UNIVERSITY OF CALCUTTA

Notification No. CSR/81/2025

It is notified for information of all concerned that in terms of the provisions of Section 54 of the Calcutta University Act, 1979, (as amended), and, in the exercise of her powers under 9(6) of the said Act, the Vice-Chancellor has, by an order dated 15.10.2025 approved the Syllabus for Semester-5 & 6 of Computer Application (Core Vocational), under CCF.

The above shall take effect from the Odd semester examinations, 2025 and onwards.

29/10/2025

Prof.(Dr.) Debasis Das

Registrar

SENATE HOUSE

Kolkata-700073

29.10.2025

University of Calcutta

B.Sc

4 - Year Degree program in

Computer Application under

Credit Frame work.

Semester – V & VI

(Core-Vocational)

Semester - 5

Theory Paper	Credit	Contact hours	Practical/Tutorial Paper	Credit	Contact hours
Data Communication and Networking	03	45	Computer Networking Lab	1	30
Introduction to Algorithms	03	45	Algorithm Lab using C	1	30
Operating System	03	45	Operating Systems Lab	1	30
Mathematical Methods	03	45	Mathematical methods Lab using Python	1	30

Theory: Data Communication and Networking
Credits - 03, Contact hours - 45.

Overview of Data Communication and Networking **Introduction:**Data communications Components, data representation, direction of data flow (simplex, halfduplex, full duplex). **Network Hardware:** Physical structure (type of connection, topology), categories of network (LAN, MAN, WAN). **Internet:** Brief history, Protocols and standards, **Reference models:** OSI reference model,properties of all the layers, TCP/IP reference model, their comparative study.	03 hours
Physical Layer **Data & Signals:** Analog & Digital Data and Signals, periodic and non-periodic signals, composite signals, bandwidth, bit rate, transmission of digital signals. **Transmission Impairments:** Attenuation, Distortion and Noise. **Data Rate Limits:** Noiseless Channel: Nyquist Data rate, Noisy Channel: Shannon's Capacity, calculation of data rate using both limits. **Digital Transmission** **Digital to Digital Conversion:** Line coding, schemes (RZ, NRZ, Manchester, DifferentialManchester), block coding. **Analog to Digital Conversion:** Sampling, Nyquist rate of sampling, Pulse code modulation(PCM), Delta Modulation (DM), Adaptive Delta Modulation (ADM), parallel and serialtransmission. **Analog Transmission** **Digital to Analog conversion:** Amplitude shift keying (ASK), Frequency Shift Keying (FSK), PhaseShift Keying (PSK), Quadrature Amplitude Modulation (QAM). **Analog to Analog Conversion (qualitative):**Amplitude Modulation (AM), Frequency Modulation (FM), Phase Modulation.	11 hours
Bandwidth Utilization Techniques Multiplexing: FDM, Synchronous & Statistical TDM, WDM.	03 hours
Transmission Media **Guided media:** Twisted pair, Coaxial, Fiber optics. **Unguided:** Radio waves, microwaves, Infrared, Antenna, Communication satellites (qualitative study only).	04 hours

Switching and Telephone network Circuit switched networks, Packet Switched networks, Virtual Circuit switch. Major components of telephone network, Dial up modem, DSL and ADSL modems, CableTV for data transfer (qualitative study only)	03 hours
Data link Layer Types of errors, framing (character and bit stuffing), error detection & correction methods, Linear and cyclic codes, checksum. **Protocols:** Stop & wait ARQ, Go-Back- N ARQ, Selective repeat ARQ, HDLC (qualitativestudy only). **Physical addressing:** MAC address and its format	03 hours
Medium Access sub layer Point to Point Protocol, **Token Ring:** Reservation, Polling. **Multiple access protocols:** Pure& Slotted ALOHA, CSMA, CSMA/CD, CSMA/CA. **Channelization:** FDMA, TDMA, CDMA (Qualitative study only). **Wired and Wireless LAN:** Standards, fast Ethernet, Protocol 802.11, Bluetooth.	05 hours
Network layer **Internetworking & devices:** Repeaters, Hubs, Bridges, Switches, Router, Gateway, **Addressing:** IP addressing, Subnetting, Routing techniques: static vs. dynamic routing, **Protocols:** RARP, ARP, IP, ICMP.	07 hours
Transport layer **Process to Process delivery:** UDP, TCP	02 hours
Application Layer Introduction to DNS, Remote logging, FTP, Electronic mail, WWW & HTTP.	04 hours

Text/ Reference Books

1. Data Communication and Networking, B.A. Forouzan, Tata McGraw Hill.
2. Computer Networks, A.S. Tanenbaum, Pearson Education.
3. Data and Computer Communication, W. Stallings, Pearson Education.
4. Data & Computer Communication,Black, PHI.
5. Internet & World Wide Web: How to program,Harvey M. Deitel& Paul J. Deitel.

Practical:Networking Lab
Credit: 01, Contact hour: 30.

1. Simulate Simplex, Half Duplex, and Full Duplex Communication Use serial and Ethernet ports to show directional data flow between two nodes.
2. Create LAN, MAN, and WAN Topologies Use appropriate devices (switches, routers, cloud) to build various category-wise networks.
3. Design Physical Topologies Implement star, bus, ring, and mesh topologies using appropriate devices.
4. Model Protocol Stack using TCP/IP Reference Model Layers Simulate communication between clients and servers, annotating each layer's activity.
5. Compare OSI vs TCP/IP Reference Models Create two scenarios showing layered encapsulation in both stacks and interpret packet flows.

6. Visualize Bandwidth and Bit Rate Concepts Use different cable types and configure link speeds to demonstrate throughput variations.
7. Simulate Signal Impairments Using Distorted Packets Introduce corruption manually and trace signal behavior across a noisy channel.
8. Compare Guided Media: Coaxial vs Fiber vs Twisted Pair Deploy multiple link types and measure latency and throughput across them.
9. Implement Multiplexing Models (FDM & TDM) Use time-sequenced communication across nodes to simulate multiplexed channel behavior.
10. Simulate Packet Switching Using Routers Design a scenario where data packets are rerouted based on availability.
11. Create a Virtual Circuit Setup Between Devices Use Label Switching or logical pathways to model a fixed route data exchange.
12. Demonstrate MAC Addressing and Frame Inspection Use simulation mode to trace MAC-level transmission and address resolution.
13. Simulate Error Detection Using Checksums and CRC Introduce errors and verify how they're flagged and corrected.
14. Frame Construction Using Bit and Character Stuffing Manual framing via scripts or event annotation showcasing stuffing techniques.
15. Compare CSMA/CD vs CSMA/CA Protocols Use wired Ethernet vs wireless 802.11 environments to simulate contention handling.
16. Create WLAN Using 802.11 Standard Devices Configure a wireless network with access points, laptops, and mobile nodes.
17. Simulate Bluetooth and PAN Communication Establish small-scale device-to-device networks to reflect low-energy protocol behavior.
18. Configure Static and Dynamic Routing Using RIP/EIGRP/OSPF Route between multiple routers with table inspection and simulation.
19. Implement IP Addressing and Subnetting Assign subnets to devices and verify connectivity across routed domains.
20. Simulate TCP and UDP Communication Use PCs and servers to demonstrate reliable vs unreliable transport with packet inspection.

Simulator:CISCO Packet Tracer

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

Theory: Introduction to Algorithms
Credits - 03, Contact hours - 45.

Introduction to Algorithms Definition, Characteristics, Recursive and Non-recursive algorithms	3 hours
Asymptotic Complexity Analysis of Algorithms Space and Time Complexity, Efficiency of an algorithm, Growth of Functions, Polynomialand Exponential Complexity, Asymptotic Notations: Big O Notation and Small o notation,Big Ω and Small ω, Big Θand Small φNotations, Properties: Best case/worst case/averagecase analysis of well-known algorithms.	6 hours
Algorithm Design Techniques Concepts and simple case studies of Greedy algorithms. Divide and conquer: Basic concepts, Case study of selected searching and sorting problems using divide and conquer techniques: Dynamic programming: General issues in Dynamic Programming.	12 hours

Graph Representation and Algorithm Graph traversal algorithms: BFS, DFS, Minimal spanning trees: Prim's Algorithm, Kruskal's Algorithm, Shortest path algorithms: Floyd's Algorithm, Floyd-Warshall Algorithm, Dijkstra's Algorithm, Graph Colouring Algorithms.	22 hours
Classification of Problems Concept of P, NP.	2 hours

Text/References Books

1. Introduction to Algorithms, Cormen, Leiserson, Rivest and Stein, TMH.
2. The Design and Analysis of Algorithms, Aho, Hopcroft and Ullman, Pearson Education.
3. The Art of Computer Programming, D.E. Knuth, Pearson Education.
4. Algorithm Design, Jon Kleiberg and Eva Tardos,Pearson Education.
5. Data Structures and Algorithms - K.Mehlhorn.
6. Computer Algorithms, S.Baase, Pearson Education.
7. Fundamentals of Computer Algorithms, E. Horowitz and Sahani, Galgotia
8. Combinational Algorithms- Theory and Practice, E.M. Reingold, J. Nievergelt and N. Deo,PHI.

Practical: **Graph algorithms Lab using C**
Credits - 01, Contact hours - 30.

Pre-requisite;

1. Programming Foundations in C
- Syntax, data types, loops, conditionals, and functions
- STL basics — especially vector, queue, stack, and map
- Dynamic memory and pointers (essential for building graph structures)

2. Graph Theory Fundamentals
- Terminology: vertices, edges, adjacency, degree, cycles
- Graph types: directed vs undirected, weighted vs unweighted
- Representations: adjacency matrix vs adjacency list

3. Math & Logic Readiness
- Basic combinatorics and logic reasoning
- Set theory (for understanding disjoint sets and connectivity)
- Matrices (helpful for adjacency matrix representations)

4. Compilers
- C compiler GCC

Laboratory exercises to be based on Graph Theory using C and based on the following;

Implementation of Graph algorithms: Single Spanning Tree Generation using - BFS, DFS, MinimalSpanning Tree Generation using - Prim's Algorithm, Kruskal’s Algorithm, Shortest Path finding usingFloyd's Algorithm, Floyd-Warshall Algorithm, Dijkstra's Algorithm, Graph Partitioning Algorithm.

Sample questions

1. Write a C program to generate a single spanning tree using Breadth-First Search (BFS). Trace and display visited nodes.

2. Develop a Depth-First Search (DFS)-based C algorithm to construct a spanning tree from a connected undirected graph. Compare traversal order with BFS.
3. Design and implement Prim's Algorithm using adjacency list and priority queue in C. Display the edges included in the MST and their weights.
4. Construct Kruskal's Algorithm using disjoint set union in C. Identify cycle prevention strategy and display final edge set of the MST.
5. Write a C program to compute the shortest paths from a single source using Dijkstra's Algorithm. Provide output trace with distances and paths.
6. Implement Floyd's Algorithm to find all-pairs shortest paths in a weighted graph using adjacency matrix representation. Analyze time complexity.
7. Demonstrate Floyd-Warshall Algorithm with intermediate node storage. Display final distance matrix and reconstructed paths.
8. Apply a basic graph partitioning strategy using BFS clustering in C. Divide the graph into subgraphs and count nodes in each.
9. Implement a heuristic-based Graph Partitioning Algorithm in C and evaluate inter-partition edge cuts. Visualize the result if possible.

Note: ***The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.***

Theory: Operating System
Credits - 03, Contact hours - 45.

Introduction Basic OS functions, types of operating systems- batch processing, multiprogramming, timesharing, multiprocessing, distributed and real time systems.	5 hours
Operating System Organization Processor and user modes, kernels, system calls and system programs.	5 hours
Process System view of the process and resources, process control block, I/O and CPU boundprocess, process hierarchy, concept of threads. Process Scheduling: Preemptive and non-preemptive scheduling, Long term scheduling,short term/CPU scheduling (FCFS, SJF, SRJF, RR and priority) and medium-termscheduling Process Synchronization: Concurrent processes, critical section, semaphores andapplication, methods for inter-process communication.	15 hours
Deadlock: Definition, Prevention, Avoidance, Detection, Recovery.	8 hours
Memory Management Physical and logical address space; memory allocation strategies – fixed and variablepartitions, paging, segmentation, virtual memory.	7 hours
Memory Management Physical and logical address space; memory allocation strategies – fixed and variablepartitions, paging, segmentation, virtual memory.	3 hours
Protection and Security Policy mechanism, Authentication.	2 hours

Text/ Reference Books

1. Operating Systems Concepts, A Silberschatz, P.B. Galvin, G. Gagne, John Wiley Publications.
2. Modern Operating Systems, A.S. Tanenbaum, 3rd Edition, Pearson Education.
3. Operating Systems: A Modern Perspective, G. Nutt, Pearson Education.
4. Operating Systems, Internals & Design Principles W.Stallings, PHI.
5. Operating Systems- Concepts and design, M. Milenkovic, Tata McGraw Hill.
6. UNIX Concepts and Applications, Sumitabha Das, Tata McGraw-Hill.
7. Understanding the Linux Kernel, D. P. Bovet and M. Cesati, O'Reilly.

Practical: Operating System Lab
Credits - 01, Contact hours - 30.

1. Write a shell script to convert the content of a file from lower case to upper case.
2. Write a shell script to count the words, lines and characters of a given file. File name should beprovided at run time.
3. Write a shell script that take a word from user and find out the frequency of the word in a givenfile.
4. Write a shell script that gets executed at the moment of user login and it displays Good Morning,Good afternoon, Good Evening, Good Night, depending upon the time at which the user logs on.
5. Write a shell script to print Pascal diamond.
6. Write a shell script to find a number using sequential search method.
7. Write a shell script to find a number using binary search technique.
8. Write a shell script to sort a set of integer numbers using bubble sort.
9. Write a shell script to find out the factorial of a given number.
10. Write a shell script to reverse a string and check whether it is a palindrome.
11. Write a shell script to find the roots of a quadratic equation ax2 + bx +c = 0, considering allpossible cases.
12. Write a shell script for menu-based system to insert records for employees with employee ID,name, designation, salary in a data file, also display records when necessary. Display salary forthe employee asked.

Note: ***The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.***

Theory: Mathematics methods
Credits - 03, Contact hours - 45.

Introduction Set Theory: Finite and Infinite Sets, Uncountable Infinite Sets, Relations: Properties ofBinary Relations, Closure, Partial Ordering Relations, Equivalence, Functions: definition, one-to-one, onto and invertible, Mathematical Functions: Exponential and Logarithmic,Counting: Mathematical Induction, Pigeonhole Principle, Permutation and Combination, Binomial Theorem, Principle of Inclusion and Exclusion.	08 hours
Introduction to Probability Elementary events, Sample space, Classical and Axiomatic definition of Probability,Theorems on Total Probability, Conditional Probability, Bernoulli Trials and BinomialDistribution, Bayes' Theorem, Random Variables, Expectation, Variance, StandardDeviation.	07 hours
Growth of Functions Asymptotic Notations, Standard notations and common functions with simple examples.	03 hours
Recurrences Relations, Generating Functions, Linear Recurrence Relations with Constant Coefficients and their solution, Substitution Method, Recurrence Trees.	04 hours
Numerical Methods (Algorithmic Approach) **Errors:** Approximate and Rounding of Numbers, Significant digits, Errors and their types, Propagation of errors. **Interpolation:** Newton Forward and Backward interpolation, Lagrange interpolation. **Solving a Set of Linear Equations:** Gaussian Elimination, Gauss–Jordan, Iteration methods and their convergence conditions, Gauss-Seidel, Gauss-Jacobi Iterative Methods. Solving Non-linear equations: Bisection, Regula-falsi, Secant and Newton-Raphson, their order of convergence. **Solving Differential Equations:** Euler, Runge-Kutta second and fourth order methods. Numerical Integration: Trapezoidal and Simpson's 1/3rd rules. **Curve fitting:** Least square approximation, Linear regression, Polynomial regression, Fitting Exponentialand Trigonometric functions.	14 hours
Graph Theory Basic Terminology, Models and Types, Multi graphs and Weighted graphs, GraphRepresentation, Graph Isomorphism, Connectivity, Euler and Hamiltonian Paths andCircuits, Planar Graphs, Trees and their basic terminologies and properties.	09 hours

Text/ Reference Books

1. Elements of Discrete mathematics, C.L. Liu & Mahapatra, Tata McGraw Hill.
2. Discrete Mathematics and Its Applications, Rosen, McGraw Hill.
3. Introduction to algorithms, T.H. Cormen, C.E. Leiserson, R. L. Rivest, Prentice Hall.

4. Discrete Mathematics with Algorithms, M. O. Albertson and J. P. Hutchinson, JohnWiley Publication.
5. Discrete Structures, Logic, and Computability, J. L. Hein, Jones and Bartlett Publishers.
6. Essentials of Discrete Mathematics, D.J. Hunter, Jones and Bartlett Publishers.
7. Numerical Analysis and Computational Procedures by Mollah, New Central Book.
8. Computer Oriented Numerical Methods, 3rd Edition, V Rajaraman, PHI
9. Graph Theory with Applications to Engineering and Computer Science by NarsinghDeo, PHI.
10. Graph Theory by J.A. Bondy and U.S.R. Murty, Springer.
11. Introduction to Graph Theory by D B West, 2nd edition, Pearson Education

Practical: Mathematical Methods Lab using Python.
Credits - 01, Contact hours - 30.

1. Set Operations: Implement union, intersection, difference, and symmetric difference on finite sets using Python's set data type.
2. Countable vs Uncountable Sets: Simulate countable sets (e.g., integers) and discuss uncountable sets (e.g., real numbers) using generator functions.
3. Binary Relations: Represent relations as sets of ordered pairs and check properties: reflexivity, symmetry, transitivity.
4. Equivalence and Partial Order: Identify equivalence classes and Hasse diagrams using adjacency matrices.
5. Function Properties: Write Python functions to test one-to-one, onto, and invertibility using mappings.
6. Sample Space Generator: Create sample spaces for coin tosses, dice rolls, and card draws.
7. Classical Probability Calculator: Compute probabilities using combinatorics (e.g., probability of drawing a red card).
8. Bayes' Theorem Simulation: Implement conditional probability and Bayes' rule with real-world examples.
9. Bernoulli Trials and Binomial Distribution: Simulate trials and plot binomial distribution using matplotlib.
10. Random Variables and Expectation: Generate discrete random variables and compute expectation, variance, and standard deviation.
11. **Asymptotic Notation Visualizer**: Plot functions like n, nlogn, n^2 2^n to compare growth rates.
12. **Recurrence Solver**: Implement substitution and recurrence tree methods to solve recurrence relations.
13. **Generating Functions**: Use symbolic computation (sympy) to derive generating functions for sequences.
14. **Error Analysis**: Compute absolute, relative, and percentage errors for approximated values.
15. **Interpolation Techniques**: Implement Newton Forward/Backward and Lagrange interpolation using tabular data.
16. **Solving Linear Systems**: Code Gaussian Elimination, Gauss-Jordan, and iterative methods (Jacobi, Gauss-Seidel).
17. **Root-Finding Algorithms**: Implement Bisection, Regula-Falsi, Secant, and Newton-Raphson methods with convergence checks.
18. **ODE Solvers**: Apply Euler and Runge-Kutta (2nd and 4th order) methods to solve first-order differential equations.
19. **Numerical Integration**: Implement Trapezoidal and Simpson's 1/3rd rule for definite integrals.
20. **Curve Fitting**: Use least squares to fit linear, polynomial, exponential, and trigonometric curves to data.
21. **Graph Representation**: Build graphs using adjacency lists and matrices; visualize with networkx.
22. **Graph Traversals**: Implement BFS and DFS; trace visited nodes and paths.

23. **Euler and Hamiltonian Paths**: Check for Eulerian and Hamiltonian circuits in undirected graphs.
24. **Planar Graph Checker**: Use Kuratowski's theorem heuristics to test planarity.
25. **Tree Properties**: Implement tree traversal algorithms and check properties like height, leaf count, and degree.

Software tool/Compiler:"Miniconda or Anaconda can be used to set up the Python environment, and Jupyter Notebook can be used for Python programming.

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

Semester - 6

Theory Paper	Credit	Contact hours	Practical/Tutorial Paper	Credit	Contact hours
Introduction to Machine Learning.	03	45	Machine learning Lab	1	30
Embedded Systems	03	45	Embedded System Labs	1	30
Software Engineering	03	45	Software Engineering Lab	1	30

Theory: Introduction to Machine Learning
Credits - 03, Contact hours - 45.

Introduction to Machine Learning Definition and scope of ML, Types of learning: Supervised, Unsupervised, Reinforcement, Applications in computer vision, NLP, and business analytics, Python libraries: NumPy, Pandas, Scikit-learn.	06 hours
Data Preprocessing and Feature Engineering Data cleaning, normalization, encoding, Handling missing values and outliers, Feature selection and dimensionality reduction, Introduction to PCA.	07 hours
Supervised Learning Algorithms Linear Regression and Logistic Regression, Decision Trees and Random Forests, Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), Evaluation metrics: Accuracy, Precision, Recall, F1-score.	11 hours
Unsupervised Learning Algorithms Clustering: k-Means, Hierarchical, Association rule mining: Apriori algorithm, Dimensionality reduction: PCA, t-SNE.	07 hours
Probability and Statistical Foundations Probability distributions: Normal, Binomial, Poisson, Bayes' Theorem and Naive Bayes Classifier, Expectation, variance, and standard deviation, Hypothesis testing basics.	07 hours
Neural Networks and Deep Learning (Introductory) Perceptron and multilayer networks, Activation functions, Introduction to TensorFlow/Keras, Simple feedforward network for classification.	07 hours

Text/ Reference Books

1. Introduction to Computation and Programming Using Python: With Application to UnderstandingData, Guttag, John V. MIT Press.
2. Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World ofComputers and Code, Shaw, Zed A, Addison-Wesley Professional.
3. Think Python 2e. Green Tea Books, Downey, Allen B.
4. Practical Programming: An Introduction to Computer Science Using Python 3.6. PragmaticBookshelf, Gries, Paul, Jennifer Campbell, and Jason Montojo.
5. Introduction to Python Programming, Gowrishankar S, Veena A, Taylor and Francis, CRC Press.
6. Python Cookbook, David Beazley and Brian K. Jones, O'Reilly.

Practical: Machine Learning Lab using Python

Credits - 01, Contact hours - 30.

Lab 1: Data Exploration and Visualization
Objective: Load a dataset, perform basic statistics, and visualize features. Task:

- Load the Iris dataset using sklearn.datasets
- Use Pandas to display summary statistics
- Plot histograms, boxplots, and scatter plots using Matplotlib/Seaborn Outcome: Understand data distribution and feature relationships.

Lab 2: Linear Regression
Objective: Predict continuous values using regression. Task:

- Use the Boston Housing dataset
- Train a Linear Regression model using sklearn.linear_model
- Evaluate using Mean Squared Error and R^2 score Outcome: Learn regression modelling and performance metrics.

Lab 3: Logistic Regression for Classification
Objective: Perform binary classification. Task:

- Use the Titanic dataset (Kaggle)
- Predict survival using logistic regression
- Evaluate with confusion matrix and ROC curve Outcome: Understand classification and model evaluation.

Lab 4: Decision Trees and Random Forests
Objective: Apply tree-based models for classification. Task:

- Use the Iris or Loan Prediction dataset
- Train Decision Tree and Random Forest models
- Visualize the tree structure Outcome: Compare tree-based classifiers and interpret decisions.

Lab 5: k-Nearest Neighbours (k-NN)
Objective: Implement instance-based learning. Task:

- Use the Wine dataset
- Train a k-NN classifier and tune k
- Evaluate using accuracy and confusion matrix Outcome: Learn distance-based classification.

Lab 6: k-Means Clustering
Objective: Perform unsupervised clustering. Task:

- Use the Mall Customer Segmentation dataset
- Apply k-Means and visualize clusters
- Use Elbow method to choose optimal k Outcome: Understand clustering and intra-cluster variance.

Lab 7: Principal Component Analysis (PCA)
Objective: Reduce dimensionality and visualize data. Task:
- Use the Breast Cancer dataset
- Apply PCA and reduce to 2D
- Plot principal components Outcome: Learn feature reduction and variance preservation.

Lab 8: Naive Bayes Classification
Objective: Apply probabilistic classification. Task:
- Use SMS Spam Collection dataset
- Preprocess text and train Naive Bayes model
- Evaluate precision and recall Outcome: Understand Bayes' theorem in classification.

Lab 9: Neural Network with Keras
Objective: Build a simple feedforward neural network. Task:
- Use MNIST or Fashion-MNIST dataset
- Train a model using TensorFlow/Keras
- Plot training accuracy and loss Outcome: Learn basics of deep learning and model tuning.

Lab 10: Mini Project
Objective: Apply end-to-end ML pipeline. Task:
- Choose a dataset (e.g., student performance, diabetes prediction)
- Perform preprocessing, model training, evaluation, and visualization Outcome: Demonstrate complete ML workflow and reporting.

Software Tool/Compiler: - Python (Miniconda/Anaconda)

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

Theory: Embedded Systems
Credits - 03, Contact hours - 45.

Introduction to Embedded Systems Definition, characteristics, and applications, Embedded vs general-purpose systems, Microcontrollers vs microprocessors, Real-time systems and constraints.	**02 hours**
Microcontroller Architecture & Programming 8-bit (8051), Memory organization (Data and Program memory), Special function registers, I/O port registers and architecture, Instruction set, timers, interrupts, Serial communications (qualitative only), Assembly Language programming.	**10 hours**
Arduino Platform & Interfacing Arduino architecture and toolchain, Programming essentials, Digital and analog I/O, PWM, serial communication, **Sensor interfacing:**temperature, humidity, pressure, soil moisture motion, light, gas sensors, location sensors (GPS), display interfacing (LED, LCD and DOT matrix), **Actuator control:**motors, buzzers.	**09 hours**

Raspberry Pi & Python Programming Introduction to Raspberry Pi architecture(qualitative) and GPIO access, Raspbian Operating system (Installation and basic GPIO configuration), Python scripting, Interfacing sensors and actuators via GPIO, introduction to Node-Red.	**07 hours**
Communication Protocols & IoT Integration Basicsof UART, SPI, I2C, **Wireless protocols:**Bluetooth, Wi-Fi, ZigBee, Cloud connectivity and MQTT basics.Sending sensor data to cloud using Raspberry Pi or ESP8266, use of NODE-RED for interfacing and IoT.	**06 hours**
Embedded System Design & Applications **Design flow:**requirement → architecture → implementation, **Case studies:**Smart agriculture, wearable health monitors, Smart home automation, Smart Farming, Smart grid, and Smart cities.	**04 hours**
Embedded AI Fundamentals (Introductory) **What is Embedded AI:** Basic idea of integrating AI into small devices (e.g., sensors, microcontrollers), **Edge vs Cloud AI:** Simple comparison of local (edge) vs remote (cloud) AI processing, **Toolchain Overview**: Brief mention of STM32CubeMX and STM32Cube.AI as tools for configuring and deploying AI on STM32 microcontrollers. **Neural Networks for Embedded Use (Elementary)** **Basic Concepts:** Introduction to neural network structure—layers, activation, and inference, **Model Creation:** Simple overview of building models using Python/Keras, **Optimization:** Concept of making models smaller and faster (quantization) for low-power devices. **AI Deployment with STM32 –(Elementary)** **Deployment Flow:** High-level steps to convert a trained model into embedded code using STM32 tools, **Use Case Examples:** Mention of simple applications like gesture recognition or sensor-based classification.	**07 hours**

References:

1. An Embedded software primer,David E.Simon, Pearson Education.
2. The 8051 Microcontroller, Kenneth J. Ayala, Thomson.
3. Embedded Systems, Raj Kamal, TMH.
4. Microcontroller, Raj Kamal, Pearson Education.
5. **The 8051 Microcontroller Based Embedded Systems,** Manish K. Patel, McGraw Hill Education (India).
6. The 8051 Microcontroller and Embedded Systems, Muhammad Ali Mazidi.
7. STM32F103 Arm Microcontroller and Embedded Systems, Sarmad Naimi, Muhammad Ali Mazidi, Sepehr Naimi, Microdigitaled.
8. **Embedded Systems and IoT,** Dr. Ratna Kamala Petla, Cengage India.
9. **Embedded System Design,** Santanu Chattopadhyay, PHI Learning.
10. TM32 Embedded Systems Design: Definitive Reference for Developers andEngineers, Richard Johnson, HiTeX Press; PublishDrive edition.
11. Online resources from ST electronics: https://wiki.st.com/stm32mcu/wiki/STM32StepByStep:Getting_started_with_STM32_:_STM32_step_by_step.

Practical: Embedded System Lab
Credits - 01, Contact hours - 30.

Laboratory Assignments on 8951 series microcontrollers using Keil

1. Copy a block of 20 bytes of data available from address 60H to 73H to the location starting from 40H.
2. Shift a block of 8 bytes of data, presently located from 50H to 57H, 1 byte up, so that the data is availablefrom 51H to 58H.
3. Twenty bytes of data are stored in location from 7FH to 6CH of internal RAM. Count the number of thosebytes, which contain 00H, and store this number of null bytes in RAM location 6BH.
4. Sixteen consecutive bytes starting from 50H have unsigned integers. Develop a program to add all these 16integers and store the 8-bit sum in memory location 60H.
5. Write a program to generate and store natural numbers starting from 1 to 'N' terms and also find the sum ofthese numbers. Assume that the value of 'N' is stored in location 30H. Store generated natural numbers from40H. Leave the sum in the accumulator.
6. Write a program to find the sum of the series 1 − 2 + 3 − 4 + … up to N terms. Assume the non-zero value ofN is available in location 30H. Store the sum in the accumulator.
7. Generate Fibonacci series up to N terms and save from location 50H onwards. Assume N being available inlocation 4FH. Also assume the value of N to be greater than 3.
8. Generate and store the following series up to N terms. Value of N is available in location 30H. The series ispresented using decimal number system.1, 2, 3, 11, 12, 13, 21, 22, 23, 31, … up to N terms.
9. Two arrays, each of 16 bytes, are stored from 40H to4FH and 50H to 5FH. Write a program to comparethese two arrays and store 00H in location 30H if theyare identical (same number and in same sequence).Otherwise, store a non-zero value in location 30H.
10. Sixteen random numbers are stored in location 40Hto 4FH. Write a program to take out alternate numbers,starting from the first number, and create asecond array of eight numbers from 50H to 57H.
11. Extend the previous program so that the first arrayis compacted within 8 bytes from 40H to 47H aftershifting eight of its entries.
12. Sixteen random numbers are stored in an array,starting from location 40H. Write a program tocount the number of non-zero elements in thisarray and store it in location 30H.
13. Find out number of 1s in a byte, available in internal data memory location 30H. Store the result in theaccumulator itself.
14. A packed BCD number is in location 30H. After unpacking, store it in R6 (lower digit) and R7 (higherdigit).
15. Two unpacked BCD digits are available in location 30H (MS digit) and 31H (LS digit). Write a program topack these two and store in R7.
16. Using only one pointer, R0, pack two arrays of BCD digits to create a third array. Th e higher digits are availablefrom 30H to 3FH. Lower digits are available from 40H to 4FH. Packed BCD numbers are to be storedfrom 50H to 5FH.
17. Find the largest and smallest from N unsigned integers. Assume the value of N to be available in the internaldata memory location 30H. The array starts from location 31H. Store the maximum integer in R4 and minimumin R3.
18. A few unsigned integers are stored from the location 31H onwards of the internal data memory area.Arrange these integers in ascending order. Th e number of terms of the array is available in the location 30H.Store the arranged integers from 31H itself.
19. A few random unsigned integers are stored from the internal data memory location 31H onwards. Numberof terms (N) is available in location 30H. Assuming that none of these

numbers is greater than 5, find thefactorials of these integers and then find their sum. Assume that the sum would not exceed 8-bit value.

20. Create a new array by removing only those integers that are perfectly divisible by 4 from an array, startingfrom 31H. Location 30H contains number of terms of this array. Th e new array is to be created from the location60H. At return, the accumulator should indicate number of terms found. Original locations with digitsdivisible by 4 should be replaced by null.
21. Register R4 would be loaded by an unsigned integer, which may vary from 0 to 5. Find the square of theinteger and store it in R5.
22. Sixteen-bit values of the square of integers arestored in a table that starts from program memory
23. location 0200H onwards. The lower byte of the 16-bit number is at the lower address, and the higher byte is at the higher address. Assuming that theaccumulator contains the integer whose squareis required, write a routine to get the result usingtable look-up procedure (MOVC instruction.

Laboratory Assignments on Arduino

24. Develop a program to generate dynamic LED patterns using Arduino's digital output pins and timed logic.
25. Interface a 16×2 LCD with Arduino to display real-time messages or sensor data in formatted text. 26. Use a dot matrix display to scroll custom messages by interfacing it with Arduino and controlling animation frames.
26. Read pressure values from a BMP180 or BMP280 sensor and display the readings using Arduino's serial monitor or LCD.
27. Measure ambient temperature and humidity using a DHT11 sensor and display the results on an LCD.
28. Detect human motion using a PIR sensor and trigger an LED or buzzer response through Arduino.
29. Measure the distance of nearby objects using the HC-SR04 ultrasonic sensor and display the result on a screen.
30. Detect proximity using an IR sensor and activate an output device such as an LED or buzzer when an object is nearby.
31. Interface a GPS module with Arduino to read and display latitude and longitude coordinates in real time.
32. Measure orientation and acceleration using the MPU6050 sensor and process the data through Arduino.
33. Detect ambient sound levels using a microphone sensor and trigger alerts when the sound exceeds a threshold.
34. Monitor air quality by interfacing an MQ-series gas sensor with Arduino and display gas concentration levels.
35. Use a soil moisture sensor to monitor plant health and trigger a visual or audible alert when moisture drops below a set level.
36. Control the angle of a servo motor using a potentiometer input and map the analog value to servo rotation.
37. Interface a stepper motor with Arduino and control its rotation direction and step count for positioning applications.

Laboratory Assignments using Raspberry Pi, NODE-RED and Python

38. Use Node-RED to control an LED connected to a GPIO pin and toggle its state via a dashboard button.
39. Write a Python script to blink an LED at regular intervals using GPIO output control.

40. Interface a push button with Raspberry Pi and use Node-RED to display its pressed/released status on a dashboard.
41. Create a Python program to read digital input from a push button and toggle an LED accordingly.
42. Use Node-RED to read temperature and humidity data from a DHT11 sensor and visualize it using a gauge widget.
43. Write a Python script to interface a PIR sensor and trigger a buzzer when motion is detected.
44. Interface an ultrasonic sensor using Python to measure distance and display the result on the terminal.
45. Use Node-RED to read distance data from an HC-SR04 sensor and plot it on a real-time chart.
46. Create a Python program to read analog light intensity using an LDR and an ADC module, then display the value.
47. Use Node-RED to monitor soil moisture levels and trigger an alert when the value drops below a threshold.
48. Interface a gas sensor using Python and display air quality status based on analog readings.
49. Use Node-RED to read GPS coordinates from a serial-connected GPS module and display location data on a map.
50. Write a Python script to read acceleration and orientation data from an MPU6050 sensor and log it to a CSV file.
51. Use Node-RED to control a servo motor's angle based on slider input from the dashboard.
52. Create a Python program to rotate a stepper motor in both directions using GPIO pins and delay logic.

Laboratory Assignments using STM32

53. Write a program to blink an LED connected to a GPIO pin using STM32's internal delay functions.
54. Configure a push button as digital input and toggle an LED each time the button is pressed.
55. Use Timer 2 to generate a precise delay and blink an LED at a fixed interval.
56. Interface a 16×2 LCD with STM32 and display a static message using GPIO and delay logic.
57. Read analog voltage from a potentiometer using ADC and display the value on the serial monitor.
58. Generate PWM signals using Timer 3 and control the brightness of an LED.
59. Interface a DHT11 sensor and display temperature and humidity readings via UART.
60. Measure distance using an HC-SR04 ultrasonic sensor and display the result on an LCD.
61. Interface a servo motor and control its angle using PWM output from STM32.
62. Configure external interrupt on a GPIO pin to trigger an LED when a falling edge is detected.

Tools/Simulator/Programming Language: PYTHON, Rasbian, Node-Red(Pre installed in Rasbian OS of Raspberry Pi).

Development board/single board-Development board based on 8951/52, Arduino Uno/MEGA, Single board computers: Raspberry Pi 3+ or higher and STM32.

Sensors required: Ultrasonic sensor (HC-SR04), PIR Sensor, IR Proximity Sensor, **Acceleration and tilt sensing and Gyroscope:**ADXL345, GPS Module (NEO-6M), Bluetooth (HC-05/HC-06), Wi-Fi (ESP8266/ESP32), LoRa Module, DHT11/DHT22, BMP180/BMP280, LM35/DS18B20, Air quality sensor-MQ 135.LCD 16 x 2, DOT Matrix display, and LED etc.

Miscellaneous: Battery, adaptors, connecting wires, and breadboard etc.

Note: ***The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.***

Theory: Software Engineering
Credits - 03, Contact hours - 45.

Introduction Defining system, open and closed system, modelling of system through computerhardware, communication systems, external agents and software systems; Importance of Engineering Methodology towards computerization of a system.	3 hours
Software Life Cycle Classical and Iterative Waterfall Model; Spiral Model; Prototype Model; Evolutionary model and its importance towards application for different system representations, Comparative Studies.	6 hours
Software Requirement and Specification Analysis Requirements Principles and its analysis principles; Specification Principles and itsRepresentationsSoftware Design Analysis – Different level of DFD Design, Physical and Logical DFD,Use and Conversions between them, Decision Tables and Trees, Structured analysis,Coupling and Cohesion of different modulesSoftware Cost Estimation Modelling –COCOMO.	15 hours
Software Testing Software Verification and Validation; Testing objectives, Testing Principles, Testability;Error and Faults; Unit Testing, White Box and Blank Box Testing, Test Case Design:Test Vector, Test Stub.	15 hours
Software Quality Assurances Concepts of Quality, Quality Control, Quality Assurance, IEEE Standard for Statistical Software Quality Assurances (SSQA) criterions.	06 hours

Text/References Books

1. Software Engineering: A Practitioner's Approach by R.S. Pressman, McGraw-Hill.
2. An Integrated Approach to Software Engineering by P. Jalote, Narosa Publishing House.
3. Software Engineering by K.K. Aggarwal and Y. Singh, New Age International Publishers.
4. Software Engineering by I. Sommerville, Addison Wesle.
5. Software Engineering for Students by D. Bell, Addison-Wesley.
6. Fundamentals of Software Engineering by R. Mall, PHI.

Practical: Software Engineering Lab
Credits - 01, Contact hours - 30.

Lab 1: Getting Started with Gaphor.
Objective: Familiarize students with Gaphor's interface and basic UML elements Tasks:

- Install Gaphor via pip install gaphor
- Create a new model and diagram
- Add basic elements: Class, Interface, Package
- Save and export the diagram as PNG Outcome: Students understand the modelling environment and diagram setup.

Lab 2: Class Diagram – Library Management System
Objective: Model object-oriented relationships **Tasks**:

- Create classes: Book, Member, Librarian, Transaction
- Define attributes and methods
- Show associations, aggregations, and multiplicity **Outcome**: Students grasp encapsulation, inheritance, and object interaction.

Lab 3: Use Case Diagram – Online Shopping Portal
Objective: Capture functional requirements **Tasks**:

- Identify actors: Customer, Admin, Payment Gateway
- Define use cases: Browse, Order, Pay, Track
- Connect actors to use cases **Outcome**: Students visualize system behaviour from user perspectives

Lab 4: Activity Diagram – ATM Withdrawal Process
Objective: Model control flow and decision logic **Tasks**:

- Create an activity diagram for Withdraw Cash
- Include decision nodes, actions, and swimlanes
- Annotate with guard conditions **Outcome**: Students understand process modelling and branching logic

Lab 5: State Machine Diagram – Traffic Light Controller
Objective: Model reactive systems Tasks:

- Define states: Red, Green, Yellow
- Add transitions and events
- Simulate timing and control logic Outcome: Students explore state-based behavior and event-driven design.

Lab 6: Component Diagram – Web Application Architecture
Objective: Visualize modular system design **Tasks**:

- Identify components: Frontend, Backend, Database, API
- Show dependencies and interfaces
- Annotate with protocols (e.g., HTTP, SQL) **Outcome**: Students learn modular design and deployment structure.

Lab 7: Package Diagram – Python Project Structure
Objective: Organize codebase and modules **Tasks**:

- Create packages: utils, models, controllers, views
- Show dependencies and containment
- Reflect on Python file structure **Outcome**: Students connect UML with actual Python project organization.

Software Tool/Compiler: Gaphor.

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*