Experience Report: Developing a Self-Profile Client-Side Application with React.js and Bootstrap

Introduction

Tackling the challenge of building a self-profile client-side application was an exhilarating journey. My goal was to create a seamless and interactive user experience utilizing React.js and Bootstrap, adhering to a specified template, and ensuring the application was responsive and intuitive.

Client-Side Application Development

React.js and Bootstrap Integration

From the outset, integrating React.js with Bootstrap presented an exciting challenge. I aimed to harness the robust UI components of Bootstrap within the dynamic and component-driven architecture of React.js. By utilizing React-Bootstrap, I could seamlessly integrate Bootstrap components, such as forms and buttons, into my React application, ensuring a cohesive and attractive interface.

Adherence to Specified Template

Adhering to the specified Bootstrap template required meticulous attention to detail. I practiced utilizing Bootstrap's grid system and responsive design features to ensure that the application's layout matched the template across different devices and screen sizes. This exercise honed my skills in crafting flexible and responsive web pages.

Proficiency in State Variables

Implementing state variables in React to manage user inputs was a critical aspect of the project. I created input fields for users to edit their names and descriptions. Leveraging React's useState hook, I could efficiently manage the state of these inputs, allowing for real-time updates and an interactive user experience.

Addition Operation: Frontend and Backend

The requirement to perform addition operations both on the client and server sides introduced me to the intricacies of full-stack development. On the client side, I implemented a simple UI with two input fields for numbers and a button to perform the addition. Using React state, the first output was immediately calculated and displayed upon user interaction.

For the second output, calculated on the server side, I developed an API using Node.js and Express.js that accepts two numbers, performs the addition, and returns the result. The client-side application then fetches this result using Axios and displays it alongside the first output. This not only demonstrated my ability to handle state and user inputs but also my capability to integrate frontend and backend technologies to create a cohesive application.

Server-Side Application Development

Node.js and Express.js Server

Building the server with Node.js and Express.js was straightforward yet insightful. I set up a simple Express server that listens for POST requests on a specific endpoint. The server receives two numbers from the request body, calculates their sum, and sends the result back to the client. This exercise solidified my understanding of creating RESTful APIs and handling HTTP requests.

Deployment

Deploying the application on an AWS EC2 instance was the final step in bringing the project to life. After building the client-side application, I placed the build folder in the public directory of my Express server. Following this, I configured a new EC2 instance, ensuring it had the necessary security group settings to allow HTTP access. Deploying the application on this instance allowed me to appreciate the nuances of cloud computing and the importance of security and accessibility in web applications.

Conclusion

This project was an awesome journey through the landscape of modern web development, from the intricacies of frontend development with React.js and Bootstrap to the backend logic with Node.js and Express.js, and finally, the deployment on AWS EC2. Regular commits to GitHub not only demonstrated my progress but also underscored the importance of version control in software development. Through this experience, I have gained a deeper appreciation for the power of combining these technologies to create compelling and functional web applications.