

Jharkhand Raksha Shakti University

Meur's Road SKIPA Premises, Ranchi



Detailed Syllabi (Second and Third Year)

of

B.Sc. (Hons.) Programme

in

Computer Application and Cyber Security (FYUGP)

(W.e.f. from Academic Year 2025-26)

DEPARTMENT OF CYBER DEFENCE

Ranish *Prasanna* *Atyain* *SSG* *Shuchi Pring* *Sh*

Course Title: Data Communication and Computer Network	
Course Code: FYCS-C301	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To develop a strong understanding of the fundamental concepts of data communication models and communication architectures.
2. To examine the characteristics of various communication media and analyze how signals propagate through them, including the impact of transmission impairments.
3. To comprehend the core principles of signal encoding, as well as error detection and error correction techniques.
4. To explore methods of flow control and multiplexing aimed at optimizing bandwidth utilization in data communication systems.
5. To gain an understanding of advanced concepts and techniques involved in efficient and reliable data communication.

Course Outcomes (COs):

After successful completion of this course, students will be able to:

CO1: Explain the fundamental concepts of data communication models and communication architectures.

CO2: Analyze the characteristics of different communication media and evaluate signal propagation along with transmission impairments.

CO3: Apply signal encoding techniques and demonstrate understanding of error detection and error correction methods.

CO4: Illustrate and implement flow control and multiplexing techniques to improve bandwidth utilization.

CO5: Evaluate and apply advanced concepts for designing efficient and reliable data communication systems.

Syllabus

Module I

[08 Hours]

Data Communications: Components, Data Representation, Data Flow; Networks: Network Criteria, Physical Structures; Network Types: Local Area Network, Wide Area Network, Switching, The Internet; Protocol Layering: Principles of Protocol Layering, Logical Connections; TCP/IP Protocol Suite: Layered Architecture, Layers in the TCP/IP Protocol Suite, Encapsulation and Decapsulation, Addressing, Multiplexing and Demultiplexing; The OSI Model: OSI versus TCP/IP

Rajiv *Deepak* *Ajay* *SSK* *Shuchi Pring*

Module II

[08 Hours]

Data and Signals; Periodic Analog Signals; Digital Signals; Transmission Impairment: Attenuation, Distortion, Noise; Data Rate Limits: Noiseless Channel: Nyquist Bit Rate, Noisy Channel: Shannon Capacity; Performance: Bandwidth, Throughput, Latency, Bandwidth-Delay Product, Jitter; Transmission Media; Unguided Media: Radio Waves, Microwaves, Infrared

Module III

[08 Hours]

Digital-to-Digital Conversion; Analog-to-Digital Conversion; Transmission Modes: Parallel Transmission, Serial Transmission; Digital-to-Analog Conversion; Analog-to-Analog Conversion

Module IV

[08 Hours]

Introduction to Data-Link Layer; Link-Layer Addressing; Error Detection and Correction; Cyclic Redundancy Check (CRC); Data Link Control (DLC); Data-Link Layer Protocols; HDLC; Point-to-Point Protocol (PPP); Media Access Control (MAC); Controlled Access; Channelization; Multiplexing; Switching: Circuit-Switched Networks, Packet Switching: Datagram Networks, Virtual-Circuit Networks

Module V

[08 Hours]

Network-Layer Services; Packet Switching; Network-Layer Performance; IPv4 Addresses; Forwarding of IP Packets; Internet Protocol (IP); ICMP; Unicast Routing; Unicast Routing Protocols; IPv6 Addressing; Transition from IPv4 to IPv6

Module VI

[08 Hours]

Functions of transport layer, Process-to-process communication, Multiplexing and demultiplexing, Connection-oriented vs connectionless services, Protocols: TCP and UDP. Principles of network applications, Client-server and peer-to-peer models, Socket interface basics, Application protocols: HTTP/HTTPS, FTP, SMTP, DNS, DHCP

Text Books:

1. Forouzan, B. A., Data Communications and Networking, 5th Edition, McGraw-Hill.

References:

1. Stallings, W., Data and Computer Communications, Pearson.
2. Forouzan, B. A., TCP/IP Protocol Suite, McGraw-Hill.
3. Tanenbaum, A. S., Computer Networks, Pearson.



Practical's:

1. Study of different types of network cables (UTP, coaxial, fiber optic) and their specifications.
2. Creation of LAN using networking devices (hub, switch, router) and IP configuration.
3. Implementation of basic networking commands (ping, ipconfig/ifconfig, tracert, netstat).
4. Simulation of OSI model using networking tools/software.
5. Study and implementation of different line coding techniques.
6. Simulation of error detection techniques (Parity Check, CRC).
7. Implementation of Stop-and-Wait and Sliding Window protocols using C/Python.
8. Simulation of CSMA/CD and CSMA/CA protocols.
9. Study of multiplexing techniques using simulation tools.
10. Implementation of shortest path routing algorithm (Dijkstra's Algorithm).
11. Configuration of basic routing using Packet Tracer/NS2/NS3.
12. Study of IP addressing and subnetting exercises.

Lawin *Shamir* *Atyain* *SSR* *Sanchi Singh* *[Signature]*

Course Title: Database Management System	
Course Code: FYCS – C302	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

The course aims to:

1. Understand the fundamental concepts, historical perspectives, current trends, structures, operations, and functions of different components of database systems.
2. Recognize the importance of database analysis and design in the implementation of database applications.
3. Describe the role of transaction processing in database systems.
4. Understand various concurrency control mechanisms used in database systems.
5. Explain the roles of recovery and security in database systems.

Course Outcomes (COs):

After successful completion of this course, students will be able to:

CO1: Analyze data organization requirements and their interrelationships.

CO2: Illustrate the features of different data models and their applications for data storage.

CO3: Design and execute queries to retrieve and maintain data effectively in databases.

CO4: Analyze physical database design and evaluate performance using normalization and query processing techniques.

CO5: Examine best practices related to indexing, transaction management, and concurrency control in database systems.

Syllabus

Module I

[09 Hours]

Purpose of Database System, View of Data, Database Languages, Transaction Management, Database Architecture, Database Users and Administrator, Types of Database System; Overview of Design Process, E-R Model, Constraints, E-R Diagram, E-R Diagram Issues, Weak Entity Sets, Extended E-R Features, Reduction to E-R Schemas

Module II

[09 Hours]

Structure of Relational Database, Codd's Rules, Fundamental Relational Algebra Operations, Additional Relational Algebra Operations, Extended Relational Algebra Operations; Data Definition, Basic Structure of SQL Queries, Set Operations, Aggregate Functions, Null Values, Nested Sub Queries, Complex Queries, Views, Modification of Database, Joined

[Signatures]

Relations, SQL Data Types and Schemas, Integrity Constraints, Authorization, Embedded SQL, Triggers

Module III

[09 Hours]

Functional Dependency, Decomposition, Normalization, First Normal Form, Second Normal Form, Third Normal Form, BCNF, Multivalued Dependencies and Fourth Normal Form, Join Dependencies and Fifth Normal Form, DKNF

Module IV

[09 Hours]

Ordered Indices, B+ Tree Index Files, B-Tree Index Files, Multiple Key Access, Static Hashing, Dynamic Hashing, Comparison of Ordered Indexing and Hashing, Index Definition in SQL; Measure of Query Cost, Selection Operation, Evaluation of Expressions

Module V

[09 Hours]

Transaction Concepts and ACID Properties, Transaction States, Implementation of Atomicity and Durability, Concurrent Executions, Serializability and Its Testing, Recoverability, Lock-Based Protocols, Validation-Based Protocol, Multiple Granularity, Multiversion Schemes, Deadlock Handling

Text Book

1. Silberschatz, A., Korth, H. F., and Sudarshan, S., Database System Concepts, 6th Edition, Tata McGraw-Hill, New Delhi, 2011. (T1)

Reference Books

1. Elmasri, R. and Navathe, S. B., Fundamentals of Database Systems, 7th Edition, Pearson Education, New Delhi, 2016. (R1)
2. Ullman, J. D. and Widom, J., A First Course in Database Systems, 3rd Edition, Pearson Education, New Delhi, 2014. (R2)

Practical's:

1. Study of DBMS Environment and Tools
Installation and familiarization with DBMS software (MySQL/Oracle/SQLite), basic commands.
2. Creation of Database and Tables
Write SQL queries to create database, tables, and define primary keys and constraints.
3. Data Manipulation Operations
Perform INSERT, UPDATE, DELETE operations on tables.
4. Basic SQL Queries
Write queries using SELECT, WHERE, ORDER BY, DISTINCT.
5. SQL Aggregate Functions and Grouping
Use COUNT, SUM, AVG, MAX, MIN, GROUP BY, HAVING.
6. Set Operations and Joins
Implement UNION, INTERSECT, DIFFERENCE; INNER JOIN, OUTER JOIN.

Ravish *Shankar* *Atyain* *SSG* *Shuchi Bhatia*

7. Nested Queries and Complex Queries
Write subqueries (nested queries) for solving complex problems.
8. Views and Indexes
Create and use views; implement indexing and analyze performance.
9. Constraints and Triggers
Apply integrity constraints; create and execute triggers.
10. Normalization and Schema Design
Perform normalization (1NF, 2NF, 3NF, BCNF) on given relations.
11. Transaction Control
Use COMMIT, ROLLBACK, SAVEPOINT; demonstrate ACID properties.
12. Concurrency Control Simulation
Demonstrate locking mechanisms, deadlock situations, and basic recovery concepts.

Latika *Deepa* *Arjun* *SSG* *Shuchi Singh*

Course Title: Computational Thinking Through MATLAB and Python	
Course Code: FYCS-SEC305	Nature of Course: Skill Enhancement Course
Credit: 3	Classes Per Week: L: 2 T: 0 P: 2

Course Objectives:

1. To introduce students to the core syntax, data structures, and execution environments of both MATLAB and Python.
2. To develop students' ability to break down complex problems and implement logical solutions using flow control, loops, and conditional statements.
3. To train students in handling, indexing, and manipulating arrays and data structures for scientific and engineering applications.
4. To teach students how to effectively generate, label, and interpret 2D data plots and visualizations.
5. To instill best practices in code reusability and organization through the creation of custom scripts, functions, and the use of external libraries.

Course Outcomes:

After successful completion of this course, students will be able to:

- CO1:** Students will be able to navigate the MATLAB desktop and Python environments to write, debug, and execute interactive commands and multi-line scripts.
- CO2:** Students will be able to construct and manipulate matrices in MATLAB and list/dictionary structures in Python to process multi-dimensional data.
- CO3:** Students will be able to design and implement loops and conditional branches to automate repetitive tasks and make algorithmic decisions.
- CO4:** Students will be able to write code that translates raw numeric data into clear, professional visual graphics with proper annotations.
- CO5:** Students will be able to develop user-defined functions and import standard libraries to build efficient, readable, and maintainable programs.

Syllabus

Module 1

[15 Hours]

The MATLAB Environment (Command Window, Workspace, Current Folder, Editor, Live Editor), Variables and Basic Arithmetic (Creating Variables, Suppressing O/P using ;, Operators +, -, *, /), Matrices and Arrays (Scalars, Vectors, Matrices and use of Colon operator), Array Indexing and Manipulation (Accessing elements, matrix Indexing, End keyword and element by element comparison), Basic Plotting and Visualization (2D plots

Ranish *Shaw* *Arjun* *SS* *Shuchi Singh*

and Annotations), Scripts vs. Functions(.m files and function keyword), and Basic Programming Flow Control(For, While, If else if else)

Module 2

[15 Hours]

Variables and Data Types (Integer, String, Boolean), Basic Data Structures (List, Dictionaries, Tuples), Indentation and Control Flow (Use of Whitespace, If/else etc.), Loops (For Loop, While Loop), Functions (def keyword). Importing and including library functions.

Textbooks

1. MATLAB: A Practical Introduction to Programming and Problem Solving – Stormy Attaway
2. MATLAB for Beginners – Peter I. Kattan
3. Python Crash Course – Eric Matthes
4. Fluent Python – Luciano Ramalho

Reference Books

5. <https://www.matlabacademy.mathworks.com>
6. Python Cookbook – David Beazley

Practical's:

1. Write a program to explore the MATLAB environment; demonstrate your familiarity by using the Command Window to define a variable, the Workspace to verify it, the Editor to save a script, and the Help section to look up a function.
2. Write a program to create a 3x3 magic square; verify the "magic" property by extracting specific elements to show that the sums of the rows and columns are equal.
3. Write a program to define two 2 x 2 matrices and compute both their standard matrix product and their element-by-element product; include a comment reflecting on why these two operations yield different results.
4. Write a program to generate a time vector from 0 to 2*pi using the colon operator; calculate the sine of these values and plot the result with a grid, title, and clearly labeled axes.
5. Write programs creating of two files a function that converts temperature from Fahrenheit to Celsius, and a script that prompts the user for input and calls that function to display the result.
6. Write a program that uses a loop and conditional if-elseif statements to iterate through a vector of grades, categorizing and counting how many are labeled as "Pass," "Fail," or "Excellent."
7. Write a program to create an interactive "Mad Libs" game that uses the input() function to collect parts of speech and displays the final story using f-strings.
8. Write a program to manage a shopping list that utilizes a while loop to allow users to continuously add, remove, or view items until they type "exit."

Rajin *Pranav* *Arjun* *SSG* *Srujan Prasad*

9. Write a program to create a digital rolodex using a dictionary implement logic that allows users to look up a phone number by name or add a new contact if the name is not found.
10. Write a program to solve the "FizzBuzz" challenge for numbers 1 to 50; use the modulo operator (%) to print "Fizz" for multiples of 3, "Buzz" for multiples of 5, and "FizzBuzz" for multiples of both.
11. Write a program that defines a function to check password strength; the function should return True or False based on whether the input string meets specific length and special character criteria.
12. Write a program that imports the random module to simulate rolling two six-sided dice 1,000 times; calculate and print the percentage of rolls where the sum equals 7.

Roujan *Debasish* *Atyain* *SSG* *Shuchi Singh* *Dr*

Course Title: Analytical and Decision-Making Skills	
Course Code: FYCS-AEC306	Nature of Course: Ability Enhancement Course
Credit: 2	Classes Per Week: L: 2 T: 0 P: 0

Course Objectives:

The course aims to:

1. Develop analytical thinking and problem-solving abilities among students.
2. Enhance logical reasoning and data interpretation skills.
3. Build decision-making capabilities in structured and unstructured situations.
4. Introduce techniques and tools for effective decision making.
5. Improve critical thinking for real-life and professional scenarios.

Course Outcomes (COs):

After successful completion of this course, students will be able to:

CO1: Apply analytical thinking to solve real-world problems.

CO2: Interpret data and draw meaningful conclusions.

CO3: Use logical reasoning for structured problem solving.

CO4: Apply decision-making techniques in various scenarios.

CO5: Evaluate alternatives and select optimal decisions.

Syllabus

Module I

[15 Hours]

Introduction to Analytical Thinking: Meaning, Importance, and Applications; Logical Reasoning: Deductive and Inductive Reasoning, Statements and Arguments, Syllogism; Quantitative Aptitude: Number Systems, Percentages, Ratios and Proportions, Averages; Data Interpretation: Tables, Bar Graphs, Pie Charts, Line Graphs; Problem Solving Techniques: Pattern Recognition, Series Completion, Puzzles; Critical Thinking: Assumptions, Inferences, Evaluation of Arguments

Module II

[15 Hours]

Introduction to Decision Making: Concepts, Types of Decisions, Decision-Making Process; Decision Making Models: Rational Model, Bounded Rationality, Intuitive Decision Making; Tools and Techniques: Decision Trees, Cost-Benefit Analysis, SWOT Analysis; Risk and Uncertainty: Types of Risk, Decision Making under Risk and Uncertainty; Group Decision Making: Brainstorming, Consensus, Delphi Technique; Ethics in Decision Making, Real-Life Case Studies

Text Books

1. Sharma, R. S., Quantitative Aptitude, McGraw-Hill.
2. Kahneman, D., Thinking, Fast and Slow, Penguin.

Reference Books

1. Bazerman, M. H., Judgment in Managerial Decision Making, Wiley.
2. Cottrell, S., Critical Thinking Skills, Palgrave Macmillan.

Ranjana *Plam* *Hyam* *SSK* *Shuchi Prinja*

Course Title: Information Security and Cyber Law	
Course Code: FYCS – C401	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

The course aims to:

1. Understand fundamental concepts of information security and cyber threats.
2. Develop knowledge of cryptographic techniques and network security.
3. Familiarize students with cyber laws and legal frameworks.
4. Understand cyber crimes, investigation techniques, and digital forensics.
5. Apply security and legal principles in real-world cyber environments.

Course Outcomes (COs):

After successful completion of this course, students will be able to:

CO1: Analyze information security requirements and threat models.

CO2: Apply cryptographic and network security techniques.

CO3: Interpret cyber laws and legal provisions related to IT systems.

CO4: Investigate cyber-crimes using forensic techniques.

CO5: Evaluate security policies, compliance, and ethical issues.

Syllabus

Module I

[09 Hours]

Information Security Concepts, Information Security Goals, Security Attacks: Passive Attacks, Active Attacks, Security Services and Mechanisms, Risk Management: Threats, Vulnerabilities, Risk Assessment, Security Policies and Standards

Module II

[09 Hours]

Cryptography Basics: Symmetric Encryption, Asymmetric Encryption, Hash Functions, Digital Signatures; Network Security: Firewalls, Intrusion Detection Systems, Virtual Private Networks, Authentication Mechanisms

Module III

[09 Hours]

Cyber Security Concepts, Cyber Threats and Attacks, Malware: Viruses, Worms, Trojans, Cyber Crimes: Hacking, Identity Theft, Phishing, Online Fraud, Ethical Hacking Concepts, Incident Response

Laish *Plank* *Hyam* *SSB* *Shuchi Singh*

Module IV

[09 Hours]

Fundamentals of Cyber Law, Information Technology Act 2000 and Amendments, Digital Signatures, Electronic Records, E-Commerce Laws, Intellectual Property Rights in Cyberspace, Cyber Jurisdiction, Cyber Offences and Penalties

Module V

[09 Hours]

Digital Forensics: Evidence Collection, Preservation, Analysis, Cyber Crime Investigation Techniques, Admissibility of Digital Evidence, Cyber Security Governance, Data Protection and Privacy, Cyber Ethics, Security Compliance and Audit

Text Books

1. Stallings, W., Cryptography and Network Security, Pearson.
2. Singhal, A. and Shivakumar, N., Cyber Security and Cyber Laws, McGraw-Hill.

Reference Books

1. Charles P. Pfleeger, Security in Computing, Pearson.
2. Nina Godbole, Information Systems Security, Wiley India.
3. IT Act 2000 Bare Act (Government of India).

Practical's:

1. Study of basic security tools and operating system security settings.
2. Implementation of basic encryption and decryption techniques (Caesar Cipher, RSA basics).
3. Password security and hashing techniques using tools/software.
4. Configuration of firewall settings in Windows/Linux.
5. Network scanning and vulnerability analysis using tools (e.g., Nmap).
6. Study and simulation of common cyber-attacks (phishing, malware behaviour).
7. Packet analysis using Wireshark.
8. Digital evidence collection and analysis basics.
9. Case study analysis of cyber-crime incidents.
10. Study of IT Act provisions through real-life examples.
11. Risk assessment and security policy drafting exercise.
12. Incident response and reporting simulation.

Ravi *Deep* *Agarwal* *SSG* *Shuchi Prinja* *BS*

Course Title: Object Oriented Programming	
Course Code: FYCS- C402	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 02

Course Objectives:

The course aims to:

1. To transition students from procedural thinking to an object-oriented mindset, utilizing classes, objects, and encapsulation.
2. To familiarize students with the Java ecosystem (JVM, JDK, JRE) and teach them how Java achieves its platform-independent capabilities.
3. To introduce core OOP principles such as inheritance, polymorphism, abstraction, and interfaces for building reusable and scalable code.
4. To train students to predict, identify, and handle runtime errors gracefully using Java's built-in exception handling framework.
5. To provide students with foundational knowledge in manipulating strings and handling basic file input/output operations.

Course Outcomes:

After successful completion of this course, students will be able to:

- CO1** Students will be able to configure a Java development environment and write syntactically correct programs utilizing basic data types, operators, and interactive console I/O.
- CO2** Students will be able to design and implement complex branching logic, loops, and multi-dimensional arrays to solve computational problems.
- CO3** Students will be able to model real-world entities by creating custom Java classes with appropriate attributes, methods, constructors, and strict data encapsulation.
- CO4** Students will be able to implement multi-level inheritance structures, polymorphic method overrides, and interfaces to reduce code redundancy.
- CO5** Students will be able to develop programs that read/write external text files and employ standard try-catch blocks to prevent sudden application crashes.

Syllabus

Module 1

[09 Hours]

Evolution of Java, JVM architecture, and the "Write Once, Run Anywhere" philosophy. Syntax Basics: Variables, Constants, and Data Types; Logic Flow: Decision-making statements – if, if-else, nested if; switch statement, Looping Constructs: for loop; while loop; do-while loop; Nested loops; Loop control statements (break, continue).

Ravish *Plav* *Arjun* *SSG* *Sandeep Singh*

Module 2

[09 Hours]

Core OOP Concepts – Abstraction, Encapsulation; Classes and Objects; Relationship between Classes and Objects; Methods and Constructors: Defining class behaviors, Method Overloading, Object initialization; Memory Management: Stack and Heap memory, Object creation and storage.

Module 3

[09 Hours]

Arrays: One-dimensional and multi-dimensional arrays; String Manipulation: String and StringBuffer classes; Vectors: Dynamic arrays and standard arrays.

Module 4

[09 Hours]

Inheritance: Extending classes, super keyword, types of inheritance (Single, Multilevel, Hierarchical); Polymorphism : Method overriding and dynamic method dispatch; Finality: final variables, methods, and classes.

Module 5

[09 Hours]

Interfaces: Implementing multiple inheritance and defining contracts for classes; Packages: Organizing code into namespaces, use of built-in packages, creating user-defined packages; Access Control: Access modifiers (public, private, protected, default).

Textbooks

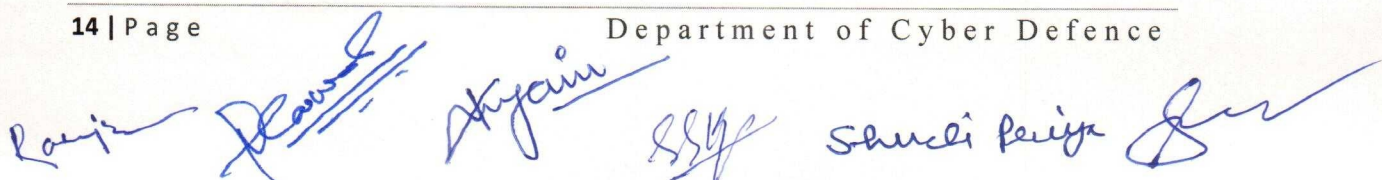
1. Object Oriented Programming with C++ by E. Balagurusamy, McGraw-Hill Education (India)
2. Java: How to Program. Deitel & Deitel
3. Effective Java – Joshua Bloch

Reference Books

1. Object-Oriented Thought Process – Matt Weisfeld
2. UML Distilled – Martin Fowler

Practical's:

1. Write a program that prompts the user to input the radius of a sphere. The program should calculate and display the diameter, surface area $4\pi r^2$, and volume $(4/3)\pi r^3$.
2. Write a program that converts an amount in one currency (e.g., USD) to three other currencies based on fixed exchange rates. Students must ensure they use proper floating-point data types (double) and format the output to two decimal places.
3. Write a program to simulate a basic ATM machine. The program should display a menu (a. Check Balance, b. Deposit, c. Withdraw, d. Exit). Use a loop to keep the program running until the user chooses to exit, updating the simulated balance appropriately.



4. Write a program to ask the user to input elements for two 3×3 matrices. The program should compute and display their sum. As a bonus, have the program find and print the transpose of the first matrix.
5. Write a program to create a `Student` class with attributes like name, rollNumber, and an array for marks in 5 subjects. Include a method to calculate the percentage and another to display the student's full profile. Instantiate at least two student objects in the main method.
6. Write a program to create an `Account` class with a private balance. Use a parameterized constructor to set the initial balance. Provide public methods to `deposit()`, `withdraw()`, and `getBalance()`. Ensure that `withdraw()` prevents the user from taking out more money than they have.
7. Write a program to create a base class `Employee` with basic details (name, ID). Derive two classes from it: `FullTimeEmployee` (adds a monthly salary) and `PartTimeEmployee` (adds hourly rate and hours worked). Demonstrate how both child classes inherit properties from the parent.
8. Create a class called `ShapeArea`. Write overloaded methods named `calculateArea()` to find the area of a square (takes 1 side), a rectangle (takes length and width), and a circle (takes radius).
9. Create a base class `Vehicle` with a method `speed()`. Create subclasses `Bicycle` and `SportsCar` that override the `speed()` method. In the driver class, create a `Vehicle` reference pointing to different objects to show different outputs at runtime.
10. Create an interface `RemoteControl` with methods like `powerOn()`, `powerOff()`, and `changeChannel()`. Create classes `TV` and `Projector` that implement this interface in their own unique ways.
11. Create a program that asks a user for an index and a number to divide by. Handle cases where the user enters a non-integer (`InputMismatchException`), tries to access a non-existent index (`ArrayIndexOutOfBoundsException`), or tries to divide by zero (`ArithmeticException`).
12. Write a program that asks the user to type several lines of text in the console. Save these lines into a file called `user_log.txt`. Then, have the program read the file back and display the contents in the console to verify successful writing.

[Handwritten signatures and initials]

Course Title: Theory of Computation	
Course Code: FYCS – C403	Nature of Course: Core
Credit: 4	Classes Per Week: L: 4 T: 0 P: 0

Course Overview:

Upon completion of this course, students will be able to:

1. Understand the concept of systems and analyze system behavior.
2. Design and analyze finite state machines along with their equivalent regular expressions.
3. Construct pushdown automata and relate them to context-free grammars.
4. Design and evaluate Turing machines and Post machines.
5. Develop a formal understanding of algorithms, procedures, and limitations of finite representations in languages and machines.

Course Outcomes:

After successful completion of the course, students will be able to:

- CO1** Explain the relationship between formal languages and mathematical models of computation.
- CO2** Analyze various classes of languages and their corresponding computational models.
- CO3** Evaluate pushdown automata and understand their applications in compiler design.
- CO4** Examine the capabilities and limitations of real-world computers, including undecidable problems.
- CO5** Analyze computational complexity classes such as P, NP, NP-Complete, and NP-Hard problems.

Syllabus

Module I

[09 Hours]

Introduction to Automata: (mathematical model of digital devices, including real computer), State Transition Graph, Finite Automaton (FA) and its types, Deterministic Finite Automaton (DFA), Non-deterministic Finite Automaton (NFA), Complement, Union, Intersection of FA's, Conversion Strategy from NFA to DFA, Minimization of FA, Finite Automaton with Output, Applications of FA.

Module II

[09 Hours]

Regular Expressions(RE): Introduction, R.E.'s and basic operations, Algebraic laws on Regular Expression, Finite and Infinite Languages, Equivalence of finite Automaton and regular expressions, Constructing NFA from Regular Expression, Pumping Lemma for Regular Language, Closure properties of Regular Languages, Non-regular languages, Applications of Regular Expression.

[Handwritten signatures]

Module III

[09 Hours]

Grammar: Introduction, Formal Definition of Grammar, The Chomsky Hierarchy of Grammar, Designing Regular grammar from DFA, Context Free Grammar, Closure properties of Context Free Languages, , CFG and Normal form: Chomsky Normal Form, Greibach Normal Form, Non-Context Free Language, Applications of CFGs.

Module IV

[09 Hours]

Push Down Automation (PDA): Introduction, Definition of PDA, Types of Pushdown Automata (DPDA and NPDA), Converting CFG to PDA, Derivation (Parsing), Parsing Techniques, Ambiguous and Unambiguous Grammar, Demerits of Ambiguous Grammar.

Module V

Turing Machine(TM): Single Tape TM, Variations of TM, Halting Problem, Turing Machine and Languages, Enumerable Languages, Decidable, Recognizable and Undecidable languages, Solvable and Unsolvable problems, Post Correspondence Problems(PCP), Classes of Problems: P, NP, NP-C and NP-Hard. (8L)

Text Book:

1. Hopcroft J.E., Motwani R. and Ullman J.D, Introduction to Automata Theory, Languages and
2. Computations, Second Edition, Pearson Education, 2008. (T1)

Reference Books:

1. Mishra K.L.P. and Chandrasekaran N. , Theory of Computer Science: Automata, Languages and Computation, 3rd Edition, PHI.(R1)
2. Martin John C., Introduction to Languages and the Theory of Computation, 3rd Edition, Tata McGraw Hill Publishing Company, New Delhi, 2007. (R2)
3. Lewis Harry R. and Papadimitriou Christos H., Elements of the theory of Computation, 2nd Edition, Prentice-Hall of India Pvt. Ltd. (R3)

Rashmi *Deepak* *Arjun* *SSG* *Shuchi Singh*

Course Title: Universal Human Values-III	
Course Code: FYCS – VAC405	Nature of Course: Value Added Course
Credit: 2	Classes Per Week: L: 2 T: 0 P: 0

Course Objective:

1. To develop a comprehensive understanding of India's historical, cultural, and social evolution
2. To familiarize students with the Indian Constitution and its key features
3. To promote awareness of civic responsibilities and democratic values
4. To enable students to appreciate India's diversity and national identity

Course Outcomes:

After successful completion of the course, students will be able to:

CO1: Understand India's historical, cultural, and social development

CO2: Analyze the diversity and unity of Indian society

CO3: Explain the structure and key features of the Indian Constitution

CO4: Recognize fundamental rights, duties, and governance systems

Syllabus

Module I

(15 Hours)

Bharatvarsha: Concept and evolution (Vedic, Epic, and Puranic traditions), Making of modern India, Ancient and medieval state formation: Mauryas, Guptas, Cholas, Vijayanagara, Religious and philosophical traditions: Vedic, Jainism, Buddhism, Bhakti and Sufism, Reform movements (Brahmo Samaj, Arya Samaj), Indian society: Diversity, caste, community, and gender, Contributions of India to the world: Literature (Panini, Kalidasa), Science and Mathematics (Aryabhata, Brahmagupta, Ramanujan), India's freedom struggle: 1857 revolt, major leaders (Mahatma Gandhi, Bhagat Singh, Subhash Chandra Bose, Birsa Munda), National movements and nation-building, Contemporary development initiatives (Make in India, Skill India, Atmanirbhar Bharat)

Module II

(15 Hours)

Making of the Indian Constitution: Historical background and Constituent Assembly, Salient features of the Constitution, Preamble and its significance. Fundamental Rights, Directive Principles of State Policy, Fundamental Duties, Structure of Government: Legislature, Executive, and Judiciary, Federal system and Centre-State relations, Election process and role of Election Commission, Amendment procedure of the Constitution, Role of Constitution in governance, democracy, and citizen responsibilities.

[Handwritten signatures and initials]

Suggested Readings

1. Dhingra, I.C. (2020). Indian Economy. Sultan Chand & Sons.
2. Dutt, R., & Sundharam, K. (2018). Indian Economy. S. Chand.
3. Gautam, A. (2009). Advanced Geography of India. Sharda Pustak Bhawan.
4. Basu, D. D. (2015). Introduction to the Constitution of India. LexisNexis.
5. Laxmikanth, M. (2022). Indian Polity. McGraw Hill.

Ravi *Danish* *Arjun* *SSG* *Sandeep Singh* *Dr*

Course Title: Computational Thinking	
Course Code: FYCS – AEC406	Nature of Course: Ability Enhancement Course
Credit: 2	Classes Per Week: L: 2 T: 0 P: 0

Course Objectives:

1. To introduce the fundamental principles of computational thinking and problem solving
2. To develop skills in algorithmic thinking, abstraction, and decomposition
3. To understand computational models and their applications across domains
4. To apply basic programming constructs for problem solving
5. To analyze data and understand the societal impact of computing

Course Outcomes (COs):

After completion of the course, students will be able to:

CO1: Apply computational thinking concepts such as decomposition, pattern recognition, and abstraction

CO2: Design algorithms using sequence, selection, and iteration constructs

CO3: Implement basic programs using standard programming tools

CO4: Analyze and manipulate data using computational techniques

CO5: Evaluate ethical and societal implications of computing systems

Syllabus

Module I

[15 Hours]

Computational Thinking: Introduction, definition, characteristics and importance, problem solving using decomposition, pattern recognition, abstraction and algorithm design, computational models and their components, fundamentals of algorithms, properties of algorithms, sequence, selection and iteration, representation of algorithms using flowcharts and pseudocode, introduction to programming concepts, variables, data types and operators.

Module II

[15 Hours]

Data and Computation: Introduction, data representation and types, basic data structures, data manipulation techniques, input and output operations, expressions and statements, implementation using basic programming concepts, debugging and testing, data visualization concepts, real world problem solving using computational approaches, societal impact of computing, ethical issues, privacy, security and social implications.

Textbook

1. Computational Thinking — Peter J. Denning, Cambridge University Press.

Rashid *Alam* *Arifain* *SSG* *Sauki Puriya* *San*

Reference Books

1. Computational Thinking A Beginner's Guide to Problem Solving and Programming Karl Beecher, BCS Learning & Development Ltd.
2. Python Crash Course A Hands-On Project-Based Introduction to Programming — Eric Matthes, No Starch Press.
3. Think Python How to Think Like a Computer Scientist — Allen B. Downey, O'Reilly Media.

Route *Planned* *Again* *2/5/24* *Sandeep Singh* *Dr*

Course Title: Introduction to Artificial Intelligence	
Course Code: FYCS-C501	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To provide a comprehensive introduction to the fundamental concepts and techniques of Artificial Intelligence, including search, logic, and reasoning.
2. To familiarize students with the mathematical foundations of Machine Learning and Deep Learning, enabling them to understand how models learn from data.
3. To teach students how to identify and formulate real-world problems as AI challenges, such as Constraint Satisfaction Problems or Optimization tasks.
4. To equip students with the programming skills necessary to implement AI solutions using modern libraries and frameworks like Scikit-Learn and PyTorch.
5. To foster a critical understanding of the social, ethical, and safety implications of AI, preparing students to build unbiased and transparent systems.

Course Outcomes

After completion of the course, students will be able to:

- CO1** Students can value a given environment and design an appropriate Intelligent Agent architecture (reflex, goal-based, or utility-based) to navigate it.
- CO2** Students can apply and compare various search algorithms (like A* Search or Minimax) to solve complex puzzles and adversarial games efficiently.
- CO3** Students can develop, train, and validate machine learning models for classification, regression, and clustering tasks using real-world datasets.
- CO4** Students can construct and fine-tune Convolutional Neural Networks (CNNs) and basic Transformer models for computer vision and natural language processing tasks.
- CO5** Students can critically assess AI systems for algorithmic bias and safety risks, and propose technical or policy-based mitigations to ensure Responsible AI deployment.

Syllabus

Module I

[09 Hours]

Introduction to AI (History, Turing Test, and the four views of AI (Acting/Thinking Humanly/Rationally). Intelligent Agents (Agent-environment loop, PEAS (Performance, Environment, Actuators, Sensors) descriptors, and agent types (Simple Reflex, Model-based, Goal-based, Utility-based), Foundational Math (Linear Algebra, Probability Theory, and Statistics for AI).

Module 2

[09 Hours]

Uninformed Search (Breadth-First, Depth-First, and Uniform Cost Search), Informed Heuristic Search (Greedy Best-First and A* Search), Adversarial Search (Games, Minimax

[Handwritten signatures]

algorithm, and Alpha-Beta Pruning.), Knowledge Representation (Propositional Logic, First-Order Logic, and Inference Rules, Resolution, Unification).

Module 3

[09 Hours]

Supervised Learning (Linear/Logistic Regression, Decision Trees, and Support Vector Machines), Unsupervised Learning (K-Means Clustering and Dimensionality Reduction, PCA), Reasoning under Uncertainty (Bayesian Networks and Hidden Markov Models), Model Evaluation (Bias-variance tradeoff, Cross-validation, and Metrics Accuracy, Precision, Recall, F1-score).

Module 4

[09 Hours]

Neural Network Fundamentals (Perceptron, Backpropagation, and Activation Functions), Computer Vision (Convolutional Neural Networks (CNNs) for image classification and object detection), Natural Language Processing (Word Embeddings, Recurrent Neural Networks), Generative AI (Basics of Large Language Models (LLMs) and Prompt Engineering)

Module 5

[09 Hours]

AI Ethics (Addressing Bias, Fairness, and Transparency in algorithmic decision-making), AI Safety & Control (The alignment problem and human-compatible AI), Agentic Frameworks (Multi-agent systems and AI agents that can use tools and take actions) Capstone Project (Design and implementation of a small-scale AI system (e.g., a chatbot, a predictive model, or a simple game-playing agent).

Textbooks

1. Artificial Intelligence: A Modern Approach By Stuart Russell And Peter Norvig
2. Artificial Intelligence: A Guide To Intelligent Systems By Michael Negnevitsky

Reference Books

1. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron.
2. Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville.

Practical's:

1. Write a program to implement a simple reflex agent for a two-room vacuum world. The agent must sense its location and whether the floor is dirty, then decide to move or "suck."
2. Write a program create a "PEAS" simulator in Python where an agent must navigate a grid to collect items while losing energy with every move, demonstrating utility-based decision making.
3. Write a program to use A* Search to solve an 8-puzzle. Students must implement at least two heuristics: Manhattan Distance and Misplaced Tiles and compare their efficiency.

Rajiv *Deep* *Arjun* *SSK* *Sandeep Singh*

3. Minimax Tic-Tac-Toe: Build a Tic-Tac-Toe game where the computer uses the Minimax algorithm with Alpha-Beta pruning to ensure it never loses against a human player.
4. Sudoku as CSP: Solve a Sudoku puzzle by representing it as a Constraint Satisfaction Problem (CSP) using backtracking search and arc consistency (\$AC-3\$).
6. Spam Classifier: Build a Spam text classifier using the Naive Bayes algorithm. Students should preprocess raw text (removing stop words/stemming) before training.
7. Predictive Housing Model: Use Linear Regression to predict house prices based on multiple features like square footage and number of rooms using libraries like Scikit-learn.
8. Customer Segmentation: Apply K-Means Clustering to a mall customer dataset to group individuals based on annual income and spending scores.
9. Handwritten Digit Recognition: Implement a Neural Network to classify the MNIST dataset of handwritten digits using TensorFlow or PyTorch.
10. Image Sentiment Analysis: Train a Convolutional Neural Network (CNN) to distinguish between "Happy" and "Sad" faces in a provided image dataset.
11. Simple Chatbot with LLM APIs: Use an API (like OpenAI or Hugging Face) to create a Simple Chatbot that can answer questions based on a specific "persona" or provided document (RAG - Retrieval-Augmented Generation).
12. Prompt Engineering Lab: Write a Python script that takes a user's "bad" prompt and uses a few-shot prompting technique to automatically refine it for better AI output.

[Handwritten signatures and initials in blue ink]

Course Title: Cryptography and Network Security	
Course Code: FYCS-C502	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To understand fundamental concepts of information and network security
2. To analyze classical and modern cryptographic techniques
3. To study symmetric and asymmetric encryption algorithms
4. To understand authentication, hashing, and digital signatures
5. To explore network and web security protocols
6. To gain practical exposure to tools like PGP, SSL/TLS, and SSH

Course Outcomes:

At the end of the course student will have ability to:

- CO1: Explain security principles, services, and attack models
- CO2: Apply symmetric and asymmetric cryptographic algorithms
- CO3: Analyze hashing, authentication, and digital signature techniques
- CO4: Evaluate network security protocols and architectures
- CO5: Implement secure communication using cryptographic tools
- CO6: Identify vulnerabilities and propose security solutions

Syllabus

Module 1

[09 Hours]

Security Concepts: Introduction to Information Security, Need for Security, Security Approaches, Principles of Security (Confidentiality, Integrity, Availability, Authentication, Non-repudiation), Types of Security Attacks (Passive and Active), Security Services, Security Mechanisms, Model for Network Security.

Cryptography Concepts and Techniques: Introduction to Cryptography, Plaintext and Ciphertext, Substitution Techniques, Transposition Techniques, Encryption and Decryption, Symmetric and Asymmetric Key Cryptography, Steganography, Key Range and Key Size, Types of Cryptographic Attacks.

Module 2

[09 Hours]

Symmetric Key Cryptography: Block Cipher Principles, Data Encryption Standard (DES), Advanced Encryption Standard (AES), Blowfish, RC5, IDEA, Modes of Operation of Block Ciphers, Stream Ciphers (RC4).

Asymmetric Key Cryptography: Principles of Public Key Cryptosystems, RSA Algorithm, ElGamal Cryptography, Diffie-Hellman Key Exchange, Knapsack Algorithm.

Ravi *Ravi* *Arjun* *SSK* *Shubh Prasad*

Module 3

[09 Hours]

Cryptographic Hash Functions and Authentication: Message Authentication, Secure Hash Algorithm (SHA-512), Message Authentication Codes (MAC), HMAC, CMAC, Digital Signatures, ElGamal Digital Signature Scheme.

Key Management and Distribution: Symmetric Key Distribution using Symmetric and Asymmetric Techniques, Public Key Distribution, Kerberos, X.509 Authentication Service, Public Key Infrastructure (PKI).

Module 4

[09 Hours]

Transport-Level Security: Web Security Considerations, Secure Socket Layer (SSL), Transport Layer Security (TLS), HTTPS, Secure Shell (SSH).

Wireless Network Security: Wireless Security Issues, Mobile Device Security, IEEE 802.11 Wireless LAN, IEEE 802.11i Wireless LAN Security.

Module 5

[09 Hours]

E-Mail and IP Security: Pretty Good Privacy (PGP), S/MIME, IP Security Overview, IP Security Architecture, Authentication Header (AH), Encapsulating Security Payload (ESP), Combining Security Associations, Internet Key Exchange (IKE).

Text Books

1. Cryptography and Network Security: Principles and Practice – William Stallings, Pearson
2. Cryptography and Network Security – Atul Kahate, McGraw Hill

Reference Books

1. Cryptography and Network Security, C.K. Shyamala, N. Harini, T.R. Padmanabhan, Wiley India, 1st Edition.
2. Cryptography and Network Security, Behrouz A. Forouzan, Debdeep Mukhopadhyay, McGraw Hill, 3rd Edition.
3. Information Security: Principles and Practice, Mark Stamp, Wiley India.
4. Principles of Computer Security, W.M. Arthur Conklin, Greg White, Tata McGraw Hill (TMH).
5. Introduction to Network Security, Neal Krawetz, Cengage Learning.
6. Network Security and Cryptography, Bernard Menezes, Cengage Learning.

Practical's:

1. Implementation of classical encryption techniques (Caesar Cipher, Playfair Cipher).
2. Implementation of Transposition Cipher (Rail Fence / Columnar).
3. Implementation of Data Encryption Standard and Advanced Encryption Standard using Python/OpenSSL.
4. Implementation of RSA algorithm for encryption and decryption.
5. Simulation of Diffie-Hellman Key Exchange for secure key sharing.
6. Implementation of hashing using Secure Hash Algorithm (SHA-256 / SHA-512).
7. Implementation of Message Authentication Code (MAC) and HMAC.

Sanjay *Sharma* *Arjun* *SSG* *Shuchi Priya*

8. Digital Signature generation and verification using RSA / ElGamal.
9. Generation and management of GNU Privacy Guard keys and sending encrypted email.
10. Configuration and analysis of Secure Socket Layer / Transport Layer Security on a web server.
11. Secure remote login using Secure Shell.
12. Packet capturing and analysis using Wireshark.
13. Demonstration of common web vulnerability: Cross-Site Scripting (XSS).
14. Mini Project: Design and implementation of a secure communication system (using encryption + authentication).

Ravish
Dhanu
Kigam
SSG
Shuchi Bhatia

Course Title: Design and Analysis of Algorithms	
Course Code: FYCS-C503	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To introduce the fundamental concepts and characteristics of algorithms.
2. To understand asymptotic notations and analyze the efficiency of algorithms.
3. To study mathematical analysis of recursive and non-recursive algorithms.
4. To learn brute force problem-solving techniques.
5. To implement and analyze basic sorting and searching algorithms.

Course Outcomes:

After completion of the course, students will be able to:

- CO1:** Understand the fundamental concepts of algorithms and their characteristics.
- CO2:** Analyze the time and space complexity of recursive and non-recursive algorithms using asymptotic notations.
- CO3:** Apply brute force techniques to solve basic computational problems.
- CO4:** Implement and evaluate basic sorting and searching algorithms.
- CO5:** Analyze simple string-matching techniques and their performance.

Syllabus

Module 1

[09 Hours]

Introduction:

Notion of Algorithm, Review of Asymptotic Notations, Mathematical Analysis of Non-Recursive and Recursive Algorithms.

Brute Force Approaches: Introduction, Selection Sort, Bubble Sort, Sequential Search, Brute Force String Matching.

Module 2

[09 Hours]

Divide And Conquer: Divide and Conquer: General Method, Defective Chess Board, Binary Search, Merge Sort, Quick Sort and its performance.

The Greedy Method: The General Method, Knapsack Problem, Job Sequencing with Deadlines, Minimum-Cost Spanning Trees: Prim's Algorithm, Kruskal's Algorithm; Single Source Shortest Paths.

Module 3

[09 Hours]

Dynamic Programming: The General Method, Warshall's Algorithm, Floyd's Algorithm for the All-Pairs Shortest Paths Problem, Single-Source Shortest Paths: General Weights, 0/1 Knapsack, The Traveling Salesperson problem.

Sanjay *Prakash* *Arjun* *SSK* *Shuchi Pring*

Module 4

[09 Hours]

Decrease-And-Conquer Approaches, Space-Time Tradeoffs: Decrease-and-Conquer Approaches: Introduction, Insertion Sort, Depth First Search and Breadth First Search, Topological Sorting Space-Time Tradeoffs: Introduction, Sorting by Counting, Input Enhancement in String Matching.

Module 5

[09 Hours]

Limitations Of Algorithmic Power and Coping with Them: Lower-Bound Arguments, Decision Trees, P, NP, and NP-Complete Problems, Challenges of Numerical Algorithms. Coping With Limitations of Algorithmic Power. Branch-and-Bound. Approximation Algorithms for NP-Hard Problems.

Textbooks

1. Anany Levitin: Introduction to The Design & Analysis of Algorithms, 2nd Edition, Pearson Education, 2007.
2. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran: Fundamentals of Computer Algorithms, 2nd Edition, Universities Press, 2007.

Reference Books

1. Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, Clifford Stein: Introduction to Algorithms, 3 rd Edition, PHI, 2010.
2. R.C.T. Lee, S.S. Tseng, R.C. Chang & Y.T.Tsai: Introduction to the Design and Analysi

Practical's:

1. Quick Sort Analysis:
Sort a given set of elements using Quick Sort and determine the time required. Repeat for different values of n and plot a graph of time taken versus n. Input may be taken from a file or generated randomly.
2. Parallel Merge Sort :
Implement Merge Sort. Measure execution time for varying n and plot performance graph.
3. Graph Algorithms – I:
 - a) Obtain Topological ordering of vertices in a given directed graph.
 - b) Compute transitive closure using Warshall's Algorithm.
4. Dynamic Programming:
Implement 0/1 Knapsack problem using Dynamic Programming approach.
5. Shortest Path Algorithm:
Find shortest paths from a given source vertex using Dijkstra's Algorithm.
6. Minimum Spanning Tree – I:
Find Minimum Cost Spanning Tree using Kruskal's Algorithm.
7. Graph Traversal:
 - a) Perform Breadth First Search (BFS) to print all reachable nodes.
 - b) Use Depth First Search (DFS) to check graph connectivity.

Sanjiv *Plav* *Arjun* *SSR* *Sanjay Singh* *Sanjay Singh*

8. Subset Sum Problem (Backtracking):
Find subsets of a given set whose sum equals a given value. Display all possible solutions or report if no solution exists.
9. Traveling Salesperson Problem (TSP):
Implement an exact solution and compare it with an approximation algorithm. Analyze and compute approximation error.
10. Minimum Spanning Tree – II:
Find Minimum Cost Spanning Tree using Prim's Algorithm.
11. All-Pairs Shortest Paths:
Implement Floyd's Algorithm and parallelize it using OpenMP. Analyze speed-up achieved.
12. N-Queens Problem:
Solve N-Queens problem using Backtracking technique.

Ravish *Shaw* *Atyain* *SSK* *Shuchitaiya* *Dr*

Course Title: Machine Learning	
Course Code: FYCS-C504	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

COURSE OBJECTIVES:

1. To introduce the basic concepts and types of machine learning (Supervised, Unsupervised, and Reinforcement) and the mathematical foundations required to build them.
2. To enable students to understand and implement core algorithms from scratch, such as the Perceptron, Multi-Layer Perceptrons, and Decision Trees.
3. To teach students how to rigorously evaluate the performance of machine learning models using statistical metrics, confusion matrices, and cross-validation.
4. To provide expertise in dimensionality reduction and feature engineering techniques like PCA and Factor Analysis to handle high-dimensional data.
5. To introduce non-traditional optimization techniques, including Genetic Algorithms and Reinforcement Learning, for solving complex search and decision-making problems.

COURSE OUTCOMES:

After completion of the course, students will be able to:

- CO1** Students can identify which type of learning paradigm (Supervised vs. Unsupervised) is appropriate for a given real-world problem and justify the choice.
- CO2** Students design and code neural network architectures, including Back-propagation and Radial Basis Function networks, to solve classification and regression tasks.
- CO3** Students can apply ensemble techniques like Boosting, Bagging, and Random Forests to improve the accuracy and robustness of baseline models.
- CO4** Students can utilize clustering algorithms (K-means) and dimensionality reduction tools (PCA) to discover hidden patterns and structures within unlabeled datasets.
- CO5** Students can demonstrate the ability to diagnose model issues such as overfitting or underfitting and apply regularization or pruning to optimize the model for deployment.

Syllabus

Module 1

[09 Hours]

Essentials of Machine Learning (ML), Types of learning (Supervised, Unsupervised, Reinforcement), Designing a learning system, and perspectives or issues in ML, Mathematical Foundations (Vectors, Matrices, and Probability basics for ML), Linear Discriminants (The Perceptron, Linear Regressors, and the Perceptron Convergence Theorem) Evaluating ML Algorithms (Overfitting, training/testing sets, the Confusion Matrix, and Accuracy Metrics and ROC curves)

[Handwritten signatures and initials]

Module 2

[09 Hours]

The Multi-Layer Perceptron (MLP) (Going forwards Biases and Activation functions, going backwards back-propagation of error), Optimization (Deriving back-propagation, Stochastic Gradient Descent, and momentum), Radial Basis Functions (RBF) (Receptive fields, RBF networks, and interpolation using splines)

Module 3

[09 Hours]

Dimensionality Reduction (The Curse of Dimensionality, Principal Components Analysis (PCA), and Factor Analysis), Unsupervised Learning (K-means clustering, Vector Quantization, and Self-Organizing Maps (SOM)), Probabilistic Learning (The Expectation-Maximization (EM) Algorithm and Nearest Neighbor methods)

Module 4

[09 Hours]

Decision Trees & Ensemble Learning, Learning with Trees (Decision Trees (ID3 algorithm), Classification and Regression Trees (CART), and Pruning.)
Ensemble Learning (Decision by Committee, Bagging, Boosting (AdaBoost), and Random Forests)

Module 5

[09 Hours]

Support Vector Machines (SVM) (Linear and non-linear classifiers, the Kernel Trick, and Support Vector Regression), Evolutionary Learning (The Genetic Algorithm (GA), string representation, fitness functions, and genetic operators Crossover and Mutation), Reinforcement Learning (Basic concepts, Q-Learning, and Markov Decision Processes)

Textbook

1. Marsland, S. Machine Learning: An Algorithmic Perspective. CRC Press.).
2. Tom M. Mitchell. Machine Learning. McGraw Hill Education.

Reference Books

1. Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer.
2. Ethem Alpaydin Introduction to Machine Learning. MIT Press.

Practical's:

1. Implement the basic Perceptron to solve a linearly separable Two-Class problem. Visualize the decision boundary shifting during iterations.
2. Build a univariate linear regression model from scratch to predict house prices or student grades. Compare the performance of Batch vs. Stochastic Gradient Descent.
3. Write a script to calculate a Confusion Matrix, and from it, derive Accuracy, Precision, Recall, and F_1 Score. Generate an ROC curve for a binary classifier.

Handwritten signatures: Laxi, [Signature], [Signature], [Signature], [Signature], Shuchi Priya, [Signature]

4. Implement the Back-propagation algorithm for a simple 3-layer network (Input-Hidden-Output) to solve the XOR problem.
5. Write a script to compare the convergence rates and final accuracy of a Neural Network using different activation functions: Sigmoid, Tanh, and ReLU on the MNIST digit dataset.
6. Write a script to design an RBF network for function approximation (e.g., approximating a sine wave) and observe how the number of basis functions affects the fit.
7. Write script to implement the K-means algorithm to cluster unlabeled data. Experiment with the "Elbow Method" to determine the optimal number of clusters k.
8. Perform PCA on a high-dimensional dataset (like the Iris dataset) to reduce it to 2D or 3D. Visualize the clusters in the reduced feature space.
9. Implement the ID3 algorithm using Information Gain as the splitting criterion. Visualize the resulting tree for a categorical dataset (e.g., "Play Tennis" or "Titanic").
10. Use a library to implement a Random Forest and compare its accuracy and robustness against a single Decision Tree on a noisy dataset.
11. Apply an SVM to a non-linearly separable dataset. Experiment with different kernels (Linear, Polynomial, RBF) and observe the change in the decision boundary.
12. Implement a basic Genetic Algorithm to find the maximum value of a mathematical function or solve the "Travelling Salesman Problem".
13. Write a Python script that takes a user's "bad" prompt and uses a few-shot prompting technique to automatically refine it for better AI output.

Laish *Planned* *Again* *SSK* *Shuch Singh* *SR*

Course Title: Software Engineering	
Course Code: FYCS-C601	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To understand the fundamental concepts of software engineering and the evolving role of software in modern systems.
2. To study various software process models and their applications in software development.
3. To learn techniques for requirement analysis, specification, and validation.
4. To understand software design principles, methodologies, and user interface design.
5. To apply different software testing strategies and techniques for quality assurance.
6. To understand software metrics, estimation techniques, quality assurance, and configuration management practices.

Course Outcomes (COs):

After successful completion of this course, students will be able to:

- CO1:** Explain the fundamental concepts of software engineering and the evolving nature of software systems.
- CO2:** Apply appropriate software process models for the development of software projects.
- CO3:** Analyze and document software requirements using standard requirement engineering techniques.
- CO4:** Design software systems using suitable design principles, methodologies, and user interface techniques.
- CO5:** Apply various software testing strategies and techniques to ensure software quality and reliability.
- CO6:** Estimate software project cost and effort using standard metrics and models, and understand software quality assurance and configuration management practices.

Syllabus

Module I

[9 Hours]

Basics of Software: Evolving Role of Software, Changing Nature of Software, Legacy Software, Software Engineering – A layered Technology.

Process Model: Process Frame work, Process Patterns, Process Models, Waterfall Model, Incremental Process Models, Evolutionary Process Models, Specialized Process Models, Unified Process Model, Agile Process Model

[Handwritten signatures and initials]

Module II

[9 Hours]

Software Requirement Analysis and specification: An approach to design and construction, Requirement specification, Initiating the Requirement Engineering Process

Functional and Non-Functional Requirements, Developing Use case, Building the Analysis Model, Negotiating Requirements, Validating Requirements, SRS document

Module III

[9 Hours]

Design Engineering: Design Process Approaches, Design Concepts, Quality, Design Models, Pattern Based Software Design. Software design approaches, Cohesion and Coupling, Software Design Object oriented vs Function Oriented designs. User Interface Design –Input and Output Interfaces, Component –Based GUI Development, a User Interface design Methodology.

Module IV

[9 Hours]

Testing Strategies and Testing Tactics: Strategic Approach to Software Testing, Validation Testing, System Testing, Requirement-based Testing, Acceptance Testing, White Box Testing(structural testing), Basis Path Testing, Control Structure Testing, Black Box Testing(functional testing), Object-Oriented Testing Methods.

Metric for process and Estimation Techniques: Process metrics, Software Measurement, Software Project Estimation, Decomposition Techniques, Empirical Estimation Models, Estimation for Object Oriented Projects Specialized Estimation Techniques, COCOMO models.

Module V

[9 Hours]

Software Quality and Configuration Management: Quality Concepts, Software Quality Assurance, Software Reliability, Software Configuration Management, SCM Repository, SCM Process. Computer Aided Software Engineering (CASE) Tools, Types of CASE tools. Current trends in Software Engineering

Textbooks

1. Software Engineering: A Practitioner's Approach by Roger S. Pressman and Bruce R. Maxim, McGraw-Hill.
2. Software Engineering by Ian Sommerville, Pearson.

Reference Books

1. Fundamentals of Software Engineering by Rajib Mall, PHI Learning.
2. Software Engineering: Theory and Practice by Shari Lawrence Pfleeger and Joanne M. Atlee, Pearson.
3. Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin, Prentice Hall.

[Handwritten signatures and marks at the bottom of the page]

Practical's:

1. Develop a problem statement for a selected software project and define its scope and objectives.
2. Perform requirement analysis by creating Data Flow Diagrams (DFD) for the selected system.
3. Prepare a Data Dictionary and develop Use Case diagrams for the given system.
4. Compute Function Points (FP) and estimate software size for the selected project.
5. Perform effort estimation using suitable estimation techniques.
6. Prepare project management artifacts including schedule, risk table, and timeline chart (Gantt chart).
7. Design the system architecture using appropriate architectural design techniques.
8. Perform data design and component-level design for the selected system.
9. Design user interfaces including input and output screens for the application.
10. Develop and execute test cases for the system using black-box and white-box testing techniques.
11. Implement validation and system testing for the developed modules.
12. Prepare complete software documentation including SRS, design document, test report, and project report.

Rajiv *Prasanna* *Arjun* *SSG* *Sandeep Bhatia* *San*

Course Title: Information And Coding Theory	
Course Code: FYCS-C602	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To understand the ultimate theoretical limits on data compression and transmission rates in any communication system.
2. To develop the ability to model communication channels (BSC, BEC, AWGN) using probabilistic frameworks.
3. To master the techniques of source coding to eliminate redundancy and represent data in its most compact form.
4. To study the design and implementation of error-control codes that ensure data integrity over noisy mediums.
5. To bridge classical Shannon theory with modern technologies like 5G, Deep Space Network, and data storage.

Course Outcomes:

After completion of the course, students will be able to:

- CO-1: Students can calculate entropy, mutual information, and channel capacity for various discrete and continuous sources.
- CO-2: Students can design and evaluate optimal prefix codes (e.g., Huffman) for diverse source distributions.
- CO-3: Students will be able to select and implement appropriate error-correcting codes (Linear Block, Cyclic, or Convolutional) based on channel characteristics.
- CO-4: Student will be able to use tools like the Viterbi algorithm and syndrome decoding to analyze the trade-off between code rate, complexity, and error probability.
- CO-5: Students can critically assess real-world communication systems against the Shannon-Hartley theorem to determine their efficiency.

Syllabus

Module I

[9 Hours]

Discrete Sources: (Definition of alphabets and random variables), Measures of Information (Self-information, Entropy (H), and Joint Entropy), Conditional Measures (Conditional Entropy and Mutual Information ($I(X;Y)$)). Key Inequalities (Jensen's Inequality and its application to the non-negativity of information measures), The Chain Rule (Decomposing complex information systems into simpler conditional steps)

[Handwritten signatures]

Module 2

[9 Hours]

Variable Length Coding (Unique decodability and the Prefix property (Instantaneous codes), Kraft's Inequality, Optimal Source Coding, Construction and efficiency of Huffman Coding, Shannon's First Theorem (The Source Coding Theorem and the fundamental limit of compression), Practical Algorithms: Overview of Dictionary-based coding (e.g., Lempel-Ziv).

Module 3

[9 Hours]

Channel Models: Binary Symmetric Channel (BSC), Binary Erasure Channel (BEC), and the Z-channel, Channel Capacity (C): Maximizing mutual information over input distributions, Shannon's Second Theorem: The Noisy-Channel Coding Theorem, Differential Entropy: Extending information theory to continuous variables and Gaussian channels, The Shannon-Hartley Theorem: The limit of bitrate in the presence of noise and finite bandwidth ($C = B \log_2(1 + S/N)$).

Module 4

[9 Hours]

Principles of Redundancy, The trade-off between Code Rate (R) and reliability), Linear Block Codes, Generator matrices (G) and Parity-check matrices (H), Hamming Distance, Geometric interpretation of error correction and detection capabilities. Syndrome Decoding, Error patterns, Cyclic Codes, Polynomials, Cyclic Redundancy Checks(CRC).

Module 5

[9 Hours]

State machine representations and Trellis diagrams, The Viterbi Algorithm, Maximum likelihood decoding for sequence estimation, Iterative Decoding(Turbo Codes or LDPC (Low-Density Parity-Check) codes), Information Theory in Networks(Multi-user channels and Network Coding)

Textbook

1. Information and Communication Theory by Stefan Host.
2. Elements of Information Theory by Thomas M. Cover and Joy A. Thomas

Reference Books

1. Information Theory, Inference, and Learning Algorithms by David J. C. MacKay.
2. A First Course in Coding Theory by Raymond Hill

Practical's:

1. Write a script that takes a string of text or a file and calculates its empirical entropy $H(X)$ based on character frequency.
2. Implement an automated Huffman coding algorithm that generates a prefix-free codebook for a given set of source probabilities.
3. Program a basic sliding-window compressor to observe how dictionary-based coding performs on repetitive vs. random data.

Rajin *Deepak* *Atyain* *SSG* *Shuchi Baisa* *Shr*

4. Create a script to calculate and visualize the mutual information $I(X;Y)$ between two correlated datasets (e.g., weather variables).
5. Build a Binary Symmetric Channel (BSC) function that introduces bit-flips at a programmable probability p and measures the resulting Bit Error Rate (BER).
6. Write a program to plot the capacity of a BSC and a Binary Erasure Channel (BEC) as a function of the error probability p in $[0, 1]$.
7. Implement the generator matrix G and parity-check matrix H to encode 4-bit messages and correct single-bit errors.
8. Write a script that generates a syndrome-to-error-pattern lookup table for a given linear block code.
9. Implement a CRC-16 encoder and detector using polynomial division (modulo-2 arithmetic).
10. Compare the performance (BER) of a hard-decision decoder against a soft-decision decoder in an Additive White Gaussian Noise (AWGN) channel.
11. Program a trellis decoder for a simple (2, 1, 3) convolutional code to reconstruct a message from a noisy stream.
12. Create a simulation that plots the performance of different codes (Hamming, Cyclic, Convolutional) against the theoretical Shannon Capacity limit.

Rashid *Planned* *Arif* *SSG* *Saudi Praga* *Br*

Course Title: Cloud Computing	
Course Code: FYCS-C603	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To provide a comprehensive understanding of cloud computing concepts, paradigms, and architectures.
2. To explore various distributed computing models and their evolution toward cloud computing.
3. To understand cloud service models (IaaS, PaaS, SaaS) and deployment strategies.
4. To develop knowledge of cloud programming environments and resource management techniques.
5. To familiarize students with leading cloud service providers and real-world cloud platforms.

Course Outcomes (COs):

After completion, students will be able to:

- CO1:** Understand various computing paradigms and their role in cloud evolution.
CO2: Analyze cloud computing architecture, characteristics, and deployment models.
CO3: Evaluate different cloud service models and their suitability for applications.
CO4: Apply cloud computing techniques for deployment, migration, and management of applications.
CO5: Compare major cloud service providers and their offerings.
CO6: Demonstrate practical skills in configuring and managing cloud resources.

Syllabus

Module I

[9 Hours]

Computing Paradigms: High-Performance Computing, Parallel Computing, Distributed Computing, Cluster Computing, Grid Computing, Cloud Computing, Bio computing, Mobile Computing, Quantum Computing, Optical Computing, Nano computing.

Module II

[9 Hours]

Cloud Computing Fundamentals: Motivation for Cloud Computing, The Need for Cloud Computing, Defining Cloud Computing, Definition of Cloud computing, Cloud Computing Is a Service, Cloud Computing Is a Platform, Principles of Cloud computing, Five Essential Characteristics, Four Cloud Deployment Models

Module III

[9 Hours]

Cloud Computing Architecture and Management: Cloud architecture, Layer, Anatomy of the Cloud, Network Connectivity in Cloud Computing, Applications, on the Cloud, Managing the Cloud,

[Handwritten signatures and initials]

Managing the Cloud Infrastructure Managing the Cloud application, Migrating Application to Cloud, Phases of Cloud Migration Approaches for Cloud Migration.

Module IV

[9 Hours]

Cloud Service Models: Infrastructure as a Service, Characteristics of IaaS. Suitability of IaaS, Pros and Cons of IaaS, Summary of IaaS Providers, Platform as a Service, Characteristics of PaaS, Suitability of PaaS, Pros and Cons of PaaS, Summary of PaaS Providers, Software as a Service, Characteristics of SaaS, Suitability of SaaS, Pros and Cons of SaaS, Summary of SaaS Providers, Other Cloud Service Models.

Module V

[9 Hours]

Cloud Service Providers: Google cloud, Amazon Web Services, Microsoft, Windows Azure, SharePoint, IBM, Cloud Models, IBM Smart Cloud, SAP Labs, Virtualization Services Provided by SAP, Sales force, Sales Cloud, Service Cloud: Knowledge as a Service, Rack space, VMware, Manjra soft, Aneka Platform.

Text Books:

1. Essentials of cloud Computing: K. Chandrasekhran, CRC press, 2014

References:

1. Rajkumar Buyya, James Broberg, Andrzej M. Goscinski – Cloud Computing: Principles and Paradigms, Wiley, 2011
2. Kai Hwang, Geoffrey C. Fox, Jack J. Dongarra – Distributed and Cloud Computing, Elsevier, 2012
3. Tim Mather, Subra Kumaraswamy, Shahed Latif – Cloud Security and Privacy, O'Reilly, 2011

Practical's:

1. Study of Cloud Computing Platforms
Introduction to cloud platforms such as Amazon Web Services and Microsoft Azure; exploration of dashboard, services, and cloud models (IaaS, PaaS, SaaS).
2. Launching and Configuring Virtual Machine
Creation and configuration of a virtual machine instance; connecting using SSH/Remote Desktop.
3. Cloud Storage Management
Creation of storage buckets; uploading, downloading, and managing files with appropriate access permissions.
4. Backup and Recovery Mechanisms
Performing data backup using snapshots and restoring data in a cloud environment.
5. Monitoring of Cloud Resources
Monitoring CPU, memory, and network usage using built-in tools; setting alerts and thresholds.
6. Virtual Private Cloud (VPC) Configuration
Creation of VPC, subnet configuration, and routing table setup.

Rajkumar Buyya *James Broberg* *Andrzej M. Goscinski* *Kai Hwang* *Geoffrey C. Fox* *Jack J. Dongarra* *Tim Mather* *Subra Kumaraswamy* *Shahed Latif*

7. Cloud Security Configuration
Implementation of security groups/firewall rules; key-based authentication and access control.
8. Static Website Deployment on Cloud
Hosting and configuring a static website using cloud storage services.
9. IP Addressing and Domain Configuration
Assignment of static IP; DNS configuration and domain mapping to cloud resources.
10. Application Deployment using PaaS
Deployment of a web application using platforms such as Google Cloud Platform App Engine or Azure App Service.
11. Cloud Migration Exercise
Migration of a local application to cloud infrastructure and analysis of performance.
12. Comparative Study of Cloud Service Providers
Comparative analysis of Amazon Web Services, Microsoft Azure, and Google Cloud Platform based on services, pricing, and features.

[Handwritten signatures and marks]

Course Title: Data Mining	
Course Code: FYCS-C604	Nature of Course: Core
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To introduce fundamental concepts of data mining and data preprocessing
2. To understand classification, clustering, and association techniques
3. To evaluate models and analyze patterns from large datasets
4. To explore anomaly detection techniques in real-world scenarios

Course Outcomes (COs):

After completion of the course, students will be able to:

CO1: Understand data mining concepts and preprocessing methods

CO2: Apply classification techniques to real-world datasets

CO3: Analyze patterns using association rule mining

CO4: Perform clustering and interpret data structures

CO5: Detect anomalies using statistical and machine learning approaches

Syllabus

Module I

[9 Hours]

Data Mining, Data Mining tasks, Types of Data, Data Quality, Data Processing, Measures of Similarity and Dissimilarity, Exploring Data: Data Set, Summary Statistics, Visualisation, OLAP and multidimensional data Analysis

Module II

[9 Hours]

Basic Concepts, Decision Trees, and Model Evaluation: General approach for solving a classification problem, Decision Tree induction, Model overfitting: Due to the presence of noise, due to a lack of representative samples, Evaluating the performance of a classifier, Rule-based classifier, Nearest Neighbours classifier, Bayesian Classifier.

Module III

[9 Hours]

Problem Definition, Frequent Item-set generation, Rule generation, compact representation of frequent item sets, FP-Growth Algorithm, handling categorical and continuous attributes, concept hierarchy, sequential patterns: Problem formulation, sequential pattern discovery, Apriori-like method, Candidate generation, Candidate pruning, support counting.

Module IV

[9 Hours]

Overview, Types of Clustering, Types of clusters, K-means, Basic Agglomerative Hierarchical clustering algorithm, DBSCAN, Cluster evaluation: Supervised and unsupervised cluster evaluation.

Ravi *Harsh* *Arjun* *SSR* *Shuchi Bhatia* *[Signature]*

Module V

[9 Hours]

Preliminaries: Causes of Anomalies, Approaches of Anomaly Detection, Issues. Statistical approaches: Detecting outliers in a univariate Normal Distribution, Outliers in a Multivariate Normal Distribution, A mixture model approach for anomaly Detection, Proximity-based outlier detection, Density-based outlier detection, Clustering-based outlier detection

Text Books:

1. Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, 2nd edition, Pearson education, 2018.

References:

1. Jlawei Han & Micheline Kamber, Data Mining, "Concepts and Techniques", 3rd edition, Morgan Kaufmann Publishers, 2012.
2. Margaret H Dunham, Data Mining Introductory and advanced topics, 6th edition, Pearson Education, 2009
3. Arun K Pujari, Data Mining Techniques, 1st edition, University Press, 2005.
4. GK Gupta, Introduction to Data Mining with Case Studies, 3rd edition, Prentice Hall, 2014.
5. K. P. Soman, Shyam Diwakar, V. Ajay, Data Mining Theory and Practice, 1st edition, PHI, 2006.

Practical's:

1. Perform data preprocessing on a given dataset to handle missing values, noise, and inconsistencies, and apply normalization techniques.
2. Apply data transformation and reduction techniques including discretization, aggregation, and feature selection on a dataset.
3. Compute similarity and dissimilarity measures such as Euclidean distance, Manhattan distance, Cosine similarity, and Jaccard coefficient for given data objects.
4. Explore and visualize a dataset using statistical summaries and plots such as histograms, box plots, and scatter plots.
5. Implement a Decision Tree classifier for a given dataset and evaluate its performance using appropriate metrics.
6. Implement the k-Nearest Neighbor (k-NN) algorithm for classification and analyze the effect of different values of k.
7. Implement the Naïve Bayes classifier for a given dataset and evaluate its classification performance.
8. Implement the Apriori algorithm to generate frequent itemsets and association rules, and compute support and confidence.
9. Implement the FP-Growth algorithm for frequent pattern mining and compare its performance with Apriori.
10. Apply K-Means clustering on a dataset, determine the optimal number of clusters, and evaluate the clustering results.

Ravi *Deep* *Arjun* *SSR* *Sanjay Singh* *Sanjay*

11. Implement Hierarchical clustering and DBSCAN on a dataset and compare their clustering results.
12. Perform anomaly detection using statistical, proximity-based, or clustering-based methods to identify outliers in a dataset.

Rach

Plam

Again

SSG

Shuchi Parag

Elective Course (Pool-2)

Course Title: Computer Architecture	
Course Code: MDC-1	Nature of Course: Multi-Disciplinary Course
Credit: 4	Classes Per Week: L: 3 T: 0 P: 02

Course Objectives:

1. To understand how the Computer Systems work and their basic principles.
2. To ALU operations, Fixed-point Arithmetic and Floating-point Arithmetic.
3. To Instruction set architecture and execution cycle and different types of Control Units.
4. To Memory System Design and different types of Memories.
5. To explain how the I/O devices are accessed and their principles.
6. To the concepts of pipelining and Parallel Processing.

Course Outcomes:

After completion of the course, students will be able to:

- CO1:** Make students understand the working of Computer Systems work and its basic principles.
- CO2:** Make students understand ALU operations, Fixed point Arithmetic and floating-point Arithmetic.
- CO3:** Make students understand different types of instructions and Instruction execution cycle along with types of Control Units.
- CO4:** Make students understand Memory System Design and different types of Memories like cache and virtual memory.
- CO5:** Make students understand the access to I/O devices and their principles.
- CO6:** Make students understand the concepts of pipelining and Parallel Processing.

Syllabus

Module I

[9 Hours]

Functional Blocks of a Computer: CPU, memory, Input-Output Unit, control unit, Basic Operational Concepts, Von Neumann Architecture. Data Representation: Signed number representation, fixed- and floating-point Representations, Computer Arithmetic– integer Addition and Subtraction, Ripple carry adder, carry look-ahead adder, Multiplication – shift-and-add, Booth multiplier, carry save Addition, Division restoring and non-restoring techniques, Floating point arithmetic.

Module II

[9 Hours]

46 | Page

Department of Cyber Defence

Lawir *Ram* *Agarwal* *SSR* *Shuchi Singh*

Introduction to x86 Architecture Instruction set. Architecture of a CPU: Register Transfer Language, Register Transfer, Memory Transfer, Instruction Cycle, Addressing modes, Instruction set, CISC vs RISC Architecture. CPU Control unit design: Hardwired and Micro-programmed design approaches.

Module III

[9 Hours]

Memory system design: Semiconductor memory technologies, internal memory organisation. Memory Organisation: Memory Hierarchy, Memory interleaving, Cache memory, mapping functions, write policies, Virtual Memory Management-Paging.

Module IV

[9 Hours]

Peripheral devices and their characteristics: I/O device interface, Data Transfer Modes, I/O transfers – program controlled, interrupt driven and DMA, interrupts and exceptions, I/O device interfaces–SCSI, USB

Module V

[9 Hours]

Pipelining: Basic concepts of pipelining, pipelining hazards. Parallel Processors: Introduction to parallel processors, Shared Memory Multiprocessors and cache coherency.

Textbooks

1. "Computer Organization and Embedded Systems", 6th Edition by Carl Hamacher, McGraw Hill Higher Education.
2. "Computer Organization and Design", The Hardware/Software Interface", 5th Edition by David A. Patterson and John L. Hennessy, Elsevier.
3. Computer System Architecture, 3rd Edition by M. Morris Mano, Pearson.

Reference Books

1. "Computer Architecture and Organisation", 3rd Edition by John P. Hayes, WCB /McGraw-Hill
2. "Computer Organisation and Architecture: Designing for Performance", 10th Edition by William Stallings, Pearson Education.
3. "Computer System Design and Architecture", 2nd Edition by Vincent P. Heuring and Harry F. Jordan, Pearson Education.

Practical's:

1. To implement and verify the truth tables of fundamental logic gates (AND, OR, NOT, NAND, NOR, XOR, XNOR) using software simulation.
2. To design and simulate a Half Adder and Full Adder circuit and verify the outputs for all combinations of input bits.
3. To perform and verify basic register operations (MOV, ADD, SUB, INC, DEC) using a simulator.

Basir *Flavio* *Again* *SSK* *Sandeep Singh*

4. To simulate and verify basic ALU operations — addition, subtraction, AND, OR, XOR, and NOT — using 4-bit inputs.
5. To simulate memory read and write operations by storing and retrieving values at specified memory addresses.
6. To execute simple assembly language instructions including loading a value into a register, adding two registers, and storing the result into memory.
7. To simulate stack operations — PUSH, POP, CALL, and RETURN — and observe how the stack grows and shrinks during execution.
8. To simulate the Program Counter (PC) and demonstrate the instruction execution cycle: Fetch → Decode → Execute.
9. To write and execute a simple assembly language program to add two numbers, subtract two numbers, and find the largest of two numbers.
10. To simulate cache memory with direct or associative mapping and demonstrate how memory addresses are mapped to cache lines.

Laure *Shan* *Arjun* *SSG* *Sanchi Pariza* *SV*

Course Title: Introduction to Digital Forensic	
Course Code: MDC-2	Nature of Course: Multi-Disciplinary Course
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To explain legal and procedural aspects of digital forensic investigations based on roles and responsibilities.
2. To develop the ability to perform basic digital forensic analysis.
3. To demonstrate the use of standard digital forensic tools and techniques.
4. To guide and execute digital forensic investigation exercises.
5. To understand current practices, limitations, and challenges in technology, policy, and legal domains.
6. To prepare students as competent next-generation cyber crime investigators.

Course Outcomes (COs):

After completion of the course, students will be able to:

- CO1:** Understand fundamental concepts of digital forensics, cybercrimes, and investigative principles.
- CO2:** Apply forensic procedures for evidence acquisition, preservation, and analysis.
- CO3:** Identify, collect, and analyze digital evidence using appropriate tools and techniques.
- CO4:** Utilize forensic tools for recovering and examining digital artifacts.
- CO5:** Interpret legal, ethical, and procedural aspects of digital evidence handling.
- CO6:** Relate digital forensic practices to real-world investigation scenarios.

Syllabus

Module I

[9 Hours]

Forensics science, Lockard's exchange principle and importance of it in forensics, Digital forensics fundamentals, Benefits of forensics, cybercrimes, cyber-attacks and types, digital evidence and courts legal concerns and private issues, Rules of Digital Forensics, Good Forensic Practices, Principles of Digital Evidence. Overview of types of Computer Forensics – Network Forensics, Mobile Forensics, Social Media Forensics and E-mail Forensics. Services offered by Digital Forensics. First Responder – Role, Toolkit and Do's and Don'ts

Module II

[9 Hours]

Introduction to Cyber Crime Investigation, securing a computer incident or crime, Procedure for Search and seizure of digital evidences in cyber-crime incident- Forensics Investigation Process, Acquisition, Duplication & Preservation of evidences, Examination and Analysis of

20/11/20 *20/11/20* *20/11/20* *20/11/20* *20/11/20*

evidences, obtaining digital hash, Storing of Evidences, Documentation and Reporting, cardinal rules of Digital forensics

Module III

[9 Hours]

Identification of digital evidence, types of digital evidences, Data acquisition- understanding storage formats and digital evidence, determining the best acquisition method, acquisition tools, Maintaining the Chain of Custody, Search and Seizure of Volatile and Non-volatile Digital Evidence, Imaging and Hashing of Digital Evidences, imaging tools, write blockers, hashing algorithms, Cryptography- Private key and public key cryptography, encryption and decryption.

Module IV

[9 Hours]

Software and Hardware tools used in cybercrime investigation – Open Source and Proprietary tools. Importance of Log Analysis in forensic analysis. Understanding Storage Formats for Digital Evidences – Raw Format, Proprietary Formats, Advanced Forensic Formats. Introduction to deleted File Recovery, Steganography and Steganalysis, Data Recovery Tools, window artifacts, file systems

Module V

[9 Hours]

Current computer forensics tools- software, hardware tools, Forensic Workstations, reporting and documentation, Forensic Scenario and analysis, Incident response team, data and metadata, DFIR, Consent, malware and antivirus, Knowledge, skills and attributes of digital forensics practitioners- First responders, Digital forensic investigators, Digital forensic analyst

Text Books

1. Nina Godbole, Sunit Belapore – Cyber Security: Understanding Cyber Crimes, Computer Forensics and Legal Perspectives, Wiley, 2011
2. Bill Nelson, Amelia Phillips, Christopher Steuart – Guide to Computer Forensics and Investigations, Cengage

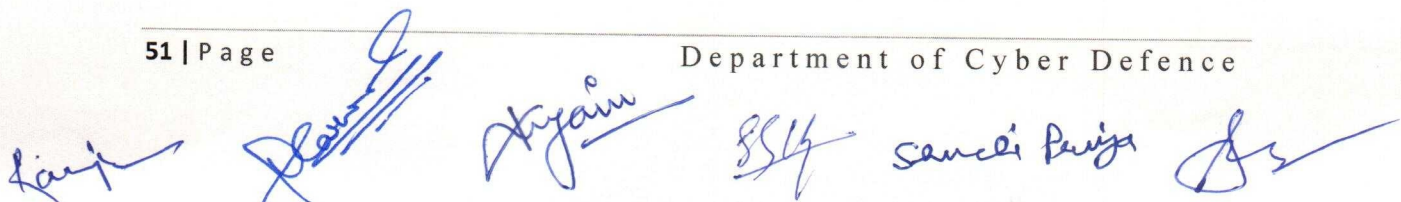
Reference Books

1. LNIN National Institute of Criminology and Forensic Science – A Forensic Guide for Crime Investigators, 2016
2. Anthony Reyes, Jack Wiles – The Best Damn Cybercrime and Digital Forensic Book, Syngress
3. Cory Altheide, Harlan Carvey – Digital Forensics with Open Source Tools, Syngress
4. Harlan Carvey – Windows Forensic Analysis Toolkit, Syngress
5. Angus M. Marshall – Digital Forensics: Digital Evidence in Criminal Investigation, Wiley
6. Bill Nelson et al. – Guide to Computer Forensics and Investigations, 4th Edition



Practical's:

1. Introduction to Digital Forensics Tools
Study and installation of basic forensic tools such as Autopsy and FTK Imager.
2. Disk Imaging and Acquisition
Create a forensic image of a storage device using imaging tools and verify integrity.
3. Hash Value Generation and Verification
Generate hash values (MD5/SHA) and verify data integrity of digital evidence.
4. File System Analysis
Examine file systems (FAT/NTFS) and identify file structures and metadata.
5. Deleted File Recovery
Recover deleted files using forensic recovery tools and analyze recovered data.
6. Registry and Log File Analysis
Analyze system logs and registry entries to reconstruct user activities.
7. Email Forensics
Analyze email headers and identify sender information and routing path.
8. Network Forensics (Packet Analysis)
Capture and analyze network packets using tools like Wireshark.
9. Steganography and Steganalysis
Hide and extract data using steganography tools and detect hidden information.
10. Malware Analysis (Basic)
Perform static analysis of suspicious files using antivirus and sandbox tools.
11. Mobile Forensics (Simulation-Based)
Extract and analyze mobile data such as call logs, messages, and app data (using tools/simulators).
12. Forensic Case Study and Report Writing
Analyze a simulated cybercrime case and prepare a structured forensic report.



Course Title: Internet Of Things	
Course Code: MDC-3	Nature of Course: Multi-Disciplinary Course
Credit: 4	Classes Per Week: L: 3 T: 0 P: 2

Course Objectives:

1. To introduce the terminology, technology and its applications.
2. To introduce the concept of M2M (machine-to-machine) with necessary protocols.
3. To introduce the Python Scripting Language, which is used in many IoT devices.
4. To introduce the Raspberry PI platform, which is widely used in IoT applications.
5. To introduce the implementation of web-based services on IoT devices.

Course Outcomes:

After completion of the course, students will be able to:

CO1: Understand IoT value chain structure (device, data cloud), application areas and technologies involved.

CO2: Understand IoT sensors and technological challenges faced by IoT devices, with a focus on wireless, energy, power, and sensing modules.

CO3: Market forecast for IoT devices with a focus on sensors.

CO4: Explore and learn about Internet of Things with the help of IOT devices

Syllabus

Module I

[9 Hours]

Definition and Characteristics of IoT, Sensors, Actuators, Physical Design of IoT(IoT Protocols, IoT communication models, IoT Communication APIs), IoT-enabled Technologies (Wireless Sensor Networks, Cloud Computing, Embedded Systems, IoT Levels and Templates), Domain Specific IoTs (Home, City, Environment, Energy, Agriculture and Industry.)

Module II

[9 Hours]

Software-defined networks, network function virtualization, difference between SDN and NFV for IoT, Basics of IoT System Management with NETCONF, YANG- NETCONF, YANG, SNMP NETOPEER

Module III

[9 Hours]

Introduction to Arduino and Raspberry Pi(Installation, Serial Interfaces, SPI, I2C) Controlling Hardware (Connecting LED, Buzzer, Switching High Power devices with transistors, Controlling AC Power devices with Relays, Controlling servo motor, speed control of DC Motor, unipolar and bipolar Stepper motors)

Module IV

[9 Hours]

Light sensor, temperature sensor with thermistor, voltage sensor, ADC and DAC, Temperature and Humidity Sensor DHT11, Motion Detection Sensors, Wireless Bluetooth



Sensors, Level Sensors, USB Sensors, Embedded Sensors, Distance Measurement with ultrasound sensor

Module V

[9 Hours]

Introduction to Cloud Storage models and communication APIs Web Server – Web server for IoT, Cloud for IoT, Python web application framework Designing a RESTful web API.

Text Books:

1. Internet of Things - A Hands-on Approach, Arshdeep Bahga and Vijay Madisetti, Universities Press, 2015, ISBN: 9788173719547.
2. Peter Waher, 'Learning Internet of Things', Packt Publishing, 2015 3. Editors Ovidiu Vermesan.

Reference Books:

1. Getting Started with Raspberry Pi, Matt Richardson & Shawn Wallace, O'Reilly (SPD), 2014, ISBN: 9789350239759.
2. Raspberry Pi Cookbook, Software and Hardware Problems and solutions, Simon Monk, O'Reilly (SPD), 2016, ISBN 7989352133895.

Practical's:

1. Program an Arduino to blink an LED and trigger a piezo buzzer simultaneously to demonstrate how microcontrollers handle basic digital signals and timing intervals.
2. Interface a DC motor with an NPN transistor acting as a switch, allowing a low-power microcontroller pin to control a high-current load without damaging the board.
3. Use a 5V relay module to create an isolation circuit that allows an Arduino to safely switch a high-voltage AC light bulb on and off.
4. Design an intelligent street lighting system that uses a Light Dependent Resistor (LDR) to read analog voltage levels and automatically trigger an LED when ambient light falls below a specific threshold.
5. Interface a DHT11 digital sensor to capture real-time ambient temperature and humidity data, then format and transmit this information to the Serial Monitor for analysis.
6. Implement an HC-SR04 ultrasonic sensor to calculate the distance of a physical object by measuring sound wave travel time, triggering a buzzer alarm if the object enters a 10cm safety zone.
7. Utilize a potentiometer to generate a variable analog input, then map that 0 to 1023 value to a 0 degree to 180 degree range to control the precise position of a servo motor.
8. Connect a unipolar or bipolar stepper motor to a driver module and write a sequence-based program to execute exactly one full 360 degree revolution in both clockwise and counter-clockwise directions.
9. Apply PWM signals to a DC motor's enable pin to demonstrate how varying the duty cycle can linearly control the rotational speed of the motor.



10. Develop a motion-activated security logic that uses a PIR sensor to detect heat signatures and immediately initiates a visual "SOS" distress signal using an LED.
11. Configure a 16 * 2 character LCD using the I²C communication bus to reduce wiring complexity while displaying dynamic sensor data for a portable user interface.
12. Write a MicroPython script for the Raspberry Pi Pico W to host a local web server, enabling a user to toggle physical hardware (like an LED) via a remote web browser interface.

Handwritten signatures: Louis, David, Arjun, SSG, Shuchi Riga

Course Title: Software Testing	
Course Code: MDC-4	Nature of Course: Multi-Disciplinary Course
Credit: 4	Classes Per Week: L: 3 T: 0 P: 02

Course Objectives:

1. To acquire knowledge of basic principles of testing and debugging.
2. To understand testing levels and generalized pseudo-code examples.
3. To apply various types of testing methodologies.
4. To differentiate between functional and structural testing.
5. To perform fault-based testing and manage process documentation.

Course Outcomes:

After completion of the course, students will be able to:

- CO1 Apply different testing techniques for software validation
- CO2 Design and manage test cases and testing processes
- CO3 Perform functional and non-functional testing
- CO4 Analyze defects and manage bug life cycle
- CO5 Understand automation testing fundamentals

Syllabus

Module I

[9 Hours]

Testing principles, Best practices in testing, Skills required for testing, Testing activities and tasks, Verification vs Validation, QA vs QC, Test strategies for conventional and object-oriented software, Levels of testing: Unit, Integration, System, Validation, Alpha and Beta testing, Non-functional testing basics (Recovery, Security, Stress, Performance), Metrics for source code and testing, Debugging process and strategies

Module II

[8 Hours]

Software testing fundamentals, Black-box and White-box testing, Basis path testing, Flow graph notation, Independent program paths, Graph matrices, Control structure testing, Condition testing, Data flow testing, Loop testing, Graph-based testing methods, Equivalence partitioning, Boundary value analysis, Object-oriented testing: Fault-based, Scenario-based, Random testing, Class and inter-class testing

Module III

[8 Hours]

Security testing, Recovery testing, Configuration testing, Compatibility testing, Installation testing, Performance testing, Parallel testing, Volume testing, Internationalization and localization testing. Test planning, Test analysis, Test design construction and verification,

[Handwritten signatures and marks]

Testing cycles, Final testing and implementation, spost-implementation activities, Software Testing Life Cycle (STLC)

Module IV

[8 Hours]

Introduction to test design, Inputs for test design, Test scenarios and test cases, Test case design techniques, Test case management, Best practices, Test data preparation, Requirement Traceability Matrix (RTM). Test execution cycle, Entry criteria, Smoke and sanity testing, Retesting and regression testing, Test closure activities, Defect/bug concepts, Causes of defects, Defect reporting, Severity and priority, Defect life cycle, Defect tracking tools.

Module V

[8 Hours]

Introduction to Agile methodology, Agile testing process, Basics of automation testing, Introduction to Selenium, WebDriver concepts, Locators (XPath, CSS Selectors), Web elements, Wait mechanisms, TestNG framework basics. Core programming basics (object-oriented concepts, exception handling), Database basics (SQL queries, joins, aggregate functions), Web technologies: HTML (forms, elements), CSS (layouts and styling concepts), Role of these technologies in testing.

Textbooks

1. Roger S. Pressman, Software Engineering: A Practitioner's Approach
2. K.K. Aggarwal & Yogesh Singh, Software Engineering

Reference Books

1. Kshirasagara Naik, Priyadarshi Tripathy: Software Testing and Quality Assurance, Wiley India 2012
2. M. G. Limaye: Software Testing-Principles, Techniques and Tools – McGraw Hill, 2009

Practical's:

1. Prepare test cases for a given software requirement.
2. Perform black-box testing using equivalence partitioning and boundary value analysis.
3. Perform white-box testing using basis path and control flow testing.
4. Design and execute unit and integration test cases.
5. Perform system and regression testing on a sample application.
6. Conduct performance and stress testing analysis.
7. Prepare Requirement Traceability Matrix (RTM) for a project.
8. Execute test cases and generate test reports.
9. Simulate defect reporting and track defect life cycle.
10. Perform automation testing using a web testing tool (conceptual/practical).
11. Execute SQL queries for database validation testing.
12. Conduct a case study on software testing process and reporting.

[Handwritten signatures and initials]