Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

Structure & Syllabus

Double Minor Courses

Offered by

B.Tech. (Computer Engineering)

With Effect from Academic Year 2025-26

Prepared by: - Board of Studies in Computer Engineering

Approved by: - Academic Board, Vishwakarma Institute of Technology, Pune

Chairman – BOS **Chairman – Academic Board**

Vision of the Institution

"To be globally acclaimed Institute in Technical Education and Research for holistic Socio-economic development".

Mission of the Institution

- To ensure that 100% students are employable and employed in Industry, Higher Studies, become Entrepreneurs, Civil / Defense Services / Govt. Jobs and other areas like Sports and Theatre.
- To strengthen Academic Practices in terms of Curriculum, Pedagogy, Assessment and Faculty Competence.
- Promote Research Culture among Students and Faculty through Projects and Consultancy.
- To make students Socially Responsible Citizen.

Vision of the Department

"To be a leader in the world of computing education practising creativity and innovation".

Mission of the Department

- To ensure students' employability by developing aptitude, computing, soft, and entrepreneurial skills
- To enhance academic excellence through effective curriculum blended learning and comprehensive assessment with active participation of industry
- To cultivate research culture resulting in knowledge-base, quality publications, innovative products and patents
- To develop ethical consciousness among students for social and professional maturity to become responsible citizens

Knowledge and Attitude Profile (WK)

WK	WK Statements
WK1	A systematic, theory-based understanding of the natural sciences applicable to the discipline and awareness of relevant social sciences.
WK2	Conceptually-based mathematics, numerical analysis, data analysis, statistics and formal aspects of computer and information science to support detailed analysis and modelling applicable to the discipline.
WK3	A systematic, theory-based formulation of engineering fundamentals required in the engineering discipline.
WK4	Engineering specialist knowledge that provides theoretical frameworks and bodies of knowledge for the accepted practice areas in the engineering discipline; much is at the forefront of the discipline.
WK5	Knowledge, including efficient resource use, environmental impacts, whole-life cost, reuse of resources, net zero carbon, and similar concepts, that supports engineering design and operations in a practice area.
WK6	Knowledge of engineering practice (technology) in the practice areas in the engineering discipline.
WK7	Knowledge of the role of engineering in society and identified issues in engineering practice in the discipline, such as the professional responsibility of an engineer to public safety and sustainable development.
WK8	Engagement with selected knowledge in the current research literature of the discipline, awareness of the power of critical thinking and creative approaches to evaluate emerging issues.
WK9	Ethics, inclusive behavior and conduct. Knowledge of professional ethics, responsibilities, and norms of engineering practice. Awareness of the need for diversity by reason of ethnicity, gender, age, physical ability etc. with mutual understanding and respect, and of inclusive attitudes.

List of Programme Outcomes [PO]

PO	PO Statements
PO1	**Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified inWK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
PO3	**Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
PO4	**Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO5	**Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6 **The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7 **Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8 **Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9 **Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10 **Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11 **Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

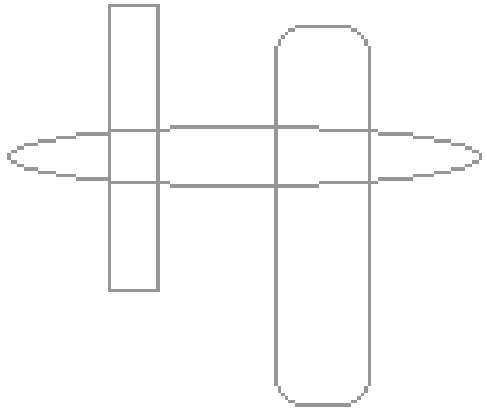
List of Programme Specific Outcomes [PSO]

List of PSO Statements

PSO 1 Demonstrate proficiency in programming with a sound understanding of fundamental computing principles

PSO 2 Conceive well-structured design and proficient implementation to address real world challenges using software paradigms, algorithms and technologies

PSO 3 Acquire and showcase expertise in emerging fields within computer science, engineering, and technology



Program Educational Objectives (PEOs)

- Demonstrate application of sound engineering foundations to be a committed technology workforce
- Apply mathematical and computing theory knowledge base to provide realistic computer engineering solutions
- Exhibit problem-solving skills and engineering practices to address problems faced by the industry with innovative methods, tools, and techniques
- Develop professional and ethical practices adopting effective guidelines to acquire desired soft skills in the societal and global context
- Aim for continuing education and entrepreneurship in emerging areas of computing

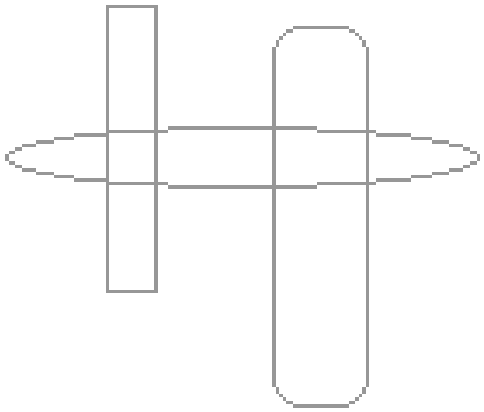
Course Name Nomenclature as per NEP (For FY and SY)

BSC: Basic Science Course	MDOE: Multi Disciplinary Open Elective
ESC: Engineering Science Course	CC: Co-curricular Course
PCC: Program Core Course	HSSM: Humanities Social Science and Management
PEC: Program Elective Course	IKS: Indian Knowledge System
ELC: Experiential Learning Course	FP: Field Project
MD: Multi Disciplinary	INT: Internship

Nomenclature for Teaching and Examination Assessment Scheme AY 2024-25

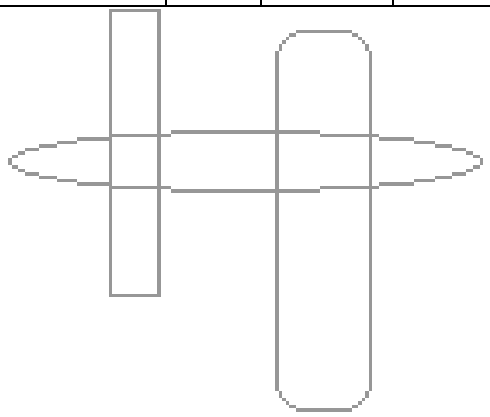
Sr No.	Category	Head of Teaching/ Assessment	Abbreviation used
1	Teaching	Theory	Th
2	Teaching	Laboratory	Lab
3	Teaching	Tutorial	Tut
4	Teaching	Open Elective	OE
5	Teaching	Multi Disciplinary	MD
6	Teaching	Computer Science	CS
7	Assessment	Laboratory Continuous Assessment	CA
8	Assessment	Mid Semester Assessment	MSA
9	Assessment	End Semester Assessment	ESE
10	Assessment	Home Assignment	HA
11	Assessment	Course Project	CP
12	Assessment	Group Discussion	GD
13	Assessment	PowerPoint Presentation	PPT
14	Assessment	Class Test –1	CT1
15	Assessment	Class Test –2	CT2
16	Assessment	Mid Semester Examination	MSE
17	Assessment	End Semester Examination	ESE
18	Assessment	Written Examination	WRT
19	Assessment	Multiple Choice Questions	MCQ
20	Assessment	Laboratory	LAB

Double Minor Courses



Double Minor Course: Full Stack Development

Course No.	Semester	Course Name	Teaching Scheme (Hours/Week)			Examination Scheme				Credits
			Th	Lab	Tut				Total	
C1	III	Java Script Programming	2	2	0					3
C2	IV	React and React Native	2	2	0					3
C3	V	Node.js	2	2	0					3
C4	VI	Mongo DB	2	2	0					3
C5	VII	Express.js	2	2	1					4
C6	VIII	Project	0	8	0					4



Double Minor in Full Stack Development

ABCXXX: JavaScript Programming

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2/Week
Total Credits: 3

Course Objectives:
1. Master Core JavaScript Syntax
2. Develop Functions and Implement Modular Code
3. Manipulate the DOM and Handle Events
4. Implement Object-Oriented Programming (OOP) Principles
5. Handle Errors and Debug JavaScript Code
6. Understand Asynchronous Programming

Course Relevance: Learning the MERN stack enables you to build full-stack web applications using a single language—JavaScript—across frontend and backend. It offers a powerful, flexible, and efficient way to create scalable, modern apps with React's dynamic UI, Express and Node's robust backend, and MongoDB's flexible database. MERN is widely used in the industry, increasing job opportunities. Its open-source nature and strong community support make development faster and easier. Overall, MERN equips you with in-demand skills to develop complete web solutions end-to-end.

Syllabus
Theory

Unit 1: Introduction to JavaScript (6 Hours)

History and evolution of JavaScript; its role in web development, Set up a local development environment using code editors like Visual Studio Code and utilize browser developer tools for debugging.Understanding Variables, Data Types, and Operators Understand how to declare variables using var, let, and const, and learn about variable scope and hoisting.Dive into JavaScript's data types, including primitive types like number, string, boolean, null, undefined, and symbol, as well as complex types like object and array.Master the use of arithmetic, assignment, comparison, and logical operators, and understand type conversion and coercion.

Unit 2: Control Flow and Functions (4 Hours)

Learn how to use if, else, else if, and switch statements to control the flow of your program based on conditions. Implement for, while, and do-while loops to repeat code execution, and understand the use of break and continue statements. Define and invoke functions, understand function parameters and return values, and explore function expressions and arrow functions.

Unit 3: Arrays and Objects (5 Hours)

Create and manipulate arrays, utilize array methods like push, pop, shift, unshift, slice, and splice, and iterate over arrays using loops and array methods. Understand how to create and access objects, work with object properties and methods, and explore object constructors and prototypes.

Unit 4: The Document Object Model (DOM) Object Oriented Programming (4 Hours)

Learn what the DOM is and how JavaScript interacts with it to manipulate HTML and CSS.Select and modify DOM elements, create and remove elements dynamically, and handle events like clicks and keypresses. Classes and objects in JavaScript.

Unit 5: Asynchronous JavaScript and Error Handling in JavaScript (6 Hours)

Understand the concept of callbacks and how they are used to handle asynchronous operations.Learn how to create and handle promises to manage asynchronous code more effectively.Simplify asynchronous code using async/await syntax, and understand how it improves readability and error handling.Identify different types of errors in JavaScript, including syntax errors, runtime errors, and logical errors.Use try, catch, and finally blocks to handle errors gracefully and prevent program crashes

Unit 6: TypeScript (5 Hours)

Understanding TypeScript as a statically typed superset of JavaScript, Setting up the development environment and compiling TypeScript code, Exploring the benefits of TypeScript in large-scale application development.Working with primitive types: string, number, Boolean.Utilizing special types: any, void, never, undefined, null, Creating and manipulating arrays and tuples, Defining object types and Enums,Implementing type aliases and interfaces

Syllabus
Laboratory

List of Experiments

1. Text Manipulation Tools: Create applications that convert text to uppercase, count vowels, or validate usernames and passwords.

2. Date and Time Display: Develop a script that displays the current date and time, updating in real-time.
3. Form Validation: Implement client-side validation for forms, ensuring correct data entry before submission.
4. Interactive UI Elements: Build features like image sliders, countdown timers, and modal pop-ups to enhance user experience
5. **JSON Data Handling**: Develop scripts that convert lists to JSON format and handle JSON data effectively.
6. **Geometry Calculators**: Build tools that calculate areas of geometric shapes like rectangles, enhancing mathematical understanding.
7. TypeScript **Basic Type Usage**: Explore TypeScript's static typing by defining variables with specific types like string, number, and boolean.
8. TypeScript **Function Annotations**: Implement functions with defined parameter and return types to understand type safety.
9. TypeScript **Object Interfaces**: Define interfaces to describe the shape of objects, ensuring consistent data structures.
10. Generics: Reverse an Array using TypeScript
11. Define a class called Animal with a name property and a method using TypeScript
12. Reverse a String Write a function that takes a string as input and returns its reverse.

Course Outcomes

Course Outcomes: Student will be able to

1. Develop and run JavaScript programs to perform calculations, manipulate data, and automate tasks.
2. Create reusable code blocks with functions, including understanding scope, parameters, and return values.
3. Identify and resolve errors using debugging tools and techniques, ensuring code reliability and performance.
4. Work with asynchronous operations using callbacks, promises, and async/await to handle tasks like data fetching without blocking the main thread.
5. Implement objects and classes to model real-world entities, promoting code organization and reuse.

Books and E-Resources

For Reference Print Book -

Text Books: (As per IEEE format)

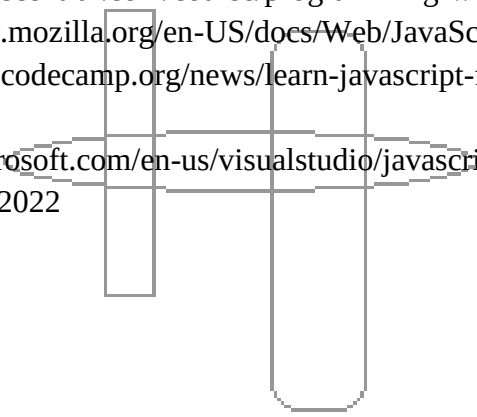
1. **Marijn Haverbeke**, Eloquent JavaScript: A Modern Introduction to Programming, 3rd ed., San Francisco, CA: No Starch Press, 2018.
2. **David Flanagan**, JavaScript: The Definitive Guide, 7th ed., Sebastopol, CA: O'Reilly Media, 2020.
3. **Boris Cherny**, Programming TypeScript: Making Your JavaScript Applications Scale, Sebastopol, CA: O'Reilly Media, 2019

Reference Books: (As per IEEE format)

1. **Douglas Crockford**, *JavaScript: The Good Parts*, Sebastopol, CA: O'Reilly Media, 2008.
2. **Kyle Simpson**, *You Don't Know JS (Yet)*, Sebastopol, CA: O'Reilly Media, 2020.
3. **Steve Fenton**, Pro TypeScript: Application-Scale JavaScript Development, 2nd ed., Berkeley, CA: Apress, 2016.
4. **Dan Vanderkam**, Effective TypeScript: 62 Specific Ways to Improve Your TypeScript, Sebastopol, CA: O'Reilly Media, 2019.

For MOOCs and other learning Resources

1. https://www.classcentral.com/course/programming-with-javascript-89876
2. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction
3. https://www.freecodecamp.org/news/learn-javascript-free-js-courses-for-beginners/
4. https://learn.microsoft.com/en-us/visualstudio/javascript/javascript-in-visual-studio?view=vs-2022



Double Minor in Full Stack Development

ABCXXX: React and React Native

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2 Hours/Week
Total Credits: 3

Syllabus

Course Objectives:

1. To Set Up and Configure a React Native Development Environment
2. To Understand Core React Native Components and JSX Syntax
3. To Implement Navigation and Routing in Mobile Applications
4. To Manage Application State with React Hooks and Context API
5. To Integrate APIs and Handle Asynchronous Operations
6. To Deploy and Maintain Cross-Platform Mobile Applications

Course Relevance: Learning the MERN stack enables you to build full-stack web applications using a single language—JavaScript—across frontend and backend. It offers a powerful, flexible, and efficient way to create scalable, modern apps with React's dynamic UI, Express and Node's robust backend, and MongoDB's flexible database. MERN is widely used in the industry, increasing job opportunities. Its open-source nature and strong community support make development faster and easier. Overall, MERN equips you with in-demand skills to develop complete web solutions end-to-end.

Syllabus
Theory

Unit 1: Foundations of React (5Hours)

Introduction to SPA (Single Page Applications),Setting up a React environment (Vite, Create React App, or Next.js basics), JSX syntax and best practices, Components: Functional vs Class (focus on Functional), Props and children, State management with useState, Event handling in React, Conditional rendering, Lists and key prop, Component folder structure & naming conventions

Unit 2: Intermediate Concepts and React Hooks (5 Hours)

useEffect for side effects, useRef and managing DOM references, Lifting state up & component composition, Controlled vs uncontrolled components (forms), Form handling and validation basics, Reusable components and prop drilling, React Context API, Basic custom hooks, Error boundaries (intro), Introduction to Redux.

Unit 3: Advanced Topics and Ecosystem Integration (5 Hours)

React Router v6: navigation, params, nested routes, Data fetching with fetch / axios, Using useEffect for API integration, Conditional rendering with async data, Optimizing performance: useMemo, useCallback, Code splitting and lazy loading, Introduction to testing React apps (Jest, React Testing Library), Basics of state libraries: Intro to deploying React apps (Vercel/Netlify)

Unit 4: **Introduction to React Native & Core Concepts (5 Hours)**

Overview of mobile app development: Native vs. Cross-platform, Introduction to React Native: Advantages and ecosystem, Setting up the development environment: NodeUnderstanding components: View, Text, Image, ScrollView, FlatList, JSX syntax and styling: Flexbox, StyleSheet, and platform-specific stylesHandling user input: TextInput, Button, Touchable components. Implementing navigation: Stack, Tab, and Drawer navigators using React Navigation, Passing data between screens and managing navigation state, Deep

Linking and nested navigation patterns, Local state management with useState and useReducer, Introduction to Context API for global state management, Hands-on project: Building a simple To-Do application.

Unit 5: Intermediate Development & API Integration (5 Hours)

Fetching data using fetch and axios, Handling asynchronous operations with async/await, Displaying data in lists and handling loading/error states, Introduction to Redux: Actions, Reducers, Store, Integrating Redux with React Native applications, Using Redux Toolkit for efficient state management, Persisting data with AsyncStorage, Introduction to SQLite and Realm for local databases, Implementing offline support and caching strategiesUsing the Animated API for basic animations (fade, slide, scale), Implementing gesture handling with React Native Gesture Handler, Creating interactive UI elements with animations.

Unit 6: Advanced Topics & Deployment (5 Hours)

Accessing device hardware: Camera, Location, Accelerometer, Handling permissions and privacy considerations, Creating and using custom native modules, Building APKs and IPAs for Android and iOS, Configuring app settings for production, Submitting applications to Google Play Store and Apple App Store.

Syllabus
Laboratory

List of Experiments

1. Create a simple React component that renders "Hello, World!" on the screen using React.
2. Build a counter app that increments and decrements a number using buttons, utilizing the useState hook using React.
3. Develop a TODO list app where users can add, delete, and mark tasks as completed. Implement state management and conditional rendering using React.
4. Create a user registration form with fields like name, email, and password. Implement basic validation and display error messages using React.
5. Use the fetch API to retrieve data from a public API and display it in a list format. Handle loading and error states appropriately using React.
6. Set up React Router in a project to navigate between different pages (e.g., Home, About, Contact) using React.
7. Render a simple "Hello, World!" message using React Native components like Text and View using react native.
8. Build a counter application with increment and decrement buttons, utilizing useState for state management using react native.
9. Develop a TODO list app where users can add, delete, and mark tasks as completed. Use FlatList for rendering the list efficiently using react native.
10. Create a form with TextInput components for user input. Implement validation and display error messages using Alert(Use react native)
11. Use the fetch API to retrieve data from a public API and display it in a list format using FlatList. Handle loading and error states appropriately using react native.
12. Implement navigation between different screens using React Navigation. Create a stack navigator with screens like Home and Details using react native

Course Outcomes

Course Outcomes: Student will able to

1. Demonstrate a solid understanding of Single Page Applications (SPA) and effectively set up development environments using tools like Vite, Create React App, or Next.js.
2. Mastery of React Hooks and Advanced State Management
3. Get Expertise in React Router and Data Integration
4. Understand of React Native Core Concepts
5. Understand Advanced React Native Development and API Integration
6. Know about Deployment and Publication of React Native Applications

Books and E-Resources

For Reference Print Book -

Text Books: (As per IEEE format)

1. Bonnie Eisenman, Learning React Native: Building Native Mobile Apps with JavaScript, 1st ed., O'Reilly Media, Inc., 2016.
2. Nader Dabit, React Native in Action, 1st ed., Manning Publications, 2019.
3. Eric Masiello and Jacob Friedmann, Mastering React Native, Packt Publishing, 2017.

Reference Books: (As per IEEE format)

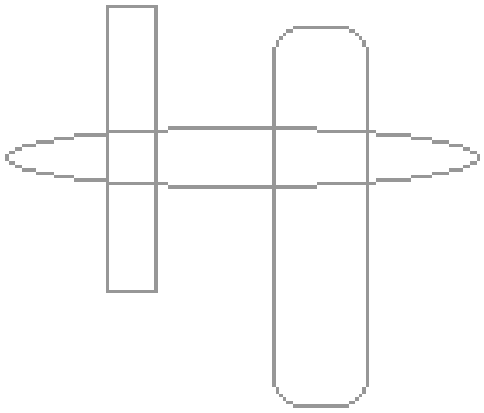
1. B. Eisenman, Learning React Native: Building Native Mobile Apps with JavaScript, 1st ed., O'Reilly Media, Inc., 2016.
2. N. Dabit, React Native in Action: Developing iOS and Android Apps with JavaScript, 1st ed., Manning Publications, 2019
3. A. Boduch and R. Derks, React and React Native: A Complete Hands-On Guide to Modern Web and Mobile Development with React.js, 3rd ed., Packt Publishing, 2020.
4. D. Abbott, H. Djirdeh, A. Accomazzo, and S. Shoemaker, Fullstack React Native: Create Beautiful Mobile Apps with JavaScript and React Native, Newline, 2019.
5. A. Paul and A. Nalwaya, React Native for iOS Development, 1st ed., Packt Publishing, 2015.
6. R. Derks, React Projects: Build Advanced Cross-Platform Projects with React and React Native to Become a Professional Developer, 2nd ed., Packt Publishing, 2022.

For Reference Electronic Book –

1.simonandschuster.co.in+3perlego.com+3ebooks.com+3perlego.com+2simonandschuster.co.in+2simonandschuster.com+2.computer-pdf.com

For MOOCs and other learning Resources

1. https://www.coursera.org/learn/react-native-coursewww.nfnlp.com
2. https://cursa.app/en/free-course/react-native-basics-



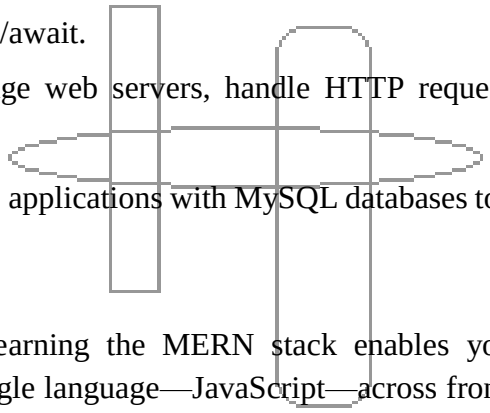
Double Minor in Full Stack Development

ABCXXX: Node.js

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2 Hours/Week
Total Credits: 3

Course Objectives:

1. To understand Node.js fundamentals, including installation, package management, and basic command-line operations.
2. Toexplore and utilize built-in and custom modules in Node.js for modular application development.
3. To learn to perform file operations and handle events using Node.js's asynchronous I/O model.
4. To master asynchronous programming techniques in Node.js, including callbacks, promises, and async/await.
5. To build and manage web servers, handle HTTP requests, and send emails using Node.js modules.
6. To integrate Node.js applications with MySQL databases to perform CRUD operations efficiently.



Course Relevance: Learning the MERN stack enables you to build full-stack web applications using a single language—JavaScript—across frontend and backend. It offers a powerful, flexible, and efficient way to create scalable, modern apps with React's dynamic UI, Express and Node's robust backend, and MongoDB's flexible database. MERN is widely used in the industry, increasing job opportunities. Its open-source nature and strong community support make development faster and easier. Overall, MERN equips you with in-demand skills to develop complete web solutions end-to-end.

Syllabus

Theory

Unit 1: Introduction **(5 Hours)**

What is Node.js?, Node.js Architecture (Event Loop, Single Threaded Model), Installing Node.js ;What is NPM, Installing Packages Locally, Installing package globally, Adding dependency in package Json, Updating packages, Initializing a Node project (npminit), Installing & using third-party packages, Semantic versioning, Package.json, Running

JavaScript files with Node, Node REPL and Basic Commands, Command-line API, Command-line options, Functions, Node.js Debugger, Type Script support

Unit 2: Node JS Modules (5 Hours)

Module, Modules Types, Core Modules, Local Modules, Modules Exports, Using built-in modules: *fs*, *path*, *url*, *buffer*, *event*, *assert*, *crypto*, *dns*, *event*, *http*, *net*, *stream*, *tls*Creating and using custom modules, Understanding CommonJS

Unit 3: File System & Events (5 Hours)

Read File, Writing a File, Opening a File, Deleting a File, Writing a file asynchronously, Other I/O Operations, Reading/Writing files asynchronously & synchronously, Creating/Deleting directories, Event Emitter and custom events, Streams and buffers

Unit 4: Asynchronous JavaScript (5 Hours)

Blocking and Non-Blocking in NodeJS, Callbacks and Events in Node JS, Async Hooks, Promises, async/await, Asynchronous context tracking, Error handling in asynchronous code, Assertion Testing: Strict and Legacy Assertion Modes

Unit 5: HTTP Module, Web Server and E-mail (5 Hours)

Web server architecture, Creating a basic HTTP server, Sending Requests, Handling HTTP requests, Handling routes manually, Sending JSON, HTML, and file responses, The Nodemailer Module, Send an Email, Multiple Receivers, Send HTML, Understanding HTTP/2 and HTTPS

Unit 6: Node.js with MySQL (5 Hours)

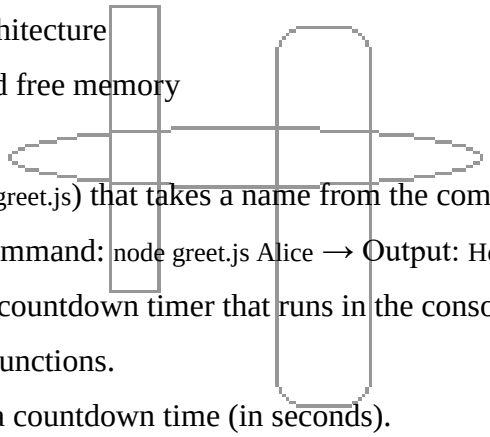
Node.js MySQL, Create Database, Create Table, Insert Into, Select From, MySQL Where, Order By, Delete, Drop Table, Update, Limit and join

Syllabus
Laboratory

List of Experiments

1. Node.js Installation and First Script and NPM & Project Initialization
2. Create a program to log system info using os and manipulate paths with path using fs, os, path, and url modules
3. Create two files: one exports a function, the other imports and uses it. Also Explain module scope and CommonJS syntax.
4. Create, read, update, and delete files both synchronously and asynchronously and log the result or error appropriately.

5. Create a custom event using events.EventEmitter.Trigger and handle the event.
6. Use streams for reading and writing large files and copy a large file using read/write streams and also demonstrate buffer handling in a basic stream.
7. Write three functions: one using a callback, one with a promise, and one with async/await for reading a file,Handle errors in each approach.
8. Create a server that responds with plain text, HTML, and JSON and Use http module and route based on URL.
9. **Send emails using Nodemailer:**
    a. Send a plain text email.
    b. Send HTML email to multiple recipients.
    c. Configure Gmail SMTP (or use test SMTP service like Mailtrap).
10. Create a script (systemInfo.js) that logs:
    a. OS type and platform
    b. CPU architecture
    c. Total and free memory
    d. Uptime
11. Create a script (greet.js) that takes a name from the command line and prints a greeting. E.g. command: node greet.js Alice → Output: Hello, Alice!
12. Create a simple countdown timer that runs in the console using JavaScript and Node.js timing functions.
    a. Ask for a countdown time (in seconds).
    b. Print the remaining time every second.
    c. Display "Time's up!" when the countdown ends.



Course Outcomes

Course Outcomes: Student will be able to

1. Set up a Node.js development environment, manage packages with NPM, and understand Node.js architecture and basic command-line operations.
2. Utilize built-in and custom modules in Node.js, applying the CommonJS module system for modular application development.
3. Perform file operations and handle events using Node.js's asynchronous I/O model, including streams and buffers.
4. Implement asynchronous programming patterns using callbacks, promises, and async/await to handle non-blocking operations.
5. Develop web servers, handle HTTP requests, and integrate email functionalities using Nodemailer in Node.js applications.
6. Integrate Node.js applications with MySQL databases to perform CRUD operations and manage data effectively.

Books and E-Resources

For Reference Print Book -

Text Books: (As per IEEE format)

1.Alex R. Young, Bradley Meck, Mike Cantelon, Tim Oxley, Marc Harter, T.J. Holowaychuk, and Nathan Rajlich, *Node.js in Action*, 2nd ed., Manning Publications, 2017

2.Mario Casciaro and Luciano Mammino, *Node.js Design Patterns*, 3rd ed., Packt Publishing, 2020.

Reference Books: (As per IEEE format)

1.Craig Buckler, *Node.js: Novice to Ninja*, SitePoint,

2.Sebastian Springer, *Node.js: The Comprehensive Guide*, SAP PRESS, 2022

3.Basarat Syed, *Beginning Node.js*, Apress, 2011.

For MOOCs and other learning Resources

1. https://www.coursera.org/learn/learn-nodejs?utm_source=chatgpt.com

Double Minor in Full Stack Development

ABCXXX: MongoDB

Teaching Scheme:
Theory: 2 Hours/Week ; Laboratory: 2/Week
Total Credits: 3

Course Objectives:
1. ToUnderstand NoSQL and MongoDB Fundamentals
2. ToMaster CRUD Operations
3. To Implement Data Modeling Techniques
4. ToUtilize Aggregation Framework
5. ToIntegrate MongoDB with Node.js and Mongoose
6. To Implement Security and Backup Strategies

Course Relevance: Learning the MERN stack enables you to build full-stack web applications using a single language—JavaScript—across frontend and backend. It offers a powerful, flexible, and efficient way to create scalable, modern apps with React's dynamic UI, Express and Node's robust backend, and MongoDB's flexible database. MERN is widely used in the industry, increasing job opportunities. Its open-source nature and strong community support make development faster and easier. Overall, MERN equips you with in-demand skills to develop complete web solutions end-to-end.

Syllabus
Theory

Unit 1: Introduction to NoSQL & MongoDB (5 Hours)
What is NoSQL?, Relational to Document Model, Comparison: SQL vs NoSQL, MongoDB Overview & Architecture, Use Cases and Benefits, Installing MongoDB locally & MongoDB Atlas, Databases, Collections, Documents, BSON Format, Basic CRUD Operations: insertOne, find, updateOne, deleteOne, MongoDB Shell and Compass GUI

Unit 2: Data Types, Operators and Indexing (5 Hours)
BSON Data Types (String, Number, Array, Date, ObjectId, etc.); Comparison and Logical Operators, Projection and Filtering documents, Need for Indexing, Creating Single Field and Compound Indexes, Indexes and Performance Impact, Explain Plan

Unit 3: Aggregation Framework and Data Modelling (5 Hours)

Introduction to Aggregation Pipeline, $match, $group, $sort, $project, $limit, Use Cases: Summarizing Data, Grouping, Filtering, Working with Arrays in Aggregation, Schema Design Basics, Schema Design Patterns and Antipatterns,, Embedding vs Referencing, One-to-One, One-to-Many, Many-to-Many Relationships, Normalization vs Denormalization

Unit 4: MongoDB with Node.js (5 Hours)

Connecting MongoDB with Node.js, Using MongoDB Native Driver, CRUD Operations with Node.js, Asynchronous Code and Promises, Query Optimization Strategies, Caching & Sharding Overview

Unit 5: MongoDB with Mongoose (5 Hours)

What is Mongoose?, Defining Schemas & Models, CRUD Operations with Mongoose, Validation and Middleware

Unit 6: Security, User Management, Backup and Restore (5 Hours)

Authentication & Authorization in MongoDB, Creating Users and Roles, Role-Based Access Control (RBAC), MongoDB Backup Tools, mongodump & mongorestore, Using MongoDB Atlas, Exporting & Importing JSON/CSV,

Syllabus
Laboratory

List of Experiments

1. Install MongoDB locally and create a free MongoDB Atlas cluster and also connect to MongoDB via Shell and Compass. List all databases and explore default ones.

2. Create a database named university and create collections: students, courses also Insert documents using insertOne() and insertMany() also retrieve documents using find().

3. Using MongoDB Shell and Compass GUI, perform the following CRUD operations on the Books collection in LibraryDB:
    a. **Create**: Insert a new book document.
    b. **Read**: Retrieve all books by a specific author.
    c. **Update**: Update the published_year of a book.
    d. **Delete**: Remove a book by its title.

4. Perform basic CRUD operations do following:
    a. insertOne, insertMany new student records.
    b. Use find() with filters.
    c. Update a student's data with updateOne().
    d. Delete a record using deleteOne().

5. Use various data types and MongoDB query operators.
    a. Create documents using String, Number, Date, Boolean, Array, ObjectId.
    b. Query using $gt, $lt, $in, $and, $or.
    c. Use projection to limit output fields

6. Utilize the aggregation pipeline to perform the following operations on the Books collection:
    a. **$match**: Filter books published after the year 2000.
    b. **$group**: Group books by genre and calculate the average rating for each genre.
    c. **$sort**: Sort the genres by average rating in descending order.
    d. **$project**: Include only the genre and averagerating fields in the output.

7. Design a schema for a blogging platform.
    a. **Embedding**: For a Post document, embed an array of comments where each comment contains author, content, and timestamp.
    b. **Referencing**: Create a separate Users collection and reference the user_id in the Post document to associate the post with a user.
    c. **Normalization**: Ensure that user information is not duplicated across multiple posts.
8. Develop a Node.js application that connects to a MongoDB database and performs CRUD operations on a Products collection.
    a. **Create**: Insert a new product document.
    b. **Read**: Retrieve all products in a specific category.

c. **Update**: Update the price of a product.
d. **Delete**: Remove a product by its product_id.
e. **Asynchronous Operations**: Implement asynchronous functions using Promises to handle database operations.

9. Using Mongoose, define a schema for a User collection with fields such as username, email, and password.
    a. **Validation**: Implement validation to ensure that email is unique and properly formatted.
    b. **Middleware**: Use pre-save middleware to hash the password before saving a user document.
    c. **CRUD Operations**: Perform CRUD operations on the User collection using Mongoose methods.

10. Perform a backup and restore operation on a MongoDB database.
    a. **Backup**: Use the mongodump utility to create a backup of the LibraryDB database.
    b. **Restore**: Use the mongorestore utility to restore the database from the backup.
    c. **Verification**: Verify that the data has been successfully restored by querying the Books collection.

11. Import and export data between MongoDB and CSV/JSON formats.
    a. **Import**: Use the mongoimport utility to import a CSV file containing book information into the Books collection.
    b. **Export**: Use the mongoexport utility to export the Books collection to a JSON file.
    c. **Data Integrity**: Ensure that the imported data matches the original CSV file.

12. Implement sharding in a MongoDB database to distribute data across multiple servers.
    a. **Sharding Key**: Choose a suitable sharding key for the Books collection.
    b. **Shard Cluster**: Set up a sharded cluster with multiple shards.
    c. **Data Distribution**: Verify that data is evenly distributed across the shards.

Course Outcomes

Course Outcomes: Student will be able to

1. Proficient in MongoDB CRUD Operations
2. Skilled in Data Modeling and Schema Design
3. Competent in Using the Aggregation Framework
4. Experienced in Indexing and Query Optimization
5. Proficient in MongoDB Integration with Node.js and Mongoose
6. Skilled in MongoDB Security and Administration

Books and E-Resources

For Reference Print Book -

Text Books: (As per IEEE format)

1. D. Hows, P. Membrey, and E. Plugge, The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing. Berkeley, CA: Apress, 2010.
2. K. Chodorow and M. Dirolf, MongoDB: The Definitive Guide. Sebastopol, CA: O'Reilly Media, 2010.
3. D. Hows, P. Membrey, E. Plugge, and T. Hawkins, The Definitive Guide to MongoDB: A Complete Guide to Dealing with Big Data Using MongoDB. 2nd ed. Berkeley, CA: Apress, 2013.
4. S. Bradshaw, E. Brazil, and K. Chodorow, MongoDB: The Definitive Guide, 3rd Edition. Sebastopol, CA: O'Reilly Media, 2019.

Reference Books: (As per IEEE format)

1. D. Hows, P. Membrey, and E. Plugge, MongoDB Basics. Berkeley, CA: Apress, 2014.
2. S. Gupta and N. Sabharwal, Practical MongoDB. Berkeley, CA: Apress, 2015.
3. D. Bierer, MongoDB 4 Quick Start Guide: Learn the Skills You Need to Work with the World's Most Popular NoSQL Database. Birmingham, UK: Packt Publishing, 2018.
4. G. Harrison and M. Harrison, MongoDB Performance Tuning. Berkeley, CA: Apress, 2021.

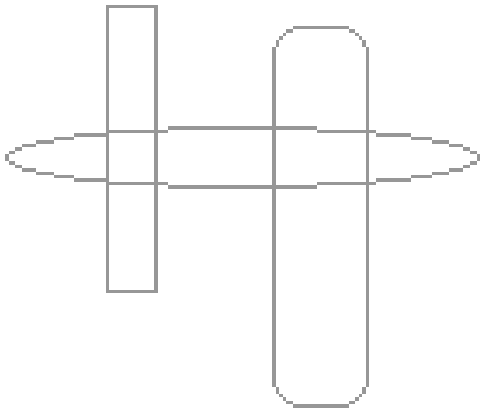
5. S. Hoberman, MongoDB Data Modeling and Schema Design. 1st ed. New York, NY: Technics Publications, 2023.
6. V. Subramanian, "MongoDB," in Pro MERN Stack, Berkeley, CA: Apress, 2019, pp. 137–169.

For Reference Electronic Book –

K. Seguin, *The Little MongoDB Book*, [Online]. Available: https://www.dbooks.org/the-little-mongodb-book-1512/. [Accessed: 6-Jun-2025].

For MOOCs and other learning Resources

1. https://learn.mongodb.com/
2. https://www.coursera.org/learn/introduction-to-mongodb
3. https://www.udemy.com/course/mongodb-the-complete-developers-guide/



Double Minor in Full Stack Development

ABCXXX: Express.js

Teaching Scheme:
Theory: 3 Hours/Week; Laboratory: 2/Week
Total Credits: 4

Course Objectives:

1. To Understand Node.js Fundamentals:
2. To Develop RESTful APIs with Express.js
3. To Integrate MongoDB for Data Management
4. To Implement Authentication and Authorization
5. To Enhance Application Security and Performance
6. To Deploy and Test Backend Applications

Course Relevance: Learning the MERN stack enables you to build full-stack web applications using a single language—JavaScript—across frontend and backend. It offers a powerful, flexible, and efficient way to create scalable, modern apps with React's dynamic UI, Express and Node's robust backend, and MongoDB's flexible database. MERN is widely used in the industry, increasing job opportunities. Its open-source nature and strong community support make development faster and easier. Overall, MERN equips you with in-demand skills to develop complete web solutions end-to-end.

Syllabus
Theory

Unit 1: Introduction, Routing and Middleware (5 Hours)

Review of Node.js: Event Loop & Architecture, What is Express.js? Why use Express? Installing Express, Express Essential Functions: Express, raw, router, static, text, urlencoded; Creating first Express app, Express Application Functions, Request Functions, Response Functions, Router Functions

Unit 2: HTTP, Routing, Middleware, Forms and Static Files (5 Hours)

HTTP Methods: GET, POST, PUT, DELETE, Creating routes with Express, Express Router, Custom Middleware, Built-in middleware (body-parser, express.static), Parsing, JSON and URL-encoded data, Handling form submission and data validation, Using Postman to test APIs, **Templating & Static Files:** Using EJS or Handlebars, Serving static files (images, CSS, JS), Creating dynamic HTML pages

Unit 3: REST API Development (5 Hours)

Creating a RESTful API for a resource**;** Enabling communication with other applications and front-end interfaces, CRUD operations, Route parameters and query strings, **Single-Page Applications (SPAs) support.**

Unit 4: MongoDB Integration using Mongoose (5 Hours)

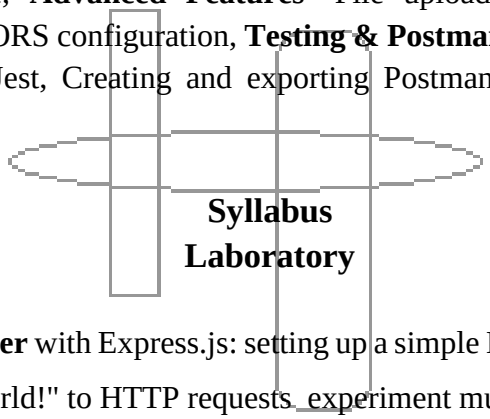
Introduction to MongoDB & Mongoose, Connecting to a MongoDB database, Creating Schemas & Models, Performing CRUD with MongoDB

Unit 5: Express Router and Modularization (5 Hours)

Separating routes into Units, Organizing controllers and models, MVC Pattern Basics, **Error Handling and Logging,** Centralized error handling middleware, HTTP Status codes, Logging with morgan or custom logger

Unit 6: Authentication , Authorization and Testing (5 Hours)

JWT Authentication, (JSON Web Token), User registration and login, protecting routes with JWT middleware, **Advanced Features-** File upload with multer, Environment variables with dotenv, CORS configuration, **Testing & Postman Collections:** Writing tests using Mocha/Chai or Jest, Creating and exporting Postman collections, Mocking and testing endpoints



Syllabus
Laboratory

List of Experiments

1. Basic **Web Server** with Express.js: setting up a simple Express server that responds with "Hello, World!" to HTTP requests. experiment must use routing, middleware, and request handling.
2. Develop a **CRUD-based to-do list** app using Express.js and MongoDB. Implement features like adding, updating, deleting, and marking tasks as completed. This should reinforces your understanding of RESTful APIs and database integration.
3. Create an application that fetches **weather** data from an external API (e.g., OpenWeatherMap) based on user input use your knowledge of handling asynchronous operations and API integrations in Express.js.
4. Build a **movie search engine** that allows users to search for movies using The Movie Database (TMDb) API. Display movie details such as title, release date, and synopsis. Apply your skills in API consumption and data rendering.

5. Implement **user authentication** in your Express.js application using Passport.js. Allow users to register, log in, and manage sessions. Use your understanding of authentication mechanisms and session management.
6. Develop **a real-time chat application** using Express.js and Socket.io. Enable users to send and receive messages instantly without refreshing the page Use WebSockets and real-time communication
7. Create a **job search portal** that allows users to search for job listings, apply filters, and save their favorite jobs. Integrate with a job listings API and implement user authentication. Use API integration with user management.
8. Build a basic **e-commerce website** where users can browse products, add items to their cart, and proceed to checkout. Implement features like product search, user authentication, and order management.
9. Develop a **collaborative drawing application** where multiple users can draw on a shared canvas in real-time. Use Express.js and Socket.io to handle real-time updates.
10. Create an application that allows users to **search for recipes** based on ingredients they have. Integrate with a recipe API and implement search functionality.
11. Build a **URL shortener service** similar to Bitly, where users can input a long URL and receive a shortened version. Implement features like custom aliases and redirect handling. Use your understanding of routing and data storage.
12. Develop an application that allows users to send emails using **Nodemailer**. Implement features like sending HTML-formatted emails and scheduling email deliveries. Use email handling and background tasks.

Course Outcomes

Course Outcomes: Student will be able to

1. **Master Node.js Fundamentals**: Understand asynchronous programming, the event loop, and core modules in Node.js.
2. **Develop RESTful APIs with Express.js**: Design and implement RESTful APIs using Express.js, handling various HTTP methods and middleware.
3. **Integrate MongoDB with Mongoose**: Connect to MongoDB databases, define schemas, and perform CRUD operations using Mongoose.
4. **Implement Authentication and Authorization**: Secure applications by implementing user authentication and authorization mechanisms.
5. **Enhance Application Security and Performance**: Apply best practices for securing applications and optimizing performance.
6. **Deploy and Test Backend Applications**: Prepare applications for production by deploying them to cloud platforms and ensuring reliability through testing.

Books and E-Resources

For Reference Print Book -

Text Books: (As per IEEE format)

1. Brad Dayley, Brendan Dayley, and Caleb Dayley, Node.js, MongoDB and Angular Web Development: The Definitive Guide to Using the MEAN Stack to Build Web Applications (2nd Edition), Addison-Wesley Professional, 2017. ISBN: 978-0-13-465553-6.
2. Greg Lim, Beginning Node.js, Express & MongoDB Development, CreateSpace Independent Publishing Platform, 2019. ISBN: 978-1-978-37955-7.
3. Simon Holmes, Getting MEAN with Mongo, Express, Angular, and Node, Manning Publications Co., 2015. ISBN: 978-1-61729-203-3.
4. Fernando Monteiro, Node.js 6.x Blueprints: Create Stunning Web Applications and RESTful APIs from Start to Finish with Express, Loopback, MongoDB, and MySQL, Packt Publishing, 2016. ISBN: 978-1-78528-370-1.

Reference Books: (As per IEEE format)

1. **Azat Mardan**, Pro Express.js: Master Express.js The Node.js Framework For Your Web Development, Apress, 2014. ISBN: 978-1-4842-0038-4.
2. **Azat Mardan**, Express.js Deep API Reference, Apress, 2014. ISBN: 978-1-4842-0782-6.
3. **Greg Lim**, Beginning Node.js, Express & MongoDB Development, CreateSpace Independent Publishing Platform, 2019. ISBN: 978-1-978-37955-7.

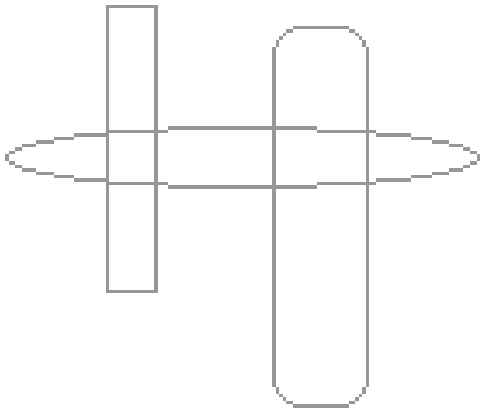
4. **Jason Krol**, Web Development with MongoDB and Node.js, Packt Publishing, 2013. ISBN: 978-1-78328-484-3.
5. **Simon Holmes**, Getting MEAN with Mongo, Express, Angular, and Node, Manning Publications Co., 2015. ISBN: 978-1-61729-203-3.

For Reference Electronic Book –

1. Azat Mardan, Pro Express.js: Master Express.js The Node.js Framework For Your Web Development, 1st ed. Berkeley, CA: Apress, 2014. [Online]. Available: https://www.oreilly.com/library/view/pro-expressjs/9781484200377

For MOOCs and other learning Resources

1. https://www.udemy.com/course/the-complete-nodejs-developer-course-2/
2. https://university.mongodb.com/courses/M001/about



Double Minor in Full Stack Development

ABCXXX: Project

Teaching Scheme: Laboratory: 8 Hours/Week
Total Credits: 4

Syllabus

Aim

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

Project Group and Topic Selection and Synopsis:

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

Overview of the Course:

1. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
2. The project requires the students to conceive, design, implement and operate a mechanism (the design problem). The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts. If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

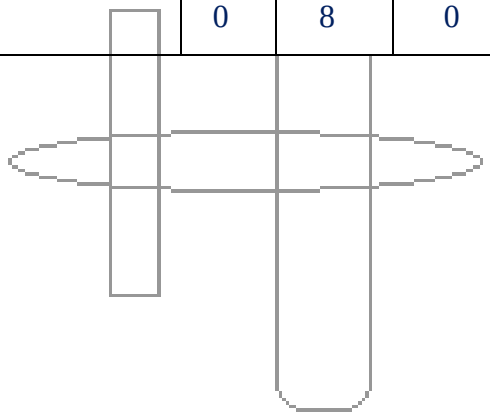
3. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
4. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
5. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
6. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
7. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
8. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

Note:

The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

Double Minor Course: Essentials of Computer Engineering

Course No.	Semester	Course Name	Teaching Scheme (Hours/Week)			Examination Scheme				Credits
			Th	Lab	Tut				Total	
C1	III	Data Structures and Algorithms	2	2	0					3
C2	IV	Database Management systems	2	2	0					3
C3	V	Operating Systems and Networking	2	2	0					3
C4	VI	Agile Methodology	2	2	0					3
C5	VII	Web Technologies	2	2	1					4
C6	VIII	Project	0	8	0					4



Double Minor in Essentials of Computer Engineering

ABCXXX: Data Structures and Algorithms

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Credits: 3

Course Prerequisites: Basic programming Skills (C/C++)

Course Objectives:

1. To introduce the basic concepts of data structures and algorithms.
2. To learn and understand linear and non-linear data structure constructs.
3. To implement searching and sorting techniques using linear data structures.
4. To understand how to solve problems using step by step approach with the help of fundamental data structures.
5. To associate data structures in developing and implementing efficient algorithms.

Course Relevance:

This is a basic Course for Computer Engineering and allied branches. This course has a high relevance in all domains of computer engineering such as in Industries; research etc. as a basic prerequisite course. Data Structures are a crucial part of computer algorithms as they allow programmers to do data management efficiently. A wise selection of data structures can improve the performance of a computer program or algorithm in a more useful way.

Syllabus
Theory

Section 1: Linear Data Structures

Unit 1: Arrays **(4 Hours)**

Asymptotic Notations, Time and Space Complexity Introduction, Memory Representation and application of Single and Multidimensional arrays, Sparse Matrix. **Searching and sorting techniques:** Linear Search, Binary search with Analysis. **Sorting Techniques:** Bubble Sort, Insertion Sort, Merge Sort, Quick Sort with Analysis and passes.

Unit 2: Linked Lists **(4 Hours)**

Dynamic memory allocation, Singly Linked Lists, Doubly linked Lists, Circular linked lists and Generalized linked lists, Applications of Linked list, introduction to Vectors and Application.

Unit 3: Stacks and Queues (6 Hours)

Stack: Stack representation and Implementation using arrays and Linked lists. Applications of stack in Recursion, Expression conversions and evaluations. **Queues:** Representation and implementation using array and Linked lists, Types of queue. Applications of Queues: Job Scheduling, Josephus problem etc.

Section 2: Non-Linear Data Structures

Unit 4: Trees (7 Hours)

Basic terminology, representation using array and linked lists. Tree Traversals: Recursive and Non recursive, Operations on binary tree. Binary Search trees (BST).

Unit 5: Graphs (7 Hours)

Terminology and representation using Adjacency Matrix and Adjacency Lists, Graph Traversals and Application: BFS and DFS, Connected graph, Bipartite Graph, Detecting Cycle in graph. Minimum Spanning tree: Prims and Kruskal's Algorithm, Shortest Path Algorithms, Union Find.

Unit 6: Hashing (2 Hours)

Hashing techniques, Hash table, Hash functions. Collision handling and Collision resolution techniques.

Syllabus
Laboratory

List of Experiments

1) To implement the different sorting algorithms.

2) To implement the linked list.

3) To implement any application of Stack data structure.

4) Implement various expression conversions using Stack.

5) To implement any application of Queue data structure.

6) To implement an algorithm to perform Binary Search Tree (BST) operations (Create, Insert, Delete and Traversals).

7) To implement an algorithm to perform various operations on Binary Tree (Mirror image, Height, Leaf node display, Level wise display etc.)

8) To implement an algorithm to perform various Tree traversals using Stack.

9) To implement Graph traversal: algorithms: Depth First Search and Breadth First Search.

10) To implement Prim's and Kruskals Algorithms to find a Minimum Spanning Tree (MST).

11) To implement Dijkstra's algorithm to solve a Single Source Shortest Path Problem.

12) To implement Hashing algorithms.

Syllabus

Course Project

List of Course Projects

1) Finding Nearest Neighbors.

2) Calendar Application using File handling.

3) Path finder in Maze.

4) Word Completion Using Trie.

5) Bloom Filters.

6) Different Management Systems.

7) Scheduling Applications and Simulation.

8) Shortest Path Applications. (Kirchhoff's Circuit, TSP with Scenarios).

9) Efficient Storage and Data Retrieval Systems.

10) Different Gaming Application.

Course Outcomes

Course Outcomes: Student will be able to –

1) Make use of single and multi-dimensional array for searching and sorting based applications.

2) Construct computer science applications with the help of dynamic storage representation.

3) Build computer science applications using stacks and queues.

4) Demonstrate the use of tree data structure to represent and manipulate hierarchically organized data in various applications.

5) Utilize graph data structure to design social media, network based and circuit applications.

6) Design and develop the single and multithreads applications by applying hash table and hash map techniques.

Books and E-Resources

Text Books:

1. E. Horwitz , S. Sahani, Anderson-Freed, " Fundamentals of Data Structures in C", Second Edition, Universities Press.

2. Y. Langsam, M.J. Augenstein, A.M.Tenenbaum, "Data structures using C and C++", Pearson Education, Second Edition.

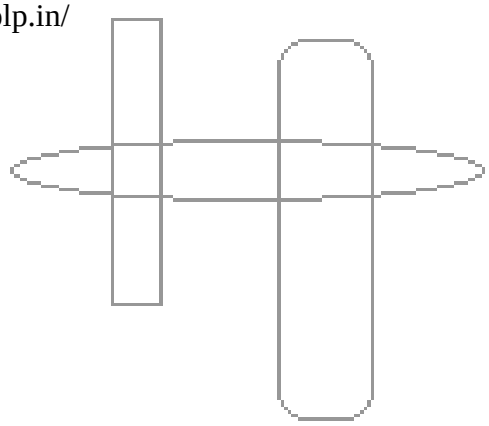
3. Narasimha Karumanchi, "Data Structures and Algorithm Made Easy", Fifth Edition, CareerMonk publication.

Reference Books:

1.J. Tremblay, P. Soresan, "An Introduction to data Structures with applications", TMHPublication, 2nd Edition.

For MOOCs and other learning Resources

1. www.nptelvideos.in,
2. www.geeksforgeeks.org
3. https://www.youtube.com/watch?v=244YpoG1pqA&list=PLrikLQMZHuSonRoDheibeb9ffd9phWIyu&index=5
4. https://classroom.volp.in/



Double Minor in Essentials of Computer Engineering

ABCXXX: Database Management systems

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2/Week;
Total Credits: 3

Prerequisite: Data Structures and Algorithms

List of Objectives

1. To understand the fundamentals of database systems and data models.
2. To learn relational algebra and SQL for data retrieval and manipulation.
3. To design normalized relational schemas using functional dependencies.
4. To study transaction management and concurrency control techniques.
5. To explore indexing and query optimization strategies.
6. To implement mini projects using modern DBMS tools.

Syllabus
Theory

SECTION I:

Unit 1: Introduction to Databases and Data Models (4 Hours)
Database system applications, DBMS architecture, data models (ER, relational, object-based), schema and instances, data independence.

Unit 2: Relational Model and Relational Algebra (5 Hours)
Basic structure, keys, relational algebra operations (select, project, join, union, etc.), expression trees, use in query processing.

Unit 3: SQL and Advanced Querying (5 Hours)
DDL, DML, DCL; constraints; nested queries; set operations; joins; views; functions; indexing basics; triggers and stored procedures.

SECTION II:

Unit 4: Database Design and Normalization (4 Hours)
ER and EER diagram design, mapping to relational model, functional dependencies, 1NF, 2NF, 3NF, BCNF; schema refinement.

Unit 5: Transaction Management and Concurrency Control (5 Hours)
Concept of transaction, ACID properties, schedules, serializability, concurrency control protocols, deadlocks, recovery mechanisms.

Unit 6: Query Optimization and Indexing (5 Hours)
Query processing, heuristics and cost-based optimization, B+ tree and hashing index structures, access path selection.

Syllabus
Laboratory

List of Experiments

1. Write SQL queries for basic CRUD operations.
2. Implement complex queries using joins, group by, and nested queries.
3. Design and create ER and relational schema.
4. Perform normalization on sample datasets.
5. Create and manipulate views, triggers, and stored procedures.
6. Simulate transaction processing and concurrency control.
7. Use indexing on sample tables and analyze performance.
8. Conduct query optimization using EXPLAIN in MySQL/PostgreSQL.
9. Integrate backend application with DBMS (e.g., using Python or PHP).
10. Implement a mini project (Library/Inventory Management etc.).

Syllabus
Course Project

List of Course Projects

1. Student Record Management System
2. Online Examination Portal DBMS backend
3. Hospital or Clinic Database System
4. Inventory or Stock Management System
5. Tourism Package Booking Database System
6. Mini Banking or ATM Simulation Database

Syllabus
Tutorials

List of Tutorials

1. ER to relational model conversion
2. Functional dependency and normalization exercise
3. Relational algebra expression solving
4. Complex SQL queries worksheet
5. Transaction schedule evaluation and serializability check
6. Index-based query analysis

Course Outcomes

Course Outcomes: Students will be able to

1. Understand database concepts and data models.
2. Apply relational algebra and SQL for data manipulation.
3. Design normalized relational databases.
4. Implement transaction processing and concurrency control.
5. Analyze and optimize queries using indexing.
6. Develop mini applications using modern DBMS tools.

Books and E-Resources

Text Books

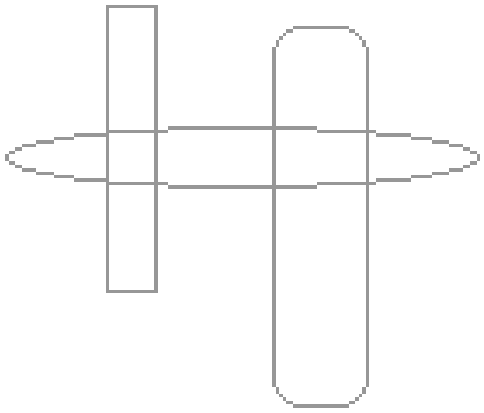
1. Abraham Silberschatz, Henry Korth, S. Sudarshan, Database System Concepts, 7th Edition, McGraw Hill.
2. Ramez Elmasri, Shamkant Navathe, Fundamentals of Database Systems, 7th Edition, Pearson.
3. Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, McGraw Hill.

Reference Books

1. Peter Rob, Carlos Coronel, Database Systems: Design, Implementation, and Management, Cengage Learning.
2. C.J. Date, An Introduction to Database Systems, 8th Edition, Pearson Education.
3. J. Ullman, J. Widom, A First Course in Database Systems, Pearson.

MOOCs and Resources

1. NPTEL Course by Prof. P. Dasgupta, IIT Kharagpur - https://nptel.ac.in/courses/106105175
2. Stanford Online - Databases by Jennifer Widom - https://online.stanford.edu/courses/sohs-ydba0001-databases
3. DBMS Specialization - Coursera (by Univ. of Colorado Boulder) - https://www.coursera.org/specializations/database-systems
4. W3Schools SQL Tutorial - https://www.w3schools.com/sql/
5. GeeksforGeeks DBMS - https://www.geeksforgeeks.org/dbms/



Double Minor in Essentials of Computer Engineering

ABCXXX: Operating Systems and Networking

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2/Week;
Total Credits: 3

Course Objectives:

1. To learn types and components of operating systems
2. To learn process and memory scheduling algorithms
3. To learn process synchronization and file system
4. To learn types of computer networks and networking models
5. To learn functions of network layer
6. To learn transport and application layer protocols

Course Relevance: The Operating System and Computer Networks courses are foundational for understanding how computers and distributed systems function. Operating Systems teach process management, memory handling, file systems, and system security, crucial for software development and system design. Computer Networks cover data communication, protocols, and network architecture, enabling understanding of the internet and distributed applications. Together, they build core skills for careers in systems programming, cloud computing, and network administration.

Syllabus

Theory

Unit 1: Introduction to Operating Systems (3 Hours)

What is an Operating System (OS)?, History and Evolution of OS, Types of Operating Systems: Batch, Multitasking, Multiprocessing, Real-Time, Distributed; OS Components: Kernel, Shell, File System; System Calls and OS Services

Unit 2: Process and Memory Management (6 Hours)

Concept of Process and Thread, Process States and Lifecycle, Context Switching, Process Control Block (PCB), Scheduling Algorithms: FCFS, SJF, Round Robin, Priority Scheduling;

Address Binding and Logical vs Physical Address, Contiguous Memory Allocation, Paging and Segmentation, Virtual Memory and Page Replacement Algorithms, FIFO, LRU, Optimal, Demand Paging and Thrashing

Unit 3: Process Synchronization and File Systems (6 Hours)

Race Conditions, Critical Section Problem, Synchronization Mechanisms: Mutex, Semaphores, Monitors; Deadlock: Conditions, Prevention, Avoidance, Detection, Recovery;

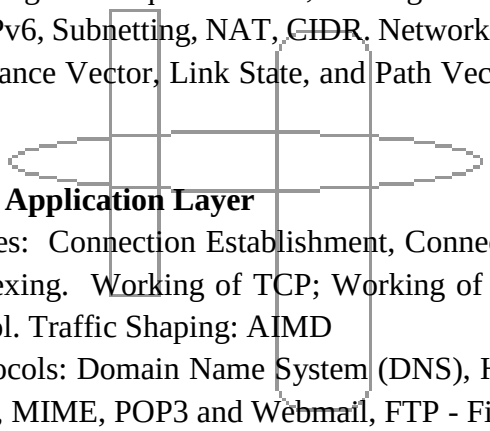
File Concepts: Structure, Types, Access Methods; Directory Structures: Single, Two-level, Tree, Acyclic Graph; File Allocation Methods: Contiguous, Linked, Indexed; Disk Scheduling Algorithms: FCFS, SSTF, SCAN, C-SCAN

Unit 4: Computer Network Reference Models and Network Devices (4 Hours)

What is Computer Networks, Types of Computer Networks: LAN, WAN, MAN, PAN, Internetworking, Internet, Intranet, Network Topologies: Bus, Ring, Mesh, Star, Tree, and Hybrid; OSI Reference Model: 7 Layers; TCP/IP Reference Model; Protocols and Standards; Network Devices: Switch, Router and Access Points

Unit 5: Network Layer (5 Hours)

Network Layer: Switching Techniques: Circuit, Message and Packet Switching. Logical Addressing: IPv4 and IPv6, Subnetting, NAT, CIDR. Network Layer Protocols: IP, ICMP, Routing Protocols: Distance Vector, Link State, and Path Vector, Congestion control and QoS



Unit 6: Transport and Application Layer (6 Hours)

Transport Layer Services: Connection Establishment, Connection Release, Flow control and Buffering, Multiplexing. Working of TCP; Working of UDP; , Quality of Service: TCP Congestion Control. Traffic Shaping: AIMD

Application Layer Protocols: Domain Name System (DNS), HTTP- Hyper Text Transfer Protocol, Email: SMTP, MIME, POP3 and Webmail, FTP - File Transfer Protocol

Syllabus
Laboratory

List of Experiments

1. Explore basic commands in Linux ls, pwd, cd, ps,top, htop, ps, kill in Linux and
2. Simulate scheduling algorithms with small code snippets or visualization tools
3. Simulate Visual demo or simulation of paging
4. Write a simple program that mimics paging or replacement
5. Simulate Disk Scheduling Algorithms with small code snippets or visualization tools
6. Implement basic semaphore logic in code

7. Demonstration of deadlock and recovery via code
8. Hands on for network commands - ping, pathping, ipconfig/ifconfig, arp, netstat, nbtstat, nslookup, route, traceroute/tracert, nmap
9. Installation and Configuration of Wireless Access Point
10. Practice on Examples of Network Layer Logical Addressing- Classful IP
11. Practice on Examples of Network Layer Logical Addressing - CIDR: Subnetting, IP Prefixes
12. Analyze TCP, UDP and HTTP traffic using Wireshark

Course Outcomes

Course Outcomes: Student will be able to

1. Select operating system as platform
2. Choose process and memory scheduling algorithms
3. Decide on type of file system
4. Select topology, essential components to design computer networks
5. Apply classful and CIDR logical addressing
6. Analyze data flow between peer to peer in an IP network using Application, Transport and Network Layer Protocols

Books and E-Resources

For Reference Print Book -

Text Books: (As per IEEE format)

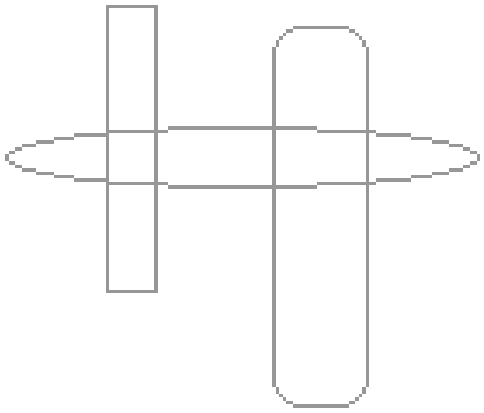
4. A. Silberschatz, P. B. Galvin, and G. Gagne, Operating System Concepts, 10th ed., Hoboken, NJ, USA: Wiley, 2018.
5. W. Stallings, Operating Systems: Internals and Design Principles, 9th ed., Boston, MA, USA: Pearson, 2018.
6. A. S. Tanenbaum and H. Bos, Modern Operating Systems, 4th ed., Upper Saddle River, NJ, USA: Pearson, 2014.
7. Andrew S. Tanenbaum, “Computer Networks”,5th Edition, PHI, ISBN 81-203-2175-8.
8. Kurose, Ross “Computer Networking a Top Down Approach Featuring the Internet”, Pearson; 6th edition (March 5, 2012), ISBN-10: 0132856204

Reference Books: (As per IEEE format)

5. A. S. Tanenbaum and H. Bos, Modern Operating Systems, 4th ed., Upper Saddle River, NJ, USA: Pearson, 2014.
6. Frouzan B., "Data Communications and Networking", 5th edition, Tata McGraw-Hill, Publications, 2006

For MOOCs and other learning Resources

5. https://www.coursera.org/learn/os-power-user
6. https://www.edx.org/learn/operating-systems
7. https://ocw.mit.edu/courses/6-828-operating-system-engineering-fall-2012/
8. https://onlinecourses.nptel.ac.in/noc24_cs19/preview
9. https://onlinecourses.swayam2.ac.in/cec22_cs05/preview



Double Minor in Essentials of Computer Engineering

ABCXXX: Agile Methodology

Teaching Scheme:
Theory: 2 Hours / Week, Tutorial: 1 Hour / Week
Total Credits: 3

Syllabus

Theory

Unit 1: Introduction to Agile Methodology (5 Hours)

Evolution from traditional to agile approaches, Agile Manifesto and 12 principles, Characteristics of agile processes, Agile vs. traditional models, Benefits and challenges of agile adoption.

Unit 2: Scrum Framework and Roles (5 Hours)

Scrum theory: roles, events, and artifacts, Product Owner, Scrum Master, Development Team, Scrum events: Sprint, Planning, Review, Retrospective, Scrum artifacts: Product Backlog, Sprint Backlog, Increment, Agile values in Scrum.

Unit 3: Agile Practices and Estimation Techniques (5 Hours)

Writing user stories and epics, Acceptance criteria, Definition of Ready/Done, Estimation: Story points, Planning Poker, Prioritization techniques: MoSCoW, Kano model, Agile contracts and budgeting basics.

Unit 4: Agile Project Tracking and Tools (5 Hours)

Velocity, Burndown and Burnup charts, Managing sprint board with JIRA / Trello, Agile project documentation, Managing changes in sprint scope, Introduction to CI/CD pipelines in agile.

Unit 5: Agile Quality and Continuous Improvement (5 Hours)

Agile testing approaches: TDD, BDD, exploratory testing, Agile metrics: Cycle time, Lead time, Defect density, Agile retrospectives and continuous improvement, Team collaboration and feedback mechanisms, QA role in agile environments.

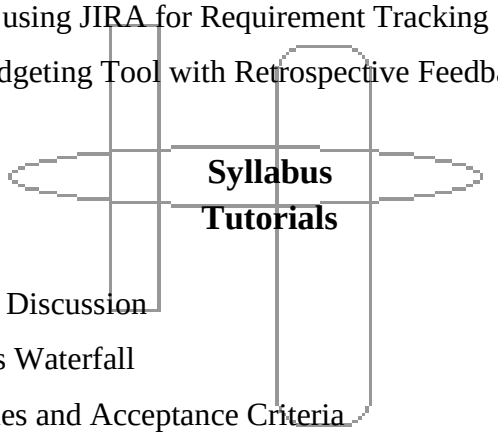
Unit 6: Scaling Agile and Industry Case Studies (5 Hours)

Introduction to SAFe, LeSS, Spotify model, Agile in large teams and distributed environments, Industry case studies (India & global), Common pitfalls and how to overcome them, Agile certifications and career tracks.

Syllabus
Course Project

List of Course Projects

1. Agile-Based Online Course Registration System
2. Scrum-Managed Inventory Management Application
3. Trello-Backed Task Manager for Students
4. Agile Tracker for Personal Productivity
5. Project Management Dashboard for Freelancers
6. E-Library Management with Agile Sprint Planning
7. Food Ordering App using Agile Lifecycle
8. Health Appointment Scheduling System with User Stories
9. Agile-Driven Hostel Complaint Management App
10. Event Planning App (Hackathons/Workshops) with Kanban Flow
11. Mini CRM App using JIRA for Requirement Tracking
12. Time-Boxed Budgeting Tool with Retrospective Feedback



Syllabus
Tutorials

List of Tutorials

1. Agile Manifesto Discussion
2. Identify Agile vs Waterfall
3. Write User Stories and Acceptance Criteria
4. Sprint Planning Exercise
5. Agile Estimation Session
6. Scrum Roles Roleplay
7. Retrospective Practice
8. Agile Metrics Interpretation
9. Backlog Grooming Session
10. Agile Tool Walkthrough
11. Case Study Analysis
12. Agile Presentation

Course Outcomes

Course Outcomes: Students will be able to

1. Compare the principles and values of Agile methodology with traditional models.
2. Apply the Scrum framework including roles, events, and artifacts in agile project settings.
3. Construct and prioritize user stories and epics, and estimate tasks using agile techniques.
4. Utilize agile tools like JIRA/Trello to manage and track agile projects effectively.
5. Analyze agile metrics and testing strategies to ensure quality and continuous improvement.
6. Evaluate agile scaling approaches and assess real-world agile adoption through case studies.

Books and E-Resources

For Reference Print Book -

1. K. Rubin, 'Essential Scrum: A Practical Guide to the Most Popular Agile Process', 1st Edition, Addison-Wesley, 2012
2. C. Stellman, J. Greene, 'Learning Agile', 1st Edition, O'Reilly Media, 2014

For Reference Electronic Book –

1. M. Cohn, 'Agile Estimating and Planning', 1st Edition, Pearson, 2005, Available: https://ebookcentral.proquest.com

For MOOCs and other learning Resources

1. John Rofrano, IBM, 'Introduction to Agile Development and Scrum', Coursera,- https://www.coursera.org/learn/agile-development-and-scrum

Double Minor in Essentials of Computer Engineering

ABCXXX: Web Technologies

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2/Week; Tutorial: 1 Hour/Week
Total Credits: 4

Course Objectives:
Students are able

1. To Master Core JavaScript Syntax
2. To Develop Functions and Implement Modular Code
3. To Manipulate the DOM and Handle Events
4. To Implement Object-Oriented Programming (OOP) Principles
5. To Handle Errors and Debug JavaScript Code
6. To Understand Asynchronous Programming

Course Relevance: The key technology of the information age is global communication. Web technology is a truly global area of study as it enables global communication with the help of web sites. Web technologies are the backbone of all IT infrastructures and their applications in the world. These technologies and applications often emerge in communication within countries of countries and spread rapidly around the world. The main objective of the course is present the basic web technology concepts that are required for developing web applications. The key technology components are descriptive languages, server-side program elements and client-side program elements. In addition, the course gives specific contents that are beneficial for developing web-based solutions, like relational data-base communication basics and information security principles and approaches. Most of the jobs available in the IT industries are web technology related.

Syllabus
Theory

Unit 1: JavaScript Programming (5 Hours)

JavaScript: Overview of JavaScript, Data types, Control Structures, Arrays, Functions, and Scopes, HTML5 forms Validation, Objects in JS, flex, DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM. Asynchronous JavaScript and Error Handling in JavaScript.

Unit 2: TypeScript (4 Hours)

Understanding TypeScript as a statically typed superset of JavaScript, Setting up the development environment and compiling TypeScript code, Exploring the benefits of TypeScript in large-scale application development. Working with primitive types: string, number, Boolean. Utilizing special types: any, void, never, undefined, null, Creating and manipulating arrays and tuples, Defining object types and Enums, Implementing type aliases and interfaces

Unit 3: React (6 Hours)

Introduction to React, React component, JSX, Render function, Component API, Component lifecycle, State, Props, Mixins, Component composition, Pass data from parent to child, Pass data from child to parent, Component styling, Forms, Events, Refs, Keys, Router, Flux, Redux

Unit 4: Node JS (5 Hours)

Introduction to Node JS, Installation of Node JS, Node JS Modules, Node Package Manager (NPM), Creating Web server, File System, Express JS, Serving Static Resources, Database connectivity.

Unit 5: Node.js with MySQL and MongoDB (6 Hours)

Node.js MySQL, Create Database, Create Table, Insert Into, Select From, MySQL Where, Order By, Delete, Drop Table, Update, Limit and join

What is NoSQL?, Relational to Document Model, Comparison: SQL vs NoSQL, MongoDB Overview & Architecture, Use Cases and Benefits, Installing MongoDB locally & MongoDB Atlas, Databases, Collections, Documents, BSON Format, Basic CRUD Operations: insertOne, find, updateOne, deleteOne, MongoDB Shell and Compass GUI

.

Unit 6: Express JS (3 Hours)

Introduction to express.js, Routing and Middleware, HTTP, Routing, Middleware, Forms and Static Files, HTTP Methods: GET, POST, PUT, DELETE, Templating & Static Files , REST API Development, Single-Page Applications (SPAs) support. MongoDB

Integration using Mongoose, Express Router and Modularization, Authentication, Authorization and Testing

Syllabus
Laboratory

List of Experiments

1. Validate HTML5 form inputs using JavaScript and manipulate the DOM based on user input. Create a registration form that checks for valid email, password strength, etc., and displays a success message dynamically.
2. Use fetch() to retrieve JSON data from a placeholder API and update DOM elements dynamically. Display a list of users fetched from https://jsonplaceholder.typicode.com/users.
3. Write and compile TypeScript code using basic types, enums, and interfaces. Create a program that models a library system using interfaces and types.
4. Demonstrate usage of tuples, arrays, and functions with strict type enforcement. Inventory tracker for a store with strict type safety.
5. Build a basic React app using components, state, and props. Add, complete, and delete tasks from a to-do list
6. Pass data from parent to child and vice versa; style using inline and external styles. Calculator app where inputs and operations are separate components.
7. Calculator app where inputs and operations are separate components. Serve a static HTML file and accept form data via POST.
8. Use Express.js to serve static files and handle routes. Serve a portfolio website with routes like /about, /projects, /contact.
9. Perform CRUD operations (Create, Read, Update, Delete) on a MySQL database. Create Student management system with database backend.
10. Connect Node.js to MongoDB and perform CRUD using Mongoose. Create Product catalog with MongoDB.
11. Build a RESTful API that performs CRUD operations on a MongoDB database. Create Blog API with endpoints like /posts, /posts/:id.
12. Use Express Router and middleware for login authentication. Create Login system where users can register and authenticate securely.

Course Outcomes

Course Outcomes: Student will be able to

1. Develop and run JavaScript programs to perform calculations, manipulate data, and automate tasks.
2. Create reusable code blocks with functions, including understanding scope, parameters, and return values.
3. Proficient in MongoDB CRUD Operations
4. Skilled in Data Modeling and Schema Design
5. **Code for** asynchronous programming, the event loop, and core modules in Node.js.
6. **Develop RESTful APIs with Express.js**: Design and implement RESTful APIs using Express.js, handling various HTTP methods and middleware.

Books and E-Resources

For Reference Print Book -

Text Books: (As per IEEE format)

1. D. Flanagan, JavaScript: The Definitive Guide, 7th ed., Sebastopol, CA, USA: O'Reilly Media, 2020.
2. B. Eich, JavaScript: The Good Parts, 1st ed., Sebastopol, CA, USA: O'Reilly Media, 2008.
3. B. Freeman and J. Powell, Learning TypeScript, 2nd ed., Birmingham, UK: Packt Publishing, 2021.

Reference Books: (As per IEEE format)

1. A. Banks and E. Porcello, *Learning React: Modern Patterns for Developing React Apps*, 3rd ed., Sebastopol, CA, USA: O'Reilly Media, 2023.
2. E. Brown, *Learning JavaScript Data Structures and Algorithms*, 3rd ed., Birmingham, UK: Packt Publishing, 2020.
3. B. Dayley and B. Dayley, *Node.js, MongoDB, and Angular Web Development*, 2nd ed., Indianapolis, IN, USA: Addison-Wesley, 2017.
4. E. Williams, *Node.js Web Development*, 5th ed., Birmingham, UK: Packt Publishing, 2022.
5. W. Eisenman, *MongoDB: The Definitive Guide*, 3rd ed., Sebastopol, CA, USA: O'Reilly Media, 2019.

For MOOCs and other learning Resources

1. https://www.classcentral.com/course/programming-with-javascript-89876
2. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction
3. https://www.freecodecamp.org/news/learn-javascript-free-js-courses-for-beginners/
4. https://learn.microsoft.com/en-us/visualstudio/javascript/javascript-in-visual-studio?view=vs-2022

Double Minor in Essentials of Computer Engineering

ABCXXX: Project

Teaching Scheme: Laboratory: 8 Hours/Week
Total Credits: 4

Syllabus

Aim

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

Project Group and Topic Selection and Synopsis:

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

Overview of the Course:

9. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
10. The project requires the students to conceive, design, implement and operate a mechanism (the design problem). The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts. If the mechanism

incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

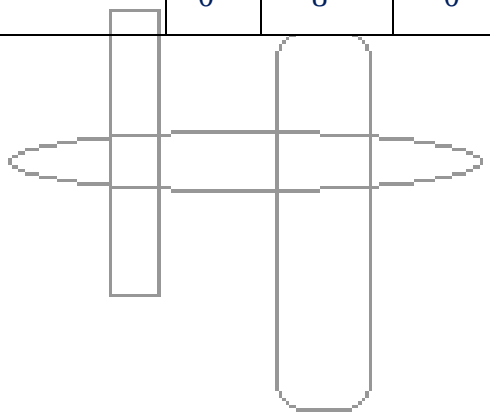
11. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
12. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
13. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
14. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
15. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
16. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

Note:

The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

Double Minor Course: UI/UX Design

Course No.	Semester	Course Name	Teaching Scheme (Hours/Week)			Examination Scheme				Credits
			Th	Lab	Tut				Total	
C1	III	Human Computer Interaction	2	2	0					3
C2	IV	Usability Evaluation Methods	2	2	0					3
C3	V	AR/VR Technology	2	2	0					3
C4	VI	Assistive Technology	2	2	0					3
C5	VII	Human-Centered Artificial Intelligence	2	2	1					4
C6	VIII	Project	0	8	0					4



Double Minor in UI/UX Design

ABCXXXX: Human-Computer Interaction

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 03

Course Objectives:

1. To categorize IT applications based on measurable human factors,
2. To study the user community through user survey and/or field visit,
3. To design user-friendly user interfaces with due consideration of interface theory and principles,
4. To apply usability evaluation methods to identify the usability issues with IT applications,
5. To understand the kind of documentation required for IT applications,
6. To integrate web and mobile app design approaches as per user requirements.

Syllabus
Theory

Unit 1: HCI Fundamentals (5 Hours)

Human-Computer Interaction (HCI); Interdisciplinary Nature; Related Disciplines; Usability; Types of Usability; User Interface (UI); Measurable Human Factors; Accessibility; Differently-abled Users.

Unit 2: Interactions Concepts and Models (5 Hours)

User Persona; User Categorization; Golden Rule of Interface Design, Miller's Principle; Task Analysis-GOMS; Contextual Inquiry; Work Models; Interaction Styles.

Unit 3: Design Process (4 Hours)

Design Concept; Three Pillars of Design; Process of Design; Ethnographics Observations; Participatory Design; Internationalization.

Unit 4: Usability Evaluation (5 Hours)

Expert-based Evaluation; User-based Evaluation; Formative Evaluation; Summative Evaluation; Heuristic Evaluation; Cognitive Walkthrough; Semiotic Analysis; Icon Categorization; User Surveys; Interviews; Usability Testing.

Unit 5: Documentation and Groupware (5 Hours)

Classification of Documents; Reading from Displays; Online Help; Tutorials; Error / Warning Messages; Groupware; Computer Supported Cooperative Work (CSCW); Dimensions of Cooperation; Challenges with Online Communications.

Unit 6: Website and Mobile App Design (4 Hours)

Content Design; Interaction and Navigation Design; Presentation Design; Differences in Design Approaches; Design and Evaluation Tools.

Syllabus
Laboratory

List of Experiments

1. Create User Persona.
2. Perform Task analysis using GOMS.
3. With necessary cultural and ethnographic considerations, design UI.
4. Perform User Survey.
5. Perform Usability Testing.
6. Design Documentation.
7. Apply UI principles to Design UI for Website.
8. Apply UI principles to Design UI for Mobile App.

Syllabus
Course Project

List of Course Projects

1. E-Commerce Website.
2. E-Commerce App.
3. Health care Website.
4. Health care App.
5. Social Messaging Website.
6. Social Messaging App.
7. Food Delivery App.
8. Travel Booking Website.
9. Travel Booking App.
10. Online Word Editor considering CSCW.

Course Outcomes

Course Outcomes: Students will be able to:

1. To appreciate the differences among IT applications and their categories based on measurable human factors.
2. To study the user community through user surveys and/or field visits.
3. To design user-friendly user interfaces as per user requirements and UI design principles.
4. To apply a suitable usability evaluation method to identify the usability issues.
5. To understand the kind of documentation required for IT applications.
6. To enhance UI designs as per desired web or mobile app design approach.

Books and E-Resources

For Reference Print Book -

1. Ben Shneiderman, Catherine Plaisant, “Designing the User Interface”, 2002 Edition, Basic Books, ISBN 100-465-06710-7.
2. Wilbert Galitz, “The Essential Guide to User Interface Design”, Second Edition, Wiley-Dreamtech India (P) Ltd., ISBN 81-265-0280-0.
3. John Carroll, “Human-Computer Interaction in the New Millenium”, Pearson Education, ISBN 81-7808-549-6.

For MOOCs and other learning Resources

1. https://onlinecourses.nptel.ac.in/noc21_ar05/preview
2. Foundations of User Experience (UX) Design | Coursera

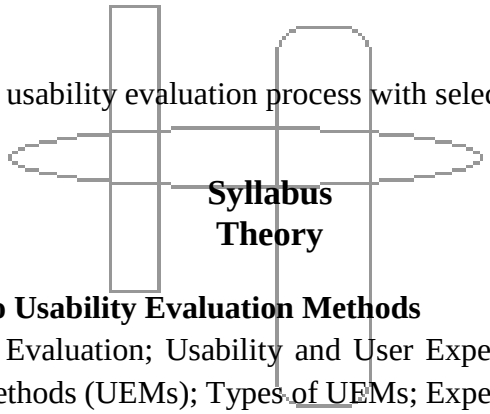
Minor in UI/UX Design

ABCXXXX: Usability Evaluation Methods

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 3

Course Objectives:

1. To understand the usability evaluation process and categorize related evaluation methods,
2. To study the expert-based and summative usability evaluation methods (UEMs),
3. To appreciate the leanings through formative UEMs for improved UI designs
4. To enhance visual UI designs through semiotic analysis of icons and other visual elements,
5. To encourage user involvement in usability evaluation process by applying user-based UEMs,
6. To design a suitable usability evaluation process with selection of appropriate UEMs.



Syllabus
Theory

Unit 1: Introduction to Usability Evaluation Methods (5 Hours)
Definition of Usability Evaluation; Usability and User Experience; Types of Usability; Usability Evaluation Methods (UEMs); Types of UEMs; Expert-based UEMs; User-based UEMs; Formative Evaluation; Summative Evaluation.

Unit 2: Heuristic Evaluation (5 Hours)
Introduction of Heuristic Evaluation (HE); Nielsen's Usability Heuristics; Choosing an Appropriate Set of Heuristics; Selection of Evaluators; Preparing the Design for Evaluation; Evaluating Independently; Prioritizing and Reporting Problems; Design Team Discussion; HE Limitations, Practical Applications and Case Studies.

Unit 3: Cognitive Walkthrough (4 Hours)
Introduction of Cognitive Walkthrough (CW); Preparation of CW; Team Roles and Responsibilities; Step-by-Step Walkthrough; Four Questions asked; Presentation and Summarization of Findings; Issue Identification and Solutions; Practical Applications and Case Studies.

Unit 4: Semiotic Analysis (5 Hours)

Introduction to Semiotics; Icons; Types of Icons; Lexical Analysis; Semantics; Pragmatics; Icon Testing; Testing without Context; Testing with Context; Testing with Alternatives; Acceptance of Icons / Visual elements; Practical Applications and Case Studies.

Unit 5: Usability Testing (5 Hours)

Usability; Users; Demographics of Users; User Categorization; Selection of Users; User Sampling and User Population; Testing Plan; Test Cases; Usability Testing; Findings; Issue Identification; Practical Applications and Case Studies.

Unit 6: Other Evaluation Methods (4 Hours)

Lab-based Evaluation Methods (e.g., Observations, Eye-Tracking, Think aloud, Field Studies; 'In the Wild' Evaluations, Usage Data / Log Analysis. Practical Applications and Case Studies.

Syllabus
Laboratory

List of Experiments

1. Perform a study of selection of UEM for given UI.
2. Perform Heuristic Evaluation of a website.
3. Perform Cognitive Walkthrough for a mobile app.
4. Perform Semiotic Analysis of icons present in a mobile app.
5. Perform Semiotic Analysis of icons present in a website.
6. Perform Usability Testing for a mobile app.
7. Perform Usability Testing of a website.
8. Perform Lab-Based Evaluation for a mobile app.

Syllabus

Course Project

List of Course Projects

1. Redesign an E-Commerce Website based on heuristic evaluation.
2. Redesign an E-Commerce Website based on cognitive walkthough.
3. Compare the E-Commerce Websites using a heuristic evaluation.
4. Redesign a Food Delivery App based on heuristic evaluation.

5. Redesign a Food Delivery App based on cognitive walkthough.
6. Compare the Food Delivery Apps using a heuristic evaluation.
7. Redesign a Travel Booking App based on heuristic evaluation & Semiotic analysis.
8. Redesign a Travel Booking App based on cognitive walkthough.
9. Compare the Travel Booking Apps using a heuristic evaluation.

Course Outcomes

Course Outcomes: Students will be able to:

1. To understand and explain key usability concepts, evaluation methods, and their applications in formative and summative evaluations.
2. To conduct Heuristic Evaluations using established heuristics and report usability issues for iterative design improvements.
3. To perform Cognitive Walkthroughs to identify and document usability issues through structured user analysis.
4. To analyze and evaluate icons and visual elements using Semiotic Analysis for improved user communication.
5. To design and execute usability testing by planning, user selection, testing, and interpreting findings to enhance user experience.
6. To compare and apply various lab-based and field-based evaluation methods to assess usability in different contexts.

Books and E-Resources

For Reference Print Book -

1. Ben Shneiderman, Catherine Plaisant, “Designing the User Interface”, 2002 Edition, Basic Books, ISBN 100-465-06710-7.
2. Jakob Nielsen, Robert Mack, “Usability Inspection Methods”, John Wiley & Sons Inc., ISBN 978-0471018773.
3. Ana Martins, Alexandra Queirós, Anabela Silva, Nelson Pacheco Rocha, “Usability Evaluation In Industry”, CRC Press, ISBN 9780748404605.

For MOOCs and other learning Resources

1. https://onlinecourses.nptel.ac.in/noc21_ar05/preview
2. Foundations of User Experience (UX) Design | Coursera

Double Minor in UI/UX Design

ABC####: AR / VR Technology

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 03

Course Objectives:

1. Define VR/AR and understand their evolution and core components.
2. Identify AR/VR hardware and sensory systems.
3. Apply rendering and transformation techniques in virtual environments.
4. Explore computer vision and tracking methods in AR.
5. Understand motion, interaction, and audio in AR/VR systems.
6. Recognize mixed reality applications and current trends.

Syllabus

Theory

Unit 1: Introduction to Virtual Reality and Augmented Reality (4 Hours)

Defining Virtual Reality and Augmented Reality; History and Evolution of VR and AR; Human Physiology and Perception relevant to VR and AR; Key Elements of VR and AR Experiences. VR and AR Systems: Inputs; Outputs (Visual, Aural, Haptic); Applications of VR and AR.

Unit 2: Hardware Components of AR and VR Systems (5 Hours)

Displays in AR and VR: Audio; Haptic; Visual, and Other Sensory Displays.
Visual Perception: Requirements and Characteristics Spatial Display Models.
Processors: Architecture and Specifications.
Tracking and Sensors: Calibration; Registration; Stationary and Mobile Tracking; Sensor Fusion.

Unit 3: Representing and Rendering the Virtual & Augmented Worlds(5 Hours)

Geometric Models and Transformations in VR: Position; Orientation; Axis-Angle Rotation; Viewing Transformations.
Human Vision and Eye Movements: Implications for VR and AR.
Visual Perception: Depth; Motion; Color; Combining Sources.
Visual Rendering Techniques: Ray Tracing; Shading Models; Rasterization; Optical Distortion Correction; Latency and Frame Rate Improvement.

Unit 4: Computer Vision, Tracking and AR Techniques (5 Hours)

Computer Vision for AR: Marker Tracking; Infrared Multi-Camera Tracking; Natural Feature Tracking; Outdoor Tracking; SLAM; AR Software Components and Content Creation Tools.

Marker-Based Tracking: Marker Types; Camera Pose Estimation; Visual Tracking; Mathematical Models.

Markerless Tracking: Localization-Based Augmentation and Real-World Examples.

Tracking Methods: Visual; Feature-Based; Hybrid; Initialization and Recovery.

Unit 5: Motion, Interaction & Devices in AR/VR (5 Hours)

Motion in Real and Virtual Worlds.

Interaction: Motor Programs; Locomotion; Manipulation; Social Interaction.

Audio: Physics of Sound; Human Hearing; Auditory Perception and Rendering; AR/VR Devices; AR Components.

Unit 6: Applications and Current Trends in Robot Learning (4 Hours)

Introduction to Mixed Reality; Applications of Mixed Reality; Input and Output in Mixed reality; Computer Vision and Mixed Reality.

Syllabus
Laboratory

List of Experiments

1. Build a simple VR environment with visual and audio components.
2. Develop AR app with Unity and C# including marker-based tracking.
3. Implement haptic feedback simulation using GHOST toolkit or similar.
4. Implement natural feature tracking and SLAM-based localization.
5. Sensor calibration and fusion experiments for tracking accuracy.
6. Experiment with motion, interaction, and audio rendering in VR.
7. Create simple mixed reality scenarios using available toolkits.
8. Marker enhancement and visual tracking using Vuforia Engine.
9. Simulate latency and frame rate optimization in rendering.
10. Mini-project integrating AR/VR interaction and tracking (e.g., virtual museum tour or AR navigation).
11. Develop a VR Ball Game. The scene should contain a play area surrounded by four walls and a ball that acts as a player. The objective of the game is to keep the ball rolling without colliding with the walls. If it collides with either of the walls, the

wall color should change and a text should display on the screen indicating the collision.

12. Develop a VR Golf Game. The scene should contain a play area (golf course), which consists of a series of cups/holes each having different scores. Display the score card.
13. Develop a VR game in Unity such that on each gun trigger click, destroy the cubes placed on the plane and gain a score point . Make a score UI and display it on the screen
14. Develop a VR Basketball Game. The scene should contain a basketball court. The developed game should be a single player game. The objective of the game is to let the player put the ball in the basket maximum number of times. Display the score card.
15. Develop an AR bowling game with one image target .The image target should include 3d models as per requirement.
16. Write a c# program to develop a score point system for bowling games. Build an apk. (Note : Vuforia plugin should be installed in unity.)
17. Develop a VR environment for flying helicopter/moving car simulation.

Syllabus
Course Project

List of Course Projects

1. VR training simulation for astronauts.
2. AR navigation for indoor environments with markerless tracking.
3. Mixed Reality collaborative workspace.
4. Deep learning-enhanced gesture interaction in AR/VR.
5. SLAM-based AR game with outdoor tracking.
6. Haptic-enabled VR educational game.
7. AR application for industrial maintenance using marker tracking.
8. Real-time multi-sensory VR storytelling experience.
9. Adaptive audio rendering in VR environments.
10. Sim2Real transfer in AR robotics applications.

Course Outcomes

Course Outcomes: Students will be able to:

1. To understand foundational principles and history of AR and VR.
2. To identify hardware and software components of AR/VR systems.
3. To apply geometric modeling, perception, and rendering techniques.
4. To use computer vision and tracking algorithms for AR applications.
5. To design and implement interactive experiences in AR/VR environments.
6. To explore advanced MR concepts and implement SLAM variants.

Books and E-Resources

For Reference Print Book -

1. Virtual Reality, Steven M. LaValle, Cambridge University Press, 2016
2. Understanding Virtual Reality: Interface, Application and Design, William R Sherman and Alan B Craig, (The Morgan Kaufmann Series in Computer Graphics)". Morgan Kaufmann Publishers, San Francisco, CA, 2002
3. Developing Virtual Reality Applications: Foundations of Effective Design, Alan B Craig, William R Sherman and Jeffrey D Will, Morgan Kaufmann, 2009.
4. Gerard Jounghyun Kim, "Designing Virtual Systems: The Structured Approach", 2005.
5. Doug A Bowman, Ernest Kuijff, Joseph J LaViola, Jr and Ivan Poupyrev, "3D User Interfaces, Theory and Practice", Addison Wesley, USA, 2005.
6. Designing for Mixed Reality, Kharis O'Connell Published by O'Reilly Media, Inc., 2016, ISBN: 9781491962381

For MOOCs and other learning Resources

1. https://nptel.ac.in/courses/106/106/106106138/
2. https://www.coursera.org/learn/introduction-virtual-reality
3. https://www.coursera.org/learn/ar
4. https://www.udemy.com/share/101XPi/

Double Minor in UI/UX Design

ABCXXXX: Assistive Technology

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 03

Course Objectives:

1. To appreciate the development of assistive technologies for diversified users needing assistance,
2. To understand the accessibility and the process of developing assistive technology,
3. To study different types of assistive technologies developed for diversified users needing assistance,
4. To integrate universal design principles and related framework in development process of assistive technology,
5. To apply various recommended standards and guidelines with assistive technology,
6. To encourage an enhancement of hands-on skills through implementation of innovative projects related to assistive technologies.

Syllabus
Theory

Unit 1: Introduction to Assistive Technology (5 Hours)

Definitions related with Assistive Technology (AT), Accessibility Demographics, Types of Users needing Assistance, Rehabilitation Goals, Perceptions, Challenges, Social Correctness, Universal Design, Types and Levels of Assistive technology.

Unit 2: Understanding Accessibility (5 Hours)

Design Thinking approach, Building empathy with the user and his/her situation, User Needs, Interviews, Observation with Real Users of Assistive Technologies, Ideation, Prototyping, Iteration to Explore, Formulate and Test Solutions.

Unit 3: Assistive Technologies used (4 Hours)

Conversational User Interfaces, Brain Computer Interfaces, IoT Devices, Assistive Robotics, Mobile Assistive Applications, Practical Applications and Case Studies, Commercial Products, Research Approach.

Unit 4: Universal Design & Its Framework (5 Hours)

Definition of Universal Design, Principles of Universal Design - Equitable Use, Tolerance for Error, Low Physical Effort, Multiple Means of Engagement, Representation, Action and Expression, Perceivable, Operable and Robust Technology.

Unit 5: Standards and Guidelines (5 Hours)

Assistive Technology Standards Board (ATSB), ESNA's Assistive Technology Standards, ISO/TC 173 & ISO 9999:2022 – Standards for Assistive Products for Persons with Disabilities, National List of Essential Assistive Products (NLEAP), National Medical Device Policy 2023, Android Accessibility Best Practices 2024, Guidelines for Assistive Technology Devices and Services for Children With Disabilities in USA.

Unit 6: AT Commercial Products and Projects (4 Hours)

Sample Popular AT Products, Literature Review of AT Projects, Leading Mobile Apps for Assistive Technologies.

Syllabus

Laboratory

List of Experiments

1. Compare multiple available assistive applications / products for hearing-impaired users.
2. Conduct literature survey for available assistive technologies for visually-impaired users.
3. Prepare a SRS document for an assistive application prototype for a specific type of users.
4. Construct UML diagrams for a specific assistive application for non-English speaking Indian users.
5. Perform an usability evaluation of an application or product with assistive technology.
6. Enlist required assistive guidelines / standards to be incorporated for compliance of a specific assistive technology.
7. Enlist applicable Universal Design Principles for effective usage of a specific assistive technology.

Syllabus
Course Project

List of Course Projects

1. Design a mobile app to assist visually-impaired users.
2. Design an ICT app to assist hearing-impaired users.
3. Design an ICT app to assist older citizens.
4. Design an ICT app to assist color-blind users.
5. Design an application prototype for non-English speaking Indian users.
6. Design a mobile app to assist patients facing mental challenges.
7. Design a mobile app to assist technology-deprived people of remote rural India.
8. Incorporate relevant accessibility guidelines / standards into assistive applications / products for differently-abled users.

Course Outcomes

Course Outcomes: Students will be able:

1. To develop the assistive technologies for diversified users needing assistance,
2. To understand the concept of accessibility and the process of developing assistive technology,
3. To analyze different types of assistive technologies developed for diversified users needing assistance,
4. To implement assistive technologies with universal design principles and related framework,
5. To incorporate relevant standards and guidelines in assistive technology development,
6. To explore and innovate in assistive technologies.

Books and E-Resources

For Reference Print Book -

1. Ben Shneiderman, Catherine Plaisant, “Designing the User Interface”, 2002 Edition, Basic Books, ISBN 100-465-06710-7.
2. Wilbert Galitz, “The Essential Guide to User Interface Design”, Second Edition, Wiley-Dreamtech India (P) Ltd., ISBN 81-265-0280-0.
3. John Carroll, “Human-Computer Interaction in the New Millenium”, Pearson Education, ISBN 81-7808-549-6.

For MOOCs and other learning Resources

1. https://onlinecourses.nptel.ac.in/noc21_ar05/preview
2. Foundations of User Experience (UX) Design | Coursera

Double Minor in UI/UX Design

ABCXXXX: Human-Centered Artificial Intelligence

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 03

Course Objectives:

1. To introduce the concepts related with Human-Centred Artificial Intelligence (HCAI),
2. To study responsible AI principles for human well-being,
3. To understand the concepts of explainable AI and methods of achieving explainability,
4. To apply the Algorithmic Fairness in development of ICT applications,
5. To follow and incorporate AI ethics in application development process,
6. To familiarize with vital research methods in Human-Centred Computing and AI.

Syllabus
Theory

Unit 1: Introduction to HCAI (5 Hours)

Definitions related with Human-Centred Artificial Intelligence (HCAI); HCAI Principles; Prioritization of Human Needs; Values; and Well-being; Human Capabilities; Augmented AI; Collaboration; Trust; Inclusive Design.

Unit 2: Responsible AI Principles (5 Hours)

Fairness; Reliability and Safety; Privacy and Security; Transparency; Accountability; Inclusiveness; Human Agency and Oversight; Societal and Environmental Well-being.

Unit 3: Explainable AI (4 Hours)

Key Concepts: Transparency, Interpretability, Explainability, Confidence, Bias Detection, Model Improvement, Regulatory Compliance, Enhanced Decision-Making,
Methods for Achieving Explainability: White Box Models, Black Box Explanations, Feature Attribution, Visualization.

Unit 4: Algorithmic Fairness (5 Hours)

Ethical Considerations, Legal Compliance, Building Trust, Promoting Equity, Bias in Training Data, Algorithmic Predisposition, Measuring Fairness, Examples of Algorithmic Fairness in Action - Loan Approval, Recruiting and Criminal Justice.

Unit 5: AI Ethics (5 Hours)

AI Ethics, Guiding Principles, Interdisciplinary Field, Principles of AI Ethics – Fairness, Transparency, Safety and Security, Human Rights, Accountability, Beneficence, Autonomy and Freedom, Respect for Persons, Justice, Reducing Risks, Promoting Societal Well-being.

Unit 6: Research Methods for Human-Centered Computing and AI (4 Hours)

Data Collection and Analysis, Automated Data Handling, Machine Learning, Natural Language Processing (NLP), Image and Video Analysis, Semantic Analysis, Literature Synthesis, Identifying Research Gaps, Findings Identification and Discussion.

Syllabus
Laboratory

List of Experiments

1. Explore the personal traits for selected users, using LLM models.
2. Perform a literature review about human autonomy through human-centred AI.
3. Investigate the impacts of auditory and visual feedback in a specific system, designed for users such as drivers.
4. Validate the algorithmic fairness of any recommendation system.
5. Estimate the relevance of AI ethics for a given expert system.
6. Compare two different research methods for implementation of a given decision support system.

Syllabus
Course Project

List of Course Projects

1. Recommendation System for Currency Notes,
2. Expert System for Career Guidance,
3. Decision Support System for Medical Diagnosis,
4. Recommendation System for Cancer Detection,
5. Expert System for Bird Identification,
6. Shortest Path Selection to Desired Destination,

7. Decision Support System for Inventory Management,
8. Recommendation System for International Vacation.

Course Outcomes

Course Outcomes: Students will be able

1. To apply the concepts related with Human-Centred Artificial Intelligence (HCAI),
2. To incorporate responsible AI principles for human well-being,
3. To utilize the concepts of explainable AI and methods of achieving explainability,
4. To design AI applications with Algorithmic Fairness,
5. To maintain the application development process compliance with AI ethics,
6. To encourage innovation by employing vital research methods in Human-Centred Computing and AI.

Books and E-Resources

For Reference Print Book -

1. Ben Shneiderman, Catherine Plaisant, "Designing the User Interface", 2002 Edition, Basic Books, ISBN 100-465-06710-7.
2. Wilbert Galitz, "The Essential Guide to User Interface Design", Second Edition, Wiley-Dreamtech India (P) Ltd., ISBN 81-265-0280-0.
3. John Carroll, "Human-Computer Interaction in the New Millenium", Pearson Education, ISBN 81-7808-549-6.

For MOOCs and other learning Resources

1. https://onlinecourses.nptel.ac.in/noc21_ar05/preview
2. Foundations of User Experience (UX) Design | Coursera

Minor in UI/UX Design

ABCXXXX: Project

Teaching Scheme: Laboratory: 8 Hours/Week

Total Credits: 4

Syllabus

Aim

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

Project Group and Topic Selection and Synopsis:

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

Overview of the Course:

17. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
18. The project requires the students to conceive, design, implement and operate a mechanism (the design problem). The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts. If the mechanism

incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

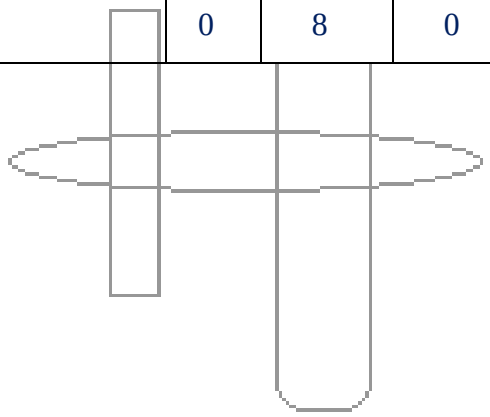
19. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
20. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
21. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
22. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
23. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
24. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

Note:

The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

Double Minor Course: Cloud Computing

Course No.	Semester	Course Name	Teaching Scheme (Hours/Week)			Examination Scheme				Credits
			Th	Lab	Tut				Total	
C1	III	Cloud Fundamentals	2	2	0					3
C2	IV	Cloud Infrastructure and Architecture	2	2	0					3
C3	V	Data Storage and Management in the Cloud	2	2	0					3
C4	VI	DevOPs for Cloud	2	2	0					3
C5	VII	Cloud Administration	2	2	1					4
C6	VIII	Project	0	8	0					4



Double Minor in Cloud Computing

ABCXXX: Cloud Fundamentals

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 3

Prerequisites: Computer Network, Programming Fundaments

Course Objectives

1. To introduce the basic concepts and evolution of cloud computing and its role in modern IT infrastructure.
2. To understand different cloud service models (IaaS, PaaS, SaaS) and deployment models (Public, Private, Hybrid, Community).
3. To explore the architecture, components, and key technologies behind cloud platforms and virtualization.
4. To provide hands-on experience with leading cloud providers (e.g., AWS, Azure, Google Cloud) for deploying and managing services.
5. To examine scalability, elasticity, and resource provisioning in cloud environments.
6. To highlight the challenges and best practices related to cloud cost management, security, and compliance.

Syllabus

Theory

Unit 1: Introduction to Cloud Computing (4 Hours)

Definition and essential characteristics,History and evolution of cloud computing,Cloud service models: IaaS, PaaS, SaaS,Cloud deployment models: Public, Private, Hybrid, Community,Benefits and challenges

Unit 2: Cloud Architecture and Technologies (4 Hours)

Cloud reference architecture,Virtualization: Concepts and types,Hypervisors: Type I and Type II,Resource pooling, multi-tenancy, scalability,Service-oriented architecture (SOA)

Unit 3: Cloud Service Providers and Platforms (5 Hours)

Overview of AWS, Microsoft Azure, Google Cloud Platform,Account setup and free tier services,Cloud consoles and CLI tools,Service comparison across providers,Case studies of real-world usage

Unit 4: Cloud Storage and Databases (5 Hours)

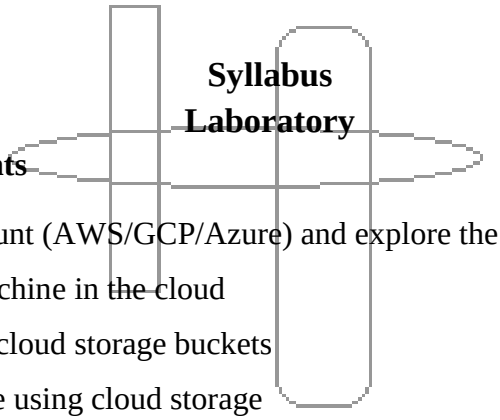
Types of cloud storage: Object, Block, File,Cloud storage services: S3, Blob, Drive,Cloud databases: RDS, Firebase, Cosmos DB,Data consistency, availability, and partition tolerance (CAP),Backup and recovery strategies

Unit 5: Security and Compliance in the Cloud (5 Hours)

Cloud security fundamentals,Data encryption and identity management,Shared responsibility model,Compliance standards: GDPR, HIPAA, ISO,Cloud risk management and governance

Unit 6: Cloud Trends and Emerging Technologies (5 Hours)

Edge computing and fog computing,Serverless and containerization overview,Green cloud,computing,Cloud automation and orchestration,Future of cloud and job roles in the cloud domain



List of Lab Experiments

1. Create a cloud account (AWS/GCP/Azure) and explore the dashboard
2. Deploy a virtual machine in the cloud
3. Create and manage cloud storage buckets
4. Host a static website using cloud storage
5. Create and test a cloud database instance
6. Setup auto-scaling for a cloud service
7. Use CLI to deploy and manage services
8. Monitor usage and costs in cloud platform
9. Configure identity and access management (IAM)
10. Use cloud load balancer for traffic distribution
11. Create backup and recovery policies
12. Secure cloud data using encryption techniques

Syllabus
Course Projects

List of Course Projects

1. Build and deploy a personal portfolio website using cloud services
2. Set up a cloud-hosted blog platform
3. Develop a simple IoT application storing data in the cloud
4. Implement an online voting system on cloud
5. Cloud-based file sharing and storage application
6. Create a secure note-taking application using cloud databases

Course Outcomes

Course Outcomes: Students will be able to

1. Understand the fundamentals and architecture of cloud computing.
2. Analyze cloud service and deployment models.
3. Demonstrate usage of major cloud service providers and platforms.
4. Implement basic cloud services including VMs, storage, and databases.
5. Evaluate security risks and compliance aspects in cloud systems.
6. Explore cloud trends and apply concepts to real-world projects.

Books and E-Resources

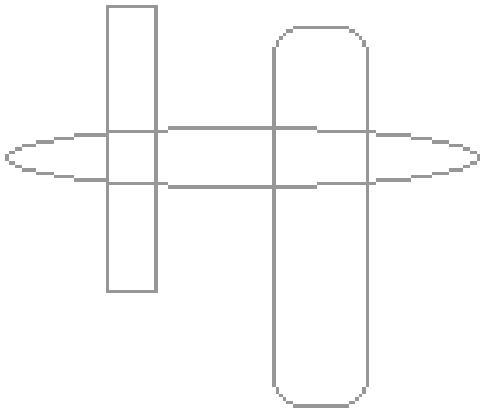
Textbooks References

1. Cloud Computing: Concepts, Technology & Architecture *Author:* Thomas Erl, Zaigham Mahmood, Ricardo Puttini *Publisher:* Prentice Hall
2. Mastering Cloud Computing *Author:* Rajkumar Buyya, Christian Vecchiola, Thamarai Selvi *Publisher:* McGraw Hill
3. Cloud Computing Bible *Author:* Barrie Sosinsky *Publisher:* Wiley *ISBN:* 978-0470903563
4. Cloud Computing: A Hands-On Approach *Author:* Arshdeep Bahga, Vijay Madisetti *Publisher:* University Press *ISBN:* 978-0996025515

MOOC References

1. Introduction to Cloud Computing – IBM (Coursera)
2. Introduction to Cloud Computing – Linux Foundation (edX)
3. AWS Cloud Practitioner Essentials – AWS Training

4. Google Cloud Digital Leader Training – Google Cloud
5. Learning Cloud Computing – LinkedIn Learning
6. Cloud Computing Basics – University of Illinois (Coursera)



Double Minor in Cloud Computing

ABCXXX: Cloud Infrastructure and Architecture

Teaching Scheme
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

Prerequisites: Computer Network, Computer Architecture, Operating System

Syllabus
Theory

Unit 1: Introduction to Cloud Infrastructure (4 Hours)
Definition of cloud infrastructure, Physical and virtual infrastructure, Cloud data centers and resource provisioning, Infrastructure as a Service (IaaS) overview, Cloud computing reference model

Unit 2: Virtualization and Resource Management (6 Hours)
Hypervisors and virtualization technologies, Virtual machines, containers, and serverless functions, Resource allocation and scheduling, Elasticity and scalability, Orchestration tools: Kubernetes, Docker Swarm

Unit 3: Cloud Networking and Storage (5 Hours)
Cloud network architecture, Virtual Private Cloud (VPC) and Subnetting, Load balancers, gateways, and firewalls, Cloud storage services: block, file, and object storage, Distributed storage systems and CDN

Unit 4: High Availability and Disaster Recovery (5 Hours)
Fault tolerance and resilience in cloud infrastructure, Replication strategies and backup systems, High Availability (HA) design patterns, Disaster recovery planning and implementation, Monitoring and alerting mechanisms

Unit 5: Infrastructure Automation and DevOps (4 Hours)
Infrastructure as Code (IaC) concepts, Tools: Terraform, Ansible, CloudFormation, CI/CD pipelines for cloud infrastructure, Automation workflows and monitoring tools, DevOps best practices in cloud environments

Unit 6: Cloud Architecture Patterns and Trends (4 Hours)
Architecting for the cloud: principles and patterns, Microservices and serverless architecture

Multi-cloud and hybrid cloud environments, Security architecture in cloud infrastructure
Future trends: AI Ops, Edge Computing

Syllabus
Laboratory

List of Lab Experiments

1. Setup and explore a Virtual Private Cloud (VPC) on AWS/Azure
2. Deploy virtual machines using IaaS services
3. Configure and test a load balancer with multiple instances
4. Use Docker to containerize a sample application
5. Orchestrate containers with Kubernetes
6. Automate VM creation using Terraform
7. Create storage buckets and manage files
8. Backup and restore cloud storage data
9. Implement high availability with multiple zones
10. Monitor cloud resources using built-in tools (CloudWatch, Azure Monitor)
11. Create and configure a CI/CD pipeline for infrastructure
12. Use Ansible for configuration management in cloud VMs

Syllabus
Course Projects

List of Course Projects

1. Design a secure cloud infrastructure for a small enterprise
2. Deploy a containerized web app using Kubernetes
3. Implement a hybrid cloud model for data synchronization
4. Develop a scalable microservices application with Docker and cloud VMs
5. Create an HA architecture using AWS or Azure services
6. Build and automate infrastructure using Terraform
7. Disaster recovery system for a cloud-based database
8. Multi-region application deployment

Course Outcomes

Course Outcomes: Students will be able to

1. Understand and explain the key components of cloud infrastructure.
2. Deploy and manage virtual resources and networks in a cloud environment.
3. Implement storage and compute services using cloud platforms.
4. Design for high availability, fault tolerance, and disaster recovery.
5. Automate infrastructure management using modern DevOps tools.
6. Apply architectural principles and trends to build scalable cloud solutions.

Books and E-Resources

Text Book References

1. Mastering Cloud Computing by Rajkumar Buyya, Christian Vecchiola, Thamarai Selvi Publisher: McGraw Hill Education
2. Cloud Computing: Concepts, Technology & Architecture by Thomas Erl, Zaigham Mahmood, Ricardo Puttini Publisher: Prentice Hall
3. Cloud Computing: A Practical Approach by Toby Velte, Anthony Velte, Robert Elsenpeter Publisher: McGraw Hill
4. Architecting the Cloud: Design Decisions for Cloud Computing Service Models by Michael J. Kavis Publisher: Wiley

MOOC References

1. Cloud Computing Specialization – University of Illinois (Coursera)
   Covers cloud fundamentals, architecture, and infrastructure.
   https://www.coursera.org/specializations/cloud-computing
2. Architecting with Google Cloud Platform – Google (Coursera)
   Focused on designing scalable and reliable applications on GCP.
   https://www.coursera.org/specializations/gcp-architecture
3. AWS Cloud Technical Essentials – Amazon Web Services (Coursera)
   Basic understanding of AWS infrastructure and cloud architecture.
   https://www.coursera.org/learn/aws-cloud-technical-essentials
4. Microsoft Azure Fundamentals (AZ-900) – edX/Microsoft
   Introductory course to Azure cloud services and architecture.
   https://learn.microsoft.com/en-us/training/paths/azure-fundamentals/
5. Cloud DevOps Engineer Nanodegree – Udacity
   Advanced cloud and infrastructure design with CI/CD integration.
   https://www.udacity.com/course/cloud-dev-ops-nanodegree--nd9991

Double Minor in Cloud Computing

ABCXXX: Data Storage and Management in the Cloud

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

Prerequisites: Computer Architecture, Operating System, DBMS, Computer Network

Course Objectives

1. Introduce the fundamental concepts of data storage technologies and cloud-based storage architectures.
2. Explore various types of cloud storage systems, including object storage, block storage, and file storage.
3. Understand data consistency, availability, durability, and replication strategies in cloud environments.
4. Develop the ability to manage and provision storage resources using cloud service providers (e.g., AWS S3, Azure Blob, Google Cloud Storage).
5. Examine data lifecycle management, tiered storage, and storage cost optimization techniques in the cloud.
6. Implement data security, privacy, backup, and disaster recovery mechanisms for cloud-based storage solutions.

Syllabus

Theory

Unit 1: Introduction to Cloud Storage (4 Hours)

Overview of cloud storage concepts, Types: Object, Block, File storage, Differences from traditional storage, Major cloud providers (AWS, Azure, GCP), Data lifecycle and management

Unit 2: Cloud Storage Architectures (4 Hours)

Distributed storage systems, Storage infrastructure and data centres, Storage access protocols (NFS, iSCSI, SMB), Storage tiering and scalability, Performance tuning in storage

Unit 3: Data Management and Governance (6 Hours)
Metadata and indexing, Data replication and consistency models, Version control and data integrity, Role-based access and auditing, Data lifecycle policies

Unit 4: Storage Security and Compliance (4 Hours)
Encryption at rest and in transit, Key management practices (KMS),Secure data deletion, Privacy regulations: GDPR, HIPAA, Backup integrity and compliance

Unit 5: Backup, Recovery, and High Availability (6 Hours)
Snapshotting and versioning**,** Replication and geo-redundancy**,** Backup automation strategies**,** Disaster Recovery as a Service (DRaaS)**,** High availability architecture

Unit 6: Trends and Emerging Technologies (4 Hours)
AI/ML in data management, Serverless storage, Storage for edge and IoT, Hybrid and multi-cloud data storage, Cost optimization and billing strategies

Syllabus
Laboratory

List of Lab Experiments

1. Setup an AWS S3 bucket and configure versioning
2. Connect Azure File storage to a VM
3. Use lifecycle policies for data archival
4. Create and test snapshot backups
5. Implement IAM access control on storage
6. Encrypt data and manage keys with KMS
7. Perform cross-region replication
8. Configure GCP Storage Transfer Service
9. Monitor storage performance and usage
10. Automate storage with scripts (Boto3, Azure CLI)
11. Integrate cloud storage with web app
12. Compare storage costs using provider calculators

Syllabus
Course Projects

List of Course Projects

1. Design and deploy a secure document storage system
2. Implement a backup and disaster recovery solution
3. Build a version-controlled data store
4. Create a hybrid cloud storage sync tool
5. Real-time data replication system
6. Deploy a file sharing web app using cloud storage
7. Automate archival of log files
8. Secure image repository using KMS
9. Compliance checker for stored data
10. Develop a cost monitoring dashboard

Course Outcomes (COs)

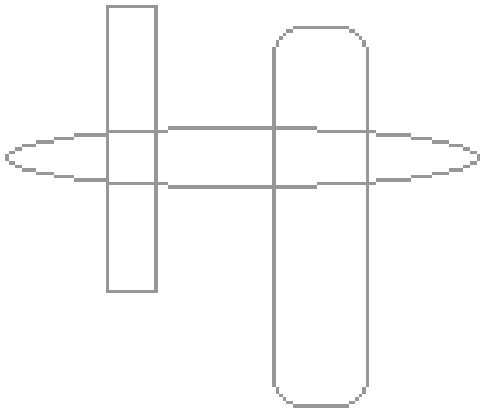
1. Understand the foundational principles of cloud-based data storage.
2. Analyze and select appropriate storage architectures for various applications.
3. Design and manage secure, compliant, and resilient storage systems.
4. Apply data lifecycle and governance strategies in cloud environments.
5. Integrate cloud storage with modern applications and automate management tasks.
6. Explore and evaluate emerging trends and tools in cloud data management.

Textbooks References

1. Cloud Storage Forensics by Darren Quick Publisher: Syngress (Elsevier)
2. Cloud Computing: Concepts, Technology & Architecture by Thomas Erl, Zaigham Mahmood, Ricardo Puttini Publisher: Prentice Hall
3. Storage Networks Explained: Modern Data Storage Technologies by Ulf Troppens, Rainer Erkens, Wolfgang Müller Publisher: Wiley
4. Cloud Data Management by Divyakant Agrawal, Amr El Abbadi, et al.Publisher: Morgan & Claypool Publishers

MOOC References

1. **Cloud Storage and Databases – University of Illinois (Coursera)**
   https://www.coursera.org/learn/cloud-storage
2. **AWS Cloud Practitioner Essentials – Amazon Web Services (Coursera)**
   https://www.coursera.org/learn/aws-cloud-practitioner-essentials
3. **Introduction to Cloud Storage with Google Cloud – Coursera**
   https://www.coursera.org/learn/introduction-to-cloud-storage
4. **Azure Storage and Data Management – Microsoft Learn (Self-paced)**
   https://learn.microsoft.com/en-us/training/paths/store-data-with-azure/
5. **Cloud Data Engineering – Udacity Nanodegree**
   https://www.udacity.com/course/cloud-data-engineer-nanodegree--nd027



Double Minor in Cloud Computing

ABCXXX: DevOPs for Cloud

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week; Tutorials: 1 Hour / Week
Total Credits: 4

Course Prerequisites:
Basics of Operating Systems, Cloud Computing Fundamentals, and Software Development Life Cycle (SDLC).

Course Objectives:

1. Understand DevOps culture and practices for modern cloud-native application development.
2. Use CI/CD tools and cloud platforms to automate software delivery pipelines.
3. Deploy and manage containerized applications in the cloud.
4. Apply Infrastructure as Code (IaC) using modern tools.
5. Integrate monitoring and security into DevOps workflows.
6. Demonstrate DevOps pipeline implementation using popular cloud services.

Course Relevance:
The DevOps for Cloud course is highly relevant in today's rapidly evolving IT landscape, where continuous integration, automation, and scalability are critical for delivering modern cloud-native applications. By integrating DevOps practices with cloud platforms, this course equips students with practical skills in CI/CD, containerization, infrastructure automation, and monitoring—essential for careers in cloud engineering, DevOps, and site reliability engineering. It bridges the gap between development and operations, fostering agility, efficiency, and collaboration in real-world software delivery.

Section 1: Topics/Contents

Unit 1: Introduction to DevOps and Cloud Integration (4 hours)

DevOps concepts: culture, lifecycle, benefits, DevOps vs traditional SDLC, Cloud-native, DevOps overview, Tools: Git, Jenkins, Docker, Kubernetes, Terraform, Ansible, Cloud Providers: AWS, Azure, GCP overview

Unit 2: Version Control and Continuous Integration (4 hours)

Git workflows (branching, pull requests, hooks), Jenkins/GitHub Actions/GitLab CI, Building pipelines with CI tools, Automating build and test cycles

Unit 3: Containers and Orchestration in the Cloud (6 hours)

Docker fundamentals: images, containers, volumes, Dockerfile, Kubernetes basics: pods, services, deployments, Kubernetes on AWS EKS / GCP GKE / Azure AKS, Helm charts and configuration management

Section 2: Topics/Contents

Unit 4: Infrastructure as Code (IaC) (4 hours)

Introduction to IaC and benefits, Terraform syntax, modules, state management, AWS CloudFormation basics, Provisioning cloud infrastructure (EC2, VPC, S3) using IaC

Unit 5: Continuous Delivery and Deployment (4 hours)

Blue-Green and Canary Deployments, CD tools and cloud integrations, Implementing CI/CD pipelines to deploy on AWS/GCP, Managing rollbacks and zero-downtime releases

Unit 6: Monitoring, Logging, and Security in DevOps (6 hours)

Monitoring tools: Prometheus, Grafana, ELK stack, Cloud-native monitoring (AWS CloudWatch, Azure Monitor), DevSecOps and SAST/DAST tools, Secrets management and identity in cloud DevOps

Syllabus
Laboratory

List of Lab Assignments:

1. Set up Git and GitHub repositories; demonstrate branching, merging, and pull request workflow for a sample project.
2. Install and configure Jenkins on a local or cloud VM. Create a basic freestyle project with build and test steps.
3. Create a Jenkins pipeline that automatically builds and tests code on every GitHub commit using a Jenkinsfile.
4. Set up GitHub Actions to automate testing and linting for a Python/Node.js project.
5. Build a Docker image for a simple web application and run it as a container. Push the image to Docker Hub.
6. Deploy a multi-container application using Docker Compose. Verify inter-container communication.
7. Deploy a containerized app to Kubernetes using Minikube or AWS EKS. Implement a rolling update strategy.
8. Write and execute Terraform scripts to provision a basic AWS infrastructure (e.g., EC2 instance, S3 bucket).
9. Use AWS CloudFormation to deploy a serverless application with Lambda and API Gateway.
10. Implement a blue-green deployment pipeline using Jenkins and deploy a versioned web app to AWS.
11. Automate end-to-end CI/CD deployment of a microservice using GitHub Actions to AWS/GCP.
12. Configure Prometheus and Grafana to monitor resource usage (CPU, memory) of containerized apps deployed in Kubernetes.

Syllabus
Course Projects

List of Course Projects:

1. Design and implement a DevOps workflow for a cloud-native application, integrating Git version control, Jenkins, and Docker for an automated build-test pipeline.
2. Develop a case study comparing traditional software development practices with DevOps in cloud environments using AWS, Azure, or GCP.
3. Create a CI pipeline using Jenkins and GitHub that includes automated code testing, linting, and notification setup via Slack/Email.
4. Develop a multi-branch CI workflow using GitHub Actions that supports parallel builds and matrix testing for multiple environments.
5. Containerize a legacy monolithic application using Docker and migrate it to a Kubernetes-based microservices architecture.
6. Deploy a Kubernetes cluster on AWS/GCP using Infrastructure as Code, and manage auto-scaling, load balancing, and health checks for deployed services.
7. Design and deploy a scalable three-tier architecture (Web, App, DB) using Terraform and AWS EC2, RDS, and VPC components.
8. Automate the provisioning of a secure cloud infrastructure using Ansible and Terraform with role-based access control and security groups.
9. Implement a full CI/CD pipeline for a Node.js/Java web app using Jenkins or GitHub Actions, enabling automated testing, containerization, and cloud deployment.
10. Set up an end-to-end blue-green or canary deployment strategy on AWS or GCP using Jenkins and Kubernetes.
11. Build a centralized monitoring and alerting system using Prometheus, Grafana, and Loki for a cloud-deployed microservices architecture.
12. Develop a secure DevOps pipeline incorporating DevSecOps tools like SonarQube, Snyk, or Trivy for vulnerability scanning and policy enforcement.

Syllabus
Tutorials

List of Tutorials:

1. Research and explain the core principles of DevOps culture and practices. Prepare a report summarizing your findings.
2. Explore AWS Cloud fundamentals and create a presentation on how cloud computing supports DevOps methodologies.
3. Set up a GitHub repository for a sample project. Demonstrate key Git operations including branching, merging, and pull requests. Submit screenshots and explanations.
4. Install Jenkins and create a basic pipeline to automate the build and testing of your sample project. Document the pipeline configuration.
5. Build a Docker image for a simple web application and run it as a container locally. Submit the Dockerfile and container logs.
6. Deploy a multi-container application using Docker Compose and explain the service interactions. Provide configuration files and demo results.
7. Set up a local Kubernetes cluster (using Minikube or kind) and deploy a containerized application. Submit deployment manifests and screenshots of the running pods.
8. Write Terraform scripts to provision an EC2 instance and an S3 bucket on AWS. Submit the Terraform configuration files and deployment output.
9. Create an AWS CloudFormation stack to deploy a simple web server infrastructure. Document the template and deployment process.
10. Implement a blue/green deployment pipeline using Jenkins or GitHub Actions for a cloud-hosted application. Submit the pipeline code and deployment screenshots.
11. Create a CI/CD workflow using GitHub Actions that builds, tests, and deploys a containerized application. Provide the workflow YAML and deployment evidence.
12. Configure Prometheus and Grafana to monitor a deployed application. Create dashboards to visualize CPU and memory usage and submit screenshots.
13. Integrate a security scanning tool (such as Snyk or Trivy) into your CI/CD pipeline. Provide evidence of scans and vulnerability reports.

Course Outcomes

Course Outcomes: Students will be able to

1. Analyze and articulate the core principles of DevOps, including automation, collaboration, and feedback loops, within the context of scalable cloud-native architectures.
2. Design, configure, and implement Continuous Integration and Continuous Delivery (CI/CD) pipelines using industry-standard tools such as Jenkins, GitHub Actions, or GitLab CI to automate build, test, and deployment processes.
3. Deploy, manage, and scale containerized applications using Docker and orchestrate them using Kubernetes clusters on public cloud platforms (AWS, Azure, or GCP).
4. Automate infrastructure provisioning and configuration using Infrastructure as Code (IaC) tools like Terraform or AWS CloudFormation, adhering to cloud security and governance policies.
5. Integrate monitoring, logging, and DevSecOps tools such as Prometheus, Grafana, and Snyk to ensure observability, traceability, and security compliance in DevOps pipelines.
6. Architect and execute full-stack DevOps workflows leveraging cloud-native services for source control, CI/CD, infrastructure automation, deployment, and monitoring.

Books and E-Resources

For Reference Print Book –

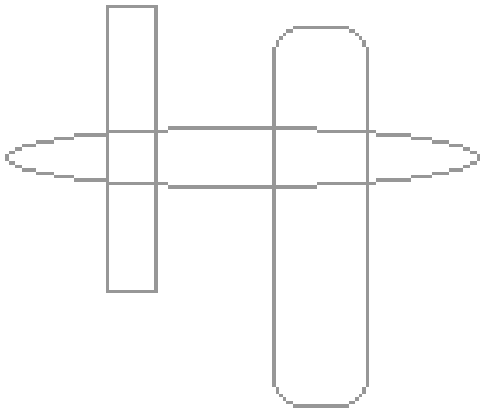
1. K. Morris; *Infrastructure as Code: Managing Servers in the Cloud*; 1st Edition; O'Reilly Media; 2016
2. S. Farcic; *DevOps Toolkit: Building the DevOps Toolkit Series*; 2nd Edition; Packt Publishing; 2021
3. N. Ernest; *Learning DevOps: Continuously Deliver Better Software*; 1st Edition; Packt Publishing; 2018

For Reference Electronic Book –

1. J. Allspaw, P. Hammond; *10+ Deploys Per Day: Dev and Ops Cooperation at Flickr*; eBook Edition; O'Reilly Media; 2010
2. D. Kuo; *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win*; eBook Edition; IT Revolution Press; 2014
3. J. Humble, D. Farley; *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*; eBook Edition; Addison-Wesley; 2010

For MOOCs and other learning Resources

1. K. Krishnan**;** *Introduction to DevOps*; 1st Edition; Coursera (offered by IBM); https://www.coursera.org/learn/devops-culture-and-mindset
2. S. Setty**;** *Getting Started with Google Kubernetes Engine*; 1st Edition; Coursera (offered by Google Cloud); https://www.coursera.org/learn/google-kubernetes-engine
3. B. Burns, Kelsey Hightower; *Architecting with Kubernetes*; 1st Edition; edX (offered by Google Cloud); https://www.edx.org/course/architecting-with-kubernetes



Double Minor in Cloud Computing

ABCXXX: Cloud Administration

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week; Tutorials: 1 Hour / Week
Total Credits: 4

Course Prerequisites: Networking Concepts, Cloud Computing, Operating System

Course Objectives:

1. To provide foundational understanding of cloud computing models, services, and deployment architectures.
2. To develop hands-on skills in managing cloud infrastructure using leading platforms such as AWS, Azure, and GCP.
3. To introduce virtualization, containerization, and orchestration tools for scalable cloud resource management.
4. To enable automation of cloud provisioning using Infrastructure as Code (IaC) and configuration management tools.
5. To learn implementing security, monitoring, and cost optimization strategies in cloud environments.
6. To integrate DevOps practices such as CI/CD pipelines for cloud-native application deployment and administration.

Course Relevance:
Cloud Administration is a critical skill in today's IT landscape, enabling scalable, secure, and cost-effective deployment of applications and infrastructure. This course equips students with practical knowledge of cloud platforms, automation tools, and DevOps practices, making them industry-ready for roles in cloud engineering, system administration, and IT operations.

Section 1: Topics/Contents

Unit 1: Introduction to Cloud Computing and Cloud Administration (4 hours)
Overview of Cloud Computing - Concepts, History, and Models (IaaS, PaaS, SaaS),Cloud Deployment Models - Public, Private, Hybrid, Community Clouds, Role of Cloud Administrator - Responsibilities, Skills, and Tools, Introduction to Cloud Service Providers (AWS, Azure, Google Cloud)

Unit 2: Cloud Infrastructure Management (5 hours)

Virtualization Fundamentals and Hypervisors, Compute Services in Cloud (VM Instances, Containers), Storage Management - Block, Object, and File Storage, Networking in Cloud - VPCs, Subnets, Load Balancers, Identity and Access Management (IAM) Basics

Unit 3: Cloud Resource Provisioning and Automation (5 hours)

Cloud Resource Provisioning and Configuration, Infrastructure as Code (IaC) Concepts and Tools (Terraform, AWS CloudFormation), Automation using Scripting and CLI Tools, Introduction to Configuration Management Tools (Ansible, Chef, Puppet), Cloud API Management and SDKs

Section 2: Topics/Contents

Unit 4: Cloud Security and Compliance (4 hours)

Cloud Security Fundamentals, Managing Cloud Identities and Permissions, Data Security and Encryption in Cloud, Compliance Standards and Best Practices (GDPR, HIPAA, PCI-DSS)

Unit 5: Monitoring, Troubleshooting, and Performance Optimization (5 hours)

Cloud Monitoring Tools and Services (CloudWatch, Azure Monitor), Logging and Auditing in Cloud Environment, Troubleshooting Cloud Infrastructure Issues, Cost Management and Optimization Techniques, Backup, Recovery, and Disaster Recovery Planning

Unit 6: Cloud Administration Case Studies and Emerging Trends (5 hours)

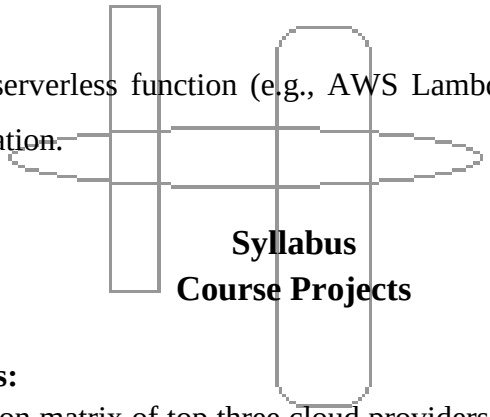
Case Study: Managing AWS Cloud Infrastructure, Case Study: Azure Cloud Administration and Best Practices, Cloud Migration Strategies and Tools, Emerging Trends in Cloud (Serverless, Edge Computing, Multi-cloud), Final Project Presentation and Course Review

Syllabus
Laboratory

List of Lab Assignments:

1. Set up a free-tier account on AWS, Azure, or Google Cloud and explore the dashboard and services.
2. Create a comparison report of IaaS, PaaS, and SaaS services offered by AWS, Azure, and GCP using real service examples.
3. Launch and configure a virtual machine instance on a cloud provider (e.g., AWS EC2 or Azure VM).

4. Create and configure cloud storage (S3 bucket in AWS or Blob Storage in Azure) and upload/download data.
5. Write a basic Infrastructure-as-Code script using Terraform to provision a cloud VM.
6. Automate the deployment of a web server using Ansible or shell scripting in a cloud VM.
7. Create and manage IAM users, groups, and roles with specific permissions in AWS or Azure.
8. Configure encryption for cloud storage and set up access policies (e.g., S3 bucket policies or Azure RBAC).
9. Enable monitoring on a cloud VM and configure CloudWatch (AWS) or Azure Monitor to track CPU and memory usage.
10. Simulate a service failure and use logs and metrics to diagnose and resolve the issue.
11. Perform a mini cloud migration project by deploying an existing local application to the cloud.
12. Deploy and test a serverless function (e.g., AWS Lambda or Azure Functions) for event-driven automation.

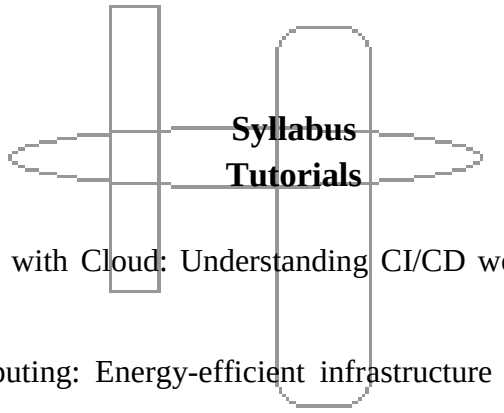


Syllabus
Course Projects

List of Course Projects:

1. Develop a comparison matrix of top three cloud providers (AWS, Azure, GCP) based on services, pricing, and global presence.
2. Create a cloud adoption roadmap for a small business including service model selection, cost estimation, and timeline.
3. Design and deploy a multi-tier cloud architecture (web server, application server, and database) using virtual machines.
4. Build a virtual private cloud (VPC) setup with custom subnets, routing tables, and firewall rules on AWS or Azure.
5. Develop and test a Terraform script to deploy a complete infrastructure with networking, compute, and storage components.

6. Create a CI/CD pipeline for automated deployment using GitHub Actions or Jenkins integrated with a cloud environment.
7. Build a secure cloud environment with IAM roles, MFA, and encryption for a sample application.
8. Conduct a compliance assessment of a cloud deployment using standard benchmarks like CIS AWS Foundations.
9. Implement monitoring and alerting for a cloud-based application using CloudWatch or Azure Monitor dashboards.
10. Perform cost optimization by analyzing billing reports and refactoring over-provisioned cloud resources.
11. Migrate a legacy web application to a serverless architecture using AWS Lambda or Azure Functions.
12. Develop a prototype of a multi-cloud or hybrid cloud solution integrating services from two providers.



List of Tutorials:

1. DevOps Integration with Cloud: Understanding CI/CD workflows and cloud-native DevOps pipelines

2. Green Cloud Computing: Energy-efficient infrastructure and sustainability in data centers

3. Cloud Billing & FinOps: Understanding cloud cost models and financial operations for multi-cloud environments

4. Disaster Recovery as a Service (DRaaS): Architecture and business continuity planning

5. Cloud-Native Design Patterns: 12-factor app model, microservices, and container orchestration

6. Cloud and Internet of Things (IoT): Integrating smart devices with cloud platforms

7. AI/ML on the Cloud: Using managed services like AWS SageMaker or Azure ML Studio

8. Open Source Cloud Platforms: Exploring OpenStack, Cloud Foundry, and Kubernetes

9. Edge Computing vs. Cloud Computing: Use cases, trade-offs, and hybrid deployments

10. Cloud Governance and Risk Management: Policies, SLAs, and governance models

11. Cloud for Startups and SMBs: Cloud adoption strategy, growth, and scalability planning

12. Quantum Computing in the Cloud: Introduction to services like IBM Q, Amazon Braket, and Microsoft's Azure Quantum

Course Outcomes

Course Outcomes: Students will be able to

1. Describe the key concepts of cloud computing including service models, deployment models, and architecture.
2. Configure and manage cloud-based resources such as compute, storage, and networking components.
3. Develop Infrastructure-as-Code scripts and automation workflows using tools like Terraform or Ansible.
4. Implement cloud security measures including IAM, encryption, and regulatory compliance practices.
5. Analyze monitoring metrics, troubleshoot cloud services, and apply performance optimization techniques.
6. Design and evaluate cloud-native solutions using real-world case studies and current trends like DevOps/server less.

Books and E-Resources

For Reference Print Book –

1. K. Morris; *Infrastructure as Code: Managing Servers in the Cloud*; 1st Edition; O'Reilly Media; 2016
2. G. Reese; *Cloud Application Architectures: Building Applications and Infrastructure in the Cloud*; 1st Edition; O'Reilly Media; 2009
3. R. Buyya, C. Vecchiola, S. T. Selvi; *Mastering Cloud Computing: Foundations and Applications Programming*; 1st Edition; McGraw Hill; 2013

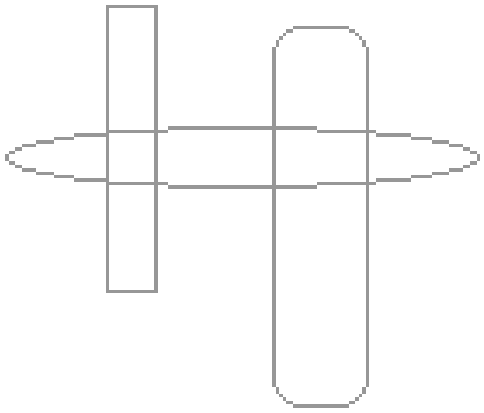
For Reference Electronic Book –

1. T. White; *Hadoop: The Definitive Guide*; 4th Edition; O'Reilly Media; 2015

2. S. Nelson-Smith; *Test-Driven Infrastructure with Chef*; 2nd Edition; O'Reilly Media; 2013
3. Mahmood; *Cloud Computing Using AWS: Cloud Implementation and Management*; 1st Edition; Springer; 2021

For MOOCs and other learning Resources

1. *Cloud Computing Specialization*; Offered by University of Illinois at Urbana-Champaign; Coursera; https://www.coursera.org/specializations/cloud-computing
2. *AWS Cloud Practitioner Essentials*; Offered by Amazon Web Services; edX; https://www.edx.org/course/aws-cloud-practitioner-essentials
3. *Google Cloud Fundamentals: Core Infrastructure*; Offered by Google Cloud; Coursera; https://www.coursera.org/learn/gcp-fundamentals



Double Minors in Cloud Computing

ABCXXX: Project

Teaching Scheme: Laboratory: 8 Hours/Week
Total Credits: 4

Syllabus

Aim

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

Project Group and Topic Selection and Synopsis:

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

Overview of the Course:

1. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
2. The project requires the students to conceive, design, implement and operate a mechanism (the design problem). The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts. If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

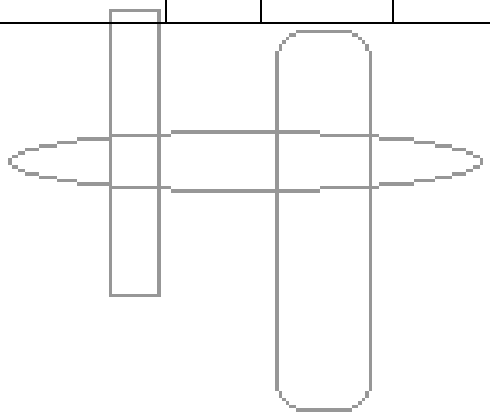
3. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
4. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
5. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
6. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
7. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
8. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

Note:

The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

Double Minor Course: Prompt Engineering

Course No.	Semester	Course Name	Teaching Scheme (Hours/Week)			Examination Scheme				Credits
			Th	Lab	Tut				Total	
C1	III	Introduction to Artificial Intelligence	2	2	0					3
C2	IV	Natural Language Processing	2	2	0					3
C3	V	Large Language Models	2	2	0					3
C4	VI	Prompt Engineering for Developers	2	2	0					3
C5	VII	Building Systems with GPT	2	2	1					4
C6	VIII	Project	0	8	0					4



Double Minor in Prompt Engineering

ABCXXX: Introduction to Artificial Intelligence

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week;
Total Credits: 3

Course Prerequisites: Basics of AI, Programming fundamentals, Understanding of LLMs (e.g., GPT-based systems)

Course Objectives:

1. To introduce the foundational principles of Artificial Intelligence and intelligent agents, with emphasis on their relevance in prompt-based Large Language Model (LLM) systems.
2. To familiarize students with uninformed and informed search strategies for resolving prompts and exploring solution spaces within prompt engineering contexts.
3. To enable learners to apply heuristic techniques and constraint satisfaction models for designing goal-oriented and optimized prompting strategies.
4. To develop skills in logic-based reasoning using propositional and first-order logic for structured prompt generation and rule-driven inference.
5. To equip students with practical knowledge of rule-based prompt engineering through scripting, JSON/YAML template design, and prompt automation using LLM APIs.
6. To impart the ability to design and implement multi-step prompt sequences using classical AI planning models such as STRIPS, planning graphs, and hierarchical planning frameworks.

Course Relevance:

This course bridges the gap between classical Artificial Intelligence techniques and modern prompt engineering for Large Language Models (LLMs). By integrating search algorithms, logic-based reasoning, heuristic planning, and rule-driven scripting, students gain a holistic understanding of how AI principles empower intelligent prompt design and automation. The course is highly relevant for careers in AI development, NLP systems, chatbot design, and LLM application engineering, making it essential for students aiming to work at the forefront of AI-driven language technologies.

Section 1: Topics/Contents

Unit 1: Foundations of AI and Intelligent Agents in Prompting Context (4 Hours)

AI in the Era of Large Language Models (LLMs): Understanding AI fundamentals with a focus on generative systems, **AI vs Non-AI Techniques in Prompt-Driven Systems:** Comparative exploration, **Knowledge Representation for LLM Systems:** Ontologies, embeddings, semantic frames, **Knowledge-Based Systems for Language Understanding:** Rule-based vs statistical methods.

Intelligent Agents: Concepts of rationality, Types of environments, Agent architectures, **Prompt Problem Formulation:** Case studies: Vacuum World, 8-Queens Puzzle, Route-Finding

Unit 2: Uninformed Search for Prompt Completion and Exploration (4 Hours)

Role of Search in LLM Prompt Resolution: Mapping prompts to internal decision/search processes, **Search Algorithms in Prompt Context:** Depth-First Search (DFS), Breadth-First Search (BFS), Depth-Limited Search (DLS), Iterative Deepening DFS, and Bidirectional Search, **Comparative Evaluation:** Time-space tradeoffs in LLM prompt handling

Unit 3: Heuristic and Goal-Oriented Prompting Strategies (6 Hours)

Heuristic Methods in Prompt Design: Generate & Test, Hill Climbing, Best-First Search, A* and AO* for prompt planning, **Constraint Satisfaction Problems (CSP):** Prompt constraint modeling, **Game-Playing in Prompting Systems:** Minimax, Alpha-Beta Pruning, Quiescence Search and LLM behavior tuning

Section 2: Topics/Contents

Unit 4: Logic-Based Prompt Reasoning and LLM Rule Structuring (6 Hours)

Logical Agents in Prompt Engineering: Wumpus World as a prompt reasoning case, **Propositional Logic:** Syntax and semantics, Inference rules: forward & backward chaining **First-Order Logic (FOL):** Representation and prompt formulation, **Pattern Matching and Rule Learning:** Structured prompting through rule-based learning

Unit 5: Rule-Based Prompt Engineering Using Scripting and DSLs (4 Hours)

Scripting Languages for Prompts: Declarative vs imperative, JSON, YAML templates **Prompt Template Design and Flow:** Conditional logic and control flow, Dynamic prompt generation using scripting, **Case Study:** Conversational system driven by rules, **LLM API Integration:** Real-time interaction with LLMs (e.g., OpenAI API, Cohere, etc.)

Unit 6: AI Planning for Prompt Sequence Design (4 Hours)

Prompt Sequence Planning: AI planning concepts in multi-step prompts, **Planning Techniques:** STRIPS, Goal Stack, Forward/Backward Planning, Planning Graphs, Hierarchical Task Networks (HTNs), **Least Commitment Principle:** Flexibility in prompt path design

Syllabus
Laboratory

List of Lab Assignments:

1. Implement Prompt Response System using BFS and DFS
2. Design Vacuum World and 8-Queens problem with prompt input
3. Implement Hill Climbing-based prompt tuning for query optimization
4. Apply A* for LLM response filtering in a chain-of-thought prompt
5. Create Prompt Templates using JSON and YAML
6. Simulate a Rule-Based System using Forward and Backward Chaining
7. Develop Minimax-based prompt for adversarial chatbot
8. Implement Prompt Planning using STRIPS
9. Build a conversational flow using YAML + Python scripting
10. Automate Prompt Chains using goal stack planning
11. Simulate a Wumpus World using logic and prompt instructions
12. Integrate OpenAI GPT API with rule-driven JSON prompt system

Syllabus
Course Projects

List of Course Projects:

1. Design an intelligent agent that interprets prompts to solve classic AI problems like 8-Queens or Route Finding using simple rules and representations.
2. Build a system that uses a structured knowledge base (e.g., JSON/OWL) to generate answers using LLM prompts, demonstrating rational agent behavior.
3. Create a tool that uses uninformed search (BFS, DFS, etc.) to generate multiple possible LLM prompt completions for open-ended questions.
4. Implement a system that explores the search tree of prompt variations using Iterative Deepening Search for prompt tuning and re-ranking.
5. Develop a heuristic-based prompt optimizer that uses A* to select the best sequence of prompt tokens for accurate model output.

6. Design a generator that satisfies logical or linguistic constraints (e.g., tone, length, keyword inclusion) using constraint satisfaction strategies.
7. Simulate the Wumpus World with prompts, where the agent uses propositional and first-order logic to make decisions through chained prompts.
8. Build a rule-based reasoning system that dynamically selects or generates prompts using forward and backward chaining techniques.
9. Create a system to generate structured prompts from user input using YAML/JSON templates and conditional scripting (e.g., Python/Jinja).
10. Develop a rule-based dialogue agent integrated with an LLM API, using declarative rules and scripted prompt branching for varied user intents.
11. Design a planner that takes a user goal and uses STRIPS to generate a logical sequence of prompts to reach that goal with an LLM.
12. Implement a hierarchical planner that breaks down complex tasks (e.g., writing an essay) into structured prompt stages using HTN (Hierarchical Task Networks).

Course Outcomes

Course Outcomes: Students will be able to

1. Explain foundational AI principles and their application in prompt-driven LLM systems
2. Apply uninformed and informed search techniques for prompt resolution and design
3. Analyze and implement heuristic-based prompting strategies and game models
4. Design logic-based rules for intelligent prompt systems using FOL and propositional logic
5. Develop rule-based prompt templates using scripting and declarative tools
6. Apply planning methods to construct efficient prompt chains and multi-step sequences

Books and E-Resources

For Reference Print Book –

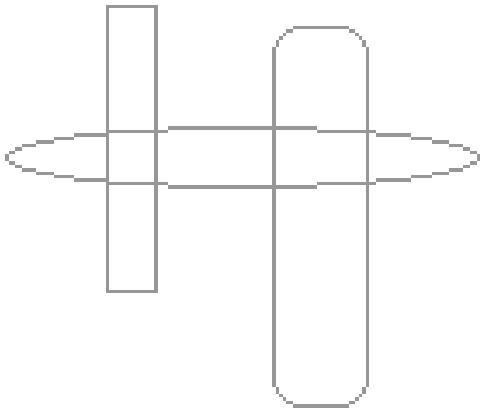
1. S. Russell, P. Norvig; *Artificial Intelligence: A Modern Approach*; 4th Edition; Pearson; 2020
2. D. Jurafsky, J.H. Martin; *Speech and Language Processing*; 3rd Draft; Pearson; 2023
3. K. Chollet; *Deep Learning with Python*; 2nd Edition; Manning; 2021

For Reference Electronic Book –

1. M. Balakrishnan, A. Prakash; *Artificial Intelligence and Machine Learning*; McGraw Hill India; 2021
2. A. Mishra; *Prompt Engineering in Generative AI*; Wiley India; 2024
3. P. Singh; *Fundamentals of AI and Prompt Programming*; Cengage Learning; 2024

For MOOCs and other learning Resources

1. **Coursera – Prompt Engineering for ChatGPT**
   https://www.coursera.org/learn/prompt-engineering
2. **DeepLearning.AI – Building Systems with the ChatGPT API**
   https://www.deeplearning.ai/short-courses/building-systems-with-chatgpt-api/
3. **edX – Artificial Intelligence (Columbia University)**
   https://www.edx.org/course/artificial-intelligence-ai



Double Minor in Prompt Engineering

ABCXXX: Foundations of Natural Language Processing

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week;
Total Credits: 3

Course Objectives:

1. To introduce the fundamentals, scope, and challenges of Natural Language Processing (NLP).
2. To understand grammar-based and statistical language models including N-gram techniques.
3. To apply regular expressions, finite automata, and text normalization techniques in NLP.
4. To analyze morphological structures using finite state transducers.
5. To design syntactic parsers for English and Indian languages using context-free grammar.
6. To explore semantic analysis, word sense disambiguation, and natural language generation.

Course Relevance: Although Natural Language Processing (NLP) has been with us for quite some time, it has only recently gained industry-wide attention, thanks to Deep Learning. Today, NLP is a core competence area in Data Science and IT, with applications spanning across sectors that rely on harnessing language data's potential. Essentially, GEN based NLP applications are designed to extract relevant and meaningful information from natural human language data and impart machines with the ability to interact with humans.

Unit 1: Introduction **(4 Hours)**
Introduction, what is natural language processing? Applications of NLP, Origins of NLP, Challenges of NLP, Language and Knowledge, Language and Grammar, Processing Indian Languages

Unit 2: Language Modelling **(6 Hours)**
Grammar-based language models, lexical functional Grammar (LFG), Government and Binding (GB), Lexical functional Grammar Model, Generative grammars, Statistical Language Model. N-gram Language Models - N-Grams, Evaluating Language Models, Sampling sentences from a language model, Generalization and Zeros, Smoothing, Kneser-

Ney Smoothing, Huge Language Models and Stupid Backoff, Advanced: Perplexity's Relation to Entropy.

Unit 3: Regular Expressions and Automata (5 Hours)
Formal Language Theory: Basic Notions, Basic Regular Expression Patterns, Disjunction, Grouping and Precedence, Advanced Operators, Substitution, Finite State Automata, NFSA. Words, Corpora, Text Normalization, Minimum Edit Distance.

Unit 4: Words and Transducers (5 Hours)
Morphology, Inflectional Morphology, Derivational Morphology, Finite State Morphological Parsing, Construction of Finite State Lexicon, Finite State Transducers

Unit 5: Syntactic Analysis (5 Hours)
Context Free Grammar, parsing, Top-down Parsing, Bottom-up parsing, Probabilistic parsing, Indian Languages parsing

Unit 6: Semantic Analysis (5 Hours)
Meaning Representation, Lexical Semantic, Ambiguity, Word Sense Disambiguation, Discourse processing, Natural Language Generation

Laboratory

List of Experiments

13. Manual Word Analysis and Word Generation on plain paper
14. Manual Morphology on plain paper
15. Manual N-Grams and N-Grams Smoothing on plain paper
16. Manual Chunking on plain paper
17. Manual Building Chunker on plain paper
18. Write a program for Word Analysis
19. Write a program for Word Generation
20. Write a program for Morphology
21. Write a program for N-Gram generations
22. Write a program for N-Grams Smoothing
23. Write a program for Chunking
24. Write a program for to build Building Chunker

Course Outcomes

Course Outcomes: Student will able to

6. Interpret language and grammar for given natural language
7. Infer grammar based language modelling for given natural language

8. Perform normalization for given natural language
9. Prepare Finite State Lexicons for Finite State Transducers
10. Construct shallow depth lexical analyzer and syntactic analyzer
11. Develop shallow depth type dependency parser for given natural language

Books and E-Resources

For Reference Print Book -

Text Books: (As per IEEE format)

1. Tanveer Siddiqui and U S Tiwary, "Natural Language Processing and Information Retrieval" Fourth Impression, Oxford, ISBN-13:978-019-569232-7.
2. Daniel Jurafsky and James H Martin., "Speech and Language Processing", 2nd edition, Pearson, Second Impression-2014,ISBN: 978-93-325-1841-4

Reference Books: (As per IEEE format)

1. Alexander Clark, Chris Fox and Shalom Lappin "The Handbook of Computational Linguistics and Natural Language Processing",Wiley-Blackwell-2013, ISBN-978-1-118-34718-8
2. Allen, James, Natural Language Understanding, Second Edition, Benjamin /Cumming, 1995.
3. Charniack, Eugene, Statistical Language Learning, MIT Press, 1993.
4. Manning, Christopher and Heinrich, Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
5. Radford, Andrew et. al., Linguistics, An Introduction, Cambridge University Press, 1999.
6. Journals : Computational Linguistics, Natural Language Engineering, Machine Learning, Machine Translation, Artificial Intelligence
7. Conferences : Annual Meeting of the Association of Computational Linguistics (ACL), Computational Linguistics (COLING), European ACL (EACL), Empirical Methods in NLP (EMNLP), Annual Meeting of the Special Interest Group in Information Retrieval (SIGIR), Human Language Technology (HLT).

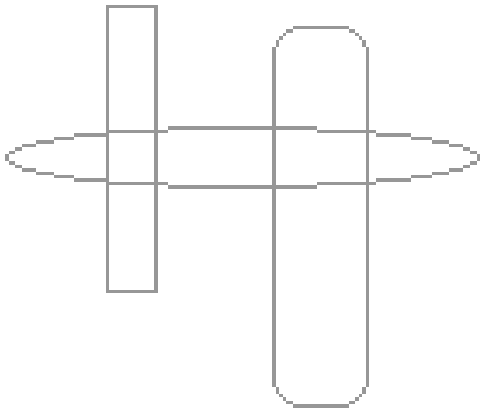
For Reference Electronic Book –

1. Dan Jurafsky and James H. Martin; " Speech and Language Processing (3rd ed. draft)"; Jan 2025.
   Available: web.stanford.edu/~jurafsky/slp3/ed3book_Jan25.pdf

For MOOCs and other learning Resources

3. www.nptelvideos.in
4. www.nfnlp.com

5. https://nlp-iiith.vlabs.ac.in/List%20of%20experiments.html?domain=Computer%20Science
6. https://www.cse.iitb.ac.in/~cs626-460-2012/



Double Minor in Prompt Engineering

Large Language Models

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 3

Course Prerequisites: NLP Fundamentals, Deep Learning

Course Objectives:

1. To introduce fundamentals and evolution of large language models.
2. To understand transformer architecture used in modern LLMs.
3. To explain pretraining and fine-tuning approaches.
4. To apply LLMs to NLP tasks like generation, summarization, QA.
5. To deploy LLMs using open libraries and APIs.
6. To explore ethical issues and trends in LLM research and applications.

Section 1: Topics/Contents

Unit 2: Introduction to Large Language Models (4 Hours)

History of NLP, shift to deep learning, word embeddings, limitations of RNNs, emergence of LLMs.

Unit 2: Transformer Architecture (5 Hours)

Self-attention, multi-head attention, encoder-decoder models, positional encoding, BERT vs GPT.

Unit 3: Pretraining and Fine-Tuning (5 Hours)

Language modeling strategies: Masked (MLM) vs Causal (CLM), Transfer learning in LLMs

Prompt tuning and few-shot learning, preprocessing large-scale datasets for LLM training

Section 2: Topics/Contents

Unit 4: Question Answering, Information Retrieval, and RAG (4 Hours)

Basics of Information Retrieval, Information Retrieval with Dense Vectors, Answering Questions with RAG, Evaluating Question Answering

Unit 5: Chatbot Implementation using Large Language Models (5 Hours)

Components of a Modern Chatbot: NLU, Dialogue Manager, NLG, Implementing a Chatbot using Pre-trained LLMs (GPT, Claude, Gemini), Multi-turn Dialogue Handling and Context Preservation, Building chatbots using GPT, Claude, Gemini, Toolkits: LangChain, Rasa, or Hugging Face Transformers, Evaluation metrics: Coherence, BLEU, Success Rate

Unit 6: Dialogue System Design and Evaluation (5 Hours)

Dialogue System Architecture and Pipeline, Designing Multi-turn Dialogues using Prompts, Context Management and Memory in LLMs, Evaluation Metrics for Dialogue Systems (BLEU, METEOR, Success Rate), Case Studies: Alexa, Google Bard, ChatGPT, Replika

Syllabus
Laboratory

List of Experiments

1. Use HuggingFace pipeline for text classification.
2. Generate text using GPT-2.
3. Fine-tune a transformer on custom dataset.
4. Use summarization model on long text.
5. Implement chatbot using OpenAI API.
6. Analyze tokenization and embeddings.
7. Compare outputs of BERT vs GPT.
8. Prompt engineering experiments.
9. Visualize attention weights.
10. Create a FastAPI wrapper for LLM service.

Syllabus
Course Projects

List of Course Projects

1. Develop an LLM-powered chatbot for domain-specific task.
2. Create summarizer for technical documents.
3. Build QA system using LLM with retrieval-based pipeline.
4. Fine-tune LLM on health/legal/coding data.

5. Build an interactive app using Gradio and HuggingFace.

Syllabus
Tutorials

List of Tutorials

1. Transformer components walkthrough.
2. Tokenization strategies comparison.
3. Pretraining vs fine-tuning tasks.
4. LLM pipeline in HuggingFace.
5. Prompt tuning techniques.
6. Use case mapping to LLM architectures.
7. Attention visualization.
8. Deployment using Gradio.
9. Error handling in API-based generation.
10. Responsible LLM usage scenarios.

Course Outcomes

Course Outcomes: Students will be able to

1. Understand the evolution and fundamentals of large language models.
2. Explain and apply the transformer architecture.
3. Apply pretraining and fine-tuning methods.
4. Implement LLMs for key NLP applications.
5. Utilize tools and frameworks to deploy LLMs.
6. Analyze ethical issues and trends in LLM development.

Books and E-Resources

Text Books

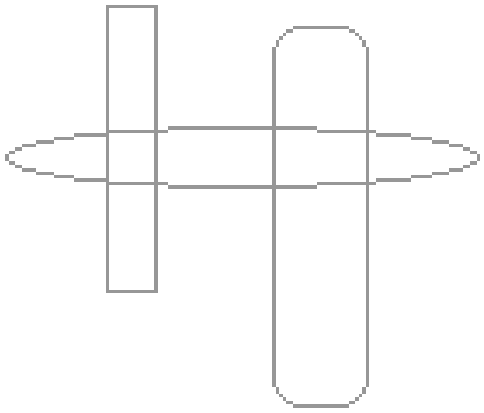
1. Jacob Eisenstein, Natural Language Processing, MIT Press, 2022.
2. Thomas Wolf et al., Transformers for NLP, Packt, 2020.

Reference Books

1. Delip Rao, Brian McMahan, Natural Language Processing with PyTorch, O'Reilly, 2019.
2. Dennis Rothman, Transformers for Natural Language Processing, Packt, 2021.
3. Sebastian Ruder, NLP Progress Blog (Online)

MOOCs and Learning Resources

1. HuggingFace Course - https://huggingface.co/course
2. OpenAI Documentation - https://platform.openai.com/docs
3. Stanford CS25 - https://web.stanford.edu/class/cs25/
4. DeepLearning.AI NLP Specialization - https://www.coursera.org/specializations/natural-language-processing
5. FastAI NLP Lectures - https://course.fast.ai



Double Minor in Prompt Engineering

ABCXXX: Prompt Engineering for Developers

Teaching Scheme
Theory: 2 Hours / Week ;Laboratory: 2 Hours / Week; Tutorials: 1 Hours / Week
Total Credits: 4

Course Prerequisites:
Students enrolling in this course are expected to have:

1. Basic understanding of Artificial Intelligence and Machine Learning concepts
2. Familiarity with Natural Language Processing (NLP) techniques
3. Programming knowledge (preferably Python)
4. Exposure to APIs and cloud-based tools is beneficial but not mandatory

Course Objectives:
On completing this course, students will be able to:

1. Understand the foundational concepts of prompt engineering and transformer-based LLMs
2. Explore different types of prompts and prompt design patterns for various NLP tasks
3. Apply decoding strategies and design effective prompts for task-specific applications
4. Optimize prompts using advanced techniques including tuning, reasoning chains, and multi-modal approaches
5. Integrate LLMs with APIs and tools for real-world deployment
6. Analyze and apply prompt security strategies to defend against prompt injection and jailbreak attacks

Course Relevance:
Prompt Engineering is an essential skill in the age of Large Language Models (LLMs), enabling precise and safe communication with AI systems. This course equips learners with practical techniques to design, evaluate, and secure prompts for applications in education, healthcare, business automation, and creative industries. With increasing adoption of LLMs across domains, the course bridges the gap between theoretical AI knowledge and hands-on deployment of intelligent systems.

Section 1: Topics/Contents

Unit 2: Foundations of Prompt Engineering and Transformer Models (4 Hours)

Rise of In-context Learning, Introduction to Prompt Engineering, Principles of Prompt Engineering, Understanding Transformers, Prompt Design: Structural Elements

Unit 2: Types of Prompts and Prompt Patterns (4 Hours)

Prompting Techniques**:** Zero-shot, One-shot, and Few-shot prompting, Instruction-based prompting, Advanced Prompt Patterns**:** Chain-of-Thought (CoT) prompting, Role and system prompts, ReAct, Tree-of-Thoughts, and Self-Consistency techniques

Unit 3: Prompt Design Techniques and Decoding Parameters (5 Hours)

Prompt Design: Linguistic Elements, Prompt Patterns and Templates, Parameters of Decoding: greedy, top-k, top-p, temperature, First Basic Prompt and iterative refinement, Settings to Keep in Mind for effective prompt control

Section 2: Topics/Contents

Unit 4: Designing Prompts for Different Tasks (5 Hours)

Text Summarization, Question Answering, Text Classification, Role Playing, Code Generation, Reasoning

Unit 5: Advanced Prompt Engineering and Optimization Techniques (5 Hours)

Prompt Optimization Strategies, Prompt Tuning and Fine-tuning, Handling Ambiguity and Uncertainty in Prompts, Multi-modal Prompt Engineering: GPT-4 Vision, CLIP, DALL·E, Flamingo, and Gemini, Tool-Augmented Prompting **(**ReAct, Tree-of-Thoughts), Ethical Considerations and Bias Mitigation

Unit 6: Applications and Security in Prompt Engineering (5 Hours)

Integration with Tools and APIs: LangChain, OpenAI API, HuggingFace , Prompt security and management, Prompt Injection, Prompt Leaking, Jail-breaking, Self-Consistency Examples, Real-world apps using secure prompting (e.g., Copilot, Bard, Khanmigo)

Syllabus
Laboratory

List of Lab Assignments:

1. Create a basic zero-shot and few-shot prompt using OpenAI’s playground
2. Implement chain-of-thought (CoT) prompting for a math reasoning problem
3. Compare outputs for different decoding parameters: top-k, top-p, temperature
4. Design prompts for summarization and question answering tasks

5. Write role-based prompts simulating a customer support agent
6. Test prompt injection attack on a basic chatbot and analyze results
7. Integrate OpenAI GPT model with Python using the OpenAI API
8. Fine-tune a simple prompt for sentiment analysis with iterative refinement
9. Create a multi-turn chatbot using system + user prompts
10. Evaluate performance of prompts using BLEU/ROUGE for summarization
11. Visual prompting with GPT-4 Vision or CLIP for image captioning (demo use)
12. Secure a prompt using input sanitization and system prompt constraints

Syllabus

Course Projects

List of Course Projects:

1. Build a domain-specific question-answering assistant using few-shot prompting
2. Design a personal study assistant that summarizes textbooks and generates quizzes
3. Develop a job application helper that generates cover letters from user input
4. Create a multi-modal prompt-based art captioning system (text + image)
5. Implement a health chatbot using CoT prompts and OpenAI API
6. Build a secure prompt-response system resistant to injection attacks
7. Compare various prompt tuning strategies for text classification
8. Build a resume analyzer with prompts that give feedback to the user
9. Develop a conversational system with ReAct-style tool usage
10. Create a legal document summarizer using prompt chaining
11. Analyze LLM behavior on biased prompts and propose mitigation strategies
12. Build a dashboard to test and evaluate prompts in real-time (UI + API)

Syllabus

Tutorials

List of Tutorials:

1. Introduction to transformer models and attention mechanisms
2. What makes a prompt effective? Exploring basic structure and tone
3. Hands-on: writing zero-shot vs few-shot prompts
4. Exploring prompt templates for summarization and translation
5. Guided exercise: create a prompt for a classroom assistant

6. Case study: How GPT-4 reasons through chain-of-thought prompts
7. How decoding parameters influence creativity and precision
8. Role play: Acting as different personas using prompts
9. Prompt security: Detecting and preventing injection attacks
10. Tool integration demo with LangChain and OpenAI APIs
11. Ethics in prompting: testing hallucinations and biases
12. Future trends in prompt engineering: agents, memory, and automation

Course Outcomes

Course Outcomes : Students will be able to

1. Explain the structure and working principles of transformer models and in-context learning in LLMs.
2. Identify and design appropriate prompts using zero-shot, few-shot, CoT, and other prompting techniques for diverse NLP tasks.
3. Implement and refine task-specific prompts using linguistic, structural, and decoding-based strategies.
4. Develop advanced prompt-driven applications with optimization, multi-modal integration, and tool augmentation.
5. Evaluate and defend prompts against common security vulnerabilities such as prompt injection and jailbreaks.
6. Build and deploy ethical, reliable, and context-aware LLM-powered applications using prompt engineering.

Books and E-Resources

For Reference Print Book –

1. S. Russell, P. Norvig; *Artificial Intelligence: A Modern Approach*; 4th Edition; Pearson; 2020
2. J. Eisenstein; *Natural Language Processing*; 1st Edition; MIT Press; 2019
3. D. Jurafsky, J.H. Martin; *Speech and Language Processing*; 3rd Edition (Draft); Pearson; 2023

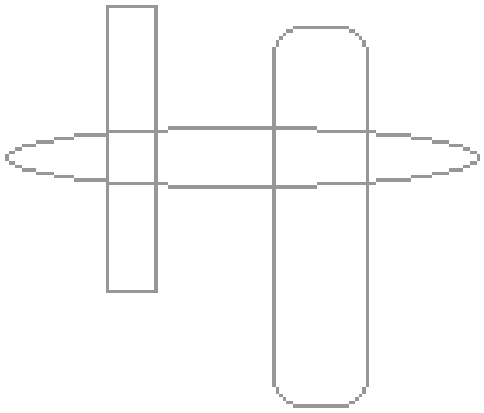
For Reference Electronic Book –

1. A. Bahri; *Prompt Engineering for Everyone*; 1st Edition; Leanpub; 2023

2. D. Shapiro; *Transformers for Natural Language Processing*; 2nd Edition; Packt Publishing; 2021
3. J. Wolf, T. Sanh, L. Debut, et al.; *Hugging Face Transformers: State-of-the-art Natural Language Processing*; 1st Edition; O'Reilly Media; 2022

For MOOCs and other learning Resources

1. A. Karpathy; *Neural Networks: Zero to Hero*; YouTube Series; 2023 https://www.youtube.com/@karpathy
2. DeepLearning.AI; *ChatGPT Prompt Engineering for Developers*; OpenAI & DeepLearning.AI; 2023 https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/
3. Coursera – Stanford University; *Natural Language Processing with Deep Learning* by C. Manning; Coursera; 2022 https://www.coursera.org/learn/natural-language-processing-with-deep-learning



Double Minor in Prompt Engineering

ABCXXX: Building Systems with GPT

Teaching Scheme:
Theory: 2 Hours / Week ;Laboratory: 2 Hours / Week; Tutorials: 1 Hour / Week
Total Credits: 4

Course Prerequisites: Python, REST APIs, Web App Development Basics

Course Objectives:
By the end of this course, students will be able to:

1. Understand the evolution, capabilities, and limitations of GPT and OpenAI APIs.
2. Design and integrate GPT-powered systems into web and desktop applications.
3. Implement function calling and multi-agent workflows for complex task automation.
4. Develop multimodal AI applications using DALL·E and Whisper APIs.
5. Deploy lightweight AI applications with effective UI/UX on cloud and local environments.
6. Package and distribute cross-platform desktop applications powered by GPT.

Course Relevance:
This course equips learners with practical skills to harness OpenAI's GPT and related APIs for building intelligent applications. Students will understand the foundations of large language models, system integration, and deployment across web and desktop platforms. The course emphasizes hands-on development, ethical use, and modern AI workflows, preparing students for real-world AI product development in diverse domains such as chatbots, content generation, multimodal apps, and productivity tools.

Section 1: Topics/Contents

Unit 1: Introduction to OpenAI API Suite & ChatGPT (5 Hours)

Overview of OpenAI's APIs: ChatGPT, DALL·E, Whisper, GPT Evolution: GPT-1 to GPT-4, Capabilities & Limitations of GPT Models, Understanding Tokenization, Context Window, and Prompt Length, comparing OpenAI vs other LLM providers (Claude, Gemini, Mistral, etc.)

Unit 2: System Design & Integration (5 Hours)

GPT in Product Workflows: Chatbots, Content Assistants, Ideation Tools, Frontend , Integration: Streamlit, Flask, React Basics, Deployment Best Practices: API Keys, Rate Limits, Billing Awareness, Logging, Prompt Versioning, and Evaluation, Introduction to Multi-Agent Systems: AutoGPT, LangGraph, CrewAI

Unit 3: Tool Use & Function Calling (4 Hours)

Function Calling in GPT: Structured Outputs & JSON Responses, Connecting to External APIs (e.g., weather, calculator, knowledge base), Multi-step Reasoning with Tool Use, Designing Task-Oriented Multi-Agent Systems (GPT + Tools)

Section 2: Topics/Contents

Unit 4: Multimodal AI Applications with DALL·E & Whisper APIs (4 Hours)

Introduction to DALL·E API (text-to-image generation), Integrating DALL·E with ChatGPT for Visual Applications, Whisper API for speech-to-text and vice versa, Building Desktop UI using PyQt or Tkinter

Unit 5: Lightweight Application Deployment & UI Integration (5 Hours)

Web App Deployment using Streamlit/Flask, Exporting Outputs: PDFs, Presentations, or Reports, Embedding ChatGPT in Existing Platforms (WordPress, LMS, etc.), UI Best Practices for AI Applications (Forms, History, Responsiveness), Sharing & Hosting with GitHub, Vercel, or Hugging Face Spaces

Unit 6: Desktop Apps & Cross-Platform Deployment (5 Hours)

Developing PyQt-Based Desktop Apps for GPT-Powered Tools (Essay Generation, Visual Tools), Managing File I/O and Exporting Outputs (PDF, PPT) in Desktop Apps, End-to-End Deployment and Packaging using PyInstaller or Electron, Integrating ChatGPT into Offline or Local Desktop Environments (Concepts & Tools)

Syllabus
Laboratory

List of Lab Assignments:

1. Basic ChatGPT API text generation and response customization
2. Implementing a chatbot using Streamlit or Flask
3. Function calling: Connect GPT to a weather API for live data
4. Generate images using the DALL·E API from text prompts
5. Speech-to-text transcription with Whisper API and text display

6. Build a simple multi-agent workflow combining GPT and a calculator API
7. Export generated text responses as PDF files from a web app
8. Create a prompt versioning system and test different prompt outputs
9. Build a desktop GUI with PyQt integrating ChatGPT API for essay writing
10. Package a PyQt desktop app as a standalone executable using PyInstaller
11. Integrate ChatGPT chatbot into a WordPress page or LMS platform
12. Deploy a web app on GitHub Pages, Vercel, or Hugging Face Spaces

Syllabus
Course Projects

List of Course Projects:

1. **AI-Powered Personal Tutor:** A chatbot that answers student queries with role-specific prompts
2. **Multimodal Creative Studio:** An app combining ChatGPT, DALL·E, and Whisper for text, image, and voice creativity
3. **Automated Report Generator:** Generate business or academic reports from user prompts and export as PDFs
4. **Code Assistant Desktop App:** A PyQt-based tool that helps review and suggest fixes for code snippets
5. **Multi-Agent Workflow Automation:** A web app that uses GPT + external APIs for booking, weather, and reminders
6. **Presentation Builder:** End-to-end generation of PowerPoint slides using ChatGPT and DALL·E APIs

Syllabus
Tutorials

List of tutorial:

1. Setting up OpenAI API keys and basic API usage
2. Understanding and optimizing prompt engineering techniques
3. Building a chatbot with Streamlit from scratch
4. Working with function calling and external API integrations

5. Introduction to PyQt for AI-powered desktop app interfaces
6. Using Whisper API for speech recognition in Python
7. Exporting AI-generated content to various file formats (PDF, PPT)
8. Best practices for API security, billing, and rate limits
9. Deploying apps on free hosting platforms: GitHub, Vercel, Hugging Face
10. Packaging desktop applications with PyInstaller
11. Combining GPT and DALL·E for creative AI projects
12. Basics of multi-agent systems and AutoGPT introduction

Course Outcomes

Course Outcomes: Students will be able to

1. Explain the architecture and functionalities of GPT and OpenAI APIs.
2. Build and integrate GPT-based solutions into web and desktop platforms.
3. Apply function calling techniques to extend GPT's capabilities with external tools.
4. Develop multimodal AI applications combining text, image, and speech.
5. Deploy AI applications effectively with user-friendly interfaces and export options
6. Package and distribute cross-platform desktop AI applications for practical use.

Books and E-Resources

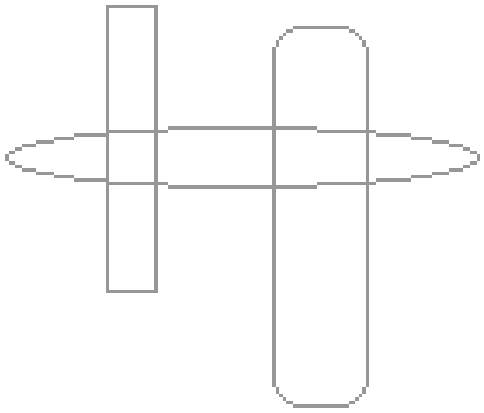
For Reference Print Book –

1. **S. Russell, P. Norvig**; *Artificial Intelligence: A Modern Approach*; 4th Edition; Pearson; 2020
2. **J. Kelleher, B. Mac Namee, A. D'Arcy**; * Fundamentals of Machine Learning for Predictive Data Analytics*; 2nd Edition; MIT Press; 2020
3. **K. Morris**; *Infrastructure as Code: Managing Servers in the Cloud*; 1st Edition; O'Reilly Media; 2016

For MOOCs and other learning Resources

1. **DeepLearning.AI**; *Generative AI with Large Language Models*; Coursera; 2023 https://www.coursera.org/learn/generative-ai-with-large-language-models
2. **Stanford University**; *Natural Language Processing with Deep Learning*; CS224N http://web.stanford.edu/class/cs224n/

3. **IBM**; *AI Engineering Professional Certificate*; Coursera
https://www.coursera.org/professional-certificates/ai-engineering



Double Minor in Prompt Engineering

ABCXXX: Project

Teaching Scheme: Laboratory: 8 Hours/Week

Total Credits: 4

Syllabus

Aim

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

Project Group and Topic Selection and Synopsis:

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

Overview of the Course:

25. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
26. The project requires the students to conceive, design, implement and operate a mechanism (the design problem). The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts. If the mechanism

incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

27. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
28. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
29. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
30. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
31. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
32. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

Note:

The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.