SEMESTER-V

S No	Course Code	Course	Course Type	L	T	P	Credits
1.	CAPC1522M	Software Engineering	Major	4		0	4
2.	ACPC1522N	Operating Systems	Minor	4	0		4
3.	CAPC3522M	Design and Analysis of Algorithms	Major	4	0	2	6
4.	CAPC2522M	Web Development using Frameworks	Major	4	0	2	6
5.	INTCS0005	Internship/ Minor Project	INT	0	0	4	4

SEMESTER-VI

S No	Course Code	Course	Course Type	L	T	P	Credits
1.	CAPC1622M	Cryptography and Network Security	Major	4	0	2	6
2.	CAPC2622M	Artificial Intelligence	Major	4	0	0	4
3.	CAPC3622M	Theory of automation	Major	4	0	0	4
4.	ACPC1622N	Programming with Python	Minor	4	0	2	6

Course Type: - Major
Paper Title: - SOFTWARE ENGINEERING
Credit Weightage: - THEORY -04; TUTORIALS- 02

Semester: - 5th
Paper Code: - CAPC1522M
Batch: - 2023

Course Objective:

- To provide an understanding of the working knowledge of the techniques for estimation, design, testing and quality management of large software development projects.
- To familiarize with process models, software requirements, software design, software testing, software process/product metrics, risk management and quality management.

Course Outcomes:

- Ability to translate end-user requirements into system and software requirements, using e.g. DFDs/UML, and structure the requirements in a Software Requirements Document (SRD).
- Identify and apply appropriate software architectures and patterns to carry out high level design of a system and be able to critically compare alternative choices.
- Will have experience and/or awareness of testing problems and will be able to develop a simple testing report.

UNIT – I

Introduction to Software Engineering: The evolving role of software, changing nature of software, software myths. A Generic view of process: Software engineering- a layered technology, a process framework, the capability maturity model integration (CMMI). Process models: The waterfall model, Spiral model and – Rapid Application Development – Agile Model.

UNIT – II

Requirement Analysis and Specification –: Functional and non-functional requirements, userrequirements, system requirements, interface specification, the software requirements document – SRS.

UNIT – III

Design Engineering: Design process and design quality, design concepts, characteristics – Cohesion & Coupling. Creating an architectural design: software architectures.

Design Model : Data design ERD,Data Flow Diagram (DFD's) ,Activity diagrams, Sequence diagrams.

UNIT – IV

Testing Strategies: A strategic approach to software testing, test strategies for conventional software, black-box and white-box testing, validation testing, system testing, the art of debugging. Metrics for Process and Products: Software measurement, metrics for software quality.

TEXT & REFERENCES:

1. An Integrated Approach to Software Engineering, Pankaj Jalote, Narosa India.
2. Software Engineering, A practitioner's Approach- Roger S. Pressman, McGraw Hill.
3. Software Engineering- Sommerville, 7th edition, Pearson Education.
4. The unified modeling language user guide Grady Booch, James Rambaugh, Ivar Jacobson,Pearson Education.

TUTORIALS -SOFTWARE ENGINEERING (CAPC1522M)

LIST OF TUTORIALS:

1. Conduct interviews or surveys to gather requirements from stakeholders.
2. Create use case diagrams and scenarios to model system behaviour.
3. Apply design principles to create a software architecture or component diagrams.
4. Implement design patterns in a small software project.
5. Develop test cases and perform unit testing on a software component.
6. Conduct integration testing to verify the correct interaction between system components.
7. Use a testing tool (e.g., JUnit) to automate testing and generate test reports.
8. Apply version control techniques using a version control system (e.g., Git).
9. Collaborate with team members using collaborative software development platforms (e.g., GitHub).
10. Analyze a given software system to identify areas for improvement or enhancement.
11. Perform code refactoring to improve code readability and maintainability.
12. Implement bug fixes or feature additions in an existing software system.
13. Create user documentation, such as user manuals or online help systems.
14. Generate technical documentation, including system architecture documents and design specifications.
15. Develop documentation templates and guidelines for consistent documentation practices.

Course Type: - Minor
Paper Title: - **OPERATING SYSTEMS**
Credit Weightage: - THEORY -04; PRACTICALS- 02

Semester: - 5th
Paper Code: - **ACAPC1522N**
Batch: - 2023

Course Objective:

- Introduce operating system concepts (i.e., processes, threads, scheduling, synchronization, deadlocks, memory management, file and I/O subsystems and protection)
- Introduce the issues to be considered in the design and development of operating system.
- Introduce basic Unix commands, system call interface for process management, inter-process communication and I/O in Unix.

Course Outcomes:

- Ability to control access to a computer and the files that may be shared.
- Demonstrate the knowledge of the components of computers and their respective roles in computing.
- Ability to recognize and resolve user problems with standard operating environments.
- Gain practical knowledge of how programming languages, operating systems, and architectures interact and how to use each effectively.

UNIT – I

Introduction - Simple Batch, Multi-programmed, Time-shared, Personal Computer, Parallel, Distributed Systems, Real-Time Systems, System components, Operating System services, System Calls.

Process and CPU Scheduling- Process concepts and scheduling, Operations on processes, Cooperating Processes, Threads.

Scheduling Criteria, Scheduling Algorithms, Multiple -Processor Scheduling. System call interface for process management-fork(), exit(), wait(), waitpid() and exec().

UNIT – II

Deadlocks: System Model, Deadlocks Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, and Recovery from Deadlock.

Process Management and Synchronization: The Critical Section Problem, Synchronization Hardware, Semaphores, and Classical Problems of Synchronization, Critical Regions, Monitors.

UNIT – III

Interprocess Communication Mechanisms: IPC between processes on a single computer system, IPC between processes on different systems, using pipes, FIFOs, message queues, shared memory.

Memory Management and Virtual Memory: Logical versus Physical Address Space, Swapping, Contiguous Allocation, Paging, Segmentation, Segmentation with Paging, Demand Paging, Page Replacement, Page Replacement Algorithms.

UNIT – IV

File System Interface and Operations: Access methods, Directory Structure, Protection, File System Structure, Allocation methods, Free-space Management. Usage of open(), create(), read(), write(), close(), lseek(), stat(), ioctl() system calls.

TEXT & REFERENCES:

1. Operating System Principles- Abraham Silberchatz, Peter B. Galvin, Greg Gagne, John Wiley.
2. Operating Systems- Internals and Design Principles, William Stallings, Pearson Education.
3. Achyut S. Godbole , Atul Kahate; "Operating Systems"; 3rd Edition, McGraw Hill.
4. Advanced programming in the UNIX environment, W.R. Stevens, Pearson education.

List of Lab Assignments:

11. Write C programs to simulate the following CPU Scheduling algorithms a) FCFS b) SJF c) Round Robin d) priority.
12. Write programs using the I/O system calls of the UNIX/LINUX operating system (open, read, write, close, fcntl, seek, stat, opendir, readdir).
13. Write a C program to simulate Bankers Algorithm for Deadlock Avoidance and Prevention.
14. Write a C program to implement the Producer – Consumer problem using semaphores using UNIX/LINUX system calls.
15. Write C programs to illustrate the following IPC mechanisms a) Pipes b) FIFOs c) Message Queues d) Shared Memory.
16. Write C programs to simulate the following memory management techniques a) Paging b) Segmentation.
17. Write C programs to simulate Page replacement policies a) FCFS b) LRU c) Optimal.

Course Type: - Major
Semester: - 5th
Paper Title: - **DESIGN AND ANALYSIS OF ALGORITHMS**
Paper Code: - **CAPC3522M**
Credit Weightage: - THEORY -04; TUTORIALS- 02
Batch: - 2023

Course Objective:

- To introduce the notations for analysis of the performance of algorithms.
- Describes how to evaluate and compare different algorithms using worst-, average-, and best case analysis.
- Explains the difference between tractable and intractable problems, and introduces the problems that are P, NP and NP complete.

Course Outcomes:

- Ability to analyse the performance of algorithms.
- Ability to choose appropriate data structures and algorithm design methods for a specified application.
- Ability to understand how the choice of data structures and the algorithm design methods impact the performance of programs.

UNIT-I

Introduction to Algorithms: Characteristics of Algorithms, Asymptotic analysis of complexity Bounds - Best, Average and Worst-case behaviour, Time and space trade-offs, Analysis of algorithm: Analysis of iterative and recursive algorithms. Solutions of recurrence relations using back substitution method. Masters' theorem.

Unit-II

Divide and Conquer methods: Overview of Divide and Conquer strategy, Binary search and its analysis,selection sort, Merge sort, Quick sort, Strassen's matrix multiplication.

Greedy Algorithm: Overview of Greedy Paradigm, Fractional knapsack problem. Minimum cost spanning tree-Prim's and Kruskal's algorithm, Single source shortest path algorithm.

Unit-III

Dynamic Programming: Matrix Chain Multiplication, Solution to 0-1 Knapsack Problem and TSP using Dynamic Programming, Floyd-Warshall Algorithm.

Back-Tracking: Backtracking Algorithms for Enumerating Independent Sets of a Graph, Graph Coloring Problem and N-Queen's Problem, Complexity Classes.

Branch & Bound: Concept of Branch and Bound, Job scheduling problem, I/O Knapsack Problem, Traveling Salesperson Problem.

Unit-IV

Lower Bound Theory. Comparison Trees for searching and sorting, Parallel Comparison Trees, Lower bounds through reduction.

NP- Hard and NP- Complete Problems: Basic concepts, Approximation Algorithm for Vertex Cover Problem, Randomized Min-Cut Algorithm, Introduction to Network Flows, Max-Flow Min-Cut Theorem, Boyer-Moore String Matching Algorithm, Knuth-Morris-Pratt Algorithm for Pattern Matching and Amortized Analysis, Cook's theorem.

TEXT & REFERENCES:

1. Fundamentals of Computer Algorithms, Ellis Horowitz, SatrajSahni and Rajasekharan, Galgotia Press.
2. Data Structures and Algorithms Made Easy, Narasimha Karumanchi, Career Monk.
3. Design and Analysis of algorithms, Aho, Ullman and Hopcroft, Pearson education.
4. NPTEL Design and Analysis of algorithms Course : @https://www.youtube.com/watch?v=u5AXxR4GnRY
5. YouTube - Algorithms by Abdul Bari: @ https://www.youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O/

TUTORIALS -DESIGN AND ANALYSIS OF ALGORITHMS (CAPC2522M)

LIST OF TUTORIALS:

1. Solve recurrence relation based on substitution method and masters theorem.
2. Write a program for Selection sort, merge sort and quick sort.
3. Solve different examples on Single source shortest path algorithm, Fractional Knapsack
4. Write a program on Floyd-warshall algorithm.
5. Write a program on TSP, Matrix chain multiplication.
6. Write a program on Graph colouring problem.

Course Type: - Minor
Paper Title: - **WEB DEVELOPMENT WITH FRAMEWORKS**
Credit Weightage: - THEORY -04; PRACTICALS- 02

Semester: - 5th
Paper Code: - ACPC1522N
Batch: - 2023

Course Objective:

- To equip students with practical web development skills, proficiency in the MERN stack, and an understanding of modern web technologies, fostering their ability to build real-world web applications.

Course Outcomes:

- Understand the fundamental concepts of full-stack web development.
- Demonstrate proficiency in using MongoDB, Express.js, React, and Node.js to build web applications.
- Develop RESTful APIs and handle client-server interactions.
- Implement user authentication and authorization in web applications.
- Utilize React to create interactive and dynamic user interfaces.
- Deploy and host MERN applications on popular platforms.

UNIT – I

Introduction to Web Development and MERN Stack: Overview of web technologies and the MERN stack, Setting up the development environment for MERN development, Basic folder structure and project organization. Front-End Development with React: Introduction to React and its core concepts (components, state, props), JSX syntax and component rendering, Handling user interactions and events in React.

UNIT – II

Back-End Development with Node.js and Express.js: Introduction to Node.js and its event-driven architecture, setting up a Node.js server with Express.js, Implementing RESTful API endpoints with Express.js. MongoDB and Mongoose: Introduction to MongoDB and NoSQL databases, setting up MongoDB and Mongoose ODM, Performing CRUD operations with Mongoose.

UNIT – III

Building RESTful APIs: Designing RESTful API routes and endpoints, Handling HTTP methods (GET, POST, PUT, DELETE), Validating and handling API requests with middleware. User Authentication and Authorization: Impiementing user registration and login functionality, Using JSON Web Tokens (JWT) for user authentication, Securing API routes and managing user sessions.

UNIT – IV

Front-End and Back-End Integration: Connecting the front-end React application with the back-end Express.js API, Managing data flow between React components and Express.js routes.

Front-End Styling and UI Frameworks: Styling web applications using CSS and CSS frameworks (e.g., Bootstrap), Creating responsive and visually appealing user interfaces.

TEXT & REFERENCES:

1) Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node" by Vasan Subramanian
2) Learning React: Functional Web Development with React and Redux" by Alex Banks and Eve Porcello
3) The Road to learn React: Your journey to master plain yet pragmatic React.js" by Robin Wieruch
4) Express in Action: Writing, building, and testing Node.js applications" by Evan Hahn
5) MongoDB: The Definitive Guide" by Kristina Chodorow and Michael Dirolf.
6) FreeCodeCamp: MERN Stack Course - https://www.freecodecamp.org/news/mern-stack-complete-tutorial/

7) MDN Web Docs (Web Development Tutorials): https://developer.mozilla.org/en-US/docs/Web/Tutorials
8) Coursera: https://www.coursera.org/specializations/full-stack-react
9) Traversy Media: https://www.youtube.com/user/TechGuyWeb
5. The Net Ninja: https://www.youtube.com/c/TheNetNinja
6.

LAB. -WEB DEVELOPMENT WITH FRAMEWORKS (ACPC1522N)

LIST OF LAB. ASSIGNMENTS :

1. Setting Up the Development Environment
   - Install Node.js and npm.
   - Set up a new project using create-react-app.
   - Create a simple React component and render it on a web page.
2. Front-End Development with React
   - Create a form with input fields for user registration.
   - Implement form validation for user input.
   - Build a React component to display a list of items fetched from an API.
   - Add buttons and functionality to interact with the list (e.g., delete, edit items).
3. Back-End Development with Node.js and Express.js
   - Set up a basic Express.js server.
   - Implement a route to handle HTTP GET requests and return sample data as JSON.
   - Create a route to handle HTTP POST requests and store data in a JSON file.
4. Database Integration with MongoDB and Mongoose
   - Set up a MongoDB database locally or use MongoDB Atlas for cloud-based hosting.
   - Define a Mongoose schema for a data entity (e.g., users, products).
5. Create routes to perform CRUD operations on the database (e.g., create, read, update, delete).
   - Building RESTful APIs with Express.js
   - Design and implement API routes for user registration and login functionality.
   - Add authentication middleware to secure certain API endpoints.
   - Create an API route to fetch data from the MongoDB database and return it as JSON.
6. User Authentication and Authorization
   - Set up user registration and login routes with JWT token generation and verification.
   - Build middleware to protect certain routes, allowing only authenticated users access.
   - Test the authentication and authorization flow with sample user credentials.
7. Deployment and Hosting
   - Prepare the MERN application for deployment by optimizing code and dependencies.
   - Choose a hosting platform and deploy the application.
   - Verify that the deployed application is accessible and functions correctly.
8. Final Project
   - Define the scope and requirements of the final project.
   - Design the user interface and application flow.
   - Implement front-end and back-end functionalities to meet the project requirements.
   - Test the application thoroughly and deploy it to a live server.

Course Type: - Internship
Paper Title: - **Internship / Minor Project**
Credit Weightage: - THEORY -0; PRACTICALS- 04

Semester: - 5th
Paper Code: - INTCS0005
Batch: - 2023

Course Type: - Major
Paper Title: - CRYPTOGRAPHY AND NETWORK SECURITY
Credit Weightage: - THEORY -04; PRACTICALS- 02

Semester: - 6th
Paper Code: -CAPC1622M
Batch: - 2023

Course Objective:

- Explain the objectives of information security.
- Explain the importance and application of each of confidentiality, integrity, authentication and availability.
- Understand various cryptographic algorithms and basic categories of threats to computers and networks.
- Describe public-key cryptosystem and enhancements made to IPv4 by IPSec.
- Understand Intrusions and intrusion detection.
- Discuss the fundamental ideas of public-key cryptography.
- Discuss Web security and Firewalls.

Course Outcomes:

- Student will be able to understand basic cryptographic algorithms, message and web authentication and security issues.
- Ability to identify information system requirements for both of them such as client and server.
- Ability to understand the current legal issues towards information security.

UNIT – I

SECURITY CONCEPTS: Introduction, The need for security, Security approaches, Principles of security, Types of Security attacks, Security services, Security Mechanisms, A model for Network Security.
Cryptography Concepts and Techniques: Introduction, plain text and cipher text, substitution techniques, transposition techniques, encryption and decryption, symmetric and asymmetric key cryptography, steganography, key range and key size, possible types of attacks.

UNIT – II

Symmetric key Ciphers: Block Cipher principles, DES, AES, Blowfish, RC5, IDEA, Block cipher operation, Stream ciphers, RC4.
Asymmetric key Ciphers: Principles of public key cryptosystems, RSA algorithm, Elgamal Cryptography, Diffie-Hellman Key Exchange, Knapsack Algorithm.

UNIT – III

Cryptographic Hash Functions: Message Authentication, Secure Hash Algorithm (SHA-512), Message authentication codes: Authentication requirements, HMAC, CMAC, Digital signatures, Elgamal Digital Signature Scheme.
Key Management and Distribution: Symmetric Key Distribution Using Symmetric & Asymmetric Encryption, Distribution of Public Keys, Kerberos.

UNIT – IV

Transport-level Security: Web security considerations, Secure Socket Layer and Transport Layer Security, HTTPS, Secure Shell (SSH).
Wireless Network Security: Wireless Security, Mobile Device Security, IEEE 802.11 Wireless LAN, IEEE 802.11i Wireless LAN Security.

TEXT &REFERENCES:

1) Cryptography and Network Security - Principles and Practice: William Stallings, Pearson Education, 6th Edition.
2) Cryptography and Network Security: Atul Kahate, Mc Graw Hill, 3rd Edition.
1. C. P. Pfleeger, S. L. Pfleeger; Security in Computing, Prentice Hall of India, 2006.

TUTORIALS -CRYPTOGRAPHY AND NETWORK SECURITY (CAPC1622M)

LIST OF EXPERIMENTS:

1. Write a program that contains a string (char pointer) with a value 'Hello world'. The program should XOR each character in this string with 0 and displays the result.
2. Write a program that contains a string (char pointer) with a value 'Hello world'. The program should AND or and XOR each character in this string with 127 and display the result.
3. Write a program to perform encryption and decryption using the following algorithms
   a)Ceaser cipher **b)**. Substitution cipher **c)**. Hill Cipher
4. Write a program to implement the DES algorithm logic.
5. Write a program to implement the Blowfish algorithm logic.
6. Write a program to implement the Rijndael algorithm logic.
7. Write the RC4 logic in Java Using Java cryptography; encrypt the text "Hello world" using Blowfish. Create your own key using Java key tool.
8. Write a program to implement RSA algorithm.
9. Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript.
10. Write a program to calculate the message digest of a text using the SHA-1 algorithm.
11. Write a program to calculate the message digest of a text using the MD5 algorithm.
12. Demonstrate the use of Network tools: ping, ipconfig, ifconfig, tracert, arp, netstat, whois.

Course Type: - Major
Paper Title: ARTIFICIAL INTELLIGENCE
Credit Weightage: - THEORY -04; PRACTICALS- 02

Semester: - 6th
Paper Code: - CAPC2622M
Batch: - 2023

Course Objective:

- To learn the distinction between optimal reasoning vs. human-like reasoning.
- To understand the concepts of state space representation, exhaustive search, heuristic search together with the time and space complexities.
- To know about the various knowledge representation methods

Course Outcomes:

- Ability to formulate an efficient problem space for a problem expressed in natural language.
- Select a search algorithm for a problem and estimate its time and space complexities.
- Understand knowledge representation methods and apply approximate reasoning.

UNIT – I

INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND PROBLEM-SOLVING AGENT: Problems of AI, AI technique, Tic– Tac – Toe problem. Intelligent Agents, Agents & environment, nature of environment, structure of agents, goal-based agents, utility-based agents, learning agents. Defining the problem as state space search, production system, problem characteristics, and issues in the design of search programs. Problem solving agents, searching for solutions; uniform search strategies: breadth first search, depth first search, depth limited search, bidirectional search, comparing uniform search strategies.

UNIT – II

SEARCH TECHNIQUES AND CONSTRAINT SATISFACTION PROBLEMS: Heuristic search strategies Greedy best -first search, A* search, AO* search, memory bounded heuristic search: local search algorithms & optimization problems: Hill climbing search, simulated annealing search, local beam search.

Local search for constraint satisfaction problems. Adversarial search, Games, optimal decisions & strategies in games, the minimax search procedure, alpha-beta pruning, additional refinements, iterative deepening.

UNIT-III

KNOWLEDGE REPRESENTATION:predicate logic- logic programming, semantic nets- frames and inheritance, constraint propagation, representing knowledge using rules, rules-based deduction systems. Reasoning under uncertainty, review of probability, Baye's probabilistic interferences and dempster-Shafer theory.

UNIT-IV

EXPERT SYSTEMS: - Introduction, basic concepts, structure of expert systems, the human element in expert systems how expert systems works, problem areas addressed by expert systems, expert systems success factors, types of expert systems, expert systems and the internet interacts web, knowledge engineering, scope of knowledge, difficulties in knowledge acquisition, methods of knowledge acquisition. Typical expert systems - MYCIN, DART, XOON, Expert systems shells.

TEXT & REFERENCES:

1) Artificial Intelligence A Modern Approach, Stuart Russell and Peter Norvig, 3rd Edition, Pearson Education.
2) Artificial Intelligence, 3rdEdn. ,E.Rich and K.Knight (TMH).
3) Tom M. Mitchell- Machine Learning - McGraw Hill Education.
4) Christopher M. Bishop Pattern Recognition and Machine Learning - Springer, 2nd edition
5) Trevor Hastie, Robert Tibshirani, and Jerome Friedman - The Elements of Statistical Learning: Data Mining, Inference, and Prediction - Springer, 2nd edition.
6) Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong, Mathematics for Machine Learning, Cambridge University Press.

Course Type: - Major
Paper Title: - **THEORY OF COMPUTATION**
Credit Weightage: - THEORY -04; PRACTICALS- 02

Semester: - 6th
Paper Code: - **CAPC3622M**
Batch: - 2023

Course Objective:

- To provide introduction to some of the central ideas of theoretical computer science from the perspective of formal languages.
- To introduce the fundamental concepts of formal languages, grammars and automata theory.
- Classify machines by their power to recognize languages.
- To understand deterministic and non-deterministic machines.
- Introduce the major concepts of language translation and compiler design and impart the knowledge of practical skills necessary for constructing a compiler.
- Topics include phases of compiler, parsing, code optimization techniques, intermediate code generation, code generation.

Course Outcomes:

- Able to understand the concept of abstract machines and their power to recognize the languages.
- Able to employ finite state machines for modelling and solving computing problems.
- Able to design context free grammars for formal languages.
- Demonstrate the ability to design a compiler given a set of language features.
- Demonstrate the knowledge of patterns, tokens & regular expressions for lexical analysis.
- Design algorithms to do code optimization in order to improve the performance of a program in terms of space and time complexity.
- Design algorithms to generate machine code.

UNIT – I

INTRODUCTION TO FINITE AUTOMATA: Structural Representations, Automata and Complexity, the Central Concepts of Automata Theory – Alphabets, Strings, Languages, Problems.

Nondeterministic Finite Automata: Formal Definition, an application, Text Search, Finite Automata with Epsilon-Transitions.

Regular Expressions: Finite Automata and Regular Expressions, Applications of Regular Expressions, Algebraic Laws for Regular Expressions, Conversion of Finite Automata to Regular Expressions, Equivalence of DFA, NFA and REs.

UNIT – II

CONTEXT FREE GRAMMARS AND PARSING: Definition of Context-Free Grammars, Derivations Using a Grammar, Leftmost and Rightmost Derivations, the Language of a Grammar, Sentential Forms, Parse Tress, Applications of Context-Free Grammars, Ambiguity in Grammars and Languages. Chomsky Normal Form.

UNIT – III

PUSHDOWN AUTOMATA: Definition of the Pushdown Automaton, the Languages of a PDA, Equivalence of PDA's and CFG's, Acceptance by final state, Acceptance by empty stack, Deterministic Pushdown Automata. From CFG to PDA, Equivalence of PDA and CFG.

UNIT – IV

TURING MACHINES: Introduction to Turing Machine, Formal Description, Instantaneous description, The language of a Turing machine.

RECURSIVE AND RECURSIVELY ENUMERABLE LANGUAGES (REL): Properties of recursive and recursively enumerable languages, Universal Turing machine, The Halting problem, Undecidable problems about TMs. Context sensitive language and linear bounded automata (LBA), Chomsky hierarchy

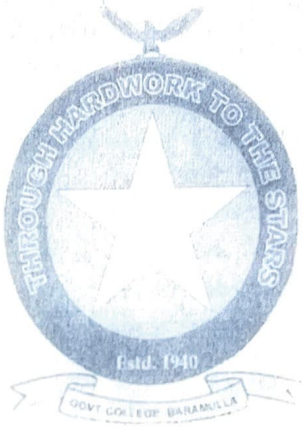
TEXT & REFERENCES:

1) John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman (2007), Introduction to Automata Theory Languages and Computation, Pearson.
2) Introduction to Computer Theory, Daniel I.A. Cohen, Wiley India
3) K. L. P Mishra, N. Chandrashekaran (2003), Theory of Computer Science-Automata Languages and Computation, 2nd edition, Prentice Hall of India, India.
4) P. Linz, An Introduction to Formal Language and Automata 4th edition Publication Jones Bartlett.
5) NPTEL Course on Theory of Computation @ https://nptel.ac.in/courses/106104148
6) YouTube TOC- @ https://www.youtube.com/playlist?list=PLBInK6fEyqRgp46KUv4ZY69yXmpwKOIev

TUTORIALS -THEORY OF COMPUTATION (CAPC3822M)

LIST OF TUTORIALS:

1. .....



Course Type: - Minor
Paper Title: - **PROGRAMMING WITH PYTHON**
Credit Weightage: - THEORY -04; PRACTICALS- 02

Semester: - 6th
Paper Code: - **ACPC1622N**
Batch: - 2023

Course Objective:

- To introduce students to Python programming and its applications in data science and machine learning.
- To provide a comprehensive understanding of essential Python concepts, data structures, and control flow.
- To equip students with the skills to work with popular Python libraries for data manipulation, analysis, and visualization.
- To enable students to apply Python programming and data science knowledge to real-world projects and problem-solving.

Course Outcomes:

- Demonstrate proficiency in Python programming, including variables, data types, functions, and control flow structures.
- Utilize NumPy to perform numerical computing and array operations efficiently.
- Understand the basics of machine learning and implement various ML models using Scikit-learn.
- Apply Python and data science skills to conduct exploratory data analysis and draw meaningful conclusions from datasets.

UNIT – I

Introduction to Python Programming: Introduction to Python and its features, Installing Python and development environments (Jupiter Notebook). Basic Python syntax: data types, Type conversion and casting, variables and constants , arithmetic operators, comparison operators, logical operators. Combining operators to form complex expressions. Control statements: if-else, loops (while, for), conditional expressions (ternary operator) for concise code. Functions and Modules: Defining functions in Python, Function parameters, return values, and recursion. Introduction to modules and the standard library, Importing and using modules in Python programs. Working with strings.

UNIT – II

PYTHON DATA STRUCTURES AND FILE HANDLING: Lists: Creating, accessing, and modifying lists. writing concise and efficient list comprehensions for creating lists based on existing data. Tuples, Sets and Dictionaries: Understanding unique elements and set operations (union, intersection, difference). Dictionaries: Associating key-value pairs for efficient data retrieval. Reading and writing binary files for textual & non-textual data (e.g., images, audio). File Handling in Python. Exception handling: try, except, finally blocks. Introduction to NumPy arrays for Efficient Numerical Computing: Array Creation and Manipulation, single and Multi-dimensional Arrays, Element-wise Operations, Mathematical Functions.

UNIT – III

Data Manipulation with Pandas: Introduction to Pandas library for data manipulation and analysis, Series and Data Frame objects for handling data, Data indexing, selection, and filtering, Data cleaning, handling missing values, and data transformation. Data Visualization with Matplotlib and Seaborn: Introduction to data visualization, Creating basic plots with Matplotlib, Enhancing visualizations with Seaborn, Customizing plots for effective data representation.

UNIT – IV

Exploratory Data Analysis (EDA): Understanding the importance of EDA in data science, Statistical summaries and data distributions, Univariate and bivariate analysis using visualizations, Correlation and feature relationships. Machine Learning in Python: Overview of Scikit-learn library, Data pre-processing and feature scaling, Building and training machine learning models, Evaluating and tuning model performance.

TEXT & REFERENCES:

1. Python Crash Course, Eric Matthes, No Starch Press.
2. Learning Python,Mark Lutz, O'Reilly.
3. How to Think Like a Computer Scientist: Learning with Python, Allen Downey, Jeff Elkner and Chris Meyers, SoHo Books, 2009.
4. Introduction to Machine Learning with Python, Andreas C. Müller & Sarah Guido, O'Reilly.
5. Python for Data Analysis, Wes McKinney, O'Reilly.

LAB. -PROGRAMMING WITH PYTHON (ACPC1622N)

LIST OF LAB. ASSIGNMENTS :

1. Write a Python program to calculate the area of a rectangle given its length and width.
2. Implement a function that takes a list of numbers as input and returns the sum of all the elements in the list.
3. Create a program that generates a random number between 1 and 100 and lets the user guess it. Provide appropriate feedback based on their guess.
4. Write a Python program to check if a given number is even or odd.
5. Create a program that prints the Fibonacci series up to a given number using a loop.
6. Write a Python function that takes a string as input and returns the reverse of the string.
7. Create a program that counts the number of vowels and consonants in a given sentence.
8. Implement a function that takes a sentence and returns the sentence with each word capitalized.
9. Write a Python program that takes a list of numbers as input and finds the maximum and minimum values.
10. Write a Python program to read a text file and count the number of words in it.
11. Create a program that reads a CSV file and calculates the average of a specific column.
12. Implement a function that takes a list of names and writes them to a new file, one name per line.
13. Create a module with a function to check if a number is prime or not.
14. Implement a program that imports the module and uses the prime-checking function to find all prime numbers in a given range.
15. Write a Python program that takes user input and handles exceptions for invalid input.
16. Implement a function that reads a number from a file and handles file-not-found exceptions.
17. Create a program that asks the user to enter two numbers and divide them, handling division by zero error.
18. Create a NumPy array with random integers and perform basic array operations like reshaping, flattening, and transposing.
19. Use Pandas to filter and slice data based on specific conditions.
20. Apply data cleaning techniques like handling missing values, removing duplicates, and transforming data.
21. Create line plots, scatter plots, and bar plots to visualize various datasets using Matplotlib.
22. Use Seaborn to create visually appealing statistical visualizations like box plots, violin plots, and pair plots.
23. Plot time series data with appropriate annotations and labels.
24. Perform EDA on a real-world dataset to understand data distributions, correlations, and feature relationships.
25. Use Matplotlib and Seaborn to create various visualizations that aid in data exploration.