**Design Decisions and Challenges in Web Application Development**

Implementing a robust and user-friendly web application involves making several critical design decisions and addressing numerous challenges. Below are the key design decisions and challenges faced during the development of this web application:

**1. Authentication and Storage using Firebase**

**Design Decision:**

**Choice of Firebase:** Firebase was chosen for authentication and storage due to its robust security features, ease of integration, and real-time database capabilities.

**Separation of Concerns:** An Auth component was created to handle authentication processes, which ensures that authentication logic is encapsulated and reusable.

**Challenges:**

* **Security:** Ensuring the secure storage of sensitive information such as email and password. This includes using Firebase Authentication to handle password hashing and secure storage.
* **Error Handling:** Properly capturing and displaying error messages during login/signup attempts to provide clear feedback to the user.

**2. User Routing and Error Handling**

**Design Decision:**

* **Routing:** Implementing routing logic to navigate users to the dashboard upon successful login/signup or show error messages if authentication fails.
* **Error Handling:** Incorporating comprehensive error handling mechanisms within the Auth service to manage login/signup failures effectively.

**Challenges:**

* **User Experience:** Providing a smooth transition between the login page and the dashboard, and ensuring error messages are clear and helpful.

**3. Dashboard Component and RESTful API Integration**

**Design Decision:**

* **RESTful API Integration:** Using GET requests to fetch data and display it on the dashboard. The dashboard component handles CRUD operations using a DataService.
* **CRUD Operations:** Allowing users to add, delete, and update rows in a table through POST, DELETE, and UPDATE requests respectively.
* **Technology stack:** Node.Js, Express.Js
* **db.json file:** Stores data to be displayed on dashboard page
* **API endpoints:** 4 API endpoints:

1. get (‘/users’) 🡪fetch data from db.json

2. post(‘/users’)🡪adds new data to db.json

3. put(‘/users/:id’)🡪updates existing data in db.json

4. delete(‘/users/:id’)🡪delete data from db.json

**Challenges:**

* **Data Consistency:** Ensuring data consistency and synchronization between the client and server.
* **Error Handling:** Managing errors that arise from API requests, such as handling network issues or invalid responses.

**4. User Profile and Logout Functionality**

**Design Decision:**

* **Profile Component:** Creating a profile component to allow users to view their information.
* **Logout Functionality:** Providing a logout option that clears user session data and redirects to the login page.

**Challenges:**

* **Redirect Logic:** Handling redirects smoothly to provide a seamless user experience.

**5. Modular Component Design**

**Design Decision:**

* **Auth Component:** This component includes an auth service for handling login/signup, error handling, and authentication methods. It also contains a user model for creating users and generating tokens.
* **Dashboard Component:** Responsible for implementing the dashboard features, utilizing the DataService for CRUD operations.
* **Header Component:** Includes navigation links for home, profile, and logout to facilitate easy navigation.
* **Profile Component:** Dedicated to displaying user information.
* **App-Routing Module:** Implements all necessary routes for navigating between components.

**Challenges:**

* **Component Interaction:** Ensuring seamless interaction between components, such as passing user data from the auth component to the dashboard and profile components.
* **Routing Configuration:** Properly configuring routes to ensure secure and intuitive navigation.

**6. Data Service for CRUD Operations**

**Design Decision:**

* **Data service:** A dedicated service used by the dashboard component to perform CRUD operations. This abstraction promotes code reuse and separation of concerns.
* **Profile data service:** A dedicated service used by the dashboard component to display user information.