Full-Stack Web Application Project: A Coding Journey

This comprehensive project challenges your abilities as a full-stack developer to construct a web application resembling real-world applications. Through simulating features found in familiar scenarios, this endeavor allows you to demonstrate your proficiency in Express.js, Node.js, React.js, and MySQL.

Express.js - Backend Development

You're tasked with constructing the server-side logic for a book-sharing application. This involves handling user requests, interacting with the database, and sending appropriate responses. Your application should encompass the following functionalities:

1. Allow users to browse a catalog of books with details such as title, author, and a short description.
2. Enable users to filter the book catalog based on genre and publication year.
3. Implement a feature for retrieving details of a specific book using its unique identifier.
4. Users should be able to add new books to the catalog, including uploading a cover image.
5. Implement a mechanism for users to edit the information of books they've added.
6. Allow users to mark a book as 'available' or 'borrowed'.
7. Set up a system to track when a borrowed book is due for return.
8. Ensure all data exchanges between the frontend and backend are secure and follow best practices.
9. Incorporate a logging mechanism to record significant server-side events.
10. Implement measures to prevent excessive requests to the server from a single source.

Node.js - Server Environment

The heart of your book-sharing application lies within the Node.js environment. Here's where you'll manage the interaction between your Express.js application, your database, and external services. Implement the following:

1. Establish a connection to a MySQL database to store and retrieve data for your book-sharing application.
2. Structure your server-side code by creating modular controllers responsible for handling specific functionalities like user management, book actions, and data interactions.
3. Define well-structured routes to map HTTP requests to corresponding controller functions, ensuring a logical flow in handling user interactions.
4. Before storing sensitive user data, such as passwords, in your database, implement a secure mechanism to transform them into irreversible representations.
5. Enhance the security of your application by employing encryption techniques to safeguard sensitive data during transmission and while at rest within your database.
6. Leverage the concept of closures to encapsulate private variables and functions within a specific scope, enhancing the modularity and maintainability of your codebase.
7. Demonstrate mastery over asynchronous programming paradigms by effectively employing callbacks, Promises, and async/await to handle operations involving database queries and external API calls.
8. Showcase your understanding of the Node.js event loop and how it manages asynchronous operations. Utilize event emitters to create responsive and non-blocking behavior within your application.
9. Organize your codebase effectively by utilizing either CommonJS or ES6 modules. Choose the module system best suited for your project structure and demonstrate its proper usage for importing and exporting modules within your Node.js environment.
10. Implement functionality to store uploaded book cover images to your server's file system. Ensure

efficient handling of file operations using appropriate Node.js modules.

React.js - Frontend Development

The user interface for your book-sharing application will be built using React.js. Your task is to create a dynamic and responsive frontend that allows users to interact with all the features you've implemented on the backend. Focus on these areas:

1. Construct a component to display a list of books retrieved from the server. Implement a pagination system to efficiently handle large lists, allowing users to navigate through pages of results.
2. Create reusable components for displaying individual book details. Each book component should dynamically fetch and display data from the server based on the selected book.
3. Develop a form component that allows users to input and submit new book information to the server, including the ability to upload a cover image.
4. Design an intuitive search bar that allows users to filter and find books within the catalog based on title, author, or genre.
5. Implement a mechanism to display appropriate messages to the user based on the outcome of their actions, like successful book additions, edits, or errors encountered during the process.
6. Incorporate a feature that lets users mark a book as a favorite. This preference should persist even after they close and reopen the application.
7. Implement a section where users can view details of only the books they've marked as favorites.
8. Write comprehensive unit and integration tests using a suitable testing library in React to ensure the robustness and correctness of your components' logic and rendering.
9. Fetch data from your backend API using Axios or a similar library. Handle different response types and potential errors gracefully to ensure a seamless user interaction.
10. Structure your React application using React Router to enable navigation between different views, such as the book catalog, individual book details, user profiles, and the book upload form.

MySQL - Data Management

The foundation of your application lies in a well-structured MySQL database. You will design the database schema, define relationships between tables, and write queries to manage the data for your book-sharing application. Here are your key tasks:

1. Design and create the necessary tables to store information about books, users, and borrowing records. Choose appropriate data types for each column.
2. Establish relationships between the tables using primary and foreign keys to ensure data integrity and efficient querying.
3. Write SQL queries to retrieve a list of books sorted by different criteria, such as popularity (number of times borrowed), publication date, or author.
4. Create queries that allow for the retrieval of all books belonging to a specific genre or written by a particular author.
5. Implement a query to fetch details of a single book using its unique identifier.
6. Write SQL statements to update book information, such as marking a book as borrowed, changing its availability status, and recording the return date.
7. Craft queries that calculate and retrieve aggregated data, such as the total number of books in the system, the average rating of a book, or the number of books currently borrowed.
8. Implement appropriate indexes on frequently queried columns to speed up data retrieval operations.
9. Experiment with different JOIN operations to understand how to retrieve related data from multiple tables efficiently.
10. Write a complex SQL query that involves filtering, sorting, and possibly aggregating data across multiple tables, demonstrating your proficiency in advanced query construction.

Submission

Your completed project should be organized and well-documented. Provide clear instructions on how to run your application. Your code will be evaluated on functionality, code style, adherence to best practices, and the quality of your testing suite.