P.G Department of Computer Applications

Govt. Degree College (Autonomous), Baramulla

COURSE STRUCTURE AND SYLLABUS

For

BACHELORS (HONS) IN COMPUTER APPLICATIONS (3+1)
Applicable for Batches 2022, 2023 and 2024 onwards

Under National Education Policy (NEP-2020)

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

Course Type: - Major **Semester**: - 5th
Paper Title: - SOFTWARE ENGINEERING **Paper Code: - CAPC1522M**
Credit Weightage: - THEORY -04; TUTORIALS- 02 Batch: - 2022 on words

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, user requirements, 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	**Semester**: - 5th
Paper Title: - **OPERATING SYSTEMS**	**Paper Code**: - A**CAPC1522N**
Credit Weightage: - THEORY -04; PRACTICALS- 02	**Batch**: - 2022 and on words

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.

TUTORIALS - OPERATING SYSTEMS (CAPC2522M)

List of Lab Assignments:

1. Write C programs to simulate the following CPU Scheduling algorithms a) FCFS b) SJF c) Round Robin d) priority.
2. Write programs using the I/O system calls of the UNIX/LINUX operating system (open, read, write, close, fcntl, seek, stat, opendir, readdir).
3. Write a C program to simulate Bankers Algorithm for Deadlock Avoidance and Prevention.
4. Write a C program to implement the Producer – Consumer problem using semaphores using UNIX/LINUX system calls.
5. Write C programs to illustrate the following IPC mechanisms a) Pipes b) FIFOs c) Message Queues d) Shared Memory.
6. Write C programs to simulate the following memory management techniques a) Paging b) Segmentation.
7. 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**: - 2022 and on words

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, Satraj Sahni 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. ……

Course Type: - Minor **Semester**: - 5th

Paper Title: - **WEB DEVELOPMENT WITH FRAMEWORKS** **Paper Code**: - ACPC1522N

Credit Weightage: - THEORY -04; PRACTICALS- 02 **Batch**: - 2022 and on words

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: Implementing 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	**Semester**: - 5th
Paper Title: - **Internship / Minor Project**	**Paper Code**: - INTCS0005
Credit Weightage: - THEORY -0; PRACTICALS- 04	**Batch**: - 2022 and on words