mAadhar Application writeup

The provided code consists of an Angular frontend and a Spring Boot backend for a mAadhar application, which appears to be a simplified system for handling Aadhar card details and user authentication. Here's an overview of the different parts of the code:

Frontend (Angular)

1. **add-aadhar.component.html**: This is the HTML template for adding Aadhar card details. It includes a form where users can input their information.

2. **add-aadhar.component.ts**: This TypeScript component corresponds to the HTML template. It handles form control using Angular Reactive Forms and communicates with the AadharService to send and receive data from the backend.

3. **login.component.html**: This template is for user login. It includes a form for users to input their login credentials.

4. **login.component.ts**: The TypeScript component for the login page. It handles form control, sends login data to the backend using the LoginService, and navigates to different pages based on the response.

5. **signup.component.html**: This template allows users to sign up for the application. It includes a form for creating a new account.

6. **signup.component.ts**: The TypeScript component for user registration. It communicates with the LoginService to send registration data to the backend.

7. **app-routing.module.ts**: Defines the routes for different components, such as login, signup, and various user dashboards.

8. **app.module.ts**: Defines the main Angular module, importing necessary modules and declaring components.

Backend (Spring Boot)

1. **MAadharAppApplication.java**: The entry point of the Spring Boot application.

2. **Aadhar.java**: JPA Entity class representing Aadhar card details.

3. **Login.java**: JPA Entity class representing user login information.

4. **AadharController.java**: RestController for handling Aadhar-related operations like storing, updating, and retrieving Aadhar card details.

5. **LoginController.java**: RestController responsible for user authentication and registration.

6. **AadharRepository.java**: Spring Data JPA repository interface for the Aadhar entity.

7. **LoginRepository.java**: Spring Data JPA repository interface for the Login entity.

8. **AadharService.java**: Service class containing business logic for Aadhar-related operations.

9. **LoginService.java**: Service class containing business logic for user authentication and registration.

10. **Dockerfile**: Defines the Docker image for running the Spring Boot application.

11. **application.properties**: Contains Spring Boot configuration properties, including database connection details.

12. **pom.xml**: Maven configuration file specifying dependencies and build settings for the Spring Boot project.