n8de
NODE EIGHT

Digital Skills Department

Fullstack Web Development Curriculum

Overview

The Full-Stack Web Development Training Program is designed to equip participants with the skills and knowledge required to excel as proficient web developers. This program enables them to create both the front-end and back-end components of modern web applications.

Throughout this program, participants will delve deeply into the realms of both front-end and back-end development, acquiring a holistic understanding of building seamless user experiences and robust server-side logic. They will explore a range of technologies and tools that drive the web, from building captivating user interfaces with HTML, CSS, and JavaScript, to designing and executing powerful APIs, databases, and server architecture.

This journey is an opportunity for growth, learning, and transformation. Participants will gain the expertise to innovate, create, and transform their ideas into fully functional web applications. The skills acquired are not only in high demand but also offer the freedom to undertake independent projects, collaborate effectively, and stand out in the dynamic realm of web development.

Program Objectives

A graduate of this program will be proficient in the following areas:

1. Understand the basics of web development: client-side vs. server-side, HTTP, APIs, etc.
2. Explore the MERN stack components: MongoDB, Express.js, React, and Node.js.
3. Set up the development environment: installing required tools, editors, and version control (Git).
4. Learn the fundamentals of Node.js: asynchronous programming, event loop, modules, etc.
5. Create RESTful APIs using Express.js: routing, middleware, request handling.
6. Interact with databases using MongoDB: CRUD operations, data modeling, validation.
7. Gain a deep understanding of React: components, JSX, state, props, and hooks.
8. Build responsive user interfaces: layout design, forms, and interaction patterns.
9. Manage state and data flow: global state management with Context API or Redux.
10. Set up communication between the frontend and backend using RESTful APIs.
11. Handle authentication and authorization: JWT, user registration, and login.
12. Secure API endpoints: input validation, authentication middleware, error handling.
13. Deploy applications using platforms like Heroku, Netlify, or Vercel.
14. Collaborate on projects using Git: branching, merging, pull requests.

Program Information

Estimated Time

6 months at 20hrs a week

Prerequisites

Basic computer skills such as managing files, running programs and using the web broswer

Week 1

Introduction to Web Development, HTML5 and CSS3

This course provides a comprehensive introduction to the fundamental concepts of web development, focusing on HTML5 and CSS3 as the building blocks for creating modern, responsive, and visually appealing websites. Through hands-on exercises, projects, and practical examples, students will gain a solid foundation in web development and design.

Lesson 1

Web Architecture and Client-Server Model

- Explain the concept of the client-server model where web browsers (clients) communicate with web servers to request and receive web content.

Lesson 2

HTTP and HTTPS Protocols

- Describe the HTTP (Hypertext Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure) protocols that facilitate communication between clients and servers, emphasizing the importance of security for data transmission.

Lesson 3

HTML5 (Hypertext Markup Language)

- Introduce the fundamental structure of HTML documents, including the <!DOCTYPE> declaration, <html>, <head>, and <body> elements.
- Teach about various HTML elements like headings (<h1>, <h2>, etc.), paragraphs (<p>), lists (<ul>, <ol>, <li>), links (<a>), and images (<img>).

Lesson 4

CSS3 (Cascading Style Sheets)

- Explain the role of CSS in styling web content, including separation of content and presentation.
- Introduce CSS selectors and basic styling properties like color, font-family, background, and border.
- Touch upon the concept of specificity in CSS.

Week 2

Semantic HTML and Responsive Design

By the end of these two weeks, students should have a solid foundation in HTML5 and CSS3, as well as the ability to create basic web pages and apply responsive design principles. This sets them up for further exploration into more advanced web development topics.

Lesson 1

Semantic HTML

- Explain the importance of semantic HTML in enhancing accessibility, SEO, and overall structure.
- Introduce semantic elements like <header>, <nav>, <main>, <article>, <section>, and <footer>.

Lesson 2

CSS Styling

- Dive deeper into CSS styling by covering advanced properties like margin, padding, display, and position.
- Introduce the concept of the CSS box model and how it affects element dimensions.

Lesson 3

Responsive Design with Flexbox and Grid

- Teach the concept of responsive design and the need for adapting layouts to different devices and screen sizes.
- Introduce Flexbox for building flexible and dynamic layouts.
- Cover CSS Grid for creating grid-based layouts that respond effectively to various screen sizes.

Mini Project

Personal Portfolio Website

Combine the knowledge gained in these two weeks to create a simple personal webpage

Todo

1. **HTML:**
   - Structure the webpage using semantic HTML elements.
   - Include sections for a header, navigation menu, main content area, and footer.
2. **CSS:**
   - Apply CSS styling to elements, utilizing properties learned in both weeks.
   - Ensure a consistent and visually appealing design.
3. **Responsive Design:**
   - Implement responsive design techniques using Flexbox and Grid.
   - Ensure that the webpage layout adjusts gracefully on different devices and screen sizes.

Week 3

JavaScript Fundamentals

Building upon the foundation of HTML and CSS, now let's delve into the world of JavaScript, which adds interactivity and dynamic behavior to web pages.

Lesson 1

Introduction to JavaScript

- Explain the role of JavaScript in adding interactivity to web pages.
- Briefly discuss the history and evolution of JavaScript.

Lesson 2

JavaScript Syntax and Variables

- Cover basic syntax rules and conventions.
- Introduce variables (var, let, const) and their scopes.

Lesson 3

Data Types and Operators

- Introduce primitive data types (e.g., strings, numbers, booleans) and composite data types (e.g., arrays, objects).
- Explain arithmetic, comparison, logical, and assignment operators.

Lesson 4

Control Structures

- Teach if statements, else clauses, and the concept of conditional execution.
- Introduce switch statements for multi-branch decision-making.

Week 4

Intermediate JavaScript

By the end of these two weeks, students should be comfortable with JavaScript syntax, basic programming concepts, and asynchronous programming using callbacks and Promises. The mini project will allow them to apply their knowledge to a practical application, solidifying their understanding of these concepts. This foundation will set them up for more advanced topics in JavaScript and web development.

Lesson 1

Functions and Scope

- Explain the concept of functions and their importance in code organization.
- Cover function declarations, expressions, and anonymous functions.

  Discuss function scope, local vs. global variables.

Lesson 2

Closures

- Define closures and explain how they work.
- Discuss practical use cases for closures in JavaScript.

Lesson 3

Asynchronous Programming

- Introduce asynchronous programming and the event loop.
- Explain why asynchronous operations are essential for non-blocking interactions.

Lesson 4

Callbacks and Promises

- Discuss callbacks as a way to handle asynchronous operations.
- Introduce Promises as a more structured approach to handling async code.
- Discuss error handling using .catch() with Promises.

Mini Project

Interactive Quiz Application

Apply the JavaScript knowledge gained to build a simple interactive quiz application

Todo

1. **HTML**:
    - Create the structure of the quiz using HTML elements.
    - Include question containers, answer choices, and a submit button.
2. **CSS**:
    - Apply basic styling to the quiz layout to make it visually appealing.
3. **JavaScript**:
    - Implement the quiz logic using JavaScript functions.
    - Use data structures like arrays or objects to store quiz questions and answers.
    - Handle user interactions, calculate scores, and display results.

Week 5

Introduction to Modern JavaScript Tools

In this phase of the course, you'll introduce students to modern JavaScript tools and frameworks, focusing on React.js for building interactive web applications.

Lesson 1

npm and Package Management

- Explain npm (Node Package Manager) and its role in managing dependencies.
- Show how to install packages, manage versions, and create package.json files.

Lesson 2

Introduction to React.js

- Explain the concept of front-end libraries and frameworks.
- Introduce React.js and its virtual DOM concept.

Lesson 3

JSX and Components

- Teach JSX (JavaScript XML) syntax for writing React components.
- Explain the idea of building UIs as reusable components.

Lesson 4

Props and Component Communication

- Introduce the concept of props (properties) for passing data to components.
- Show how parent and child components can communicate using props.

Week 6

Building Interactive Web Applications with React.js

By the end of these two weeks, students should have a solid foundation in HTML5 and CSS3, as well as the ability to create basic web pages and apply responsive design principles. This sets them up for further exploration into more advanced web development topics.

Lesson 1

State and React Hooks

- Introduce component state and why it's crucial for dynamic behavior.
- Cover React Hooks, focusing on useState for managing state within functional components.

Lesson 2

Managing Forms and User Input

- Explain how to handle user input and forms using controlled components in React.

Lesson 3

Conditional Rendering and Lists

- Teach conditional rendering techniques to show or hide components based on conditions.
- Introduce mapping over arrays to dynamically render lists of elements.

Lesson 4

Context and State Management

- Explain the Context API for sharing state between components without prop drilling.
- Show how to create and use a context provider and consumer.

Mini Project

Building a To-Do List Application with React

Apply the knowledge gained to create an interactive to-do list application

Todo

1. **Component Structure:**
   - Design a component structure for the to-do list application.
   - Identify components for the task list, input form, and individual tasks.
2. **State Management:**
   - Use React Hooks to manage the state of the task list.
   - Implement functions to add, update, and delete tasks.
3. **User Interaction:**
   - Allow users to mark tasks as completed and filter tasks based on their status.
4. **Styling and Responsiveness:**
   - Apply CSS to style the to-do list application and make it responsive.

Week 7

Introduction to Back-End Development

Now that your students have a solid understanding of front-end development using React.js, it's time to introduce them to back-end development using Node.js and Express.js.

Lesson 1

Server-Side Programming

- Explain the difference between front-end and back-end development.
- Discuss the role of back-end languages in handling server-side logic, data storage, and communication.

Lesson 2

Introduction to Node.js

- Introduce Node.js as a server-side JavaScript runtime environment.
- Discuss the benefits of using JavaScript for both front-end and back-end development.

Lesson 3

Basic HTTP Concepts

- Review the HTTP protocol and its methods (GET, POST, PUT, DELETE).
- Discuss the concepts of requests and responses in the context of server-side programming.

Week 8

Introduction to Express.js

By the end of these two weeks, students should have a foundational understanding of back-end development concepts, Node.js, and Express.js. The mini project will give them practical experience in setting up a basic back-end server and creating routes for API endpoints. This knowledge will be essential as they progress to more complex back-end development topics and full-stack application development.

Lesson 1

What is Express.js

- Introduce Express.js as a popular web application framework for Node.js.
- Explain how Express simplifies routing, middleware, and handling HTTP requests.

Lesson 2

Setting Up an Express Server

- Guide students through setting up a basic Express server.
- Show how to install Express, create a server instance, and listen for incoming requests.

Lesson 3

Creating Routes

- Explain the concept of routes and their role in handling different endpoints.
- Demonstrate how to create routes for various HTTP methods.

Lesson 4

Middleware and Error Handling

- Introduce middleware and how it can be used to modify request and response objects.
- Discuss error handling and middleware functions for dealing with errors.

Mini Project

Setting Up an Express Server and Creating API Routes

In this mini project, students will apply their knowledge to create a simple Express server and set up routes for basic API endpoints:

Todo

1. **Express Server Setup:**
    - Help students install Node.js and Express.
    - Guide them through creating a basic Express server file.
2. **API Routes:**
    - Teach students how to define routes for different endpoints (e.g., GET, POST) using Express.
    - Implement basic CRUD operations (create, read, update, delete) for a mock data store.
3. **Testing API Endpoints:**
    - Show students how to test their API endpoints using tools like Postman or browser URLs.

Week 9

Introduction to Relational Databases and Data Modeling

In this phase of the course, you'll introduce students to the concepts of relational databases, the importance of data modeling, and how to interact with databases using SQL.

Lesson 1

Relational Databases Overview

- Explain the concept of relational databases and their significance in web development.
- Introduce the key components of a relational database: tables, rows, columns, and relationships.

Lesson 2

Data Modeling and Entity-Relationship Diagrams

- Teach the basics of data modeling using ER diagrams.
- Explain how to define entities, attributes, and relationships between tables.

Lesson 3

Normalization

- Discuss the importance of normalization in database design to minimize redundancy and improve data integrity.
- Introduce different normalization forms (1NF, 2NF, 3NF) and their benefits.

Week 10

Querying and Database Design

By the end of these two weeks, students should have a strong understanding of relational databases, data modeling, and SQL querying. The mini project will give them hands-on experience in designing a database schema and writing SQL queries to manipulate data. This foundation will prepare them for more advanced topics in database management and backend development.

Lesson 1

Introduction to SQL

- Introduce SQL (Structured Query Language) as the language used to manage and manipulate relational databases.
- Explain the role of SQL in querying and interacting with databases.

Lesson 2

Basic SQL Queries

- Teach basic SQL commands: SELECT, INSERT, UPDATE, DELETE.
- Show how to retrieve data from a single table using the SELECT statement.

Lesson 3

Advanced SQL Queries and Joins

- Introduce JOIN operations to combine data from multiple tables.
- Teach different types of joins: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN.

Lesson 4

Database Design Principles

- Discuss principles of good database design, including choosing appropriate data types, primary and foreign keys, and indexes.

Mini Project

Designing a Relational Database and Writing SQL Queries

In this mini project, students will apply their knowledge of relational databases and SQL to design a simple database schema and write SQL queries:

Todo

1. **Database Schema Design:**
    - Instruct students to design a schema for a simple relational database using ER diagrams.
    - Define tables, their attributes, primary keys, and relationships.
2. **Creating the Database:**
    - Help students set up a database using a database management system like MySQL or SQLite.
3. **Writing SQL Queries:**
    - Assign tasks for students to write SQL queries to retrieve, insert, update, and delete data.
    - Include examples of basic and advanced queries involving joins.

Week 11

Introduction to NoSQL Databases and MongoDB

Welcome to the next phase of our journey! In these weeks, we'll explore NoSQL databases, particularly MongoDB, and learn how to develop APIs for data management.

Lesson 1

NoSQL Databases Overview

- Explain the concept of NoSQL databases and their differences from traditional relational databases.
- Discuss scenarios where NoSQL databases are a better fit.

Lesson 2

Introduction to MongoDB

- Introduce MongoDB as a popular NoSQL database.
- Highlight its document-based structure, collections, and documents.

Lesson 3

Document Structure in MongoDB

- Dive into MongoDB's JSON-like documents and their flexibility.
- Explain how data can be nested and complex within a document.

Week 12

MongoDB Query Language (MQL) and API Development

By the end of these two weeks, you'll have a deep understanding of NoSQL databases, particularly MongoDB, and how to build powerful APIs using Express.js. The mini project will showcase your ability to create a real-world API that interacts with a database. You're now on the path to becoming proficient full-stack developers who can handle both front-end and back-end development with ease! Keep up the great work!

Lesson 1

MongoDB Query Language (MQL)

- Introduce MQL for CRUD operations: insert, find, update, and delete.
- Show how to query MongoDB to retrieve and manipulate data.

Lesson 2

Building RESTful APIs with Express.js and MongoDB

- Explain the concept of RESTful APIs and their importance in modern web development.
- Guide students in creating APIs using Express.js to interact with MongoDB for data storage.

Lesson 3

Routing and Controller Logic

- Teach how to define routes for various API endpoints using Express.
- Implement controller functions that handle the logic for CRUD operations.

Lesson 4

Error Handling and Validation

- Discuss error-handling techniques for API requests.
- Show how to validate incoming data before processing.

Mini Project

Developing a Blogging Platform API with MongoDB

Let's put our knowledge into practice with a dynamic project: building a blogging platform API!

Todo

1. **API Setup:**
    - Guide you in setting up an Express server and connecting it to MongoDB.
    - Create routes for managing articles: creating, retrieving, updating, and deleting.
2. **MongoDB Integration:**
    - Implement MQL queries to interact with MongoDB collections.
    - Use MongoDB to store and manage article data.
3. **RESTful API Endpoints:**
    - Define routes for articles, allowing users to perform CRUD operations.
    - Implement features like searching for articles and retrieving articles by categories or tags.
4. **Error Handling and Validation:**
    - Ensure that your API handles errors gracefully and validates input data.

Week 13

Web Security and Best Practices

In these weeks, we'll dive into the critical topic of web security and learn how to implement user authentication and authorization to keep our applications safe.

Lesson 1

Understanding Web Vulnerabilities

- Discuss common web vulnerabilities such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and SQL injection.
- Emphasize the importance of security in modern web applications.

Lesson 2

Best Practices for Web Security

- Introduce security best practices such as input validation, sanitization, and output encoding.
- Discuss using HTTPS, secure cookies, and setting proper headers to prevent attacks.

Lesson 3

Implementing Security Measures

- Explain the role of security libraries and tools in enhancing application security.
- Explore security-focused packages like Helmet.js for Express applications.

Week 14

User Authentication and Authorization

By the end of these two weeks, you'll be equipped with essential knowledge about web security, user authentication, and authorization. The mini-project will challenge you to apply this knowledge by securing your API endpoints using authentication middleware. With security at the forefront of your development practices, you're on your way to building robust and safe applications. Keep up the great work and stay security-aware!

Lesson 1

Introduction to User Authentication

- Define authentication and its role in verifying user identities.
- Discuss the concept of passwords, hashes, and salt for secure password storage.

Lesson 2

Using JSON Web Tokens (JWT)

- Introduce JSON Web Tokens as a way to securely transmit information between parties.
- Show how JWT can be used for stateless user authentication.

Lesson 3

User Authorization and Roles

- Explain the distinction between authentication and authorization.
- Teach how to implement role-based authorization to control access to different parts of the application.

Mini Project

Secure API Endpoints with Authentication Middleware

It's time to put on your security hats and implement what you've learned in a real project!

Todo

1. **Setting Up JWT Authentication:**
    - Guide you through integrating JWT authentication into your Express application.
    - Show how to generate and validate tokens for user sessions.
2. **Implementing Authentication Middleware:**
    - Create middleware functions that ensure only authenticated users can access specific routes.
    - Build middleware for role-based authorization to control access based on user roles.
3. **Testing Security Measures:**
    - Walk you through testing your authentication and authorization setup to ensure its effectiveness.

Week 15

Front-End Performance Optimization

Front-end performance optimization is a critical aspect of web development that focuses on improving the speed, responsiveness, and overall user experience of a website or web application. It involves various techniques and best practices aimed at reducing page load times, minimizing rendering delays, and ensuring smooth interactions for users. Here are some key topics and strategies often covered in Week 15 of a curriculum focused on front-end performance optimization:

Lesson 1

Web Performance Importance

- Discuss the impact of website speed on user experience and conversion rates.
- Introduce the "three-second rule" and its significance.

Lesson 2

Image Optimization and Lazy Loading

- Teach techniques for optimizing images, such as compression and resizing.
- Explain lazy loading images to improve initial page load times.

Lesson 3

Code Splitting and Bundle Optimization

- Introduce code splitting to load only what's needed.
- Discuss tools and techniques to optimize JavaScript bundles.

Week 16

Responsive Design and Media Queries

In Week 16 of a curriculum focused on responsive design and media queries, you would likely delve deeper into the concepts and implementation details of these crucial aspects of web development. Here's what you might cover during this week:

Lesson 1

Responsive Design Principles

- Reinforce the importance of responsive design for a seamless experience across devices.
- Discuss fluid grids, flexible images, and media queries.

Lesson 2

Media Queries in Depth

- Dive deeper into media queries, breakpoints, and how to target specific screen sizes.
- Discuss the mobile-first approach to responsive design.

Lesson 3

Responsive Images

- Explain techniques like using the srcset attribute and picture element to serve different images based on screen size.

Mini Project

Optimizing Performance and Responsive Design

Get ready to apply your newfound knowledge to your existing projects!

Todo

1. **Front-End Performance Optimization:**
    - Identify areas in your to-do list application that can be optimized for better performance.
    - Implement image optimization, lazy loading, and code-splitting techniques.
2. **Responsive Design Implementation:**
    - Use media queries to make your to-do list application responsive.
    - Ensure that it looks and works seamlessly on various screen sizes.

Week 17

Advanced State Management with Redux

In Week 17 of a curriculum focused on advanced state management with Redux, you would delve into more complex topics related to state management in large-scale web applications. Redux is a popular JavaScript library used to manage application state in a predictable and efficient manner. Here's what you might cover during this week:

Lesson 1

Introduction to Redux

- Discuss the need for centralized state management in larger applications.
- Introduce Redux as a predictable state container for managing complex states.

Lesson 2

Redux Principles

- Explain the key principles of Redux: single source of truth, state is read-only, changes are made through pure functions.
- Discuss the Redux store, actions, and reducers.

Lesson 3

Actions and Reducers

- Dive into creating actions to describe state changes.
- Implement reducers to specify how state changes in response to actions.

Lesson 4

Redux Middleware

- Cover other middleware options like Redux Saga or Redux Observable for handling complex asynchronous flows.
- Explain how middleware intercepts actions and can modify them before they reach reducers.

Lesson 5

Normalizing State Shape

- Introduce the concept of normalizing the state to optimize performance and simplify data management.
- Explain how to use libraries like normalizr to normalize and denormalize data.

Week 18

React Router

In Week 18 of a curriculum focused on React Router, you would dive into the world of client-side routing in React applications. React Router is a popular library that enables you to create dynamic and navigable user interfaces by managing different routes within a single-page application. Here's what you might cover this week

Lesson 1

Introduction to React Router

- Explain the concept of client-side routing and its benefits for single-page applications.
- Introduce React Router and its core components: BrowserRouter, Route, and Link.

Lesson 2

Basic Routing

- Demonstrate how to set up basic routes using the Route component.

  Cover route matching, rendering components, and handling 404 not-found scenarios.

Lesson 3

Nested Routes

- Discuss the concept of nested routes and their use cases.
- Show how to create parent and child route structures to manage complex UI hierarchies.

Lesson 4

Route Parameters and URL Parameters

- Introduce route parameters as a way to make dynamic routes.

  Explain how to access URL parameters and use them in component rendering.

Lesson 5

Query Parameters

- Discuss query parameters and their role in passing data through URLs.
- Show how to access and parse query parameters using React Router.

Lesson 6

Programmatic Navigation

- Explain how to navigate between routes programmatically using the history object or hooks.
- Cover scenarios like redirecting users and preserving query parameters.

Lesson 7

Route Guards and Authentication

- Introduce the concept of route guards for protecting routes based on user authentication.
- Explain how to implement basic authentication checks using higher-order components or hooks.

Lesson 8

Route Transitions and Animations

- Cover techniques for adding transitions and animations when navigating between routes.
- Discuss libraries like react-transition-group for creating smooth route transitions.

Lesson 9

Lazy Loading and Code Splitting

- Explain the benefits of lazy loading and code splitting for optimizing application performance.
- Demonstrate how to use React Router's built-in lazy loading capabilities.

Mini Project

Online Blog Platform

Create an online blog platform where users can view and create blog posts. The platform should have multiple routes for different views, including a homepage, individual blog post pages, and a create/edit post page.

Todo

1. **Homepage**: Display a list of featured blog posts. Clicking on a post should navigate to the individual post page.
2. **Individual Post Page:** Show the full content of a blog post, including the title and body. Allow users to navigate back to the homepage.
3. **Create/Edit Post Page**: Implement a form for creating/editing blog posts. Use query parameters to differentiate between creating and editing. After submission, navigate back to the homepage.
4. **Navigation**: Include a navigation bar with links to the homepage and create post page.
5. **Route Guards**: Add a simple authentication system. Users need to log in to access the create/edit post page. Implement a route guard to protect this route.
6. **Route Transitions**: Apply smooth route transitions and animations using the react-transition-group library.

Week 19

Integrating Front-End and Back-End

students would learn how to connect the user interface they've been working on to a server and database on the back end. This week would cover topics like setting up server environments, handling API requests, and managing data interactions

Lesson 1

Integrating React with Express APIs

- Recap the Express API you've set up and its endpoints.
- Integrate your React application with the Express API by making API requests.
- Display data from your API responses in your React components.

Lesson 2

Making API Calls in React

- Explore the native fetch API for making asynchronous requests in React.

  Introduce the Axios library as an alternative for more streamlined API calls.
- Compare and contrast the features of fetch and Axios.

Lesson 3

Handling Data Flow

- Discuss strategies for managing the flow of data between your front-end and back-end.
- Implement loading states to improve the user experience during data fetching.
- Explore error handling techniques for API responses and displaying appropriate error messages.

Lesson 4

Pagination and Data Display

- Cover techniques for handling pagination and displaying large datasets.
- Implement pagination controls and fetch only the necessary data chunks.

Lesson 5

User Authentication Flow

- Review authentication mechanisms and JWT from earlier in the curriculum.
- Implement a user authentication flow, where the front end interacts with the back end for user registration and login.

Lesson 6

User Input and Form Submissions

- Discuss handling user input and form submissions in React.
- Create forms for interactions like creating new items, sending comments, or updating user profiles.

Lesson 7

Real-Time Updates with WebSockets

- Introduce WebSockets as a way to achieve real-time communication.
- Implement a simple chat or notification system using WebSockets to showcase real-time updates.

Lesson 8

Cross-Origin Resource Sharing (CORS)

- Explore the importance of CORS and security considerations.
- Configure CORS on your back-end to ensure that your API is accessible from your React front-end.

Lesson 9

Advanced API Design and Documentation

- Discuss more advanced API design considerations, such as versioning and handling complex relationships.
- Touch upon the importance of API documentation and tools like Swagger or Postman.

Mini Project

Online E-Commerce Store

Create an online e-commerce store where users can browse, search, and purchase products. The project will involve building both the front-end user interface and the back-end API to manage products, user accounts, and orders.

Todo

1. **Product Listings**: Display a list of products with their names, images, prices, and brief descriptions.
2. **Product Details**: Implement individual product pages with detailed information about each product.
3. **Search and Filtering**: Allow users to search for products and filter them based on categories, prices, etc.
4. **Shopping Cart**: Implement a shopping cart where users can add products, view cart contents, and proceed to checkout.
5. **User Authentication**: Enable user registration and login to track user profiles and order history.
6. **Order Processing**: Implement the ability to place orders and track order statuses.
7. **Admin Dashboard**: Create an admin panel to manage products, inventory, and orders.

Week 20

Real-Time Applications with WebSockets

By completing this week's curriculum, students will gain practical experience in building real-time applications using WebSockets. They'll understand the concepts of real-time communication, broadcasting messages, and handling user presence. These skills are valuable for creating dynamic and engaging web applications that provide instant updates and interactions for users.

Lesson 1

Introduction to Real-Time Communication

- Explain the concept of real-time communication and its applications.
- Discuss scenarios where real-time updates are essential.

Lesson 2

WebSockets and Socket.IO

- Dive into WebSockets as a protocol for real-time communication.
- Explore Socket.IO as a library that simplifies WebSockets usage.

Lesson 3

Implementing Real-Time Notifications

- Discuss scenarios where real-time notifications are valuable in applications.
- Implement real-time notifications using WebSockets, notifying users of new events instantly.

Lesson 4

Real-Time Collaborative Editing

- Introduce the concept of collaborative editing and its use in applications like Google Docs.
- Implement a simple collaborative text editor using WebSockets for real-time synchronization.

Lesson 5

Broadcasting Updates Across Clients

- Explore techniques for broadcasting updates to multiple clients using WebSockets.
- Implement a feature where user actions trigger real-time updates for all connected clients.

Lesson 6

Handling Presence and Online Status

- Discuss the challenges of tracking user presence and online/offline status.
- Implement a feature that shows when users are online and actively interacting with the application.

Lesson 7

Real-Time Gaming or Polling App

- Choose between creating a real-time gaming application or a live polling app.
- Implement the chosen project using WebSockets to showcase real-time interactions and updates.

Lesson 8

Advanced Socket.IO Features

- Dive deeper into Socket.IO's advanced features like namespaces and rooms.
- Discuss use cases for these features and implement them in your real-time application.

Lesson 9

Error Handling and Resilience

- Discuss strategies for handling errors and ensuring the resilience of real-time applications.
- Explore techniques like reconnection strategies and message acknowledgment.

Lesson 10

Scaling Real-Time Applications

- Introduce the challenges of scaling real-time applications for a larger user base.
- Discuss strategies for scaling WebSocket servers and handling increased traffic.

Mini Project

Live Auction Platform

Create a real-time live auction platform where users can bid on items in real time. This project will involve building a front-end interface for users to participate in auctions and a back-end WebSocket server to handle real-time bidding updates.

Todo

1. **Auction Listings**: Display a list of auction items with details like name, description, and current highest bid.
2. **Real-Time Bidding**: Allow users to place bids on items, with real-time updates for all participants.
3. **Countdown Timer**: Implement a countdown timer for each auction item to create a sense of urgency.
4. **Bid History**: Show a bid history for each item, displaying the usernames and amounts of recent bids.
5. **Outbid Notifications**: Notify users in real time when they've been outbid on an item.
6. **Auction End**: Implement logic to determine the winner when an auction ends, considering the highest bid.

Week 21

Server-Side Rendering (SSR)

By completing this week's curriculum, students will gain practical experience in building applications with server-side rendering using Next.js. They'll understand the benefits and challenges of SSR, dynamic routing, data fetching, and SEO optimization. These skills are crucial for creating web applications that prioritize performance, user experience, and search engine visibility.

Lesson 1

Understanding SSR

- Explain the difference between client-side rendering (CSR) and server-side rendering (SSR).
- Discuss the benefits of SSR in terms of performance, SEO, and initial page load.

Lesson 2

Introduction to Next.js

- Introduce Next.js as a framework for server-side rendering with React.
- Discuss its features, routing, and server-side data fetching.

Lesson 3

Converting to SSR with Next.js

- Guide you in converting a React application to use Next.js for server-side rendering.
- Show how to migrate components and set up server-side rendering.

Lesson 4

Next.js Routing and Data Fetching

- Introduce Next.js routing and its differences from traditional React routing.
- Cover the basics of data fetching on the server side and how it differs from client-side data fetching.

Lesson 5

Dynamic Pages and Routing Parameters

- Explore dynamic routing in Next.js for creating pages with dynamic content.
- Implement pages that use routing parameters to fetch specific data based on URLs.

Lesson 6

SEO Optimization with SSR

- Discuss the SEO benefits of server-side rendering and its impact on search engine rankings.
- Explore how SSR ensures that search engines can index content effectively.

Lesson 7

Handling Authentication in SSR

- Discuss strategies for handling user authentication in server-side-rendered applications.
- Implement user authentication and explore how it interacts with server-side rendering.

Lesson 8

Performance Considerations with SSR

- Explore the performance implications of server-side rendering.

  Discuss scenarios where SSR might be more or less performant compared to client-side rendering.

Lesson 9

Code Splitting and SSR

- Discuss code splitting in the context of SSR.
- Explore techniques to achieve efficient code splitting for improved performance.

Lesson 10

Server-Side Data Fetching Strategies

- Compare and contrast different server-side data fetching strategies.
- Explore scenarios where static site generation (SSG) might be more suitable than traditional SSR.

Mini Project

Social Media Feed with SSR

Create a social media feed application that utilizes Server-Side Rendering (SSR) to ensure fast loading, SEO optimization, and real-time updates. Users can post messages, like and comment on posts, and view their feed with real-time updates.

Todo

1. **User Authentication**: Allow users to sign up, log in, and authenticate their sessions.
2. **Posting and Commenting**: Implement the ability to post messages and comment on posts.
3. **Real-Time Feed Updates**: Utilize SSR to load the initial feed content and then use real-time updates to show new posts and interactions.
4. **User Profiles**: Create user profiles where users can view their posts, comments, and liked posts.
5. **Responsive Design**: Ensure the application is responsive and user-friendly across various devices.

Week 22

RESTful API Testing

By completing this week's curriculum, students will gain practical experience in testing RESTful APIs for reliability and functionality. They'll demonstrate proficiency in writing unit tests, integration tests, and API documentation, which are crucial skills for ensuring the quality of software projects and maintaining a robust codebase.

Lesson 1

Principles of API Testing:

- Discuss the importance of testing APIs for reliability and functionality.
- Explain unit testing, integration testing, and end-to-end testing.

Lesson 2

Unit Testing Your Express APIs

- Dive into unit testing principles and how to write tests for your Express API routes.
- Use testing libraries like Jest and Supertest to write and run tests.

Lesson 3

Integration Testing for Express APIs

- Discuss integration testing and its role in ensuring the correct behavior of API endpoints.
- Explore testing libraries like Jest and Supertest for writing and running integration tests.

Lesson 4

Mocking Dependencies in Tests

- Introduce the concept of mocking external dependencies for more controlled testing.
- Demonstrate how to use mock libraries to simulate behavior in integration tests.

Lesson 5

End-to-End Testing with Cypress

- Discuss the concept of end-to-end testing for ensuring the complete functionality of applications.
- Introduce Cypress as a tool for writing and running end-to-end tests on the client side.

Lesson 6

Choosing the Right Test Cases

- Discuss strategies for selecting appropriate test cases to achieve comprehensive coverage.
- Explore boundary testing, edge cases, and negative scenarios.

Lesson 7

Continuous Integration and Testing

- Introduce the importance of continuous integration (CI) in ensuring code quality.
- Discuss integrating your testing suite with CI tools like Travis CI or GitHub Actions.

Lesson 8

Testing Best Practices and Patterns

- Cover testing best practices, including arranging, acting, and asserting in test cases.
- Discuss testing patterns like Arrange-Act-Assert (AAA) and Given-When-Then.

Lesson 9

Test Automation and CI/CD Pipelines

- Discuss strategies for automating test execution and integrating testing into CI/CD pipelines.
- Explore how automated testing helps catch regressions early.

Lesson 10

API Documentation and Testing

- Discuss the importance of API documentation in ensuring developers understand endpoints and functionality.
- Introduce tools like Swagger for generating API documentation and testing endpoints.

Mini Project

Task Management API and Testing Suite

Create a RESTful API for managing tasks and build a comprehensive testing suite to ensure its functionality and reliability. This project will involve developing the API using Express, writing unit and integration tests, and generating API documentation.

Todo

1. **Task CRUD Operations**: Implement API routes for creating, reading, updating, and deleting tasks.
2. **User Authentication**: Include user authentication to ensure that users can only manipulate their own tasks.
3. **Unit Tests**: Write unit tests to ensure the individual components of the API work as expected.
4. **Integration Tests**: Develop integration tests to verify that the API endpoints interact correctly with the database and other components.
5. **API Documentation**: Generate API documentation using tools like Swagger to provide clear instructions for using the API.

Week 24

Project Management Methodologies

By completing this week's curriculum, students will gain practical experience in project management methodologies, specifically Agile. They'll demonstrate proficiency in understanding and applying Agile concepts, managing backlogs, and conducting stand-up meetings. These skills are essential for collaborating effectively within development teams and delivering successful software projects.

Lesson 1

Understanding Project Management:

- Introduce the importance of project management in software development.
- Discuss how effective project management can lead to successful outcomes.

Lesson 2

Agile Methodology

- Dive into Agile as a popular project management methodology.
- Explain the concepts of sprints, user stories, and continuous iterations.

Lesson 3

Implementing Agile in Development

- Discuss how to implement Agile practices in software development projects.
- Explain the roles of Scrum Master, Product Owner, and Development Team.

Lesson 3

Scrum Framework

- Explore the Scrum framework as a specific implementation of Agile.
- Discuss the roles of Scrum Master, Product Owner, and Development Team in Scrum.

Lesson 4

Sprint Planning and Backlog Management

- Explain the process of sprint planning and how user stories are prioritized.
- Discuss techniques for managing the product backlog effectively.

Lesson 5

Daily Stand-ups and Sprint Review

- Introduce the concept of daily stand-up meetings and their purpose.
- Discuss the importance of the sprint review meeting for evaluating progress and adjusting the backlog.

Lesson 6

Agile Tools and Software

- Explore tools commonly used in Agile projects for managing backlogs and tracking progress.
- Introduce concepts like Kanban boards and issue tracking systems.

Lesson 7

Waterfall Methodology

- Contrast the Agile approach with the traditional Waterfall methodology.
- Discuss scenarios where Waterfall might still be applicable and its strengths and weaknesses.

Lesson 8

Hybrid Approaches

- Discuss hybrid approaches that combine elements of Agile and Waterfall.
- Explore how hybrid methodologies can be tailored to specific project needs.

Lesson 9

Project Metrics and Continuous Improvement

- Introduce project metrics for measuring progress and success.
- Discuss the importance of continuous improvement and how it's embedded in Agile methodologies.

Lesson 10

Choosing the Right Methodology

- Discuss factors to consider when choosing a project management methodology.
- Explore how to adapt methodologies based on project size, team composition, and project requirements.

Capstone project

Todo

1. **Project Scope and Complexity:**
   - Design a project that demonstrates your proficiency in both front-end and back-end MERN technologies.
   - Ensure the project's complexity aligns with the skills covered in the curriculum.
2. **Functional Requirements:**
   - Specify features like user authentication, CRUD operations, real-time interactions, and API usage.
   - Encourage the use of React components, Express routes, MongoDB for data storage, and WebSockets for real-time updates.
3. **User Interface (UI) and User Experience (UX):**
   - Design a responsive and visually appealing UI using React for the front-end.
   - Prioritize intuitive navigation, clear feedback, and a cohesive design.
4. **Authentication and Security:**
   - Implement user registration, login, and authentication using tools like Passport.js.
   - Focus on secure password storage, session management, and data privacy.
5. **Database Design and Management:**
   - Design MongoDB schemas that capture the data structure of your application.
   - Emphasize efficient querying, indexing, and relationships.
6. **API Design and Endpoints:**
   - Clearly outline Express API routes for different features and interactions.
   - Define RESTful routes and corresponding CRUD operations.
7. **Real-Time Interactions:**
   - Consider implementing real-time updates using WebSockets, perhaps for real-time comments or notifications.
8. **Testing and Quality Assurance:**
   - Require unit and integration tests for both front-end and back-end components.
   - Encourage using testing libraries like Jest for the React front-end and Mocha/Chai for the Express back-end.
9. **Documentation:**
   - Include detailed instructions on how to set up the project locally, including dependencies and environment variables.
   - Document the API endpoints, expected responses, and any specific requirements.
10. **Code Quality and Best Practices:**
    - Stress the importance of adhering to React best practices, clean code, and proper naming conventions.
    - Encourage modularization and separation of concerns.
11. **Version Control:**
    - Require the use of Git for version control, with regular commits and descriptive commit messages.