic operations.
4. To understand the concept of memory organization and I/O fundamentals
5. To understand the microcontroller architectural designs and parallel computing systems.

Module:1	Digital Systems and Combinational Circuits	7 hours

Introduction to Digital systems: Number system, Data representation: Fixed and Floating point, Logic Gates, Simplification of Boolean function using K-Map, Design of combinational circuit: Adder, Carry look-ahead adder, Encoder and Decoder, Multiplexer and Demultiplexer.

Module:2	Sequential Circuits	9 hours

Flip Flops, Sequential circuit design and Analysis: Finite State Machines-Mealy and Moore, Registers, and Counters: Synchronous and Asynchronous, Timing Sequences.

Module:3	Computer Architectures and Arithmetic Algorithms	10 hours

Introduction to Von Neumann machine and Harvard architecture, Fixed point arithmetic operations: Multiplication (Booths, Modified Booths), Division (restoring and non-restoring). Instruction set architecture, Instruction formats, Instruction types, addressing modes, Instruction cycle, Single cycle Datapath design, Multicycle Datapath design.

Module:4	Memory and I/O Peripherals	8 hours

Memory Organization, Memory Hierarchy, Memory Design, Cache Memory-Mapping Policies, Replacement Algorithms, I/O Fundamentals- programmed I/O and Interrupt driven I/O, Direct Memory Access, Shared Memory Architectures.

Module:5	Advanced Architecture	9 hours

Introduction to Microcontroller, ARM Architecture Versions– Instruction Set – Stacks and Subroutines. Parallel processing Techniques: Instruction Level parallelism, and Thread Level Parallelism.

Module:6	Contemporary Issues	2 hours
	Total Lecture hours:	45 hours

Text Books

1.	M. Morris Mano, Digital Logic and Computer Design, Fifth Edition, Pearson India Education Publishers, 2019
2.	Patterson, David A., and John L. Hennessy, Computer organization and design ARM edition: the hardware software interface, First Edition, Morgan kaufmann, 2016.
Reference Books	
1.	William Stallings, Computer Architecture and Organization-Designing for Performance, Eleventh edition, Pearson Education series, 2022.
2.	Ananth Grama, Anshul Gupta, George Karypis and Vipin Kumar, Introduction to Parallel Computing, 2nd Edition, Pearson, 2008.
Mode of Evaluation: Quiz, Assignment, CAT and FAT	
List of Experiments (Indicative)	
1	Combinational Circuit Design - Full Adder, Full Subtractor, Binary Parallel Adder. Consider there are two counter circuit employed in a food mall one at the front door near the staircase and another beside the elevator, to count the number of customers. At time 't' the number of customers entered through staircase and elevator are counted by the two counter circuits as 2 and 3 respectively. Design a suitable combinational circuit to find the sum of these two-counter output at time 't'.
2	Combinational Circuit Design - Decoder, Encoder. Build a decoder with three input lines but with only six output lines. If the value of the input corresponds to 6 or 7, then all output lines should be asserted to signal an error.
3	Combinational Circuit Design - MUX, DMUX. You are required by your manager to design a 4-bit prime number checker – you want to output 1 if the binary number is primer else 0 (note 0 and 1 are not prime numbers). a. Show the truth table for the primer number checker. b. Design the circuit using single 4x1 MUX and a minimal number of extra AND, OR or NOT gates if needed (i.e., no NAND, NOR, XOR or XNOR, etc.). Show your work including any simplifications done. You must show the logic diagram with all the pins of the mux labeled properly. State any assumptions that you make.
4	Sequential Circuit Design - Flipflop, Shift Register. Use a SIPO shift register to display a binary counter on 8 LEDs. Each second, the counter increments by 1, and the value is sent serially to the register.
5	Sequential Circuit Design - Synchronous and Asynchronous Counters. A bank queuing system has a capacity of 5 customers which serves on first come first served basis. A display unit is used to display the number of customers waiting in the queue. Whenever a customer leaves the queue, the count is reduced by one and the count is increased by one if a customer joins a queue. Two sensors (control signals) are used to sense customers leaving

	and joining the queue respectively. Design a circuit that displays the number of customers waiting in the queue in binary format using LEDs. Binary '1' is represented by LED glow and '0' otherwise.
6	Introduction to Microcontroller -To design and implement a simple calculator
7	Interfacing with GPIO - To interface with GPIO pins by controlling an LED and detecting a button press using a microcontroller.
8	Interfacing Bluetooth module with microcontroller - Connect a Bluetooth module (e.g., HC-05) to the microcontroller and send/receive simple text messages using serial window.
9	Introduction to parallel programming using OpenMP - Write a OpenMP program to find the largest of three numbers and smallest of three numbers using explicit thread identification.
10	OpenMP work sharing constructs - Write a program to perform matrix operations and compare the sequential and parallel execution time.
11	OpenMP synchronization constructs- Demonstrate an example by using Single, Master, Barrier, and critical constructs of OpenMP.
12	Using Performance Analyzer tools find the execution time and memory usage of a parallel program.
	Total hours: 30 hours

Mode of Evaluation:			
Recommended by Board of Studies			
Approved by Academic Council	**No.**	**Date**	

Course Code	Course Title	L	T	P	C
BACSE104	Structured and Object-Oriented Programming	2	0	4	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To foster a strong foundation in structured programming using C and object-oriented programming using C++, with a focus on memory management, pointers, and dynamic memory allocation for real-world applications
2. To facilitate modular, reusable, and efficient software development through functions, user-defined data types, and object-oriented design principles, emphasizing best programming practices.
3. To build hands-on programming skills and structure problem-solving abilities by designing and implementing applications addressing real-world scenarios.

Course Outcomes

1. To analyze and implement basic programming constructs, decision-making statements, and loops in C/C++
2. To apply modular programming techniques by utilizing functions, structures, and files for efficient problem-solving in real-world applications.
3. To apply pointers and dynamic memory allocation techniques to manage data efficiently in C/C++ programs.
4. To demonstrate object-oriented principles (encapsulation, inheritance, and polymorphism) to design structured and scalable applications.
5. To develop efficient and scalable solutions and to apply advanced programming paradigms, including exception handling and the Standard Template Library (STL) with set, list, and map.

Module:1	Fundamentals of C Programming	6 hours

Structure of C Program, Compiler Vs Interpreter, Variables, Reserved words, Data types, Operators - precedence and associativity, Type conversions, I/O statements, control structures - conditional and looping statements, 1-D and 2-D Arrays, String operations, Functions – user defined functions - call by value and call by reference - recursion, storage classes - scope- visibility - lifetime of variables.

Module:2	Pointers Structures and File Handling in C	7 hours

Pointers - pointer arithmetic, Pointers with Arrays and functions, function pointers, Dynamic Memory Allocation, Structures- array of structures - nested structures - structures with functions - self-referential structures - bit fields - pointers to structures, Unions- comparison with structures, File operations- handling text and binary files – read - write - seek, command-line arguments.

Module:3	Introduction to Object Oriented Programming	5 hours

Features of object-oriented programming, classes and objects, access specifiers, constructors and destructors, *this* pointer, Static: data members - member functions -objects, constant member functions, inline member functions, friend functions and friend classes, scope resolution operator, lambda expression		
Module:4	**Inheritance and Polymorphism in CPP**	**6 hours**
Types of inheritance - single, multiple, multilevel, hierarchical, multipath inheritance, ambiguity in multipath inheritance, Constructors under inheritance, Function overloading, Operator overloading, Function overriding in Dynamic Polymorphism, Virtual functions, Virtual destructors, Pure virtual functions - Abstract classes.		
Module:5	**Exceptions Templates and STL in CPP**	**4 hours**
Exception handling - try, catch, throw constructs, handling runtime errors, custom exception, re-throwing exceptions for hierarchical error management, Function templates and Class templates, Introduction to type parameters and basic template specialization, Standard Template Library (STL) containers - vector, list, set, and map. Iterators for traversing containers - application of standard STL algorithms		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**30 hours**
Text Book(s)		
Paul Deitel and Harvey Deitel , " **C How to Program**", Pearson Education, 9th Edition, **2023** Herbert Schildt, " **C++: The Complete Reference**", McGraw Hill Education, 4th Edition, **2017**		
Reference Books		
Yashavant Kanetkar, "**Let Us C**", BPB Publications, 20th Edition, **2024** Herbert Schildt, "**The Complete Reference C**", McGrawHill Education, 4th Edition, **2017** Reema Thareja , "**Programming in C**", Oxford University Press, 2nd Edition, **2016** Paul Deitel and Harvey Deitel, " **C++ How to Program**", Pearson Education, 10th Edition, **2023**		
Indicative Experiments		
1. **Domestic Electricity Bill Computation** In urban households, electricity consumption is monitored monthly. To promote energy conservation and fair usage, electricity boards apply slab-wise billing based on units consumed. You are tasked to write a C program that helps households to compute their monthly electricity bill based on actual		4 hours

domestic usage. The bill is calculated based on the number of units consumed using a slab system as follows: 0–100 units: Rs. 1.50 per unit 101–300 units: Rs. 2.00 per unit Above 300 units: Rs. 3.00 per unit The code should: Prompt the user to enter the number of units consumed Calculate the bill based on slabs Display the units consumed and the total bill amount Ask the user if they wish to compute another bill Exit gracefully with a thank-you message	
2. **Fuel Cost Estimator for Road Trips** Planning a road trip requires estimating travel expenses, especially fuel costs. With fluctuating fuel prices and varying mileage across vehicles, travelers often struggle to predict their trip cost accurately. To aid this, design a program to estimate fuel cost based on distance, mileage, and fuel price. Write a C program to help a user estimate the total fuel expense for a road trip. The user needs to enter the total distance of the trip (in kilometers), the vehicle's mileage (in km/l), and the current fuel price (in Rs. per litre). The program should: accept trip distance, mileage, and fuel cost as inputs Calculate: Litres Needed = Distance / Mileage Fuel Cost = Litres Needed × Fuel Price Display the calculated values Allow repeated estimation based on user input (Yes/No) Exit gracefully with a thank-you message	4 hours
3. **Student Marks Analyzer for Internal Assessment** A college faculty member wants to automate the process of analyzing internal marks of students for a subject. Instead of manually reviewing each score, a program is needed to accept marks for multiple students, calculate the class average, and identify the highest and lowest scorers. Write a C program that	4 hours

helps an instructor analyze internal marks of students in a subject. The program should: Ask the user to enter the number of students in the class. Accept individual marks (out of 100) for all students Calculate and display: Average marks of the class Highest and lowest marks scored Allow the user to repeat the process for a different class Exit gracefully with a thank-you message	
4. **Smart Grocery Billing System with Limited-Time Offers** A local grocery store is offering limited-time discounts on selected items for a day. The store manager wants a billing assistant that can: Take product names, prices, and quantities for a customer's purchase. Calculate total price and display an item-wise summary Apply a one-time discount only for the first bill generated in a day. Provide a running total for customers. Develop using C program for a smart grocery billing system that performs the following: For each product, input: Product name Quantity Unit price Calculate: Total price for each item Grand total If it's the first customer of the day, apply a 10% discount on the bill. Store and display the summary in a structured format. Repeat the process for the next customer based on user choice Exit with thanks note.	4 hours
5.	4 hours

Health Tracker: BMI Analyzer with Memory-Saving Feature A digital health startup is developing a lightweight health tracking tool to help users calculate and monitor their Body Mass Index (BMI). In order to reduce memory usage and allow efficient access, the system is required to operate with minimal variables and use references wherever it is necessary. You've been asked to simulate the logic of this core functionality. Develop a C program that computes the Body Mass Index (BMI) for multiple individuals and classifies them based on health categories. The program should: Accept the number of users to track For each user: Input name Weight in kilograms Height in meters Compute BMI using the formula: BMI = weight / (height * height) Classify as: BMI < 18.5: Underweight 18.5 ≤ BMI < 25: Normal 25 ≤ BMI < 30: Overweight BMI ≥ 30: Obese	
6. **Conference Registration System with Dynamic Allocation** An academic institute is organizing a national-level technical conference. Participants from various colleges register online, but the number of registrants isn't fixed in advance. To efficiently manage memory, the system needs to allocate space dynamically based on the actual number of registrations. Write a C program to develop a **conference registration system** where: The user inputs the number of participants registering for the event. For each participant, input:	4 hours

Full Name Institution Name Email ID Dynamically allocate memory to store participant details using malloc() or calloc(). After all entries: Display the registered list in a tabular format. Free the dynamically allocated memory after use. Ask whether the user wants to proceed to register a new batch. Exit with a thank-you message.	
7. **Digital Payment Ledger System** A local cooperative society is planning to digitize the ledger for tracking payments made by its members. The system should support entering new transactions, storing them in files, retrieving records, and displaying summaries. Since the number of transactions varies, memory and data handling must be dynamic. Develop a C program that simulates a **digital payment ledger system** with the following features: Accept the number of transactions for the day and dynamically allocate memory. For each transaction, input: Member Name Transaction ID Payment Amount Payment Method (UPI/Cash/Card) Store these entries in a structure using dynamic memory allocation. Display the transactions in a summary table. Write all transactions to a file named ledger.txt. Allow the user to read and display the full ledger history from the file. Ask the user whether to continue or exit.	4 hours
8.	4 hours

Hospital Patient Record Tracker A small clinic wants to keep track of patients' visit information. For different types of patients (in-patient and out-patient), certain information is handled differently. Use structures and unions to design a memory-efficient system. Write a C program to: Use a structure named Patient with common fields: Name Age Gender Patient Type (I for in-patient, O for out-patient) Include a union inside the structure to hold: For in-patients: number of days admitted and room number. For out-patients: consultation fee and doctor's name. Accept n patients' details from the user. Store their information in an array of structures. Display a summary for all patients, handling the union properly.	
9. **Faculty Feedback Archiver System** Your college wants to digitize the storage of semester-wise faculty feedback. Each feedback entry consists of the following information: Faculty Name Course Code Student Registration Number Feedback Score (out of 10) Feedback Comments Each feedback should be written to a text file named feedback.txt. Later, the system should allow retrieval of all feedback for a specific faculty. Write a C program to: Accept the number of feedback entries from the user. Dynamically collect data for each feedback entry using a structure.	4 hours

Write each entry to feedback.txt in append mode. After data entry, ask for a faculty name and return the entire feedback given to that faculty.	
10. **Library Book Management System** A college library wants to maintain its book inventory digitally. The system should allow librarians to add new books, store them in a file, retrieve the complete catalog, and manage dynamic memory allocation as the number of books may vary. The system should also leverage C++ features like classes, constructors, and file streams. Develop a **C++ program** to simulate a **Library Book Management System** with the following features: **Specifications:** Create a class Book with the following attributes: title (string) author (string) ISBN (string) price (float) Dynamically allocate memory for storing multiple books (preferably using vector) Allow the user to: Add new books to the catalog. Display the current book list in tabular format. Choose to continue or exit.	4 hours
11. **Student Admission Record System** An engineering college is developing a software module to automate its student admission process. The system should handle new student entries, display current records, count the number of students admitted so far, and store all details persistently. The system must manage memory dynamically, ensure data encapsulation, and clean up resources properly once the program ends.	4 hours

Develop a **C++ program** for the **Student Admission Record System** with the following features: **Specifications:** Each student record should include: Student Name Admission Number (auto-generated or input) Department Fees Paid Track the **total number of students admitted using friend function** (across program runs in a session). Dynamically store and display student records in a table. Ensure resource cleanup when records are no longer needed **.**	
12. **Library Book Cataloging and Serial Number Generator** A school library is digitizing its book catalog. Every time a new book is added, it should automatically be assigned a unique serial number. The librarian will enter book details, and the system should display all books with their assigned serials. The serial numbers must be generated **internally** and **incremented automatically** starting from 1001. Each book has: Title Author Publisher Year of Publication The system must: Use a **class to represent a book** Assign and manage **serial numbers automatically** using a **static data member** Use dynamic memory allocation inside the **constructors** to initialize book entries Display details of each book added On exit, confirm that memory is released (implicitly show destructor effect)	4 hours

Write a **C++ program** that: Accepts the number of books to enter. For each book, accepts title, author, publisher, and year. Assigns a unique serial number internally starting from 1001. Displays the full catalog. Demonstrates the **constructor and destructor calls** **Expected Features to Implement:** **Static data member** for serial number generation **Parameterized constructor** to initialize book details **Destructor** to display a message when book objects are destroyed Optionally overload a function to display different formats	
13. **Fitness Tracker - Personal Health Metrics System** A fitness center wants to offer its members personal health metrics tracking software. Each member should be able to record their health data like height, weight, and activity level. The system should calculate and display health indicators like BMI (Body Mass Index) or calorie burn. Members may record partial or complete information depending on availability. Develop a C++ program for the Personal Health Metrics System with the following features: For each member, collect: Member Name Height (in meters) Weight (in kg) Activity Level (optional) — Low / Moderate / High The system should: Calculate and display BMI. Optionally calculate the daily calorie burn estimate, if activity level is provided Handle different types of calculations using multiple versions of functions. Allow repeated entry for multiple members.	4 hours
14.	4 hours

Event Registration and Access System A college is organizing a multi-track tech fest. Participants can register as attendees, presenters, or organizers. Each category has different access rights and information to record. The system should manage participant entries, display details based on roles, and provide a report showing individual access rights in the event. Develop a **C++ program** to simulate an **event registration and access system** with the following features: **Specifications:** Create a base class Participant with: Name, Registration ID, and a virtual function showAccess() Derive three classes from it: Attendee: Can attend sessions Presenter: Can present papers Organizer: Can access all areas For each entry, the system should: Accept type of participant Create objects dynamically and store them (polymorphic array/pointer list) Display a report with role-specific access info using **runtime polymorphism**	
15. Digital Certificate Verification System A university is issuing **digital certificates** for different types of achievements — like academic excellence, sports achievements, and community service. The system should manage certificates, verify their authenticity, and prioritize verification based on issuance year or recipient name. Since the certificates vary in type, you'll use **class templates** to generalize the design, **function templates** for comparing and searching, and **STL containers** to store them dynamically. Write a C++ program to design a system with the following requirements:	2 hours

Class Template: Certificate Template parameter T refers to the **type of achievement** (e.g., string for sports, academic discipline, or service type) Data Members: recipientName (string) certificateID (string) categoryInfo (T) → type-specific detail like "CSE", "Basketball", "Blood Donation" issuedYear (int) Member Functions: display() – show certificate details verifyYear(int) – return true if issuedYear matches given year **Function Template: compareCertificates()** Accepts two Certificate objects Returns the one **issued earlier**, or based on name lexicographically **STL Usage** Use std::vector to store certificate entries dynamically Use iterators to: Display all certificates Search by issued year Filter by recipient name	
16. **Bank Loan Eligibility Checker** A cooperative bank is developing a loan eligibility checker. The system collects basic customer details and validates them against a set of criteria. If any of the eligibility rules are violated, the system must throw an appropriate error and display a meaningful message without crashing the program. Develop a C++ program that: Collects: Customer Name Age Monthly Income Requested Loan Amount Validates:	2 hours

Age must be between 21 and 60 Monthly income must be >= ₹15,000 Loan amount must be <= 10 times the monthly income If any rule is violated, throw a specific exception Catch and handle all exceptions gracefully Display success message only for eligible customers Allow processing of multiple customers (say, 3) **Features to Implement:** Use **custom exception classes** (e.g., InvalidAgeException, LowIncomeException, LoanLimitExceeded) Use **try-catch blocks** Input validation inside constructors or dedicated functions Clean and readable class design	
Total Laboratory Hours:	**60 hours**

Reference Books

Stanley Lippman and Josee Lajoie, "**C++ Primer**", Addison Wesley publishers, 5th Edition, **2012**
John Horton, " **Learn C++ by Example,** ", Manning Publications, 1st Edition, **2023**
Ken Youens-Clark, "**Tiny C Projects**", Manning Publications, 1st Edition, **2021**

Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Seminar

Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE104	Structured and Object-Oriented Programming	2	0	4	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To foster a strong foundation in structured programming using C and object-oriented programming using C++, with a focus on memory management, pointers, and dynamic memory allocation for real-world applications
2. To facilitate modular, reusable, and efficient software development through functions, user-defined data types, and object-oriented design principles, emphasizing best programming practices.
3. To build hands-on programming skills and structure problem-solving abilities by designing and implementing applications addressing real-world scenarios.

Course Outcomes

1. To analyze and implement basic programming constructs, decision-making statements, and loops in C/C++
2. To apply modular programming techniques by utilizing functions, structures, and files for efficient problem-solving in real-world applications.
3. To apply pointers and dynamic memory allocation techniques to manage data efficiently in C/C++ programs.
4. To demonstrate object-oriented principles (encapsulation, inheritance, and polymorphism) to design structured and scalable applications.
5. To develop efficient and scalable solutions and to apply advanced programming paradigms, including exception handling and the Standard Template Library (STL) with set, list, and map.

Module:1	Fundamentals of C Programming	6 hours

Structure of C Program, Compiler Vs Interpreter, Variables, Reserved words, Data types, Operators - precedence and associativity, Type conversions, I/O statements, control structures - conditional and looping statements, 1-D and 2-D Arrays, String operations, Functions – user defined functions - call by value and call by reference - recursion, storage classes - scope- visibility - lifetime of variables.

Module:2	Pointers Structures and File Handling in C	7 hours

Pointers - pointer arithmetic, Pointers with Arrays and functions, function pointers, Dynamic Memory Allocation, Structures- array of structures - nested structures - structures with functions - self-referential structures - bit fields - pointers to structures, Unions- comparison with structures, File operations- handling text and binary files – read - write - seek, command-line arguments.

Module:3	Introduction to Object Oriented Programming	5 hours

Features of object-oriented programming, classes and objects, access specifiers, constructors and destructors, *this* pointer, Static: data members - member functions -objects, constant member functions, inline member functions, friend functions and friend classes, scope resolution operator, lambda expression		
Module:4	**Inheritance and Polymorphism in CPP**	**6 hours**
Types of inheritance - single, multiple, multilevel, hierarchical, multipath inheritance, ambiguity in multipath inheritance, Constructors under inheritance, Function overloading, Operator overloading, Function overriding in Dynamic Polymorphism, Virtual functions, Virtual destructors, Pure virtual functions - Abstract classes.		
Module:5	**Exceptions Templates and STL in CPP**	**4 hours**
Exception handling - try, catch, throw constructs, handling runtime errors, custom exception, re-throwing exceptions for hierarchical error management, Function templates and Class templates, Introduction to type parameters and basic template specialization, Standard Template Library (STL) containers - vector, list, set, and map. Iterators for traversing containers - application of standard STL algorithms		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**30 hours**

Text Book(s)

Paul Deitel and Harvey Deitel , " **C How to Program**", Pearson Education, 9th Edition, **2023**
Herbert Schildt, " **C++: The Complete Reference**", McGraw Hill Education, 4th Edition, **2017**

Reference Books

Yashavant Kanetkar, "**Let Us C**", BPB Publications, 20th Edition, **2024**
Herbert Schildt, "**The Complete Reference C**", McGrawHill Education, 4th Edition, **2017**
Reema Thareja , "**Programming in C**", Oxford University Press, 2nd Edition, **2016**
Paul Deitel and Harvey Deitel, " **C++ How to Program**", Pearson Education, 10th Edition, **2023**

Indicative Experiments

1. **Domestic Electricity Bill Computation** In urban households, electricity consumption is monitored monthly. To promote energy conservation and fair usage, electricity boards apply slab-wise billing based on units consumed. You are tasked to write a C program that helps households to compute their monthly electricity bill based on actual	4 hours

domestic usage. The bill is calculated based on the number of units consumed using a slab system as follows: 0–100 units: Rs. 1.50 per unit 101–300 units: Rs. 2.00 per unit Above 300 units: Rs. 3.00 per unit The code should: Prompt the user to enter the number of units consumed Calculate the bill based on slabs Display the units consumed and the total bill amount Ask the user if they wish to compute another bill Exit gracefully with a thank-you message	
2. **Fuel Cost Estimator for Road Trips** Planning a road trip requires estimating travel expenses, especially fuel costs. With fluctuating fuel prices and varying mileage across vehicles, travelers often struggle to predict their trip cost accurately. To aid this, design a program to estimate fuel cost based on distance, mileage, and fuel price. Write a C program to help a user estimate the total fuel expense for a road trip. The user needs to enter the total distance of the trip (in kilometers), the vehicle's mileage (in km/l), and the current fuel price (in Rs. per litre). The program should: accept trip distance, mileage, and fuel cost as inputs Calculate: Litres Needed = Distance / Mileage Fuel Cost = Litres Needed × Fuel Price Display the calculated values Allow repeated estimation based on user input (Yes/No) Exit gracefully with a thank-you message	4 hours
3. **Student Marks Analyzer for Internal Assessment** A college faculty member wants to automate the process of analyzing internal marks of students for a subject. Instead of manually reviewing each score, a program is needed to accept marks for multiple students, calculate the class average, and identify the highest and lowest scorers. Write a C program that	4 hours

helps an instructor analyze internal marks of students in a subject. The program should: Ask the user to enter the number of students in the class. Accept individual marks (out of 100) for all students Calculate and display: Average marks of the class Highest and lowest marks scored Allow the user to repeat the process for a different class Exit gracefully with a thank-you message	
4. **Smart Grocery Billing System with Limited-Time Offers** A local grocery store is offering limited-time discounts on selected items for a day. The store manager wants a billing assistant that can: Take product names, prices, and quantities for a customer's purchase. Calculate total price and display an item-wise summary Apply a one-time discount only for the first bill generated in a day. Provide a running total for customers. Develop using C program for a smart grocery billing system that performs the following: For each product, input: Product name Quantity Unit price Calculate: Total price for each item Grand total If it's the first customer of the day, apply a 10% discount on the bill. Store and display the summary in a structured format. Repeat the process for the next customer based on user choice Exit with thanks note.	4 hours
5.	4 hours

Health Tracker: BMI Analyzer with Memory-Saving Feature A digital health startup is developing a lightweight health tracking tool to help users calculate and monitor their Body Mass Index (BMI). In order to reduce memory usage and allow efficient access, the system is required to operate with minimal variables and use references wherever it is necessary. You've been asked to simulate the logic of this core functionality. Develop a C program that computes the Body Mass Index (BMI) for multiple individuals and classifies them based on health categories. The program should: Accept the number of users to track For each user: Input name Weight in kilograms Height in meters Compute BMI using the formula: BMI = weight / (height * height) Classify as: BMI < 18.5: Underweight 18.5 ≤ BMI < 25: Normal 25 ≤ BMI < 30: Overweight BMI ≥ 30: Obese	
6. **Conference Registration System with Dynamic Allocation** An academic institute is organizing a national-level technical conference. Participants from various colleges register online, but the number of registrants isn't fixed in advance. To efficiently manage memory, the system needs to allocate space dynamically based on the actual number of registrations. Write a C program to develop a **conference registration system** where: The user inputs the number of participants registering for the event. For each participant, input:	4 hours

Full Name Institution Name Email ID Dynamically allocate memory to store participant details using malloc() or calloc(). After all entries: Display the registered list in a tabular format. Free the dynamically allocated memory after use. Ask whether the user wants to proceed to register a new batch. Exit with a thank-you message.	
7. **Digital Payment Ledger System** A local cooperative society is planning to digitize the ledger for tracking payments made by its members. The system should support entering new transactions, storing them in files, retrieving records, and displaying summaries. Since the number of transactions varies, memory and data handling must be dynamic. Develop a C program that simulates a **digital payment ledger system** with the following features: Accept the number of transactions for the day and dynamically allocate memory. For each transaction, input: Member Name Transaction ID Payment Amount Payment Method (UPI/Cash/Card) Store these entries in a structure using dynamic memory allocation. Display the transactions in a summary table. Write all transactions to a file named ledger.txt. Allow the user to read and display the full ledger history from the file. Ask the user whether to continue or exit.	4 hours
8.	4 hours

Hospital Patient Record Tracker A small clinic wants to keep track of patients' visit information. For different types of patients (in-patient and out-patient), certain information is handled differently. Use structures and unions to design a memory-efficient system. Write a C program to: Use a structure named Patient with common fields: Name Age Gender Patient Type (I for in-patient, O for out-patient) Include a union inside the structure to hold: For in-patients: number of days admitted and room number. For out-patients: consultation fee and doctor's name. Accept n patients' details from the user. Store their information in an array of structures. Display a summary for all patients, handling the union properly.	
9. **Faculty Feedback Archiver System** Your college wants to digitize the storage of semester-wise faculty feedback. Each feedback entry consists of the following information: Faculty Name Course Code Student Registration Number Feedback Score (out of 10) Feedback Comments Each feedback should be written to a text file named feedback.txt. Later, the system should allow retrieval of all feedback for a specific faculty. Write a C program to: Accept the number of feedback entries from the user. Dynamically collect data for each feedback entry using a structure.	4 hours

Write each entry to feedback.txt in append mode. After data entry, ask for a faculty name and return the entire feedback given to that faculty.	
10. **Library Book Management System** A college library wants to maintain its book inventory digitally. The system should allow librarians to add new books, store them in a file, retrieve the complete catalog, and manage dynamic memory allocation as the number of books may vary. The system should also leverage C++ features like classes, constructors, and file streams. Develop a **C++ program** to simulate a **Library Book Management System** with the following features: **Specifications:** Create a class Book with the following attributes: title (string) author (string) ISBN (string) price (float) Dynamically allocate memory for storing multiple books (preferably using vector) Allow the user to: Add new books to the catalog. Display the current book list in tabular format. Choose to continue or exit.	4 hours
11. **Student Admission Record System** An engineering college is developing a software module to automate its student admission process. The system should handle new student entries, display current records, count the number of students admitted so far, and store all details persistently. The system must manage memory dynamically, ensure data encapsulation, and clean up resources properly once the program ends.	4 hours

Develop a **C++ program** for the **Student Admission Record System** with the following features: **Specifications:** Each student record should include: Student Name Admission Number (auto-generated or input) Department Fees Paid Track the **total number of students admitted using friend function** (across program runs in a session). Dynamically store and display student records in a table. Ensure resource cleanup when records are no longer needed **.**	
12. **Library Book Cataloging and Serial Number Generator** A school library is digitizing its book catalog. Every time a new book is added, it should automatically be assigned a unique serial number. The librarian will enter book details, and the system should display all books with their assigned serials. The serial numbers must be generated **internally** and **incremented automatically** starting from 1001. Each book has: Title Author Publisher Year of Publication The system must: Use a **class to represent a book** Assign and manage **serial numbers automatically** using a **static data member** Use dynamic memory allocation inside the **constructors** to initialize book entries Display details of each book added On exit, confirm that memory is released (implicitly show destructor effect)	4 hours

Write a **C++ program** that: Accepts the number of books to enter. For each book, accepts title, author, publisher, and year. Assigns a unique serial number internally starting from 1001. Displays the full catalog. Demonstrates the **constructor and destructor calls** **Expected Features to Implement:** **Static data member** for serial number generation **Parameterized constructor** to initialize book details **Destructor** to display a message when book objects are destroyed Optionally overload a function to display different formats	
13. **Fitness Tracker - Personal Health Metrics System** A fitness center wants to offer its members personal health metrics tracking software. Each member should be able to record their health data like height, weight, and activity level. The system should calculate and display health indicators like BMI (Body Mass Index) or calorie burn. Members may record partial or complete information depending on availability. Develop a C++ program for the Personal Health Metrics System with the following features: For each member, collect: Member Name Height (in meters) Weight (in kg) Activity Level (optional) — Low / Moderate / High The system should: Calculate and display BMI. Optionally calculate the daily calorie burn estimate, if activity level is provided Handle different types of calculations using multiple versions of functions. Allow repeated entry for multiple members.	4 hours
14.	4 hours

Event Registration and Access System A college is organizing a multi-track tech fest. Participants can register as attendees, presenters, or organizers. Each category has different access rights and information to record. The system should manage participant entries, display details based on roles, and provide a report showing individual access rights in the event. Develop a **C++ program** to simulate an **event registration and access system** with the following features: **Specifications:** Create a base class Participant with: Name, Registration ID, and a virtual function showAccess() Derive three classes from it: Attendee: Can attend sessions Presenter: Can present papers Organizer: Can access all areas For each entry, the system should: Accept type of participant Create objects dynamically and store them (polymorphic array/pointer list) Display a report with role-specific access info using **runtime polymorphism**	
15. Digital Certificate Verification System A university is issuing **digital certificates** for different types of achievements — like academic excellence, sports achievements, and community service. The system should manage certificates, verify their authenticity, and prioritize verification based on issuance year or recipient name. Since the certificates vary in type, you'll use **class templates** to generalize the design, **function templates** for comparing and searching, and **STL containers** to store them dynamically. Write a C++ program to design a system with the following requirements:	2 hours

Class Template: Certificate Template parameter T refers to the **type of achievement** (e.g., string for sports, academic discipline, or service type) Data Members: recipientName (string) certificateID (string) categoryInfo (T) → type-specific detail like "CSE", "Basketball", "Blood Donation" issuedYear (int) Member Functions: display() – show certificate details verifyYear(int) – return true if issuedYear matches given year **Function Template: compareCertificates()** Accepts two Certificate objects Returns the one **issued earlier**, or based on name lexicographically **STL Usage** Use std::vector to store certificate entries dynamically Use iterators to: Display all certificates Search by issued year Filter by recipient name	
16. **Bank Loan Eligibility Checker** A cooperative bank is developing a loan eligibility checker. The system collects basic customer details and validates them against a set of criteria. If any of the eligibility rules are violated, the system must throw an appropriate error and display a meaningful message without crashing the program. Develop a C++ program that: Collects: Customer Name Age Monthly Income Requested Loan Amount Validates:	2 hours

Age must be between 21 and 60 Monthly income must be >= ₹15,000 Loan amount must be <= 10 times the monthly income If any rule is violated, throw a specific exception Catch and handle all exceptions gracefully Display success message only for eligible customers Allow processing of multiple customers (say, 3) **Features to Implement:** Use **custom exception classes** (e.g., InvalidAgeException, LowIncomeException, LoanLimitExceeded) Use **try-catch blocks** Input validation inside constructors or dedicated functions Clean and readable class design	
Total Laboratory Hours:	**60 hours**

Reference Books
Stanley Lippman and Josee Lajoie, "**C++ Primer**", Addison Wesley publishers, 5th Edition, **2012** John Horton, " **Learn C++ by Example,** ", Manning Publications, 1st Edition, **2023** Ken Youens-Clark, "**Tiny C Projects**", Manning Publications, 1st Edition, **2021**

Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Seminar	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE105	Data Structures and Algorithms	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. Analyze algorithm efficiency using asymptotic notation to compare the complexities in various problem-solving scenarios.
2. Understand and implement core data structures and their applications in solving computational problems
3. Design and apply appropriate algorithmic strategies and data structures to solve real-world problems using suitable techniques

Course Outcomes

1. Design and analyze efficient algorithms using complexity analysis and formal development techniques
2. Implement and utilize linear and non-linear data structures for effective data management and algorithmic problem-solving
3. Apply various design strategies to solve real-world problems efficiently
4. Analyze and solve complex computational problems using advanced algorithmic techniques and optimization methods
5. Demonstrate the ability to implement and apply fundamental and advanced algorithms using various data structures to solve computational problems

Module:1	Algorithm Analysis	8 hours

Importance of algorithms and data structures - Fundamentals of algorithm analysis: Space and time complexity of an algorithm, Types of asymptotic notations and orders of growth - Algorithm efficiency – best case, worst case, average case - Analysis of non-recursive and recursive algorithms - Asymptotic analysis for recurrence relation: Iteration Method, Substitution Method, Master Method and Recursive Tree Method, Proof of Correctness of the algorithm.

Module:2	Data Structures Linear and Non Linear	14 hours

Arrays: 1D and 2D array- Stack, Queue - Types of Queue: Circular Queue, Double Ended Queue (deQueue), List: Singly linked lists, Doubly linked lists, Circular linked lists. Binary Tree: Definition and Properties - Tree Traversals- Binary Search Trees, Application, Balanced Binary Trees-AVL Tree, Graphs: representation, Traversal: BFS & DFS, Hashing

Module:3	Algorithm Strategies Divide Conquer Back tracking Branch and Bound	7 hours

Divide and Conquer: Quick sort, Merge Sort, Karatsuba faster integer

multiplication algorithm

Backtracking: N-Queens problem, Graph Coloring, 0/1 knapscak Branch & Bound: LIFO-BB and FIFO BB methods: Job Selection problem, Subset Sum		
Module:4	**Algorithm Strategies Greedy and Dynamic Programming**	**7 hours**
Greedy Technique: Huffman Coding, Minimum Spanning Tree: Prims & Kruskal's, Shortest Path : Dijkstra's, Dynamic programming: Matrix Chain Multiplication, Longest Common Subsequence		
Module:5	**Classes of Complexity and Approximation Algorithms**	**7 hours**
The Class P - The Class NP - Reducibility and NP-completeness – SAT (Problem Definition and statement), 3SAT, Independent Set, Clique, Approximation Algorithm – Vertex Cover, Set Cover and Travelling salesman		
	Total Lecture Hours:	**43 hours**

Text Book(s)

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C., "**Introduction to algorithms**", MIT press, 4th Edition, **2022**

Reference Books

Kleinberg, J., & Tardos, E, "**Algorithm design**", Pearson Education India, 1st Edition, **2006**

Mark A. Weiss, "**Data Structures & Algorithm Analysis in C++**", Pearson Education, 4th Edition, **2013**

Horowitz, Sahni and S. Anderson-Freed, "**Fundamentals of Data Structures in C**", Universities Press, 2nd Edition, **2008**

Karumanchi, N, "**Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles**", VISIONIAS, 5th Edition, **2017**

Indicative Experiments

1. Arrays: 1D,2D, functions Tentative Scenario: Assume that you are a game developer designng a multiplayer battle game called " Battle of Legends". This game involves five players in real time. Each player has the following score : Health score (starts from 1000), Magic score (starts from 50) and Status (A-Active, KO-Knocked Out, D-Disconnected). Write a program to to perform / monitor: Health score and Magic score of each player, Update the values during the game play, Display the current statistics.	2 hours

2. Pointers Tentative Scenario: A college auditorium is used for various events like workshops, seminars, and fests. For each event, the seating arrangement may vary in terms of Number of rows, Number of seats per row, Dynamic seat availability (booked vs. empty). The administration needs a flexible program to configure the seat layout dynamically and simulate real-time seat booking and cancellation. Use the pointers to design this dynamic seat management system.	2 hours
3. Stacks and Applications: a) Infix to postfix conversion, b) Expression Evaluation c) Tower of Hanoi Tentative Scenario: In programming languages, expressions and code blocks often use grouping symbols like (), {}, and []. Ensuring these symbols are properly nested and balanced is crucial for: Compiling code correctly, Preventing runtime or syntax errors, Assisting with code auto-completion and formatting in IDEs. This checking is performed during lexical and syntactic analysis in compilers and interpreters — a task that is ideally solved using stacks due to their LIFO (Last-In, First-Out) nature. Write a C program that checks whether all opening and closing brackets in an input expression are correctly matched and properly nested.	2 hours
4. Queue: Bounded Queue, Priority Queue Tentative Scenario: In a busy call center, incoming customer calls are handled in the exact order they arrive to ensure fairness and efficiency. When all agents are occupied, incoming calls wait in a queue until an agent becomes available. Since the call center has a limited number of agents and finite queue capacity, calls may have to wait or sometimes be rejected when the queue is full. Demonstrates the concept of a bounded queue where the queue has a maximum size to implement this scenario.	2 hours

5. Linked list: Polynomial Manipulation Tentative Scenario: In mathematical computing, polynomials are often represented as a series of terms, each consisting of a coefficient and exponent. The number of terms and their order may vary dynamically based on operations like addition, multiplication, or simplification. Create a program to represent two polynomials using singly linked lists and perform addition / multiplication.	4 hours
6. Trees: Binary Search tree: Expression trees, Finding Kth Min and Max element, Binary tree traversals Tentative Scenario: Design a program that parses and analyzes a mathematical expression using both Expression Trees and Binary Search Trees (BSTs). The goal is to build an expression tree from an arithmetic expression, evaluate it, and store all constants in a BST to perform analysis like finding the K-th smallest and largest constants.	2 hours
7. AVL trees – Rotations Tentative Scenario: Imagine you're building a real-time leaderboard system for a competitive coding platform like HackerRank or Codeforces. Thousands of participants submit solutions, and their scores keep changing dynamically. The leaderboard must: Insert new users quickly, Update scores efficiently, Maintain ranking in sorted order, Always provide balanced search performance. A regular binary search tree may become imbalanced after multiple insertions, leading to inefficient lookups (O(n) time).	2 hours

To prevent this, we use AVL Trees — a type of self-balancing binary search tree. Create a program that builds an AVL Tree for storing user scores. Each node stores a user ID and their score. After each insertion, apply necessary rotations to keep the tree height-balanced.	
8. Graphs Tentative Scenario: You are tasked with designing a water-pipeline system for a town. The town layout is represented by an undirected, weighted graph G = (V, E), where: |V| = n represents pipeline junctions, |E| = m represents possible pipe segments, each edge (u, v) ∈ E has a weight w(u, v) indicating the pipe installation cost or distance. Tasks: Traverse the Network (a) Use Breadth-First Search (BFS) starting from a given junction s to determine if all junctions are reachable and record the order of discovery. (b) Use Depth-First Search (DFS) from the same source s to produce a traversal order. (Algorithms should run in O(n + m) time.) (c)Compute Shortest Pipeline Routes Apply Dijkstra's algorithm from source s to calculate the minimum-cost path from s to every other junction. Specify the resulting distance and predecessor arrays. (Assume non-negative edge weights; the time complexity is O(m + n log n).) Design the Overall Pipeline Network Construct a Minimum Spanning Tree (MST) of the graph using Kruskal's or Prim's algorithm.	6 hours

Provide the set of edges chosen and the total installation cost of the MST. (Highlight that MST ensures full connectivity with minimal total weight.)	
9. Use Dynamic Programming to solve Matrix Chain Multiplication, LCS, 0/1 Knapsack problems.	4 hours
10. Demonstrate the following sorting techniques: Quick Sort, Merge Sort, Heap Sort.	2 hours
11. Write a program to solve the subset sum problem using the branch & bound strategy.	2 hours
Total Laboratory Hours:	**30 hours**

Text Book(s)

Horowitz, Sahni and S. Anderson-Freed, "**Fundamentals of Data Structures in C**", Universities Press, 2nd Edition, **2008**

Reference Books

Mark A. Weiss, "**Data Structures & Algorithm Analysis in C++**", Pearson Education, 4th Edition, **2013**

Karumanchi, N, "**Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles**", VISIONIAS, 5th Edition, **2017**

Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Seminar, Presentaion

Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE105	Data Structures and Algorithms	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. Analyze algorithm efficiency using asymptotic notation to compare the complexities in various problem-solving scenarios.
2. Understand and implement core data structures and their applications in solving computational problems
3. Design and apply appropriate algorithmic strategies and data structures to solve real-world problems using suitable techniques

Course Outcomes

1. Design and analyze efficient algorithms using complexity analysis and formal development techniques
2. Implement and utilize linear and non-linear data structures for effective data management and algorithmic problem-solving
3. Apply various design strategies to solve real-world problems efficiently
4. Analyze and solve complex computational problems using advanced algorithmic techniques and optimization methods
5. Demonstrate the ability to implement and apply fundamental and advanced algorithms using various data structures to solve computational problems

Module:1	Algorithm Analysis	8 hours

Importance of algorithms and data structures - Fundamentals of algorithm analysis: Space and time complexity of an algorithm, Types of asymptotic notations and orders of growth - Algorithm efficiency – best case, worst case, average case - Analysis of non-recursive and recursive algorithms - Asymptotic analysis for recurrence relation: Iteration Method, Substitution Method, Master Method and Recursive Tree Method, Proof of Correctness of the algorithm.

Module:2	Data Structures Linear and Non Linear	14 hours

Arrays: 1D and 2D array- Stack, Queue - Types of Queue: Circular Queue, Double Ended Queue (deQueue), List: Singly linked lists, Doubly linked lists, Circular linked lists. Binary Tree: Definition and Properties - Tree Traversals- Binary Search Trees, Application, Balanced Binary Trees-AVL Tree, Graphs: representation, Traversal: BFS & DFS, Hashing

Module:3	Algorithm Strategies Divide Conquer Back tracking Branch and Bound	7 hours

Divide and Conquer: Quick sort, Merge Sort, Karatsuba faster integer

multiplication algorithm

Backtracking: N-Queens problem, Graph Coloring, 0/1 knapscak Branch & Bound: LIFO-BB and FIFO BB methods: Job Selection problem, Subset Sum		
Module:4	**Algorithm Strategies Greedy and Dynamic Programming**	**7 hours**
Greedy Technique: Huffman Coding, Minimum Spanning Tree: Prims & Kruskal's, Shortest Path : Dijkstra's, Dynamic programming: Matrix Chain Multiplication, Longest Common Subsequence		
Module:5	**Classes of Complexity and Approximation Algorithms**	**7 hours**
The Class P - The Class NP - Reducibility and NP-completeness – SAT (Problem Definition and statement), 3SAT, Independent Set, Clique, Approximation Algorithm – Vertex Cover, Set Cover and Travelling salesman		
	Total Lecture Hours:	**43 hours**
Text Book(s)		
Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C., "**Introduction to algorithms**", MIT press, 4th Edition, **2022**		
Reference Books		
Kleinberg, J., & Tardos, E, "**Algorithm design**", Pearson Education India, 1st Edition, **2006** Mark A. Weiss, "**Data Structures & Algorithm Analysis in C++**", Pearson Education, 4th Edition, **2013** Horowitz, Sahni and S. Anderson-Freed, "**Fundamentals of Data Structures in C**", Universities Press, 2nd Edition, **2008** Karumanchi, N, "**Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles**", VISIONIAS, 5th Edition, **2017**		
Indicative Experiments		
1. Arrays: 1D,2D, functions Tentative Scenario: Assume that you are a game developer designng a multiplayer battle game called " Battle of Legends". This game involves five players in real time. Each player has the following score : Health score (starts from 1000), Magic score (starts from 50) and Status (A-Active, KO-Knocked Out, D-Disconnected). Write a program to to perform / monitor: Health score and Magic score of each player, Update the values during the game play, Display the current statistics.		2 hours

2. Pointers Tentative Scenario: A college auditorium is used for various events like workshops, seminars, and fests. For each event, the seating arrangement may vary in terms of Number of rows, Number of seats per row, Dynamic seat availability (booked vs. empty). The administration needs a flexible program to configure the seat layout dynamically and simulate real-time seat booking and cancellation. Use the pointers to design this dynamic seat management system.	2 hours
3. Stacks and Applications: a) Infix to postfix conversion, b) Expression Evaluation c) Tower of Hanoi Tentative Scenario: In programming languages, expressions and code blocks often use grouping symbols like (), {}, and []. Ensuring these symbols are properly nested and balanced is crucial for: Compiling code correctly, Preventing runtime or syntax errors, Assisting with code auto-completion and formatting in IDEs. This checking is performed during lexical and syntactic analysis in compilers and interpreters — a task that is ideally solved using stacks due to their LIFO (Last-In, First-Out) nature. Write a C program that checks whether all opening and closing brackets in an input expression are correctly matched and properly nested.	2 hours
4. Queue: Bounded Queue, Priority Queue Tentative Scenario: In a busy call center, incoming customer calls are handled in the exact order they arrive to ensure fairness and efficiency. When all agents are occupied, incoming calls wait in a queue until an agent becomes available. Since the call center has a limited number of agents and finite queue capacity, calls may have to wait or sometimes be rejected when the queue is full. Demonstrates the concept of a bounded queue where the queue has a maximum size to implement this scenario.	2 hours

5. Linked list: Polynomial Manipulation Tentative Scenario: In mathematical computing, polynomials are often represented as a series of terms, each consisting of a coefficient and exponent. The number of terms and their order may vary dynamically based on operations like addition, multiplication, or simplification. Create a program to represent two polynomials using singly linked lists and perform addition / multiplication.	4 hours
6. Trees: Binary Search tree: Expression trees, Finding Kth Min and Max element, Binary tree traversals Tentative Scenario: Design a program that parses and analyzes a mathematical expression using both Expression Trees and Binary Search Trees (BSTs). The goal is to build an expression tree from an arithmetic expression, evaluate it, and store all constants in a BST to perform analysis like finding the K-th smallest and largest constants.	2 hours
7. AVL trees – Rotations Tentative Scenario: Imagine you're building a real-time leaderboard system for a competitive coding platform like HackerRank or Codeforces. Thousands of participants submit solutions, and their scores keep changing dynamically. The leaderboard must: Insert new users quickly, Update scores efficiently, Maintain ranking in sorted order, Always provide balanced search performance. A regular binary search tree may become imbalanced after multiple insertions, leading to inefficient lookups (O(n) time).	2 hours

To prevent this, we use AVL Trees — a type of self-balancing binary search tree. Create a program that builds an AVL Tree for storing user scores. Each node stores a user ID and their score. After each insertion, apply necessary rotations to keep the tree height-balanced.	
8. Graphs Tentative Scenario: You are tasked with designing a water-pipeline system for a town. The town layout is represented by an undirected, weighted graph G = (V, E), where: |V| = n represents pipeline junctions, |E| = m represents possible pipe segments, each edge (u, v) ∈ E has a weight w(u, v) indicating the pipe installation cost or distance. Tasks: Traverse the Network (a) Use Breadth-First Search (BFS) starting from a given junction s to determine if all junctions are reachable and record the order of discovery. (b) Use Depth-First Search (DFS) from the same source s to produce a traversal order. (Algorithms should run in O(n + m) time.) (c)Compute Shortest Pipeline Routes Apply Dijkstra's algorithm from source s to calculate the minimum-cost path from s to every other junction. Specify the resulting distance and predecessor arrays. (Assume non-negative edge weights; the time complexity is O(m + n log n).) Design the Overall Pipeline Network Construct a Minimum Spanning Tree (MST) of the graph using Kruskal's or Prim's algorithm.	6 hours

Provide the set of edges chosen and the total installation cost of the MST. (Highlight that MST ensures full connectivity with minimal total weight.)	
9. Use Dynamic Programming to solve Matrix Chain Multiplication, LCS, 0/1 Knapsack problems.	4 hours
10. Demonstrate the following sorting techniques: Quick Sort, Merge Sort, Heap Sort.	2 hours
11. Write a program to solve the subset sum problem using the branch & bound strategy.	2 hours
Total Laboratory Hours:	**30 hours**

Text Book(s)

Horowitz, Sahni and S. Anderson-Freed, "**Fundamentals of Data Structures in C**", Universities Press, 2nd Edition, **2008**

Reference Books

Mark A. Weiss, "**Data Structures & Algorithm Analysis in C++**", Pearson Education, 4th Edition, **2013**

Karumanchi, N, "**Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles**", VISIONIAS, 5th Edition, **2017**

Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Seminar, Presentaion

Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE106	Operating Systems	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To introduce the fundamental concepts, design issues, and core functionalities of operating systems.
2. To build skills in managing processes, memory, files, devices, and ensuring protection and security, using algorithms and practical simulations.
3. To introduce modern developments like virtualization, distributed operating systems, and cloud-based systems.

Course Outcomes

1. Understand the structure, functionalities, and foundational concepts of operating systems.
2. Apply process scheduling algorithms and analyze thread models and deadlock-handling methods.
3. Implement synchronization mechanisms and solve concurrency problems in multi-process systems.
4. Evaluate memory management and file system strategies used in operating systems.
5. Analyze device management, protection mechanisms, and virtualization techniques in operating systems.

Module:1	Operating Systems and Structure	6 hours

Operating system basics - History of operating systems - OS structure: Monolithic, Layered, Microkernel, Modular - OS services - System calls and types - Interrupts - OS design issues – Building and booting an OS.

Module:2	Process, Scheduling, Threads and Deadlocks	10 hours

Process concepts and states - Process Scheduling - Operation on Processes: Creation, Termination and Cooperation - Threads and multithreading models - Scheduling criteria - CPU scheduling: Non-preemptive (FCFS, SJF, Priority), Preemptive (Round Robin, SRTF, Preemptive Priority), Multi-level Queue, Multi-level Feedback Queue - Deadlock characterization - Resource Allocation Graphs (RAG) – Deadlock handling methods.

Module:3	Process Synchronization	9 hours

IPC: Shared memory, message passing - Race condition – Critical section problem - Peterson's solution – Bakery Algorithm - Hardware synchronization: Test-and-Set, Swap - Mutex locks - Semaphores – Classical synchronization problems: Bounded buffer, Readers-Writers, Dining Philosophers – Monitors.

Module:4	Memory and File Management	10 hours

Address binding - Logical and physical address space - Contiguous memory allocation - Fragmentation - Placement Algorithms: First Fit, Best Fit, Worst Fit – Paging: Structure with TLB - Segmentation - Virtual memory: Demand paging - Page replacement: FIFO, Optimal, LRU – Thrashing and Working Set - File system concepts - File access methods - Directory structures and implementation - File allocation methods.		
Module:5	**Managing Devices, Security and Protection and Virtualization**	**8 hours**
I/O device management basics: RAID structure, Disk structure, Disk Scheduling Algorithms (FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK), Program and network threats – Cryptography as a security tool – Domains of protection – Access matrix – Capability based systems - Need for virtualization - Virtual machines and architectures – Hypervisors - Virtualization Technologies: Para Virtualization, Full Virtualization - OS-Level Virtualization: Containers, Docker Basics, Containers vs Virtual Machines.		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

Abraham Silberschatz, Peter B. Galvin, Greg Gagne, "**Operating System Concepts**", Wiley, 10th Edition, **2018**

Reference Books

Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau, "**Operating Systems: Three Easy Pieces**", Arpaci Dusseau Books, 1st Edition, **2023**

Andrew S. Tanenbaum, Herbert Bos, "**Modern Operating Systems**", Pearson Education, 4th Edition, **2015**

William Stallings, "**Operating Systems: Internals and Design Principles**", Pearson, 9th Edition, **2018**

Indicative Experiments

1. Study of basic Linux commands and shell scripting	2 hours
2. Implementation of Bootloader	2 hours
3. Process creation using fork(), Orphan and Zombie processes	2 hours
4. Thread Creation	2 hours
5. CPU scheduling algorithms (FCFS, SJF, Priority)	2 hours
6. CPU scheduling algorithms (Round Robin, SRTF, Pre-emptive Priority)	2 hours

7. Banker's Algorithm for Deadlock Avoidance (Safe state checking and additional resource request granting)	2 hours
8. Shared Memory Creation and sharing contents between processes	2 hours
9. Implementation of Producer-Consumer/Readers Writers using Semaphore	2 hours
10. Implementation of Dining Philosophers using Semaphore	2 hours
11. Memory Allocation Strategies: First Fit, Best Fit, Worst Fit	2 hours
12. Page Replacement Algorithms: FIFO, LRU, Optimal	2 hours
13. Disk Scheduling Algorithms: FCFS, SSTF, SCAN	2 hours
14. Disk Scheduling Algorithms: C-SCAN, LOOK, C-LOOK	2 hours
15. File system operations using system calls	2 hours
Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE106	Operating Systems	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To introduce the fundamental concepts, design issues, and core functionalities of operating systems.
2. To build skills in managing processes, memory, files, devices, and ensuring protection and security, using algorithms and practical simulations.
3. To introduce modern developments like virtualization, distributed operating systems, and cloud-based systems.

Course Outcomes

1. Understand the structure, functionalities, and foundational concepts of operating systems.
2. Apply process scheduling algorithms and analyze thread models and deadlock-handling methods.
3. Implement synchronization mechanisms and solve concurrency problems in multi-process systems.
4. Evaluate memory management and file system strategies used in operating systems.
5. Analyze device management, protection mechanisms, and virtualization techniques in operating systems.

Module:1	Operating Systems and Structure	6 hours

Operating system basics - History of operating systems - OS structure: Monolithic, Layered, Microkernel, Modular - OS services - System calls and types - Interrupts - OS design issues – Building and booting an OS.

Module:2	Process, Scheduling, Threads and Deadlocks	10 hours

Process concepts and states - Process Scheduling - Operation on Processes: Creation, Termination and Cooperation - Threads and multithreading models - Scheduling criteria - CPU scheduling: Non-preemptive (FCFS, SJF, Priority), Preemptive (Round Robin, SRTF, Preemptive Priority), Multi-level Queue, Multi-level Feedback Queue - Deadlock characterization - Resource Allocation Graphs (RAG) – Deadlock handling methods.

Module:3	Process Synchronization	9 hours

IPC: Shared memory, message passing - Race condition – Critical section problem - Peterson's solution – Bakery Algorithm - Hardware synchronization: Test-and-Set, Swap - Mutex locks - Semaphores – Classical synchronization problems: Bounded buffer, Readers-Writers, Dining Philosophers – Monitors.

Module:4	Memory and File Management	10 hours

Address binding - Logical and physical address space - Contiguous memory allocation - Fragmentation - Placement Algorithms: First Fit, Best Fit, Worst Fit – Paging: Structure with TLB - Segmentation - Virtual memory: Demand paging - Page replacement: FIFO, Optimal, LRU – Thrashing and Working Set - File system concepts - File access methods - Directory structures and implementation - File allocation methods.		
Module:5	**Managing Devices, Security and Protection and Virtualization**	**8 hours**
I/O device management basics: RAID structure, Disk structure, Disk Scheduling Algorithms (FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK), Program and network threats – Cryptography as a security tool – Domains of protection – Access matrix – Capability based systems - Need for virtualization - Virtual machines and architectures – Hypervisors - Virtualization Technologies: Para Virtualization, Full Virtualization - OS-Level Virtualization: Containers, Docker Basics, Containers vs Virtual Machines.		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

Abraham Silberschatz, Peter B. Galvin, Greg Gagne, "**Operating System Concepts**", Wiley, 10th Edition, **2018**

Reference Books

Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau, "**Operating Systems: Three Easy Pieces**", Arpaci Dusseau Books, 1st Edition, **2023**

Andrew S. Tanenbaum, Herbert Bos, "**Modern Operating Systems**", Pearson Education, 4th Edition, **2015**

William Stallings, "**Operating Systems: Internals and Design Principles**", Pearson, 9th Edition, **2018**

Indicative Experiments

1. Study of basic Linux commands and shell scripting	2 hours
2. Implementation of Bootloader	2 hours
3. Process creation using fork(), Orphan and Zombie processes	2 hours
4. Thread Creation	2 hours
5. CPU scheduling algorithms (FCFS, SJF, Priority)	2 hours
6. CPU scheduling algorithms (Round Robin, SRTF, Pre-emptive Priority)	2 hours

7. Banker's Algorithm for Deadlock Avoidance (Safe state checking and additional resource request granting)	2 hours
8. Shared Memory Creation and sharing contents between processes	2 hours
9. Implementation of Producer-Consumer/Readers Writers using Semaphore	2 hours
10. Implementation of Dining Philosophers using Semaphore	2 hours
11. Memory Allocation Strategies: First Fit, Best Fit, Worst Fit	2 hours
12. Page Replacement Algorithms: FIFO, LRU, Optimal	2 hours
13. Disk Scheduling Algorithms: FCFS, SSTF, SCAN	2 hours
14. Disk Scheduling Algorithms: C-SCAN, LOOK, C-LOOK	2 hours
15. File system operations using system calls	2 hours
Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE201	Models of Computation	3	1	0	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To introduce the fundamental concepts of formal languages, grammars, and automata theory to provide a foundation for the theoretical understanding of computation
2. To explore the design and analysis of finite automata, regular expressions, context-free grammar, pushdown automata, and Turing machines, emphasizing their role in language recognition and parsing
3. To analyze problems related to decidability, reducibility, and undecidability, and to understand the limitations of algorithmic computation

Course Outcomes

1. Apply various proof techniques such as induction, contradiction, and contraposition in the context of formal languages and automata
2. Design and analyze finite automata and regular expressions to solve lexical analysis problems and recognize regular languages
3. Construct and simplify context-free grammar and apply parsing algorithms to determine syntactic structure in languages
4. Develop pushdown automata and understand parsing strategies such as LL(1) and LR for context-free languages
5. Understand and evaluate the computational power and limitations of Turing machines, and distinguish between decidable, semi-decidable, and undecidable problems

Module:1	Foundations of Formal Languages and Proof Techniques	5 hours

Proof techniques: Induction, Contradiction, Contrapositive, Introduction to formal languages and operations - Types of grammars - Chomsky hierarchy- Overview of models of automata

Module:2	Finite Automata and Regular Languages	9 hours

Deterministic and Non-Deterministic Finite Automata (DFA & NFA) - ε-moves in NFA - NFA to DFA conversion - Minimization of DFA - Regular expressions - Lexical Analysis using FA and Regular Expression, Program Constructs using Regular Expression, Equivalence of finite automata and regular expressions - Regular languages - Pumping Lemma

Module:3	Context Free Grammars and Languages	9 hours

Context-Free Grammars (CFG) - Context-Free Languages - Derivations – Ambiguous and Unambiguous grammars - First and Follow - Left Recursion and Left Factoring – Simplification of CFG: Chomsky Normal Form – Greibach Normal Form - CYK Algorithm		
Module:4	**Parsing and Pushdown Automata**	**11 hours**
Introduction to parsing: Top-down and Bottom-up - LL (1) Parsing - SLR Parsing - Pushdown Automata (PDA) - Deterministic and Non-deterministic PDA - Interpretation of syntactic statements using PDA.		
Module:5	**Turing Machines and Undecidability**	**9 hours**
Turing Machines (acceptors and transducers) - Universal Turing machine and encoding - Church Turing Thesis - Recursive Enumerable Languages -Recursively Enumerable Languages -Non Recursive Languages -Reducibility - Undecidability: Halting Problem - Post Correspondence Problem – Tools: JFLAP, REGEX, LEX, YACC		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**
	Tutorial Hours:	**15 hours**

Text Book(s)

John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, "**Introduction to Automata Theory, Languages, and Computation**", Pearson Education, 3rd Edition, **2020**

Reference Books

Peter Linz, Susan H. Rodger, Jones, "**An Introduction to Formal Languages and Automata**", Bartlett Publishers, 7th Edition, **2022**

Dan A. Simovici, "**Introduction to the Theory of Formal Languages**", World Scientific, **2024**

Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, Sorav Bansal, "**Compilers: Principles, Techniques, and Tools, Updated**", Pearson India, 2nd Edition, **2006**

Indicative Experiments

Total Laboratory Hours:	**0 hours**

Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Seminar

Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE202	Database Systems	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To familiarize with database concepts and entity relationship modeling.
2. To impart knowledge on relational database design, normalization, physical database design, query optimization and distributed databases.
3. To equip with the strategies for handling transactions, concurrency control and database recovery.

Course Outcomes

1. Understand the role of database management systems in organizations and design relational data models and their operations.
2. Analyze and apply indexing structures for efficient data access and query optimization.
3. Gain insight into transaction management, concurrency control, and recovery mechanisms in databases.
4. Demonstrate the knowledge on distributed databases and NoSQL databases.
5. Apply database design principles to create scalable and maintainable data models for real-world systems.

Module:1	Database Systems and Data Models	7 hours

Introduction to database systems - Characteristics of database approach - Advantages of using DBMS approach - Actors on the scene - Classification of DBMS - Data Models - Schemas and Instances - Three-Schema Architecture - Database System Environment - Introduction to relational, semi structured, graph, vector, time series, distributed and unstructured databases - Relational model concepts: characteristics of relations - Relational model constraints and relational database schemas: domain constraints - key constraints and constraints on null values - entity integrity, referential integrity, and foreign Keys - dealing with constraint violations.

Module:2	ER Modeling and Relational Database Design	10 hours

Entity Relationship Model: Types of attributes, relationship types and sets, roles, structural constraints, weak entity -Mapping ER model to a relational schema - Extended ER Model: generalization - specialization – aggregations Guidelines for relational schema - Functional dependencies: inference rules, equivalence, and minimal cover - axioms on functional dependencies - Normalization: first, second and third normal forms - boyce codd normal form, multi-valued dependency and fourth normal form - join dependency and fifth normal form.

Module:3	Physical Database Design and Query Processing	9 hours

File structures: operations on files - files of unordered and ordered records - hashing techniques: static and dynamic - indexing: single level indexing, multi-level indexing, dynamic multilevel indexing - B+ tree indexing – LSM trees and its variants - Relational Algebra: operators - translating SQL queries into relational algebra - Tuple Relational Calculus - Query Optimization: query trees and heuristic optimization of query trees.		
Module:4	**Transaction Processing, Concurrency Control and Database Recovery**	**10 hours**
Introduction to transaction processing and concepts: ACID properties of transactions, transaction states - serial and serializable schedules - schedules based on recoverability - schedules based on serializability - conflict serializabilty - concurrent transactions – need of concurrency control - Concurrency control techniques: two phase locking techniques – guaranteeing serializability by two-phase locking – dealing with deadlock and starvation – concurrency control based on timestamp ordering: timestamp ordering algorithm – granularity of data items and multiple granularity locking - Recovery Concepts: categorization of recovery algorithms - recovery based on deferred update and immediate update.		
Module:5	**Distributed and NOSQL Databases**	**7 hours**
Distributed Database: introduction - architectures - data fragmentation, replication, and allocation techniques - transaction management, concurrency control and recovery - query processing and optimization - NoSQL Databases: introduction - characteristics of NoSQL Systems, CAP theorem, ACID vs BASE properties, NoSQL modeling techniques - aggregate data models, distribution models, map reduce, types of NoSQL databases: key-value databases, document based databases, column based databases and graph databases.		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

R. Elmasri and S. B. Navathe, "**Fundamentals of Database Systems**", Pearson India Education, 7th Edition, **2021**

Reference Books

A. Silberschatz, H. F. Korth and S. Sudarshan, "**Database System Concepts**", McGraw Hill India, 7th Edition, **2021**

Mark L. Gillenson and Pradeep Singh, "**Fundamentals of Database Management Systems - An Indian Adaptation**", Wiley, India, 3rd Edition, **2025**

M. Tamer Özsu , Patrick Valduriez, "**Principles of Distributed Database Systems**", Springer Nature, 4th Edition, **2020**

Gerardus Blokdyk, "**NoSQL Databases A Complete Guide**", 5STARCooks, 1st Edition, **2021**

Indicative Experiments

1. Design an ER / EER diagram for any real-world data requirements (E.g.: Bank Database) using any suitable drawing tool.	2 hours
2. Translate the ER diagram into the corresponding relational databases.	2 hours
3. Apply the Normalization concept and normalize the given table into 1NF, 2NF and 3NF.	2 hours
4. Populate the relational database with real world Customer and Transaction data (refer to Kaggle or any open data sources for real world datasets) using Data Manipulation Languages. Use Data Control Language and Transaction Control Language commands.	2 hours
5. Apply retrieval of data using operators, functions like arithmetic, comparison, logical and like Operators.	2 hours
6. Apply the single row functions and group functions on the normalized tables.	2 hours
7. Retrieve the data from the relational database using sub queries, Joins and Views.	4 hours
8. Use features of PL/SQL like control structures, conditional statements, user defined functions and procedures, cursors and triggers to retrieve or modify data.	6 hours
9. Interface the database with Front Ends using ODBC and JDBC. Perform the retrieval and DML operations from the Front End.	2 hours
10. Implementation of database creation, deletion, insertion, updation, view and retrieval of documents in a NoSQL database environment. Implement a scale in / scale out operation on NoSQL database.	2 hours
11. Implementation of database replication, backup and restore of files in a NoSQL database environment. Demonstrate how data is shared in NoSQL environment.	2 hours
12. Create and configure cloud resources, then perform CRUD operations. Integrate the database with a sample application to demonstrate real-world usage.	2 hours
Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE202	Database Systems	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To familiarize with database concepts and entity relationship modeling.
2. To impart knowledge on relational database design, normalization, physical database design, query optimization and distributed databases.
3. To equip with the strategies for handling transactions, concurrency control and database recovery.

Course Outcomes

1. Understand the role of database management systems in organizations and design relational data models and their operations.
2. Analyze and apply indexing structures for efficient data access and query optimization.
3. Gain insight into transaction management, concurrency control, and recovery mechanisms in databases.
4. Demonstrate the knowledge on distributed databases and NoSQL databases.
5. Apply database design principles to create scalable and maintainable data models for real-world systems.

Module:1	Database Systems and Data Models	7 hours

Introduction to database systems - Characteristics of database approach - Advantages of using DBMS approach - Actors on the scene - Classification of DBMS - Data Models - Schemas and Instances - Three-Schema Architecture - Database System Environment - Introduction to relational, semi structured, graph, vector, time series, distributed and unstructured databases - Relational model concepts: characteristics of relations - Relational model constraints and relational database schemas: domain constraints - key constraints and constraints on null values - entity integrity, referential integrity, and foreign Keys - dealing with constraint violations.

Module:2	ER Modeling and Relational Database Design	10 hours

Entity Relationship Model: Types of attributes, relationship types and sets, roles, structural constraints, weak entity -Mapping ER model to a relational schema - Extended ER Model: generalization - specialization – aggregations Guidelines for relational schema - Functional dependencies: inference rules, equivalence, and minimal cover - axioms on functional dependencies - Normalization: first, second and third normal forms - boyce codd normal form, multi-valued dependency and fourth normal form - join dependency and fifth normal form.

Module:3	Physical Database Design and Query Processing	9 hours

File structures: operations on files - files of unordered and ordered records - hashing techniques: static and dynamic - indexing: single level indexing, multi-level indexing, dynamic multilevel indexing - B+ tree indexing – LSM trees and its variants - Relational Algebra: operators - translating SQL queries into relational algebra - Tuple Relational Calculus - Query Optimization: query trees and heuristic optimization of query trees.		
Module:4	**Transaction Processing, Concurrency Control and Database Recovery**	**10 hours**
Introduction to transaction processing and concepts: ACID properties of transactions, transaction states - serial and serializable schedules - schedules based on recoverability - schedules based on serializability - conflict serializabilty - concurrent transactions – need of concurrency control - Concurrency control techniques: two phase locking techniques – guaranteeing serializability by two-phase locking – dealing with deadlock and starvation – concurrency control based on timestamp ordering: timestamp ordering algorithm – granularity of data items and multiple granularity locking - Recovery Concepts: categorization of recovery algorithms - recovery based on deferred update and immediate update.		
Module:5	**Distributed and NOSQL Databases**	**7 hours**
Distributed Database: introduction - architectures - data fragmentation, replication, and allocation techniques - transaction management, concurrency control and recovery - query processing and optimization - NoSQL Databases: introduction - characteristics of NoSQL Systems, CAP theorem, ACID vs BASE properties, NoSQL modeling techniques - aggregate data models, distribution models, map reduce, types of NoSQL databases: key-value databases, document based databases, column based databases and graph databases.		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

R. Elmasri and S. B. Navathe, "**Fundamentals of Database Systems**", Pearson India Education, 7th Edition, **2021**

Reference Books

A. Silberschatz, H. F. Korth and S. Sudarshan, "**Database System Concepts**", McGraw Hill India, 7th Edition, **2021**

Mark L. Gillenson and Pradeep Singh, "**Fundamentals of Database Management Systems - An Indian Adaptation**", Wiley, India, 3rd Edition, **2025**

M. Tamer Özsu , Patrick Valduriez, "**Principles of Distributed Database Systems**", Springer Nature, 4th Edition, **2020**

Gerardus Blokdyk, "**NoSQL Databases A Complete Guide**", 5STARCooks, 1st Edition, **2021**

Indicative Experiments

1. Design an ER / EER diagram for any real-world data requirements (E.g.: Bank Database) using any suitable drawing tool.	2 hours
2. Translate the ER diagram into the corresponding relational databases.	2 hours
3. Apply the Normalization concept and normalize the given table into 1NF, 2NF and 3NF.	2 hours
4. Populate the relational database with real world Customer and Transaction data (refer to Kaggle or any open data sources for real world datasets) using Data Manipulation Languages. Use Data Control Language and Transaction Control Language commands.	2 hours
5. Apply retrieval of data using operators, functions like arithmetic, comparison, logical and like Operators.	2 hours
6. Apply the single row functions and group functions on the normalized tables.	2 hours
7. Retrieve the data from the relational database using sub queries, Joins and Views.	4 hours
8. Use features of PL/SQL like control structures, conditional statements, user defined functions and procedures, cursors and triggers to retrieve or modify data.	6 hours
9. Interface the database with Front Ends using ODBC and JDBC. Perform the retrieval and DML operations from the Front End.	2 hours
10. Implementation of database creation, deletion, insertion, updation, view and retrieval of documents in a NoSQL database environment. Implement a scale in / scale out operation on NoSQL database.	2 hours
11. Implementation of database replication, backup and restore of files in a NoSQL database environment. Demonstrate how data is shared in NoSQL environment.	2 hours
12. Create and configure cloud resources, then perform CRUD operations. Integrate the database with a sample application to demonstrate real-world usage.	2 hours
Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE203	Computer Networks	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To foster students' understanding of the core concepts of computer networking, including protocols, architectures, and their applications.
2. To empower students with the knowledge to design, implement, and analyze the performance of OSI and TCP/IP architectures effectively.
3. To identify appropriate application layer protocols for specific applications and their corresponding security mechanisms.

Course Outcomes

1. Interpret the different building blocks of networks.
2. Analyze the error and flow control mechanisms for connecting a network.
3. Design and configure efficient networks using switching, IP, routing, NAT multicast, and IPv6, to optimize scalability and performance
4. Examine the essential factors for reliable transport layer services, flow control, connection management and analyze the consequences of congestion on their performance.
5. Assess the significance of application layer services and their associated protocols for seamless network communication and functionality

Module:1	Networking Foundations	8 hours

Layering and Protocols-Network Topology - Encapsulation -Multiplexing and De-multiplexing - OSI Model, TCP/IP model- Basics of Physical Layer-Internet Architecture-Application Programming Interface (Sockets) - Performance Metrics (Throughput, Bandwidth, Delay and Latency) - High-Speed Networks.

Module:2	Direct Links	9 hours

Connecting to a Network - Encoding – Framing: Byte-Oriented Protocols (PPP), Bit-Oriented Protocols (HDLC); Ethernet (IEEE 802.3); Error Detection: Internet Checksum Algorithm, Cyclic Redundancy Check –Error Correction- Hamming Code; Reliable Transmission: Stop-and-Wait, Sliding Window - Multiple-access Networks: Access Protocol, Wireless Networks: 802.11/Wi-Fi, Bluetooth (802.15.1) - Access Networks, Cellular Network.

Module:3	Internetworking	10 hours

Switching Basics – Datagrams, Virtual Circuit Switching, Virtual LANs (VLANs); Internet (IP) – Internetwork, Service Model, Global Addresses, Datagram Forwarding in IP, Classful addressing, Subnetting and Classless Addressing, Address Resolution Protocol (ARP), Dynamic Host Configuration Protocol (DHCP), Error Reporting-ICMP, Network Address Translation (NAT), Routing - Network as a Graph, Distance-Vector-RIP, Link State-OSPF, Metrics, Implementation-Performance Analysis- Packet Tracer; Global Internet - Routing

Areas, Interdomain Routing -BGP; IP Version 6 - Addresses and Routing, Packet Format, Advanced Capabilities; Multicast - Multicast Addresses, Multicast Routing -DVMRP		
Module:4	**End to End Protocols**	**8 hours**
Simple Demultiplexer (UDP), Reliable Byte Stream (TCP) - End-to-End Issues, Segment Format, Connection Establishment and Termination, Sliding Window, Triggering Transmission, Adaptive Retransmission, Record Boundaries, TCP Extensions, Performance, Alternative Design Choices-SCTP. Remote Procedure Call (RPC)- Fundamentals, Quality of Service -Allocating Resources - Issues in Resource Allocation; Queuing Disciplines - FIFO, Fair Queuing; TCP Congestion Control; Advanced Congestion Control, Quality of Service		
Module:5	**End to End Data**	**8 hours**
Security Attacks - Trust and Threats, IP Security (IPsec), Firewalls; Applications - Electronic Mail (SMTP, MIME, POP3), World Wide Web (HTTP), FTP, Telnet, Infrastructure Applications-DNS, SNMP; Software Defined Network (SDN)-introduction.		
Module:6	**Contemporary Issues**	**2 hours**
Industry practices like Netconf and Telemetry		
	Total Lecture Hours:	**45 hours**

Text Book(s)

Larry L. Peterson, Bruce S. Davie, "**Computer Networks : A Systems Approach**", The Morgan Kaufmann Series in Networking, 6th Edition, **2021**

Reference Books

Behrouz A. Forouzan, "**Data communication and Networking**", McGraw Hill Education, 6th Edition, **2022**

James F. Kurose and Keith W.Ross, "**Computer Networking: A Top-Down Approach**", Pearson Education, 9th Edition, **2025**

William Stallings, "**Data and Computer Communication**", Pearson India Education Services Pvt Ltd, 10th Edition, **2022**

Indicative Experiments

1. Study of Basic Network Commands, Demonstration of all networking hardware and their functionalities.	2 hours
2. Error detection and Correction mechanisms (CRC, Checksum, Hamming Code)	2 hours
3. Flow control mechanisms (Stop and Wait, Sliding Window)	4 hours
4. IP Addressing and Subnetting	2 hours
5. Cisco Packet Tracer Simulation – LAN, Static routing, VLANs	4 hours
6. Socket programming – UDP, TCP, multi-client chatting	4 hours

7. Adaptive Retransmission- Nagle's Algorithm, Karn's Patridge Algorithm, Jacobson/Karels algorithm	4 hours
8. Remote Procedure Call	2 hours
9. Hands on WireShark - Capture filter, Display filter, simple attacks	2 hours
10. Implementing basic SDN topology with Mininet	4 hours
Total Laboratory Hours:	**30 hours**

Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Seminar, Group Discussion, Presentaion	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE203	Computer Networks	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To foster students' understanding of the core concepts of computer networking, including protocols, architectures, and their applications.
2. To empower students with the knowledge to design, implement, and analyze the performance of OSI and TCP/IP architectures effectively.
3. To identify appropriate application layer protocols for specific applications and their corresponding security mechanisms.

Course Outcomes

1. Interpret the different building blocks of networks.
2. Analyze the error and flow control mechanisms for connecting a network.
3. Design and configure efficient networks using switching, IP, routing, NAT multicast, and IPv6, to optimize scalability and performance
4. Examine the essential factors for reliable transport layer services, flow control, connection management and analyze the consequences of congestion on their performance.
5. Assess the significance of application layer services and their associated protocols for seamless network communication and functionality

Module:1	Networking Foundations	8 hours

Layering and Protocols-Network Topology - Encapsulation -Multiplexing and De-multiplexing - OSI Model, TCP/IP model- Basics of Physical Layer-Internet Architecture-Application Programming Interface (Sockets) - Performance Metrics (Throughput, Bandwidth, Delay and Latency) - High-Speed Networks.

Module:2	Direct Links	9 hours

Connecting to a Network - Encoding – Framing: Byte-Oriented Protocols (PPP), Bit-Oriented Protocols (HDLC); Ethernet (IEEE 802.3); Error Detection: Internet Checksum Algorithm, Cyclic Redundancy Check –Error Correction- Hamming Code; Reliable Transmission: Stop-and-Wait, Sliding Window - Multiple-access Networks: Access Protocol, Wireless Networks: 802.11/Wi-Fi, Bluetooth (802.15.1) - Access Networks, Cellular Network.

Module:3	Internetworking	10 hours

Switching Basics – Datagrams, Virtual Circuit Switching, Virtual LANs (VLANs); Internet (IP) – Internetwork, Service Model, Global Addresses, Datagram Forwarding in IP, Classful addressing, Subnetting and Classless Addressing, Address Resolution Protocol (ARP), Dynamic Host Configuration Protocol (DHCP), Error Reporting-ICMP, Network Address Translation (NAT), Routing - Network as a Graph, Distance-Vector-RIP, Link State-OSPF, Metrics, Implementation-Performance Analysis- Packet Tracer; Global Internet - Routing

Areas, Interdomain Routing -BGP; IP Version 6 - Addresses and Routing, Packet Format, Advanced Capabilities; Multicast - Multicast Addresses, Multicast Routing -DVMRP		
Module:4	**End to End Protocols**	**8 hours**
Simple Demultiplexer (UDP), Reliable Byte Stream (TCP) - End-to-End Issues, Segment Format, Connection Establishment and Termination, Sliding Window, Triggering Transmission, Adaptive Retransmission, Record Boundaries, TCP Extensions, Performance, Alternative Design Choices-SCTP. Remote Procedure Call (RPC)- Fundamentals, Quality of Service -Allocating Resources - Issues in Resource Allocation; Queuing Disciplines - FIFO, Fair Queuing; TCP Congestion Control; Advanced Congestion Control, Quality of Service		
Module:5	**End to End Data**	**8 hours**
Security Attacks - Trust and Threats, IP Security (IPsec), Firewalls; Applications - Electronic Mail (SMTP, MIME, POP3), World Wide Web (HTTP), FTP, Telnet, Infrastructure Applications-DNS, SNMP; Software Defined Network (SDN)-introduction.		
Module:6	**Contemporary Issues**	**2 hours**
Industry practices like Netconf and Telemetry		
	Total Lecture Hours:	**45 hours**

Text Book(s)

Larry L. Peterson, Bruce S. Davie, "**Computer Networks : A Systems Approach**", The Morgan Kaufmann Series in Networking, 6th Edition, **2021**

Reference Books

Behrouz A. Forouzan, "**Data communication and Networking**", McGraw Hill Education, 6th Edition, **2022**

James F. Kurose and Keith W.Ross, "**Computer Networking: A Top-Down Approach**", Pearson Education, 9th Edition, **2025**

William Stallings, "**Data and Computer Communication**", Pearson India Education Services Pvt Ltd, 10th Edition, **2022**

Indicative Experiments

1. Study of Basic Network Commands, Demonstration of all networking hardware and their functionalities.	2 hours
2. Error detection and Correction mechanisms (CRC, Checksum, Hamming Code)	2 hours
3. Flow control mechanisms (Stop and Wait, Sliding Window)	4 hours
4. IP Addressing and Subnetting	2 hours
5. Cisco Packet Tracer Simulation – LAN, Static routing, VLANs	4 hours
6. Socket programming – UDP, TCP, multi-client chatting	4 hours

7. Adaptive Retransmission- Nagle's Algorithm, Karn's Patridge Algorithm, Jacobson/Karels algorithm	4 hours
8. Remote Procedure Call	2 hours
9. Hands on WireShark - Capture filter, Display filter, simple attacks	2 hours
10. Implementing basic SDN topology with Mininet	4 hours
Total Laboratory Hours:	**30 hours**

Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Seminar, Group Discussion, Presentaion	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE204	Software Engineering	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To provide a comprehensive overview of software engineering principles, lifecycle models, and development methodologies for systematic software design and implementation.
2. To impart knowledge on project management frameworks and applying best strategies for planning, executing, and controlling software projects.
3. To apply effective testing and maintenance techniques and implement quality assurance practices for developing reliable software.

Course Outcomes

1. Demonstrate the ability to select appropriate software development methodologies for different project requirements.
2. Develop a comprehensive software project plan incorporating key project management activities.
3. Ability to model system requirements, design and apply testing strategies to ensure software reliability.
4. Comprehend the software maintenance practices and quality management techniques that support the long-term sustainability of software products.
5. Implement software development life cycle for real time projects.

Module:1	Introduction	5 hours

Nature of Software, Software Engineering. Software process, Software Engineering Practice, Generic Process Model, Perspective Process models, Specialized Process Model, Unified Process, Introduction to Agility - Agile Process-Extreme programming, Scrum, Other Agile Process Models ,Agile Tools, System Engineering.

Module:2	Software Project Management	10 hours

The Management Spectrum, People, Product, Process, Project, Process and Project metrics, Estimation for software project- Project Planning Process- Milestones and Deliverables, Software Scope and Feasibility, Resources, Decomposition Techniques- Work breakdown structure, Empirical Estimation Models, Project Scheduling, RISK management, CASE Tools.

Module:3	Requirements Modeling and Design	12 hours

Requirement Engineering process - Eliciting Requirements - Functional and Non Functional Requirements - Requirements Traceability Matrix - Validating Requirements - Requirement Analysis and Modeling: Data Modeling, Flow Oriented Modeling - Transaction, Transformation, Behavioral Modeling - Structured Language Specification - Formal Specification - Design concepts and principles,

Modularity: Cohesion, coupling, Architectural design, Architectural styles, Detailed Design- Object-oriented analysis and design- Design patterns, User Interface Design.		
Module:4	**Software Testing and Metrics**	**8 hours**
Software Implementation guidelines – Testing - Unit Testing - Integration Testing - Validation Testing - System Testing - Acceptance Testing and its Types – Static testing – Dynamic testing – Black Box Testing – Boundary value analysis, Equivalence partitioning- Decision Tables - White Box Testing – Structural Testing, Computation of Logical Complexity – Cyclomatic Complexity- Testing Mobile applications – Testing Web applications - Regression Testing – Debugging - Automation tools-Selenium, Appium- Software Metrics – Requirements Metrics - OO design metrics – Testing and SCM metrics		
Module:5	**Maintenance and Quality Management**	**8 hours**
Software Maintenance, Types of Maintenance, Re-Engineering, Reverse Engineering, Software Configuration Management (SCM) Overview, SCM Process, Quality management – Software quality, Software quality assurance- Elements of software quality assurance, Statistical software quality assurance, Software reliability- Process improvement models - CMM and CMMI.		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**
Text Book(s)		
Roger S. Pressman and Bruce R. Maxim, "**Software Engineering: A Practitioner's Approach**", McGraw Hill Education, 9th Edition, **2023**		
Reference Books		
Rajib Mall, "**Fundamentals of software engineering**", PHI Learning Pvt Ltd, 5th Edition, **2018** Ian Somerville, "**Software Engineering**", Addison Wesley, 10th Edition, **2017** William E. Lewis, "**Software Testing and Continuous Quality Improvement**", Auerbach Publications, 3rd Edition, **2017**		
Indicative Experiments		
1. Analysis and identification of a suitable process model based on a selected business application. Sample Scenario: A university intends to develop an online examination platform to facilitate internal assessments and semester-end exams. The system should include features such as secure user authentication, randomized question paper generation, automated grading for objective-type questions, manual evaluation support for subjective		3 hours

responses, and comprehensive report generation. The project will be piloted with a small user group prior to full deployment. The development timeline is five months, with built-in flexibility to incorporate feedback during testing. Ongoing collaboration with faculty and IT staff is expected throughout the development life cycle.	
2. Develop a Work Breakdown Structure (WBS) using Process-Based, Product-Based, Geographical-Based and Role-Based approaches for a business application of your choice. Sample Scenario: A startup plans to launch a food delivery platform similar to Swiggy or Zomato. The app should allow users to browse restaurants, place orders, make payments, and track deliveries in real time. It must also support restaurant onboarding, delivery partner management, and customer feedback. The system should be scalable and secure, with a user-friendly interface.	3 hours
3. Requirement modeling using Entity Relationship Diagram (Structural Modeling) Sample Scenario: A company wants to develop an online bookstore where users can browse books, place orders, and make payments. The system should also allow authors and publishers to manage their book listings. The bookstore must track inventory, customer details, and order history	3 hours
4. Requirement modeling using Context flow diagram, DFD (Functional Modeling) Sample Scenario: A restaurant chain wants to develop an online food ordering system where customers can browse menus, place orders, and make payments. The system should also allow restaurant staff to manage orders and delivery personnel to track assigned deliveries.	3 hours
5. Requirement modeling using State Transition Diagram (Behavioral Modeling) Sample Scenario : Develop a state machine diagram for an e-commerce application where customers can create service requests such as querying goods, purchasing items, and managing accounts (open, close, deposit, receive payment, reopen, resume). Account operations follow specific rules: opening requires an initial deposit, purchases are restricted by balance limits, and accounts can be suspended or permanently closed based on overdue payments. Design the state transitions based on these conditions.	3 hours
6. Story Boarding and User Interface design Modeling Sample Scenario	3 hours

You are tasked with designing a mobile banking application. Create a storyboard depicting the process of a user logging in, viewing their balance, transferring money, and receiving confirmation. Accompany your storyboard with user interface mockups for each step.	
7. OO design – Use case Model, Class Model Sample Scenario A university wants to develop an online platform where students can register for courses each semester. The system should allow students to log in, view available courses, register or drop courses, and view their schedules. Faculty members should be able to manage course offerings, approve enrollments, and communicate with students. Administrators will oversee the system, manage user roles, and generate reports.	3 hours
8. OO design – Interaction Models Sample Scenario A healthcare provider plans to develop an online appointment booking system that allows patients to register, search for doctors by specialty, book appointments, receive confirmations, and reschedule or cancel them as needed. Doctors should be able to manage availability, view schedules, and approve or reject appointments. The system must also notify patients via email or SMS. Design a UML interaction model that captures key interactions—such as 'Book Appointment' and 'Cancel Appointment'—involving users, system components, and external services.	3 hours
9. OO design – Package, Component and deployment models Sample Scenario A national bank is developing an online banking platform that allows customers to perform transactions such as checking balances, transferring funds, paying bills, and managing accounts. The system must also support admin operations like fraud detection, user management, and audit logging. It should be accessible via web and mobile apps, with secure backend services and integration with third-party payment gateways.	3 hours
10. Design and demonstration of test cases. Functional Testing and Non-Functional Testing Sample Scenario A national railway service is developing an online ticket booking platform. The system allows users to search for trains, check seat availability, book tickets, make payments, and receive e-tickets. It also supports cancellation, refund processing, and real-time updates on train status. The platform must be accessible via web and mobile, and should handle high traffic during peak hours.	3 hours

Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Seminar, Presentaion	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE204	Software Engineering	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To provide a comprehensive overview of software engineering principles, lifecycle models, and development methodologies for systematic software design and implementation.
2. To impart knowledge on project management frameworks and applying best strategies for planning, executing, and controlling software projects.
3. To apply effective testing and maintenance techniques and implement quality assurance practices for developing reliable software.

Course Outcomes

1. Demonstrate the ability to select appropriate software development methodologies for different project requirements.
2. Develop a comprehensive software project plan incorporating key project management activities.
3. Ability to model system requirements, design and apply testing strategies to ensure software reliability.
4. Comprehend the software maintenance practices and quality management techniques that support the long-term sustainability of software products.
5. Implement software development life cycle for real time projects.

Module:1	Introduction	5 hours

Nature of Software, Software Engineering. Software process, Software Engineering Practice, Generic Process Model, Perspective Process models, Specialized Process Model, Unified Process, Introduction to Agility - Agile Process-Extreme programming, Scrum, Other Agile Process Models ,Agile Tools, System Engineering.

Module:2	Software Project Management	10 hours

The Management Spectrum, People, Product, Process, Project, Process and Project metrics, Estimation for software project- Project Planning Process- Milestones and Deliverables, Software Scope and Feasibility, Resources, Decomposition Techniques- Work breakdown structure, Empirical Estimation Models, Project Scheduling, RISK management, CASE Tools.

Module:3	Requirements Modeling and Design	12 hours

Requirement Engineering process - Eliciting Requirements - Functional and Non Functional Requirements - Requirements Traceability Matrix - Validating Requirements - Requirement Analysis and Modeling: Data Modeling, Flow Oriented Modeling - Transaction, Transformation, Behavioral Modeling - Structured Language Specification - Formal Specification - Design concepts and principles,

Modularity: Cohesion, coupling, Architectural design, Architectural styles, Detailed Design- Object-oriented analysis and design- Design patterns, User Interface Design.		
Module:4	**Software Testing and Metrics**	**8 hours**
Software Implementation guidelines – Testing - Unit Testing - Integration Testing - Validation Testing - System Testing - Acceptance Testing and its Types – Static testing – Dynamic testing – Black Box Testing – Boundary value analysis, Equivalence partitioning- Decision Tables - White Box Testing – Structural Testing, Computation of Logical Complexity – Cyclomatic Complexity- Testing Mobile applications – Testing Web applications - Regression Testing – Debugging - Automation tools-Selenium, Appium- Software Metrics – Requirements Metrics - OO design metrics – Testing and SCM metrics		
Module:5	**Maintenance and Quality Management**	**8 hours**
Software Maintenance, Types of Maintenance, Re-Engineering, Reverse Engineering, Software Configuration Management (SCM) Overview, SCM Process, Quality management – Software quality, Software quality assurance- Elements of software quality assurance, Statistical software quality assurance, Software reliability- Process improvement models - CMM and CMMI.		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**
Text Book(s)		
Roger S. Pressman and Bruce R. Maxim, "**Software Engineering: A Practitioner's Approach**", McGraw Hill Education, 9th Edition, **2023**		
Reference Books		
Rajib Mall, "**Fundamentals of software engineering**", PHI Learning Pvt Ltd, 5th Edition, **2018** Ian Somerville, "**Software Engineering**", Addison Wesley, 10th Edition, **2017** William E. Lewis, "**Software Testing and Continuous Quality Improvement**", Auerbach Publications, 3rd Edition, **2017**		
Indicative Experiments		
1. Analysis and identification of a suitable process model based on a selected business application. Sample Scenario: A university intends to develop an online examination platform to facilitate internal assessments and semester-end exams. The system should include features such as secure user authentication, randomized question paper generation, automated grading for objective-type questions, manual evaluation support for subjective		3 hours

responses, and comprehensive report generation. The project will be piloted with a small user group prior to full deployment. The development timeline is five months, with built-in flexibility to incorporate feedback during testing. Ongoing collaboration with faculty and IT staff is expected throughout the development life cycle.	
2. Develop a Work Breakdown Structure (WBS) using Process-Based, Product-Based, Geographical-Based and Role-Based approaches for a business application of your choice. Sample Scenario: A startup plans to launch a food delivery platform similar to Swiggy or Zomato. The app should allow users to browse restaurants, place orders, make payments, and track deliveries in real time. It must also support restaurant onboarding, delivery partner management, and customer feedback. The system should be scalable and secure, with a user-friendly interface.	3 hours
3. Requirement modeling using Entity Relationship Diagram (Structural Modeling) Sample Scenario: A company wants to develop an online bookstore where users can browse books, place orders, and make payments. The system should also allow authors and publishers to manage their book listings. The bookstore must track inventory, customer details, and order history	3 hours
4. Requirement modeling using Context flow diagram, DFD (Functional Modeling) Sample Scenario: A restaurant chain wants to develop an online food ordering system where customers can browse menus, place orders, and make payments. The system should also allow restaurant staff to manage orders and delivery personnel to track assigned deliveries.	3 hours
5. Requirement modeling using State Transition Diagram (Behavioral Modeling) Sample Scenario : Develop a state machine diagram for an e-commerce application where customers can create service requests such as querying goods, purchasing items, and managing accounts (open, close, deposit, receive payment, reopen, resume). Account operations follow specific rules: opening requires an initial deposit, purchases are restricted by balance limits, and accounts can be suspended or permanently closed based on overdue payments. Design the state transitions based on these conditions.	3 hours
6. Story Boarding and User Interface design Modeling Sample Scenario	3 hours

You are tasked with designing a mobile banking application. Create a storyboard depicting the process of a user logging in, viewing their balance, transferring money, and receiving confirmation. Accompany your storyboard with user interface mockups for each step.	
7. OO design – Use case Model, Class Model Sample Scenario A university wants to develop an online platform where students can register for courses each semester. The system should allow students to log in, view available courses, register or drop courses, and view their schedules. Faculty members should be able to manage course offerings, approve enrollments, and communicate with students. Administrators will oversee the system, manage user roles, and generate reports.	3 hours
8. OO design – Interaction Models Sample Scenario A healthcare provider plans to develop an online appointment booking system that allows patients to register, search for doctors by specialty, book appointments, receive confirmations, and reschedule or cancel them as needed. Doctors should be able to manage availability, view schedules, and approve or reject appointments. The system must also notify patients via email or SMS. Design a UML interaction model that captures key interactions—such as 'Book Appointment' and 'Cancel Appointment'—involving users, system components, and external services.	3 hours
9. OO design – Package, Component and deployment models Sample Scenario A national bank is developing an online banking platform that allows customers to perform transactions such as checking balances, transferring funds, paying bills, and managing accounts. The system must also support admin operations like fraud detection, user management, and audit logging. It should be accessible via web and mobile apps, with secure backend services and integration with third-party payment gateways.	3 hours
10. Design and demonstration of test cases. Functional Testing and Non-Functional Testing Sample Scenario A national railway service is developing an online ticket booking platform. The system allows users to search for trains, check seat availability, book tickets, make payments, and receive e-tickets. It also supports cancellation, refund processing, and real-time updates on train status. The platform must be accessible via web and mobile, and should handle high traffic during peak hours.	3 hours

Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Seminar, Presentaion	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE205	Fundamentals of Artificial Intelligence and Machine Learning	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To teach the theoretical foundations of Artificial Intelligence.
2. To train the students for better understanding, the context of supervised and unsupervised learning through real-life examples.
3. To understand the need for Reinforcement learning in real – world problems.

Course Outcomes

1. Understand the fundamental concepts of Artificial Intelligence, including intelligent agents, problem-solving strategies, and search algorithms.
2. Apply knowledge representation techniques and reasoning methods to make informed decisions.
3. Implement supervised learning algorithms to train predictive models and assess their performance.
4. Use unsupervised learning techniques to discover patterns and extract meaningful insights from unlabelled datasets.
5. Develop reinforcement learning agents that learn optimal behaviours through interaction with dynamic environments and feedback mechanisms.

Module:1	Foundations of AI and Intelligent Agents	10 hours

Introduction to AI - Types of AI - Structure of Intelligent Agents: PEAS framework and Agent Types - Search Techniques: Uninformed Search: Breadth First Search, Depth First Search, Uniform Cost Search, Iterative Deepening Search, Informed Search: Greedy Best First Search, A* Search - Adversarial Search: Mini Max Algorithm, Alpha Beta Pruning.

Module:2	Knowledge Representation and Logical Reasoning	6 hours

Propositional Logic - First-Order Logic (FOL) - Knowledge representation and reasoning - Forward chaining, backward chaining, unification and resolution techniques.

Module:3	Supervised Learning and Model Evaluation	11 hours

Machine Learning Life cycle - Gradient Descent and Stochastic Gradient Descent – Bias Variance Trade-off - Performance metrics – Regression: Linear Regression, Multiple Linear Regression. Classification: Logistic Regression, Decision Trees, ID3, CART, Bayesian Classification Methods, K Nearest Neighbor, Support Vector Machine- Perceptron - Backpropagation algorithm.

Module:4	Unsupervised Learning and Dimensionality Reduction	9 hours

Clustering Techniques: Partitional-Based Clustering (K-Means, K-Mode) - Hierarchical Clustering (AGNES, DIANA) - Density-Based Clustering (DBSCAN) -– Optimal cluster size determining methods- Self Organizing Map (SOM) – Expectation Maximization - Dimensionality reduction methods: PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis).		
Module:5	**Reinforcement Learning**	**7 hours**
Core Concepts in Reinforcement Learning (RL): Foundation - Components - Types. Model-Free Methods: Q-Learning with Temporal Difference (TD) - SARSA - Monte Carlo -Policy-Based Methods: Proximal Policy Optimization (PPO).		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

Stuart Russell and Peter Norvig , "**Artificial Intelligence - A Modern Approach**", Prentice Hall, 4th Edition, **2022**
Tom Mitchell, "**Machine Learning**", McGraw Hill, 3rd Edition, **1997**

Reference Books

Ethem Alpaydın , **"Introduction to Machine Learning**", MIT Press, 4th Edition, **2021**
Kevin Murphy, "**Machine learning: a probabilistic perspective**", MIT Press, 1st Edition, **2012**
Bishop, Christopher M., and Nasser M. Nasrabadi, "**Pattern recognition and Machine Learning**", Springer, 1st Edition, **2006**

Indicative Experiments

1. AI Agents & Search Techniques • A vacuum cleaner reflex agent operates in a 5x5 grid where each cell may be clean or dirty. The agent senses only the current cell: if dirty, it cleans; if clean, it moves randomly. Simulate this behavior using Pygame and visualize the cleaning process. • A robot must find the shortest path from a start point to a goal in a 2D grid with obstacles. Implement and visualize search algorithms like BFS, DFS, and A* using Pygame to simulate the robot's pathfinding. Compare the performance of each algorithm in terms of path length and nodes explored.	5 hours
2. Knowledge Representation and Reasoning A security system monitors rooms using sensors that report simple true/false conditions (e.g., motion detected, door open). Use PyDatalog to represent these conditions in propositional logic and determine if a	5 hours

room is secure or not. Visualize the results in Pygame, showing sensor states and logic outcomes for each room.	
3. Supervised Learning • A real estate agency wants to predict house prices based on features like size, number of bedrooms, and location score. Implement a Linear Regression model using scikit-learn or TensorFlow to train on historical data and make predictions. Visualize the results, showing the regression line and prediction accuracy. • A healthcare provider wants to classify patients as at-risk or not based on features like age, blood pressure, and cholesterol levels. Use Decision Trees and Random Forest from scikit-learn to build classification models. Compare their performance using accuracy, confusion matrix, and feature importance visualization.	5 hours
4. Unsupervised Learning and Dimensionality Reduction • A shopping website wants to segment customers based on their purchasing behavior. Use K-Means and DBSCAN from scikit-learn to cluster customer data and identify distinct groups. Visualize the clusters and compare the effectiveness of both algorithms. • An e-commerce platform wants to visualize high-dimensional customer data for pattern discovery. Apply PCA (using scikit-learn) and Self-Organizing Maps (SOM) (using MiniSom) to reduce data dimensions. Compare and visualize the results to highlight how each method captures customer groupings.	5 hours
5. Reinforcement Learning • A game agent must learn to navigate a grid-world maze with rewards and penalties, without knowing the environment dynamics. Implement Q-Learning, SARSA, and Monte Carlo methods using TensorFlow or PyTorch to train the agent through trial-and-error. Compare their learning curves, policies, and cumulative rewards over episodes. • A robotic arm must learn to reach a target position with continuous control in a simulated environment. Implement Proximal Policy Optimization (PPO) using TensorFlow or PyTorch to train the agent through policy-based reinforcement learning. Evaluate and visualize the agent's performance over training episodes using reward plots and action trajectories.	5 hours
6. Model Evaluation • Supervised Learning: accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC (scikit-learn). • Unsupervised Learning: silhouette score, Davies-Bouldin Index, adjusted Rand index (scikit-learn)	5 hours

• Reinforcement Learning: cumulative reward, average episode return, learning curve (TensorFlow, PyTorch, Matplotlib)	
Total Laboratory Hours:	**30 hours**
Reference Books	
Al Sweigart , "**Making Games with Python & Pygame**", CreateSpace Independent Publishing Platform, 1st Edition, **2012**	
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE205	Fundamentals of Artificial Intelligence and Machine Learning	3	0	2	4
Pre-requisite	NIL	Syllabus Version			
		1			

Course Objectives

1. To teach the theoretical foundations of Artificial Intelligence.
2. To train the students for better understanding, the context of supervised and unsupervised learning through real-life examples.
3. To understand the need for Reinforcement learning in real – world problems.

Course Outcomes

1. Understand the fundamental concepts of Artificial Intelligence, including intelligent agents, problem-solving strategies, and search algorithms.
2. Apply knowledge representation techniques and reasoning methods to make informed decisions.
3. Implement supervised learning algorithms to train predictive models and assess their performance.
4. Use unsupervised learning techniques to discover patterns and extract meaningful insights from unlabelled datasets.
5. Develop reinforcement learning agents that learn optimal behaviours through interaction with dynamic environments and feedback mechanisms.

Module:1	Foundations of AI and Intelligent Agents	10 hours

Introduction to AI - Types of AI - Structure of Intelligent Agents: PEAS framework and Agent Types - Search Techniques: Uninformed Search: Breadth First Search, Depth First Search, Uniform Cost Search, Iterative Deepening Search, Informed Search: Greedy Best First Search, A* Search - Adversarial Search: Mini Max Algorithm, Alpha Beta Pruning.

Module:2	Knowledge Representation and Logical Reasoning	6 hours

Propositional Logic - First-Order Logic (FOL) - Knowledge representation and reasoning - Forward chaining, backward chaining, unification and resolution techniques.

Module:3	Supervised Learning and Model Evaluation	11 hours

Machine Learning Life cycle - Gradient Descent and Stochastic Gradient Descent – Bias Variance Trade-off - Performance metrics – Regression: Linear Regression, Multiple Linear Regression. Classification: Logistic Regression, Decision Trees, ID3, CART, Bayesian Classification Methods, K Nearest Neighbor, Support Vector Machine- Perceptron - Backpropagation algorithm.

Module:4	Unsupervised Learning and Dimensionality Reduction	9 hours

Clustering Techniques: Partitional-Based Clustering (K-Means, K-Mode) - Hierarchical Clustering (AGNES, DIANA) - Density-Based Clustering (DBSCAN) -– Optimal cluster size determining methods- Self Organizing Map (SOM) – Expectation Maximization - Dimensionality reduction methods: PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis).		
Module:5	**Reinforcement Learning**	**7 hours**
Core Concepts in Reinforcement Learning (RL): Foundation - Components - Types. Model-Free Methods: Q-Learning with Temporal Difference (TD) - SARSA - Monte Carlo -Policy-Based Methods: Proximal Policy Optimization (PPO).		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

Stuart Russell and Peter Norvig , "**Artificial Intelligence - A Modern Approach**", Prentice Hall, 4th Edition, **2022**
Tom Mitchell, "**Machine Learning**", McGraw Hill, 3rd Edition, **1997**

Reference Books

Ethem Alpaydın , **"Introduction to Machine Learning**", MIT Press, 4th Edition, **2021**
Kevin Murphy, "**Machine learning: a probabilistic perspective**", MIT Press, 1st Edition, **2012**
Bishop, Christopher M., and Nasser M. Nasrabadi, "**Pattern recognition and Machine Learning**", Springer, 1st Edition, **2006**

Indicative Experiments

1. AI Agents & Search Techniques • A vacuum cleaner reflex agent operates in a 5x5 grid where each cell may be clean or dirty. The agent senses only the current cell: if dirty, it cleans; if clean, it moves randomly. Simulate this behavior using Pygame and visualize the cleaning process. • A robot must find the shortest path from a start point to a goal in a 2D grid with obstacles. Implement and visualize search algorithms like BFS, DFS, and A* using Pygame to simulate the robot's pathfinding. Compare the performance of each algorithm in terms of path length and nodes explored.	5 hours
2. Knowledge Representation and Reasoning A security system monitors rooms using sensors that report simple true/false conditions (e.g., motion detected, door open). Use PyDatalog to represent these conditions in propositional logic and determine if a	5 hours

room is secure or not. Visualize the results in Pygame, showing sensor states and logic outcomes for each room.	
3. Supervised Learning • A real estate agency wants to predict house prices based on features like size, number of bedrooms, and location score. Implement a Linear Regression model using scikit-learn or TensorFlow to train on historical data and make predictions. Visualize the results, showing the regression line and prediction accuracy. • A healthcare provider wants to classify patients as at-risk or not based on features like age, blood pressure, and cholesterol levels. Use Decision Trees and Random Forest from scikit-learn to build classification models. Compare their performance using accuracy, confusion matrix, and feature importance visualization.	5 hours
4. Unsupervised Learning and Dimensionality Reduction • A shopping website wants to segment customers based on their purchasing behavior. Use K-Means and DBSCAN from scikit-learn to cluster customer data and identify distinct groups. Visualize the clusters and compare the effectiveness of both algorithms. • An e-commerce platform wants to visualize high-dimensional customer data for pattern discovery. Apply PCA (using scikit-learn) and Self-Organizing Maps (SOM) (using MiniSom) to reduce data dimensions. Compare and visualize the results to highlight how each method captures customer groupings.	5 hours
5. Reinforcement Learning • A game agent must learn to navigate a grid-world maze with rewards and penalties, without knowing the environment dynamics. Implement Q-Learning, SARSA, and Monte Carlo methods using TensorFlow or PyTorch to train the agent through trial-and-error. Compare their learning curves, policies, and cumulative rewards over episodes. • A robotic arm must learn to reach a target position with continuous control in a simulated environment. Implement Proximal Policy Optimization (PPO) using TensorFlow or PyTorch to train the agent through policy-based reinforcement learning. Evaluate and visualize the agent's performance over training episodes using reward plots and action trajectories.	5 hours
6. Model Evaluation • Supervised Learning: accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC (scikit-learn). • Unsupervised Learning: silhouette score, Davies-Bouldin Index, adjusted Rand index (scikit-learn)	5 hours

• Reinforcement Learning: cumulative reward, average episode return, learning curve (TensorFlow, PyTorch, Matplotlib)	
Total Laboratory Hours:	**30 hours**
Reference Books	
Al Sweigart , "**Making Games with Python & Pygame**", CreateSpace Independent Publishing Platform, 1st Edition, **2012**	
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BAMAT205	Discrete Mathematics and Linear Algebra	3	1	0	4
Pre-requisite	Multivariable Calculus and Differential Equations	Syllabus Version			
		1.1			

Course Objectives

1. To develop logical reasoning and problem-solving skills by understanding mathematical logic, inference theory, and predicate calculus for engineering applications.
2. To apply counting techniques, algebraic structures, and lattice theory to optimize combinatorial problems and system designs in engineering.
3. To utilize vector spaces, linear transformations, and inner product spaces for efficient representation, computation, and analysis in engineering modeling and simulations.

Course Outcomes

At the end of this course, the student is able to

1. apply proof techniques in solving logical problems
2. solve engineering problems involving counting principles.
3. relate algebraic structures to enhance problem-solving techniques.
4. understand the concepts of vector space, subspaces and linear transformations.
5. apply vector normalization techniques to generate orthonormal basis.

Module:1	Mathematical Logic	9 hours

Statements and Notation-Connectives–Tautologies-Equivalence - Implications–Normal forms - The Theory of Inference for the Statement Calculus - Predicate Calculus – Inference Theory of the Predicate Calculus.

Applications: *Validity of Algorithms by Inference*

Module:2	Counting Techniques	8 hours

Basics of counting-Pigeonhole principle- Permutations and combinations-Inclusion-exclusion principle-Recurrence relations - Solving recurrence relations-Generating functions.

Applications: *Analyzing Recursive Functions, Complexity of Algorithms*

Module:3	Algebraic Structures and Lattices	8 hours

Groups – Subgroups – Lagrange's Theorem Homomorphism –Properties-
Partially Ordered Relations -Lattices as Posets – Hasse Diagram – Properties of Lattices. Boolean Algebra.

Applications: *Coding Theory -Error Correction and Detection in Communication.*

Module:4	Vector Spaces and Linear Transformations	10 hours

The Euclidean space R^n Vector space and subspace, linear combination, span, linear dependence and independence, basis and dimension. Row space, column space, null space, and Rank-Nullity Theorem. Linear transformations – definition, properties, matrix representation, invertibility, and change of basis.

Applications: *Computer Graphics - Rotation, Scaling and Translations, JPEG & MPEG Compression.*

Module:5	Inner Product Spaces	8 hours

Dot and inner products, length of a vector, angle between vectors, and matrix representation of inner products. Gram-Schmidt orthogonalization, orthonormal basis. Quadratic forms and their nature – positive definite, negative definite, and indefinite forms.

Applications: *Modeling Signals with Orthonormal Basis, Computer Vision.*

Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture hours: **Total Tutorial hours:**	**45 hours** **15 hours**

Text Book(s)	
1.	J.P.Trembley and R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, 2017, 35th Reprint, Tata Mc Graw Hill, New Delhi.
2.	Stephen Andrilli and David Hecker, Elementary Linear Algebra, 5th Edition, Academic Press, 2016
Reference Books	
1.	Kenneth H. Rosen, Discrete Mathematics and its applications, 2019, 8th Edition, Tata McGraw Hill, Noida.
2.	Gilbert Strang, Introduction to Linear Algebra, 2023, 6th Edition, Wellesley- Cambridge Press.
Mode of Evaluation: Quizzes, Assignments, CATs and FAT	

Recommended by Board of Studies	14-May-2025		
Approved by Academic Council		Date	

Course Code	Course Title	L	T	P	C
BACSE319	Cryptography and Network Security	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

To explore the concepts of security, number theory and types of cryptosystems.
To impart the concepts of digital signatures, authentication and hashing.
To apply the concepts of internet security and Network Endpoint.

Course Outcomes

Realize the fundamentals of security services and number theory.
Explore the different types of cryptosystems.
Comprehend the concepts of data integrity algorithms.
Analyze mutual trust models and user authentication.
Deduce the various network and Internet security mechanisms.

Module:1	Cryptography Concepts	8 hours

Information and Network Security Concepts - Security attacks - Security Services - Security Mechanisms - Elementary Number Theory - Euclidean Algorithm – Modular Arithmetic – Prime Numbers – Fermat's and Eulers Theorems – Testing for Primality – Discrete Logarithms - Pseudo Random Bit Generation.

Module:2	Cryptographic Algorithms	9 hours

Symmetric Key Cryptography - Stream Cipher – RC4 algorithm - Block Cipher - DES, Triple – DES, AES, Block Cipher Modes of Operation - Asymmetric Key Cryptography - RSA Algorithm, Elgamal Algorithm, Diffie-Hellman Key Exchange - Elliptic Curve Cryptography.

Module:3	Data Integrity Algorithms	9 hours

Message Authentication Code - Secure Hash Algorithm: SHA 256 – SHA 512- Hash based Message Authentication Code –Entity Authentication-Digital Signatures– Standards - RSA Digital Signature - ElGamal based Digital Signature.

Module:4	Mutual Trust	7 hours

Cryptographic Key Management and Distribution – Distribution of public keys – X.509 Certificates – Public Key Infrastructure – User Authentication – Remote User Authentication Principles – Kerberos – Federated Identity Management.

Module:5	**Network Security and Post Quantum Cryptography**	**10 hours**
Electronic Mail Security – PGP,S/MIME - Transport Level Security – IP Security - Intrusion Detection System - Wireless Network Security -Introduction to Post Quantum Cryptography – Overview of NIST PQC standardization - PQC Algorithms - Lattice Based Cryptography.		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

William Stallings, "**Cryptography and Network Security Principles and Practice**", Pearson Education, 8th Edition, **2023**

Reference Books

Behrouz A Forouzan and Debdeep Mukhopadhyay, "**Cryptography and Network Security**", McGraw Hill, 3rd Edition, **2015**

Christof Paar, Jan Pelzl, Tim Güneysu, "**Understanding Cryptography From Established Symmetric and Asymmetric Ciphers to Post-Quantum Algorithms**", Springer, 2nd Edition, **2024**

Indicative Experiments

1. Consider a sender and receiver who need to exchange data confidentially using symmetric encryption. Write program that implements DES encryption and decryption.	3 hours
2. Consider a sender and receiver who need to exchange data confidentially using symmetric encryption. Write program that implements AES-128/192/256 cryptosystem.	3 hours
3. (i) Implement RSA algorithm to verify the user transmitted messages. (ii) Implement Elliptic Curve Cryptography	2 hours
4. Implement hashing algorithm that generates hash-based Message Authentication Code.	2 hours
5. Find a Message Authentication Code for given variable size message by using SHA-128 and SHA-256 Hash algorithm. Measure the time consumptions for varying message size for both SHA-128 and SHA-256.	4 hours
6. Apply Digital Signature Standard for verifying the legal communication parties.	4 hours

7. Verify the keys shared by implementing Diffie Hellman multiparty key exchange protocol while performing Man-in-the- Middle Attack.	2 hours
8. Write a simple client and server application using SSL socket communication.	2 hours
9. Write a simple client server model and capture the packets using packet capturing tools. Analyze the PCAP file and get the transmitted data (plain text) using any packet capturing tool. Implement the above scenario using SSH and observe the data.	2 hours
10. Construct Intrusion Detection System rules to notify attack traffic flows.	2 hours
11. Send and receive an encrypted email message using S/MIME.	2 hours
12. Construct IP tables for incoming network traffic flows.	2 hours
Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Oral Examination, Seminar, Group Discussion, Presentation	
Recommended by Board of Studies :	02-06-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE319	Cryptography and Network Security	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

To explore the concepts of security, number theory and types of cryptosystems.
To impart the concepts of digital signatures, authentication and hashing.
To apply the concepts of internet security and Network Endpoint.

Course Outcomes

Realize the fundamentals of security services and number theory.
Explore the different types of cryptosystems.
Comprehend the concepts of data integrity algorithms.
Analyze mutual trust models and user authentication.
Deduce the various network and Internet security mechanisms.

Module:1	Cryptography Concepts	8 hours

Information and Network Security Concepts - Security attacks - Security Services - Security Mechanisms - Elementary Number Theory - Euclidean Algorithm – Modular Arithmetic – Prime Numbers – Fermat's and Eulers Theorems – Testing for Primality – Discrete Logarithms - Pseudo Random Bit Generation.

Module:2	Cryptographic Algorithms	9 hours

Symmetric Key Cryptography - Stream Cipher – RC4 algorithm - Block Cipher - DES, Triple – DES, AES, Block Cipher Modes of Operation - Asymmetric Key Cryptography - RSA Algorithm, Elgamal Algorithm, Diffie-Hellman Key Exchange - Elliptic Curve Cryptography.

Module:3	Data Integrity Algorithms	9 hours

Message Authentication Code - Secure Hash Algorithm: SHA 256 – SHA 512- Hash based Message Authentication Code –Entity Authentication-Digital Signatures– Standards - RSA Digital Signature - ElGamal based Digital Signature.

Module:4	Mutual Trust	7 hours

Cryptographic Key Management and Distribution – Distribution of public keys – X.509 Certificates – Public Key Infrastructure – User Authentication – Remote User Authentication Principles – Kerberos – Federated Identity Management.

Module:5	**Network Security and Post Quantum Cryptography**	**10 hours**
Electronic Mail Security – PGP,S/MIME - Transport Level Security – IP Security - Intrusion Detection System - Wireless Network Security -Introduction to Post Quantum Cryptography – Overview of NIST PQC standardization - PQC Algorithms - Lattice Based Cryptography.		
Module:6	**Contemporary Issues**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

William Stallings, "**Cryptography and Network Security Principles and Practice**", Pearson Education, 8th Edition, **2023**

Reference Books

Behrouz A Forouzan and Debdeep Mukhopadhyay, "**Cryptography and Network Security**", McGraw Hill, 3rd Edition, **2015**

Christof Paar, Jan Pelzl, Tim Güneysu, "**Understanding Cryptography From Established Symmetric and Asymmetric Ciphers to Post-Quantum Algorithms**", Springer, 2nd Edition, **2024**

Indicative Experiments

1. Consider a sender and receiver who need to exchange data confidentially using symmetric encryption. Write program that implements DES encryption and decryption.	3 hours
2. Consider a sender and receiver who need to exchange data confidentially using symmetric encryption. Write program that implements AES-128/192/256 cryptosystem.	3 hours
3. (i) Implement RSA algorithm to verify the user transmitted messages. (ii) Implement Elliptic Curve Cryptography	2 hours
4. Implement hashing algorithm that generates hash-based Message Authentication Code.	2 hours
5. Find a Message Authentication Code for given variable size message by using SHA-128 and SHA-256 Hash algorithm. Measure the time consumptions for varying message size for both SHA-128 and SHA-256.	4 hours
6. Apply Digital Signature Standard for verifying the legal communication parties.	4 hours

7. Verify the keys shared by implementing Diffie Hellman multiparty key exchange protocol while performing Man-in-the- Middle Attack.	2 hours
8. Write a simple client and server application using SSL socket communication.	2 hours
9. Write a simple client server model and capture the packets using packet capturing tools. Analyze the PCAP file and get the transmitted data (plain text) using any packet capturing tool. Implement the above scenario using SSH and observe the data.	2 hours
10. Construct Intrusion Detection System rules to notify attack traffic flows.	2 hours
11. Send and receive an encrypted email message using S/MIME.	2 hours
12. Construct IP tables for incoming network traffic flows.	2 hours
Total Laboratory Hours:	**30 hours**
Mode of Evaluation : Continuous Assessment Test, Digital Assignment, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Oral Examination, Seminar, Group Discussion, Presentation	
Recommended by Board of Studies :	02-06-2025
Approved by Academic Council : No. 78	12-06-2025

Course Code	Course Title	L	T	P	C
BACSE324	Embedded Systems Architecture	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

1. To explain embedded system constraints and relate them to design requirements and architecture.
2. To configure and program sensors, actuators, data converters and serial interfaces for reliable device integration.
3. To develop embedded software that meets resource and deadline constraints using real-time scheduling.

Course Outcomes

1. Articulate the key concepts of embedded system using various microcontrollers and interfaces.
2. Construct embedded systems by interfacing memory and I/O devices.
3. Analyze various tools to optimize the performance of the code.
4. Implement real-time scheduling strategies under resource constraints to meet deadlines and report deadline miss rates.
5. Evaluate the principles of serial communication protocols and their practical applications.

Module:1	**Introduction**	**9 hours**

Embedding Computers - Characteristics of Embedded Computing Applications - Challenges in Embedded Computing System Design - Performance of Embedded Computing Systems - Embedded System Design Process - Requirements - Specification - Architecture Design - Designing Hardware and Software Components - System Integration - Microcontroller Architecture - 8051, PIC, ARM - Case Study: IoT Board.

Module:2	**I/O Interfacing Techniques**	**9 hours**

Memory Interfacing - Analog-to-Digital and Digital-to-Analog Interfacing - Timers - Watchdog Timer - Counters - Universal Asynchronous Receiver/Transmitter (UART) - Sensor Interfacing - Pulse Width Modulation (PWM) Control of DC Motor - Stepper Motor Interfacing.

Module:3	**Programming Tools**	**8 hours**

Embedded Programming Tools - Modeling Programs: Data Flow Graph (DFG), Control Data Flow Graph (CDFG), Finite State Machine (FSM) - Code Optimization - Logic Analyzers.

Module:4	**Real Time Operating System**	**8 hours**

Classification of Real Time Systems (RTS) - Issues and Challenges in RTS - Priority-Based Scheduling -

Real Time Scheduling Schemes - Shared Resources - Embedded Configurable Operating System (eCos) and Portable Operating System Interface (POSIX).		
Module:5	**Embedded Networking Protocols and Applications**	**9 hours**
Inter-Integrated Circuits (I2C) - Controller Area Network - Embedded Ethernet Controller - RS232 - Bluetooth - Zigbee - WiFi - Embedded System Applications Using Case Studies: Agriculture Sector, Automotive Electronics, Consumer Electronics, Industrial Controls, Medical Electronics, Embedded AI Systems, Quantum Embedded Systems Architecture.		
Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

1. Marilyn Wolf, "**Computers as Componentsâ€“Principles of Embedded Computing System Design**", Morgan Kaufman Publishers, 5th Edition, **2022**
2. Muhammad Ali Mazidi, Janice Gillispie Mazidi and Rolin D. McKinlay, "**The 8051 Microcontroller and Embedded Systems: Using Assembly and C**", Pearson, 2nd Edition, **2013**

Reference Books

1. Raj Kamal, "**Embedded Systems Architecture, Programming and Design**", McGraw Hill Education, 3rd Edition, **2015**
2. Tammy Noergaard, "**Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers**", Elsevier Science, 2nd Edition, **2013**
3. Edward Ashford Lee and Sanjit Arunkumar Seshia, "**Introduction to Embedded Systems: A Cyber Physical System Approach**", MIT Press, 2nd Edition, **2017**

Indicative Experiments

1. 8051 Microcontroller I/O operations: Embedded C programs.	4 hours
2. Design a miniature robotic arm system that uses servo motors for joint movement. The system must rotate the servo(s) to specific angles accurately (0°-180°). Angle commands can be received via manual input, sensor feedback, or serial communication (UART). The system may require smooth motion, multi-servo coordination, and position monitoring via LCD or Serial Monitor.	2 hours
3. Design a basic automated environmental control system using Arduino Uno. The system reads real-time values from sensors (e.g., temperature, light, or motion sensors). Based on sensor readings, it automatically controls actuators such as LEDs, fans, buzzers, or relays. For example: o Turn ON a fan when temperature exceeds a threshold.	4 hours

o Turn ON a light when ambient light is below a level. o Activate a buzzer when motion is detected. Sensor readings and actuator status can optionally be displayed on LCD or Serial Monitor.	
4. Design a Smart Environmental Monitoring System for a greenhouse or indoor environment. The system uses a DHT11 (or DHT22) temperature and humidity sensor to measure the ambient conditions and displays real-time values on an LCD screen. If the temperature or humidity crosses safe limits, the system can also trigger alerts (like turning ON a fan or buzzer).	2 hours
5. Designing an intelligent water tank monitoring system for a household or industrial tank. The system uses an ultrasonic sensor (HC-SR04) to measure water level and displays the real-time water level on an LCD. Additionally, the system can trigger alarms, LEDs, or pumps when water reaches critical levels (low or overflow).	2 hours
6. Design a smart obstacle detection system for a robotic vehicle or automated trolley. The system uses an ultrasonic sensor (HC-SR04) or IR proximity sensor to detect obstacles in real-time. When an obstacle is detected within a preset threshold distance, the system triggers alerts (LED, buzzer) and can optionally stop or change the direction of the vehicle. The system may also display distance readings on an LCD for monitoring.	4 hours
7. Design a smart lighting system for an office, classroom, or room.The system uses a PIR (Passive Infrared) occupancy sensor to detect human presence. When the sensor detects someone entering the room, the lights automatically turn ON. When no motion is detected for a preset time, the lights automatically turn OFF. Optionally, the system can display occupancy and light status on an LCD or Serial Monitor for monitoring and debugging.	4 hours
8. Design a smart parking system for a parking lot or garage. The system uses ultrasonic sensors, IR sensors, or proximity sensors to detect the presence of vehicles in parking slots. The system can monitor available and occupied slots in real-time. Display the number of available slots on an LCD or seven-segment display. Optionally, control entry/exit barriers or LED indicators for occupied/free slots. This helps in efficient parking management and reduces human intervention.	4 hours
9. Case studies based real-time implementation.	4 hours
Total Laboratory Hours:	**30 hours**

Mode of Evaluation: Continuous Assessment Test, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Oral Examination, Presentation, Course Project, Case Study	
Recommended by Board of Studies :	16-10-2025

Approved by Academic Council : No. 80	20-11-2025

Course Code	Course Title	L	T	P	C
BACSE324	Embedded Systems Architecture	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

1. To explain embedded system constraints and relate them to design requirements and architecture.
2. To configure and program sensors, actuators, data converters and serial interfaces for reliable device integration.
3. To develop embedded software that meets resource and deadline constraints using real-time scheduling.

Course Outcomes

1. Articulate the key concepts of embedded system using various microcontrollers and interfaces.
2. Construct embedded systems by interfacing memory and I/O devices.
3. Analyze various tools to optimize the performance of the code.
4. Implement real-time scheduling strategies under resource constraints to meet deadlines and report deadline miss rates.
5. Evaluate the principles of serial communication protocols and their practical applications.

Module:1	**Introduction**	**9 hours**

Embedding Computers - Characteristics of Embedded Computing Applications - Challenges in Embedded Computing System Design - Performance of Embedded Computing Systems - Embedded System Design Process - Requirements - Specification - Architecture Design - Designing Hardware and Software Components - System Integration - Microcontroller Architecture - 8051, PIC, ARM - Case Study: IoT Board.

Module:2	**I/O Interfacing Techniques**	**9 hours**

Memory Interfacing - Analog-to-Digital and Digital-to-Analog Interfacing - Timers - Watchdog Timer - Counters - Universal Asynchronous Receiver/Transmitter (UART) - Sensor Interfacing - Pulse Width Modulation (PWM) Control of DC Motor - Stepper Motor Interfacing.

Module:3	**Programming Tools**	**8 hours**

Embedded Programming Tools - Modeling Programs: Data Flow Graph (DFG), Control Data Flow Graph (CDFG), Finite State Machine (FSM) - Code Optimization - Logic Analyzers.

Module:4	**Real Time Operating System**	**8 hours**

Classification of Real Time Systems (RTS) - Issues and Challenges in RTS - Priority-Based Scheduling -

Real Time Scheduling Schemes - Shared Resources - Embedded Configurable Operating System (eCos) and Portable Operating System Interface (POSIX).		
Module:5	**Embedded Networking Protocols and Applications**	**9 hours**
Inter-Integrated Circuits (I2C) - Controller Area Network - Embedded Ethernet Controller - RS232 - Bluetooth - Zigbee - WiFi - Embedded System Applications Using Case Studies: Agriculture Sector, Automotive Electronics, Consumer Electronics, Industrial Controls, Medical Electronics, Embedded AI Systems, Quantum Embedded Systems Architecture.		
Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture Hours:	**45 hours**

Text Book(s)

1. Marilyn Wolf, "**Computers as Componentsâ€“Principles of Embedded Computing System Design**", Morgan Kaufman Publishers, 5th Edition, **2022**
2. Muhammad Ali Mazidi, Janice Gillispie Mazidi and Rolin D. McKinlay, "**The 8051 Microcontroller and Embedded Systems: Using Assembly and C**", Pearson, 2nd Edition, **2013**

Reference Books

1. Raj Kamal, "**Embedded Systems Architecture, Programming and Design**", McGraw Hill Education, 3rd Edition, **2015**
2. Tammy Noergaard, "**Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers**", Elsevier Science, 2nd Edition, **2013**
3. Edward Ashford Lee and Sanjit Arunkumar Seshia, "**Introduction to Embedded Systems: A Cyber Physical System Approach**", MIT Press, 2nd Edition, **2017**

Indicative Experiments

1. 8051 Microcontroller I/O operations: Embedded C programs.	4 hours
2. Design a miniature robotic arm system that uses servo motors for joint movement. The system must rotate the servo(s) to specific angles accurately (0°-180°). Angle commands can be received via manual input, sensor feedback, or serial communication (UART). The system may require smooth motion, multi-servo coordination, and position monitoring via LCD or Serial Monitor.	2 hours
3. Design a basic automated environmental control system using Arduino Uno. The system reads real-time values from sensors (e.g., temperature, light, or motion sensors). Based on sensor readings, it automatically controls actuators such as LEDs, fans, buzzers, or relays. For example: o Turn ON a fan when temperature exceeds a threshold.	4 hours

o Turn ON a light when ambient light is below a level. o Activate a buzzer when motion is detected. Sensor readings and actuator status can optionally be displayed on LCD or Serial Monitor.	
4. Design a Smart Environmental Monitoring System for a greenhouse or indoor environment. The system uses a DHT11 (or DHT22) temperature and humidity sensor to measure the ambient conditions and displays real-time values on an LCD screen. If the temperature or humidity crosses safe limits, the system can also trigger alerts (like turning ON a fan or buzzer).	2 hours
5. Designing an intelligent water tank monitoring system for a household or industrial tank. The system uses an ultrasonic sensor (HC-SR04) to measure water level and displays the real-time water level on an LCD. Additionally, the system can trigger alarms, LEDs, or pumps when water reaches critical levels (low or overflow).	2 hours
6. Design a smart obstacle detection system for a robotic vehicle or automated trolley. The system uses an ultrasonic sensor (HC-SR04) or IR proximity sensor to detect obstacles in real-time. When an obstacle is detected within a preset threshold distance, the system triggers alerts (LED, buzzer) and can optionally stop or change the direction of the vehicle. The system may also display distance readings on an LCD for monitoring.	4 hours
7. Design a smart lighting system for an office, classroom, or room.The system uses a PIR (Passive Infrared) occupancy sensor to detect human presence. When the sensor detects someone entering the room, the lights automatically turn ON. When no motion is detected for a preset time, the lights automatically turn OFF. Optionally, the system can display occupancy and light status on an LCD or Serial Monitor for monitoring and debugging.	4 hours
8. Design a smart parking system for a parking lot or garage. The system uses ultrasonic sensors, IR sensors, or proximity sensors to detect the presence of vehicles in parking slots. The system can monitor available and occupied slots in real-time. Display the number of available slots on an LCD or seven-segment display. Optionally, control entry/exit barriers or LED indicators for occupied/free slots. This helps in efficient parking management and reduces human intervention.	4 hours
9. Case studies based real-time implementation.	4 hours
Total Laboratory Hours:	**30 hours**

Mode of Evaluation: Continuous Assessment Test, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Oral Examination, Presentation, Course Project, Case Study	
Recommended by Board of Studies :	16-10-2025

Approved by Academic Council : No. 80	20-11-2025

Course Code	Course Title	L	T	P	C
BACSE325	Full Stack Development	2	0	4	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

To provide foundational knowledge of web technologies including HyperText Markup Language (HTML), Cascading Style Sheets CSS), JavaScript and client-side frameworks.
To design and develop dynamic, interactive and responsive web applications using React and Angular.
To develop competency in server-side programming with Node.js, ExpressJS and MongoDB for building and deploying full-stack applications

Course Outcomes

Apply HTML, CSS and JavaScript to create structured, styled and interactive web pages.
Build component-based applications in React with state management, routing and event handling.
Develop Angular applications using TypeScript with templates, services, routing and validation
Integrate MongoDB with Node.js applications for effective database operations, access control and data handling.
Design and implement full stack web applications with form handling, authentication and client-server communication.

Module:1	**Web Foundations**	**5 hours**

HTML Basics: Tags, Attributes, Meta Tags, Lists, Tables, Images, Forms- CSS Fundamentals: Selectors, Properties, Box Model- CSS Animations and Frameworks- JavaScript Essentials: Data types, Functions, Objects - Document Object Model (DOM) Basics and Event Handling: onclick, onchange, onmouseover - Simple Form Validation Using Regular Expressions.

Module:2	**React Essentials**	**6 hours**

React: Architecture and Setup - JavaScript XML (JSX), Components, Props and State - React Styling: Inline and External CSS - Event Handling and Modularization - React

Hooks: useState, useEffect - React Routing - Simple Client-Side Form Programming - React Native and Cross-Platform Development - Core Components -Styling.		
Module:3	**Angular with TypeScript**	**6 hours**
TypeScript - Angular Setup - Angular Components, Templates and Data Binding - Directives for DOM Manipulation and Dynamic Views - Forms and Basic Validation - Services and Routing - Communicating with Representational State Transfer (REST) Application Programming Interface (APIs).		
Module:4	**Node.js and MongoDB**	**6 hours**
Node.js Setup and NPM Packages - Events, Listeners, Timers, Callbacks - Building Simple HTTP Services in Node.js - MongoDB: Environment, Users, Access Control - Managing Databases and Collections - Connecting Node.js with MongoDB.		
Module:5	**ExpressJS and RESTful APIs**	**5 hours**
Setting up ExpressJS - RESTful APIs and Routing - Templating and Data Handling - Form Handling and Authentication: Cookies and Sessions - Express on the Node.js runtime.		
Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture Hours:	**30 hours**
Text Book(s)		
Brad Dayley, Brendan Dayley and Caleb Dayley, "**Node.js, MongoDB and Angular Web Development**", Addison-Wesley, 2nd Edition, **2019** Daniel Bugl and Matthias Zronek, "**Modern Full-Stack React Projects: Build, maintain, and deploy modern web apps using MongoDB, Express, React, and Node.js**", Packt Publishers, 1st Edition, **2024**		
Reference Books		
Philip Ackermann, "**Full Stack Web Development**", SAP Press, 1st Edition, **2023** Frank Zammetti, "**Modern Full Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker**", Oâ€™Reilly Media, 1st Edition, **2021** Vasan Subramanian, "**Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node**", Apress, 2nd Edition, **2019**		
Indicative Experiments		

1. Build a Profile Page: In this exercise, you will build a personal profile page step by step to learn HTML fundamentals. Start with a basic HTML template and progressively add tags and attributes such as lang, id, class, and title. Include essential meta tags for charset, viewport, and description, then create lists: an unordered list for hobbies, an ordered list for goals, and a definition list for technical terms. Next, design a table with headings and rows to show educational details, followed by inserting images for a profile picture and gallery with proper alt text and captions. Build a form with inputs, radio buttons, checkboxes, and validation attributes like required and type="email". After each step, a feedback panel should appear with hints, and if multiple edits are made quickly, they should be queued so validation and tips appear in sequence. Finally, attempt bonus challenges like adding a video, audio, or hyperlinks, ensuring smooth progression without overlapping or skipped steps	4 hours
2. Design a digital recipe card to explore CSS fundamentals. Begin with applying basic selectors (element, class, id) to style headings, text, and images, then use properties like color, font-size, margin, and padding to refine the layout. Implement the box model by clearly showing content, padding, border, and margin for each recipe section, ensuring proper spacing and alignment. Add simple CSS animations such as fading in the title, scaling images on hover, and smoothly transitioning button colors. Finally, experiment with a CSS framework like Bootstrap or Tailwind to quickly style a responsive recipe card layout, comparing framework defaults with your custom CSS rules.	4 hours
3. Build a small login form to practice JavaScript essentials. Start by declaring variables of different data types (string for username, number for a simple PIN). Write a function that checks if the entered details match stored values inside an object. Use DOM methods to read input fields and update messages on the page. Add event handling such as onclick for the login button and onmouseover to highlight the input box. Finally, apply a simple regular expression to validate that the username is an email format before allowing login.	4 hours
4. Create a simple â€œTo-Do Listâ€• application to practice React fundamentals. Begin by setting up a React project and using JSX to design the layout with a heading, input box, and button. Break the UI into components such as TodoInput, TodoItem, and TodoList, passing data through props to display tasks. Use state to store the list of tasks	4 hours

and update it dynamically when a new task is added or deleted. Finally, apply React styling (inline styles, CSS modules, or styled-components) to format the task list, highlight completed items, and style buttons for a polished look.	
5. Build a small â€œColor Changerâ€• webpage to practice inline and external CSS along with JavaScript event handling and modularization. Start by applying inline CSS to style a heading and external CSS to design the overall page layout. Add buttons labeled with different colors, and use event handling (onclick) so that clicking a button changes the background color of the page. Organize the JavaScript code into separate functions (modules) such as changeToRed(), changeToBlue(), and changeToGreen() to demonstrate modularization. Finally, ensure that all styles and scripts are properly separated into external files for clean structure, while still showing how inline styles can override external rules when needed	4 hours
6. Build a small â€œStudent Portalâ€• app to practice React hooks, routing, and form handling. Begin with a homepage component styled using JSX, then create two routes using React Router: one for â€œStudent Listâ€• and another for â€œRegister Student.â€• In the registration form, use useState to store input values such as name and email, and apply simple validation before submission. Implement useEffect to display a welcome message when the form loads and to log form submissions to the console whenever the state updates. Finally, ensure smooth navigation between pages using Link components, and style the form with clear input boxes and a submit button for a polished client-side experience.	4 hours
7. Build a simple â€œProduct Catalogâ€• app to practice Angular components, templates, and data binding. Begin by creating a root component with a title and a ProductListComponent that displays product details such as name, price, and availability using interpolation. Add property binding to dynamically change the product image source and class binding to highlight items that are on sale. Use event binding with a button click to add a product to a favorites list, and apply two-way data binding with ngModel in an input box to filter products by name. Finally, structure the template for clarity with proper Angular directives like *ngFor and *ngIf, ensuring smooth interaction between components and the displayed data.	4 hours

8. Build a simple â€œEvent Bookingâ€• app to practice Angular directives and form validation. Begin with a form where users enter their name, select an event from a dropdown, and choose the number of tickets. Use two-way data binding with ngModel to capture input values, and apply structural directives like *ngIf to display a confirmation message only after the form is submitted. Add *ngFor to dynamically render the list of available events, and use attribute directives such as ngClass to highlight the selected option. Implement basic validation rules like required for all fields and a numeric range check for ticket count, showing real-time error messages under each field. Finally, on submission, display the booking summary in a dynamic view while disabling the form to prevent duplicate submissions.	4 hours
9. Create a â€œWeather Dashboardâ€• app to practice Angular services, routing, and API communication. Begin with two routes: a home page with a search box and a results page to display weather details. Create a service that communicates with a public weather REST API to fetch temperature, humidity, and forecast based on the entered city. Use dependency injection to provide the service to multiple components, ensuring data flow is consistent. On the results page, display the fetched data dynamically with proper formatting, and handle errors gracefully if the API returns invalid input. Finally, style the dashboard with a clear layout, and enable smooth navigation between the home and results views using Angular routing.	4 hours
10. Create a simple â€œTask Reminderâ€• app using Node.js to practice events, listeners, timers, and callbacks while building an HTTP service. Start by setting up an HTTP server that responds with a welcome message when accessed from the browser. Use the built-in events module to create a custom event called taskAdded, and attach a listener that logs when a new task is added. Implement a timer with setTimeout to simulate sending a reminder after a few seconds, and use a callback function to confirm completion. Finally, expose an endpoint (e.g., /addtask) where users can add tasks, triggering the event and timer, and return a dynamic HTTP response showing the task details.	4 hours
11. Set up a MongoDB environment and practice user management with access control. Begin by installing and starting the MongoDB server, then create a new database named studentDB. Inside it, add a collection students with a few sample documents containing fields like name, age, and course. Configure user accounts by creating an	4 hours

administrator with full privileges and an application user with read/write access limited only to studentDB. Test access control by logging in with different users and performing operations such as inserting, updating, or querying documents to verify permissions. Finally, demonstrate how to enable authentication in the MongoDB configuration file and reconnect securely to the environment.	
12. Build a simple â€œLibrary Managerâ€• app to practice managing databases and collections while connecting Node.js with MongoDB. Start by creating a database named libraryDB and a collection books with documents containing fields like title, author, and year. Use the official MongoDB Node.js driver (or Mongoose) to connect your Node.js application to the database. Implement routes in your Node.js server to perform basic operations: inserting a new book, retrieving all books, updating a book's year, and deleting a book by ID. Test these routes using a browser or Postman, ensuring that each operation modifies the collection as expected. Finally, log meaningful success/error messages to the console so you can track database interactions in real time	4 hours
13. Build a â€œStudent Records Managerâ€• app to practice ExpressJS setup, RESTful APIs, routing, templating, and data handling. Begin by setting up a basic ExpressJS server and define routes for GET /students to fetch all students, POST /students to add a new student, PUT /students/:id to update details, and DELETE /students/:id to remove a record. Use middleware to parse incoming JSON data and log requests to the console. Integrate a templating engine like EJS or Handlebars to render a dynamic HTML page that displays student records in a table. On form submission, handle the input data in the server, update the in-memory student list (or connect to a database), and re-render the updated page. Finally, test your routes with Postman and through the browser to ensure both API responses and rendered views work correctly.	4 hours
14. Build a â€œMembership Subscriptionâ€• app to practice form handling and authentication using cookies and sessions. Start with a registration form where users provide their name, email, and password, and a login form to authenticate existing members. Use ExpressJS to process form submissions, validate input, and store user data in memory or a database. Implement sessions to track logged-in users and set cookies to maintain session state across page reloads. Once authenticated, display a dashboard with subscription details, and ensure	4 hours

that accessing this page without a valid session redirects users to the login page. Finally, include a logout option that clears the session and cookies, demonstrating secure management of user authentication and access control.	
15. Build a â€œSimple Blogâ€• application to practice integrating Express with Node.js. Begin by setting up a Node.js project and installing Express to handle server-side routing. Create routes for GET /posts to display all blog posts, POST /posts to add a new post, and DELETE /posts/:id to remove a post. Use middleware to parse incoming JSON and URL-encoded data, and serve static files such as CSS for basic styling. Integrate dynamic HTML rendering using a templating engine like EJS or Pug to display posts on the browser. Finally, test the server by sending requests from the browser and Postman, ensuring that all routes work correctly and data flows seamlessly between the server, templates, and client.	4 hours
Total Laboratory Hours:	**60 hours**
Mode of Evaluation: Continuous Assessment Test, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Oral Examination, Group Discussion, Presentation, Course Project, Case Study	
Recommended by Board of Studies :	16-10-2025
Approved by Academic Council : No. 80	20-11-2025

Course Code	Course Title	L	T	P	C
BACSE326	Distributed System Design	3	0	2	4
Pre-requisite	NIL	**Syllabus Version**			
		1			

Course Objectives

To understand the core principles of distributed systems.
To impart knowledge of communication, data management, reliability and security in distributed environments.
To design cloud-native solutions and DevOps pipelines to build, deploy and manage large-scale applications.

Course Outcomes

Comprehend the fundamentals of distributed system architectures and its components.
Analyze the Application Programming Interfaces (APIs) and communication mechanisms used in distributed systems.
Implement scalable, fault-tolerant and consistent data management solutions.
Apply reliability, scalability and security principles to distributed systems.
Develop and deploy large scale cloud native applications.

Module:1	Fundamentals of Distributed System Design	9 hours

Distributed System: Key Characteristics, Challenges - Core Architectural Models: Client-Server, Peer-to-Peer - Monolithic Architecture: Benefits and Limitations - Service-Oriented Architecture (SOA) vs. Microservices Architecture - Principles of Microservices: Decoupling, Single Responsibility, APIs - Introduction to Resilience Patterns, Cache as core microservice concepts, Benefits and Challenges -
Load Balancing: Need, Round - Robin, Least Connections, Consistent Hashing, Trade-offs in Load Balancing - Load Balancers like Nginx and Amazon Web Services (AWS) - Elastic Load Balancer (ELB).

Module:2	Distributed APIs and Communication	9 hours

API Design and Communication Protocols: Representational State Transfer (REST) Principles, GraphQL, Google Remote Procedure Call (gRPC) - Communication Patterns: HTTP/HTTPS, WebSockets - Decoupling with Asynchronous

Communication: Principles of Event - Driven Architecture, Message Queues vs. Message Brokers, Kafka: Log-based, RabbitMQ : Queue-based and ActiveMQ, API Gateways: Routing, Composition, Offloading Cross-Cutting Concerns - Distributed Caching: Redis, Memcached.		
Module:3	**Distributed Data Management and Fault Tolerance**	**8 hours**
Database Fundamentals and Scaling: The birth of NoSQL, Trade-offs and Use Cases of Document, Key-Value, Column, Graph databases - Database Indexing and Query Optimization - Scaling Reads and Writes: Database Replication, Database Sharding - Distributed Data Consistency: The CAP Theorem with examples - Consistency Models: Strong, Weak and Eventual Consistency - Distributed Consensus: The problem, Introduction to Paxos and Raft algorithms - Conflict Resolution in Eventually Consistent Systems: Vector Clocks.		
Module:4	**Reliability, Scalability and Security**	**8 hours**
Core Reliability Concepts: Fault Tolerance vs. Resilience - Redundancy - Failover Strategies: Hot/Cold Standby - Advanced Resilience Patterns: Retry, Circuit Breaker, Bulkheading, Timeout, Fall Back, Load Shedding - Scalability: Throttling and Rate Limiting, Content Delivery Networks (CDNs) - Security in Distributed Systems - The Shared Responsibility Model: Cloud - Authentication and Authorization: OAuth 2.0 / OpenID Connect (OIDC) - JSON Web Token (JWT) - Secure API Design : Input Validation, HTTPS - Mitigating DDoS Attacks.		
Module:5	**Deployment Strategies and Real-World Case Studies**	**9 hours**
Cloud-Native Paradigm: IaaS, PaaS, SaaS - Containers: Docker - Orchestration: Kubernetes - Infrastructure as Code and Automation: Principles, Terraform, Ansible, Trade-offs and Use Cases - Continuous Integration and Continuous Delivery (CI/CD) Pipelines for Distributed Systems: Jenkins for Complex Pipelines, GitHub Actions - Key Pipeline Stages: Build, Test, Containerize, Security Scanning, Deployment Strategies Blue -Green and Canary - Case Studies: Netflix, Uber, Amazon.		
Module:6	**Contemporary Topics**	**2 hours**
	Total Lecture Hours:	**45 hours**
Text Book(s)		
Maarten Van Steen and Andrew S Tanenbaum, "**Distributed Systems**", Pearson Education, 5th Edition, **2025** Chris Richardson, "**Microservices Patterns: With Examples in Java**", Manning Publications, 2nd Edition, **2025**		

Reference Books	
Martin Kleppmann, "**Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems**", Oâ€™Reilly Media, 2nd Edition, **2022** Kasun Indrasiri and Prabath Siriwardena, "**Design Patterns for Cloud-Native Applications**", Oâ€™Reilly Media, 7th Edition, **2021** Gary Archer, Judith Kahrer and MichaÅ, Trojanowski, **"Cloud Native Data Security with OAuth: A Scalable Zero Trust Architecture"**, Oâ€™Reilly Media, 1st Edition, **2025**	
Indicative Experiments	
1. Containerize microservices and run them with Docker Compose; observe service lifecycle. Example: Create Dockerfiles, a docker-compose.yml, bring up a multi-service stack, simulate failure of one service and restart.	2 hours
2. Deploy microservices on Kubernetes; use Deployments, Services, and HPA. Example: Write Deployment/Service manifests, expose via NodePort/Ingress, configure Horizontal Pod Autoscaler, test scaling by load.	2 hours
3. Add load balancing and caching to improve latency and throughput. Example: Place Nginx/load-balancer in front, add Redis cache for a â€œread-heavyâ€• endpoint, and benchmark latencies before/after.	2 hours
4. Build asynchronous communication using a message broker (publish/subscribe). Example: Implement producer sending events, consumer processing them; show retry/failure handling; implement a simple event store.	2 hours
5. Implement a real-time component (chat/notifications) using WebSockets. Example: Build server and client; scale to multiple clients; show reconnection handling.	2 hours
6. Explore database partitioning and replication strategies.	2 hours

Example: Configure replication and simple sharding; run queries that hit different shards; simulate node failure and failover.	
7. Demonstrate CAP tradeoffs and eventual vs. strong consistency behavior. Example: Create scenarios: normal, partitioned network, and observe reads/writes and staleness.	2 hours
8. Understand and run a consensus protocol (Raft) and observe leader election and log replication. Example: Run a 3-node cluster, perform writes, kill leaders, observe re-election and log consistency	2 hours
9. Implement circuit breaker and bulkhead patterns and test graceful degradation. Example: Introduce latency/failure in downstream service; show circuit breaker tripping and subsequent recovery; isolate resources via bulkhead.	2 hours
10. Design a URL Shortener (Bit.ly, TinyURL)	2 hours
11. Design Social Media Feeds (Facebook/Twitter)	2 hours
12. Design a Cloud Storage System (Google Drive/Dropbox)	2 hours
13. Design a Video Streaming Service (YouTube/Netflix)	2 hours
14. Design a Ride-Sharing App (Uber, Lyft)	4 hours
Total Laboratory Hours:	**30 hours**
Mode of Evaluation: Continuous Assessment Test, Quiz, Final Assessment Test, Lab Continuous Assessment, Lab Final Assessment, Course Project, Case Study	
Recommended by Board of Studies :	16-10-2025
Approved by Academic Council : No. 80	20-11-2025