PROJECT REPORT

***Submitted by***

## DIVIJA A [Reg. No.: RA2211003010556]

## SUBASHRI VAZHAI PARATHAN [Reg. No.: RA2211003010559]

***Under the Guidance of***

**Dr. MURALIDHARAN**

**Associate Professor, Department of Computing Technologies**

***In partial satisfaction of the requirements for the degree of***

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING**



**SCHOOL OF COMPUTING**

**COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR - 603203**

**MAY 2023**



SRM INSTITUTION OF SCIENCE AND TECHNOLOGY KATTANKULATHUR-603203

**BONAFIDE CERTIFICATE**

Certified that this Project Report titled **ONLINE SHOPPING SYSTEM** is the bonafide work done by Divija(556) and Subashri vazhai parathan(559)who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work

|  |  |
| --- | --- |
| **SIGNATURE** | **SIGNATURE** |
| **Dr.Muralidharan** | **Dr.M.Pushpalatha** |
| **OODP – Course Faculty** | Professor & Head |
| Associate Professor | Department of Computing Technologies |
| Department of Computing Technologies | School of Computing |
| SRMIST | SRMIST |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No** | **CONTENTS** | **PAGE NO** |
| 1. | Problem Statement | 4 |
| 2. | Modules of Project | **5** |
| 3. | Diagrams |  |
|  | a. Use case Diagram | **6** |
|  | b. Class Diagram | **7** |
|  | c. Deployment Diagram | **8** |
|  | d. Collaboration Diagram | **9** |
|  | e. Sequence Diagram | **10** |
|  | f. Package Diagram | **11** |
|  | g. Component Diagram | **12** |
|  | h. Activity Diagram | **13** |
|  | i. State Chart Diagram | **14** |
| 4. | Code/Output Screenshots | **16 , 29** |
| 5. | Conclusion and Results | **31** |
| 6. | References | **32** |

# PROBLEM STATEMENT

Current online shopping systems often suffer from poor user experience and inefficient workflows, leading to frustration for users and lost sales for businesses. Our project aims to address these issues by developing an innovative shopping platform that streamlines the purchase process and provides a seamless, intuitive experience for users. Our objectives include improving user satisfaction, increasing conversion rates, and reducing cart abandonment. However, we face challenges such as limited resources and the need to balance technical complexity with ease of use. By developing a cutting-edge shopping system that meets the needs of both users and businesses, we aim to revolutionize the online shopping landscape.

# MODULES OF PROJECT

**User Management:** This module will handle user registration, login, and authentication. It will also manage user profiles, passwords, and other account-related settings.

**Product Catalogue:** This module will manage the list of products available for purchase. It will store product information such as name, description, image, price, and availability. It will also allow the user to filter and search for products.

**Shopping Cart:** This module will allow users to add products to a cart, modify the quantity of products, and remove products from the cart. It will also display the total cost of the products in the cart.

**Checkout:** This module will handle the payment process, including collecting billing and shipping information and processing payments through a payment gateway.

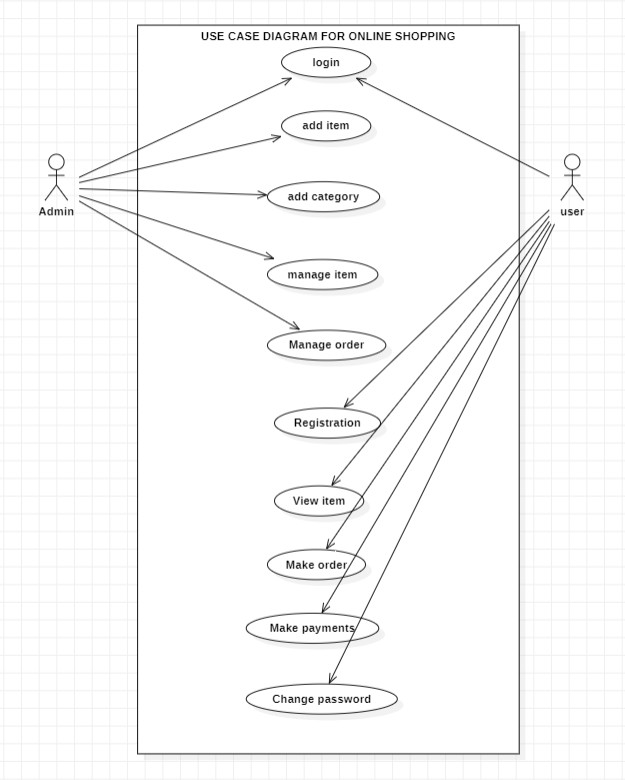
**Order Management:** This module will manage the user's order history and provide information such as order status, tracking information, and order details.

**Admin Panel:** This module will provide an interface for the site administrator to manage the product catalogue, orders, and user information. It will also allow the administrator to generate reports on sales and other metrics.

**Shipping and Logistics:** This module will manage the shipping process, including generating shipping labels, tracking packages, and handling returns and refunds.

**Customer Support:** This module will provide customer support features such as a help centre, contact forms, and live chat support. It will also manage customer feedback and reviews.

**Use Case Diagram for Online Shopping System**

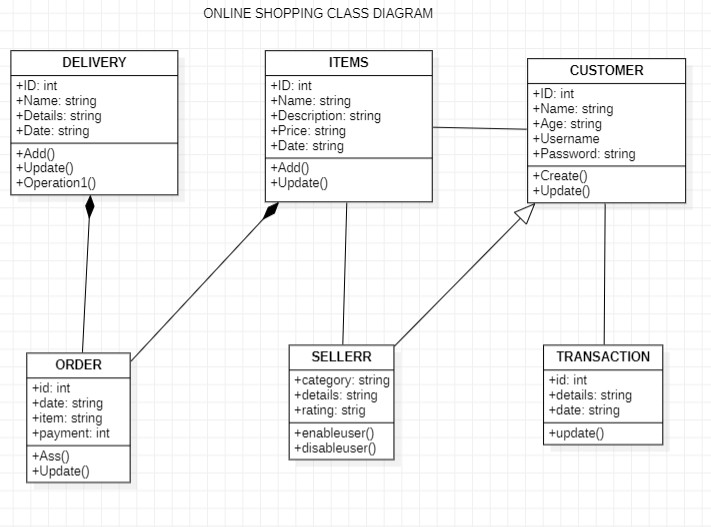


1. Actors: Actors are the entities that interact with the system. They can be human users, external systems, or other software components. In a use case diagram, actors are represented as stick figures, and their interactions with the system are represented as arrows.
2. Use Cases: Use cases are the specific tasks or functions that the system performs for its users. They describe the steps involved in completing a specific task or achieving a specific goal. Use cases are represented as ovals in a use case diagram, and the arrows connecting them to actors represent the interactions between the actors and the system.
3. Relationships: Relationships between use cases can be represented in a use case diagram

using various symbols, such as arrows and lines. These relationships help to clarify how the system functions and how different use cases are related to each other.

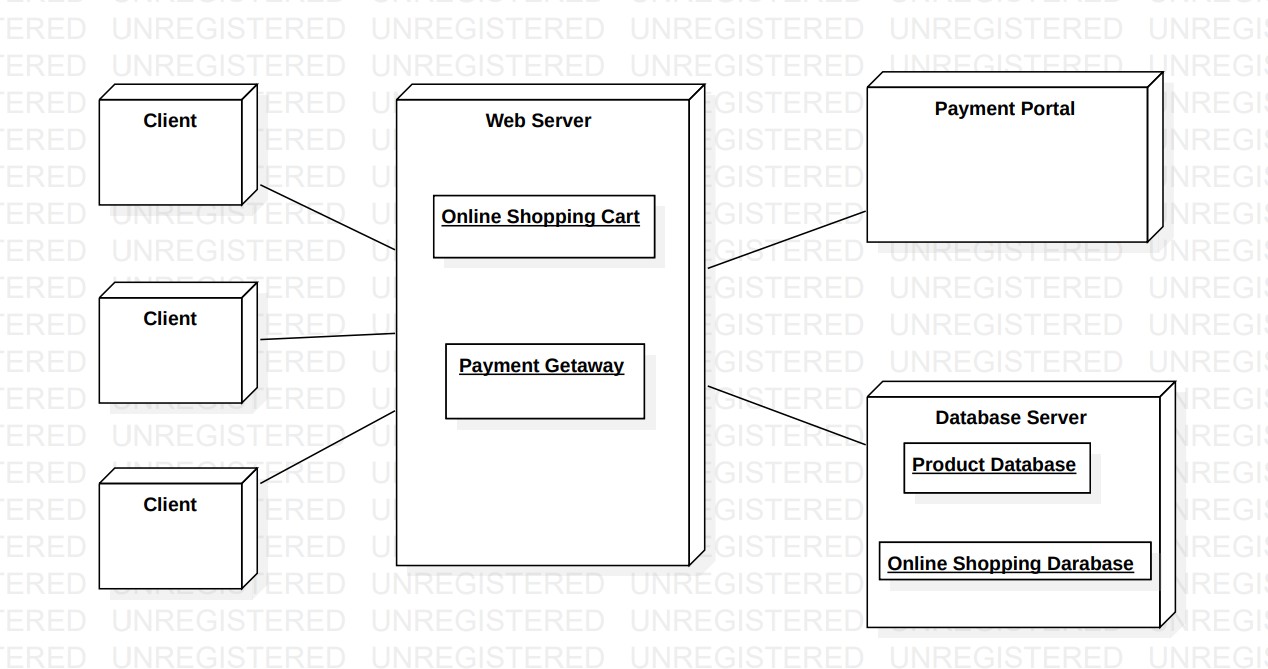
1. System Boundaries: The system boundary is the line that separates the system from the external environment. Use case diagrams should clearly show this boundary to help users understand what is within the scope of the system and what is outside it.
2. System Goals: The system goals should be clearly identified in the use case diagram. These goals help users understand what the system is designed to achieve and how it will be used to meet specific objectives. The use case diagram should clearly show how different use cases contribute to achieving these goals.

## Class Diagram for Online Shopping System



1. Classes: Classes represent the objects in a system and are typically nouns. Each class has attributes (properties) and methods (behaviors). The class name should be descriptive and reflect the purpose of the object it represents.
2. Attributes: Attributes are the properties of a class that describe its state. They can be of different types, such as integer, Boolean, string, etc. Attributes are typically represented as variables within the class.
3. Methods: Methods are the behaviors of a class that describe its actions. They define the operations that can be performed on the object represented by the class. Methods are typically represented as functions within the class.
4. Relationships: Relationships between classes are represented by different types of associations such Multiplicity.
5. Multiplicity represents the number of instances of one class that can be related to another class.as inheritance, aggregation, and composition

## Deployment Diagram for Online Shopping System

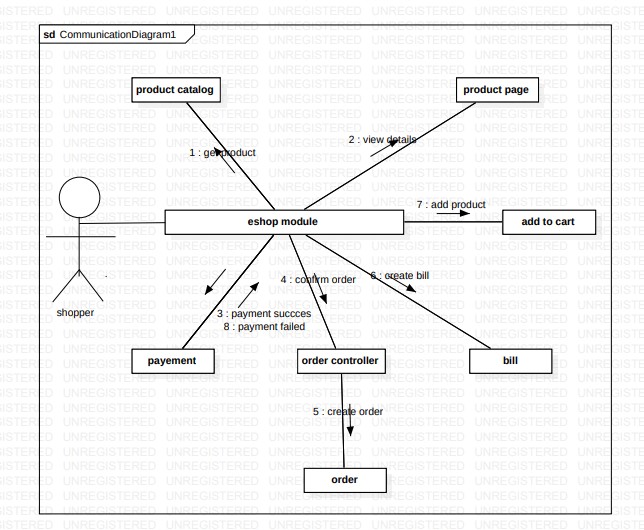


* 1. Nodes: Nodes represent the physical hardware or software components that are part of the system. They can be servers, clients, routers, or other devices. Nodes are typically represented by rectangles with the node name written inside.
  2. Components: Components represent the software modules or subsystems that run on the

nodes.

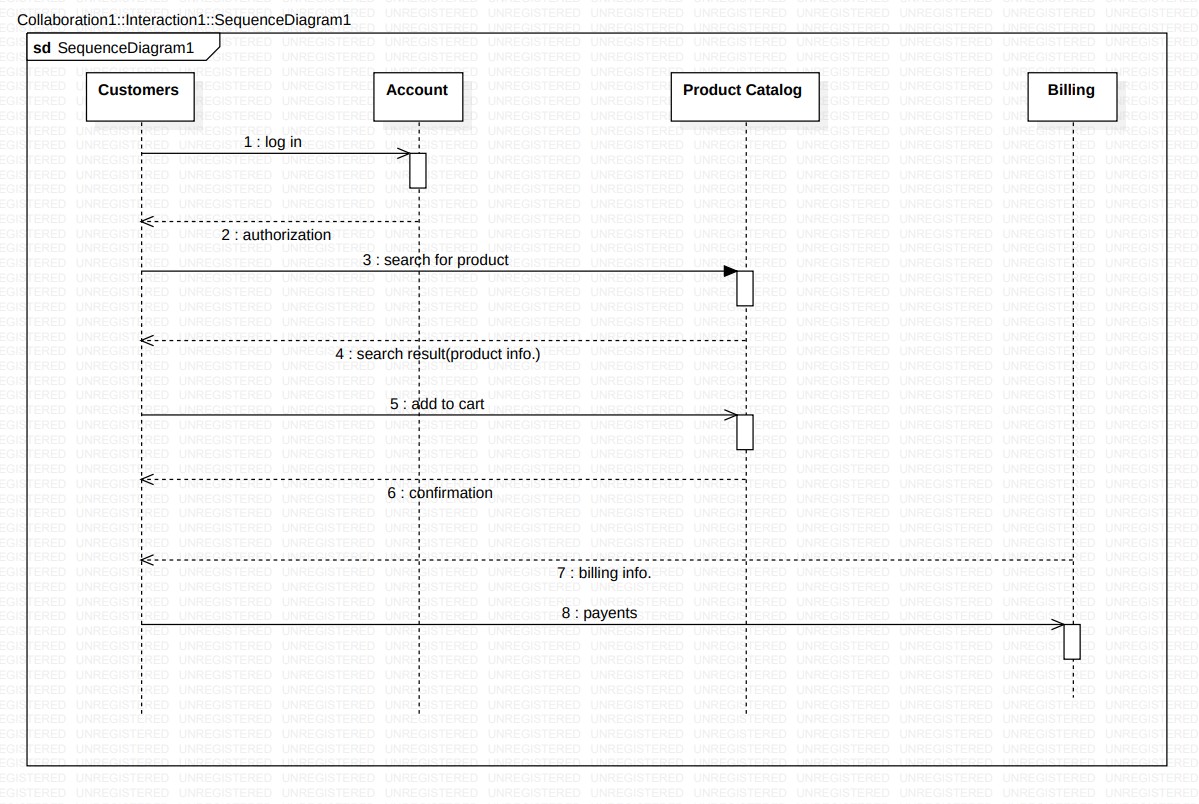
* 1. They can be applications, databases, or middleware. Components are typically represented by rectangles with the component name written inside, and they are connected to the nodes by deployment arrows.
  2. Deployment Arrows: Deployment arrows represent the deployment of components onto nodes.
  3. They indicate which components run on which nodes, and they are typically represented as arrows that point from the component to the node.

## Communication Diagram for Online Shopping System



1. Objects: Objects represent instances of classes or components in a system. They are the actors in the communication diagram, and they exchange messages to achieve a goal. Objects are typically represented as rectangles with the object name written inside.
2. Messages: Messages are the means by which objects communicate with each other. They represent a specific action or task that one object wants another object to perform.
3. Messages can be synchronous or asynchronous, and they can have parameters and return values. Messages are typically represented as arrows between objects, with the message name written above the arrow.
4. Lifelines: Lifelines represent the existence of an object during the communication process. They show the duration of an object's participation in the communication, from the start of the message exchange to the end.
5. Ordering: The ordering of messages in a communication diagram is important to show the sequence of actions that occur during the communication process

## Sequence Diagram for Online Shopping System

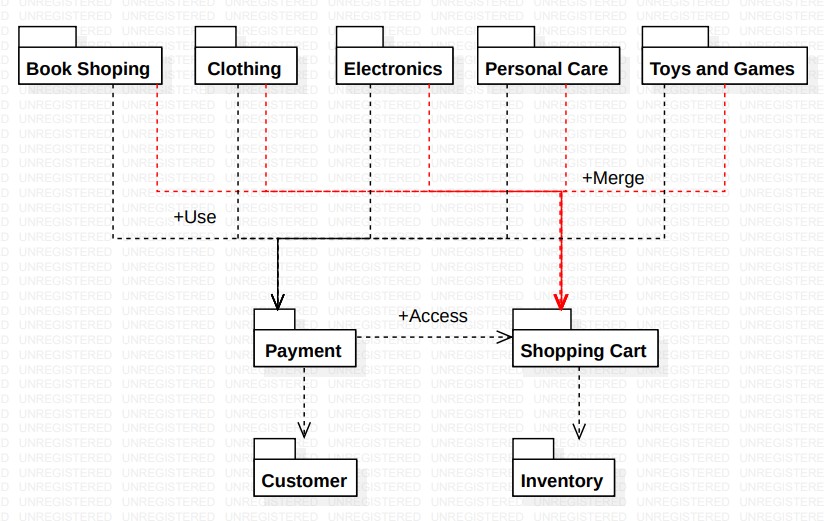


1. Lifelines: Lifelines represent the objects or components that participate in the interaction. They are depicted as vertical lines, and each lifeline represents an instance of a class or a

component in the system.

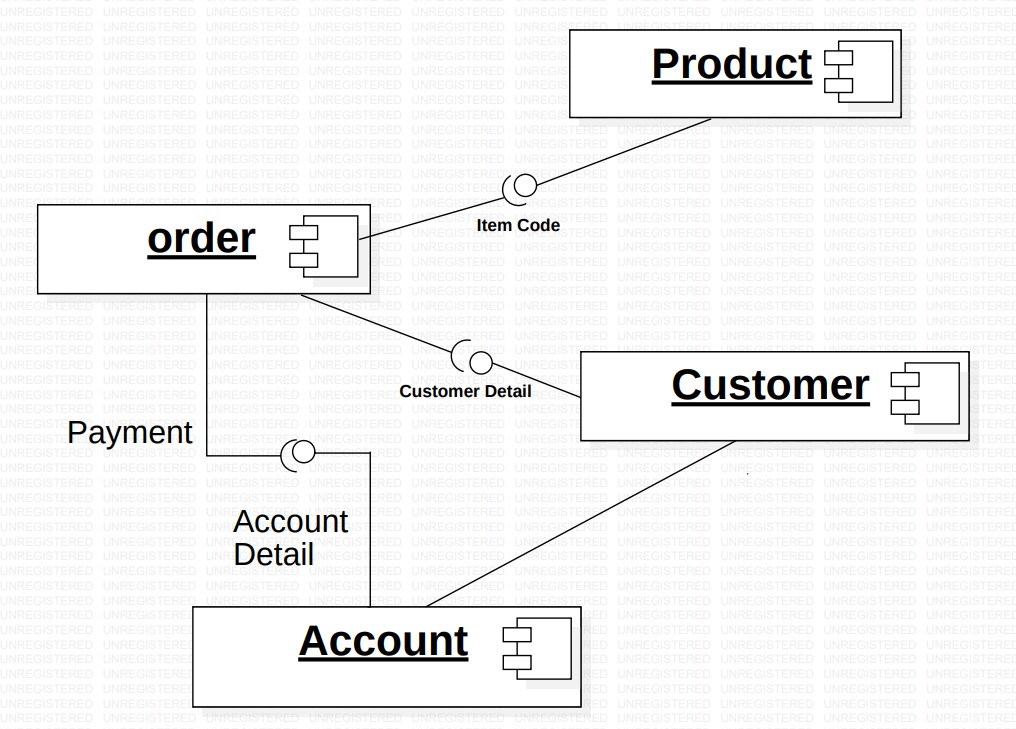
1. Messages: Messages represent the communication between the objects or components. Messages can be synchronous or asynchronous, and they can have parameters and return values. Messages are depicted as horizontal arrows that connect the lifelines, and the name of the message is written above the arrow.
2. Activation Bar: The activation bar shows the period of time during which an object is active and processing a message. It is represented by a rectangle that is placed on the lifeline of the object
3. Focus of Control: The focus of control in a sequence diagram indicates which object is in control of the communication at a given time.
4. It is represented by a numbered sequence of steps or actions, with the object in control at each step indicated by a vertical dashed line.

## PACKAGE DIAGRAM for Online Shopping System



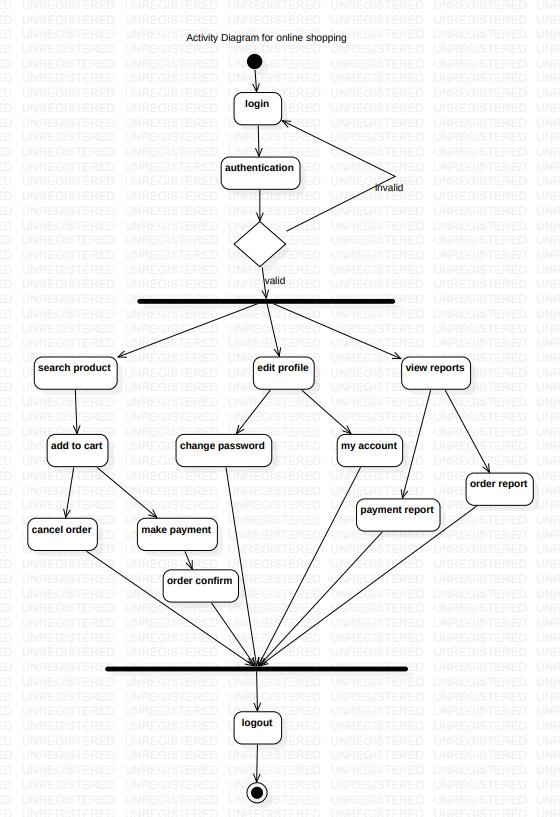
1. Packages: Packages represent a group of related classes, components, or subsystems that provide a cohesive functionality in the system. They can be nested within other packages to create a hierarchical structure.
2. Packages are typically represented by rectangles with the package name written inside.
3. Dependencies: Dependencies represent the relationship between packages in terms of the use of classes or components from one package in another package.
4. They can be dependencies between individual classes or components, or they can be dependencies between packages. Dependencies are typically represented by dashed arrows that point from the dependent package to the package that is being used.
5. Containment: Containment represents the relationship between packages in terms of the nesting of packages within other packages. It is represented by a solid line that connects the containing package to the contained package.

## Component Diagram for Online Shopping System



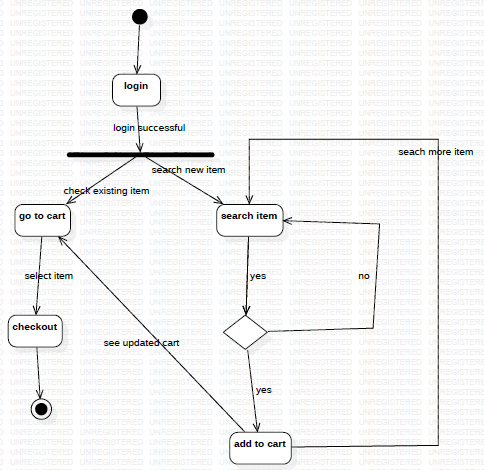
1. Components: Components represent the software modules or subsystems that make up the system.
2. They can be applications, libraries, or middleware. Components are typically represented by rectangles with the component name written inside.
3. Interfaces: Interfaces represent the contract that defines the behavior of a component. They specify the operations that can be performed by the component, and they can be implemented by multiple classes or other components. Interfaces are typically represented by rectangles with the interface name written inside.
4. Ports: Ports represent the points of interaction between components or between a component and its environment. They are typically represented by small squares or circles on the edge of the component.
5. Ports can have different types, such as input or output ports, and they can be connected to other ports through connectors.

## ACTIVITY DIAGRAM for Online Shopping System



1. Actions: Actions represent the individual steps or tasks that need to be performed in the system. Actions can be simple or complex, and they are typically represented as rounded rectangles with the action name written inside.
2. Control Flow: The control flow represents the sequence of actions in the system. It is represented by arrows that connect the actions, and the arrows indicate the direction of the flow. The control flow can be conditional or unconditional, and it can have loops and branches.
3. Decision Points: Decision points represent a branch in the control flow where a decision needs to be made based on a condition.
4. They are typically represented by a diamond shape, with the condition written inside. The control flow branches out from the decision point based on the outcome of the condition.
5. Swimlanes: Swimlanes represent the different actors or components in the system that are responsible for the actions.

## STATE CHART DIAGRAM for Online Shopping System



1. State chart diagrams model the behavior of a system and show how it changes state in response to events.
2. They consist of states, transitions, and events, and can model a wide range of systems.
3. They help developers understand and communicate system behavior, identify issues, and generate code.
4. They are part of the UML and often used in combination with other diagrams.
5. They provide a visual representation of a system's architecture and behavior.

# CODE:

# // C++ program to implement the program

# // that illustrates Online shopping

# #include <bits/stdc++.h>

# #include <cstring>

# #include <iostream>

# #include <map>

# using namespace std;

# char c1, confirm\_quantity;

# float quantity;

# int selectedNum;

# double total\_amount = 0;

# int flag = 0;

# // Stores items with their corresponding

# // price

# map<string, double> items = {

# { "Samsung", 15000 },

# { "Redmi", 12000 },

# { "Apple", 100000 },

# { "Macbook", 250000 },

# { "HP", 40000 },

# { "Lenovo", 35000 },

# { "C", 1000 },

# { "C++", 3000 },

# { "Java", 4000 },

# { "Python", 3500 }

# };

# // Stores the selected items with

# // their quantity

# map<string, int> selected\_items;

# // Function to print the bill after shopping

# // is completed prints the items, quantity,

# // their cost along with total amount

# void printBill(map<string, double> items,

# map<string, int> selected\_items,

# float total\_amount)

# {

# cout << "Item "

# << "Quantity "

# << "Cost\n";

# for (auto j = selected\_items.begin();

# j != selected\_items.end(); j++) {

# cout << j->first << " ";

# cout << j->second << " ";

# cout << (selected\_items[j->first])

# \* (items[j->first])

# << endl;

# }

# cout << "-----------------------"

# << "-------------\n";

# cout << "Total amount: "

# << total\_amount << endl;

# cout << "-----------------------"

# << "-------------\n";

# cout << "\*\*\*THANK YOU && HAPPY"

# << " ONLINE SHOPPING\*\*\*";

# }

# // Function to ask the basic details of

# // any customer

# void customerDetails()

# {

# cout << "Enter your name: ";

# string customer\_name;

# getline(cin, customer\_name);

# cout << "WELCOME ";

# for (int i = 0;

# i < customer\_name.length();

# i++) {

# cout << char(toupper(

# customer\_name[i]));

# }

# cout << "\n";

# }

# // showMenu() is to print the

# // menu to the user

# void showMenu()

# {

# cout << "Menu\n";

# cout << "= = = = = = = = "

# << " = = = = = \n";

# cout << "1.Mobile\n2.laptop\n3"

# << ".Computer courses\n";

# cout << "= = = = = = = = "

# << " = = = = = \n";

# }

# // Function to display the mobile products

# void showMobileMenu()

# {

# cout << "- - - - - - - - - - -"

# << " - -\nItem Cost\n";

# cout << "1.Samsung Rs.15, 000/-\n";

# cout << "2.Redmi Rs.12, 000/-\n";

# cout << "3.Apple Rs.1, 00, 000/-\n";

# cout << "- - - - - - - - - - - - -\n";

# }

# // Function to display Laptop products

# void showLaptopMenu()

# {

# cout << "- - - - - - - - - - -"

# << " - -\nItem Cost\n";

# cout << "1.Macbook Rs.2, 00, 000/-\n";

# cout << "2.HP Rs.40, 000/-\n";

# cout << "3.Lenovo Rs.35, 000/-\n";

# cout << "- - - - - - - - - - - - -\n";

# }

# // if the user selects computer courses,

# // then courses list will be displayed

# void showComputerCourseMenu()

# {

# cout << "- - - - - - - - - - "

# << " - -\nItem Cost\n";

# cout << "1.C Rs.1, 000/-\n";

# cout << "2.C++ Rs.3, 000/-\n";

# cout << "3.Java Rs.4, 000/-\n";

# cout << "4.Python Rs.3, 500/-\n";

# cout << "- - - - - - - - - - - - -\n";

# }

# // Function to display the mobile category

# void selectedMobile()

# {

# cout << "Do you wish to conti"

# <<"nue?(for yes" <<"press (Y/y ), "

# << " if no press other letter ): ";

# cin >> c1;

# if (c1 == 'Y' || c1 == 'y') {

# cout << "Enter respective number: ";

# cin >> selectedNum;

# if (selectedNum == 1

# || selectedNum == 2

# || selectedNum == 3) {

# // Selected Samsung

# if (selectedNum == 1) {

# cout << "selected Samsung\n";

# do {

# cout << "Quantity: ";

# cin >> quantity;

# cout << "You have selected Samsung - "

# << quantity << endl;

# cout << "Are you sure?"

# << "(for yes press (Y/y ), "

# << " if no press other letter): ";

# cin >> confirm\_quantity;

# } while ((confirm\_quantity != 'y'

# && confirm\_quantity != 'Y')

# || (quantity < 0)

# || (ceil(quantity) != floor(quantity)));

# if (confirm\_quantity == 'y'

# || confirm\_quantity == 'Y') {

# total\_amount += quantity

# \* items["Samsung"];

# selected\_items["Samsung"] = quantity;

# cout << "amount = "

# << total\_amount << endl;

# }

# }

# // Selected Redmi

# if (selectedNum == 2) {

# cout << "selected Redmi\n";

# do {

# cout << "Quantity: ";

# cin >> quantity;

# cout << "You have selec"

# << "ted Redmi - "

# << quantity << endl;

# cout << "Are you sure?(f"

# << "or yes press (Y/y ), "

# << " if no press other letter ): ";

# cin >> confirm\_quantity;

# } while ((confirm\_quantity != 'y'

# && confirm\_quantity != 'Y')

# || (quantity < 0)

# || (ceil(quantity)

# != floor(quantity)));

# if (confirm\_quantity == 'y'

# || confirm\_quantity == 'Y') {

# total\_amount += quantity

# \* items["Redmi"];

# selected\_items["Redmi"] = quantity;

# cout << "amount = "

# << total\_amount << endl;

# }

# }

# // Selected Apple

# if (selectedNum == 3) {

# cout << "You have selected Apple\n";

# do {

# cout << "Quantity: ";

# cin >> quantity;

# cout << "You have selected"

# << " Apple - "

# << quantity

# << endl;

# cout << "Are you sure?"

# << "(for yes press (Y/y )"

# << ", if no press other letter ): ";

# cin >> confirm\_quantity;

# } while ((confirm\_quantity != 'y'

# && confirm\_quantity != 'Y')

# || (quantity < 0)

# || (ceil(quantity)

# != floor(quantity)));

# if (confirm\_quantity == 'y'

# || confirm\_quantity == 'Y') {

# total\_amount += quantity

# \* items["Apple"];

# selected\_items["Apple"] = quantity;

# cout << "amount = "

# << total\_amount

# << endl;

# }

# }

# }

# else {

# flag = 1;

# }

# }

# else {

# flag = 1;

# }

# }

# // If Laptop category is selected

# void selectedLaptop()

# {

# cout << "Do you wish to continue?"

# << "(for yes press (Y/y ), "

# << "if no press other letter): ";

# cin >> c1;

# if (c1 == 'Y' || c1 == 'y') {

# cout << "Enter respective number: ";

# cin >> selectedNum;

# if (selectedNum == 1

# || selectedNum == 2

# || selectedNum == 3) {

# // selected Macbook

# if (selectedNum == 1) {

# cout << "selected Macbook\n";

# do {

# cout << "Quantity: ";

# cin >> quantity;

# cout << "You have selected"

# << " Macbook - "

# << quantity << endl;

# cout << "Are you sure?"

# << "(for yes press (Y/y ), "

# << " if no press other letter ): ";

# cin >> confirm\_quantity;

# } while ((confirm\_quantity != 'y'

# && confirm\_quantity != 'Y')

# || (quantity < 0)

# || (ceil(quantity)

# != floor(quantity)));

# if (confirm\_quantity == 'y'

# || confirm\_quantity == 'Y') {

# total\_amount += quantity

# \* items["Macbook"];

# selected\_items["Macbook"] = quantity;

# cout << "amount = "

# << total\_amount

# << endl;

# }

# }

# // selected HP

# if (selectedNum == 2) {

# cout << "selected HP\n";

# do {

# cout << "Quantity: ";

# cin >> quantity;

# cout << "You have selected"

# << " HP - "

# << quantity << endl;

# cout << "Are you sure?"

# << "(for yes press (Y/y ), "

# << " if no press other letter ): ";

# cin >> confirm\_quantity;

# } while ((confirm\_quantity

# != 'y'

# && confirm\_quantity != 'Y')

# || (quantity < 0)

# || (ceil(quantity)

# != floor(quantity)));

# if (confirm\_quantity == 'y'

# || confirm\_quantity == 'Y') {

# total\_amount += quantity

# \* items["HP"];

# selected\_items["HP"] = quantity;

# cout << "amount = "

# << total\_amount

# << endl;

# }

# }

# // selected Lenovo

# if (selectedNum == 3) {

# cout << "selected Lenovo\n";

# do {

# cout << "Quantity: ";

# cin >> quantity;

# cout << "You have selected"

# " Lenovo - "

# << quantity << endl;

# cout << "Are you sure?"

# << "(for yes press (Y/y ), "

# << "if no press other letter ): ";

# cin >> confirm\_quantity;

# } while ((confirm\_quantity != 'y'

# && confirm\_quantity != 'Y')

# || (quantity < 0)

# || (ceil(quantity)

# != floor(quantity)));

# if (confirm\_quantity == 'y'

# || confirm\_quantity == 'Y') {

# total\_amount += quantity

# \* items["Lenovo"];

# selected\_items["Lenovo"] = quantity;

# cout << "amount = "

# << total\_amount

# << endl;

# }

# }

# }

# else {

# flag = 1;

# }

# }

# else {

# flag = 1;

# }

# }

# // If computer course

# // category is selected

# void selectedCourses()

# {

# cout << "Do you wish to continue?"

# << "(for yes press (Y/y ), "

# << " if no press other letter ): ";

# cin >> c1;

# if (c1 == 'Y' || c1 == 'y') {

# cout << "Enter the respective number: ";

# cin >> selectedNum;

# if (selectedNum == 1

# || selectedNum == 2

# || selectedNum == 3

# || selectedNum == 4) {

# // selected C

# if (selectedNum == 1) {

# cout << "selected C Language"

# << " course\n";

# total\_amount += items["C"];

# selected\_items["C"]++;

# cout << "amount = "

# << total\_amount

# << endl;

# }

# // selected C++

# if (selectedNum == 2) {

# cout << "selected C++ Language course\n";

# total\_amount += items["C++"];

# selected\_items["C++"]++;

# cout << "amount = " << total\_amount << endl;

# }

# // selected Java

# if (selectedNum == 3) {

# cout << "selected Java Language course\n";

# total\_amount += items["Java"];

# selected\_items["Java"]++;

# cout << "amount = " << total\_amount << endl;

# }

# // selected python

# if (selectedNum == 4) {

# cout << "selected Python"

# << " Language course\n";

# total\_amount += items["Python"];

# selected\_items["Python"]++;

# cout << "amount = "

# << total\_amount

# << endl;

# }

# }

# else {

# flag = 1;

# }

# }

# else {

# flag = 1;

# }

# }

# // Driver code

# int main()

# {

# // function call

# customerDetails();

# do {

# showMenu();

# cout << "Do you wish to continue?"

# << "(for yes press (Y/y ), "

# << " if no press other letter ): ";

# char c;

# cin >> c;

# if (c == 'Y' || c == 'y') {

# cout << "Enter respective number: ";

# int num;

# cin >> num;

# if (num == 1 || num == 2

# || num == 3) {

# switch (num) {

# case 1:

# // For Mobile

# showMobileMenu();

# selectedMobile();

# break;

# case 2:

# // For Laptop

# showLaptopMenu();

# selectedLaptop();

# break;

# case 3:

# // For computer course

# showComputerCourseMenu();

# selectedCourses();

# break;

# }

# }

# else {

# flag = 1;

# }

# }

# else {

# flag = 1;

# }

# } while (flag == 0);

# // print bill

# printBill(items, selected\_items,

# total\_amount);

# }

# OUTPUT

# Graphical user interface, text, application, email Description automatically generated

# Text Description automatically generated

# CONCLUSION AND RESULT

UML diagrams are an essential tool for designing and documenting an online shopping system. UML diagrams can help identify the relationships and interactions between different modules of the system, which can ensure the system functions as intended and meets the needs of customers.

Some of the UML diagrams that can be used for an online shopping system includeA use case diagram can provide an overview of the different interactions between the users and the system. It can help identify the different use cases and how they relate to each other.

A class diagram can illustrate the different classes, objects, and attributes of the system. It can help identify the relationships between the different classes and how they interact with each other.

A sequence diagram can show the sequence of interactions between the user and the system. It can help identify the order of operations and the different actors involved in each interaction.

An activity diagram can illustrate the different activities and processes involved in the system. It can help identify the different steps involved in a process and how they relate to each other.

Overall, UML diagrams can help ensure the online shopping system is designed and documented in a clear and structured manner, which can lead to a more efficient and effective system**.**

# REFERENCE

UML modeling for online shopping system" by D. M. Thilini and D. D. Dhananjaya. International Journal of Computer Science and Network Security, Vol. 16, No. 12, December 2016.

"Design and Implementation of Online Shopping System based on UML" by Junhao Sun and Yan Wang. 2015 International Conference on Computational Science and Engineering.

"Modeling of Online Shopping System Using UML" by M. R. Rashid, M. A. Hossain, and

M. S. Rahman. International Journal of Computer Science and Information Security, Vol. 11, No. 6, June 2013.

"An Application of UML for the Design of an Online Shopping System" by Ahmed Alzahrani and Jules White. International Journal of Computer and Information Technology, Vol. 2, No. 1, January 2013