**Readme**

**Application:**

Pharmacare is an online pharmacy and healthcare web application designed to provide a solution for managing health-related needs online. It allows users to order medicines, schedule doctor appointments, and access health tips on nutrition and wellness. The platform addresses key challenges faced by traditional pharmacies, such as limited stock availability, restricted operating hours, and a lack of health information, offering 24/7 access to services.

**Features**

* **Medicine Ordering**: Users can search for medicines, add them to a cart, and purchase them.
* **Doctor Appointments**: Schedule online or offline consultations with doctors.
* **Health Tips**: Access content on health, nutrition, and wellness to manage health proactively.
* **User Authentication**: Secure login and registration system.
* **Admin Panel**: Manage users, orders, and content of the application.
* **Chatbot Assistance**: Helps users navigate through the app and answer inquiries.
* **Order History**: View past orders and appointment history.

**Tech Stack**

* **Frontend**: React.js, HTML, CSS, Bootstrap
* **Backend**: Spring Boot (Java)
* **Database**: MySQL
* **Deployment**: AWS EC2, Jenkins, Apache Tomcat
* **Security**: Spring Security (JWT-based authentication)
* **API Testing**: Swagger
* **Version Control**: Git, GitHub

**Prerequisites**

* Node.js
* npm (Node Package Manager)
* Java JDK 17
* Maven
* MySQL
* Git

**Setting up frontend**

**#**clone the repository

* git clone [repository-url]
* CD PharmaCareClient

#install dependencies and start

* npm install
* npm start

**Setting up backend**

* cd PharmaCareServer
* mvn clean install
* java -jar target/pharmacare-0.0.1-SNAPSHOT.jar

**Deploying on AWS EC2**

* Set up an AWS EC2 instance.
* Install Java, Maven, and Tomcat.
* Configure security groups and permissions.
* Set up Jenkins for continuous integration and delivery.

**Testing**

The project uses Swagger for API testing. Below are some key API endpoints tested:

* **GET** /user/{id}: Retrieves user details.
* **DELETE** /user/{id}: Deletes a user by ID.
* **GET** /medicine/{id}: Retrieves medicine details by ID.
* **GET** /doctor/{id}: Retrieves doctor details by ID.

Testing was conducted through Swagger to ensure smooth interaction between frontend and backend components.

**Screenshots**

* **Dashboard:** View and manage user interactions, orders, and health content.

A person at a pharmacy

Description automatically generated

* **Medicine Search**: Search for medicines using various filters and order them.

A screenshot of a computer

Description automatically generated

* **Appointment Booking**: Schedule online/offline appointments with doctors.

A screenshot of a computer

Description automatically generated

* **Admin Home:** Manage users, orders, and the content of the application.

A screenshot of a computer

Description automatically generated