**MINI PROJECT - 2**

PATIENT MEDICINE AND APPOINTMENT SYSTEM

**Table of contents: -**

|  |  |  |
| --- | --- | --- |
| 1 | Project Introduction | 2 |
| 2 | Requirements | 3 |
| 3 | Project Setup | 4 |
| 4 | Project Configuration | 5 |
| 5 | Running the Application | 6 |
| 6 | API End Point Documentation | 7 |
| 7 | Validation Rules | 8 |

Introduction

This documentation describes about a **Patient Medicine and Appointment System**. This is a web-based healthcare application designed to streamline the interaction between patients and doctors within a hospital.

The system facilitates essential services such as patient and doctor registration, appointment scheduling, medicine prescriptions, and user profile management.

Key Features: -

1. Hospital Home Page

* A new Patient registration
* A new Doctor Registration
* Access to doctor/patient portal

1. Patient Portal

* Patient Portal has the following functionalities

1. To Book Appointments (Should be able to see all available doctors in hospital, can book appointment as per slots available)
2. To show prescribed medicines list (All medicines prescribed by other doctors in the same hospital would be listed)
3. Update profile
4. Delete profile
5. Logout
6. Doctor Portal

* Doctor Portal has the following functionalities

1. Show Booked Appointments (Should be able to view patient details, add advice and add new medicine or alter previous medicines)
2. Update profile
3. Delete profile
4. Logout

By providing these services in a single platform, helps hospitals and clinics efficiently manage appointments and patient information, reduces administrative workload, and allows for a more seamless experience for both doctors and patients.

Requirements

Ensure you have below items before you start setting up my project in your local machine.

1. An IDE ( lntelliJ, Visual Studio) 🡪 I used intelliJ IDE here..
2. JDK 17
3. Maven (Project Build tool)
4. MySQL along with an IDE(like MySQL Workbench)
5. A browser (like Chrome)

Now, let us proceed to the set-up and configuration of the project.

Project Setup

Here we are going to clone my GitHub repository and import the project into IDE

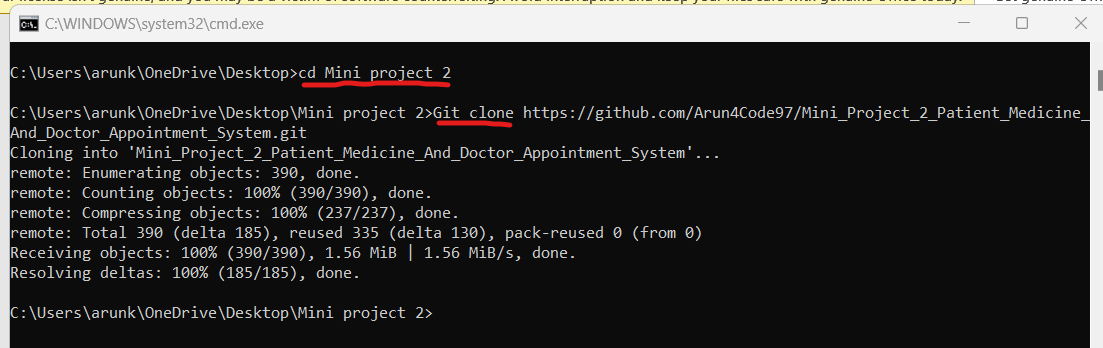
**Step 1: Clone the project repository which I have mentioned in Zen project submission portal:**

1.Goto command prompt and ensure you have created a dedicated directory. In my case, Folder name is **Mini Project 2.**

**cd C:\Users\arunk\OneDrive\Desktop\Mini Project 2**

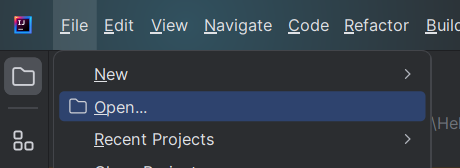
2.Ensure you are located inside the miniProject-2 folder and issue the below command for cloning the repository

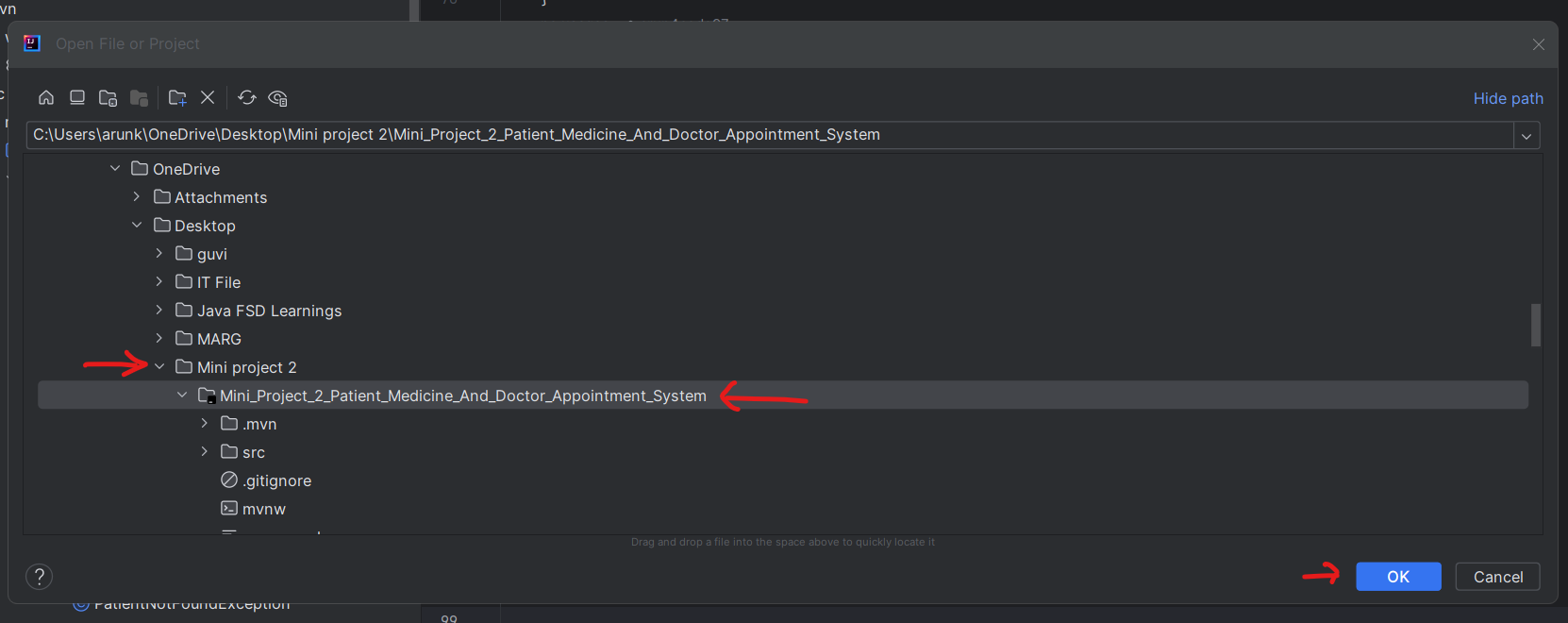
**Git clone https://github.com/Arun4Code97/Mini\_Project\_2\_Patient\_Medicine\_And\_Doctor\_Appointment\_System.git**

****

**Step 2: Import the project into IDE :-**

1. **Opening project** - Goto intellij, In the left top corner, click on open project, select the backend folder and click on **“Ok”**



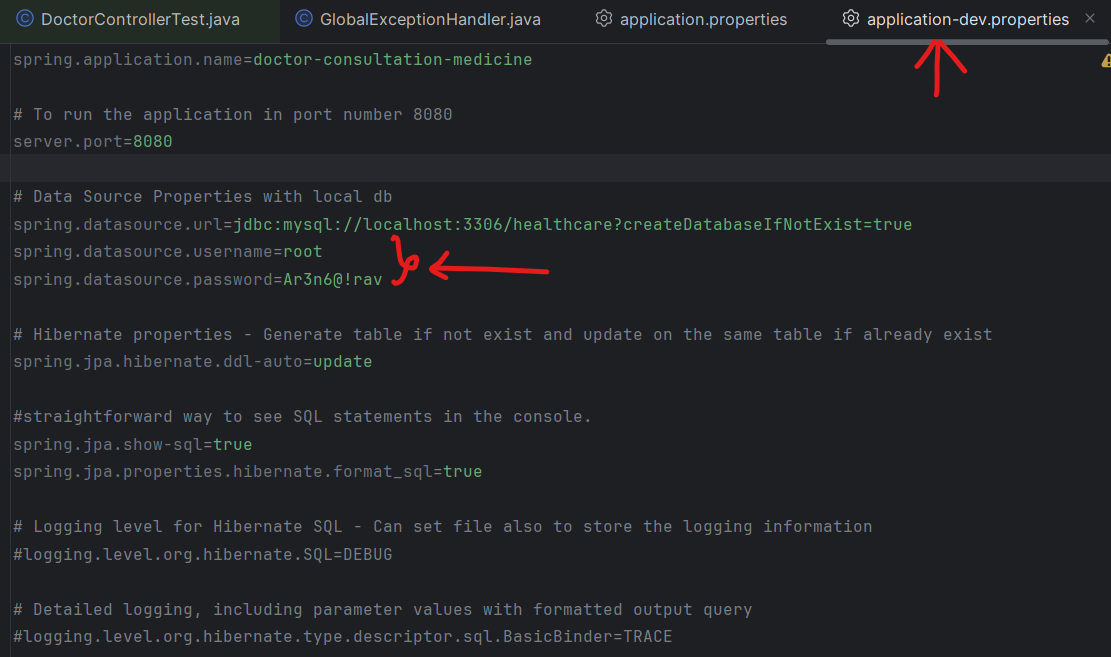


Project Configuration

**Database Configuration:**

Here, We need to connect to your MySQL database for which do the below changes in application-dev.properties which resides under src->Resources->Config .

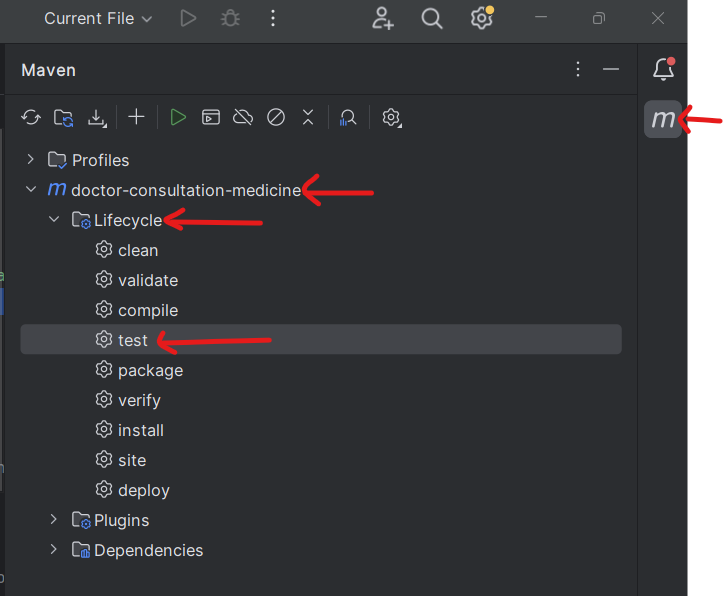
You don’t need to create a database named as **healthcare** which will be automatically created if the database does not exist.   
spring.datasource.username = enter your MySQL user name here  
spring.datasource.password = enter your MySQL password here

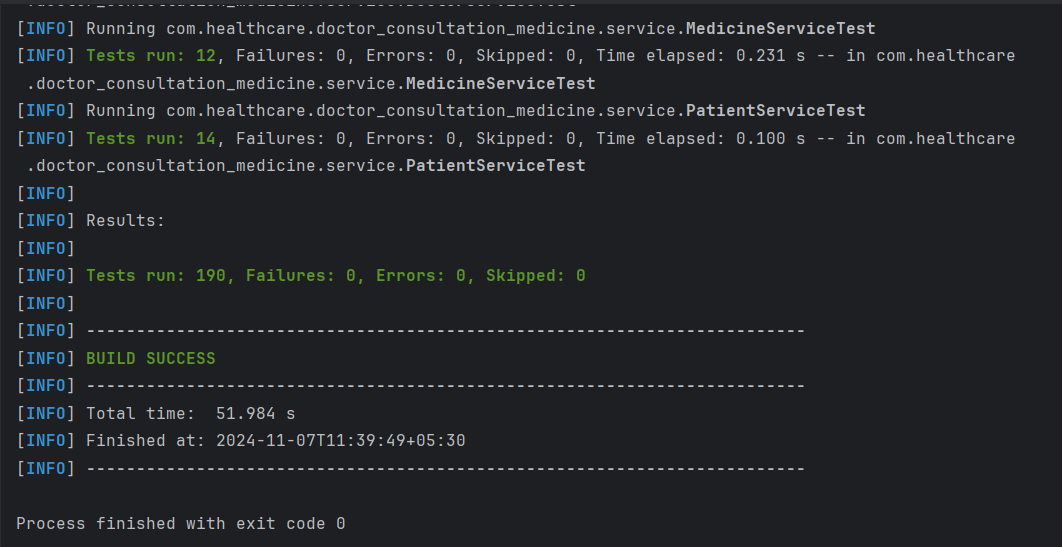


Unit Testing: -

**To run unit testing** of all the layers using Maven test and ensure all **190 test cases** run successfully.

To run the maven test. Click on ‘**m**’ icon on the top right corner and click on doctor-consultation-medicine -> Lifecycle -> test

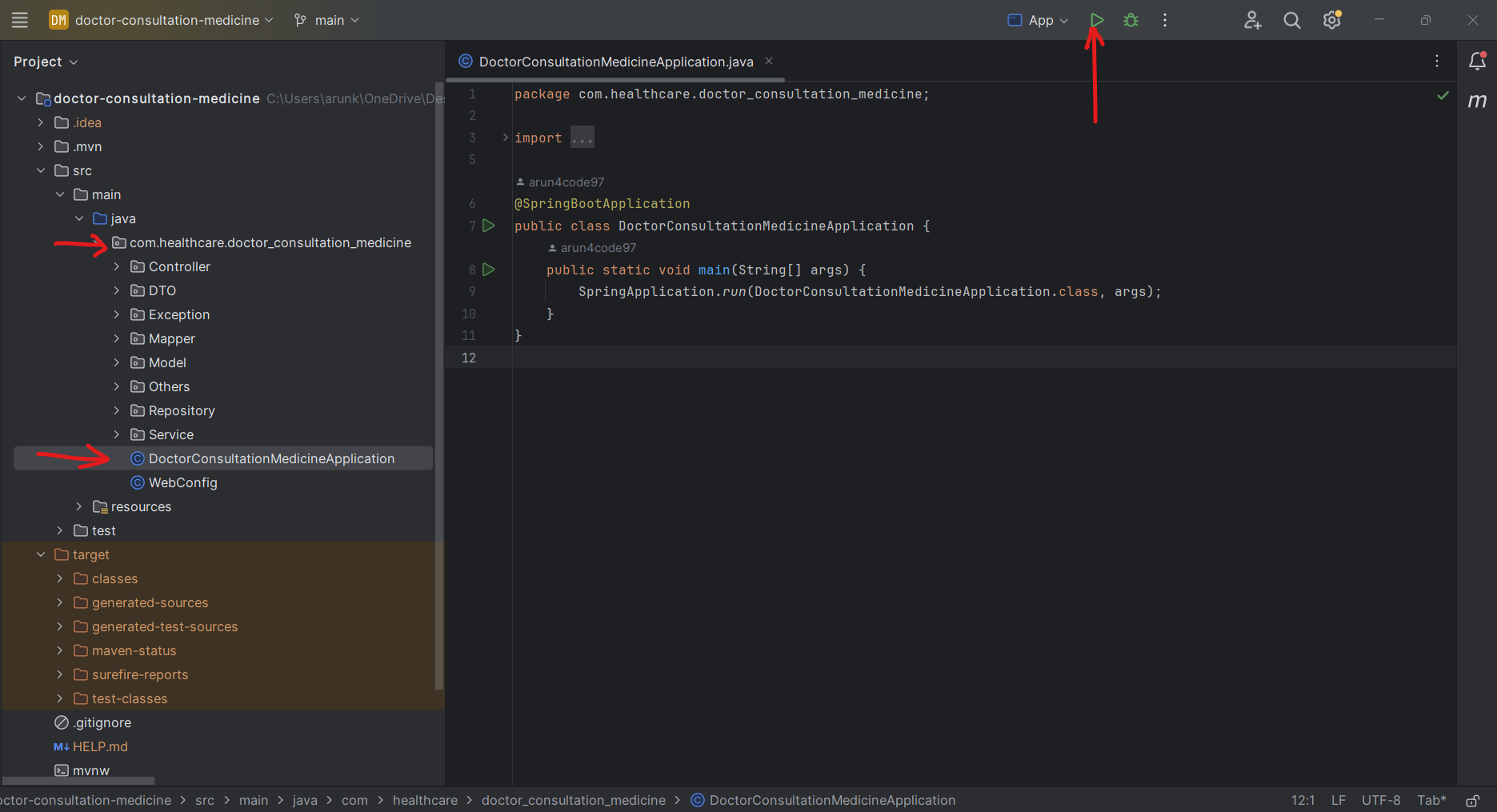


Unit Test output screenshot 

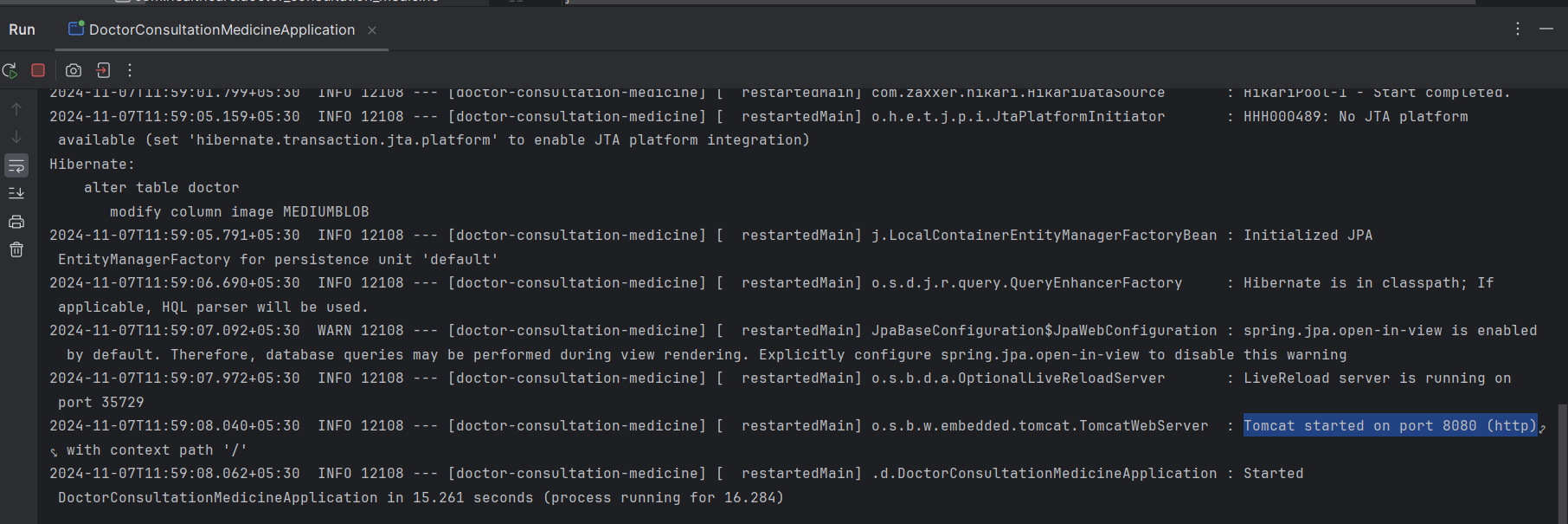
Running the DoctorConsultationMedicineApplication: -

To run my application , go to doctorConsultationMedicineApplication.java which is in the project directory where click on run symbol on the class name or in the top ribbon. **Running the doctorConsultationMedicineApplication.java** and ensure application is running on the port number 8080 as per the application-dev.properties file configuration.

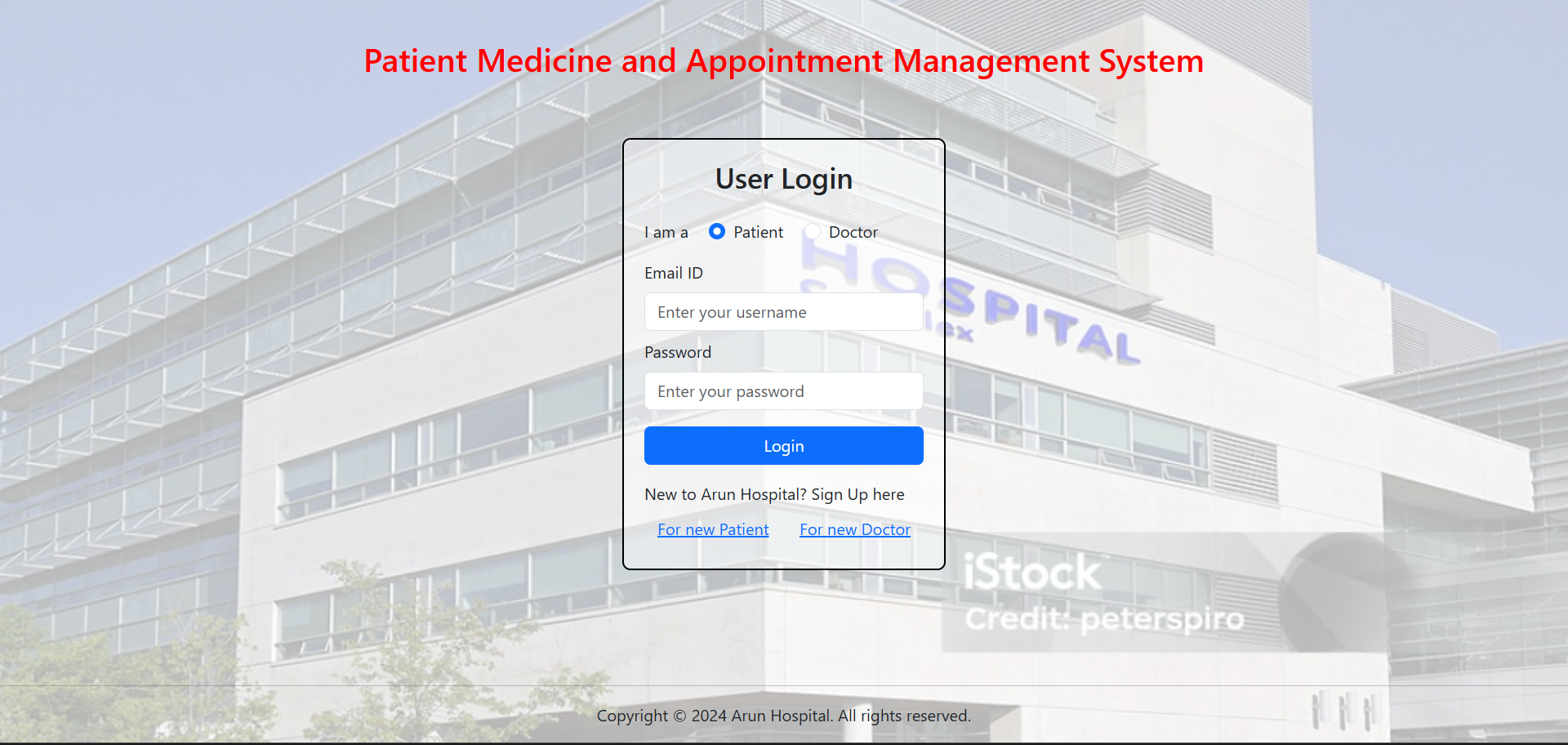
**Step 1:** Locating to doctorConsultationMedicineApplication.java



**Step 2**: Running the application and ensure which is successfully started on port 8080.



**Step 3:** Landing to login Page

Now, go to your browser and hit [http://localhost:8080/hospital](http://localhost:8080/hospital%20) . This would be rendering the login page of my application like below screenshot.

Troubleshooting: -

Common Errors in Backend:

* **Could not connect to database:** Ensure your MySQL database is up and running and your credentials (username and password) are correct.
* **Port already in use:** Cancel the already running application in port 8080 or else change the port in application-dev. properties to different port like 8081,8082.

API Documentation: -

Below is a summary of the all-API endpoints, including details on request/response formats.

*Home Controller End points: -*

1. Render Hospital home page/Login page

*End point:* /hospital

*Method:***GET**

*Description:* Renders the home page

*Response:* Returns an HTML document of the home page(homepage.html).

1. Render Hospital login page

*End point:* /hospital/login

*Method:***POST**

*Request body params:* Model object of UserLoginDto which has username, password and user Type fields.

*Description:* Handles login requests for both patients and doctors by validating user credentials and directing them to the appropriate portal (Patient portal or Doctor portal).

If validation errors are present or user type is invalid, the form is re-rendered with appropriate error message.

*Validation Rules:*

* All fields in UserLoginDto are mandatory.
* Email filed should be valid email id.

*Response:* Returns an HTML document of the appropriate portal page (**doctorportal.html / patientPortal.html**)/ The form is re-rendered with an appropriate error message displayed.

*Patient Controller End points: -*

1. Render Add patient form

*End point:* /hospital/addPatient

*Method:* GET

*Description:* Renders the "Add Patient" form page where new patients can enter their details.

*Response:* Returns an HTML document of the add patient page(addpatient.html) with showPatientForm fragment enabled.

1. Add New Patient

*End point:* /hospital/addPatient

*Method:* POST

*Request body params:* Model object of PatientDto which has all required patient data.

*Description:* Handles the submission of the "Add Patient" form. Validates the input fields and checks if the email already exists.

If validation passes and email is unique, the user is redirected to the setPassword endpoint for further setup.

If validation fails or the email is already in use, the form is re-rendered with an appropriate error message displayed.

*Response:* Successful redirection to the setPassword endpoint with savedPatientId /The form is re-rendered with an appropriate error message displayed.

1. Render patient set-password form

*End point:* /hospital/addPatient/setPassword

*Method:* GET

*Request body params:* SavedPatientId (Long)

*Description:* Renders the "SetPassword" form page where new patients can enter their password.

*Response:* Returns an HTML document of the add patient page(addpatient.html) with setPassword fragment enabled.

1. Set password for new Patient

*End point:* /hospital/addPatient/setPassword

*Method:* POST

*Request body params:* Model attribute – credentials (CredentailDto) and SavedPatientId (Long).

*Description:* Handles the setting of a new password for a patient who has just registered. This endpoint validates the password fields to ensure they match before saving, and updates the password for the patient in the database.

*Validation Rules:* Validates that password and confirmPassword are identical.

*Response:* Returns the patient/addPatient page with a success fragment enabled and then redirects to patient portal using java script / Re-renders the same page with relevant error messages when validation fails or passwords don’t match.

1. view patient profile

*End point:* /hospital/patientPortal/{id}

*Method:* GET

*Path variable:* id (Long)

*Description:* Retrieves the profile information of a specific patient by their ID and displays an error message if patient does not exist.

*Response:* Returns a HTML document of patientPortal populated with the patient's details with mode as **view** / Returns the error page if a PatientNotFoundException occurs, with a 404 Not Found status.

1. Render patient form for updating

*End point:* /hospital/patientPortal/updateProfile/{id}

*Method:* GET

*Path variable:* id (Long)

*Description:* Retrieves the profile information of a specific patient by their ID and displays an error message if patient does not exist.

*Response:* Returns a HTML document of patientPortal populated with the patient's details with mode as **update** / Returns the error page if a PatientNotFoundException occurs, with a 404 Not Found status.

1. Update patient profile

*End point:* /hospital/patientPortal/updateProfile/{id}

*Method:* PUT

*Path variable:* id (Long)

*Request body params:* Model attribute – patient (PatientDto).

*Description:* Validates and updates the profile information of a specific patient by their ID with newly added changes and displays an error message if validation fails or patient does not exist.

*Response:* Redirects to specific patientPortal home page which is /hospital/patientPortal/{id}/ Returns the error message if patient does not exist.

1. Delete patient profile view

*End point:* /hospital/patientPortal/deletePatient/{id}

*Method:* GET

*Path variable:* id (Long)

*Description:* Prepares a confirmation view to delete a specific patient's profile based on their ID. This endpoint fetches the patient’s details and renders the patientPortal page in delete mode.

*Response:* Returns a HTML document of patientPortal configured for delete confirmation/ Returns the error page if a PatientNotFoundException is raised, with a 404 Not Found status.

1. Delete patient profile

*End point:* /hospital/patientPortal/deletePatient/{id}

*Method:* DELETE

*Path variable:* id (Long)

*Description:* Deletes the patient completely from the database and redirects to hospital login page.

*Response:* Redirects to hospital home page after successful deletion.

1. Show Doctors by Specialization

*End point:* /hospital/patientPortal/{id}/showDoctorsBySpecialization

*Method:* GET

*Path variable:* id (Long)

*Description:* Displays a list available doctor and their profile information and including images.

*Response:* Returns an HTML document of patientPortal displaying the list of doctors and their details. / Displays an error message if no doctors found in the hospital. Returns an error page with a 404 Not Found status if PatientNotFoundException is raised.

*Doctor Controller End points: -*

1. Render Add Doctor Form

*End point:* /hospital/addDoctor

*Method:* GET

*Description:* Renders the "Add Doctor" form page where new doctors can enter their details.

*Response:* Returns an HTML document of the add doctor page(addDoctor.html) with showDoctorForm fragment enabled.

1. Add New Doctor

*End point:* /hospital/addDoctor

*Method:* POST

*Request body params:* Model attribute *-* doctor (DoctorDTO) and imageFile(MultiPartFile)

*Description:* Handles the submission of the "Add Doctor" form. Validates the input fields and checks if the email already exists.

If validation passes and email is unique, the user is redirected to the setPassword endpoint for further setup.

If validation fails or the email is already in use, the form is re-rendered with an appropriate error message displayed.

*Response:* Successful redirection to the setPassword endpoint with savedDoctorId /The form is re-rendered with an appropriate error message displayed.

1. Render doctor set-password form

*End point:* /hospital/addDoctor/setPassword

*Method:* GET

*Request body params:* savedDoctorId (Long)

*Description:* Renders the "SetPassword" form page where new doctor can enter their password.

*Response:* Returns an HTML document of the add doctor page(adddoctor.html) with setPassword fragment enabled.

1. Set password for new Doctor

*End point:* /hospital/addDoctor/setPassword

*Method:* POST

*Request body params:* Model attribute – credentials (CredentailDto) and SavedDoctorId (Long).

*Description:* Handles the setting of a new password for a doctor who has just registered. This endpoint validates the password fields to ensure they match before saving, and updates the password for the doctor in the database.

*Validation Rules:* Validates that password and confirmPassword are identical.

*Response:* Returns the addDoctor.html page with a success fragment enabled and then redirects to doctor portal using java script / Re-renders the same page with relevant error messages when validation fails or passwords don’t match.