**SOFTWARE ENGINEERING LAB**

**PROJECT REPORT**

Pharmacy Management System

* SURAJ P

19BCE1044

**CONTENTS**

* SRS Document
* Use Case Diagram
* Class Diagram
* Sequence Diagram
* Work Breakdown Structure
* Gantt Chart
* Data Flow Diagram
* Activity Diagram
* State Diagram
* Deployment Diagram
* Test Case Sheet
* Conclusion

**#1 SRS Document**

**INDEX**

1.INTRODUCTION

1.1 Purpose

1.2 Project Scope

1. OVERALL DESCRIPTION

2.1 Product Perspective

2.2 Product Features

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

3. SYSTEM FEATURES

3.1 Description and Priority

3.2 Stimulus/ Response Sequences

3.3 Functional Requirements

4. REQUIREMENTS

4(a) External Interface Requirements

4.1 User Interfaces

4.2 Hardware Interfaces

4.3 Software Interfaces

4.4 Communication Interfaces

4(b) Non-functional Requirements

4.1 Performance Requirements

4.2 Safety Requirements

4.3 Security Requirements

4.4 Software Requirements

**1.INTRODUCTION**

1.1 Purpose

The software developed is that of a Pharmacy Management System. The product created can be used to effectively store and manage medicines. The sole purpose of creating an SRS document is in the fact that it will explain the purpose and salient features of the software. It also conveys information about the application's requirements, both functional and nonfunctional.

This document will be most important to the application developers who would find this document fruitful to understand how the code needs to be written to understand the user’s needs. Similarly, the user will also need to read the document to a create a bridge between his/her requirements and the development team.

1.2 Project Scope

The Pharmacy Management System has been designed to assist pharmacists as managing with handwritten records can tend to be quite tedious and difficult to keep track. It allows the pharmacist to keep track of all transactions occurring, the medicines available, the expired medicines and can also create an invoice. All the data is stored securely in a relational database with encryption to provide a sense of security.

It also ensures a very user-friendly interface with vivid visuals for ease of use. The software created does not require internet connectivity thus making it hassle free to use.

**2.OVERALL DESCRIPTION**

2.1 Product Perspective

The software’s main objective is to develop a platform for the pharmacist to do all his work in one place without having to worry about maintaining a separate record. Hence the application created is self-contained with its own front end and back end. It is a standalone product and does not depend on the availability of other software.

2.2 Product Features

The product implements the following functions:

1. Display medicines in the inventory
2. Medicine Purchases
3. Perform sales of medicines based on availability
4. Add and display dealers
5. Keeping a track of expired medicines

2.3 User Classes and Characteristics

User: Pharmacist (Admin)

Characteristics: As the shop’s admin he/she has control over most of the operations to be performed. The admin is thus responsible for the entire software’s working and thus has a good understanding of its working. The admin account also has its own login credentials to ensure security.

He/She can perform sales, add more medicines along with dealers and check the inventory regularly.

2.4 Operating Environment

The Pharmacy Management System works on any Operating System.

It is available on any system with Windows 7 and above or Mac OS (10) or Linux (Ubuntu 12).

The software uses a MySQL database management system along with the actual software to operate.

2.5 Design and Implementation Constraints

* The code should be fragmented into modules
* Implement the database using a centralized database management system.

**3.SYSTEM FEATURES**

(1) **View Inventory**

Description

Allow the pharmacist to view the medicines available in his inventory. He can also search for a specific medicine or a list of medicines based on his requirements.

Stimulus/Response

After selecting this option in the main menu,

1. Enter the value of different parameters such as name, quantity, price etc.

List of all medicines with that value will be displayed.

1. A separate button View the Inventory is pressed

The entire inventory details are displayed.

Functional Requirements

REQ 1: The user shall be able to view medicines based on different options

REQ 2: The user shall be able to view all medicines

(2) **Purchase Medicine**

Description

In this operation, the medicines can be purchased from respective dealers. The complete purchase history of medicines in the pharmacy can be viewed based on the user’s requirements from the database.

Stimulus/Response

1. Enter the name, quantity, dealer, and price with which it was bought as individual inputs. Click on add to Inventory.

It is then stored in the database with an ID.

Functional Requirements

REQ 1: The user shall be able add medicines to the inventory

REQ 2: The user shall be able to search for a purchase

1. **Add and view Dealers**

Description

In this operation, the details about the various dealers that the pharmacist does business with are stored in the database. It can also display the dealers already present based on the name, address etc.

Response/Stimulus

1. Enter the dealer name, id, phone, and address and click on add to list

The data is stored in the database.

1. Click on the List of Dealers button

The entire dealer’s data is displayed.

Functional Requirements

REQ 1: The user shall be able add new dealers to his database

REQ 2: The user shall be able to view all dealers

1. **Medicine Sales and Generate Invoice**

Description

In this operation, the user can perform sales with a customer and along with it an invoice is generated

Response/Stimulus

1. Enter the Name of medicine

Checks if medicine name exists

1. Enter quantity to be bought

Checks if quantity to be bought is less or equal to available in inventory and displays price

1. Enter amount to be paid after all medicines bought

Balance is displayed after transaction is made and database is updated

1. Customer name is accepted

Name of Customer stored in database

1. Generate Bill is clicked

Bill is generated and displayed.

Functional Requirements

REQ 1: The user shall be able to perform sales based on availability

REQ 2: An invoice is generated and stored for future reference

(5) **View Expired Inventory**

Description

In this operation, the pharmacist can view the medicines whose expiry date have been crossed. The medicines are automatically removed from the main inventory and stored in expired inventory based on system date.

Response/Stimulus

1. The user enters the value of one of the different parameters such as name, quantity, price etc. and search button is clicked.

The expired medicine data is displayed from database of that value

1. A separate button expired inventory is clicked

The entire expired inventory is displayed

Functional Requirements

REQ 1: The user shall be able to expire inventory based on different options

REQ 2: The user shall be able to view all the expired medicines

**4. REQUIREMENTS**

4(a) EXTERNAL INTERFACE REQUIREMENTS

4.1 User Interfaces

The User Interface is done through a GUI in Python.

1. First a login screen is displayed to enter username and password. To create a more user-friendly experience the login works even by just clicking the enter button on the keyboard.
2. After which a menu of buttons with each task to be performed such as medicine purchase, sales, inventory, Expired Inventory.
3. Each of the above menu has its own separate page.

4.In the case of errors during transactions, appropriate error messages are displayed in form of message boxes.

4.2 Hardware Interfaces

It is implemented on an Intel® Core™ i5 or i7 processor on a 64-bit computer with at least 1GB RAM. The invoice generated can be printed by connecting it to an external printer.

4.3 Software Interfaces

1. Python 3.8 in IDLE with tkinter GUI

2.MySQL 8.0 along with MySQL workbench

must be installed

4.4 Communication Interfaces

1.The software communicates with database using an SQL cursor perform SQL statements on a relational database named pharmacy.

2.Uses tkinter to generate a dynamic user interface

Other protocols not required as software not connected to internet

4(b) Non-Functional Requirements

4.1 Performance Requirements

1.Since the software uses Relational Database through MySQL server it can store large amounts of data without any error.

2.Any operation performed displays the result almost immediately with time lag of at most 3 milliseconds.

3.Even errors are displayed immediately to prevent loss of data.

4.2 Safety Requirements

The software has also been created in such a way to prevent any corruption of the system through any malware or virus. The user must perform one function at a time and database should be maintained carefully to prevent overlapping and ultimate loss of data.

4.3 Security Requirements

The database storing the pharmacy details can only be accessed through a password.

Hence the data is secure. Separate login credentials have been assigned to the user to ensure security.

The framework should be accessible only by the admin pharmacist(user)

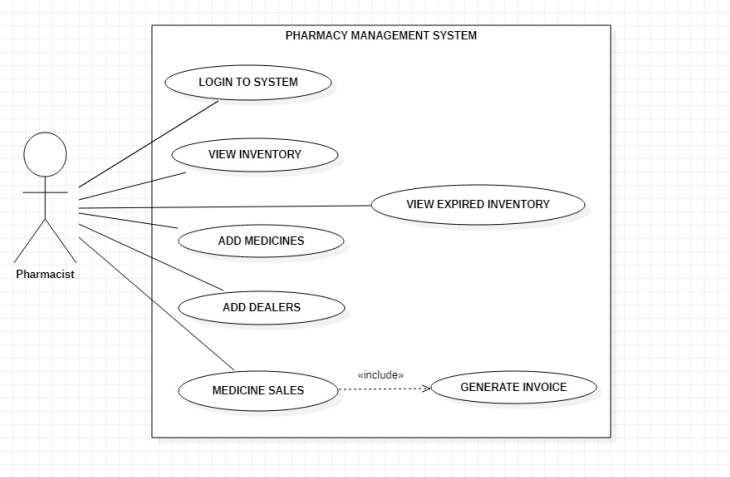
4.4 Software Quality Attributes

1. AVAILABILITY: The medicines present in the inventory should be made available to the customers during sales
2. CORRECTNESS: The quantity of stock available for each medicine should be updated and changed every time a sales transaction occurs
3. EFFICIENCY: The system should provide all the right tools to support all the features mentioned.

4.5 Other Requirements

All requirements have been specified.

**#2 Use Case Diagram**



From the above diagram, we can get a clear view of all the possible use cases present in the project. It provides information regarding the basic functionality of the software in clear diagrammatic representation.

The Primary Actor is the Pharmacist. He/She interacts with the software and preforms events such as

1.Logging In

2.View Inventory

3.Add Medicines

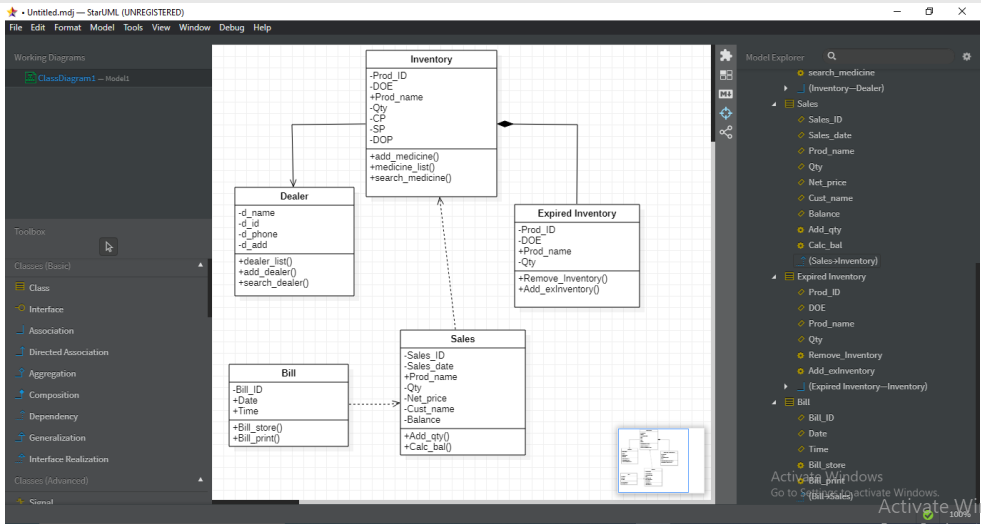
4.Add Dealers

5.Sales

6.View Expired Inventory

The Sales use case also “includes” the module Generate Invoice as the actor can only view the bill after the actual sales is done.

**#3 Class Diagram**

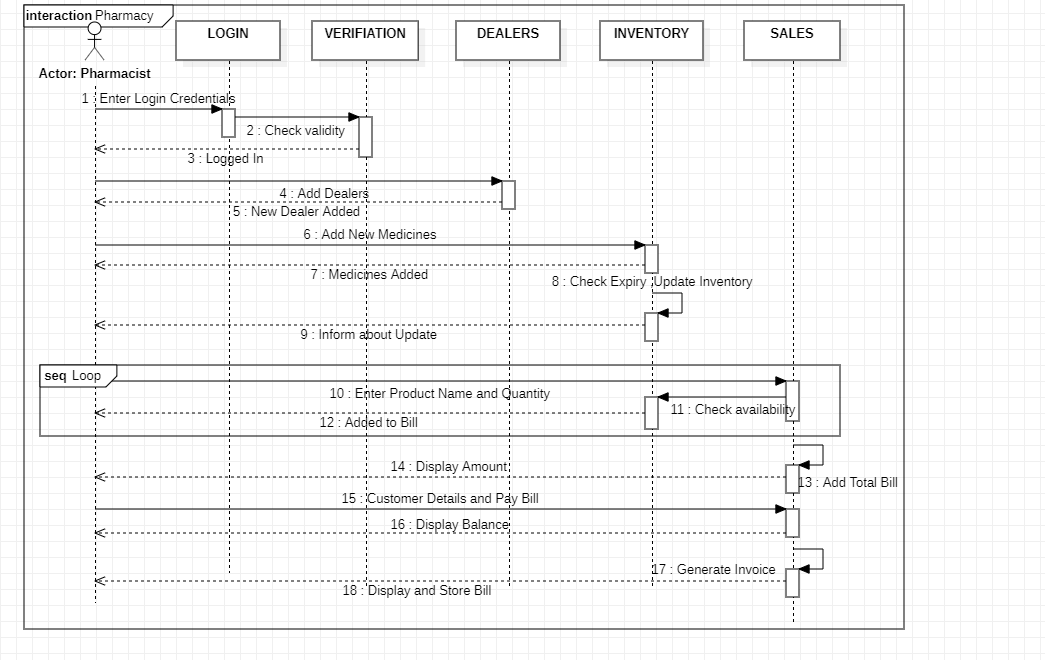


The above diagram describes the system through an object-oriented approach by visualizing the different types of classes and the kinds of static relationships that exist among them. It also illustrates the operations and attributes of the classes.

All the classes used in the project ranging from Dealers, Inventory, Sales, Expired Inventory and Bill have been identified. The attributes and most functions that are public(denoted by ‘+’) are accessible by other class objects such as Prod Name and the private attributes(denoted by ‘-’) can be accessed by the specific class only.

Now that the classes have been properly identified we move the sequence diagram to understand the appropriate sequence of actions of the functions defined in each class.

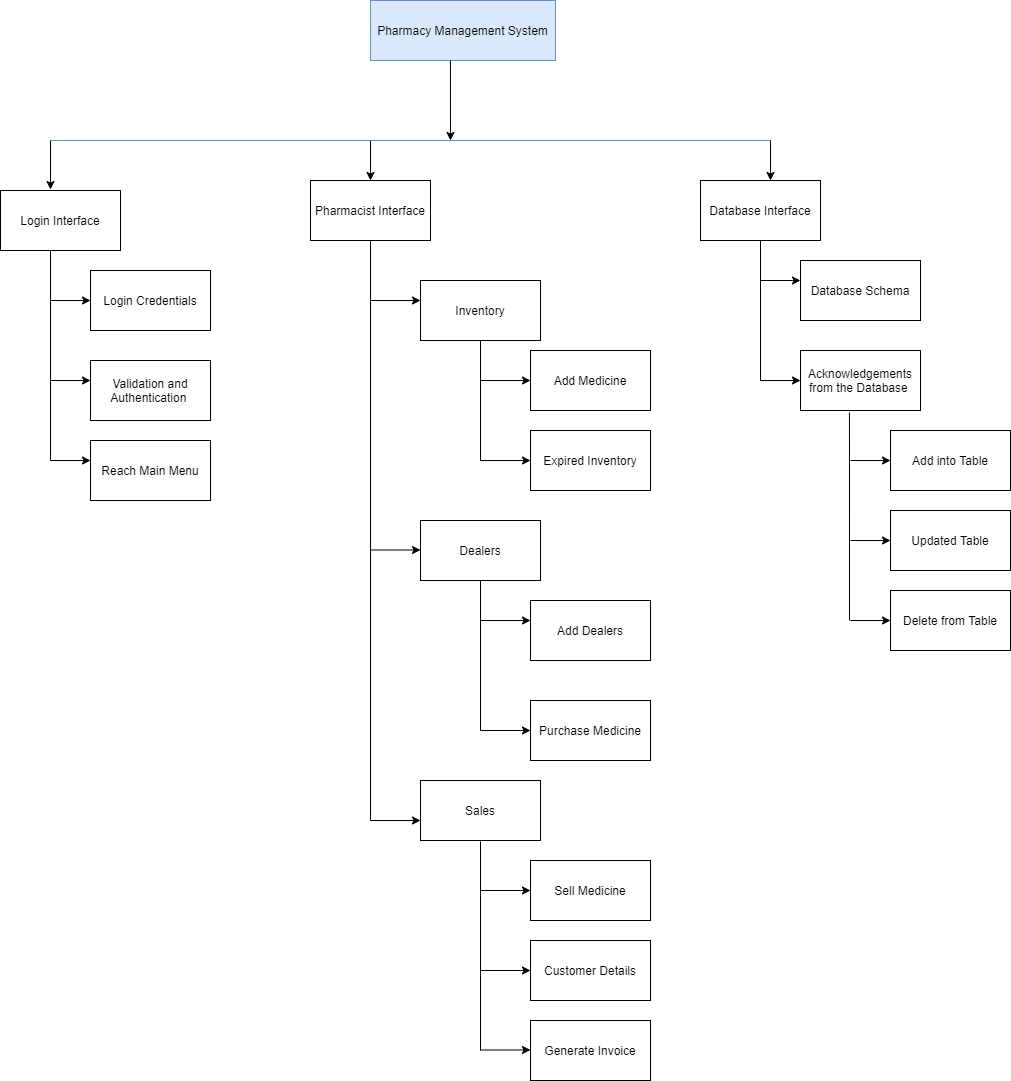
**#4 Sequence Diagram**



In the above diagram we now identify the sequence of steps in the project right from login to Medicine sales. It depicts interaction between objects in a sequential order Sequence diagrams describe how and in what order the objects in a system function.

Here the actor is the Pharmacist, after logging in, he can add new dealers to the Pharmacy. From the dealers, new medicines can be bought, and expired ones can be sent as an alert. The medicines can then be sold to customers. The medicine to be sold is entered and stock is checked. This process is placed in a loop till bill ends. Then the total bill amount is displayed. After payment, the invoice is created and stored.

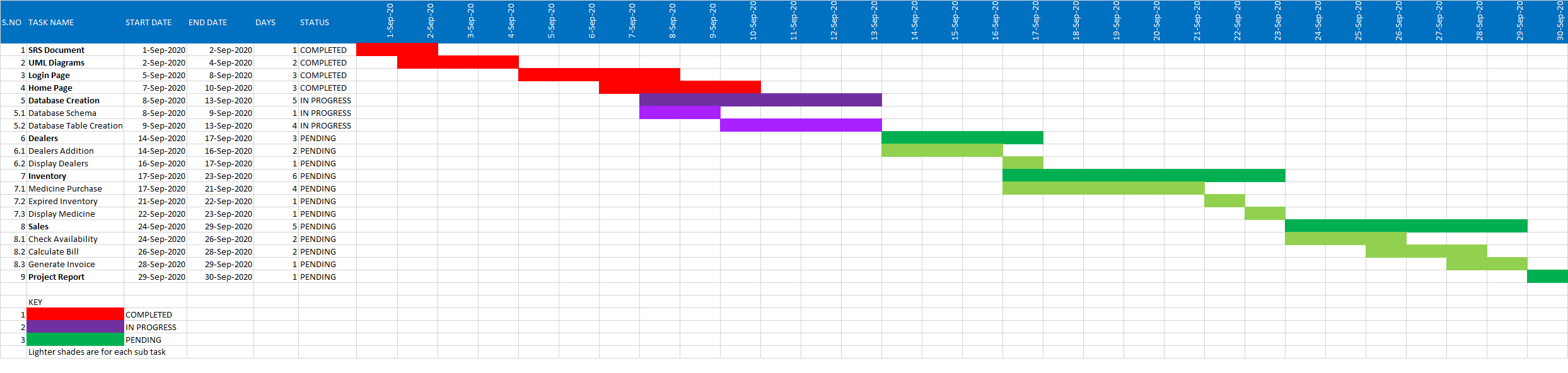
**#5 Work Breakdown Structure**



From the diagram we see that the diagram begins with a box at the top, which represents the Pharmacy Management System. Then, there are sub-components under the box, identifying all the interfaces. Under each interface, there are sub-elements listed. These sub-elements are the activities that should be performed to achieve the deliverables. Thus, WBS is the cornerstone of effective project planning, execution and monitoring.

For example, under the Login Interface only after proper validation of credentials the user can move to the Main Page which are listed as sub-elements.

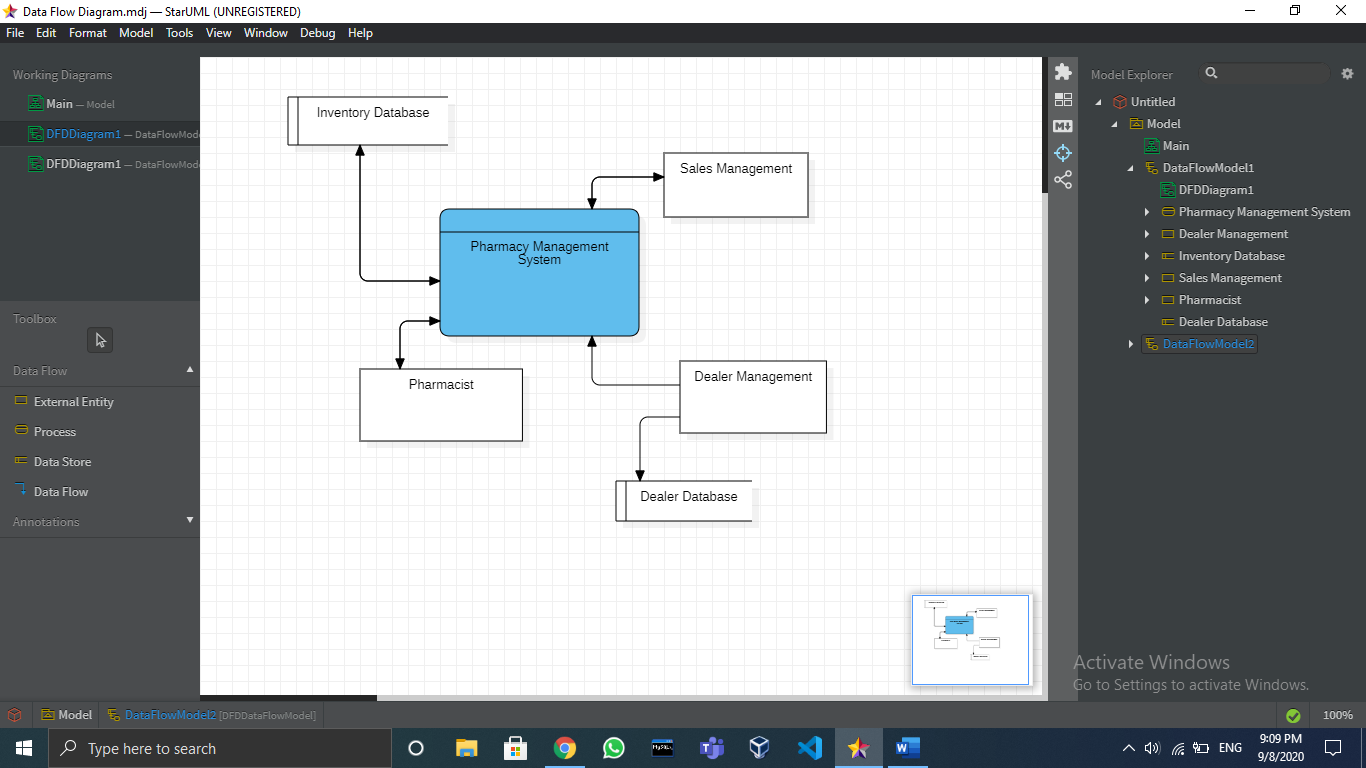
**#6 Gantt Chart**

****

From the image above we can see that on the left of the chart is a list of the activities performed while creating the software with start and end date and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration, and end date of the activity. Depending on the status of the work done, colour codes have been assigned. Hence Gantt Chart is a useful way of showing activities displayed against time.

**#7 Data Flow Diagram**

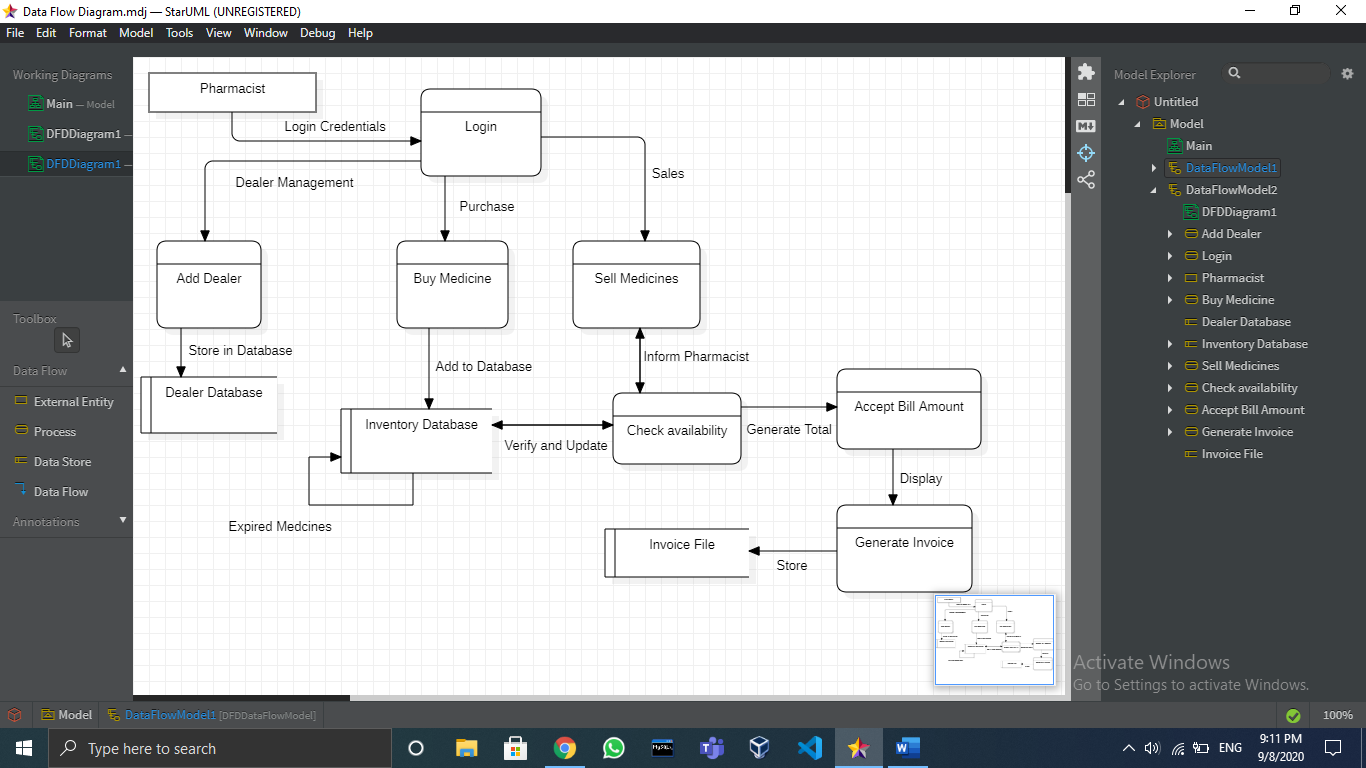
1. LEVEL 0 DIAGRAM



The above diagram shows how information enters and leaves the system, what changes the information and where information is stored. It helps to visualize the information flows within a system.

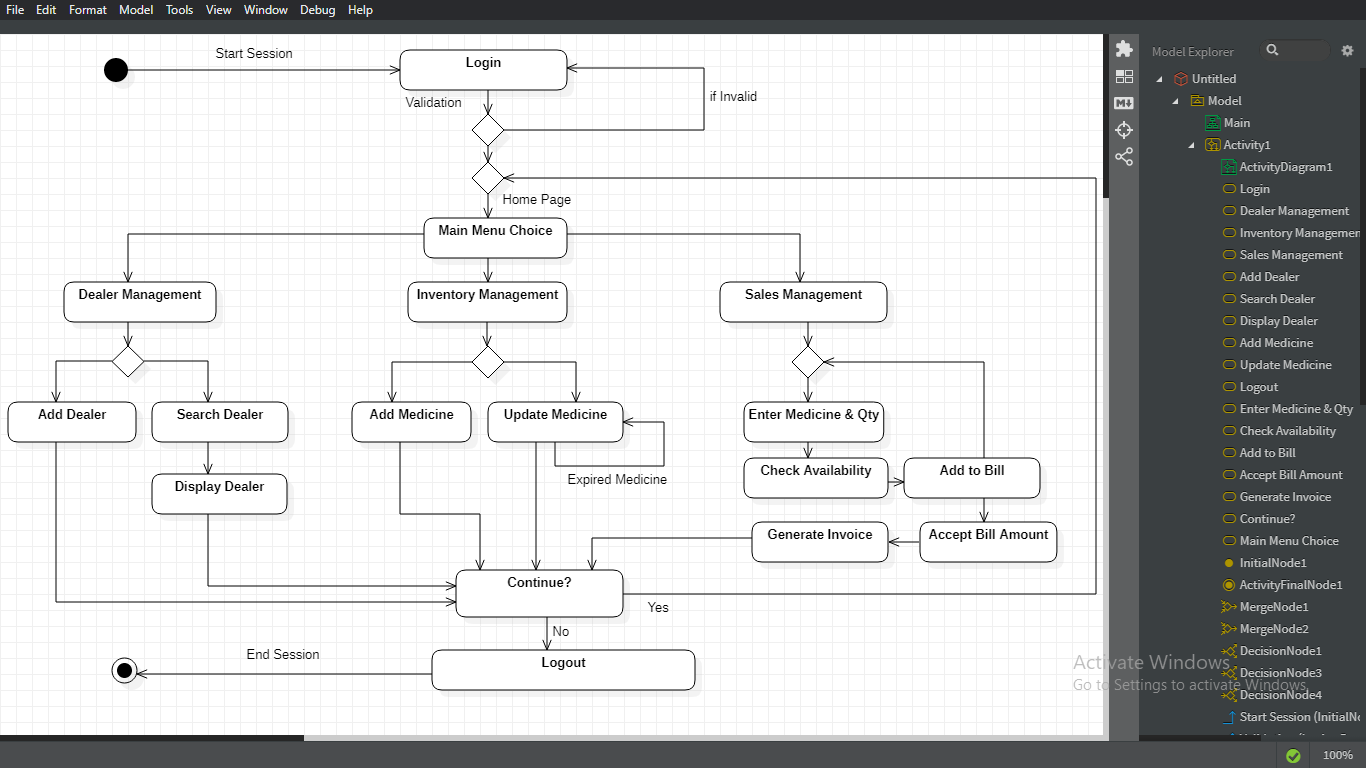
However, it does not paint the entire picture of the project and does not depict all the possible functionalities. Hence, we go for the Level 1 Diagram.

2.LEVEL 1 DIAGRAM



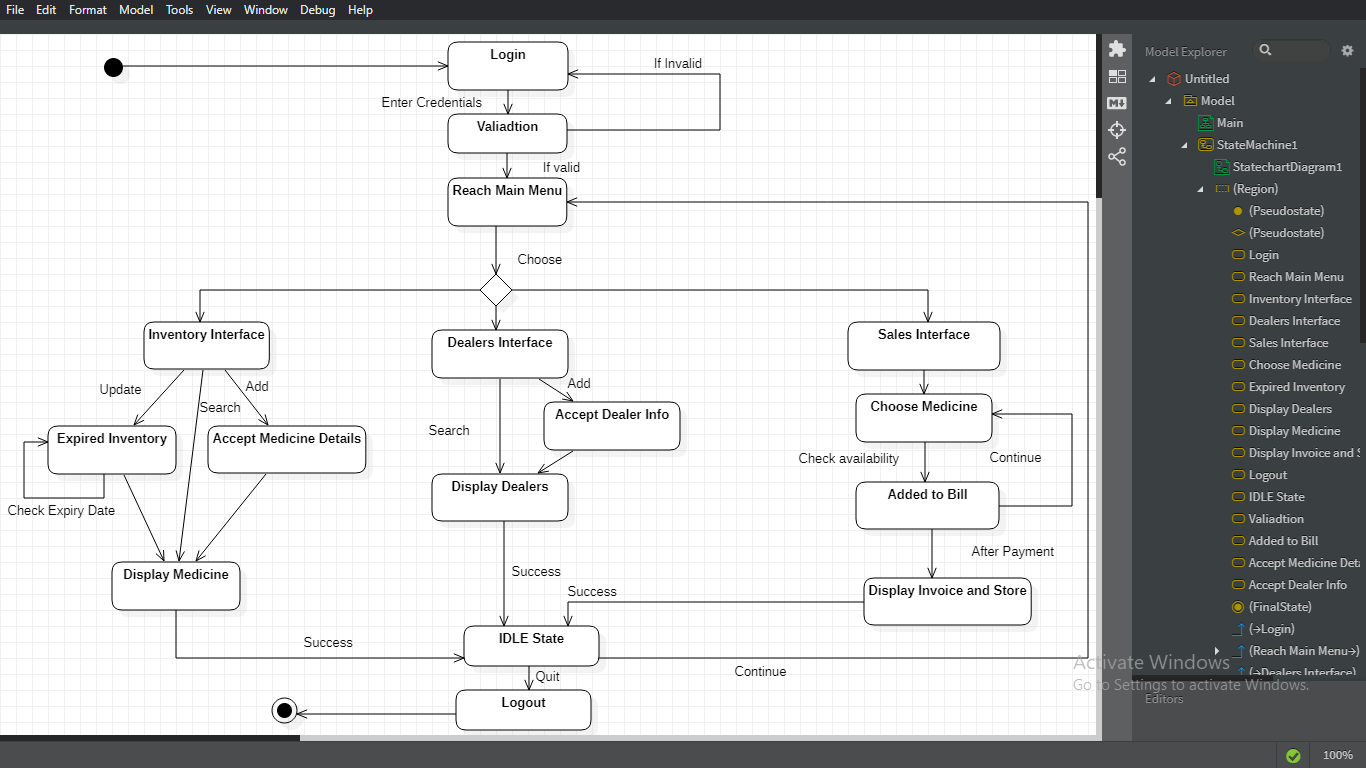
DFD Level 1 provides a more detailed breakout of pieces of the Level 0 Diagram. Here we will highlight the main functions carried out by the system, as you break down the high-level process of the Diagram into its subprocesses.

**#8 Activity Diagram**



The above diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We are able understand the activity flow of a system, describe the sequence from one activity to another and the parallel, branched, and concurrent flow within the system. Through this diagram we can view the flow right from the Login Page to Main Menu. After choosing which management the Pharmacist wants, we delve into the different possible operations to be performed. After which we can either continue or quit the system and Logout.

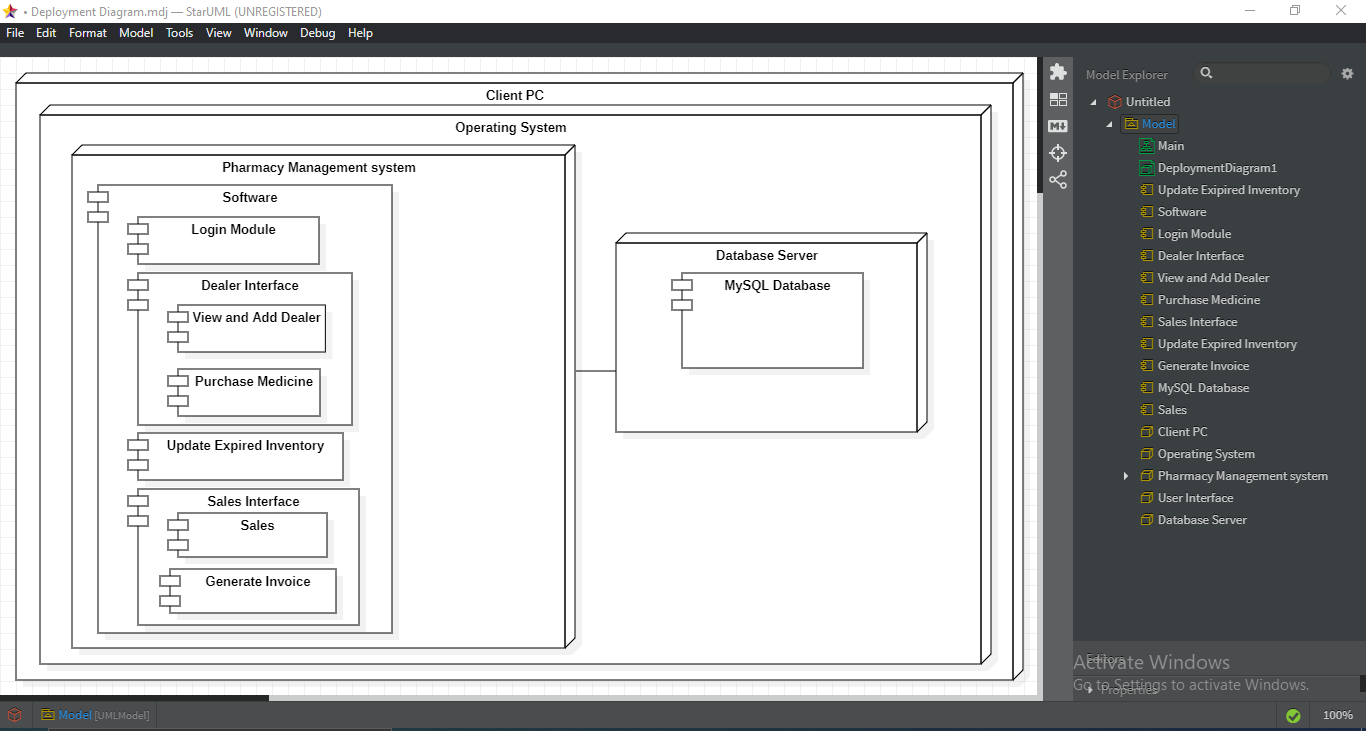
**#9 State Diagram**



The above diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists, and it changes when some event is triggered. State chart diagram helps model the lifetime of an object from creation to termination.

We first begin by identifying all the states possible in the system and find how the different events trigger those states. For example, after reaching the Main Menu we choose a particular interface. After activity is performed the system reaches Idle state when it can either end or continue and reach back to Main Menu again.

**#10 Deployment Diagram**

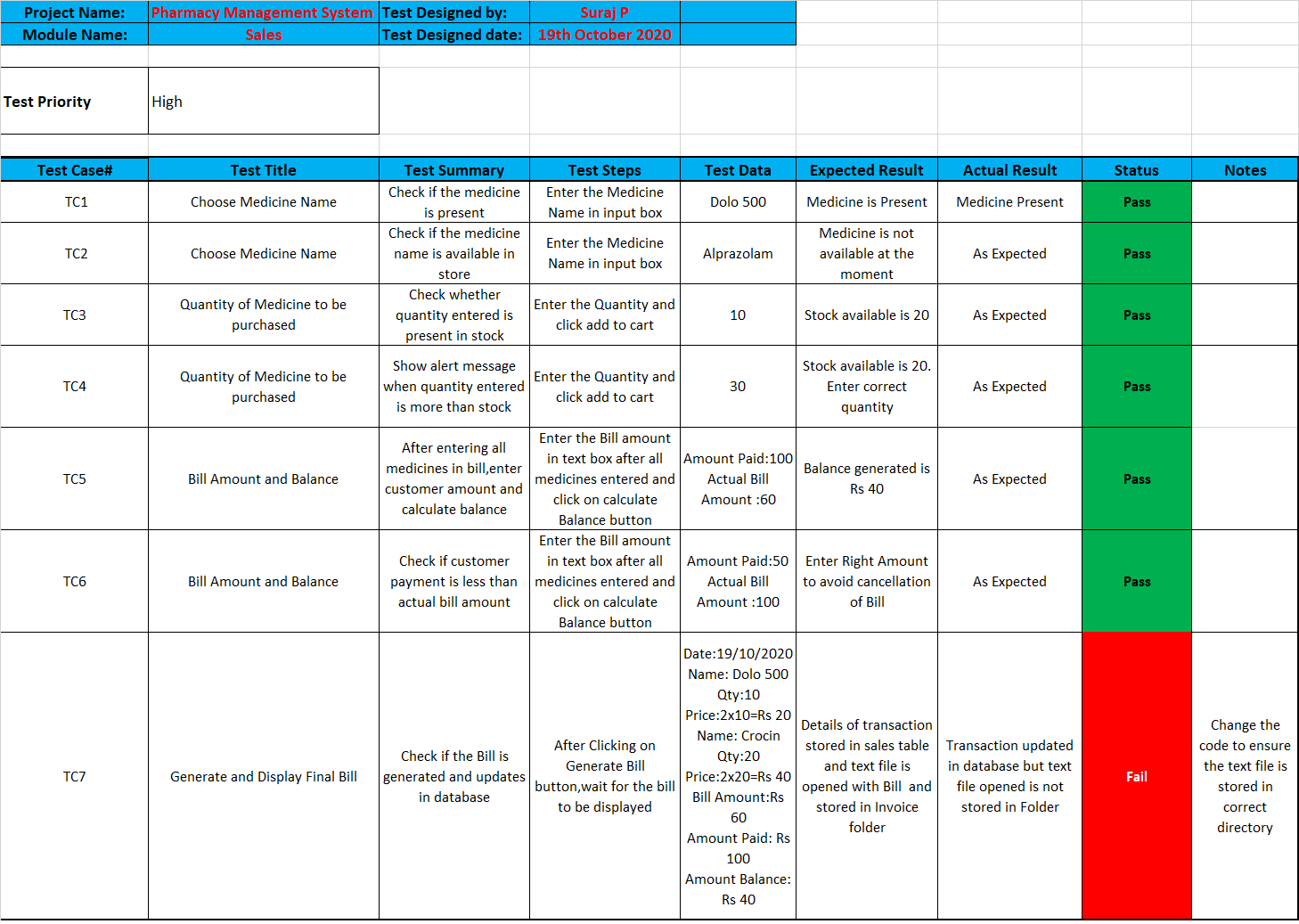


Hence the deployment diagrams are used for describing the hardware components, where software components are deployed. It shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.

Thus, it provides an overview of the entire project and concludes the UML diagrams of this project. We then move onto the testing of the actual code.

**#11 Code Test Case**

Module 1



Module 2

****

**CONCLUSION**

Through the lens of this project, Pharmacy Management System, I have truly understood the need to visualize and model a software before its actual implementation. It helps differentiate between code and an actual software. Through the SRS Document we can fully realize the actual requirements of the user and create a well-documented software. The UML diagrams help analyze the interactions between different modules of the project, the flow of the data, the states of the system and how to finally integrate the software and the corresponding database with the hardware.

Hence at the end of this course I was able to fully understand how to create and implement a software project successfully.