**Project: E-Medicare**

|  |  |  |
| --- | --- | --- |
|  | **FUNCTIONAL SPECIFICATION** | |
|  |  |  |
| **Project Team:** | **9** |  |
| **Project Name:** |  | E-Medicare |

**Table of Contents**

1. INTRODUCTION………………………………………………………………………………

2. SYSTEM OVERVIEW …………………….………………………………………………..…

3. SUB-SYSTEM DETAILS ……………...……………………………………………………...

4. DATA ORGANIZATION………………………………………………………………………

5.  REST API’s TO BE BUILT  ………………………………………………………………….

6. ASSUMPTIONS ……………………………………………………………………….....…….

7. EXPECTATIONS ……………………………………………………………………………….

8. ADVANTAGES…………………………………………………………………………………

9.ACRONYMS..............................................................................…………………………………

10.OUTPUT SCREENSHOTS FOR YOUR PROJECT…………………………………………...

11.CONCLUSION…………………………………………………………………………………

**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.

**1.** **Introduction**

E-Medicareis a web application which is **developed in Java.**  this project is to develop an online web portal that can handle Medicine information, can booking from distributors very fast from all over the world and customer support for distributors. This type of application will atomize the procedure of medicine supply through the Pharmaceuticals Company and improve business standards and customer relationships. This application is use distributors can view detailed information of transactions and get medicine information and see future orders from his account. Customers can easily visit this site and register them self, by filling a registration form. Once a customer is registered he/she can login using their email and password and can buy products available.

**Overview**

People need medicine when they become sick and unable to enjoy their normal life. People have to go in the pharmacy for buying medicine. But sometimes they can’t get the proper medicine due to the lack of easy availability. So, the patients have to search for the desired medicine in every medicine shop which wastes their time and energy. Sometimes, in emergency cases, the condition of patients become very serious without taking medicine at the right time.

E-Medicare will give the opportunity to buy proper medicine sitting at home without any trouble. There are many medicine shops in every city but online medicine shop is very rare. The online medicine shop is operated over the internet. It will bring comfort to every buyer and seller of the medicine. The seller can manage the shop, store the details and relevant information of the medicine and sell the medicine in one system easily. The customer can see, search, buy medicine and give review about the service quality in this website. So, Online Medicine Shop will create a convenient way of selling and buying medicine and will make our life easier and hassle free.

**Objective**

The main objective is to manage the details of customer, vendor, medicine, stock, order and sell the medicine in online. In this technological era, everything becomes very faster. People love to enjoy a comfortable life. This online medicine shop will bring many facilities to our life. The seller can easily sell the medicine and preserves the records and documents of the medicine. And the profit will be high also. That’s why, Online Medicine Shop is developed where the management of the medicine shop is web based through which one can manage a medicine shop easily from anywhere at anytime.

**Scope of the Project:-**

1. To provide the better stock maintenance of the medicines.
2. To generate sales/purchase report.
3. To manage the transaction such as product sales and other expenses.
4. Least amount of medicines allows alerting at the login.
5. Life of (expiry date) medicines allows informing before two months.
6. Keep track status of its medicine details.

All this work is done manually by the admin and other Medicare staff. Admin have to remember various medicines available for customers.

**TECHNOLOGY USED**

* HTML : Page layout has been designed in HTML
* CSS : CSS has been used for all the designing part
* JavaScript : All the validation task and animations has been developed by JavaScript
* JSP : All the front end logic has been written in JSP
* Java : All the business logic has been written in Java
* My-SQL : My-SQL database has been used as database for the project
* Angular CLI : Command-line interface tool that we use to initialize.

**2**. **System Overview**

The “E-Medicare” should support basic functionalities for all below listed users.

* Admin Module: Used for managing medicine details and user information.
* Users Module: Used for managing the users of the system.
* Login Module: Used for managing the login details.
* Contact\_us: Used if customer is having complaints
* Medicine Module: Used for managing the Medicine details.
* Cart Module: Used for managing the details of Cart.
* Payment Module: It is used for managing the payment details.

**2.1 Authentication & Authorization**

**2.1.1** **Authentication**:

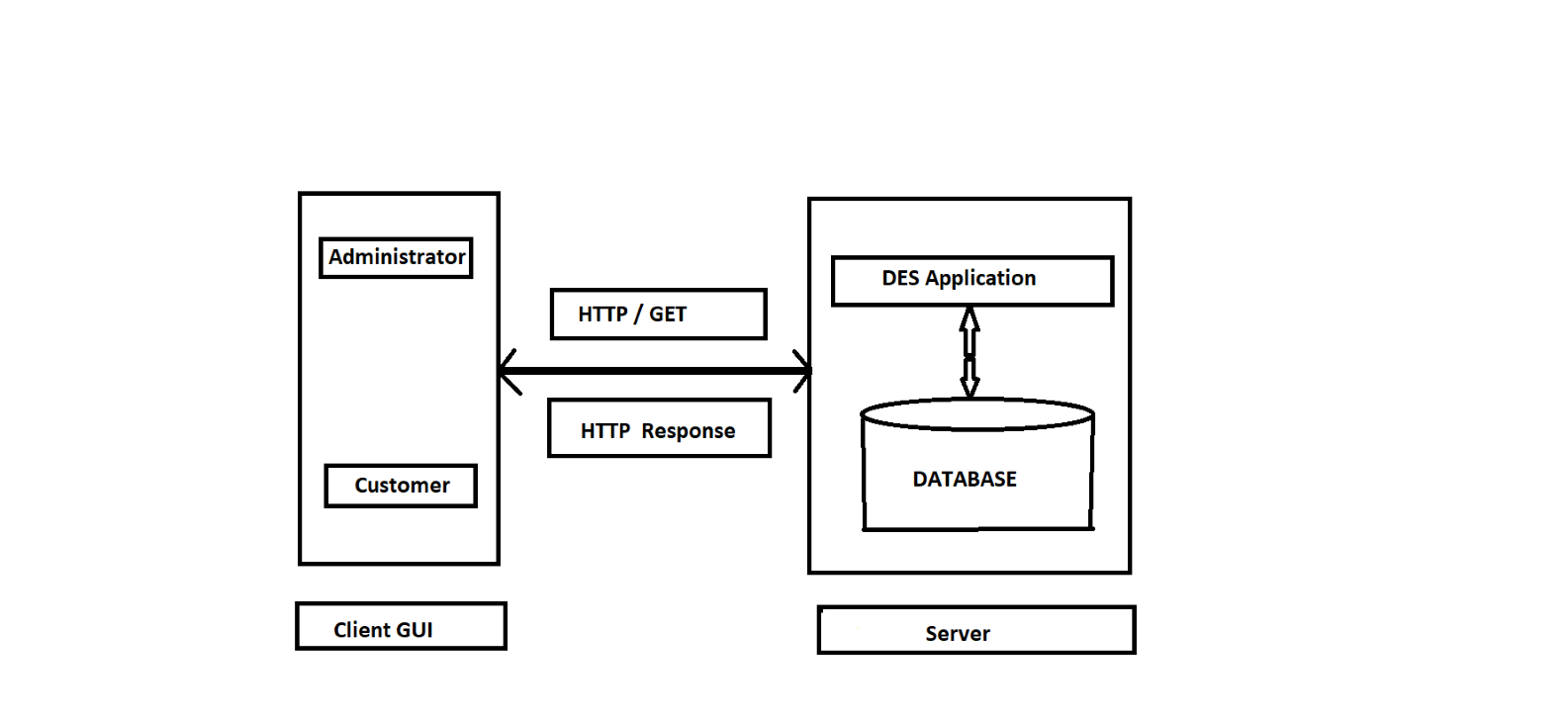
Any end-user should be authenticated using a unique userid and password.

**2.1.2** **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 medicines

Whereas User/Buyer has a right to Add, Remove and Clear all the products from cart.

**2.2 Functional Flow**

The functional flow of the messages across different application components is shown below. Ex.WebApplication. 

|  |  |  |
| --- | --- | --- |
| **2.3 Environment:** |  |  |

#### HARDWARE REQUIREMENTS:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list , especially in case of operating systems. An hardware compatibility lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

#### SOFTWARE REQUIREMENTS:

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

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

**3.** **Sub-system Details**

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

Find below tables that provides functionality descriptions for each type of user / sub-system. Against each requirement, indicative data is listed in column ‘Data to include’.

**3.1 Admin**

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

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Objects | Operations | Data to include |
| 1       To  10 | Medicines | Add  View  Delete  Modify  Search | Medicine\_id, desc,Manufacture\_date,medicine\_name,price,seller,type,expdate,status |
| 5       To  10 | Users | View | User\_id,age,first\_name,last\_name,gender,Mob\_no,username,password. |
| 52        to  53 | Cart | View | Medicine\_Id, description,Medicine\_name,  Price. |

**3.2 Customer**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Objects | Operations | Data to include |  |
| 1 | User | Register | UserId, Username, Password, Email, Phone Number,etc. |  |
| 2 | Medicines | Add to Cart.  Delete from Cart.  Delete all products from cart. | Medicine\_Id, Medicine\_name , Price, Quantity, Status |  |
| 3 | Checkout | Add User Details   and Price | CartId, UserId and Total Price |  |

**3.3 Login | Logout**

* 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.

**Password Rules:**

When it comes to password safety, the longer and more complex the password is, the better. We think its good practice to enforce certain minimum requirements when asking users to create a new password. Of course, you have to find a balance between these requirements and user experience. If you make the sign-up process too tedious, you could be driving users away. To enforce password strength, you should define a set of rules that a password must satisfy and then enforce these with form validation.

Example password strength rules:

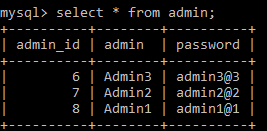
* + - Minimum of 8 characters
    - At least one uppercase letter
    - At least one number
    - At least one special character

**4.** **Data Organization**

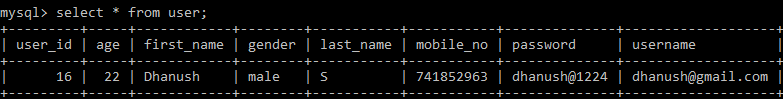
MySQL is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications. The most common use for mySQL however, is for the purpose of a web database.

**4.1 Table: Admin**

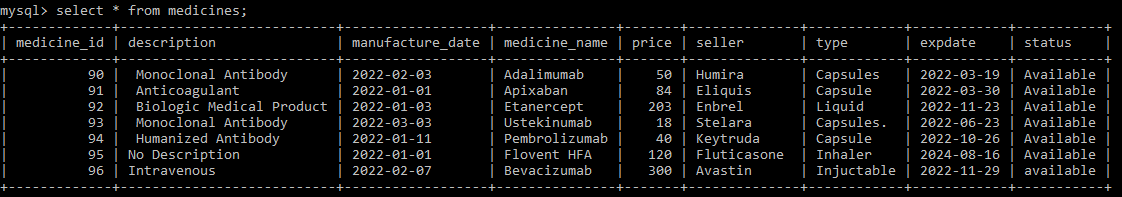
The Admin specific details such as username,password etc. Authentication, and authorization / privileges should be kept in one or more tables, as necessary and applicable.



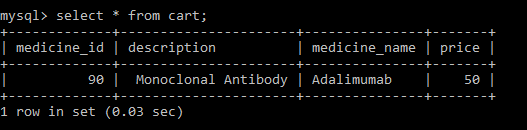
**4.2 Table: user**his table contains information related to a User



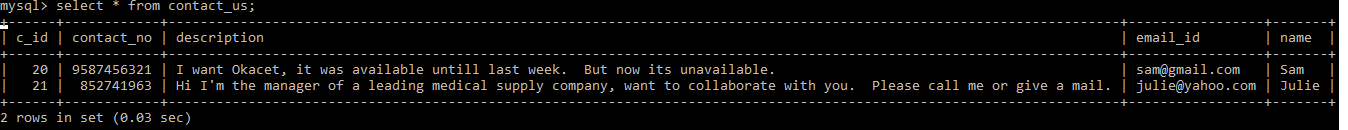
**4.3 Table: Medicines** his table contains information related to a Medicines



**4.4 Table:Cart** This table contains information related to cart details.



**4.5 Table: Contact\_Us** his table contains information related to a Contact\_Us



**5.  REST API’s TO BE BUILT**

Create following REST resources which are required in the application,

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

with Spring Data JPA

**Technology stack:**

* Spring Boot
* Spring REST
* Spring Data JPA

Here will have multiple layers into the application:

1. Create an Entity: User
2. Create a UserRepository interface and will make use of Spring Data JPA

1. Add the User details
2. Create a UserService class and will expose all these services.
3. Finally, create a UserRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHODS** | **Description** | **Format** |
| /api/users | POST | Add the user details | JSON |

1. Creating **Medicine** Entity:

Build a RESTful resource for **Medicines** manipulations, 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 :
3. Will have findByMedicineName method.
4. Add the Medicine details method.

1. Will have deleteMedicineById method.

1. Will have findAllMedicines method.
2. Create a MedicineService class and will expose all these services.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHODS** | **Description** | **Format** |
| /api/v2/medicines | GET | Get all the medicines | JSON |
| /api/v2/medicines | GET | Add a single medicine | JSON |
| /api/v2/medicines | POST | Update medicine | JSON |
| /api/v2/ /medicines/{medicineId} | DELETE | Delete a medicine based on medicine id | JSON |

1. Creating **Adminlogin** Entity:

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

1. Create an Entity: Adminlogin
2. Create a  adminlogin Repository interface and will make use of Spring Data JPA

1. Add the adminlogin details

1. Create a Adminlogin Service class and will expose all these services.
2. Finally, create a Adminlogin RestController will have the following uris:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHODS** | **Description** | **Format** |
| /api/v1/admins | POST | Add a single admin. | JSON |

4.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:

5.Create an Entity: Cart

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

a.Add the Cart details.

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| URI | METHODS | Description | Format |
| /api/v5/cart/{medicineId} | POST | Add the user details with total price. | JSON |
| /api/v5/cart | GET | Add a single medicine | JSON |
| /api/v5/cart/{medicineId} | DELETE | Delete a medicine based on medicine id | JSON |

**6.Assumptions :**

* User Interface: The type of client interface (front-end) to be supported - Angular based

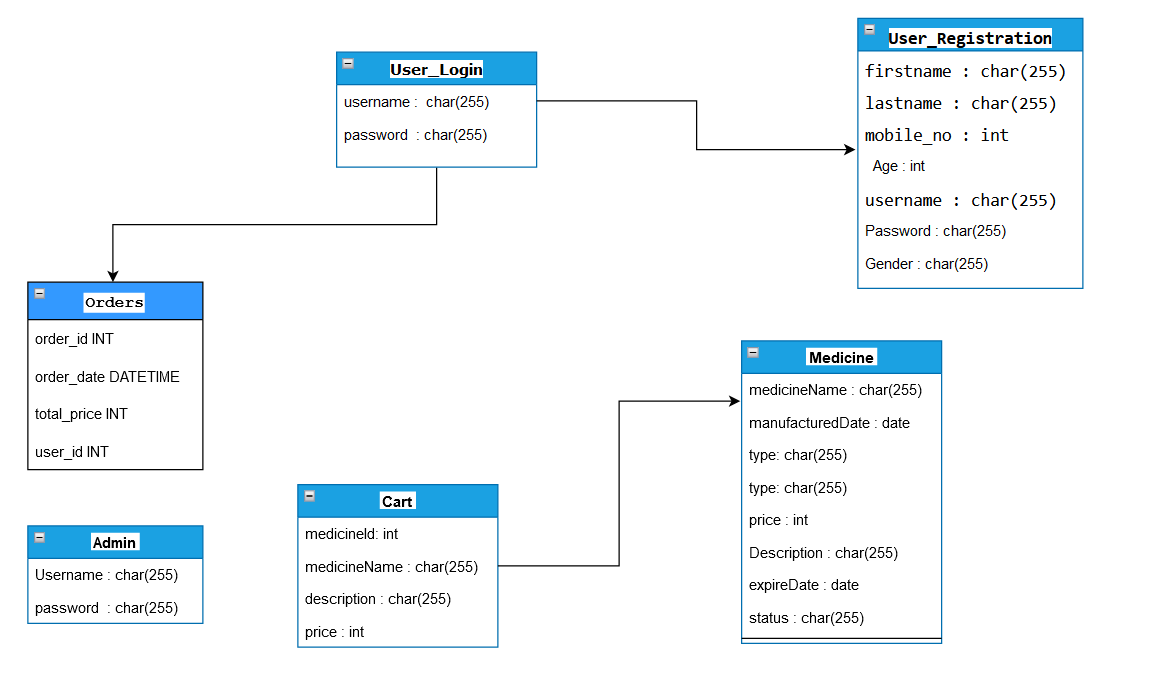
* The administrator can add and remove medicines into the database on a weekly basis.
* You must not allow user to add same medicine twice.

* When you add medicines into cart the No. of medicines selected will be incremented.
* If you remove the medicine from the cart, the counter will be decremented.
* The clear will remove all the medicines so that the No. of products will be zero
* The total amount will be calculated based on the medicine, accordingly, change the medicine counter & total amount.

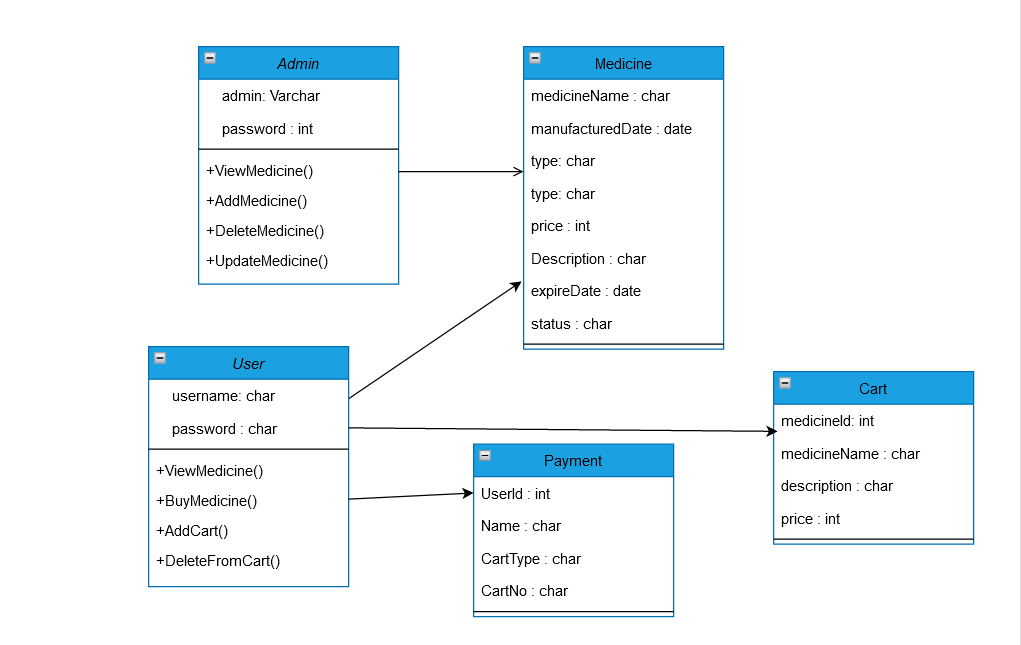
**7.General Expectations:**

* The server should be a concurrent server servicing multiple clients.
* Database can be implemented using Oracle 11g or above.
* To begin with, the application should support at least 1 admin and 2 customers.
* Compilation and Build should be done using Eclipse IDE or STS

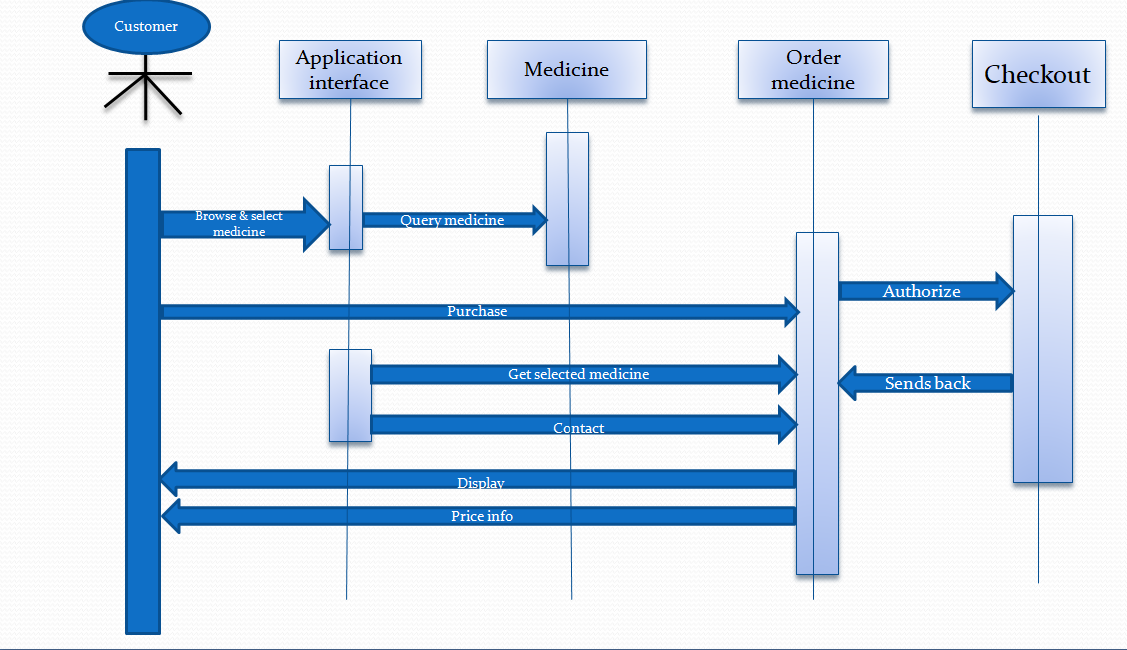
**7.1 ER Diagram:**



**7.2 Class Diagram:**

****

**7.3ER-Diagram:**

****

**7.4** **Unit testing :**

JUnit is a unit testing framework for Java programming language. It plays a crucial role test-driven development, and is a family of unit testing frameworks collectively known as xUnit.

JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. This approach is like "test a little, code a little, test a little, code a little." It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging.

**Features of JUnit:**

* JUnit is an open source framework, which is used for writing and running tests.
* Provides annotations to identify test methods.
* Provides assertions for testing expected results.
* Provides test runners for running tests.
* JUnit tests allow you to write codes faster, which increases quality.
* JUnit is elegantly simple. It is less complex and takes less time.
* JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
* JUnit tests can be organized into test suites containing test cases and even other test suites.
* JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

**Unit Test Case**

A Unit Test Case is a part of code, which ensures that another part of code (method) works as expected. To achieve the desired results quickly, a test framework is required. JUnit is a perfect unit test framework for Java programming language.

A formal written unit test case is characterized by a known input and an expected output, which is worked out before the test is executed. The known input should test a precondition and the expected output should test a post-condition.

There must be at least two unit test cases for each requirement − one positive test and one negative test. If a requirement has sub-requirements, each sub-requirement must have at least two test cases as positive and negative.

**7.5 Functional Testing using POSTMAN tool:**

Tests are automated by creating test suites that can run again and again. Postman can be used to automate many types of tests including unit tests, functional tests, integration tests, end-to-end tests, regression tests, mock tests, etc. Automated testing prevents human error and streamlines testing.

**8.Advantages**

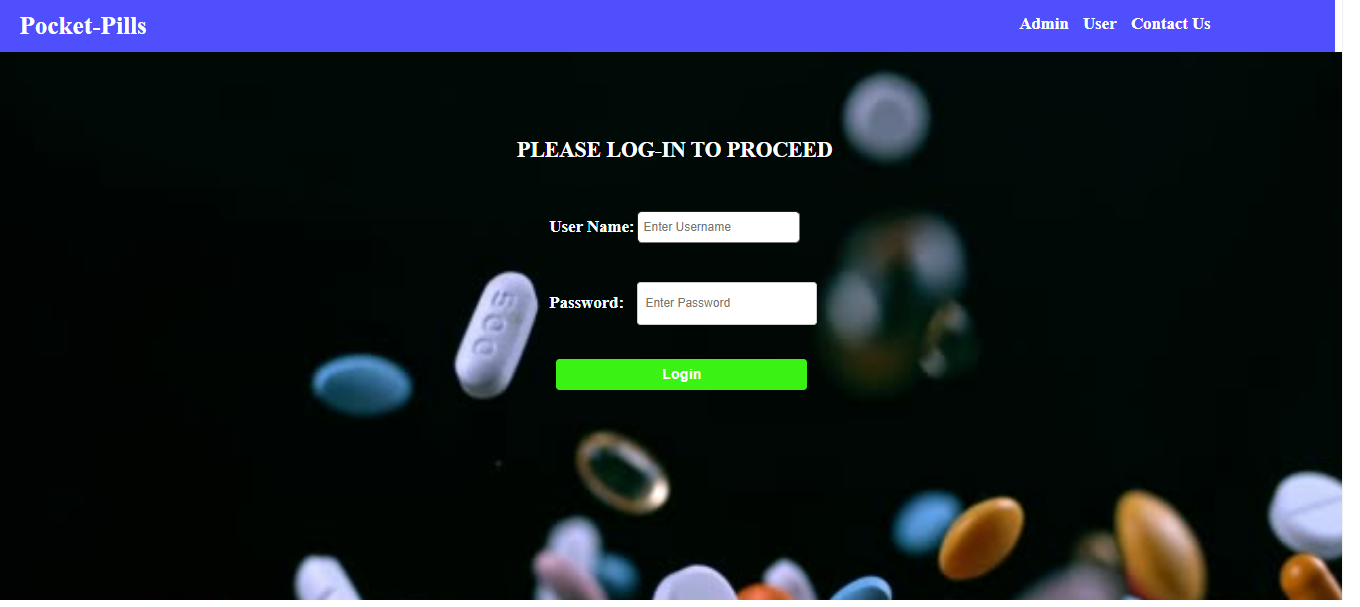
* User can view details of the medicines without going anywhere.
* It is convenient for users as this system provides accurate cost and description of the system.
* The website is flexible to be used and for e-shopping.
* User can view different categories of product of different pharma company at a single place.

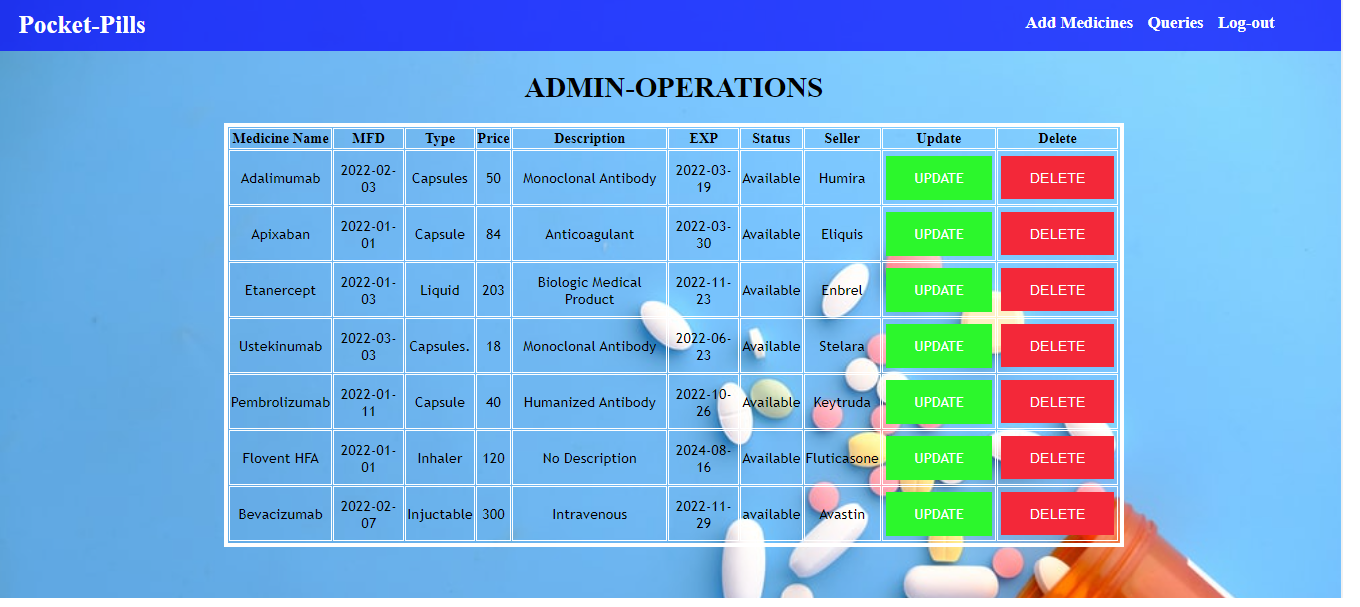
**9.Acronyms and Glossary**

Acronym and glossary for this document mentioned in the below table.

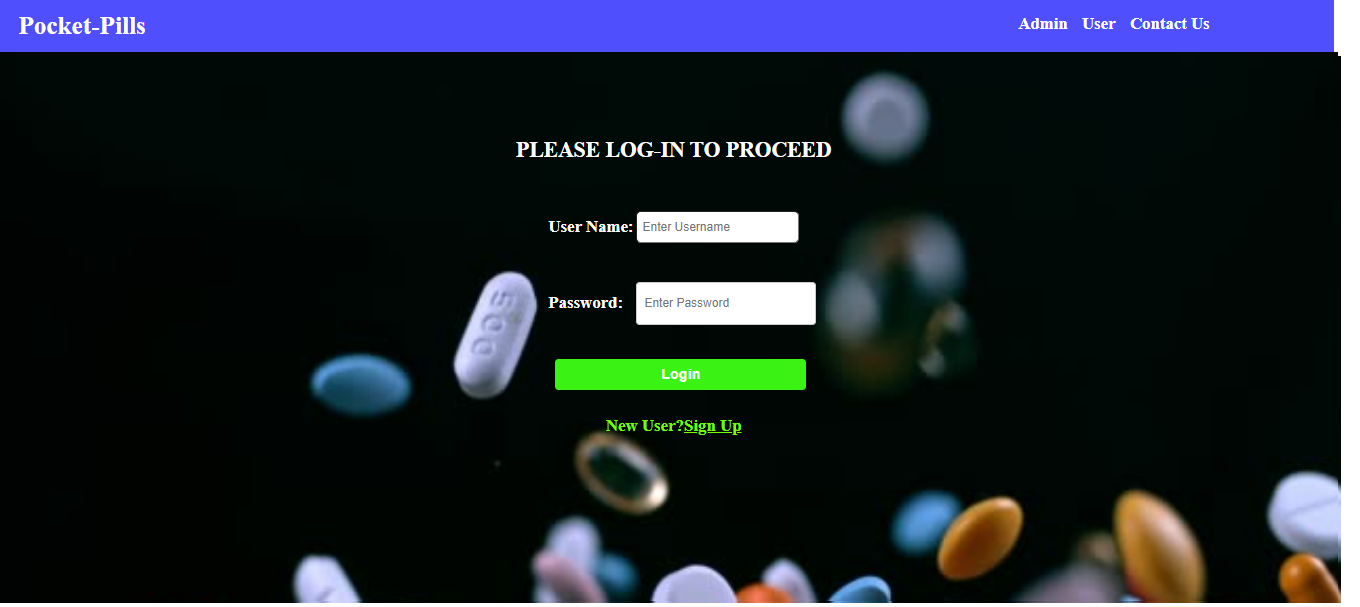
|  |  |
| --- | --- |
| **Abbreviation** | **Remark** |
| Uml | Unified Modeling Language, |
| ER Diagram | entity relationship diagram |
| JUnit | Joint unit |

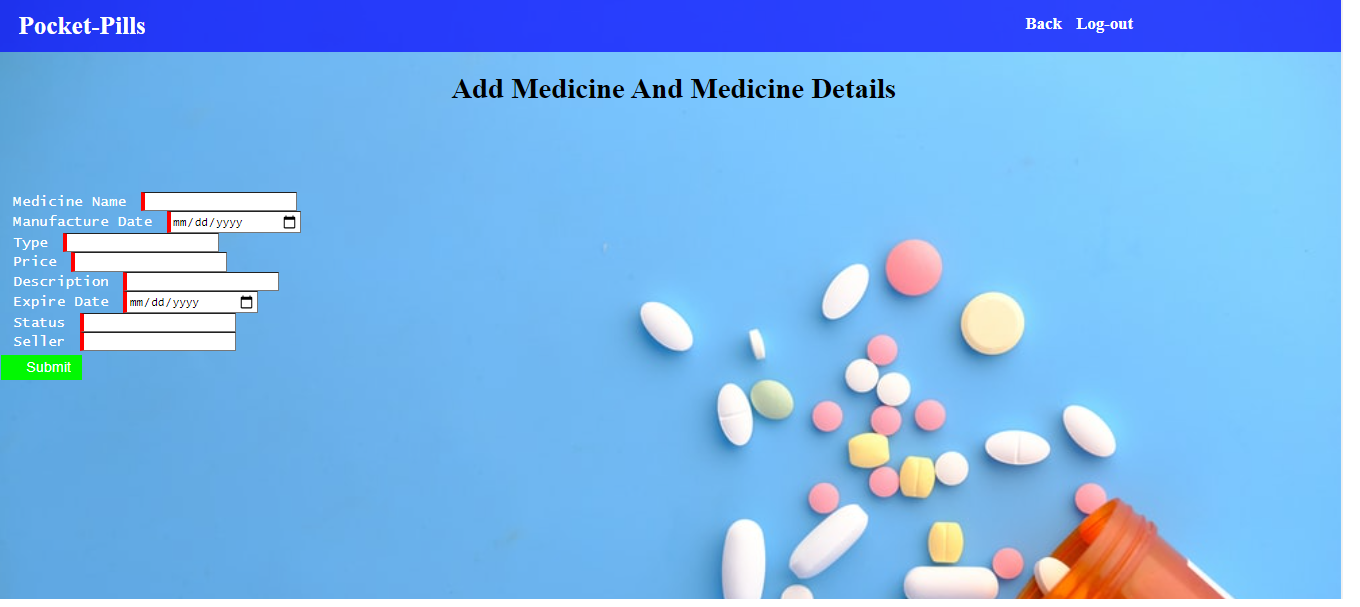
**10.Output Screenshots for your Project**



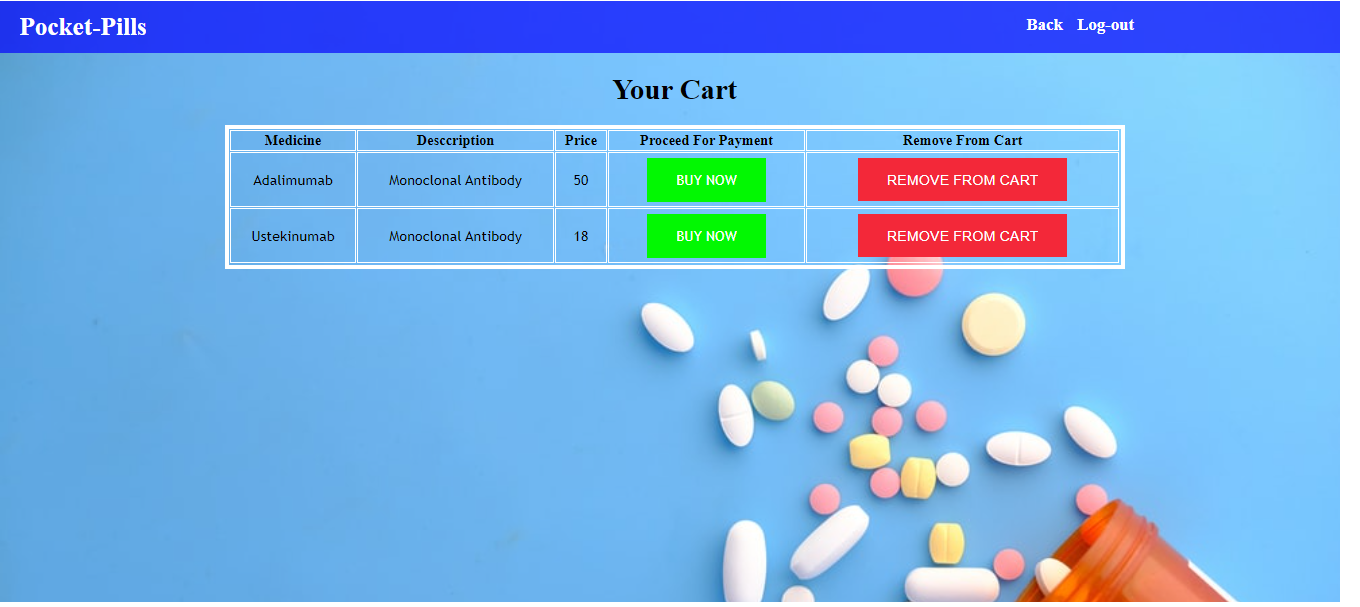


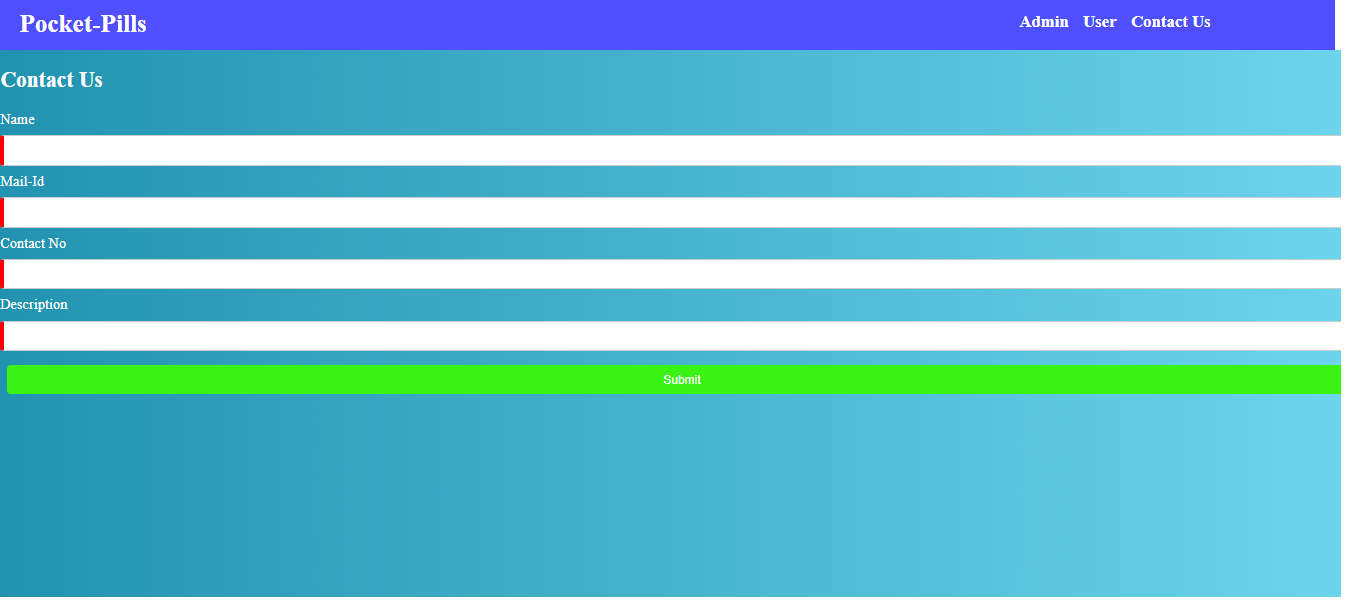


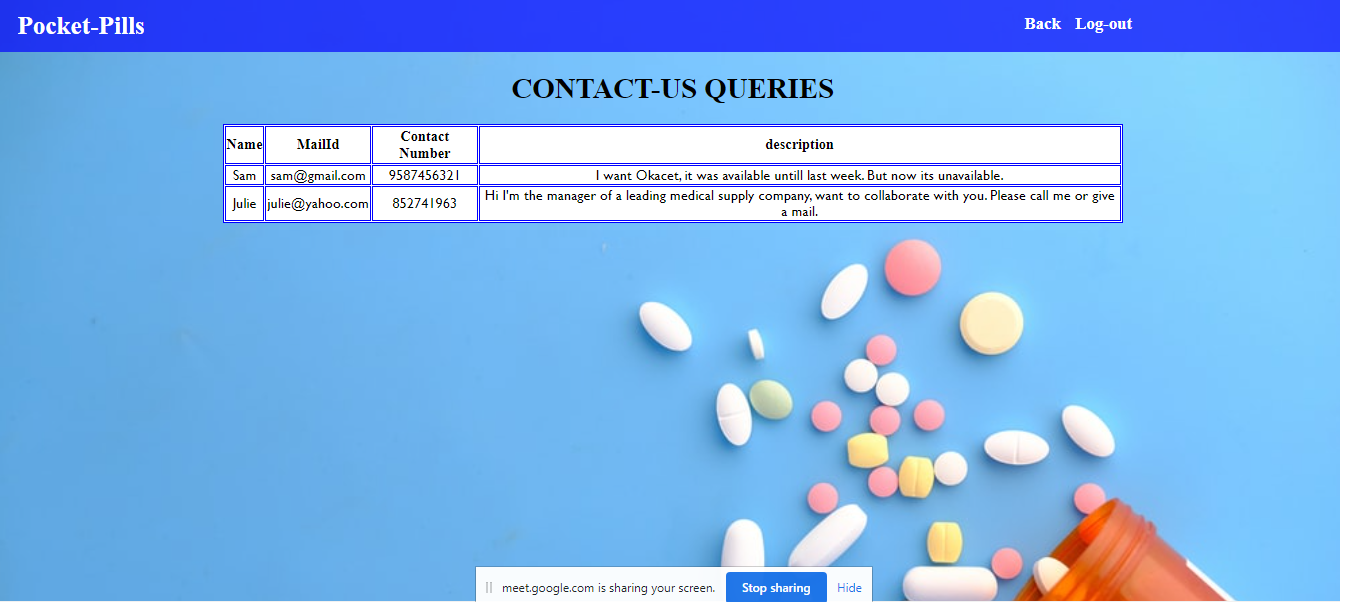










****

**11.CONCLUSION**

Our project is only a humble venture to satisfy the needs to manage their project work. Several user-friendly coding has also adopted. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

    At the end it is concluded that we have made effort on following points

* We understand the problem domain and produce a model of the system, which describes operations that can be performed on the system.
* We included features and operations in detail, including screen layouts.
* We designed user interface and security issues related to system.
* Finally, the system is implemented and tested according to test cases.