**Project Documentation: Healthcare Management System**

**1. Project Overview**

The **Healthcare Management System** provides a web-based platform to manage patient registration, appointment scheduling, medication management, and user profiles. The system includes both frontend and backend components that interact seamlessly to provide a full-fledged healthcare service experience.

**Core Features:**

* **User Registration and Authentication**: Secure user registration and login.
* **Appointment Scheduling**: Allows patients to schedule appointments with doctors.
* **Medication Management**: Enables patients to track their medications and dosage schedules.
* **User Profiles**: Manage and update patient or doctor profile details.
* **Admin Dashboard**: Allows admins to manage users, appointments, and medications.

**2. Setup Instructions**

**Prerequisites:**

* Java 17 or higher
* Spring Boot 3.x
* Maven 3.8 or higher
* MySQL or PostgreSQL Database
* Node.js (for front-end if using React/Angular)

**Clone the repository:**

bash

Copy code

git clone https://github.com/your-repo/healthcare-management-system.git

cd healthcare-management-system

**Backend Setup:**

1. **Install dependencies**:
   * Make sure you have Maven installed and then run:

bash

Copy code

mvn clean install

1. **Configure Database**:
   * Edit src/main/resources/application.properties or application.yml to configure your database connection:

properties

Copy code

spring.datasource.url=jdbc:mysql://localhost:3306/healthcare\_db

spring.datasource.username=root

spring.datasource.password=password

1. **Run the application**:

bash

Copy code

mvn spring-boot:run

**Frontend Setup:**

1. **Install Node.js and NPM**.
2. Navigate to the frontend directory (if you are using a separate frontend project).
3. **Install dependencies**:

bash

Copy code

npm install

1. **Run the frontend**:

bash

Copy code

npm start

**3. Features**

**User Registration and Authentication**

* Users can register by providing a username, password, and role (Patient/Doctor).
* JWT-based authentication is used for secure login and token management.

**Appointment Scheduling**

* Patients can schedule appointments by selecting a date, time, and doctor.
* Admins can view, update, or delete scheduled appointments.

**Medication Management**

* Patients can add, update, and track their medication, dosage, and frequency.
* Notifications can be set to remind patients of their medication schedule.

**User Profiles**

* Both doctors and patients have editable profiles with personal details.
* Doctors can view patient history and prescribe medications.

**4. Database Schema**

Below is a simple schema for the project. This can be extended based on your requirements.

**User Table**

sql

Copy code

CREATE TABLE user (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

role ENUM('PATIENT', 'DOCTOR', 'ADMIN') NOT NULL,

first\_name VARCHAR(255),

last\_name VARCHAR(255),

email VARCHAR(255),

phone VARCHAR(255)

);

**Appointment Table**

sql

Copy code

CREATE TABLE appointment (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

patient\_id BIGINT NOT NULL,

doctor\_id BIGINT NOT NULL,

appointment\_date DATETIME NOT NULL,

purpose VARCHAR(255),

FOREIGN KEY (patient\_id) REFERENCES user(id),

FOREIGN KEY (doctor\_id) REFERENCES user(id)

);

**Medication Table**

sql

Copy code

CREATE TABLE medication (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

patient\_id BIGINT NOT NULL,

name VARCHAR(255) NOT NULL,

dosage VARCHAR(255),

frequency VARCHAR(255),

start\_date DATE,

end\_date DATE,

FOREIGN KEY (patient\_id) REFERENCES user(id)

);

**5. API Documentation**

**1. User Registration and Authentication**

* **POST /api/auth/register**

**Request:**

json

Copy code

{

"username": "testuser",

"password": "password123",

"role": "PATIENT",

"first\_name": "John",

"last\_name": "Doe",

"email": "john.doe@example.com",

"phone": "123456789"

}

**Response:**

json

Copy code

{

"message": "User registered successfully"

}

* **POST /api/auth/login**

**Request:**

json

Copy code

{

"username": "testuser",

"password": "password123"

}

**Response:**

json

Copy code

{

"token": "JWT\_TOKEN"

}

**2. Appointment Scheduling**

* **POST /api/appointments**

**Request:**

json

Copy code

{

"patient\_id": 1,

"doctor\_id": 2,

"appointment\_date": "2024-12-05T10:00:00",

"purpose": "Routine Checkup"

}

**Response:**

json

Copy code

{

"message": "Appointment scheduled successfully"

}

**3. Medication Management**

* **POST /api/medications**

**Request:**

json

Copy code

{

"patient\_id": 1,

"name": "Paracetamol",

"dosage": "500mg",

"frequency": "Twice a day",

"start\_date": "2024-12-01",

"end\_date": "2024-12-10"

}

**Response:**

json

Copy code

{

"message": "Medication added successfully"

}

**6. Testing**

The project includes **unit tests** and **integration tests**. All tests are written using **JUnit 5** and **Mockito**.

To run the tests, use:

bash

Copy code

mvn test

Ensure the following:

* Test cases for **user registration** and **login** functionality.
* Test cases for **appointment scheduling** and handling edge cases like invalid date/time.
* Test cases for **medication management**, ensuring proper medication data is tracked.