Project Documentation

Introduction

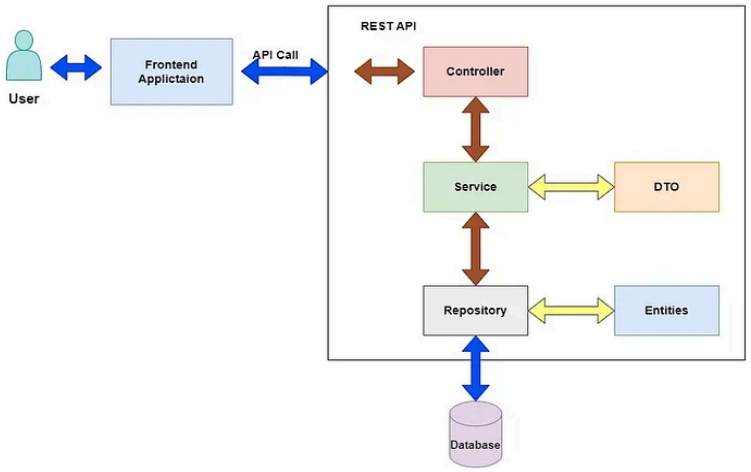
The CMS application is designed to manage and publish articles and images. It allows users to perform CRUD operations on articles and manage pictures associated with those articles. Spring Boot was used to develop the CMS application.
Key features of the applications are :

1. User authentication and authorization
2. API for content managing
3. Integration with PostgreSQL
4. API documentation with Swagger

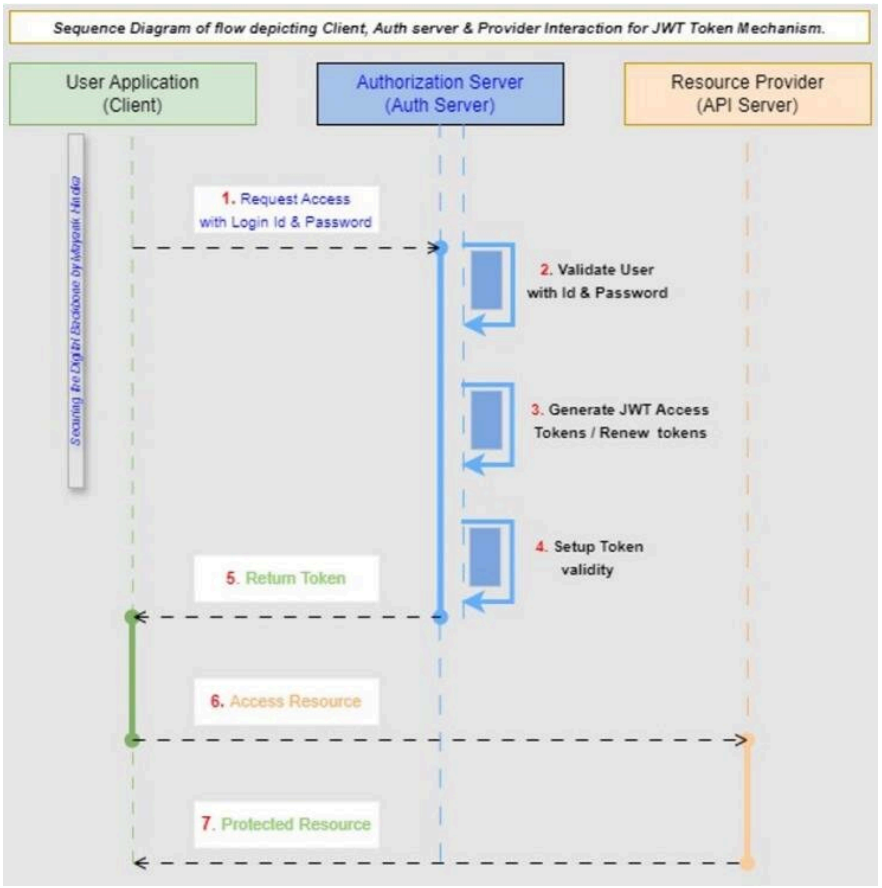
Development Process

- Tools and Technologies
    a. Programming Language: Java
    b. Framework: Spring Boot
    c. Build Tool: Maven
    d. Database: PostgreSQL
    e. API Documentation: Swagger
- Development Steps
    a. Project Initialization: Creating the Spring Boot project using Initializr and selecting the required dependencies (Spring Web, PostgreSQL, Spring Security, Lombok)
    b. Database Setup: Database created and properties.yml file modified to allow connection with the PostgreSQL database

c. Project Basic Structure build based on the following diagram :



d. Models and Repositories creation: Article, Image, and User classes and their repositories created
e. Services Interfaces - Classes implementation
f. Controller creation: ApiController implementation
g. JWT Token Authentication Implemented based on below diagram:



h. Basic API Documentation with Swagger

Packages and classes Explanation

Auth →

1. AuthenticationController**:** handles login requests, receives an AuthenticationRequest, authenticates it using AuthService, and returns an AuthenticationResponse.
2. AuthenticationRequest**:** is used to encapsulate login credentials (username and password) sent from the client to the server during the authentication process.
3. AuthenticationResponse**:** returns an accessToken after successful authentication, allowing subsequent API calls with the token.
4. AuthService**:** handles authentication, checks for an admin user, creates one if missing, validates credentials, generates a JWT token upon success, and returns it in an AuthenticationResponse.

config →

1. ApplicationConfig **:** sets up Spring Security, providing user details retrieval, password encoding, and authentication management beans for the application.
2. JWTFilter **:** extracts and validates JWT tokens, and sets up authentication for authorized users, enhancing security in the application.
3. JWTService **:** handles JWT token generation, extraction, validation, and expiration checking.
4. OpenApiConfig **:** defines Swagger/OpenAPI specifications for the CMS application, including metadata, server URL, and security requirements for JWT authentication.
5. SecurityConfig **:** configures Spring Security, enabling stateless authentication, defining allowed paths, and integrating JWT filtering.

controller →

1. ApiController **:** handles CRUD operations for Articles and Images, providing REST endpoints for creating, reading, updating, and deleting these entities

model →

1. Article, Image, User**:** represent entities with annotations for JPA persistence.

repository →

1. ArticleRepo, ImageRepo, UserRepo **:** extend JpaRepository to provide basic CRUD while UserRepo adds a method to find user by username.

service →

1. ArticleService**:** defines methods for CRUD operations on Article objects.
2. ArticleServiceImpl**:** implements the ArticleService interface.
3. ImageService**:** defines methods for retrieving and creating images.
4. ImageServiceImpl**:** implements the ImageService interface.

Environment Setup

- Requirements :
  - **Java** (JDK 8 or higher)
  - **Maven**
  - **PostgreSQL**
  - **Git**

- Clone the Repository :
  - git clone https://github.com/klebash/cms-application.git
- Create the Database named cms_db
- Open the properties.yml file in `src/main/resources` and update the database connection details using your own credentials.
- Set Up Dependencies with **./mvnw clean install** (linux) and run the Application with **./mvnw spring-boot:run (** If you're using Windows, use mvnw.cmd instead of mvnw) or use an IDE (IntelliJ IDEA or Eclipse).
- Use Postman collection to test it.

API Endpoints

Method	Endpoint	Description	Auth Required
POST	/api/articles	Create a new article.	Yes
POST	/api/images	Upload an image.	Yes
POST	/api/auth/login	Admin user login.	No
GET	/api/articles	Retrieve a list of all articles.	No
GET	/api/articles/{id}	Retrieve a single article by ID.	No
GET	/api/images/{id}	Retrieve an image by ID.	No
PUT	/api/articles/{id}	Update an existing article.	Yes
DELETE	/api/articles/{id}	Delete an article by ID	Yes

Authentication

- ❖ JWT Token authentication is used to secure certain endpoints. To access these, you must:
  - ➢ Authenticate using api/auth/login to receive a token with credentials:

```
{
    "username" : "admin",
    "password" : "admin123"
}
```

Resources

- https://jwt.io/introduction
- https://www.javatpoint.com/spring-boot-tutorial
- https://asecuritysite.com/encryption/plain
- https://www.tutorialspoint.com/spring_boot/index.html
- https://www.codejava.net/frameworks/spring-boot/connect-to-postgresql-database-examples
- https://www.bezkoder.com/spring-boot-jwt-authentication/