MERN Stack powered by Mongo DB

Course Objectives	Course Outcomes
Comprehend Web Development Architecture: Gain knowledge of how frontend and backend components interact & Comprehension of the client-server architecture.	Develop foundational skills in frontend development by mastering HTML/CSS basics, advanced techniques like flexbox and grid layouts, and DOM manipulation in this introductory Full Stack course.
Frontend Development with React.js: Proficiency in building user interfaces using React.js.	Exhibit essential JavaScript skills, mastering variables, control flow statements, ES6 features, and asynchronous programming. Implement to manipulate the DOM, handle events, and create dynamic web experiences.
Backend Development with Node.js and Express.js: Middleware and how it's used in Express & Creating RESTful APIs using Express.js.	Setup a development environment, mastering JSX syntax, creating reusable components, and efficiently passing data using props. You'll implement conditional rendering, handle events, manage state, and work with forms effectively. Additionally, you'll optimize performance using lifecycle methods and explore React hooks for efficient state management.
Database Management with MongoDB: Designing and implementing MongoDB schemas.	Perform essential skills in Node.js with this comprehensive course, covering everything from setting up Express.js environments to mastering RESTful API development, asynchronous programming, error handling, debugging, and security best practices, empowering you to build robust and secure applications.
Full Stack Integration: Connecting the frontend and backend components of an application.	Implement essential skills in MongoDB with this course and analyze the setup, schema building, CRUD operations, optimization, authentication, MERN performance, and deployment strategies for React and Node.js applications.

Asynchronous Programming: Basics of asynchronous programming in JavaScript.	
Performance Optimization: Techniques for optimizing the performance of both frontend and backend code.	

Course Duration: 45 Hours

Course Content:

Unit 1: Introduction to FSD, Basics Web Technologies and JavaScript
Intro to Fullstack – What is Frontend? – What is the backend? – Roles & Responsibility for Full Stack Developer – Environment Setup - Introduction to HTML & CSS – Basic elements, DOM- create/delete elements – Selectors –Advanced CSS techniques like flexbox and grid - Introduction to Javascript – Variables, datatypes, and operators – Control flow statements (if-else, for, while, switch)

Unit 2: ES6 Essentials
Introduction to ES6 (let, const, template strings) – Arrow function – Spread operator – destructing – Callback – Promise – JavaScript fundamentals: functions, objects, arrays – Manipulating the DOM with JavaScript – Handling events and user interactions with JavaScript.

Unit 3: React JS
Introduction to React and its features – Setting up a React development environment – JSX syntax and its benefits – Creating React components – Creating reusable React components – Using props to pass data between components – Creating conditional rendering and handling events in React – React State, Event Handling & Forms – Understanding state and its importance in React – Setting state and handling events in React – Using forms and controlled components in React – Handling errors and edge cases in React – Understanding the React lifecycle and its phases – Using lifecycle methods to optimize performance – Introduction to React hooks – Implementing custom hooks in React.

Unit 4: Node JS & Express JS
Introduction to Node.js and its features – Understanding the basics of web servers and HTTP requests Setting up an Express.js development environment – Building a simple Express.js server – Understanding the principles of RESTful APIs – Building CRUD operations with Express.js - Implementing middleware in Express.js – Understanding the Node.js event loop and asynchronous programming – Using callbacks, promises, and async/await in Node.js – Handling errors and debugging Node.js applications – Implementing security best practices in Node.js

Unit 5: MongoDB, Performance optimization & Deployment

Understanding NoSQL databases and MongoDB – Setting up a MongoDB development environment – Building MongoDB schema and models with Mongoose – Using Mongoose to perform CRUD operations in MongoDB – Understanding MongoDB indexing and aggregation –Implementing authentication and authorization with MongoDB – Techniques to optimize the performance of MERN applications, including code splitting and lazy loading – Introduction to CI & CD pipelines – Deploying React applications using hosting services – Deploying Node.js applications with server configurations.

Test Projects:

Use Cases

OVERALL COURSE LEARNING OUTCOME ASSESSMENT CRITERIA AND USE-CASES		
LEARNING OUTCOME	**ASSESSMENT CRITERIA**	**USE-CASES**
• Implement foundational skills in frontend development by mastering HTML/CSS basics, advanced techniques like flexbox and grid layouts, and DOM manipulation in this introductory Full Stack course.	• Developing of Frontend and Backend: Assessing the ability to differentiate between frontend and backend development, and describe the responsibilities associated with each role. • Proficiency in HTML/CSS and DOM Manipulation: Evaluating proficiency in HTML/CSS fundamentals, including creating/deleting DOM elements, and applying advanced CSS techniques like flexbox and grid layouts.	**Use Case 1:** Personal Blog Website. **Scenario:** Sanjana is passionate about cooking and wants to share her recipes, cooking tips, and culinary adventures with the world. She envisions a personal blog website where she can showcase her content in an organized and visually appealing manner. She also aims to enhance her digital presence through the website. **Task:** Design the website layout using HTML and CSS to ensure responsiveness across various devices and screen sizes. Apply responsive design principles such as fluid grids, flexible images, and media queries to adapt the layout dynamically. Create a visually appealing design by incorporating custom fonts, colors, and graphics that reflect Emily's culinary theme. Utilize CSS styling techniques to enhance the aesthetics of the website, including typography, spacing, and transitions.

		Use Case 2: Online Portfolio for a Freelance Graphic Designer. **Scenario:** Gopal is a freelance graphic designer looking to establish a strong online presence and attract potential clients. He wants to showcase his portfolio of design projects, including logos, branding materials, and website designs, in a professional and visually compelling manner. **Task:** Develop a responsive online portfolio website using HTML and CSS to effectively showcase Gopal's design work across various devices and screen sizes. Implement a clean and modern layout that emphasizes visual elements such as images, graphics, and interactive design components. Utilize CSS techniques to create polished animations, transitions, and hover effects that enhance the user experience and engage visitors

• Exhibit essential JavaScript skills, mastering variables, control flow statements, ES6 features, and asynchronous programming. Learn to manipulate the DOM, handle events, and create dynamic web experiences.	• Demonstrate proficiency in acquiring essential knowledge for ES6 like arrow functions, spread operator, rest operator,etc. • Exhibit promises which will help in understanding asynchronous programming. And also get to know about Document Object Model (DOM).	**Use Case 1:** Dynamic Event Booking Website. **Scenario:** Shyam is an event organizer planning a series of workshops and conferences. He wants to create an interactive website where attendees can view upcoming events, register for tickets, and receive event updates in real-time. He aims to build a user-friendly platform that dynamically updates event information, handles user registrations, and provides a seamless booking experience for attendees. **Task:** Develop a dynamic event booking website using JavaScript DOM manipulation to enhance interactivity and functionality. Design a responsive and visually appealing layout that displays upcoming events, event details, and registration forms. Utilize JavaScript to manipulate the DOM elements dynamically, updating event information and user interface elements in response to user actions. Utilize DOM manipulation techniques to dynamically add, remove, or modify HTML elements based on user input or server responses. Integrate form validation using JavaScript to ensure that user input is accurate and complete before submitting registration details. Implement asynchronous requests using AJAX to communicate with the server, fetch event data, and handle registration submissions without reloading the entire page.

		Use Case 2: Interactive Task Management Application. **Scenario:** Jessica is a project manager overseeing multiple teams and tasks. She needs a centralized platform to manage project workflows, assign tasks to team members, and track progress in real-time. Jessica envisions an interactive task management application that allows users to create tasks, set deadlines, assign priorities, and collaborate with team members seamlessly. **Task:** Develop an interactive task management application using JavaScript DOM manipulation to facilitate efficient task tracking and collaboration among team members. Design a responsive and intuitive user interface that enables users to create, edit, and delete tasks dynamically. Utilize JavaScript to manipulate the DOM elements in real-time, updating task lists, statuses, and details based on user interactions and server responses.

• Setup a development environment, mastering JSX syntax, creating reusable components, and efficiently passing data using props. You'll learn to implement conditional rendering, handle events, manage state, and work with forms effectively. Additionally, you'll optimize performance using lifecycle methods and explore React hooks for efficient state management.	• Demonstration of React Concepts: Assess the implementation of JSX syntax, component creation, data handling with props, state management, form usage, and understanding of the React lifecycle. • Proficiency with React Hooks: Evaluate the adeptness in utilizing React hooks, encompassing both built-in and custom hooks, for effective state management and performance optimization in React applications.	**Use Case 1:** Online Learning Platform like MOOC. **Scenario:** A client wants to create an interactive learning platform where students can enroll in courses, access instructional materials, participate in quizzes and assignments, and engage with instructors and peers in discussion forums. The client envisions a modern and user-friendly platform that offers a seamless learning experience with interactive content and personalized learning paths. **Task:** Develop an online learning platform using React.js. Design a responsive and visually appealing user interface that allows students to browse, search, and enroll in courses, as well as track their progress and achievements across different devices and screen sizes. Implement client-side routing using React Router to enable navigation between different views within the application, such as course catalog, course details, user profile, and discussion forums. Utilize state management libraries like Context API to manage application state, including course data, user authentication, and user progress, ensuring consistency and synchronization across different components. Integrate API calls to fetch course data from a backend server (you can use free to use api from the internet), handle CRUD operations. **Use Case 2:** Fitness Tracking Application. **Scenario:** Ravi is a fitness enthusiast who wants to track

		his workouts, set fitness goals, and monitor his progress over time. He envisions a comprehensive fitness tracking application that allows him to log his exercises, record his nutrition intake, track his weight and body measurements, and visualize his progress through charts and graphs. Ravi wants a user-friendly platform that offers a seamless experience across different devices and provides actionable insights to help him achieve his fitness goals effectively. **Task:** Design a responsive and visually appealing user interface that allows users to log workouts, record nutrition intake, track weight and body measurements, and view progress charts and graphs across different devices and screen sizes. Utilize React components to modularize the application's UI elements, including workout logs, nutrition tracker, progress charts, and user settings, making it easier to manage and scale the application. Implement client-side routing using React Router to enable navigation between different views within the application, such as workout log, nutrition tracker, progress dashboard, and user profile pages. Utilize state management libraries like Context API to manage application state, including user data, fitness logs, and progress metrics, ensuring consistency and synchronization across different components. Integrate third-party APIs or libraries for fitness tracking, nutrition data, and weight management to provide users with accurate and up-to-date information for tracking their fitness goals.

• Perform essential skills in Node.js with this comprehensive course, covering everything from setting up Express.js environments to mastering RESTful API development, asynchronous programming, error handling, debugging, and security best practices, empowering you to build robust and secure applications.	• Delve into the backend technology and get familiar with Node js by making a server. • Gain hands-on experience in implementing and building RESTful APIs. • Acquire a deep understanding of streams in Node js. And also the file system and operating system methods.	**Use Case 1:** Online Marketplace API. **Scenario:** Samantha is an entrepreneur who wants to create an online marketplace platform where users can buy and sell products within specific categories. She envisions a platform similar to flipkart or amazon, where sellers can create listings for their products, manage inventory, and communicate with buyers, while buyers can browse products, make purchases, and provide feedback. **Task:** Develop an online marketplace platform using Node.js and Express.js to facilitate buying and selling of products for users. Design a robust and scalable backend architecture that handles user authentication, product listings, inventory management, order processing, and communication between buyers and sellers. Implement RESTful API endpoints using Express.js to handle CRUD operations for managing users, products, orders, and transactions, ensuring secure data exchange between the client and server. Utilize middleware functions in Express.js to implement authentication and authorization. **Use Case 2:** Task Management API. **Scenario:** Emily is a project manager overseeing multiple teams and projects within her organization. She needs a centralized platform to manage tasks, deadlines, and team collaboration efficiently. Emily envisions a RESTful API that her team can integrate into their existing project

		management tools, allowing them to create, update, and track tasks programmatically. **Task:** Develop a task management API using Node.js and Express.js to provide CRUD operations for managing tasks and facilitating team collaboration for users like Emily. Design a robust and scalable backend server using Node.js and Express.js to handle HTTP requests, route them to the appropriate endpoints, and interact with the database. Implement RESTful API endpoints using Express.js to handle CRUD operations for managing tasks, task assignments, deadlines, priorities, and task statuses, ensuring consistent and predictable behavior for client applications. Utilize middleware functions in Express.js to implement authentication and authorization mechanisms. And use databases like Firebase as it is easier to set up for small tasks.
• Implement essential skills in MongoDB with this course. Learn setup, schema building, CRUD operations, optimization, authentication, MERN performance, and deployment strategies for React and Node.js applications.	• Ability to explain the concept of document-oriented databases and contrast them with traditional relational databases. • Proficiency in understanding MongoDB's data model, including collections, documents, and fields. • Familiarity with MongoDB's query language (MongoDB Query Language) and its syntax for CRUD (Create, Read, Update, Delete) operations. • Ability to explain the concept of schema-less data models and understand how they differ from schema-based models.	**Use Case 1:** Online Bookstore Application. **Scenario:** Sunny is an entrepreneur who wants to launch an online bookstore where users can browse, purchase, and review books from various genres. He envisions a platform similar to Amazon, offering a vast selection of books, personalized recommendations, and seamless checkout experiences. He needs a scalable and flexible database solution to store book information, user profiles, order details, and reviews efficiently

	• Proficiency in designing and working with flexible schemas in MongoDB, including dynamic schema changes and performing CRUD operations.	**Task:** Develop an online bookstore application using MongoDB to manage book data and facilitate e-commerce transactions for users like Alex. Design a backend architecture that integrates MongoDB as the primary database to store and manage book information, user profiles, order details, and reviews. Implement MongoDB collections for storing data entities such as books, users, orders, and reviews. Utilize MongoDB's flexible schema design to accommodate diverse book metadata such as title, author, genre, publication date, ISBN, and cover image URL, allowing for easy querying and indexing of book data. Implement CRUD operations using MongoDB's native drivers or an ORM (Object-Relational Mapping) library like Mongoose to interact with the database, allowing users to browse, search, and purchase books seamlessly. **Use Case 2:** Online Auction Platform. **Scenario:** Mark is an entrepreneur who wants to create an online auction platform where users can buy and sell a variety of items through bidding. He envisions a platform similar to eBay, where users can list items for auction, place bids on items, and monitor auction progress in real-time. Mark needs a scalable and flexible database solution to store item listings, bid history, user profiles, and transaction details securely.

		Task: Design a backend architecture that integrates MongoDB as the primary database to store data entities such as items, bids, users, transactions, and relationships between them. Implement MongoDB collections for storing item listings, bid history, user profiles, transaction details, and other auction-related data, ensuring data consistency and scalability as the platform grows. Utilize MongoDB's flexible schema design to accommodate diverse item data such as title, description, images, starting price, bid history, and user information, allowing for easy querying and indexing of auction data. Implement CRUD operations using MongoDB's native drivers or an ORM (Object-Relational Mapping) library like Mongoose to interact with the database, allowing users to list items for auction, place bids on items, and manage their auction activities seamlessly. Utilize MongoDB's aggregation framework to perform complex queries and aggregations, such as calculating highest bids, generating auction analytics, and identifying trending items based on user activity.