**Pharmacy Management System**

# A Project Report

Submitted in partial fulfillment of the

Requirements for the award of the Degree of

**BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)**

**By**

Rhythm Shah

UID / 20BIT053

**Under the esteemed guidance of**

**Prof Wilson Rao,**

**Co-ordinator**

**and**

**Prof Ms. Bertilla Fernandez,**

**Assistant Professor**



**DEPARTMENT OF INFORMATION TECHNOLOGY JAI HIND COLLEGE**

***(Autonomous)***

# MUMBAI, 400020

**MAHARASHTRA**

**2020-21**

1

**JAI HIND COLLEGE**

***(Autonomous)***

**MUMBAI, 400020 MAHARASHTRA**

# DEPARTMENT OF INFORMATION TECHNOLOGY



## CERTIFICATE

This is to certify that the project entitled, “Pharmacy Management System”, is bonafied work of RHYTHM SHAH bearing UID (20BIT053) submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from Jai Hind College Autonomous (University of Mumbai).

**Internal Guide Coordinator**

**External Examiner**

**Date: College Seal**

# DECLARATION

I hereby declare that the project entitled, “Pharmacy Management System” done at Jai Hind College, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other

university.

The project is done in partial fulfillment of the requirements for the award of

degree of BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY) to

be submitted as final semester project as part of our curriculum.

**Name and Signature of the Student**

# ACKNOWLEDGEMENT

I would like to express my thanks to the people who have helped me most throughout my project. I am grateful to my Prof. Miss. Bertilla Fernandes for nonstop support for the project. I can’t say thank you enough for her tremendous support and help.

I owe my deep gratitude to our HOD of Information Technology Department Mr. Wilson Rao who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

At last but not the least I want to thank all of my friends who helped/treasured me out in completing the project, where they all exchanged their own interesting ideas, thoughts and made this possible to complete my project with all accurate information. I wish to thank my parents for their personal support or attention who inspired/encouraged me to go my own way.

**ABSTRACT**

Increasing health concerns among people demands efficiency in Medical Stores. The need to migrate from originally paper-based to electronic notebooks with data storage, computational features and reliable electronic documentation has aroused. For this project, I have designed a Medical Store Management System based on Python and sqlite3 Database Access for small businesses. The software with an easy to use interface has been designed to allow stock maintenance, database access and electronic documentation for billing purposes with valued customer support. These features have been later used to compute discounts for customers, daily revenues, and determine steps to prevent potential revenue loss. Being an open source software it can be used by a wide variety of retail and wholesale outlets to automate the process of manually maintaining the records and cash flows. The results have been potentially generalizable. This software saves your time and money.

# TABLE OF CONTENTS

## 1. INTRODUCTION 8

1.1 **Background**………………………………………………………………………8

1.2 **Objectives**………………………………………………………………………..8

1.3 **Purpose, Scope, and Applicability**……………………………………………..9

1.3.1 Purpose…………………………………………………………………….9

1.3.2 Scope………………………………………………………………………9

1.3.3 Applicability………….……………………………………………………9

1.4 **Achievements**……………………………………………………………………9

1.5 **Organization of Report**…………………………………………………………10

1. **SURVEY OF TECHNOLOGIES 10**
2. **REQUIREMENT AND ANALYSIS 11**
   1. **Problem Definition**……………………………………………………………...11
   2. **Requirement Specification**……………………………………………………..11
   3. **Planning and Scheduling**……………………………………………………….12
   4. **Software and Hardware Requirements**……………………………………………………………………13
   5. **Preliminary Product Description**…………………………………………….......................................14
   6. **Conceptual Models**……………………………………………………………...14

## 4. SYSTEM DESIGN 24

4.1 **Basic Modules**……………………………………………………………...........24

4.2 **Data Design**……………………………………………………………...............24

4.2.1 Schema Design…………………………………………………………...24

4.2.2 Data Integrity and Constraints……………………………………………25

4.3 **User Interface Design**…………………………………………….......................26

4.4 **Security Issues**……………………………………………..................................28 4.5 **Test Cases Design**…………………………………………….............................30

## 5. IMPLEMENTATION AND TESTING 30

5.1 **Implementation Approach**……………………………………………………..30

5.2 **Coding Details and Code Efficiency**…………………………………………...31

5.2.1 Code Efficiency…………………………………………..........................54

5.3 **Testing Approach**………………………………………….................................54

5.3.1 Unit Testing…………………………………………................................54

5.3.2 Integration Testing………………………………………….....................54

5.3.3 System Testing…………………………………………...........................55

5.4 **Modifications and Improvements**……………………………………………...55

## 6. RESULTS AND DISCUSSION 55

6.1 **Test Reports**…………………………………………………………..55

6.2 **User Documentation**…………………………………………………...56

## 7. CONCLUSION 62

7.1 **Conclusion**………...………………………………………………….62

7.2 **Limitations of the System…..**………………….……………………….62

7.3 **Future Scope of the Project**…………...…………………………....….62

**List of Figures**

1. Gantt Chart……………………………………………………………...12
2. PERT Chart……………………………………………………………..13
3. Event Table……………………………………………………………...14
4. ER Diagram…………………………………………………………......15
5. Class Diagram…………………………………………………………..16
6. Use Case Diagram………………………………………………………16
7. Object Diagram…………………………………………………………17
8. Activity Diagram………………………………………………………..18
9. Sequence Diagram………………………………………………………19
10. State Chart Diagram……..……………………………………………...20
11. Package Diagram………………………………………………………..21
12. Component Diagram……………………………………………………22
13. Deployment Diagram…………………………………………………...22
14. Data Flow Level 0 Diagram…………………………………………….23
15. Data Flow Level 1 Diagram…………………………………………….23
16. Database Schema Design……………………………………………….24
17. Login Page………………………………………………………………26
18. Home Page………………………………………………………………27
19. Valued Customer...……………………………………………………...28
20. Search…………………………………………………………………...28
21. Login……………………………………………………………………56
22. Homepage…………………………………………………………….....57
23. Valued Customer………………………………………………………..57
24. Add ………….………………………………………………………….58
25. Delete……………………………………………………………….…..58
26. Modify ………….………………………………………………………59
27. Search…………………………………………………………………...59
28. Check Expired Medicines…………………………………………...….60
29. Billing…………………………………………………………………...60
30. Check Today’s Revenue………………………………………………..61

# 1 Introduction

## 1.1 Background

## The role of pharmacy management is to supervise and manage the pharmacy employees in

## order to preserve excellent working relationships and outcomes. To improve the services offered in every Pharmacy, the formulated proposal, abstract and modules of this Project management should be present in the PDF documentation.

## A great system for storing data and maintaining and organizing the use and process of

## medications in pharmacy should be the abstract content of this Management System Project. This must be a computer software system that have been programmed to perform the various tasks required in the operation of a pharmacy. This system will improve the efficiency of the company and enable for the storing of digital records.

## This Pharmacy management system database design was made based on managing

## pharmacy requirements.

## 1.2 Objectives

## The system can encode both customers and medicines’ information.

## Pharmacy admin can have access to the medicines status and information in terms of the number sold and remains to identify the inventories of sales and stocks. They can handle the data needed in managing pharmacy as well as the sales of medicines and customer records.

## 1) Easy for record keeping, tracking stocks, tracking revenue.

## 2) The database is regularly update and is updated online, which makes is easier for the manager

## or the owner to keep a record

## 3) Convenience, time saving and easy to use.

## 4) Increases sales for pharmacies as staff expense is lowered.

## 1.3 Purpose and Scope

### 1.3.1 Purpose

### This is an Open Source Software designed for medical store management purposes for small businesses. In this era of growing demands of Pharmaceutical and Chemical products for health and medicine; every Medical store, be it a small of medium sized businesses are running rampant towards efficient store management and rapid revenue computation. Instead of using the conventional style of paper-based records and documentation, electronic methods are preferred for stock maintenance, automatic updation of records while billing and reliable electronic documentation. The software would reduce daily effort wage and labour numbers. Crushing numbers can now be left to the system. The system computes and displays the daily revenue total and also notifies the products to expire the oncoming week. The system is based on Python Programming language for scripting and SQL Database Access for storage of records. The advantages of these features are explained in the next section.

### 1.3.2 Scope

### There is a very wide scope as any number of pharmacies can use this management system. It may help collecting perfect management in details, In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to Pharmacy Management System. It will be also reduced the cost of collecting the management &amp; collection procedure will go on smoothly.

### 1.3.3 Applicability

## The aim is to automate its existing manual system by the help of computerized equipments 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. Basically the project describes how. to manage for good performance and better services for the clients.

## 1.4 Achievements

The project objectives are met successfully as it eliminates the need for manual work in order to do calculations and generate billing and also maintaining stocks maually

## 1.5 Organization of Report

# In the chapters that follow, the requirement analysis, design, implementation, testing and discussions will be described. The survey of technologies will compare and decide on a certain technology suitable for the website. I have chosen to build the application using Visual Studio Code. The framework that I will be using is TKINTER and the languages that will be used are: Python. For the backend database, I will be using Sqlite3. The requirement analysis and specification phase with decide the software and hardware requirements. The designing phase will specify the schema design along with the UI design. The implementation of the project will include designing and programming the modules according to the mentioned specifications. The testing phase will include testing all the specifications and all modules thoroughly. Then the project can be fully functional and can meet all its objectives efficiently.

# 2 Survey of Technologies

In order to design maintenance software the basic requirements would be: a powerful Platform, reliable storage capability and a simple interface. All these have been accomplished by Python, a powerful cross-platform language that can run in any device with a Windows or GNU/Linux Interface. When coding an application, there is a requirement for dynamic content fulfilled by SQL. The mainly used technologies for the coding process of the software have been introduced below.

**Python Programming**

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Tkinter GUI & sqlite Database access has been immensly used to design this software.**

The software is designed to use this technology due to the above said reasons, to connect to the valued customers securely.

**Sqlite3**

SQLite is a C-language library that implements a [small](https://www.sqlite.org/footprint.html), [fast](https://www.sqlite.org/fasterthanfs.html), [self-contained](https://www.sqlite.org/selfcontained.html), [high-reliability](https://www.sqlite.org/hirely.html), [full-featured](https://www.sqlite.org/fullsql.html), SQL database engine. SQLite is the [most used](https://www.sqlite.org/mostdeployed.html) database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day

**3 Requirement and analysis**

## 3.1 Problem Definition

### As we know, nowadays people are opting for digital solutions for daily necessities because of busy schedule. So, in this busy world, people tend to save time. Thus, we need such systems which will provide good facilities to manage products.

### Existing System

### The existing system is limited by paper and files and other stationary items. Franchises can keep a record more easily with this proposed system

### Proposed System

In the proposed system, this platform will provide an easy way to manage stocks using the application. The staff can focus more on the customers and inventory due to the automation

provided on this platform. This system generates types of information than can be used for various purposes

## 3.2 Requirement Specification

### ● Functional Requirements

System needs store information about new entry of Pharmacy.

✓ System needs to help the internal staff to keep information of Medicines and find

✓ them as per various queries

✓ System need to maintain quantity record.

✓ System need to keep the record of Stocks

✓ System need to update and delete the record.

✓ System also needs a search area.

✓ It also needs a security system to prevent data.

### ● Non-Functional Requirements

Availability: This system will be easily available to authorized users

✓ Reliability: The system is highly reliable and secure

✓ Maintainability: The system is easily maintainable and provides an easy access of

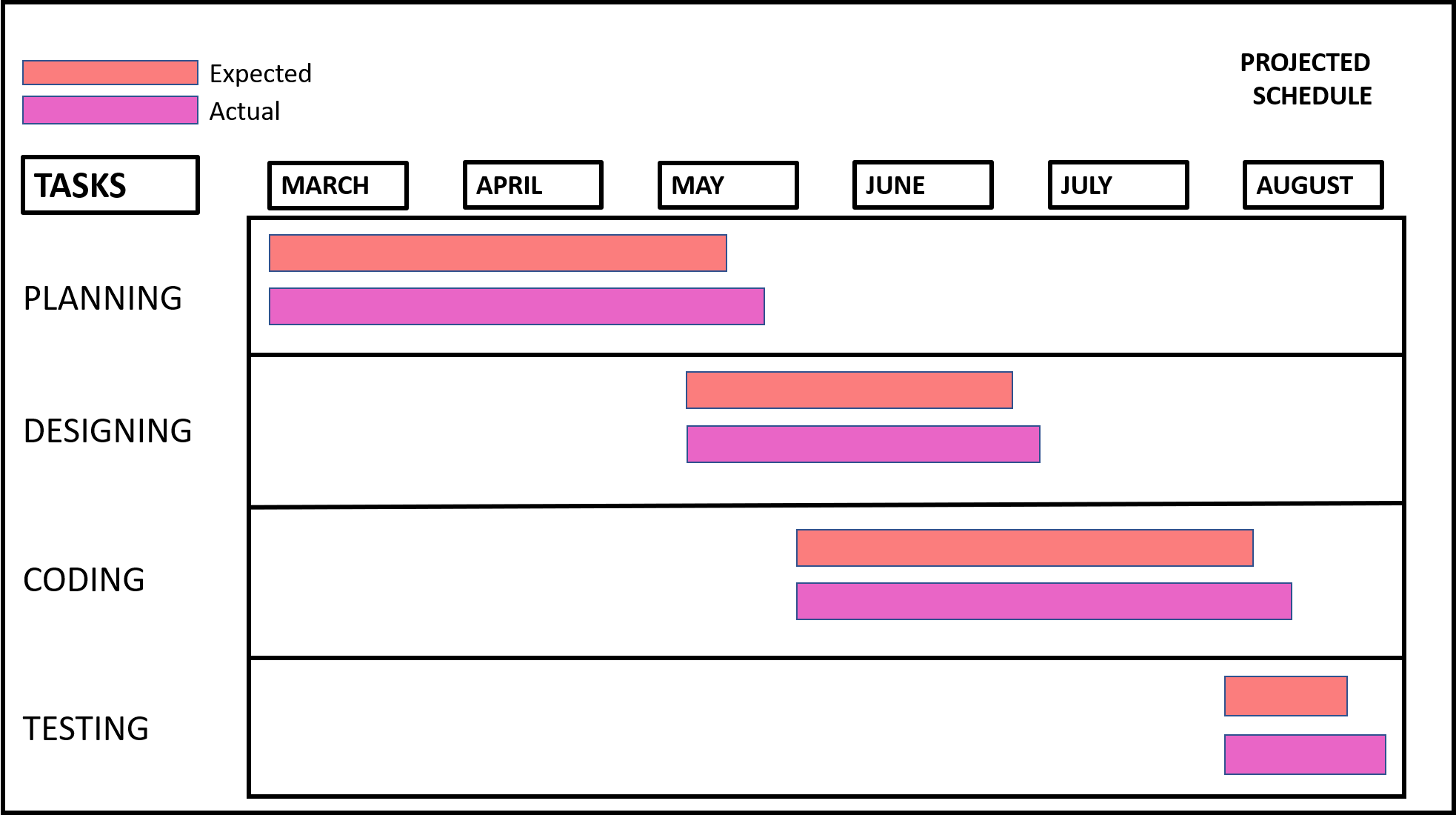
resources

✓ The application is user-friendly and simple.

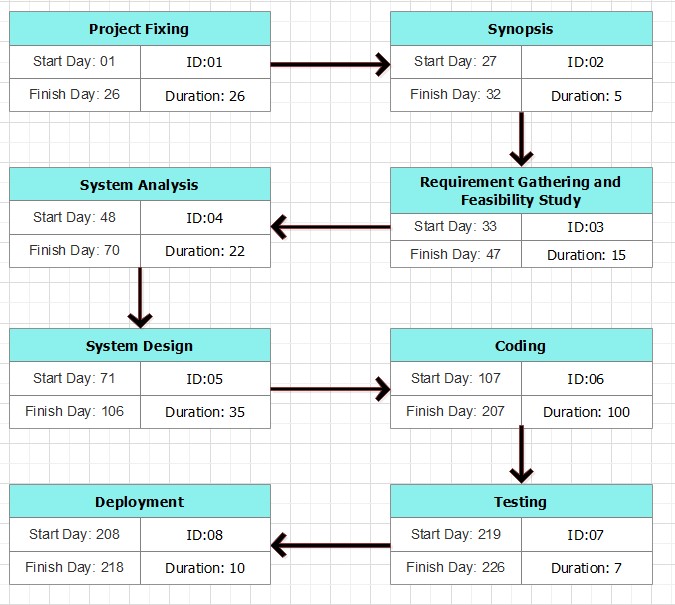
✓ Flexible i.e., changes can be made, if any.

✓ Efficiency in performance

## 3.3 Planning and Scheduling



**Figure 1:** GANTT Chart



**Figure 2**: PERT Chart

## 3.4 Software and Hardware Requirements

### ● Hardware

1. OS-Windows 7 or higher
2. Memory-4GB RAM

This application will need a windows based machine with a minimum 4 GB RAM with a working internet connection.

### ● Software

1. Visual Studio Code
2. Languages: Python. Sqlite3

## 3.5 Preliminary Product Description

The application will contain the following **modules:**

1. Log In

This module handles the login of the user. It validates the username and password and then allows the login

1. Homepage

The user once logged in has select from Stock maintenance, Access Database or Handle cash flows modules

1. Stock Maintainence

The user can select from New V.C, Add product, Delete product

1. Access Database

The user can select from Modify, search, expiry check

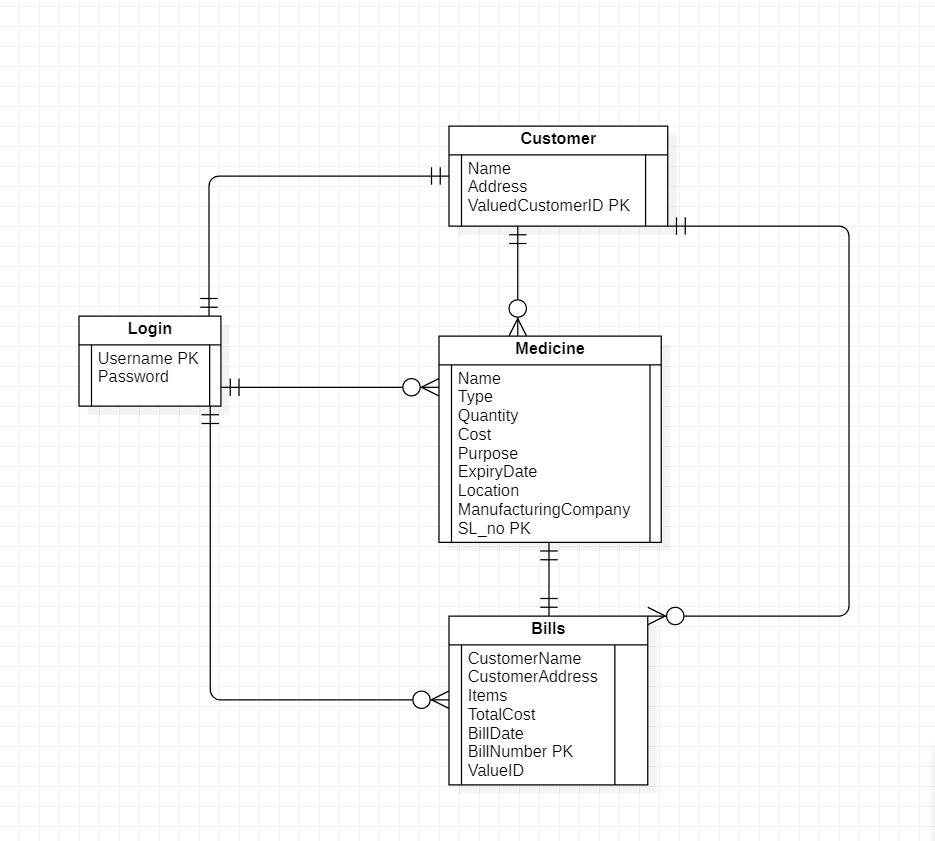
1. Handle cash flows

The user can select from Billing, Check Today’s revenue, Logout

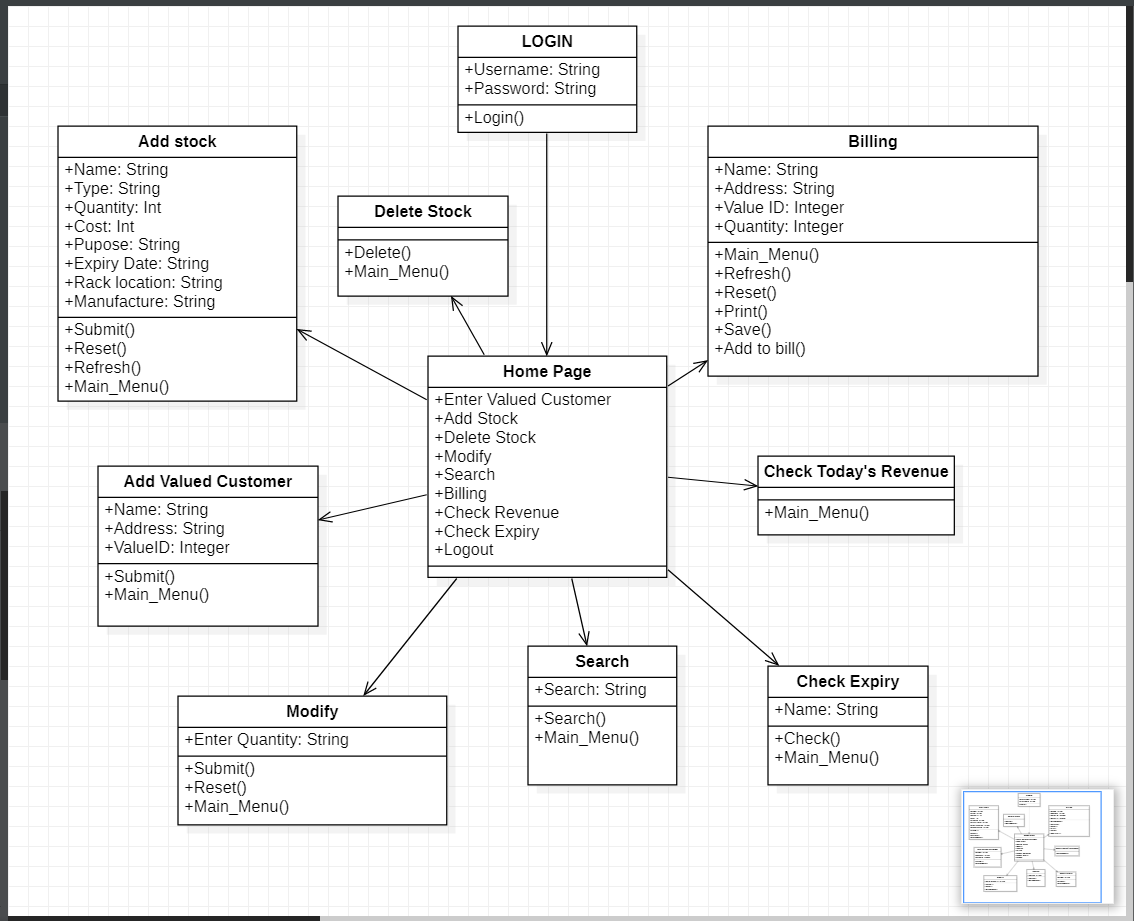
## 3.6 Conceptual Models

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr. No | Event | Trigger | Source | Use Case | Response | Destination |
| 1 | User login | Login Request | User | Validate Details | Logged in successful | User |
| 2 | Home page | Stock maintenance, Access Database, Handle cash flows | User | Select any option | Redirect to that page | User |
| 3 | Stock maintenance | New V.C, Add product, Delete product | User | Select any option | Redirect to that page | User |
| 4 | Access Database | Modify, search, expiry check | User | Select any option | Redirect to that page | User |
| 5 | Handle Cash Flows | Billing, Check Today’s revenue, Logout | User | Select any option | Redirect to that page | User |

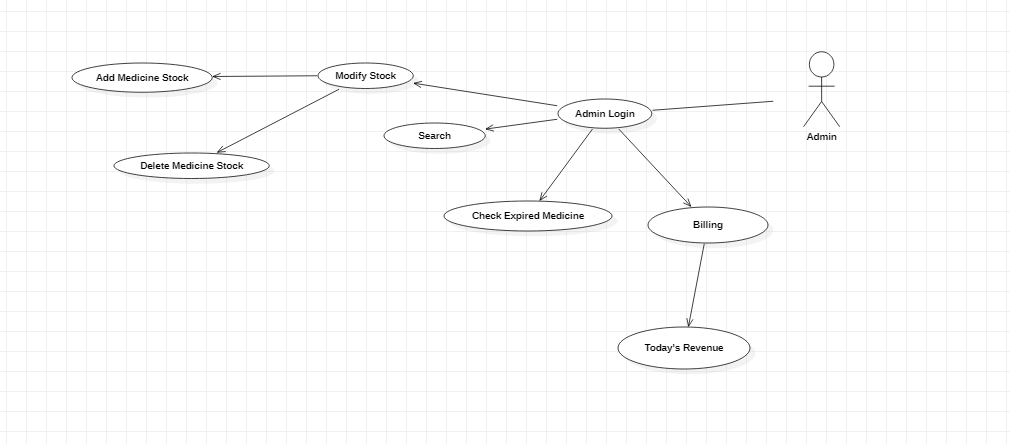
**Figure 3:** Event Table



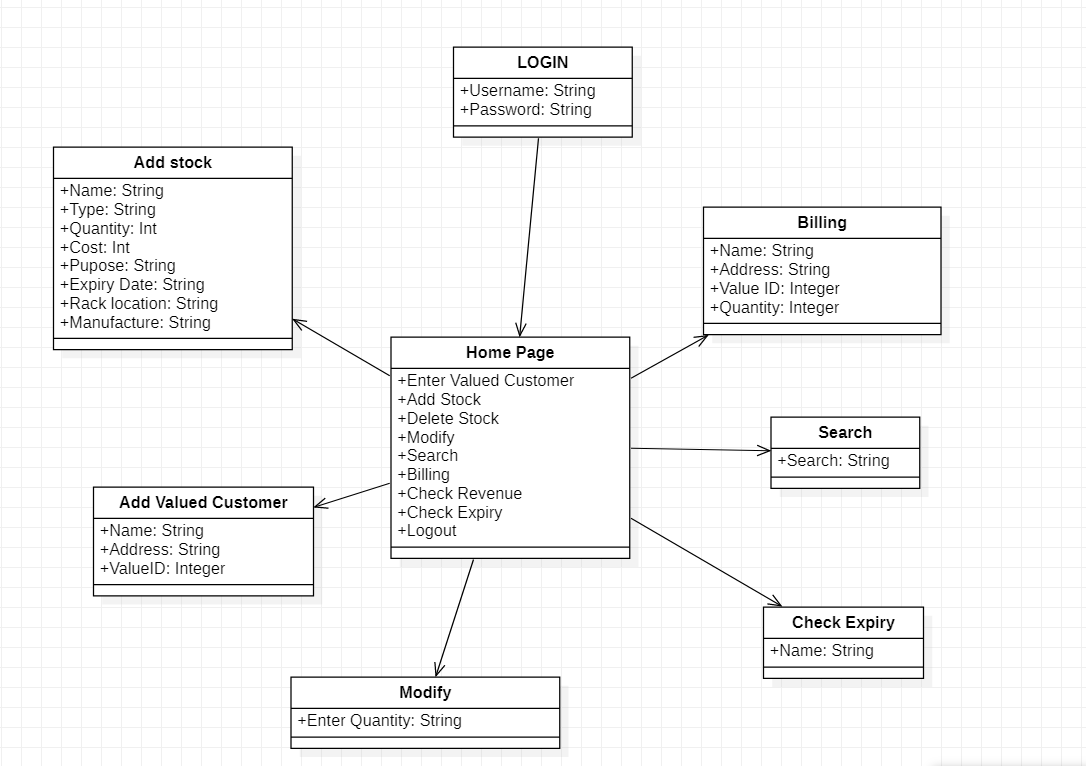
**Figure 4:** E-R Diagram



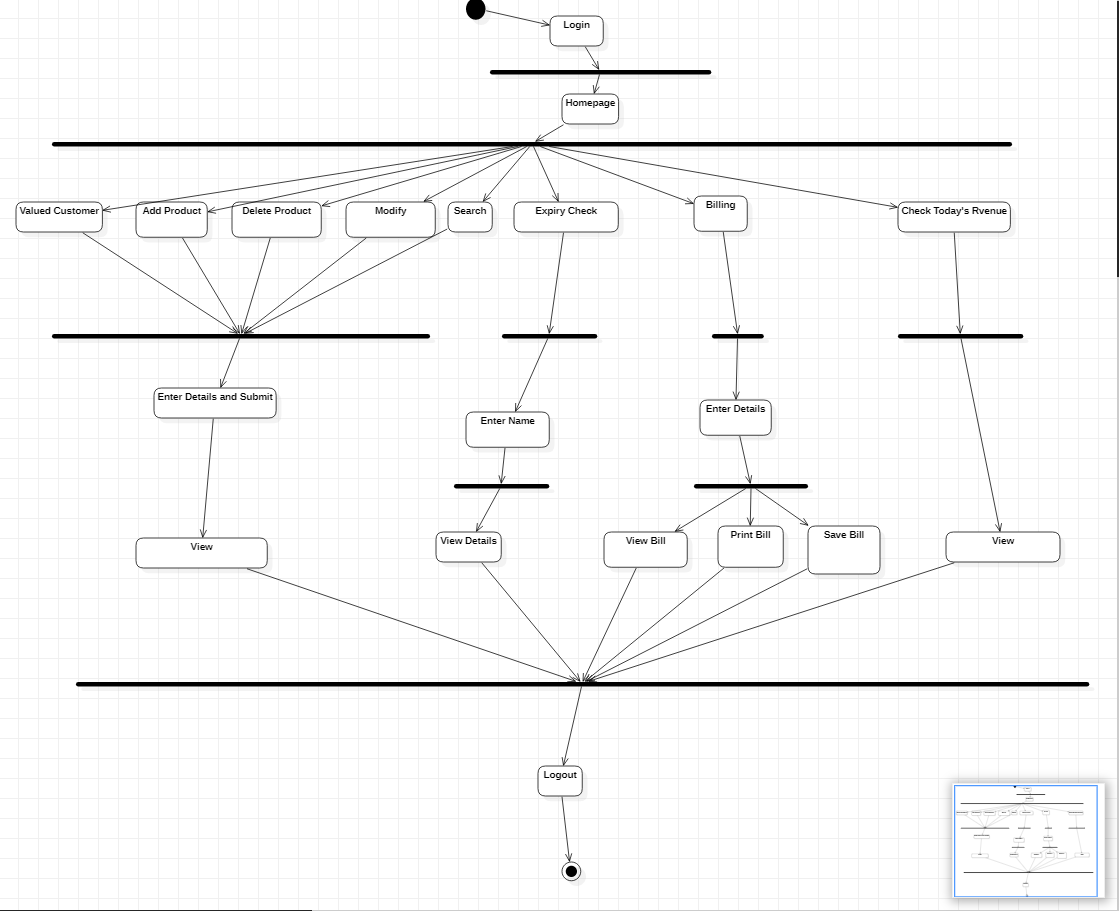
**Figure 5:** Class Diagram



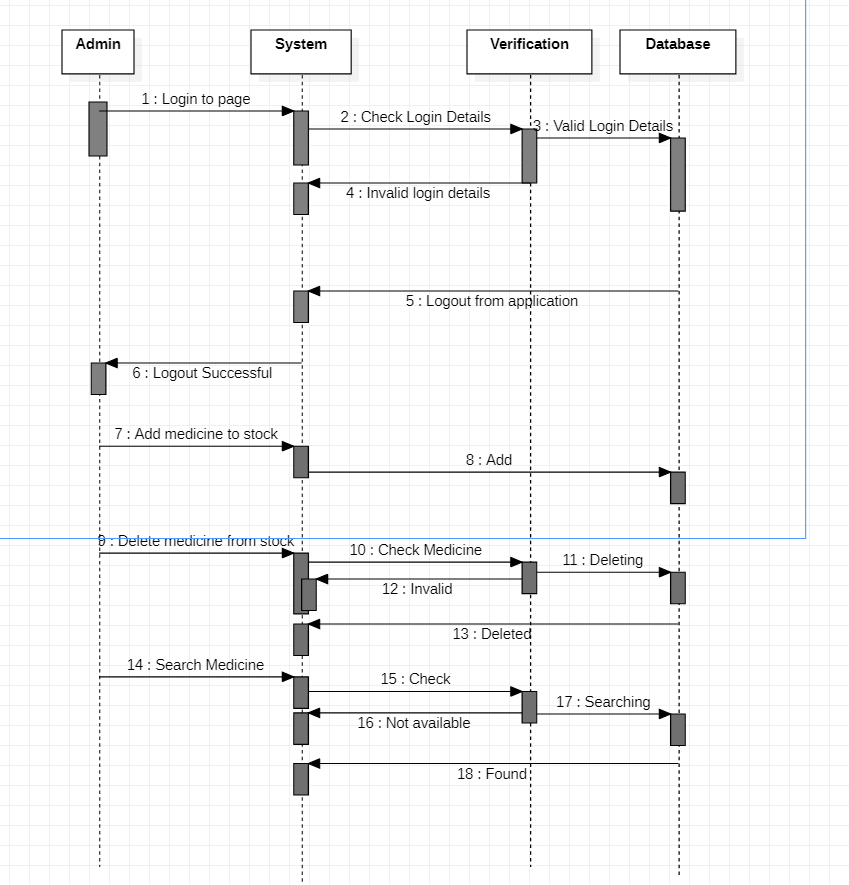
**Figure 6:** Use Case Diagram



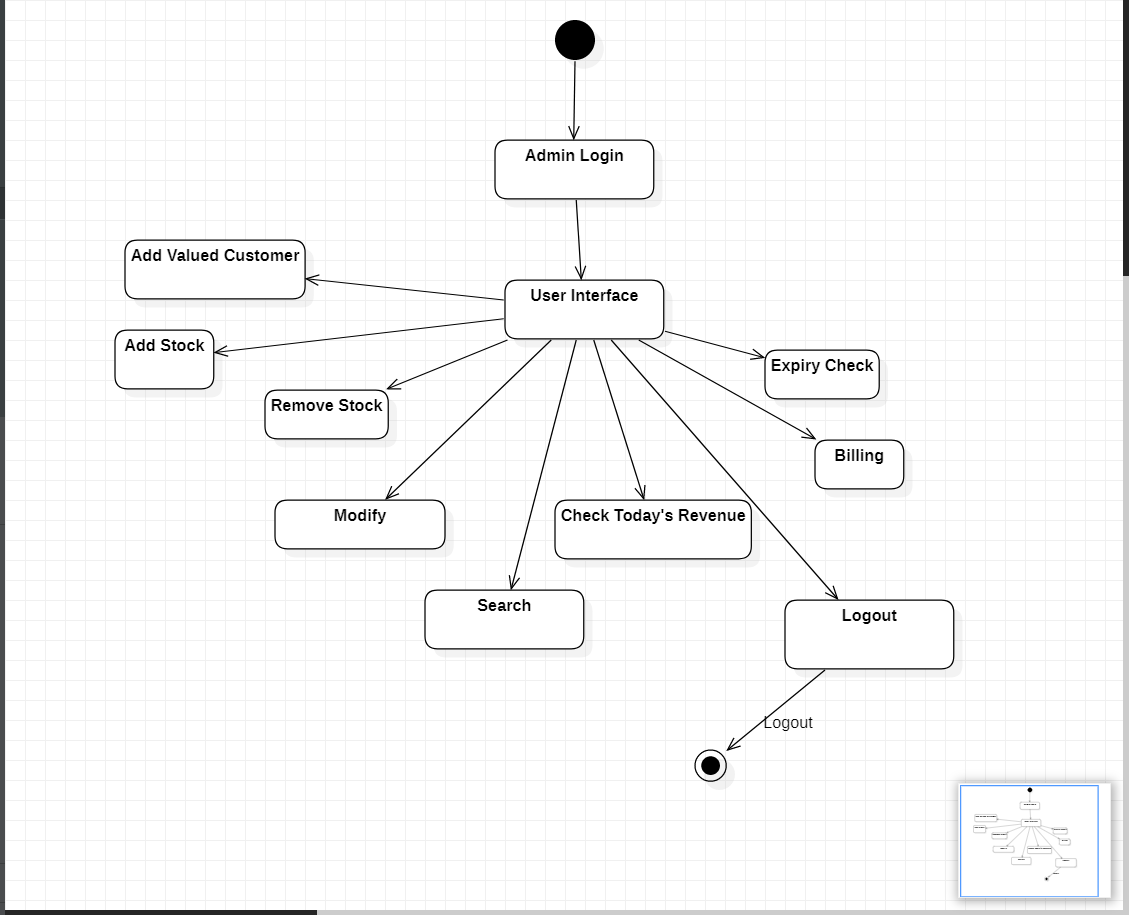
**Figure 7:** Object Diagram



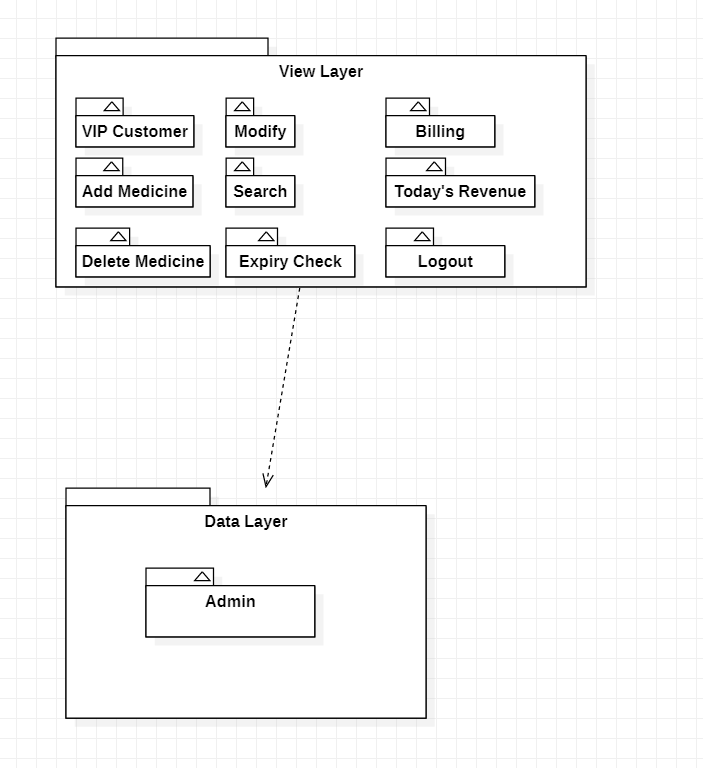
**Figure 8:** Activity Diagram



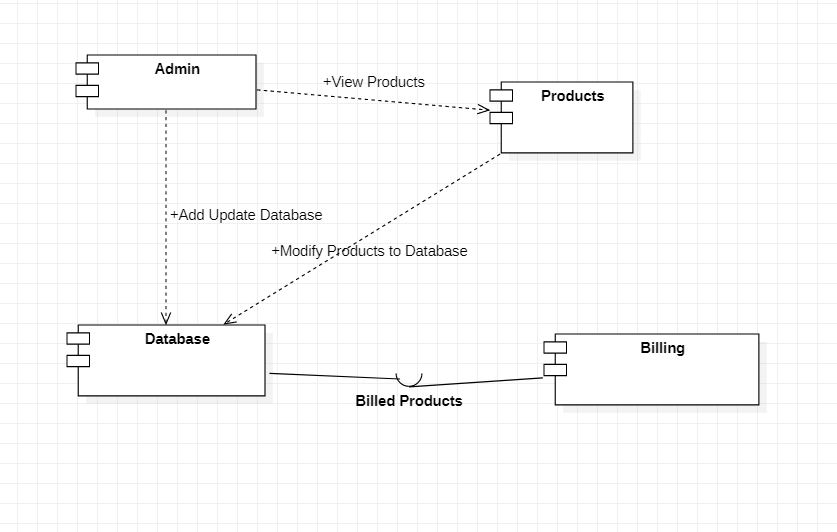
**Figure 9:** Sequence Diagram



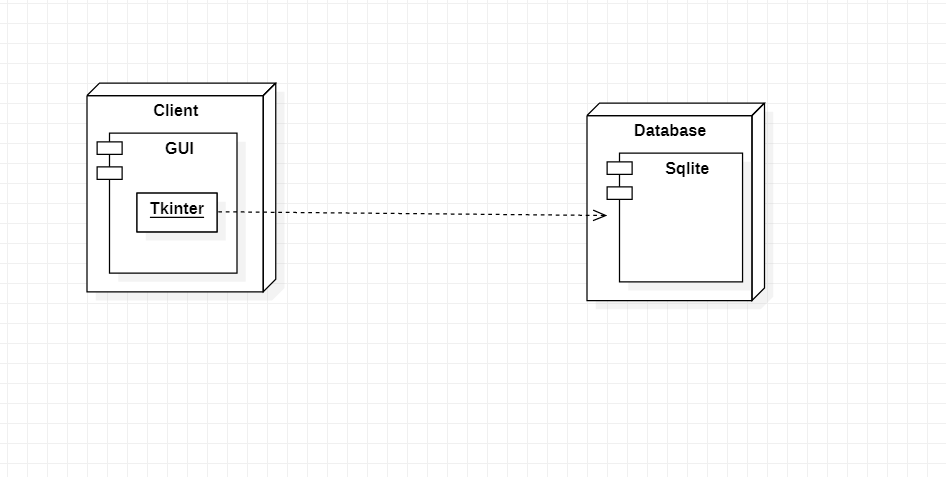
**Figure 10:** State Chart Diagram



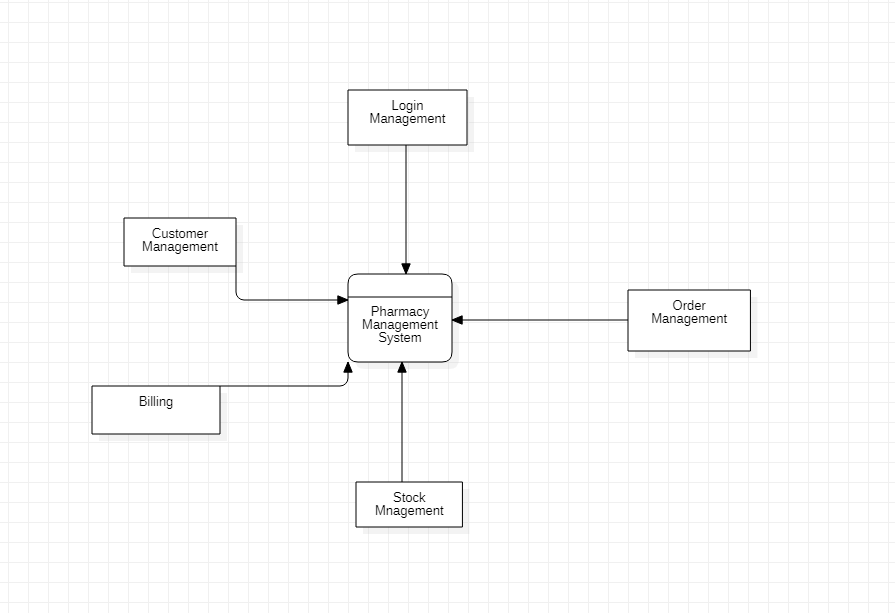
**Figure 11:** Package Diagram



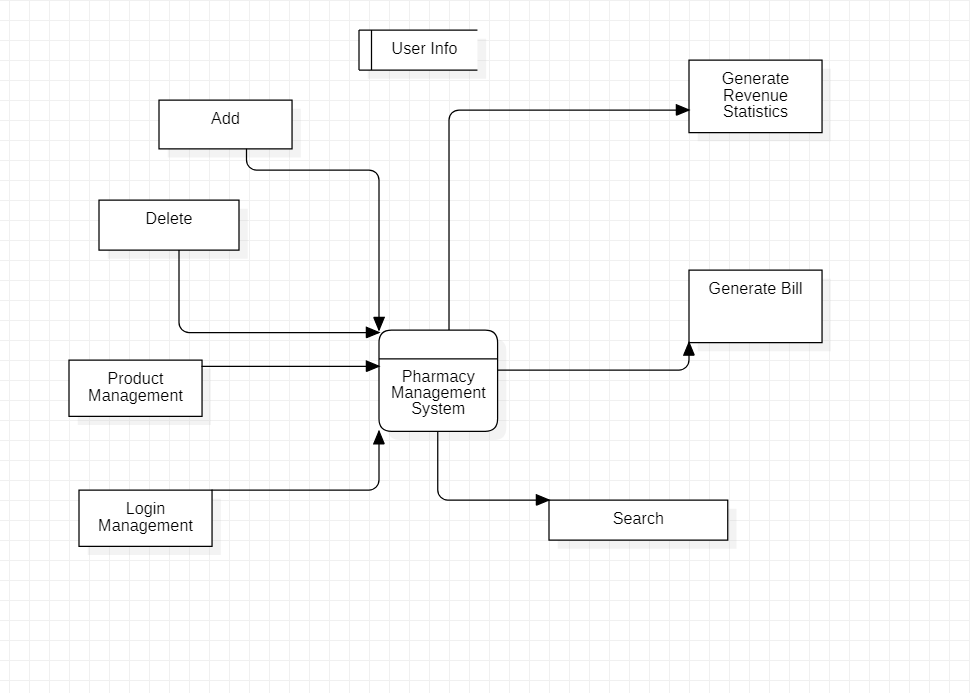
**Figure 12:** Component Diagram



**Figure 13:** Deployment Diagram



**Figure 14:** Data Flow Diagram (Context Level)



**Figure 15:** Data Flow Diagram (Level 1)

# 4. SYSTEM DESIGN

## 4.1 Basic Modules

* Log In

This module handles the login of the user.

* Homepage

The homepage contains the Stock maintainence, access database and handle cash flows options

* Stock Maintainence

The user can select from New V.C, Add product, Delete product

* Access Database

The user can select from Modify, search, expiry check

* Handle cash flows

The user can select from Billing, Check Today’s revenue, Logout

## 4.2 Data Design

**4.2.1 Schema Design**

# Bills

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Data Type | Null | Constraints |
| Cus\_name | **Varchar(30)** | **Yes** |  |
| cus\_add | **Varchar(30)** | **Yes** |  |
| items | **Varchar (30)** | **Yes** |  |
| Total\_cost | **Integer (10)** | **Yes** |  |
| bill\_dt | **Varchar (30)** | **Yes** |  |
| bill\_no | **Integer (10)** | **Yes** | **Primary Key** |
| bill | **Varchar(30)** | **Yes** |  |
| val\_id | **Integer(10)** | **Yes** |  |

# Cus

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Data Type | Null | Constraints |
| name\_vc | **Varchar(30)** | **No** |  |
| add\_vc | **Varchar(30)** | **No** |  |
| vc\_id | **Integer (10)** | **No** | **Primary Key,Unique** |

# med

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Data Type | Null | Constraints |
| sl\_no | **Int** | **No** | **Primary Key,Unique** |
| name | **Varchar(30)** | **No** |  |
| type | **Varchar(30)** | **Yes** |  |
| qty\_left | **Integer (10)** | **Yes** |  |
| cost | **Integer (10)** | **Yes** |  |
| purpose | **Varchar(30)** | **Yes** |  |
| expdt | **Date** | **Yes** |  |
| loc | **Varchar(30)** | **Yes** |  |
| mfg | **Varchar(30)** | **Yes** |  |

# log

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Data Type | Null | Constraints |
| username | **Varchar(30)** | **No** | **Primary Key,Unique** |
| password | **Varchar(30)** | **No** |  |

**Figure 16:** Database and Schema Design

## 4.2.2 Data Integrity and Constraints

* **Auto Increment Constraint**

This constraint is required to check.

* **Primary Key Constraint**

Uniquely identifies each row/record in a database table

* **Foreign key Constraint**

Uniquely identifies each row/record in a database table

* **Check Constraint**

Ensures that all values in a column satisfy certain conditions.

* **Not NULL Constraint**

It is Required Because, User’s Username and Password, can’t be Null, Because that’s

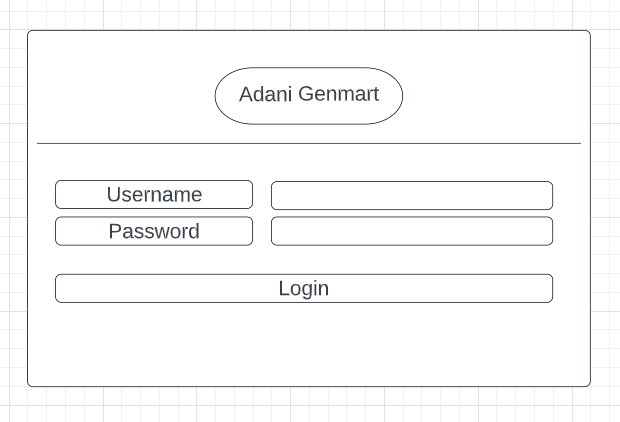
the only way they have to Authenticate Himself/Her- self.

* **Uniqueness Constraint**

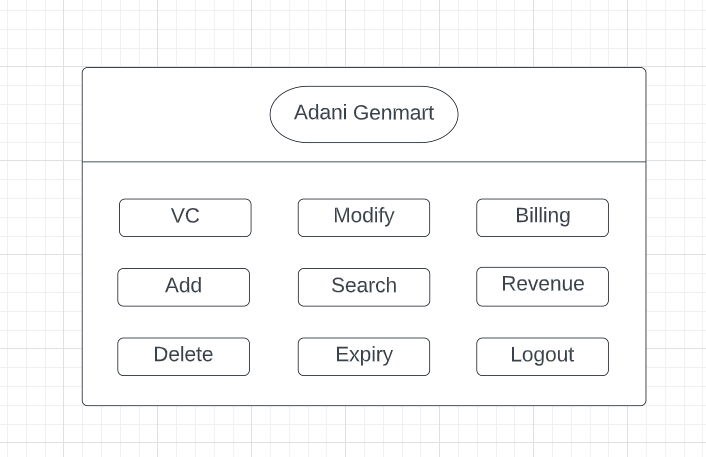
It is Required Because, the Username Columns shouldn’t contain Repeated Value.

Every User will have unique Username to identify or authenticate himself/herself.

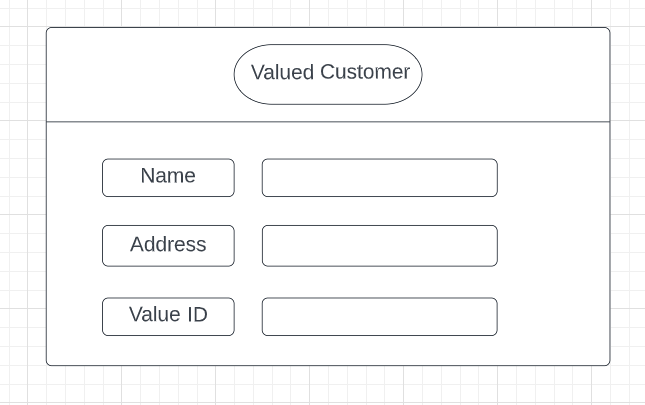
## 4.3 User Interface Designs



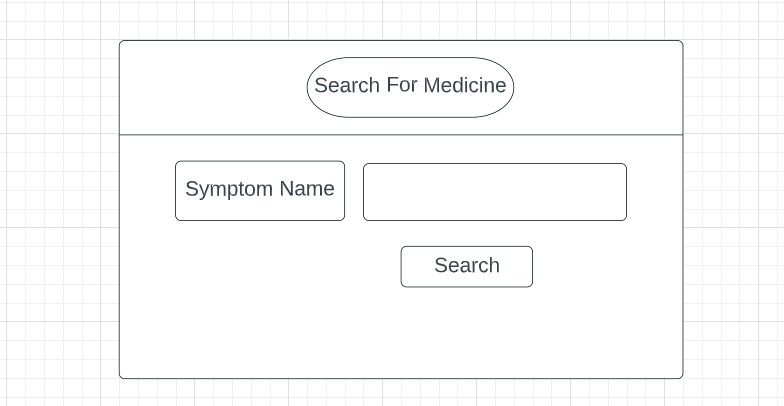
**Figure 17:** Login



**Figure 18:** Homepage



**Figure 19:** Valued Customer



**Figure 20:** Search Module

### 4.4 Security Issues

* **Authorization**

System is authorized to give permission to register users to the system and system has to check whether the user is a normal user or a hacker because he/she can use our data or

information and misuse it or sell to other third parties.

* **Authentication**

We are taking this approach because every user has to authenticate himself/herself to

operate our system. Every user will have unique username or UserID.

* **DOS attack prevention**

Web sites are the most attractive target for hackers because even an unsophisticated hacker can bring down a server by repeatedly calling a web portals.

**SQL Injection attack prevention**

SQL Injection flaws are introduced when software developers create dynamic database queries that include user supplied input. In order to prevent these from happening we are validating the user input properly and using parameterized SQL queries with stored

procedures.

* **Password encryption**

Network eavesdropping and many other attacks can be prevented when storing the

password in an encrypted format in the database.

* **SSL Encryption**

The HTTPS protocol is more secure than HTTP protocol because it includes the Secure Sockets Layer/Transport Layer Security (SSL/TLS) protocol. It is a more secure way to send a request to the server from a client, also the communication is purely encrypted which means no one can understand what you are looking for. For SSL connection a web

### 4.5 Test Case Design

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Condition** | **Input Selection** | **Expected** | **Actual** |
| **User/Admin Login** | **Username=” “**  **Password=” “**  **Adminmail=” “** | **Username and password fields are empty** | **Username and password fields are empty** |
| **Username=”abcd”**  **Password=”123!@#”** | **Verify and redirect** | **Verify and redirect** |

# 5 IMPLEMENTATION AND TESTING

## 5.1 Implementation Approach

* **Introduction**

The website starts with the Login page. After this the homepage will have all the options displayed. The user can choose from the following main dashboards: Handle Cash Flows, Stock Maintenance, Access Database

**Input and Output Design Implementation**

This project was created by first deciding on the number of modules to be made, along with the functionalities provided to the admin and the user specifically. The database design is of major importance to this project since all three major modules are storing the backend information in a database.

* **Code Module**

This Activity demonstrates the technologies used for the Front-End and Back-End and which Database. I am using the Tkinter framework for my front end which will be based on Python. The database used is Sqlite3

* **System Implementation**

The user is first directed to the login page where the user will be asked to login. After this the homepage will have all the options displayed. The user can choose from the following main dashboards: Handle Cash Flows, Stock Maintenance, Access Database

**Project Summary**

After logging into the system the user is presented with the Homepage of the system which can redirect the user to the following main pages: Valued Customer, Add , Delete, Modify, Search, Expiry Check, Billing, Revenue, Logout.

## 5.2 Code Details and Code Efficiency

**Main.py**

|  |
| --- |
| *import* time  *from* tkinter *import* \*  *import* sqlite3  *import* random  *import* tempfile  f = ''  flag = ''  flags = ''  login = sqlite3.connect("admin.db")  l = login.cursor()  c = sqlite3.connect("medicine.db")  cur = c.cursor()  columns = ('Sl No', 'Name', 'Type', 'Quantity Left', 'Cost', 'Purpose', 'Expiry Date', 'Rack location', 'Manufacture')  def open\_win():      global apt, flag      flag = 'apt'      apt = Tk()      apt.title("Interface")      Label(apt, *text*="ADANI GENMART").grid(*row*=0, *column*=0, *columnspan*=3)      Label(apt, *text*='\*' \* 80).grid(*row*=1, *column*=0, *columnspan*=3)      Label(apt, *text*='-' \* 80).grid(*row*=3, *column*=0, *columnspan*=3)      Label(apt, *text*="Stock Maintenance", *bg*='green', *fg*='white').grid(*row*=2, *column*=0)      Button(apt, *text*='New V.C.', *width*=25, *bg*='green', *fg*='white', *command*=val\_cus).grid(*row*=4, *column*=0)      Button(apt, *text*='Add product to Stock', *bg*='green', *fg*='white', *width*=25, *command*=stock).grid(*row*=5, *column*=0)      Button(apt, *text*='Delete product from Stock', *bg*='red', *fg*='white', *width*=25, *command*=delete\_stock).grid(*row*=6,*column*=0)      Label(apt, *text*="Access Database", *bg*='blue', *fg*='white').grid(*row*=2, *column*=1)      Button(apt, *text*='Modify', *width*=15, *bg*='blue', *fg*='white', *command*=modify).grid(*row*=4, *column*=1)      Button(apt, *text*='Search', *width*=15, *bg*='blue', *fg*='white', *command*=search).grid(*row*=5, *column*=1)      Button(apt, *text*='Expiry Check', *bg*='red', *fg*='white', *width*=15, *command*=exp\_date).grid(*row*=6, *column*=1)      Label(apt, *text*="Handle Cash Flows", *bg*='skyblue', *fg*='black').grid(*row*=2, *column*=2)      Button(apt, *text*="Check Today's Revenue", *bg*='skyblue', *fg*='black', *width*=20, *command*=show\_rev).grid(*row*=5,*column*=2)      Button(apt, *text*='Billing', *width*=20, *bg*='skyblue', *fg*='black', *command*=billing).grid(*row*=4, *column*=2)      Button(apt, *text*='Logout', *bg*='red', *fg*='white', *width*=20, *command*=again).grid(*row*=6, *column*=2)      apt.mainloop()  def delete\_stock():      global cur, c, flag, lb1, d      apt.destroy()      flag = 'd'      d = Tk()      d.title("Delete a product from Stock")      Label(d, *text*='Enter Product to delete:').grid(*row*=0, *column*=0)      Label(d, *text*='', *width*=30, *bg*='white').grid(*row*=0, *column*=1)      Label(d, *text*='Product').grid(*row*=2, *column*=0)      Label(d, *text*='Qty.  Exp.dt.     Cost                           ').grid(*row*=2, *column*=1)      ren()      b = Button(d, *width*=20, *text*='Delete', *bg*='red', *fg*='white', *command*=delt).grid(*row*=0, *column*=3)      b = Button(d, *width*=20, *text*='Main Menu', *bg*='green', *fg*='white', *command*=main\_menu).grid(*row*=5, *column*=3)      d.mainloop()  def ren():      global lb1, d, cur, c      def onvsb(\**args*):          lb1.yview(\*args)          lb2.yview(\*args)      def onmousewheel():          lb1.ywiew = ('scroll', events.delta, 'units')          lb2.ywiew = ('scroll', events.delta, 'units')  *return* 'break'      cx = 0      vsb = Scrollbar(*orient*='vertical', *command*=onvsb)      lb1 = Listbox(d, *width*=25, *yscrollcommand*=vsb.set)      lb2 = Listbox(d, *width*=30, *yscrollcommand*=vsb.set)      vsb.grid(*row*=3, *column*=2, *sticky*=N + S)      lb1.grid(*row*=3, *column*=0)      lb2.grid(*row*=3, *column*=1)      lb1.bind('<MouseWheel>', onmousewheel)      lb2.bind('<MouseWheel>', onmousewheel)      cur.execute("select \*from med")  *for* i *in* cur:          cx += 1          s1 = [str(i[0]), str(i[1])]          s2 = [str(i[3]), str(i[6]), str(i[4])]          lb1.insert(cx, '. '.join(s1))          lb2.insert(cx, '   '.join(s2))      c.commit()      lb1.bind('<<ListboxSelect>>', sel\_del)  def sel\_del(*e*):      global lb1, d, cur, c, p, sl2      p = lb1.curselection()      print(p)      x = 0      sl2 = ''      cur.execute("select \* from med")  *for* i *in* cur:          print(x, p[0])  *if* x == int(p[0]):              sl2 = i[0]  *break*          x += 1      c.commit()      print(sl2)      Label(d, *text*=' ', *bg*='white', *width*=20).grid(*row*=0, *column*=1)      cur.execute('Select \* from med')  *for* i *in* cur:  *if* i[0] == sl2:              Label(d, *text*=i[0] + '. ' + i[1], *bg*='white').grid(*row*=0, *column*=1)      c.commit()  def delt():      global p, c, cur, d      cur.execute("delete from med where sl\_no=?", (sl2,))      c.commit()      ren()  def modify():      global cur, c, accept, flag, att, up, n, name\_, apt, st, col, col\_n      col = ('', '', 'type', 'qty\_left', 'cost', 'purpose', 'expdt', 'loc', 'mfg')      col\_n = ('', '', 'Type', 'Quantity Left', 'Cost', 'Purpose', 'Expiry Date', 'Rack location', 'Manufacture')      flag = 'st'      name\_ = ''      apt.destroy()      n = []      cur.execute("select \* from med")  *for* i *in* cur:          n.append(i[1])      c.commit()      st = Tk()      st.title('MODIFY')      Label(st, *text*='-' \* 48 + ' MODIFY DATABASE ' + '-' \* 48).grid(*row*=0, *column*=0, *columnspan*=6)      def onvsb(\**args*):          name\_.yview(\*args)      def onmousewheel():          name\_.ywiew = ('scroll', event.delta, 'units')  *return* 'break'      cx = 0      vsb = Scrollbar(*orient*='vertical', *command*=onvsb)      vsb.grid(*row*=1, *column*=3, *sticky*=N + S)      name\_ = Listbox(st, *width*=43, *yscrollcommand*=vsb.set)      cur.execute("select \*from med")  *for* i *in* cur:          cx += 1          name\_.insert(cx, (str(i[0]) + '.  ' + str(i[1])))          name\_.grid(*row*=1, *column*=1, *columnspan*=2)      c.commit()      name\_.bind('<MouseWheel>', onmousewheel)      name\_.bind('<<ListboxSelect>>', sel\_mn)      Label(st, *text*='Enter Medicine Name: ').grid(*row*=1, *column*=0)      Label(st, *text*='Enter changed Value of: ').grid(*row*=2, *column*=0)      att = Spinbox(st, *values*=col\_n)      att.grid(*row*=2, *column*=1)      up = Entry(st)      up.grid(*row*=2, *column*=2)      Button(st, *width*=10, *text*='Submit', *bg*='green', *fg*='white', *command*=save\_mod).grid(*row*=2, *column*=4)      Button(st, *width*=10, *text*='Reset', *bg*='red', *fg*='white', *command*=res).grid(*row*=2, *column*=5)      Button(st, *width*=10, *text*='Show data', *bg*='blue', *fg*='white', *command*=show\_val).grid(*row*=1, *column*=4)      Label(st, *text*='-' \* 120).grid(*row*=3, *column*=0, *columnspan*=6)      Button(st, *width*=10, *text*='Main Menu', *bg*='green', *fg*='white', *command*=main\_menu).grid(*row*=5, *column*=5)      st.mainloop()  def res():      global st, up      up = Entry(st)      up.grid(*row*=2, *column*=2)      Label(st, *width*=20, *text*='                         ').grid(*row*=5, *column*=1)  def sel\_mn(*e*):      global n, name\_, name\_mn, sl, c, cur      name\_mn = ''      p = name\_.curselection()      print(p)      x = 0      sl = ''      cur.execute("select \* from med")  *for* i *in* cur:          print(x, p[0])  *if* x == int(p[0]):              sl = i[0]  *break*          x += 1      c.commit()      print(sl)      name\_nm = n[int(sl)]      print(name\_nm)  def show\_val():      global st, name\_mn, att, cur, c, col, col\_n, sl  *for* i *in* range(3):          Label(st, *width*=20, *text*='                         ').grid(*row*=5, *column*=i)      cur.execute("select \* from med")  *for* i *in* cur:  *for* j *in* range(9):  *if* att.get() == col\_n[j] and sl == i[0]:                  Label(st, *text*=str(i[0])).grid(*row*=5, *column*=0)                  Label(st, *text*=str(i[1])).grid(*row*=5, *column*=1)                  Label(st, *text*=str(i[j])).grid(*row*=5, *column*=2)      c.commit()  def save\_mod():  *# save modified data*      global cur, c, att, name\_mn, st, up, col\_n, sl  *for* i *in* range(9):  *if* att.get() == col\_n[i]:              a = col[i]      sql = "update med set '%s' = '%s' where sl\_no = '%s'" % (a, up.get(), sl)      cur.execute(sql)      c.commit()      Label(st, *text*='Updated!').grid(*row*=5, *column*=4)  def stock():      global cur, c, columns, accept, flag, sto, apt      apt.destroy()      flag = 'sto'      accept = [''] \* 10      sto = Tk()      sto.title('STOCK ENTRY')      Label(sto, *text*='ENTER NEW PRODUCT DATA TO THE STOCK').grid(*row*=0, *column*=0, *columnspan*=2)      Label(sto, *text*='-' \* 50).grid(*row*=1, *column*=0, *columnspan*=2)  *for* i *in* range(1, len(columns)):          Label(sto, *width*=15, *text*=' ' \* (14 - len(str(columns[i]))) + str(columns[i]) + ':').grid(*row*=i + 2, *column*=0)          accept[i] = Entry(sto)          accept[i].grid(*row*=i + 2, *column*=1)      Button(sto, *width*=15, *text*='Submit', *bg*='blue', *fg*='white', *command*=submit).grid(*row*=12, *column*=1)      Label(sto, *text*='-' \* 165).grid(*row*=13, *column*=0, *columnspan*=7)      Button(sto, *width*=15, *text*='Reset', *bg*='red', *fg*='white', *command*=reset).grid(*row*=12, *column*=0)      Button(sto, *width*=15, *text*='Refresh stock', *bg*='skyblue', *fg*='black', *command*=ref).grid(*row*=12, *column*=4)  *for* i *in* range(1, 6):          Label(sto, *text*=columns[i]).grid(*row*=14, *column*=i - 1)      Label(sto, *text*='Exp           Rack   Manufacturer                      ').grid(*row*=14, *column*=5)      Button(sto, *width*=10, *text*='Main Menu', *bg*='green', *fg*='white', *command*=main\_menu).grid(*row*=12, *column*=5)      ref()      sto.mainloop()  def ref():      global sto, c, cur      def onvsb(\**args*):          lb1.yview(\*args)          lb2.yview(\*args)          lb3.yview(\*args)          lb4.yview(\*args)          lb5.yview(\*args)          lb6.yview(\*args)      def onmousewheel():          lb1.ywiew = ('scroll', events.delta, 'units')          lb2.ywiew = ('scroll', events.delta, 'units')          lb3.ywiew = ('scroll', events.delta, 'units')          lb4.ywiew = ('scroll', events.delta, 'units')          lb5.ywiew = ('scroll', events.delta, 'units')          lb6.ywiew = ('scroll', events.delta, 'units')  *return* 'break'      cx = 0      vsb = Scrollbar(*orient*='vertical', *command*=onvsb)      lb1 = Listbox(sto, *yscrollcommand*=vsb.set)      lb2 = Listbox(sto, *yscrollcommand*=vsb.set)      lb3 = Listbox(sto, *yscrollcommand*=vsb.set, *width*=10)      lb4 = Listbox(sto, *yscrollcommand*=vsb.set, *width*=7)      lb5 = Listbox(sto, *yscrollcommand*=vsb.set, *width*=25)      lb6 = Listbox(sto, *yscrollcommand*=vsb.set, *width*=37)      vsb.grid(*row*=15, *column*=6, *sticky*=N + S)      lb1.grid(*row*=15, *column*=0)      lb2.grid(*row*=15, *column*=1)      lb3.grid(*row*=15, *column*=2)      lb4.grid(*row*=15, *column*=3)      lb5.grid(*row*=15, *column*=4)      lb6.grid(*row*=15, *column*=5)      lb1.bind('<MouseWheel>', onmousewheel)      lb2.bind('<MouseWheel>', onmousewheel)      lb3.bind('<MouseWheel>', onmousewheel)      lb4.bind('<MouseWheel>', onmousewheel)      lb5.bind('<MouseWheel>', onmousewheel)      lb6.bind('<MouseWheel>', onmousewheel)      cur.execute("select \*from med")  *for* i *in* cur:          cx += 1          seq = (str(i[0]), str(i[1]))          lb1.insert(cx, '. '.join(seq))          lb2.insert(cx, i[2])          lb3.insert(cx, i[3])          lb4.insert(cx, i[4])          lb5.insert(cx, i[5])          lb6.insert(cx, i[6] + '    ' + i[7] + '    ' + i[8])      c.commit()  def reset():      global sto, accept  *for* i *in* range(1, len(columns)):          Label(sto, *width*=15, *text*=' ' \* (14 - len(str(columns[i]))) + str(columns[i]) + ':').grid(*row*=i + 2, *column*=0)          accept[i] = Entry(sto)          accept[i].grid(*row*=i + 2, *column*=1)  def submit():      global accept, c, cur, columns, sto      x = [''] \* 10      cur.execute("select \* from med")  *for* i *in* cur:          y = int(i[0])  *for* i *in* range(1, 9):          x[i] = accept[i].get()      sql = "insert into med values('%s','%s','%s','%s','%s','%s','%s','%s','%s')" % (      y + 1, x[1], x[2], x[3], x[4], x[5], x[6], x[7], x[8])      cur.execute(sql)      cur.execute("select \* from med")      c.commit()      top = Tk()      Label(top, *width*=20, *text*='Success!').pack()      top.mainloop()      main\_menu()  def chk():      global cur, c, accept, sto      cur.execute("select \* from med")  *for* i *in* cur:  *if* accept[6].get() == i[6] and i[1] == accept[1].get():              sql = "update med set qty\_left = '%s' where name = '%s'" % (              str(int(i[3]) + int(accept[3].get())), accept[1].get())              cur.execute(sql)              c.commit()              top = Tk()              Label(top, *width*=20, *text*='Modified!').pack()              top.mainloop()              main\_menu()  *else*:              submit()      c.commit()  def exp\_date():      global exp, s, c, cur, flag, apt, flags      apt.destroy()      flag = 'exp'  *from* datetime *import* date      now = time.localtime()      n = []      cur.execute("select \*from med")  *for* i *in* cur:          n.append(i[1])      c.commit()      exp = Tk()      exp.title('EXPIRY CHECK')      Label(exp, *text*='Today : ' + str(now[2]) + '/' + str(now[1]) + '/' + str(now[0])).grid(*row*=0, *column*=0,  *columnspan*=3)      Label(exp, *text*='Selling Expired Medicines and Drugs is Illegal').grid(*row*=1, *column*=0, *columnspan*=3)      Label(exp, *text*='-' \* 80).grid(*row*=2, *column*=0, *columnspan*=3)      s = Spinbox(exp, *values*=n)      s.grid(*row*=3, *column*=0)      Button(exp, *text*='Check Expiry date', *bg*='red', *fg*='white', *command*=s\_exp).grid(*row*=3, *column*=1)      Label(exp, *text*='-' \* 80).grid(*row*=4, *column*=0, *columnspan*=3)  *if* flags == 'apt1':          Button(exp, *text*='Main Menu', *bg*='green', *fg*='white').grid(*row*=5, *column*=2)  *else*:          Button(exp, *width*=20, *text*='Check Products expiring', *bg*='red', *fg*='white', *command*=exp\_dt).grid(*row*=5,  *column*=0)          Button(exp, *text*='Main Menu', *bg*='green', *fg*='white', *command*=main\_menu).grid(*row*=5, *column*=2)      exp.mainloop()  def s\_exp():      global c, cur, s, exp, top  *from* datetime *import* date      now = time.localtime()      d1 = date(now[0], now[1], now[2])      cur.execute("select \* from med")  *for* i *in* cur:  *if* (i[1] == s.get()):              q = i[6]              d2 = date(int('20' + q[8:10]), int(q[3:5]), int(q[0:2]))  *if* d1 > d2:                  Label(exp, *text*='EXPIRED! on ' + i[6]).grid(*row*=3, *column*=2)                  top = Tk()                  Label(top, *text*='EXPIRED!').pack()  *else*:                  Label(exp, *text*=i[6]).grid(*row*=3, *column*=2)      c.commit()  def exp\_dt():      global c, cur, exp, top      x = 0      z = 1  *from* datetime *import* datetime, timedelta      N = 7      dt = datetime.now() + timedelta(*days*=N)      d = str(dt)  *from* datetime *import* date      now = time.localtime()      d1 = date(now[0], now[1], now[2])      d3 = date(int(d[0:4]), int(d[5:7]), int(d[8:10]))      Label(exp, *text*='S.No' + '   ' + 'Name' + '     Qty.    ' + 'Exp\_date').grid(*row*=6, *column*=0, *columnspan*=2)      cur.execute("select \* from med")  *for* i *in* cur:          s = i[6]          d2 = date(int('20' + s[8:10]), int(s[3:5]), int(s[0:2]))  *if* d1 < d2 < d3:              Label(exp, *text*=str(z) + '.      ' + str(i[1]) + '    ' + str(i[3]) + '    ' + str(i[6])).grid(*row*=x + 7,  *column*=0,  *columnspan*=2)              x += 1              z += 1  *elif* d1 > d2:              top = Tk()              Label(top, *width*=20, *text*=str(i[1]) + ' is EXPIRED!').pack()      c.commit()  def billing():      global c, cur, apt, flag, t, name, name1, add, st, names, qty, sl, qtys, vc\_id, n, namee, lb1      t = 0      vc\_id = ''      names = []      qty = []      sl = []      n = []      qtys = [''] \* 10      cur.execute("select \*from med")  *for* i *in* cur:          n.append(i[1])      c.commit()  *if* flag == 'st':          st.destroy()  *else*:          apt.destroy()      flag = 'st'      st = Tk()      st.title('BILLING SYSTEM')      Label(st, *text*='-' \* 48 + 'BILLING SYSTEM' + '-' \* 49).grid(*row*=0, *column*=0, *columnspan*=7)      Label(st, *text*='Enter Name: ').grid(*row*=1, *column*=0)      name1 = Entry(st)      name1.grid(*row*=1, *column*=1)      Label(st, *text*='Enter Address: ').grid(*row*=2, *column*=0)      add = Entry(st)      add.grid(*row*=2, *column*=1)      Label(st, *text*="Value Id (if available)").grid(*row*=3, *column*=0)      vc\_id = Entry(st)      vc\_id.grid(*row*=3, *column*=1)      Button(st, *text*='Check V.C.', *bg*='green', *fg*='white', *command*=blue).grid(*row*=4, *column*=0)      Label(st, *text*='-' \* 115).grid(*row*=6, *column*=0, *columnspan*=7)      Label(st, *text*='SELECT PRODUCT', *width*=25, *relief*='ridge').grid(*row*=7, *column*=0)      Label(st, *text*=' RACK  QTY LEFT     COST          ', *width*=25, *relief*='ridge').grid(*row*=7, *column*=1)      Button(st, *text*='Add to bill', *bg*='blue', *fg*='white', *width*=15, *command*=append2bill).grid(*row*=8, *column*=6)      Label(st, *text*='QUANTITY', *width*=20, *relief*='ridge').grid(*row*=7, *column*=5)      qtys = Entry(st)      qtys.grid(*row*=8, *column*=5)      refresh()      Button(st, *width*=15, *text*='Main Menu', *bg*='green', *fg*='white', *command*=main\_menu).grid(*row*=1, *column*=6)      Button(st, *width*=15, *text*='Refresh Stock', *bg*='skyblue', *fg*='black', *command*=refresh).grid(*row*=3, *column*=6)      Button(st, *width*=15, *text*='Reset Bill', *bg*='red', *fg*='white', *command*=billing).grid(*row*=4, *column*=6)      Button(st, *width*=15, *text*='Print Bill', *bg*='orange', *fg*='white', *command*=print\_bill).grid(*row*=5, *column*=6)      Button(st, *width*=15, *text*='Save Bill', *bg*='blue', *fg*='white', *command*=make\_bill).grid(*row*=7, *column*=6)      st.mainloop()  def refresh():      global cur, c, st, lb1, lb2, vsb      def onvsb(\**args*):          lb1.yview(\*args)          lb2.yview(\*args)      cx = 0      vsb = Scrollbar(*orient*='vertical', *command*=onvsb)      lb1 = Listbox(st, *width*=25, *yscrollcommand*=vsb.set)      lb2 = Listbox(st, *width*=25, *yscrollcommand*=vsb.set)      vsb.grid(*row*=8, *column*=2, *sticky*=N + S)      lb1.grid(*row*=8, *column*=0)      lb2.grid(*row*=8, *column*=1)      cur.execute("select \*from med")  *for* i *in* cur:          cx += 1          lb1.insert(cx, str(i[0]) + '. ' + str(i[1]))          lb2.insert(cx, ' ' + str(i[7]) + '        ' + str(i[3]) + '             Rs ' + str(i[4]))      c.commit()      lb1.bind('<<ListboxSelect>>', select\_mn)  def select\_mn(*e*):      global st, lb1, n, p, nm, sl1      p = lb1.curselection()      x = 0      sl1 = ''  *from* datetime *import* date      now = time.localtime()      d1 = date(now[0], now[1], now[2])      cur.execute("select \* from med")  *for* i *in* cur:  *if* x == int(p[0]):              sl1 = int(i[0])  *break*          x += 1      c.commit()      print(sl1)      nm = n[x]      print(nm)  def append2bill():      global st, names, nm, qty, sl, cur, c, sl1      sl.append(sl1)      names.append(nm)      qty.append(qtys.get())      print(qty)      print(sl[len(sl) - 1], names[len(names) - 1], qty[len(qty) - 1])  def blue():      global st, c, cur, named, addd, t, vc\_id      cur.execute("select \* from cus")  *for* i *in* cur:  *if* vc\_id.get() != '' and int(vc\_id.get()) == i[2]:              named = i[0]              addd = i[1]              Label(st, *text*=named, *width*=20).grid(*row*=1, *column*=1)              Label(st, *text*=addd, *width*=20).grid(*row*=2, *column*=1)              Label(st, *text*=i[2], *width*=20).grid(*row*=3, *column*=1)              Label(st, *text*='Valued Customer!').grid(*row*=4, *column*=1)              t = 1  *break*      c.commit()  def make\_bill():      global t, c, B, cur, st, names, qty, sl, named, addd, name1, add, det, vc\_id      price = [0.0] \* 10      q = 0      det = ['', '', '', '', '', '', '', '']      det[2] = str(sl)  *for* i *in* range(len(sl)):          print(sl[i], ' ', qty[i], ' ', names[i])  *for* k *in* range(len(sl)):          cur.execute("select \* from med where sl\_no=?", (sl[k],))  *for* i *in* cur:              price[k] = int(qty[k]) \* float(i[4])              print(qty[k], price[k])              cur.execute("update med set qty\_left=? where sl\_no=?", (int(i[3]) - int(qty[k]), sl[k]))          c.commit()      det[5] = str(random.randint(100, 999))      B = 'bill\_' + str(det[5]) + '.txt'      total = 0.00  *for* i *in* range(10):  *if* price[i] != '':              total += price[i]  *# totalling*      m = '\n\n\n'      m += "===============================================\n"      m += "                                  No :%s\n\n" % det[5]      m += " ADANI GENMART\n"      m += "-----------------------------------------------\n"  *if* t == 1:          m += "Name: %s\n" % named          m += "Address: %s\n" % addd          det[0] = named          det[1] = addd          cur.execute('select \* from cus')  *for* i *in* cur:  *if* i[0] == named:                  det[7] = i[2]  *else*:          m += "Name: %s\n" % name1.get()          m += "Address: %s\n" % add.get()          det[0] = name1.get()          det[1] = add.get()      m += "-----------------------------------------------\n"      m += "Product                      Qty.       Price\n"      m += "-----------------------------------------------\n"  *for* i *in* range(len(sl)):  *if* names[i] != 'nil':              s1 = ' '              s1 = (names[i]) + (s1 \* (27 - len(names[i]))) + s1 \* (3 - len(qty[i])) + qty[i] + s1 \* (                          15 - len(str(price[i]))) + str(price[i]) + '\n'              m += s1      m += "\n-----------------------------------------------\n"  *if* t == 1:          ntotal = total \* 0.8          m += 'Total' + (' ' \* 25) + (' ' \* (15 - len(str(total)))) + str(total) + '\n'          m += "Valued customer Discount" + (' ' \* (20 - len(str(total - ntotal)))) + '-' + str(total - ntotal) + '\n'          m += "-----------------------------------------------\n"          m += 'Total' + (' ' \* 25) + (' ' \* (12 - len(str(ntotal)))) + 'Rs ' + str(ntotal) + '\n'          det[3] = str(ntotal)  *else*:          m += 'Total' + (' ' \* 25) + (' ' \* (12 - len(str(total)))) + 'Rs ' + str(total) + '\n'          det[3] = str(total)      m += "-----------------------------------------------\n\n"      m += "Dealer 's signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"      m += "===============================================\n"      print(m)      p = time.localtime()      det[4] = str(p[2]) + '/' + str(p[1]) + '/' + str(p[0])      det[6] = m      bill = open(B, 'w')      bill.write(m)      bill.close()      cb = ('cus\_name', 'cus\_add', 'items', 'Total\_cost', 'bill\_dt', 'bill\_no', 'bill', 'val\_id')      cur.execute('insert into bills values(?,?,?,?,?,?,?,?)',                  (det[0], det[1], det[2], det[3], det[4], det[5], det[6], det[7]))      c.commit()  def print\_bill():      win32api.ShellExecute(0, "print", B, '/d:"%s"' % win32print.GetDefaultPrinter(), ".", 0)  def show\_rev():      global c, cur, flag, rev      apt.destroy()      cb = ('cus\_name', 'cus\_add', 'items', 'Total\_cost', 'bill\_dt', 'bill\_no', 'bill', 'val\_id')      flag = 'rev'      rev = Tk()      total = 0.0      today = str(time.localtime()[2]) + '/' + str(time.localtime()[1]) + '/' + str(time.localtime()[0])      Label(rev, *text*='Today: ' + today).grid(*row*=0, *column*=0)      cur.execute('select \* from bills')  *for* i *in* cur:  *if* i[4] == today:              total += float(i[3])      print(total)      Label(rev, *width*=22, *text*='Total revenue: Rs ' + str(total), *bg*='blue', *fg*='white').grid(*row*=1, *column*=0)      cx = 0      vsb = Scrollbar(*orient*='vertical')      lb1 = Listbox(rev, *width*=25, *yscrollcommand*=vsb.set)      vsb.grid(*row*=2, *column*=1, *sticky*=N + S)      lb1.grid(*row*=2, *column*=0)      vsb.config(*command*=lb1.yview)      cur.execute("select \* from bills")  *for* i *in* cur:  *if* i[4] == today:              cx += 1              lb1.insert(cx, 'Bill No.: ' + str(i[5]) + '    : Rs ' + str(i[3]))      c.commit()      Button(rev, *text*='Main Menu', *bg*='green', *fg*='white', *command*=main\_menu).grid(*row*=15, *column*=0)      rev.mainloop()  def search():      global c, cur, flag, st, mn, sym, flags      flag = 'st'      apt.destroy()      cur.execute("Select \* from med")      symp = ['nil']      med\_name = ['nil']  *for* i *in* cur:          symp.append(i[5])          med\_name.append(i[1])      st = Tk()      st.title('SEARCH')      Label(st, *bg*='green', *fg*='white', *text*=' SEARCH FOR MEDICINE ').grid(*row*=0, *column*=0, *columnspan*=3)      Label(st, *text*='~' \* 40).grid(*row*=1, *column*=0, *columnspan*=3)      Label(st, *text*='Symptom Name').grid(*row*=3, *column*=0)      sym = Spinbox(st, *values*=symp)      sym.grid(*row*=3, *column*=1)      Button(st, *width*=15, *text*='Search', *bg*='blue', *fg*='white', *command*=search\_med).grid(*row*=3, *column*=2)      Label(st, *text*='-' \* 70).grid(*row*=4, *column*=0, *columnspan*=3)  *if* flags == 'apt1':          Button(st, *width*=15, *text*='Main Menu', *bg*='green', *fg*='white').grid(*row*=6, *column*=2)  *else*:          Button(st, *width*=15, *text*='Main Menu', *bg*='green', *fg*='white', *command*=main\_menu).grid(*row*=6, *column*=2)      st.mainloop()  def search\_med():      global c, cur, st, sym, columns      cur.execute("select \* from med")      y = []      x = 0  *for* i *in* cur:  *if* i[5] == sym.get():              y.append(                  str(i[0]) + '. ' + str(i[1]) + '  Rs ' + str(i[4]) + '    Rack : ' + str(i[7]) + '    Mfg : ' + str(                      i[8]))              x = x + 1      top = Tk()  *for* i *in* range(len(y)):          Label(top, *text*=y[i]).grid(*row*=i, *column*=0)      Button(top, *text*='OK', *command*=top.destroy).grid(*row*=5, *column*=0)      c.commit()      top.mainloop()  def val\_cus():      global val, flag, dbt, name\_vc, add\_vc, cur, c, vc\_id      apt.destroy()      cur.execute("select \* from cus")      flag = 'val'      val = Tk()      Label(val, *bg*='blue', *fg*='white', *text*="\*\*\*\*ENTER VALUED CUSTOMER DETAILS\*\*\*\*").grid(*row*=0, *column*=0, *columnspan*=3)      Label(val, *text*="-" \* 60).grid(*row*=1, *column*=0, *columnspan*=3)      Label(val, *text*="Name: ").grid(*row*=2, *column*=0)      name\_vc = Entry(val)      name\_vc.grid(*row*=2, *column*=1)      Label(val, *text*="Address: ").grid(*row*=3, *column*=0)      add\_vc = Entry(val)      add\_vc.grid(*row*=3, *column*=1)      Label(val, *text*="Value Id: ").grid(*row*=4, *column*=0)      vc\_id = Entry(val)      vc\_id.grid(*row*=4, *column*=1)      Button(val, *text*='Submit', *bg*='blue', *fg*='white', *command*=val\_get).grid(*row*=5, *column*=1)      Button(val, *text*='Main Menu', *bg*='green', *fg*='white', *command*=main\_menu).grid(*row*=5, *column*=2)      Label(val, *text*='-' \* 60).grid(*row*=6, *column*=0, *columnspan*=3)      val.mainloop()  def val\_get():      global name\_vc, add\_vc, val, dbt, c, cur, apt, vc\_id      cur.execute("insert into cus values(?,?,?)", (name\_vc.get(), add\_vc.get(), vc\_id.get()))      l.execute("insert into log values(?,?)", (name\_vc.get(), vc\_id.get()))      cur.execute("select \* from cus")  *for* i *in* cur:          print(i[0], i[1], i[2])      c.commit()      login.commit()  def again():      global un, pwd, flag, root, apt  *if* flag == 'apt':          apt.destroy()      root = Tk()      root.geometry('300x150')      root.title('Login')      Label(root, *text*='ADANI GENMART').grid(*row*=0, *column*=0, *columnspan*=5)      Label(root, *text*='-------------------------------------------------------').grid(*row*=1, *column*=0, *columnspan*=5)      Label(root, *text*='Username').grid(*row*=3, *column*=0)      un = Entry(root, *width*=30)      un.grid(*row*=3, *column*=1)      Label(root, *text*='Password').grid(*row*=4, *column*=0)      pwd = Entry(root, *width*=30)      pwd.grid(*row*=4, *column*=1)      pwd.config(*show*='\*')      Button(root, *width*=6, *bg*='blue', *fg*='white', *text*='Enter', *command*=check).grid(*row*=5, *column*=1)      root.mainloop()  def check():      global un, pwd, login, l, root      u = un.get()      p = pwd.get()      l.execute("select \* from log")  *for* i *in* l:  *if* i[0] == u and i[1] == p and u == 'admin':              root.destroy()              open\_win()      login.commit()  def main\_menu():      global sto, apt, flag, root, st, val, exp, st1, rev  *if* flag == 'sto':          sto.destroy()  *if* flag == 'rev':          rev.destroy()  *elif* flag == 'st':          st.destroy()  *elif* flag == 'st1':          st1.destroy()  *elif* flag == 'val':          val.destroy()  *elif* flag == 'exp':          exp.destroy()  *elif* flag == 'd':          d.destroy()      open\_win()  def again1():      global flags      apt.destroy()      flags = ''      again()  again() |

### 5.2.1 Code Efficiency

The Tkinter framework makes use of Python provides us a **faster and efficient way** in order to build useful applications that would have taken much time if you had to program directly in C/C++ with the help of native OS system libraries.

5.3 Testing Approach

Following testing approaches have been performed on the system: Unit Testing,

Integration Testing, System Testing

**5.3.1 Unit Testing**

Unit Testing is done by Testing each module as a separate function. Every module in the project is tested separately.

**5.3.2 Integration Testing**

Integration Testing means Testing more than one Module at a time.

**5.3.3 System Testing**

System testing has been performed on a completely integrated system to assess whether the system meets the requirements and passes the test cases defined above. The application has successfully met all the criteria and has passed all the tests as separate units and as a completely integrated system.

## 5.4 Modification and Improvements

After testing and handling all errors, exceptions and bugs, errors were solved by

troubleshooting, exceptions were handled by try catch block at the correct places avoiding redundancy of code from index files. Bugs can never be removed completely but most of them are solved and remaining are under development. The project can be further improved by adding points system to every one who buys anything from their shop which can be claimed on their next order as a discount.

## 6 RESULTS AND DISCUSSIONS

### 6.1 Test Report

|  |  |  |  |
| --- | --- | --- | --- |
| Test Condition | Input | Expected result | Actual Result |
| Login | Email  Password | Validate and redirect to homepage | Validate and redirect to homepage |
| Add Medicine | Name  Qty  Rack Location  Manufacturer  Cost  Purpose  Expiry Date | Add Medicine to Database. Add all details of the medicine such as Qty, Rack Location,  Manufacturer, Cost, Purpose and Expiry Date | Add Medicine to Database. Add all details of the medicine such as Qty, Rack Location,  Manufacturer, Cost, Purpose and Expiry Date |
| Delete Medicine | Name  QTY | Delete Medicine from Database | Delete Medicine from Database |
| Add VIP customer | Name  Address  VIP ID | Add To Database | Adds to database |

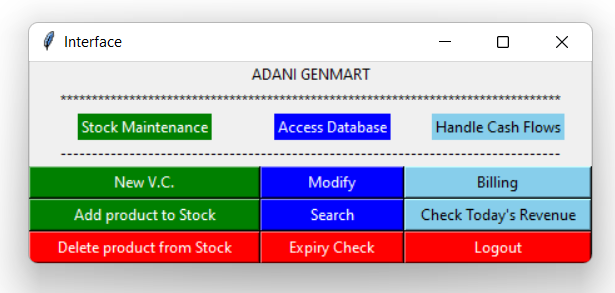
### 6.2 User Documentation

**Module 1: Log In**



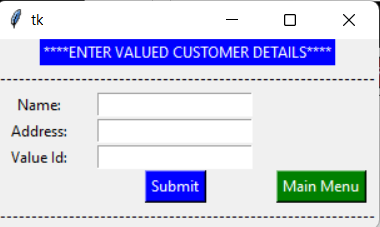
**Figure 21:** Login

**Module 2:** Homepage



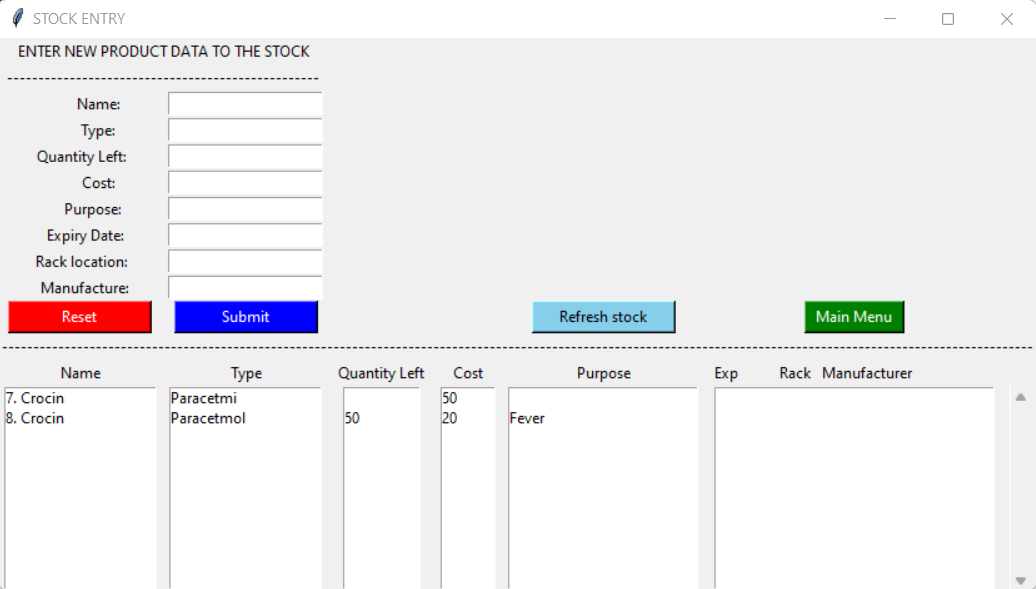
**Figure 22:** Homepage

**Module 3:** Add Valued Customer



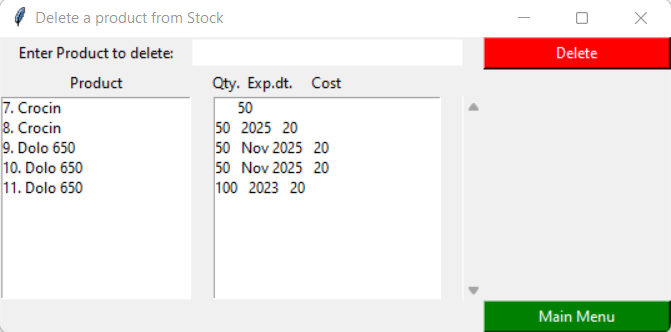
**Figure 23:** Add Valued Customer

**Module 4: Add product to stock**

****

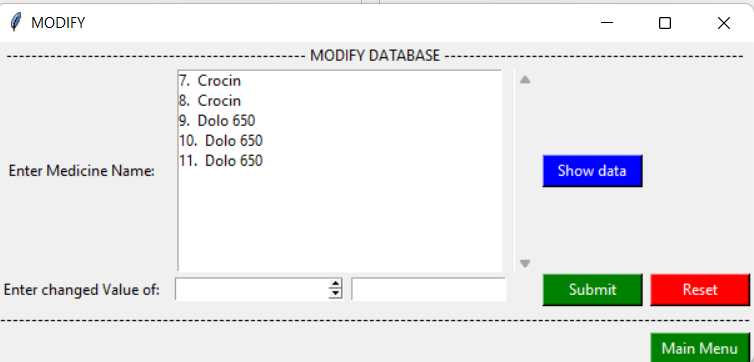
**Figure 24:** Add Product

**Module 5: Delete product to stock**



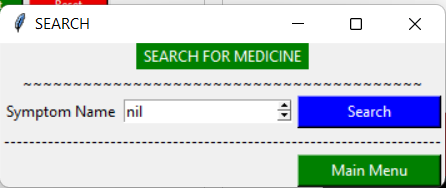
**Figure 25:** Delete Product

**Module 6: Modify**



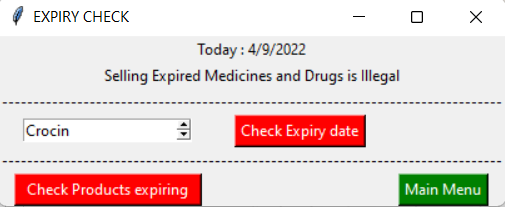
**Figure 26:** Modify

**Module 7: Search**

****

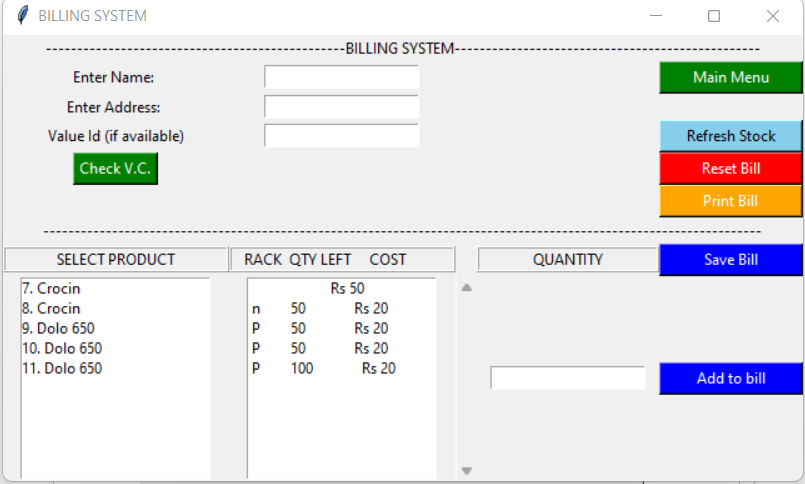
**Figure 27:** Search

**Module 8: Expiry Check**



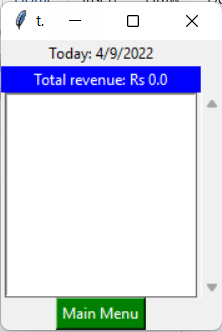
**Figure 28:** Check expired medicines

**Module 10:Billing**

****

**Figure292:** Billing

**Module 11: Check Today’s Revenue**

****

**Figure 30:** Check Revenue

## 7 CONCLUSION

### 7.1 Conclusion

The project has been implemented according to the objectives mentioned in the beginning. The dynamic nature of the project allows an easier way to maintain the infrastructure.

### 7.2 Limitations

* This project is not accessible to users, we can add user interface for better reach and use.

### 7.3 Future scope of the project

* The project can be implemented in a way that allows multiple pharmacies to store their information.
* The project can be designed in a way that allows a ‘Staff’ module to be made.
* The project can be made to add various other modules related to an medical store..

## References

* https://docs.microsoft.com/
* https://social.msdn.microsoft.com/
* https://stackoverflow.com/
* https://www.tutorialspoint.com/
* https://www.tutorialsteacher.com/