**E-MEDICARE**

**By SEEMA BHUYAN**

**Git Hub Link:**

[**https://github.com/Seema-beep/SimplilearnMedicareCapstoneProject.git**](https://github.com/Seema-beep/SimplilearnMedicareCapstoneProject.git)

**ABSTRACT**

The purpose of E-Medicare System is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same.

E-Medicare System, as described above, can lead to error free, secure, reliable and fast management system.

The aim is to automate its existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| s.no | Title | Page no. |
| 1 | Introduction | 4 |
| 2 | Functionalities Provided by E-Medicare system  2.1 E-Medicare system | 4 |
| 3 | System Overview  3.1 Authentication & Authorization  3.1.1 Authentication  3.1.2 Authorization  3.2 Environment | 5 |
| 4 | Sub-System Details  4.1 Admin  4.2 Users  4.3 Login/Logout | 6 |
| 5 | Data Organization  5.1 Admin Details  5.2 User Signup Details  5.3 User Login Details  5.4 Medicine Details  5.5 Cart Details  5.6 Orders Details | 7 |
| 6 | REST APIs to be BUILT | 9 |
| 7 | Feasibility Study | 12 |
| 8 | Assumptions | 12 |
| 9 | UML Diagrams  9.1 Class Diagram  9.2 Sequence Diagram  9.3 ER Diagram | 13 |
| 10 | Testing | 17 |
| 11 | Output Screenshots | 18 |

**1**.**INTRODUCTION**

The “E-Medicare System” has been developed to override the problems prevailing in the practicing manual system. This software is supposed to eliminate and reduce the hardships faced by the existing system. E-Medicare System can lead to error free, secure, reliable and fast management system.

"E-Medicare System" - web application [J2EE Batches - Web Application], where users can register, login, purchase medicines e.g. Antibiotics, Antipyretics. and manage their orders in the system.

**1.1 Scope and Overview:**

The scope of the “E-Medicare System” will be to provide the functionality as described below. The system will be developed on a Windows operating system using Java/J2EE, Hibernate, Spring.

**2.FUNCTIONALITIES PROVIDED BY E-MEDICARE SYSTEM ARE AS FOLLOWS:**

* Provides the searching facilities based on various factors. Such as Customer, Medicine, Order.
* E-Medicare System also manage the Payment details online for Order details, Customer.
* It tracks all the information of Medicine Company, Payment etc.
* Manage the information medicine company.
* Shows the information and description of the customer and medicine.
* Editing adding and updating of records is improved which results in proper resource management of customer data.

**2.1 MODULES OF E-MEDICARE SYSTEM:**

* Customer Module: Used for managing the Customer details.
* Order Module: Used for managing the details of Order.
* Cart Module: Used for adding medicines.
* Medicine Module: Used for managing the Medicine details.
* Login Module: Used for managing the login details.
* Users Module: Used for managing the users of the system.
* Admin Module: Used for managing medicine details and user information.

**3**.**SYSTEM OVERVIEW**

The “E-Medicare System” should support basic functionalities (explained in section 2.1) for all below listed users.

* Admin (A)
* User (C)

**3.1 AUTHENTICATION & AUTHORIZATION**

**3.1.1** **Authentication:**

  Any end-user should be authenticated using a unique user-id and password.

**3.1.2** **Authorization:**

  The operations supported and allowed would be based on the user type. For example, Administrator has the rights to add product information and view customer details. He can also view order details and purchase details of a product. Whereas Customer/Buyer has a right to Add, Remove and Clear all the products from cart.

**3.2 ENVIRONMENT:**

  The system will be developed on any Windows OS machine using J2EE, Hibernate and Spring.

* Intel hardware machine (PC P4-2.26 GHz, 512 MB RAM, 40 GB HDD)
* Server – Apache Tomcat 8
* Database – My SQL
* My SQL J Connector
* Node Version 10
* Angular CLI
* JDK 1.8
* Eclipse IDE or Spring Tool Suite

**4**.**SUB-SYSTEM DETAILS**

The E-Medicare System is defined, wherein all users need to login successfully before performing any of their respective operations.

Find below (section 3.1 & 3.2) tables that provides functionality descriptions for each type of user / sub-system. Against each requirement, indicative data is listed in column ‘Data to include’. Further, suggested to add/modify more details wherever required with an approval from customer/faculty.

**4.1 ADMIN**

The administrator as a user is defined to perform below listed operations after successful login.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Objects | Operations | Data to include |
|  | Medicines | Add  View  Delete  Modify | Medicine Id, Medicine Name, Price, description, manufacture date, expiry date and status |
|  | User | View | Username, contact no, dob, email, gender |
|  | Order and Purchase | View | User Id, item Id, Price, Quantity |

**4.2 USER**

The user is defined to perform below listed operations after successful login.

|  |  |  |
| --- | --- | --- |
| **Objects** | **Operations** | **Data to include** |
| User | Register | User Id, Username, Password, Email, Phone, Number, dob, gender |
| Product | Add to Cart  Delete from Cart  Delete all Products from cart | Product Id, Product Name, Price, Quantity, Status |
| Checkout | Add User Details and Price | Cart-Id, User-Id and Total Price |

**4.3 LOGIN | LOGOUT**

**[Web Application - J2EE, Hibernate, Spring]**

* Go to Registration screen when you click on Register link.
* Go to Success screen when you login successfully after entering valid username & password fetched from the database.
* Redirect back to same login screen if username & password are not matching.
* Implement Session tracking for all logged in users before allowing access to application features. Anonymous users should be checked, unless explicitly mentioned.

**5**.**DATA ORGANIZATION**

This section explains the data storage requirements of the Product Order Entry System and **indicative** data description along with suggested table (database) structure. The following section explains few of the tables (fields) with description. However, in similar approach need to be considered for all other tables.

**5.1 TABLE: ADMIN\_DETAILS**

The admin specific details such as user-id and password. This table contains information related to Admin details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| userid | User-ID is auto generated after registration and it is used as |
|  | Login-ID. |
| password | Admin Password |

**5.2 TABLE: USER\_SIGNUP\_DETAILS**

The user specific details such as user-id, first name, last name, date of birth, email.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| u\_id | User Id is the primary field, it is auto generated after registration and it is used as Login-Id |
| firstname | First name of the User/Customer |
| lastname | Lastname of the User/Customer |
| email | User/Customer Email Id |
| dob | Date of Birth of User/Customer |
| gender | Gender of User/Customer |
| contact\_no | 10-digit contact number of the User/Customer |
| password | Password of the User/Customer |

**5.3 TABLE: USER\_LOGIN\_DETAILS**

This table contains information related to User Login details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| userid | User-Id of the Customer. It is auto generated after registration |
| password | Password of the Customer/User |

**5.4 TABLE: MEDICINE\_DETAILS**

This table contains information related to Medicine details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| medid | Here Medicine Id will be Primary Key |
| medName | Name of the Medicine e.g., Aspirin, Paracetamol etc. |
| manfdate | Manufacture Date of the Medicine |
| expdate | Expiry Date of the Medicine |
| description | Description of the Medicine |
| price | Price of the Medicine |
| status | Medicine Availability Status, e.g., Pending or Placed |

**5.5 TABLE: CART\_DETAILS**

This table contains information related to cart details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| user\_id | UserId corresponding to logged in user. This field must be foreign key. |
| quantity | Quantity of the Product e.g., 2 or 3 number of Products. |
| item\_id | Item id of the Medicine. This field must contain Primary Key |

**5.6 TABLE: ORDERS\_DETAILS**

This table contains information related to View Orders details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| order\_id | Order Id of the Purchased Medicines. This field must be Primary Key. |
| user\_id | Registered User Id, this field should be Foreign Key. |
| total\_price | Total Price of the Purchased Medicines. |
| order\_date | Date on which Order becomes final. |

**6. REST APIs TO BE BUILT.**

Create following REST resources which are required in the application

**1.Creating User Entity: Create Spring Boot Application with Spring Data JPA**

**Technology stack:**

* Spring Boot: Java Spring Framework (Spring Framework) is a popular, open source, enterprise-level framework for creating standalone, production-grade applications that run on the Java Virtual Machine (JVM).
* Spring REST: REST is basically a convention for building these HTTP web Services. A REST API can help you retrieve information from a system and Spring MVC, a controller can handle the requests for all HTTP methods, which is a backbone of RESTful web services. For example, you can handle a GET method to perform read operations, POST methods to create resources, PUT methods to update resources, and DELETE methods to remove resources from the server.
* Spring Data JPA: Spring Data JPA, makes it easy to easily implement JPA based repositories. It uses a default JPA Implementation called as Hibernate. Spring Data JPA is add-on for JPA. It provides a complete abstraction over the Data Access Layer in a Project.

Here will have multiple layers into the application:

1.Create an Entity: User

2.Create a User signup Repository interface and will make use of Spring Data JPA

a) Will have find By User Name method.

b) Add the User details

3.Create a User signup Service class and will expose all these services.

4.Finally, create a User Rest Controller will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHODS** | **Description** | **Format** |
| /users | GET | This gets function retrieves all users in the database. | JSON |
| /users/userid | GET | This function retrieves the data based on the userid. | String |
| /users | POST | Add the user details | JSON |
| /users/userid | PUT | It modifies the existing user details | JSON |
| /users/userid | DELETE | It Delete user by id | String |

**2. Creating Medicine Entity**:

Build a RESTful resource for Medicinemanipulations, where CRUD operations to be carried out. Here will have multiple layers into the application:

1. Create an Entity: Medicine

2. Create a MedicineRepository interface and will make use of Spring Data JPA

* Will have findByMedicineName method.
* Add the Medicine details method.
* Will have deleteMedicineById method.
* Will have findAllMedicine method.

3. Create a MedicineService class and will expose all these services.

4. Finally, create a MedicineRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHOD** | **Description** | **Format** |
| /Medicines | GET | Get all available medicines | JSON |
| /medicine/id | GET | Get medicine description based on the given id | JSON |
| /medicine | POST | Add the medicine details | JSON |
| /medicine/id | DELETE | Delete a medicine based on medicine id | JSON |

**3.Creating Cart Entity:**

Build a RESTful resource for **Cart** manipulations, where following operations to be carried out. Here will have multiple layers into the application:

1. Create an Entity: Cart

2. Create a CartRepository interface and will make use of Spring Data JPA

* Add the Cart details
* Will have deleteCartById method to remove item with specific product Id from cart.
* Will have All cart details.
* Will have updatecart to change the status from available to sold-out and vice-versa.

3. Create a CartService class and will expose all these services.

4. Finally, create a CartRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHOD** | **Description** | **Format** |
| /Cart | GET | It gets available cart details | JSON |
| /Cart | POST | Add the medicines to cart | JSON |
| /Cart/id | PUT | Update the cart details | JSON |
| /Cart/id | DELETE | Delete the medicines from cart | JSON |

**4. Creating order Entity:**

Build a RESTful resource for ordermanipulations, operations to be carried out. Here will have multiple layers into the application:

1. Create an Entity: Vieworder

2. Create a OrderRepository interface and will make use of Spring Data JP

* Will have findByorderName method
* Add the order details method
* Will have deleteorderById method
* Will have findAllorders method

3. Create a VieworderService class and will expose all these services.

4. Finally, create a MedicineRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHODS** | **Description** | **Format** |
| /orders | GET | Get all orders. | JSON |
| /order/id | GET | Get orderdescription based in the given id. | JSON |
| /order | POST | Add the order details | JSON |
| /order/id | POST | Delete order based on medicine id | JSON |

**7.FEASIBILITY STUDY:**

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements

**A. Economic Feasibility**

This is a very important aspect to be considered while developing a project. We decided the technology based on minimum possible cost factor.

• All hardware and software cost has to be borne by the organization.

• we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for system

**B. Technical Feasibility**

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system.

**C. Operational Feasibility**

This Project is very user friendly and all inputs to be taken all self-explanatory even to a layman.

**8. ASSUMPTIONS**

Project assumption is the events or conditions that are most likely to occur when a project life-cycle takes place. It is also deemed as an element in the planning phase of a particular project that is assumed to be considered as true and actual. Assumptions are beliefs based on previous experience and the information available to you.

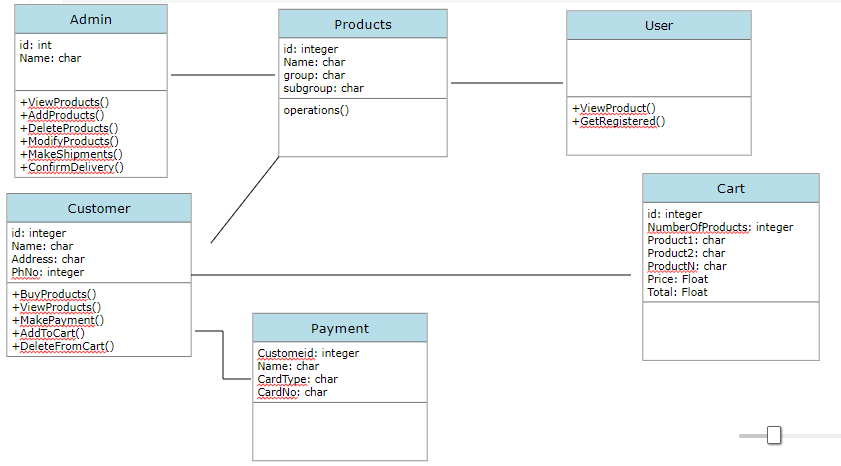
* The product does require back-end database server MySQL for storing for storing the username and password for different types of users of the system.
* User must be trained for basic computer functionalities.
* User must have the basic knowledge of English.
* The system must be able to respond to database software within reasonable time
* User Interface: The type of client interface (front-end) to be supported - Angular based
* The administrator can add and remove medicine products into the database on a weekly basis.
* You must not allow user to add same medicine product twice.
* When you add medicine product into cart the No. of medicine Products selected will be incremented.
* If you delete the medicine product from the cart, the counter will be decremented.
* The clear will remove all the medicine products so that the No. of medicine products will be zero
* The total amount will be calculated based on the No. of medicine product, accordingly, change the product counter & total amount.

**9.UML DIAGRAMS**

Unified Modeling Language (UML) is a general-purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

**9.1 CLASS DIAGRAM:**

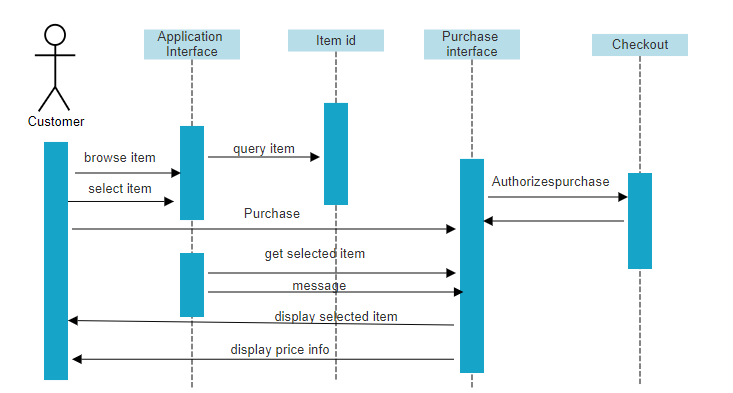
* Admin: Mainly responsible for account management and adding or modifying new product categories.
* Customer: Members can perform all the activities that guests can, in addition to which, they can place orders and add new products to sell.
* Product: This class will encapsulate the entity that the users of our system will be buying and selling. Each Product will belong to a Product Category.
* Guest: Guests can search for and view products, and add them in the shopping cart. To place an order, they have to become a registered member.
* Cart: Users will add product items that they intend to buy to the shopping cart.
* Payment: This class will encapsulate the payment for an order. Members can pay through credit card or electronic bank transfer.

****

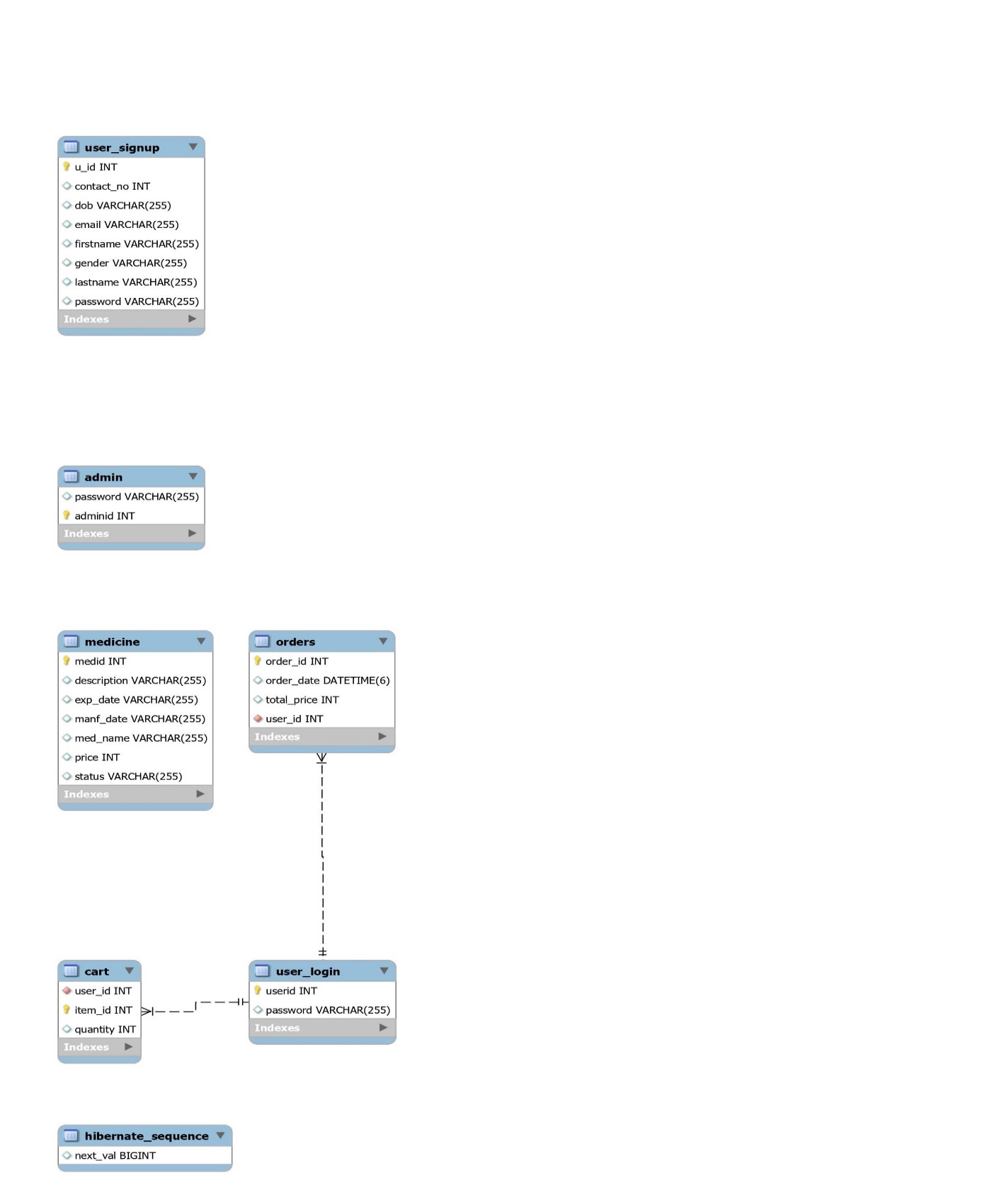
**9.2 SEQUENCE DIAGRAM:**

A sequence diagram is structured in such a way that it represents a timeline which begins at the top and descends gradually to mark the sequence of interactions. Each object has a column and the messages exchanged between them are represented by arrows.

The diagram presented here will show you a detailed illustration of the sequence of events happen in E-Medicare System.

****

**9.3 ER DIAGRAM:**

****

**10.TESTING:**

**1.Unit Testing:**

Each module is considered independently. it focuses on each unit of software as implemented in the source code it is white box testing.

**2.Integration Testing:**

Integration testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules.

**3.Validation Testing:**

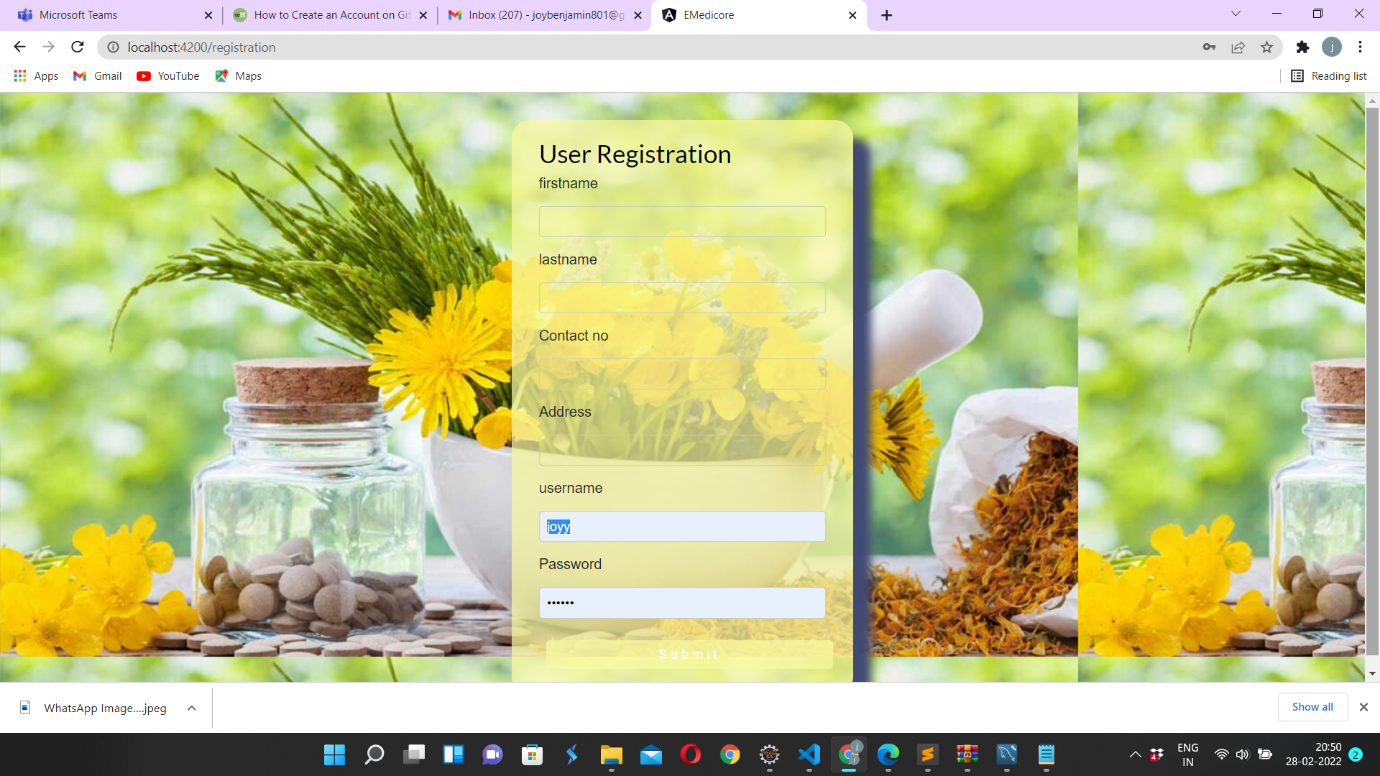
Validation testing was performed to ensure that all the functional and performance requirements are met.

**4.System Testing:**

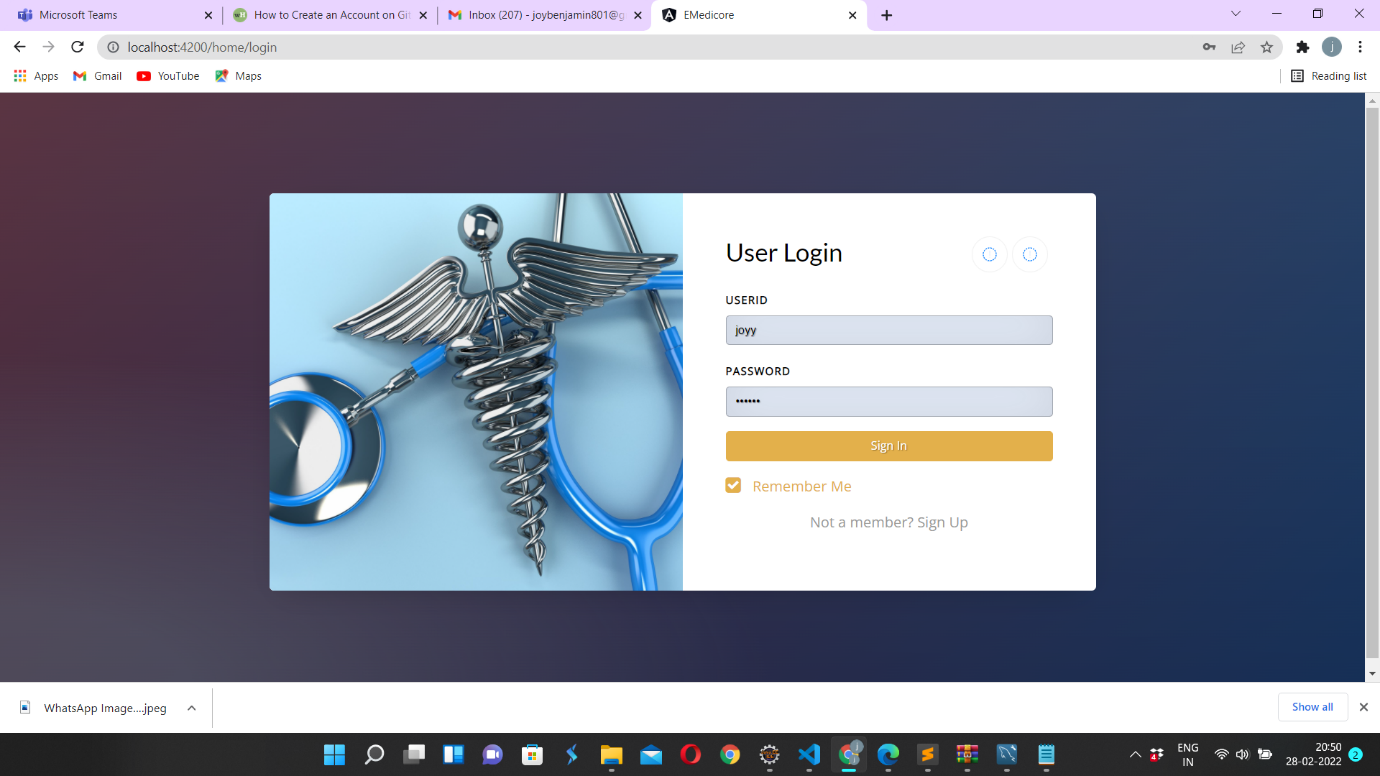
It is executing programs to check logical changes made in it with intention of finding errors, a system is tested for online response, volume of transaction, recovery from failure etc.

**11.Output Screenshots**

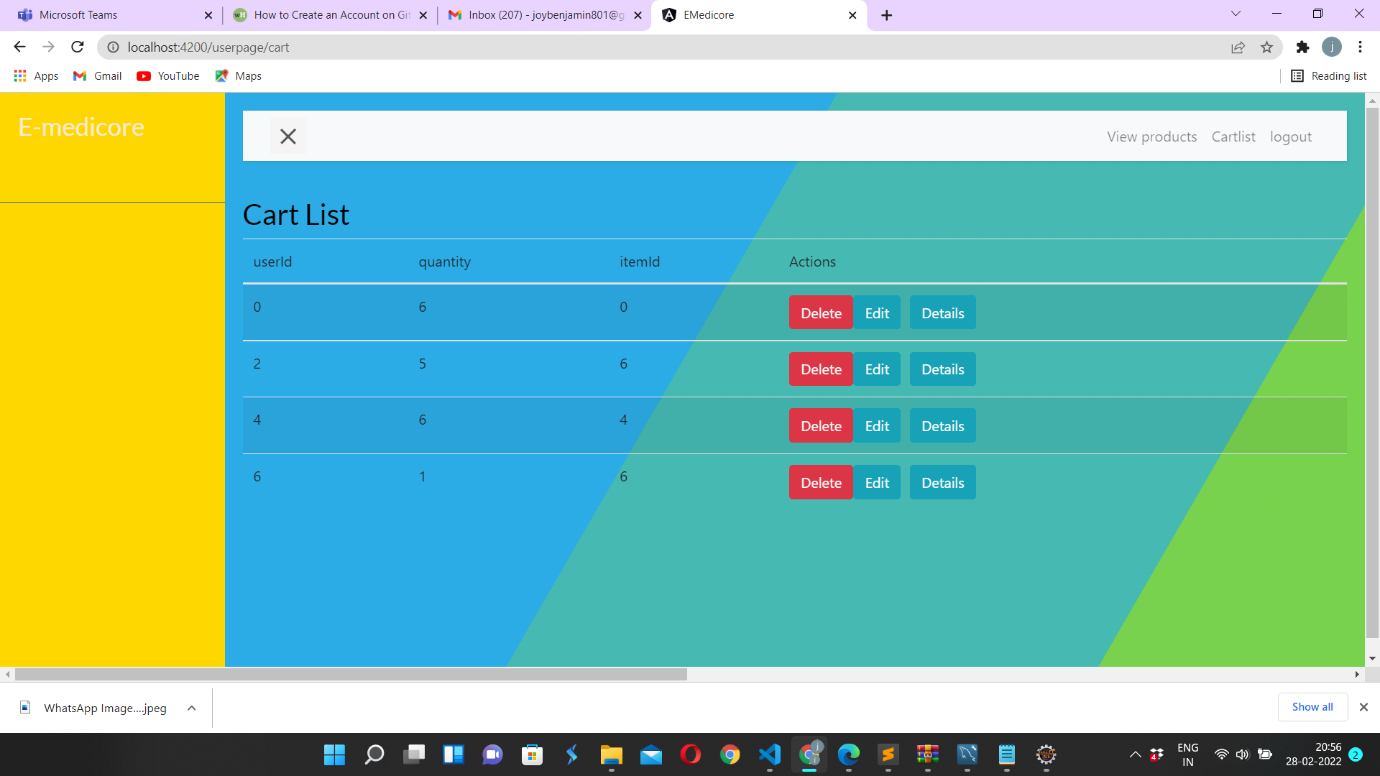
**Sign Up:**



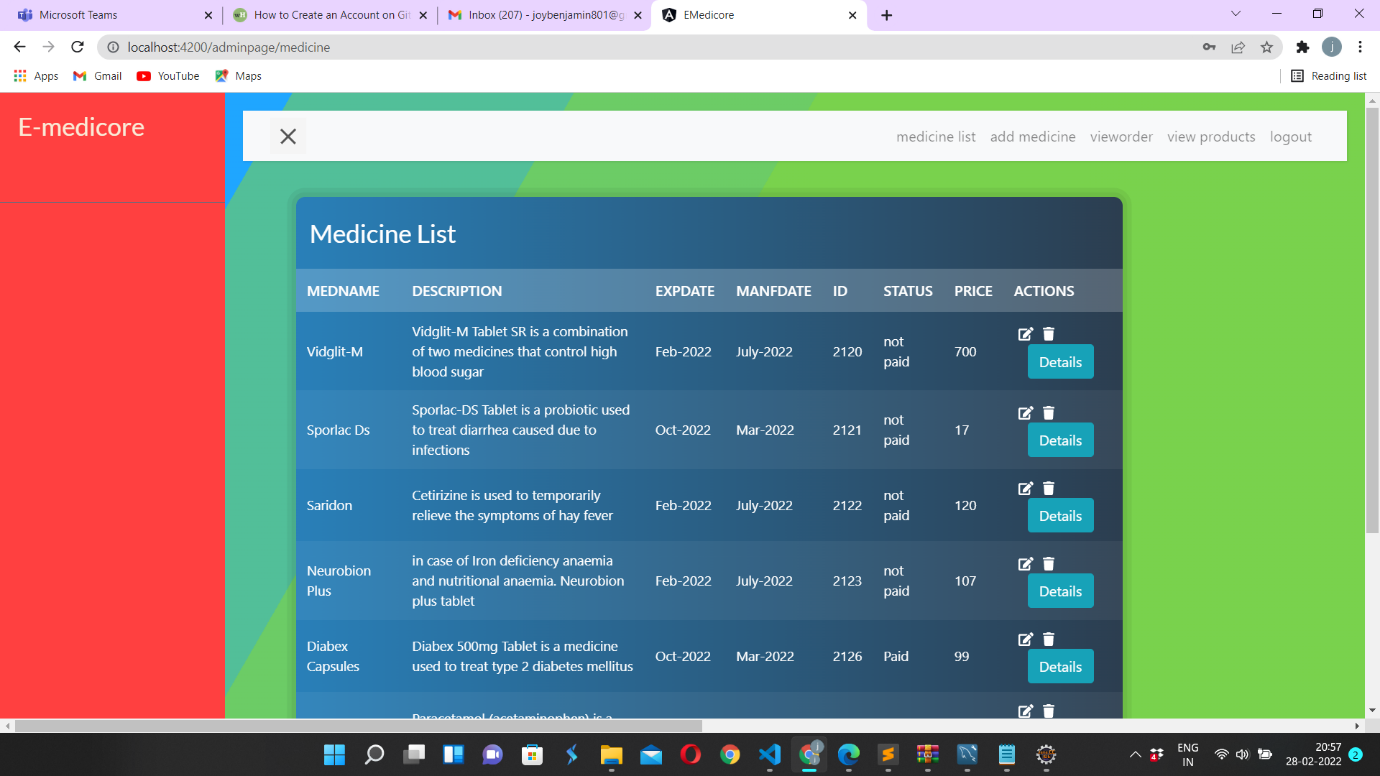
**User Sign In:**



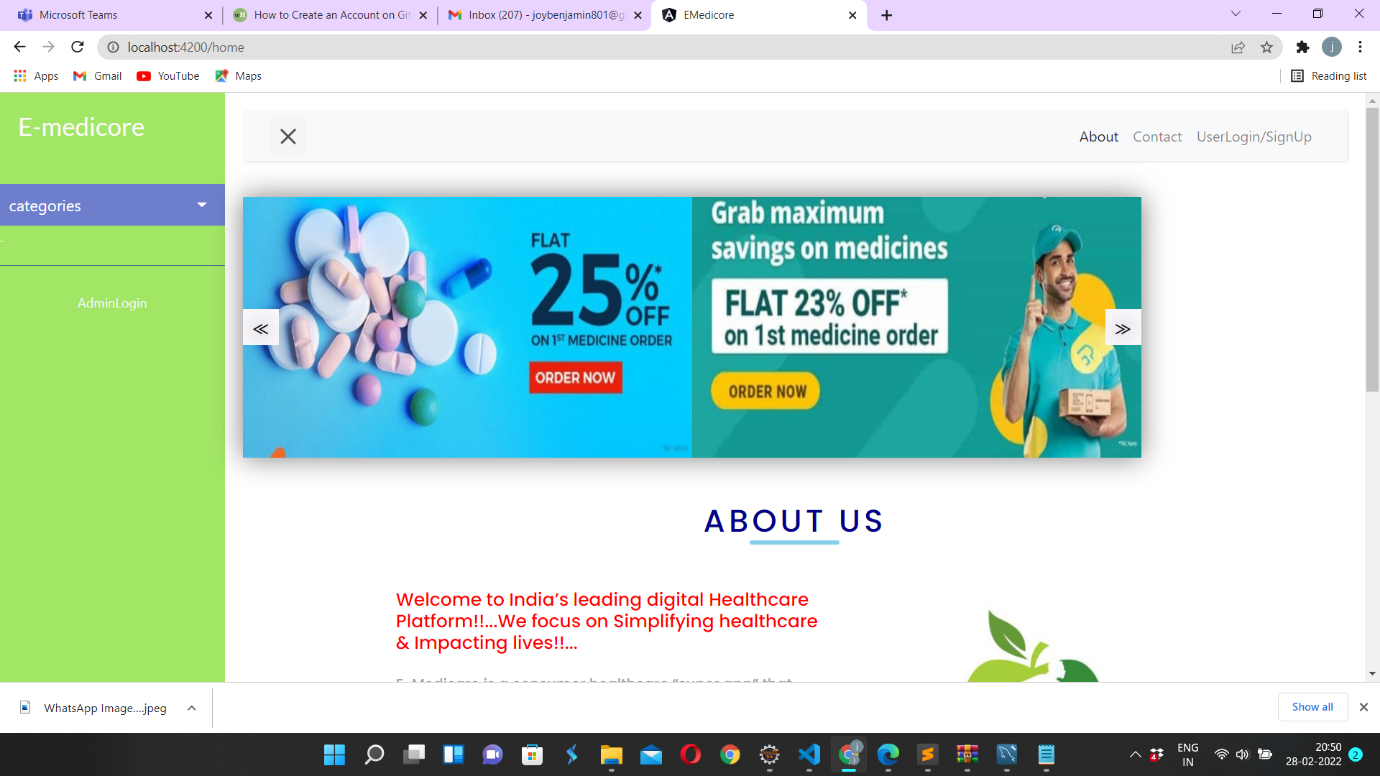
**Cart List:**



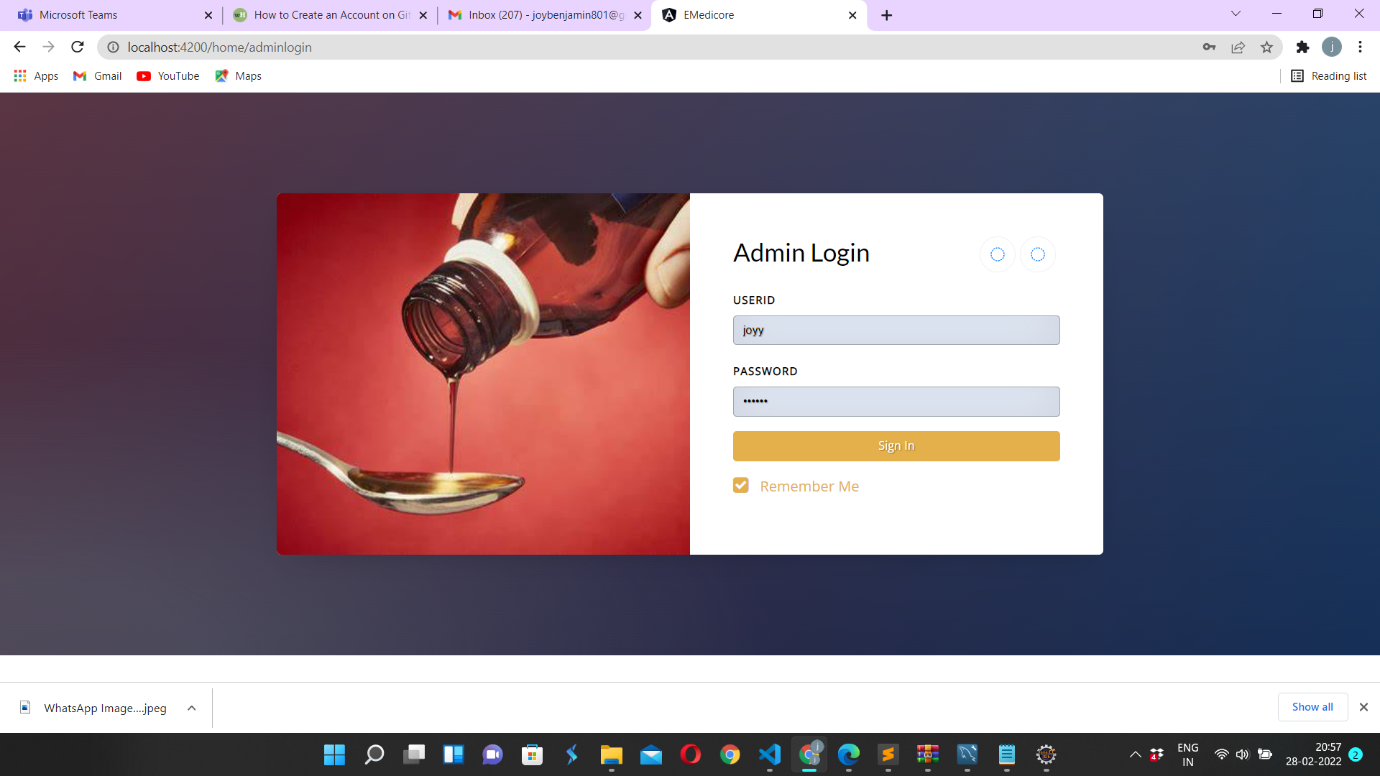
**Medicine List:**



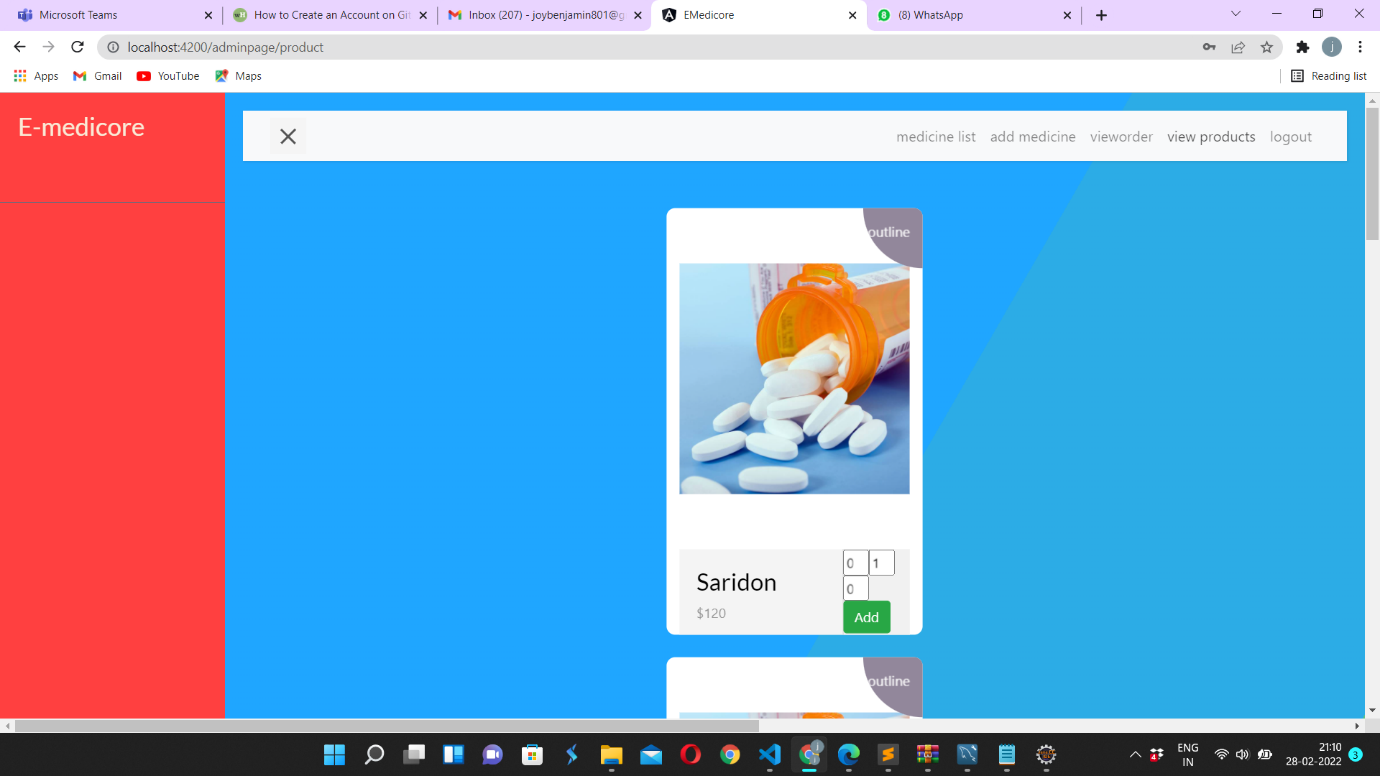
**Home Page:**

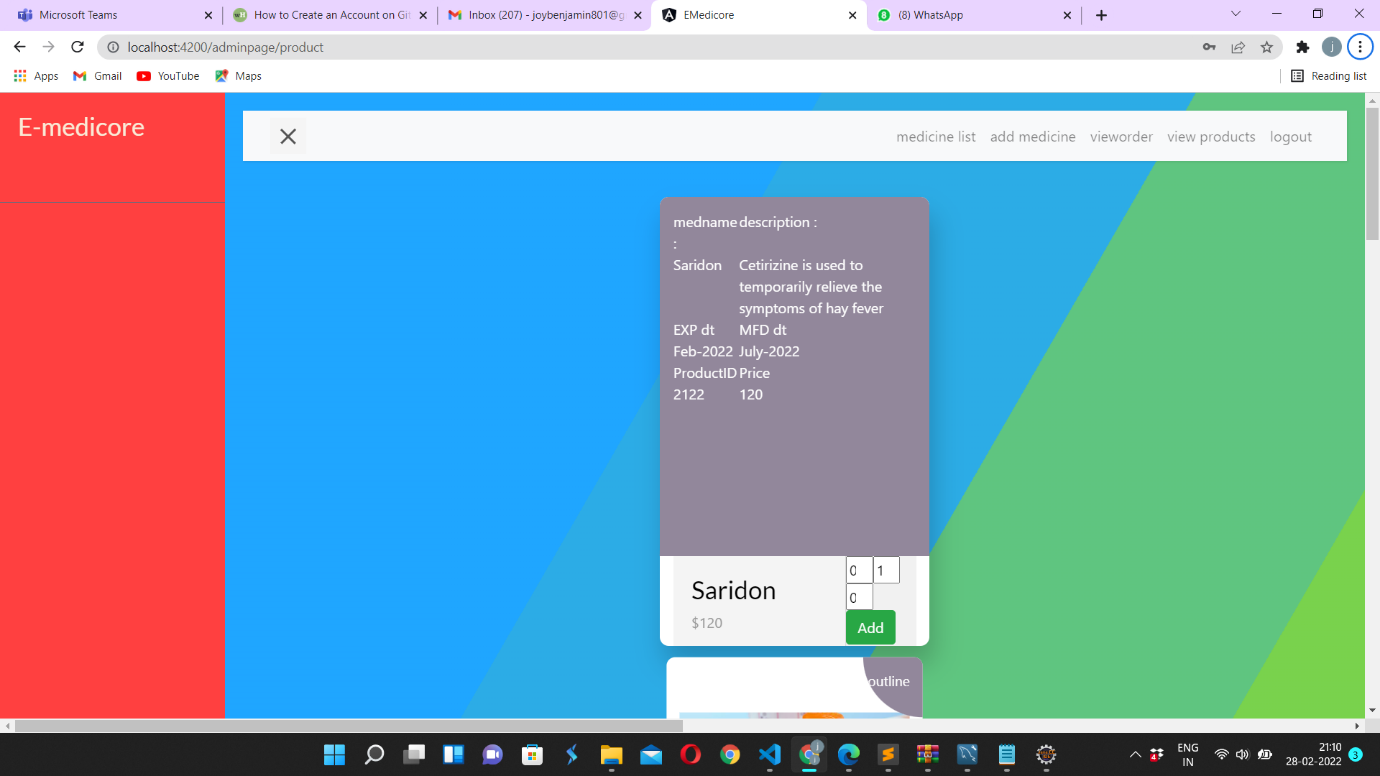


**Admin Sign In:**

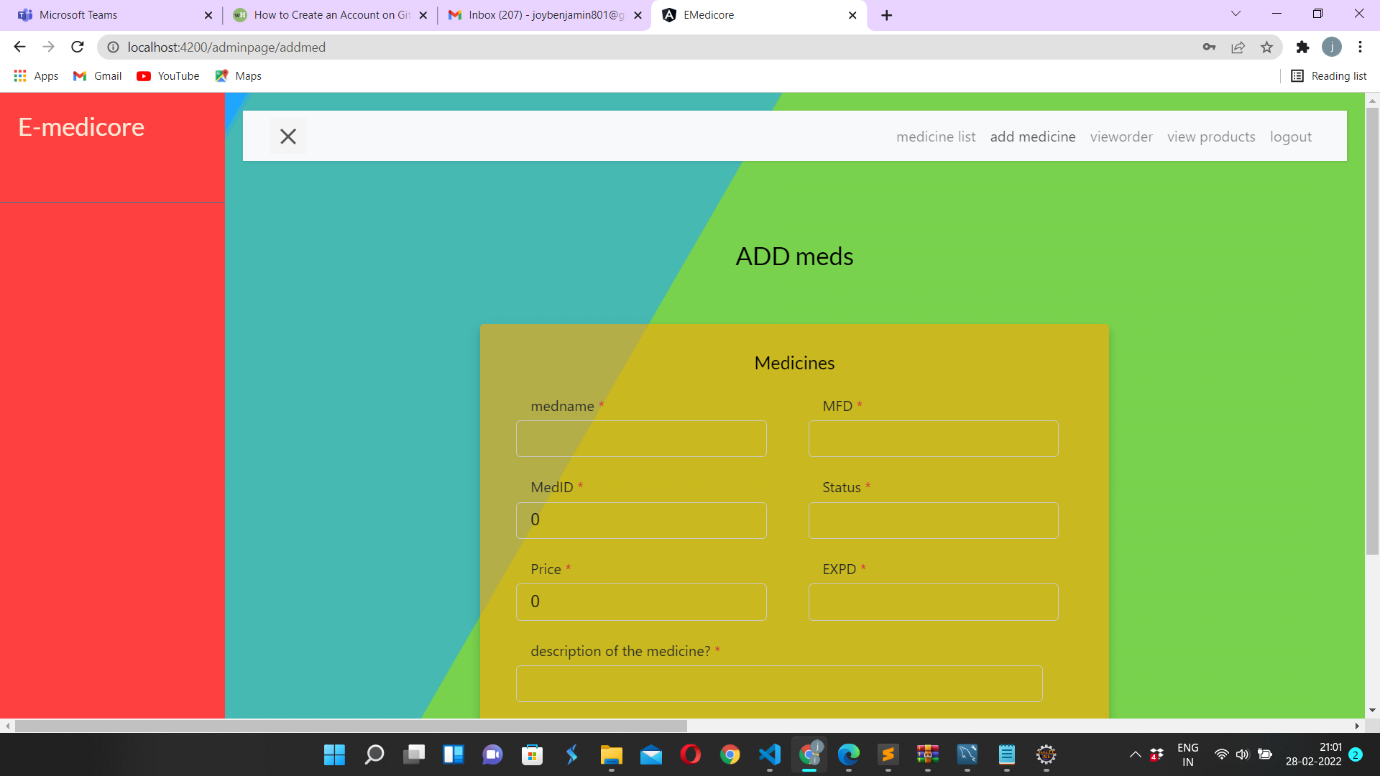


**View products:**

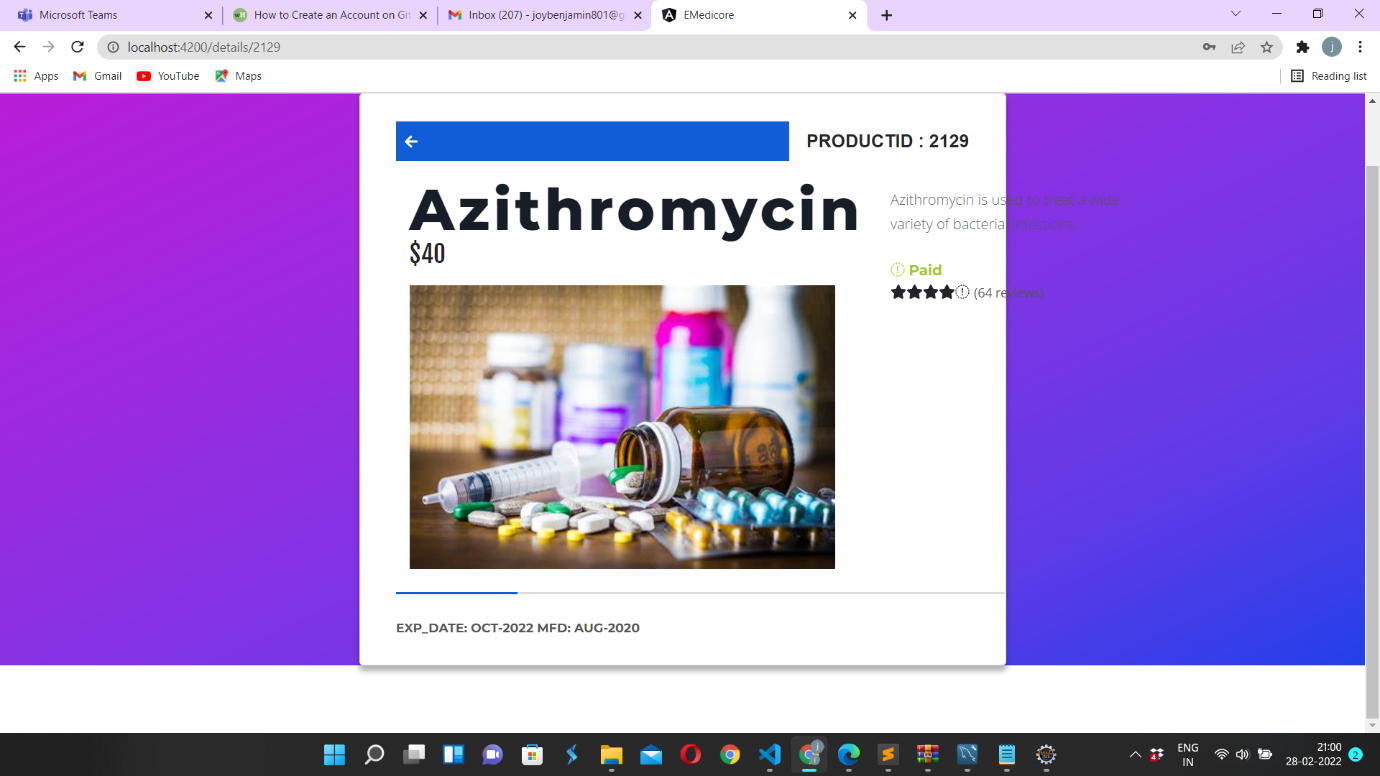




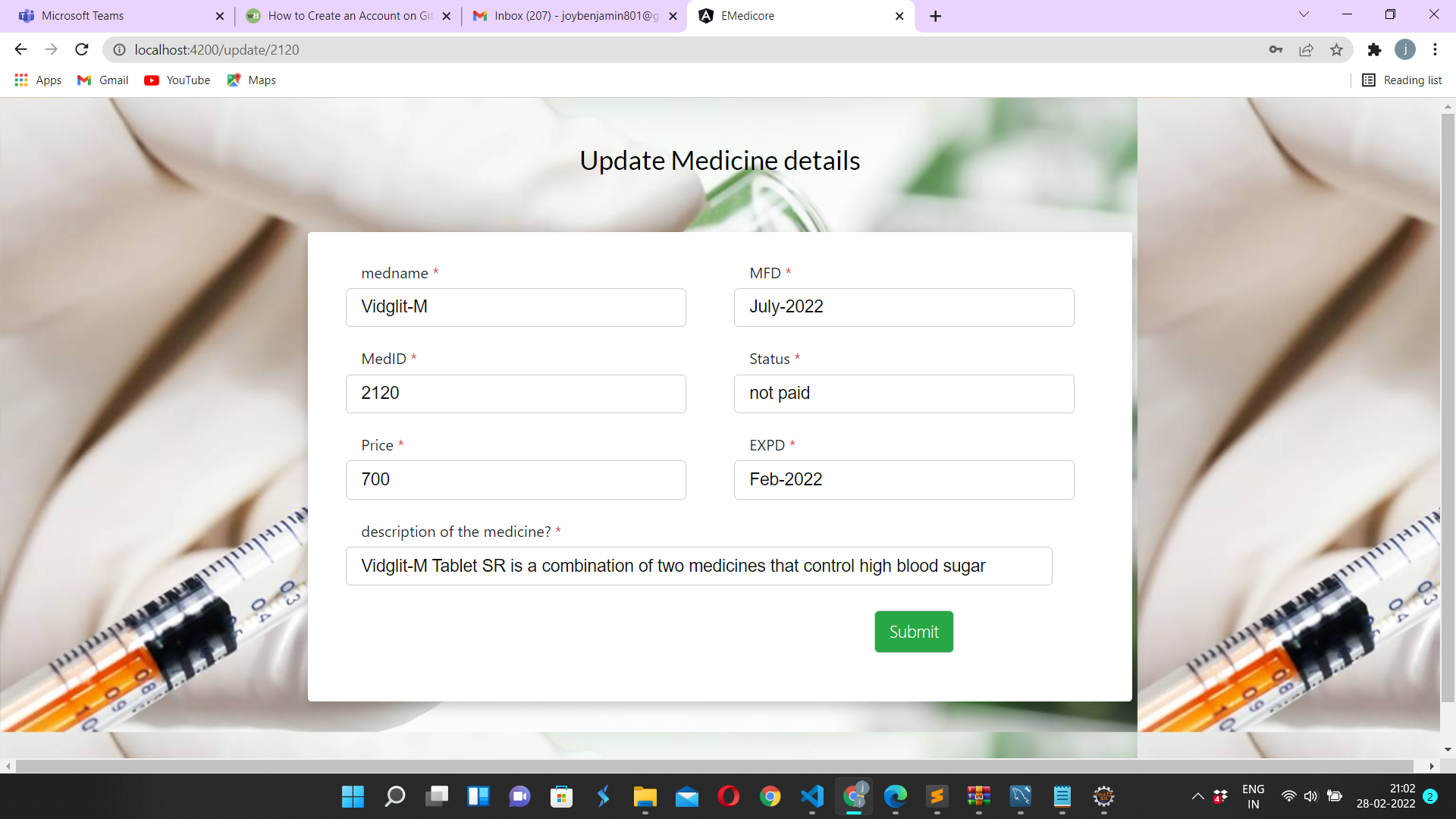
**Add Medicines List:**



**Medicines details: after clicking details in medicine list**



**Update Medicines List:**



**About us:**

