Design Document: Simple Calculator Application

Overview The aim of this project is to create a simple calculator application with basic arithmetic functions. The application will consist of a browser-based interface built with React.js and Next.js for the frontend, and a Node.js server with Nest.js for the backend. The application will allow users to perform calculations following the order of operations (PEMDAS) and will also provide authentication functionality for registration and login.

Frontend Design (React.js and Next.js) Components and Layout

The frontend will comprise the following components:

- Numeric Keypad: A grid of buttons containing digits from 0 to 9 and a decimal point.
- Operation Buttons: Buttons for addition (+), subtraction (-), multiplication (*), and division (/).
- Memory Buttons: Buttons for memory functions (M+, M-, MR, MC).
- Other Functions: Buttons for percentage (%), square root (√), exponential (^).
- Display: An area that shows the current input and calculation result.
- Authentication Panel: A navigation bar or panel displaying user authentication status and providing options for registration and login.

Workflow

- Users can click numeric keypad buttons to input digits and the decimal point.
- Users can click operation buttons to select arithmetic operations.
- Users can use memory buttons for memory-related functions.
- Users can perform other mathematical functions like percentage, square root, and exponential.
- The display will show the current input and calculated result.
- Users can register and log in using the authentication panel.

Additional Features

- User Session
- History Functionality

Backend Design (Node.js and Nest.js)

API Endpoints Authentication: Implement API endpoints for user registration and login.

- /user: Register a new user with email and password.
- /login: Log in a user with email and password.

Authentication and Security

- User Authentication: Implement user authentication using JWT tokens.

- Password Hashing: Store user passwords securely using bcrypt or a similar hashing algorithm.
- Middleware: Implement middleware to authenticate API requests and authorize access.

Data Storage

- User Data: Store user information (email, hashed password) in a Sqlite database.
- Calculation History: Store calculation history in local storage on the frontend.

Deployment

- Deploy the frontend using Vercel and the backend using Fly.io.

Conclusion

The decision was made to utilize React.js with Next.js for the frontend and Node.js with Nest.js for the backend, both with TypeScript. This ensures efficient and consistent development, providing frontend SSR for enhanced performance and SEO, and a modular backend structure for simplified scalability and maintenance. For styling, Styled Components were employed, as well as Axios for HTTP request management. For arithmetic operations logic, the Math.js library was used, offering security, control, and advanced features for evaluating mathematical expressions, mitigating security risks, and enabling complex calculation manipulations. In the backend, Prisma was employed for database management, alongside Passport for authentication and bcrypt for password encryption in the database.