PHARMACY MANAGEMENT SYSTEM

PROJECT REPORT

18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

HARISH G (RA2111026010284)

MANISHA MANOJ (RA2111026010274)

Under the guidance of

Dr. M FERNI UKRIT

Associate Professor

Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**SMART FARM MONITORING SYSTEM"** is the bonafide work of **ABHISHEK MATHUR (RA2111026010100) ANURADHA KRISHNAN (RA2111026010134)** who undertook the task of completing the project within the allotted time.

Signature of the Guide
Dr. M. Ferni Ukrit
Associate Professor
Department of CINTEL,
SRM Institute of Science and Technology

Signature of the II Year Academic Advisor
Dr. N. Arivazhagan
Professor and Head
Department of CINTEL
SRM Institute of Science and Technology

About the course:-

18CSC202J/ 8AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives:

The student should be made to:

- Learn the basics of OOP concepts in C++
- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques

Course Learning Rationale (CLR): The purpose of learning this course is to:

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

Course Learning Outcomes (CLO): At the end of this course, learners will be able to:

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using standard templates
4. Construct programs using exceptional handling and collections
5. Create UML component diagram and deployment diagram
6. Create programs using object oriented approach and design methodologies

Table 1: Rubrics for Laboratory Exercises

(Internal Mark Splitup:- As per Curriculum)

CLAP-1	5=(2(E-lab Completion) + 2(Simple Exercises)(from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5=(2.0(E-lab Completion)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5=(2.0(E-lab Completion(80 Pgms)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	**2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program) **2 Mark -** Code to UML conversion GCR Exercises **3.5 Mark - Hacker Rank** Coding challenge completion
CLAP-4	5= 3 (Model Practical) + 2(Oral Viva)	• **3 Mark** – Model Test • **2 Mark** – Oral Viva
Total	25	

COURSE ASSESSMENT PLAN FOR OODP LAB

S.No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3,	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given
13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10
15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

LIST OF EXPERIMNENTS FOR UML DESIGN AND MODELLING:

To develop a mini-project by following the exercises listed below.

1. To develop a problem statement.
2. Identify Use Cases and develop the Use Case model.
3. Identify the conceptual classes and develop a domain model with UML Class diagram.
4. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.
5. Draw relevant state charts and activity diagrams.
6. Identify the User Interface, Domain objects, and Technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools for UML:

StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

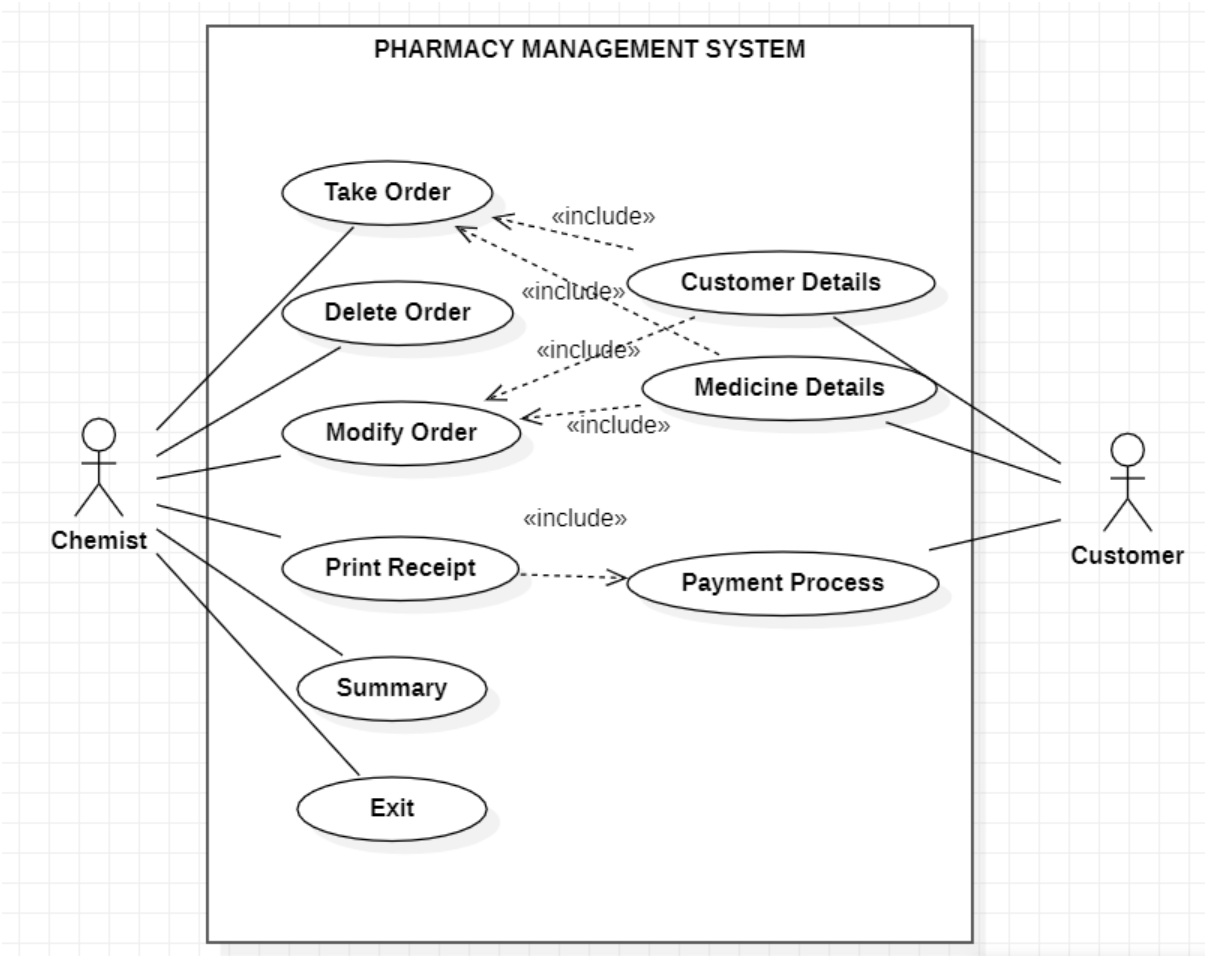
Table of Contents

ABSTRACT

Online Pharmacy Management System is developed to automate the existing system of manually maintained records of the patients, medicines, doctors etc. This application can be used by any user to automate the process of manually purchasing medicines and seeking the doctor advice etc.The main aim of developing this application is to supply the medicines all over the country by just a single click and to reduce the time consumption. Online pharmacy is a web-based application. The user can post requirement for medicine. User can purchase medicine online. Medicine is provided at your doorstep by the nearest associate store. As per the prescription, the user can search medicine and useful information. This application also helps in getting consultation from the doctor in Online, without visit doctor's place. This application provides user login to the customer. And admin can get the all expired medicines information and he can able to see all orders information of clients. The purpose of this Pharmacy Management System project is to improve the maintenance and manipulation of the drugs in the medicals. The pharmacy management system will be used to minimize the time and resource by maintaining the details of the drug systemically so that the data can be used in possible quickest time. While the resource which is minimized are workforce, money, papers, etc. The system is user-friendly and will help the pharmacist. This Pharmacy Management System will reduce the burden on pharmacist and will make the system efficient by providing the more accurate details about drugs in the medical. The pharmacy management system is intended to increase accuracy, safety, and efficiency in the pharmacy. It is a computer-based system that stores important information and makes medical stores run better.

MODULE DESCRIPTION

USE CASE DIAGRAM



A diagram that shows a set of use cases and actors and their relationships. Use cases represent system functionality, the requirements of the system from the user's perspective.

Use Case is a description of a set of sequences of actions, including variants, that system performs that yields an observable value to an actor.
Actors are the people or systems that provide or receive information from the system; they are among the stakeholders of a system.

include

• Specifies that the source use case explicitly incorporates the behaviour of another use case at a location specified by the source.

extend

• Specifies that the target use case extends the behaviour of the source.

SUMMARY

• The purpose of use case diagram is to capture the dynamic aspect

of a system.
• Used for describing a set of user **scenarios**
• Mainly used for capturing user requirements
• Work like a **contract** between end user and software developers.

Here, there are two actors i.e., Chemist and Customer. These actors provide or receive information from the system. Let's see the set of sequences and their operations one by one:

Take Order:

This use case is responsible for taking order from the customer. The Chemist first enters common details like Order Number, Date and then he enters required customer details like Name. Then Chemist chooses the required medicine by customer and number of tablets from the list.

Delete Order:

This use case is responsible for deleting order. By any chance, if the order is cancelled or returned because of any reasons this option is used to remove that order from the database.

Modify Order:

This use case is responsible for modification or order list. In case any mistakes in entering name, date or medicine type, we can change in modify order option.

Print Receipt:

After choosing the required medicine, it is requested to print the receipt. The receipt includes name of the medicine, quantity of medicine and its cost and atlast total cost. This option also includes the payment process, here the customer can pay the amount cost of the purchase.

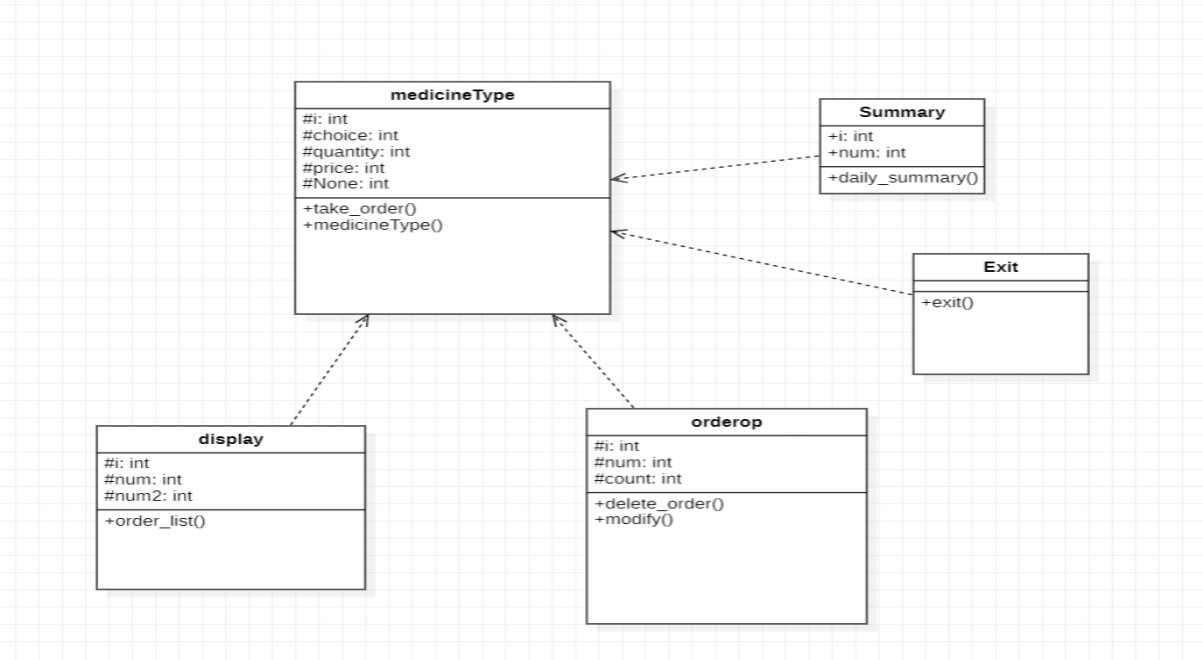
Summary:

Summary option displays all purchase happened that day. It shows order number, medicines purchased, and total purchase that day. This makes easy work for chemist to calculate daily sales.

Exit:

This is just a simple option to leave the program. Once the use of the program is completed, you can use this option to leave the program Idle.

CLASS DIAGRAM



•It is the most common diagram type for software documentation
•Class diagrams contain classes with their attributes (data members) and their behaviours (member functions)
•Each class has 3 fields:

- Class Name
- Attributes
- Behaviours

An *attribute* is a named property of a class that describes the object being modelled. In the class diagram, attributes appear in the second compartment just below the name-compartment. *Operations* describe the class behaviour and appear in the third compartment.

Basic components of a class diagram
The standard class diagram is composed of three sections:
Upper section: Contains the name of the class. This section is always required, to know whether it represents the classifier or an object.
Middle section: Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.

Bottom section: Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

Class: MedicineType

This is like a super class here. This class is responsible for getting inputs. This class has the functions which get the order from the customer. This input includes order number, customer name, number of medicines they purchase, details of medicine like name, quantity, cost per pill etc.

The data member "choice" is used to get choice of the chemist to run the program, whether to get input, print receipt, summary, delete or modify the order. "Quantity" and "price" data members used for the calculation of amount the customer need to pay.

Take_order() function is where everything is defined to get the order from the customer.

Class: orderop

This class holds two functions of the project which plays an optional but important role in the Pharmacy Management System. Firstly, delete_order() function is used to delete the order in case if the order is cancelled or returned because of any reasons this option is used to remove that order from the database.

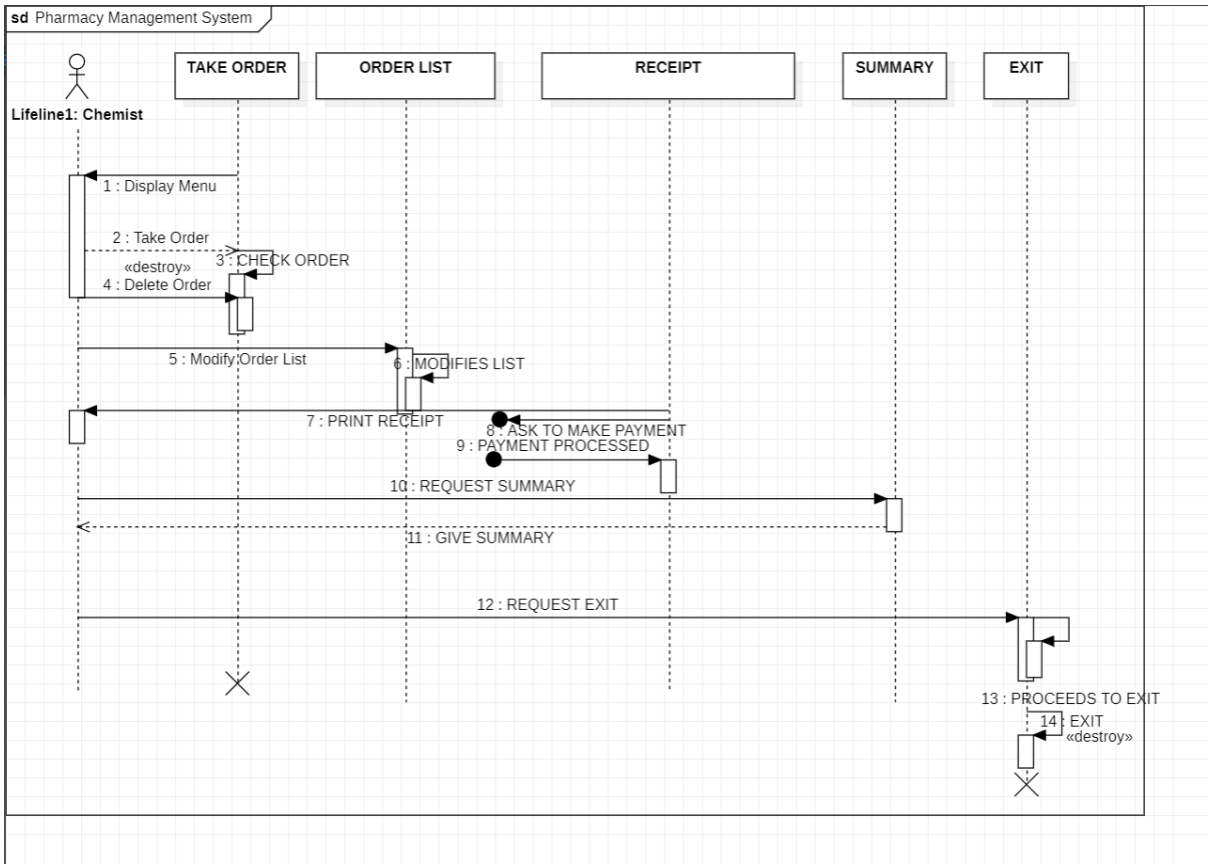
Secondly, modify() function is responsible for modification or order list. In case any mistakes in entering name, date or medicine type, we can change in modify order option.

Class: display

This class is declared to print the order list which is entered by the chemist according to the requirements of the customer. By entering the order number, it can print the receipt of the particular order along with the customer details. It also prints the total bill amount so that the customer can proceed with the payment option. Chemist need to enter the amount, the customer need to pay for the payment to succeed. If the order number entered by chemist is wrong, it will display an empty receipt with the alt text "There is no order to show, So the list is empty."

These two classes (display and orderop) shows their depencency to the class MedicineType. These classes are able to perform their functions only if the order is taken and that where the MedicineType class comes in action.

SEQUENCE DIAGRAM



A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. It emphasizes on time sequence of messages from one object to another.

This is the interactions of Chemist in Pharmacy Management.

There are 5 Objects namely :

- Take order
- Order list
- Receipt
- Summary
- Exit

The Chemist is the main actor here, which interacts with the objects.

Let’s see the interaction of objects with the flow of the pharmacy management system.

TAKE ORDER ----→ Chemist:

Firstly, the Take Order object based on taking order as the name says. It displays the medicine available in the pharmacy.

Chemist ----→ TAKE ORDER:

The Chemist then enteres the details of customer, date, and required medicine for the customer. Also, if order need to be deleted. take_order() function is used here.

TAKE ORDER:

That object self-checks the order with the given number of medicines and also calculates the sum of the purchase.

Chemist ----→ ORDER LIST:

The interactions with the order list is to modify the order list. The order list modification includes modification of name, date, medicine. Modify() function is used to modify the list.

MODIFY LIST:

After the modified details entered by chemist the list should be modified. The list then removes the previous inputs and modifies it with the new inputs.

RECEIPT:

This object is responsible to print the receipt. display() function is used to display the receipt of the purchase.

PAYMENT:

The payment done by customer is represented here as a hidden messages that is lost and found messages.

Chemist ----→ SUMMARY:

Here the chemist requests to print summary of the sales. summary() function is used to print the summary of the sales.

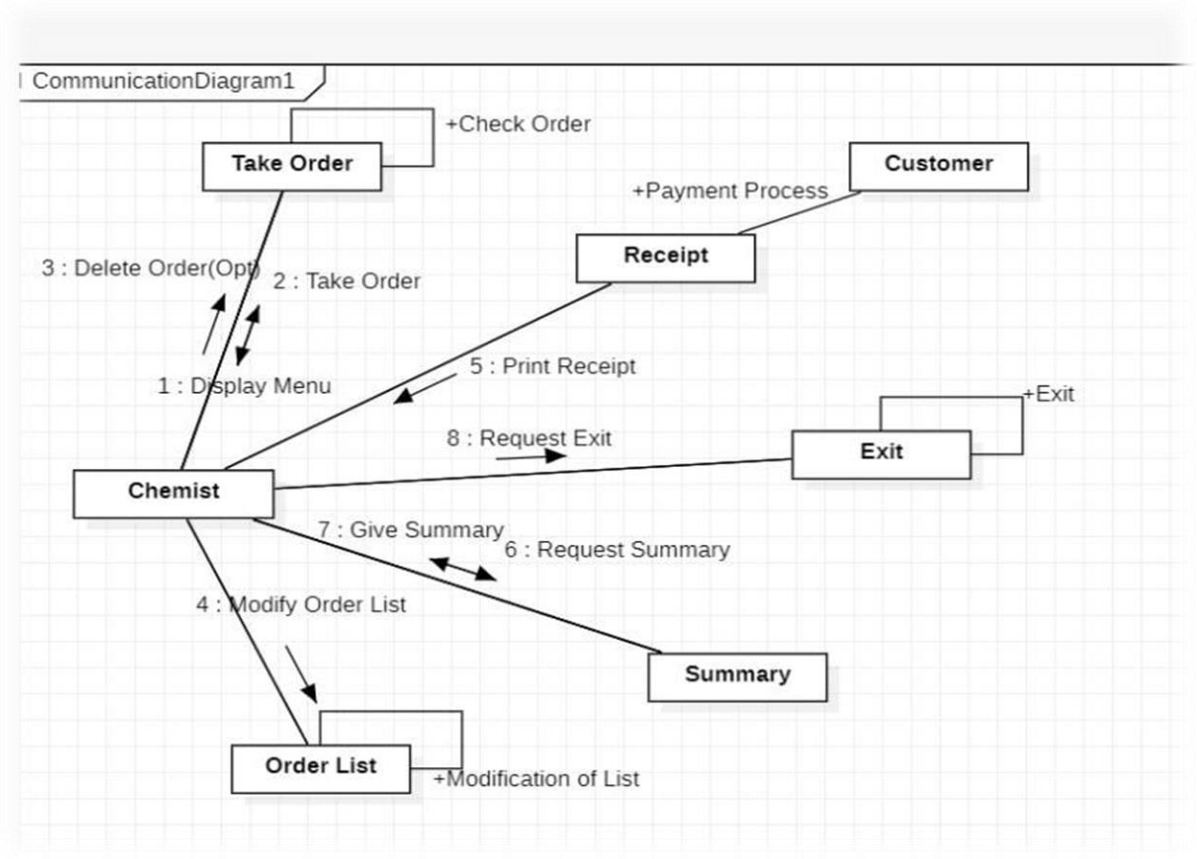
SUMMARY----→ Chemist:

It prints the summary of the details of the sales entered by the chemist.

Chemist----→EXIT:

Here, the chemist requests to exit the program. Exit() function is used to quit from the program.

COMMUNICATION DIAGRAM



Communication diagrams, formerly known as collaboration diagrams, but they focus more on the relationships of objects—how they associate and connect through messages in a sequence rather than interactions.

This report explains the interactions of a chemist in Pharmacy Mangament .

There are 5 Objects namely :

- Take order
- Order list
- Receipt
- Summary
- Exit

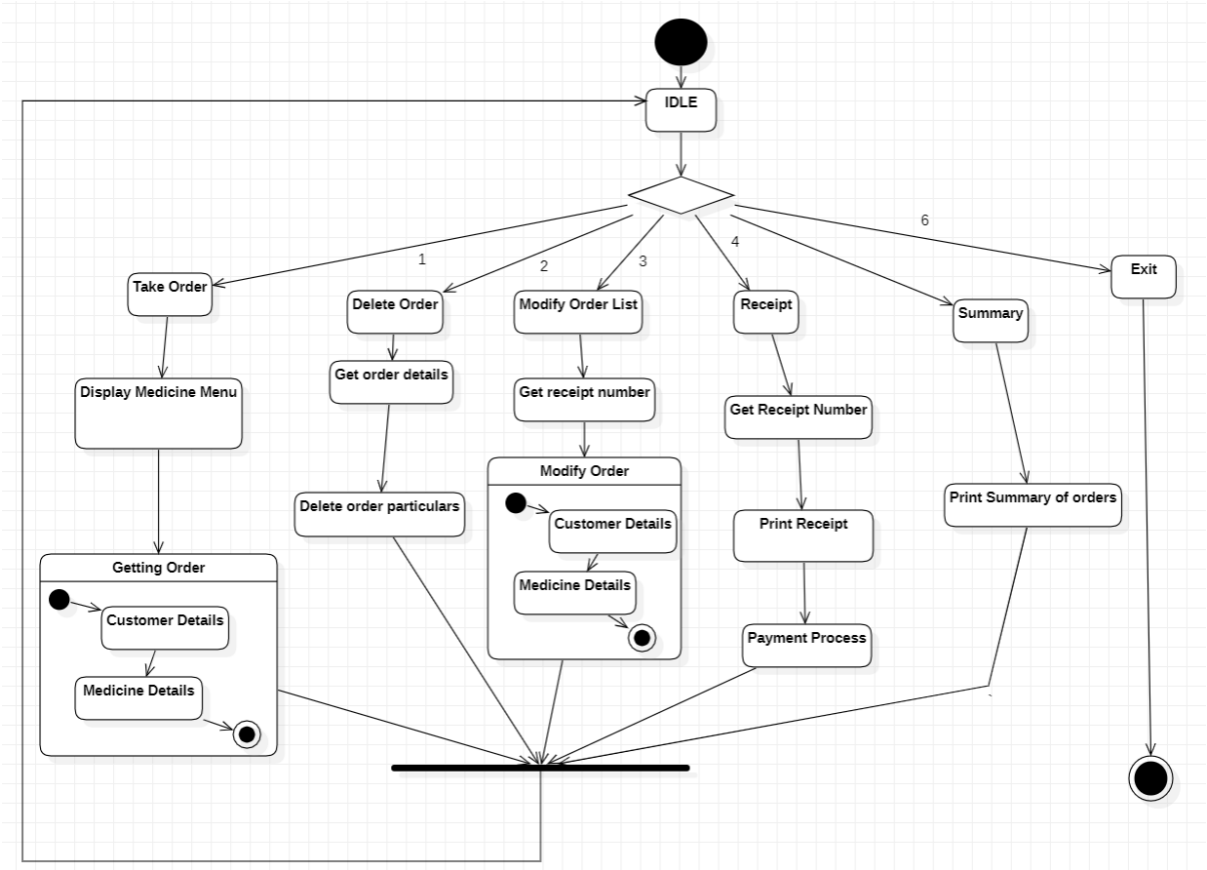
The Chemist and The Customer are Actors here . He interacts with all the objects here . Following are the operations followed in this Collaboration Diagram:

1. **Take Order to Chemist**-The display Menu () displays the list of medicines to the chemist .
2. **Chemist to Take order** –The chemist sends a reply message[shown in dotted lines] to the object[Take Order].
3. **Take Order to Chemist** – Chemist sends an optional message to the Object to delete order.
4. **Chemist to Order List** – The Chemist sends instructions to the Object to Modify the Order list ().
5. **Order list to Order list** – This object sends a self -message to itself after it gets an instruction from the chemist to modify the list.
6. **Receipt to Customer** – The Object sends a message to the customer to make payment.
7. **Customer to Receipt** – The Customer make the payment and sends a message to the object to print the Receipt.
8. **Receipt to Chemist** – Here a Reverse message is sent to the chemist from the object to print receipt ().
9. **Chemist to Summary** – The Chemist requests a summary from the system.
10. **Summary to Chemist** – The Object provides the Summary to the Chemist.
11. **Chemist to Exit**- The Chemist sends a message to the Object to exit from the system.

12. **Exit to Exit** – The object sends a self-message to itself to exit

from the system.

STATE CHART DIAGRAM



• A state diagram is used to represent the condition of the system or part of the system at finite instances of time.

• It's a behavioral diagram and it represents the behavior using finite state transitions.

• State diagrams are also referred to as State machines and state-chart Diagrams.

So the main usages can be described as:

• To model object states of a system.

• To model reactive system. Reactive system consists of reactive

objects.

• To identify events responsible for state changes.

• Forward and reverse engineering.

At first the program stays IDLE, displaying the menu for the operations. The program starts when the choice is entered. The choices are numbered as 1 to 6.

1. If the choice is 1 the program will start getting order by the customer. It displays the available medicine with their price per pill. Chemist need to enter order number, Name of customer, Date of purchase. Then the medicine is chosen and the quantity is entered.

2. If the choice is 2 the program refers to delete the order. It requests the order number which need to be deleted. If the order number is entered, then the details of that order is permanently deleted from the database.

3. If the choice is 3, it is functioned to modify the order list. The order list can be modified of following inputs.

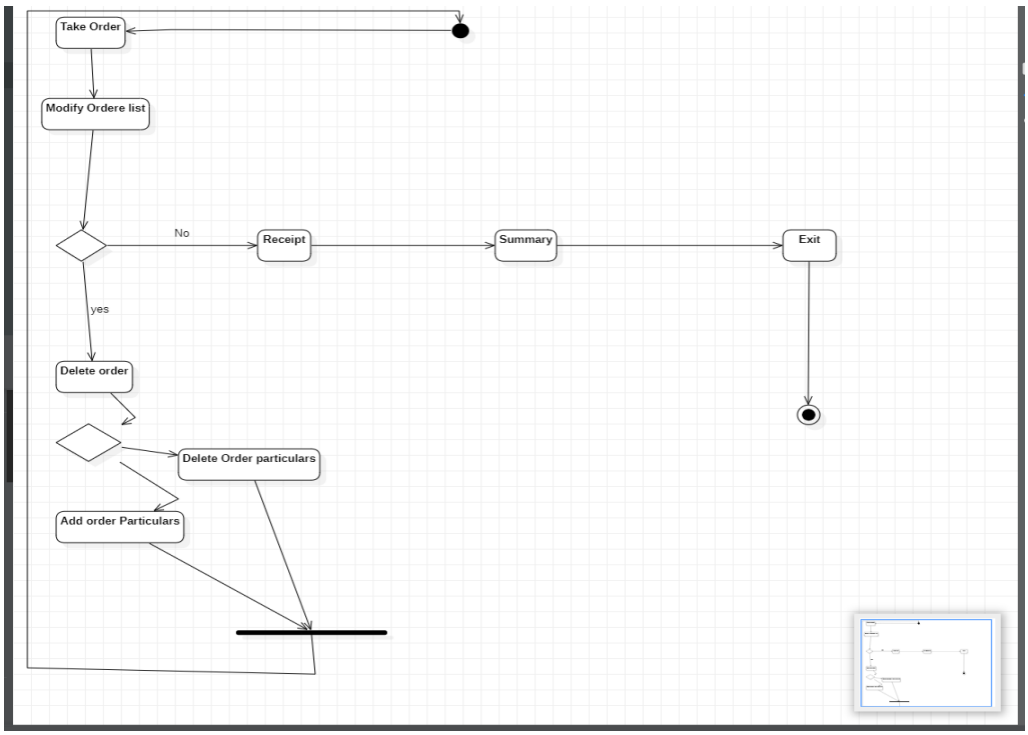
- Order number
- Name of customer
- Date of purchase
- Quantity of medicine
- Name of medicine

4. If the choice is 4, then it prints the receipt of the order and it proceeds with the payment. It print the receipt with bill number, customer details and medicine details and total cost of the purchase. After printing the receipt, the customer can pay the amount of the bill.

5. If the choice is 5, it prints the summary of the sales happened on that particular day. It shows order number, medicines purchased, and total purchase that day. This makes easy work for chemist to calculate daily sales.

6. If the choice is 6, This is just a simple option to leave the program. Once the use of the program is completed, you can use this option to leave the program IDLE.

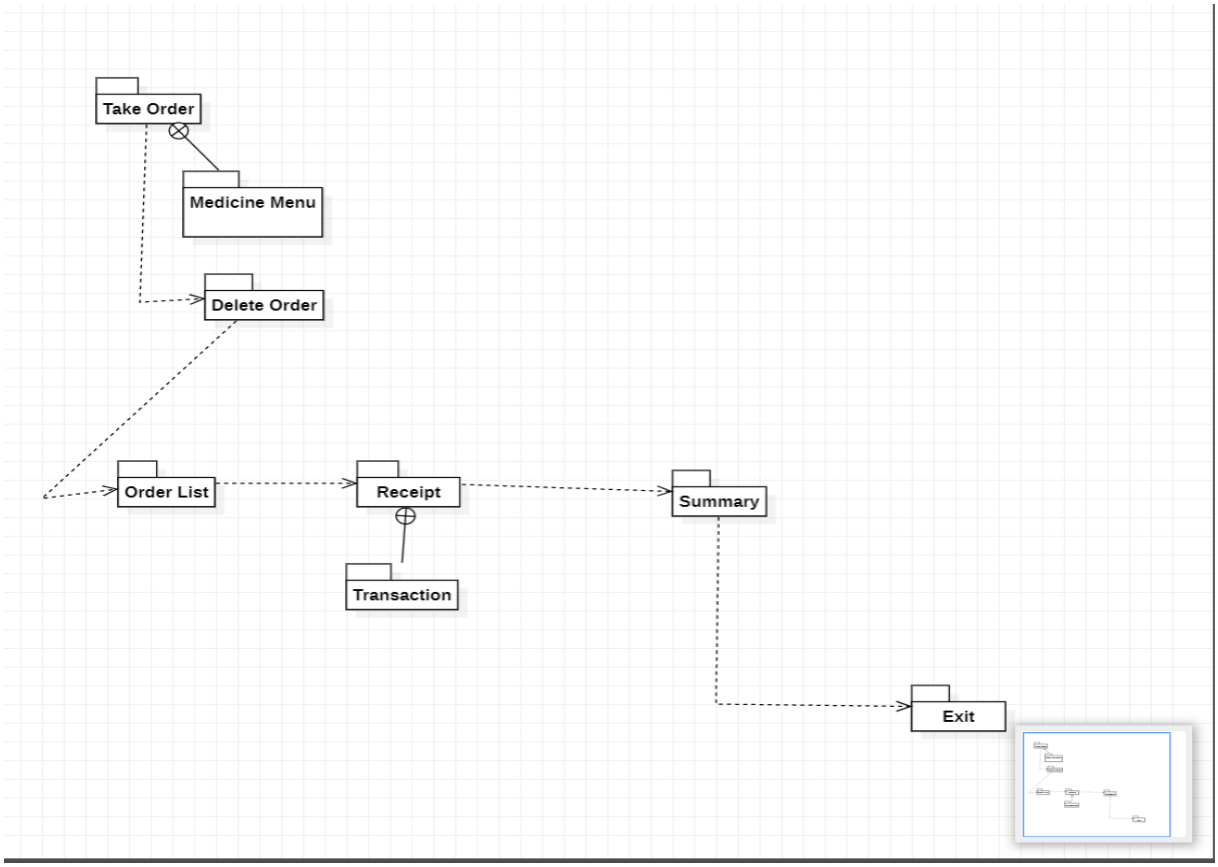
ACTIVITY DIAGRAM



Explanation:

Here the diagram represents the chemist as the initial state. The chemist heads to the Take order. Once the order is Order is taken if the chemist wishes to modify the order then it leads to the delete order section wherein its again has a decision box to add or delete the order particulars. once the modification is done, we have used a fork to join the boxes and then it leads to the chemist. Then again it continues, later the chemist heads to take a receipt. The customer receives a summary and the line leads to exit.

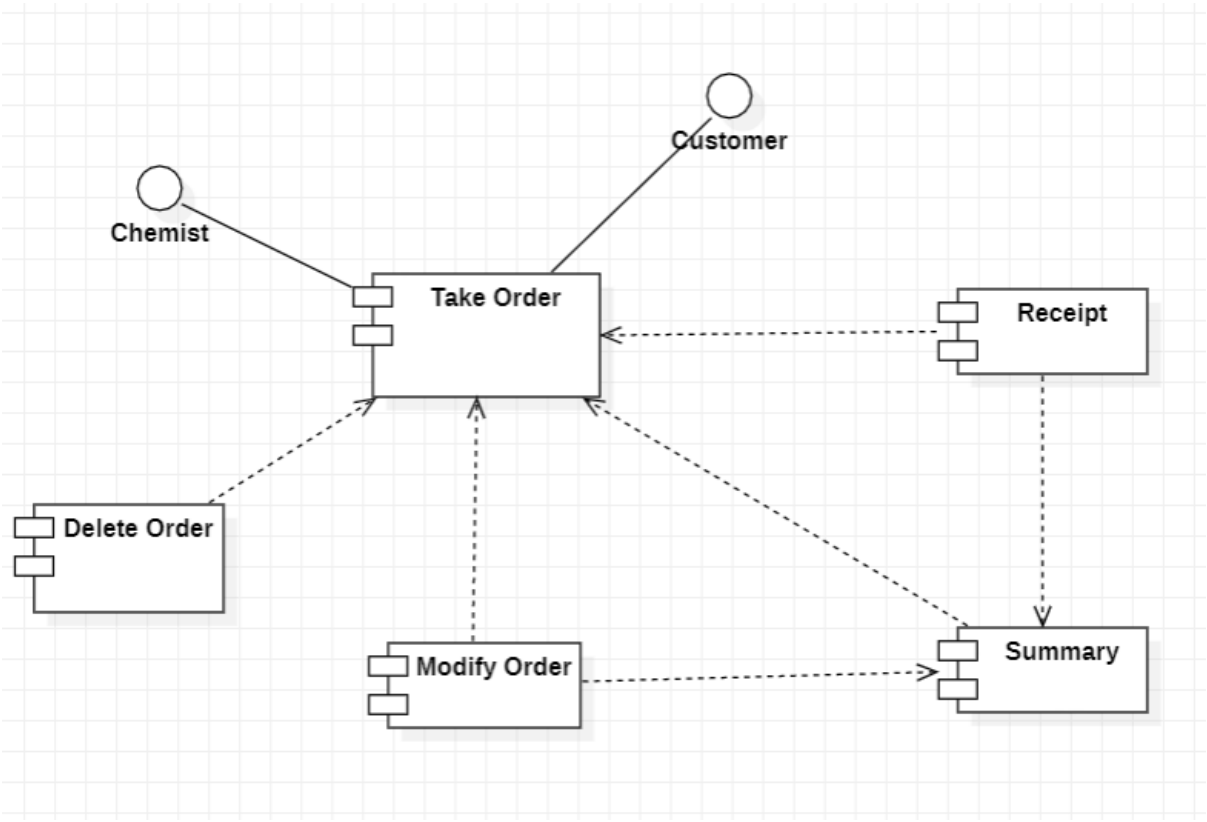
PACKAGE DIAGRAM



Explanation:

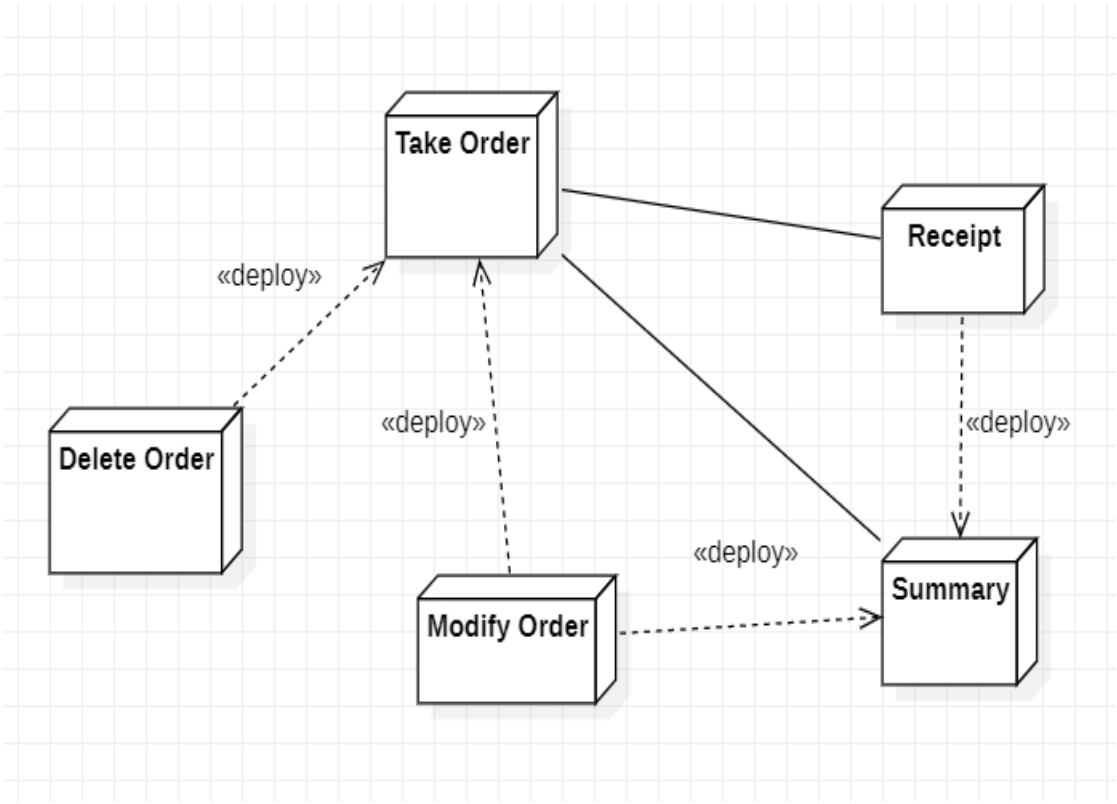
The diagram represents packages . initially the Take order has a sub package called Medicine Menu . since the chemist has to go through it . Then the delete order to do certain modifications. Once the list has been modified the line leads to Order list. Furthur the list generates a receipt . Under the receipt package a transaction sub package has been included to show the process to generate a receipt . Then summary is given and the process leads to an exit .

COMPONENT DIAGRAM



It has two interfaces Chemist and Customer. They both involve in the take order component, therefore the interface realization is involved. The other components Delete Order, Modify Order, Summary and Receipt and dependent on the take order component. Delete order gets the order details to delete. Modify the order in modify order like changing medicines, name, date etc. Summary provides the summary of sales happened in that day. Receipt involves printing of the receipt and involves the transaction process.

DEPLOYMENT DIAGRAM



The take order node is involved as the superior node. The other nodes Delete Order, Modify Order, Summary and Receipt and dependent on the take order component. Delete order gets the order details to delete. Modify the order in modify order like changing medicines, name, date etc. Summary provides the summary of sales happened in that day. Receipt involves printing of the receipt and involves the transaction process. The modify order has a deployed to take order and summary. The receipt is deployed to the summary.

CONCLUSION

Pharmacy management system is actually a software which handle the essential data and save the data and actually about the database of a pharmacy and its management. This software helps in effectively management of the pharmaceutical store or shop. It provides the statistics about medicine or drugs which are in stocks which data can also be updated and edited. It works as per the requirement of the user and have options accordingly. It allows user to enter manufacturing as well as the expiry date of medicine placing in stock and for sales transaction. This software also has ability to print reports and receipts etc.

Effective implementation of this software will take care of the basic requirements of the pharmacy management system because it is capable of providing easy and effective storage of information related to activities happening in the stipulated area. With these, the objectives of the system design will be achieved.

REFERENCES

https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm

https://www.javatpoint.com/uml-diagrams

https://docs.staruml.io/working-with-uml-diagrams/use-casediagram