ONLINE SHOPPING STORE

PROJECT REPORT

18CSE202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

CHIRAG THAKUR (RA2111003010071)

TANISHQUE KUMAR (RA2111003010086)

Under the guidance of

Dr. C. Vijayakumaran

Associate Professor

Department of Computing Technologies

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kanchipuram

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**ONLIE SHOPPING CART"** is the bonafide work of **CHIRAG THAKUR (RA2111003010071) and TANISHQUE KUMAR (RA2111003010086)** who undertook the task of completing the project within the allotted time.

Signature of the Guide

Dr. C. Vijayakumaran

Associate Professor

Department of Computing Technologies

SRM Institute of Science and Technology

Signature of the II Year Academic Advisor

Professor and Head

Department of Computing Technologies

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 Split-up: - As per Curriculum)

CLAP-1	5=(2(E-lab Completion) + 2(Simple Exercises)(from Code Zinger, and any other coding platform) + 1(Hacker Rank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5=(2.0(E-lab Completion)+ 2.0 (Simple Exercises)(from Code Zinger, and any other coding platform) + 3.5 (Hacker Rank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5=(2.0(E-lab Completion(80 Pgms)+ 2.0 (Simple Exercises)(from Code Zinger, and any other coding platform) + 3.5 (Hacker Rank/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

1. Problem Description

The Online Shopping is a web based application intended for online retailers. The main objective of this application is to make it interactive and its ease of use. It would make searching, viewing and selection of a product easier. It contains a sophisticated search engine for users to search for products specific to their needs. The search engine provides an easy and convenient way to search for products where a user can Search for a product interactively and the search engine would refine the products available based on the user's input. The user can then view the complete specification of each product. They can also view the product reviews and also write their own reviews. The application also provides a drag and drop feature so that a user can add a product to the shopping cart by dragging the item in to the shopping cart. The main emphasis lies in providing a user-friendly search engine for effectively showing the desired results and its drag and drop behavior.

2. Software Requirements Specification

2.1. Introduction

A detailed description regarding the software requirements for an online shopping.

2.1.1. Purpose

The purpose of this application is to ease the process of ordering online. This is handled by automating various processes.

2.1.2. Document Conventions

N/A

2.1.3. Indented Audience and Reading Suggestion

This document should be read by developers, users, project managers and testers. The developers should read every section to ensure that there is an understanding of the project. The main sections for the customers to review are section 2.1.4 Project Scope, 2.2.7 Assumptions and section 2.4 Features.

2.1.4. Project Scope

The main aim of the project is to help owner of an online store manage the stock and transaction history. From the client side the main focus is to ease the process of searching and ordering items while providing ample customer support.

2.1.5. References

None

2.2. Overall Description

2.2.1. Product Perspective

The online shopping system provides a simple mechanism for the user to manage their account, order items and search for items according to their preference.

2.2.2. Product Functions

The Application includes a range of functions that enables the account holder to manage their bank account and to keep track on their expenditures seamlessly: Secure One-Time Authentication.

- Classification of items on basis of company, type, etc.
- Check stock and perform the necessary actions i.e., reorder stock if quantity is less or withhold stock if there is an excess amount.
- Provide a means to perform secure online transactions with a two-step authenticator app.
- Order tracking and history to re-order previous items and track the progress of the order.

2.2.3. User Classes and Characteristics

Any person who has an account can access the online store. They can order, return or even sell items on the platform. The owner can maintain inventory, verify transactions among other things.

2.2.4. Operating Environment

The Application will operate in the following operates environment:

- Android OS
- IOS

2.2.5. Design and Implementation Constrains

The Application's Android variant is created using Java programming language and the Android API. So, the Android variant is compatible with android devices running Android 4.1 or above with a minimum RAM of 1 GB. The Application's iOS variant is created using Objective C++ programming language and the iOS API. So, the iOS variant is compatible with iOS devices running iOS 7 or above. For Language support expect from the Basic English language pack the user can download and enable the language pack of their choice from the list of available languages within the application. For connection stream TCP/IP is used as it is the common gateway for internet applications.

2.2.6. User Documentation

There will be a basic tutorial document along with an in-app tutorial to aid users.

2.2.7. Assumptions and Dependencies

- The bank will provide full control for the app over the users' account.
- The app remains stable and compatible with Android 4.0 and greater.
- The app will be completely functional.

2.3. External Interface Requirements

2.3.1. User Interface

The look and feel must be simple and elegant for users to like it. The app will follow the color code of the bank in which the user holds the account. The font size is appropriate and the currency symbols are synchronized.

2.3.2. Hardware Interface

The app primarily runs on smart phones. So, the interface through which the user interacts should be touch enabled.

For the communication purpose, the program needs these protocols to be installed:

- TCP for the client to connect to the server in online mode.

Since the bank client runs behind a security system, the appropriate ports must be port forwarded or port triggered for clients to connect.

2.3.3. Software Interface

The software interface will be Android or iOS.

2.3.4. Communication Interface

Setting up the server into server mode requires that there will be open ports for accepting connections from the clients. The connection between the client and the server uses Connection-oriented communication, via TCP/IP — Transfer Control Protocol / Internet Protocol, implements reliable delivery of messages. Connection-oriented communication makes programming easier because the protocol includes mechanisms for detecting and handling error and an acknowledgement mechanism between client and server.

2.4. System Features

2.4.1. Secure Login

The application securely links the account to your device and makes sure the link is established successfully every time the application is run. This saves the valuable time of the user as it is not required to sign in to the bank account for every program session.

2.4.2. Classification of items

Classification of items on the basis of company, type, category, etc. makes it simpler and easily understandable.

2.4.3. View Order History

This system allows for users to view their past transactions and also re-order previously ordered items. This reduces the hassle in checking order status and also enables the online store to view

2.4.4. Secure Online Transactions

Secure Online Transactions through Two-Step Verification via an Authenticator App. The Two-Step Verification includes phase one authentication where the user's bank login credentials are verified and phase two authentication includes verification against a randomly generated PIN at real time.

2.4.5. Automatic Stock ordering

A system is implemented to automatically to check the status of the stock and order on a need

2.5. Other Non-Functional Requirements

2.5.1. Performance Requirements

Checking the fact that the system must perform as every user expects. So, in every action-response of the system, there is no immediate delay. In case of opening the store front page, popping of error messages and savings the settings or sessions there is delay much below 1.5 seconds. Also, when connecting to the server the delay is based on the distance between the main server and the client and the configuration between them, so there is high probability that there will be a successful connection in less than 15 seconds.

2.5.2. Safety Requirements

Checking the fact that all the clients must be attachable to one server, so there would be appropriate control of the test statistics and information. Also, in case of a potential loss of connection between the client and the server, the client's current progress is lost. The client must be able to restore progress at any given time. This is handled by locally caching the data.

2.5.3. Security Requirements

This application uses objected oriented mechanisms to protect its data using methods. It also uses industrial grade security protocols to protect its client's data. Thus, the log files are encrypted and heavily protected.

2.5.4. Software Quality Attributes

Availability: Checking that the system always has something to function and always pop-up error messages in case of component failure. In that case the error messages appear when something goes wrong so to prevail availability problems.

Usability: Checking that the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and traverses quickly between its states.

Functionality: Checking that the system provides the right tools for managing the online store inventory, carrying out secure transactions and updating and searching stock details.

2.5.5. Business Rules

This includes:

- Only transaction below Rs. 50,000 can be done using the app.
- The app has a limit of amount that can be transacted by the user in one single day as laid down by the authorities.
- If the customer is underage, features are not available for them.

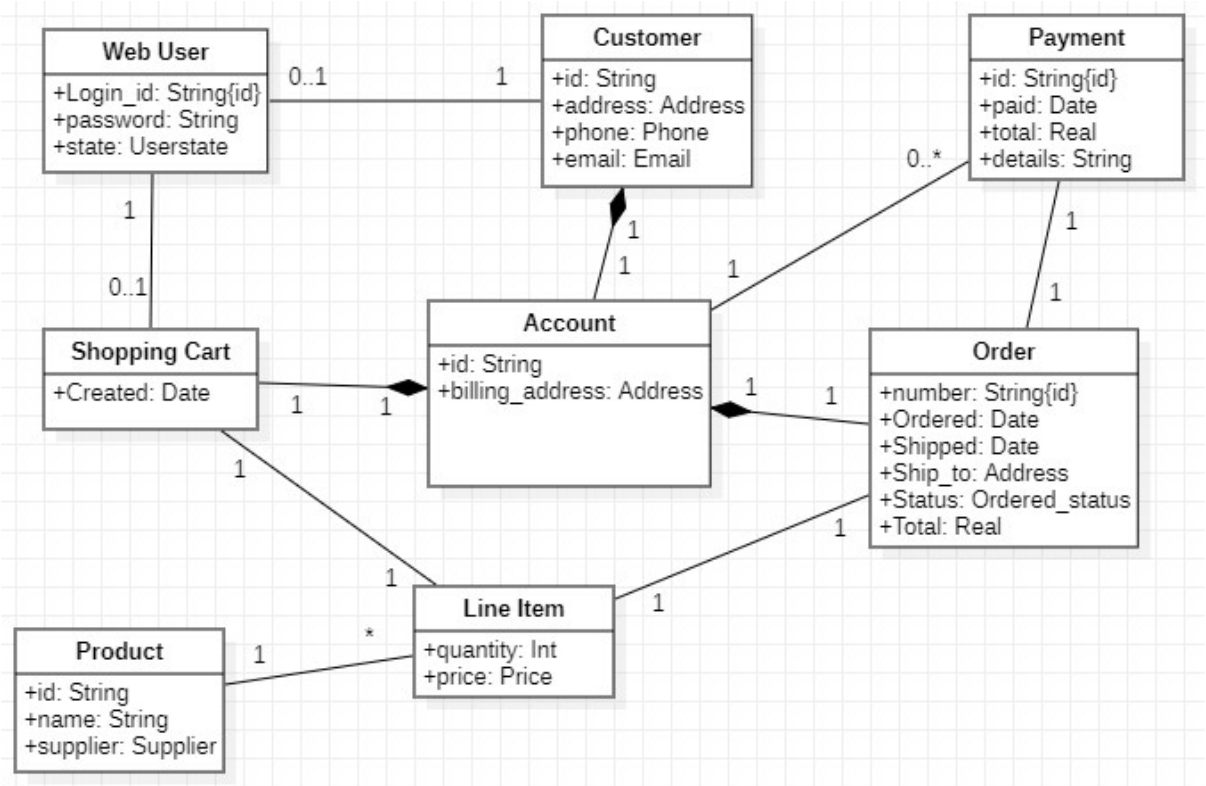
2.6. Other Requirements

N/A

Class Diagram

Class Diagram

The various classes required for the execution of the system are mentioned below. By creating the necessary classes, we can create the program required to execute the program. Account class contains the necessary functions to check for the authority of the user. The Customer class checks the role of the user and can modify the search user role in the system. The account class contains details about the user and can edit the values in the system. The items class contains details about the items and performs functions like adding, editing, deleting etc. Product Class check the availability of the items and the user can then update or add new stock.



SI.No	Parameters	Permissior	Role	User	Items	Sales	Stocks	Receipt
1.	Ensure that you model relationships horizontally	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2.	Collaboration means a need for a relationship	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3.	Model a dependency when a relationship is in transition	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4.	Depict similar relationships involving a common class as a tree	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5.	As a rule it is best to always indicate the multiplicity	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6.	Avoid multiplicity <<include>> to avoid confusion	Yes	Yes	Yes	Yes	Yes	Yes	Yes
7.	Replace relationships by indicating attribute types.	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8.	Never model implied relationships	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9.	Never model every single dependency	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10.	Center names on associations	Yes	Yes	Yes	Yes	Yes	Yes	Yes
11.	Write concise association names in active voice	Yes	Yes	Yes	Yes	Yes	Yes	Yes
12.	Indicate directionality to clarify an association name	Yes	Yes	Yes	Yes	Yes	Yes	Yes
13.	Name unidirectional associations in the same directions	Yes	Yes	Yes	Yes	Yes	Yes	Yes
14.	Word association names left-to-right	Yes	Yes	Yes	Yes	Yes	Yes	Yes
15.	Indicate role names when multiple associations between two classes exist	Yes	Yes	Yes	Yes	Yes	Yes	Yes
16.	Indicate role names on recursive associations	Yes	Yes	Yes	Yes	Yes	Yes	Yes
17.	Make associations bi-directional only when collaboration occurs in both directions	Yes	Yes	Yes	Yes	Yes	Yes	Yes

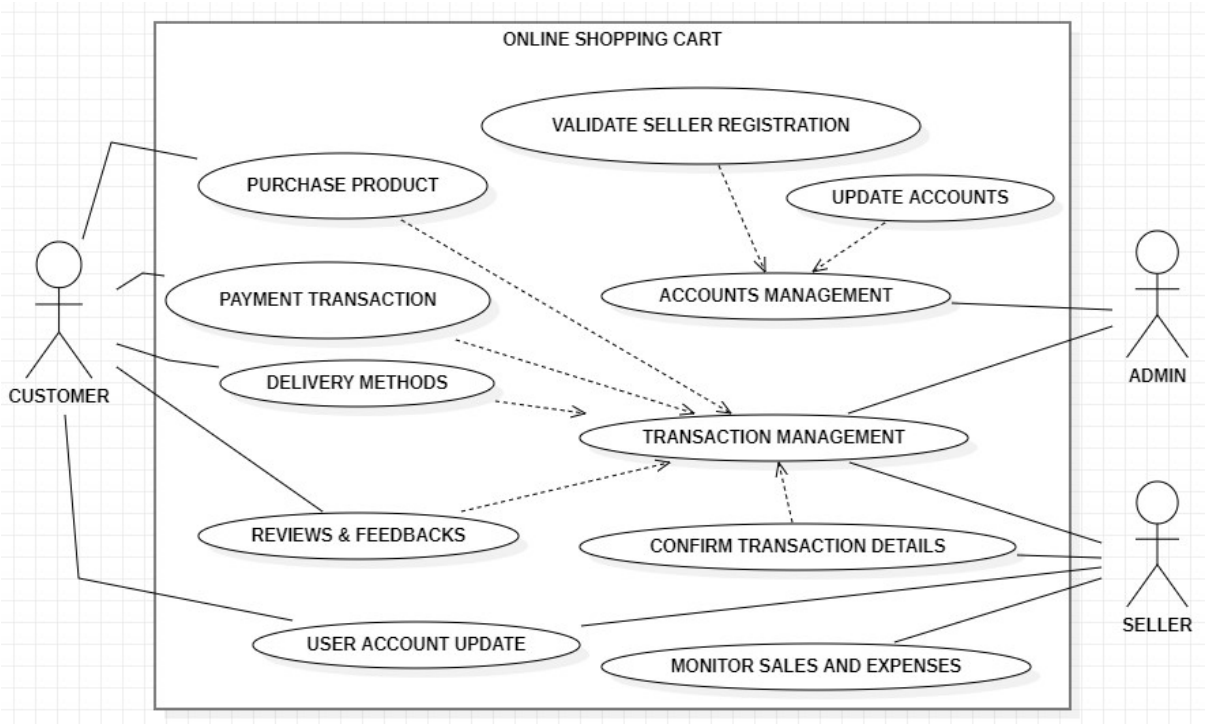
Use Case Diagram

USE CASE MODEL REPORT

Use cases are used to describe how the system should behave under stimuli. The intent is to anchor the process of domain analysis in actual work that needs to be done by the system. As you will see, the uses cases drive Rosenberg's design method at every stage.

Note that uses cases are part of the "domain model" not the design model. This is to say they record how an actor might interact with the objects that make up the problem domain. Naturally we will attempt to structure our design so that some of the classes of our design, the so called "entity" classes, correspond to the problem domain.

In this particular use case diagram, the three actors provide the necessary input to the system. The customer performs tasks like searching for the required items and making payments if the item has been purchased. The system admin performs tasks such as managing details about various products present in the online store and managing customer data. The system admin has access to all aspects of the system and is able to modify all the necessary data. The seller checks the various products present in the shop and handles the sales aspect of the shop. The seller is responsible for the actual resale of items and manages the finances in the shop.



Use Case to Search

Use Case Name	Search
Actor	Customer
Pre-Condition	Logged into the website
Post-Condition	-
Include	-
Extend	-
Frequency of Use	High frequency
Normal Course of Events	The customer enters the name of the item
Notes	-

Payments Use Case

Use Case Name	Payments
Actor	Customer, Admin
Pre-Condition	Logged into the website, items are at the checkout page
Post-Condition	-
Include	-
Extend	-
Frequency of Use	Low frequency
Normal Course of Events	The customer enters the name of the item and the item is returned
Notes	-

Manage Items Use Case

Use Case Name	Manage Items
Actor	Seller, Admin
Pre-Condition	Access to the inventory
Post-Condition	-
Include	-
Extend	-
Frequency of Use	Very low frequency
Normal Course of Events	The seller can check the quantity and order more based on the quantity
Notes	-

Accounts Use Case

Use Case Name	Search
Actor	Customer, Admin
Pre-Condition	None
Post-Condition	-
Include	-
Extend	-
Frequency of Use	Low frequency
Normal Course of Events	The customer enters the details and they are verified by the server according to the standards set by the admin
Notes	-

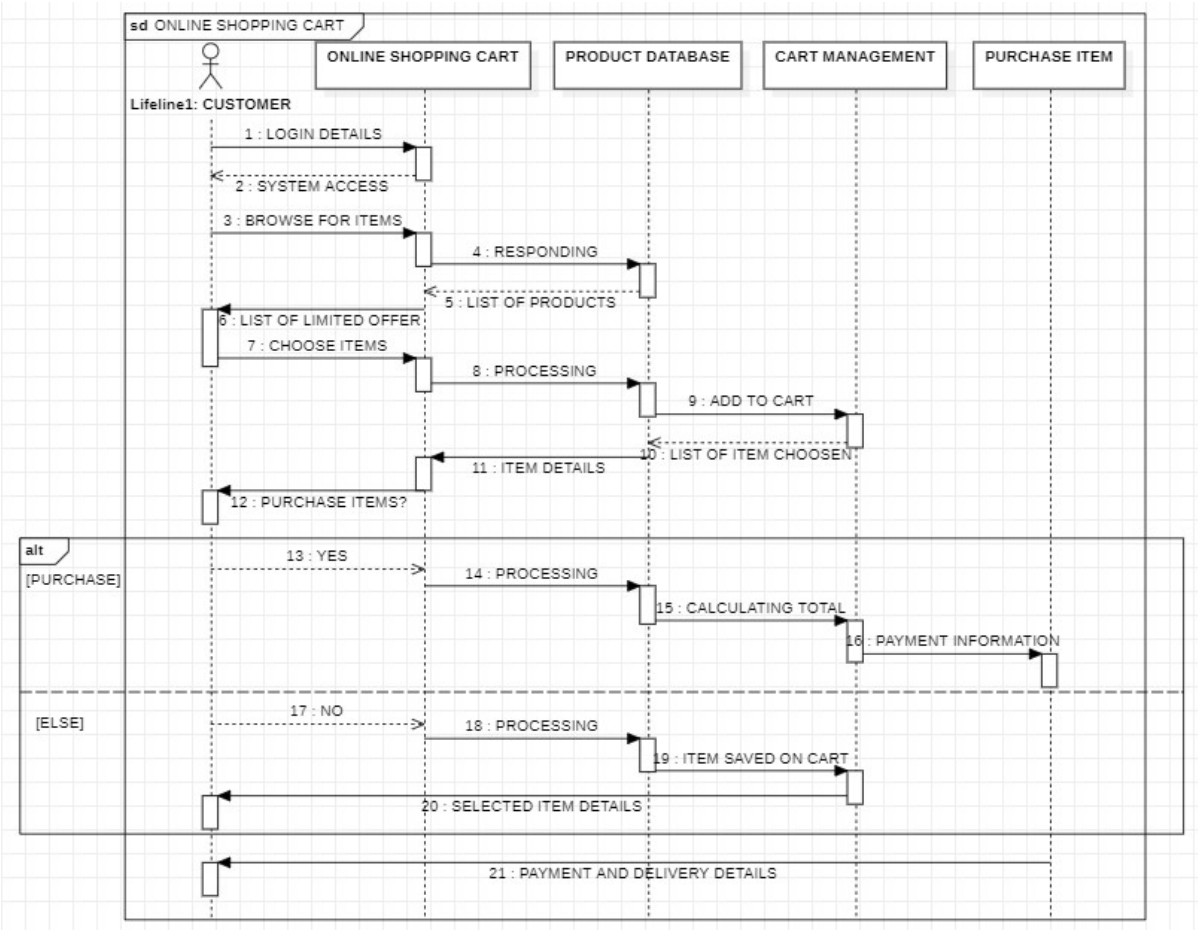
Checklist

Sl. No.	Conventions to be Checked	Use Case Checklist						
		Search For Items	View Accounts	Make Payments	Manage Items	Manage Stock	User Details	Feedback and Queries
1	Use case name begins with a strong verb	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	Name use cases using Domain Terminology	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	Place your primary use cases in the top left corner of diagram	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	Imply timing consideration by stacking Use Cases	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	Use Case names use an active voice	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6	Use case names use the present tense	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Sequence Diagram

Sequence Diagram Report

The sequence diagram shows the process involved for performing a particular action. The sequence diagram shows the process of how a user can login into the system and access the items from the given options. It also shows the process involved in selecting the items and performing the necessary actions involved in order to successfully place an order for the item. The user, depending on the level of permission can access the system in order to perform actions like placing an order, managing sales details etc. For such a process to take place the user has to have adequate permissions.



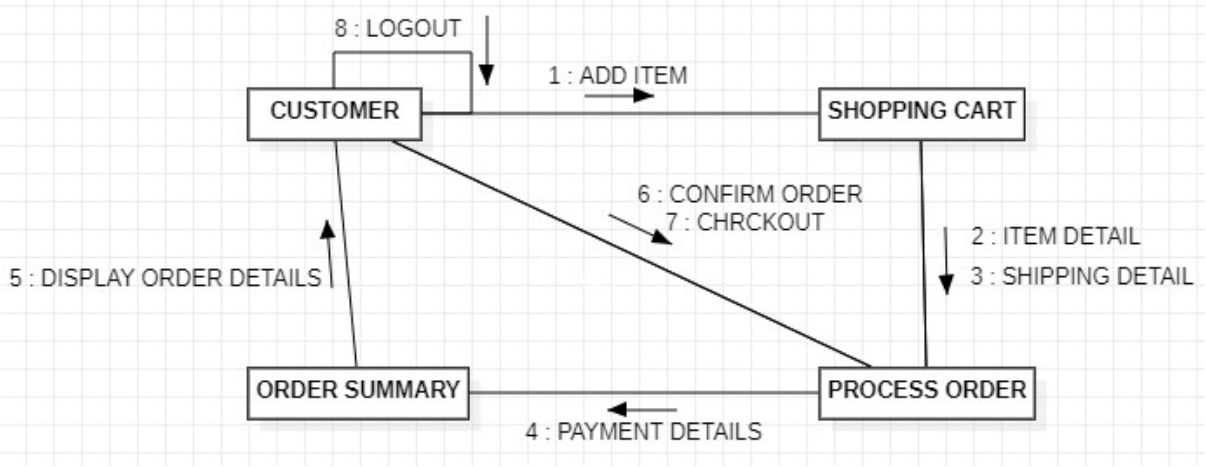
Checklist

S. No.	Sequence Checklist	Check(Y/N)
1	Messages are from Left-To-Right	Yes
2	Actors named consistently with Use Case Diagram	Yes
3	Classes named consistently with class Diagram	Yes
4	Human and Organisation actors on left most side	Yes
5	Reactive system actors on right most side	Yes
6	Proactive system actors on left most side	Yes
7	Message names beside arrowhead justified	Yes
8	Do not return value when it is obvious	Yes
9	Use return value only when you need to refer it elsewhere	Yes

Collaboration Diagram

Collaboration Diagram Report

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming.
The given collaboration diagram shows the process for logging into the system to access the online shop and perform the necessary transactions for the sale of the items.



Checklist

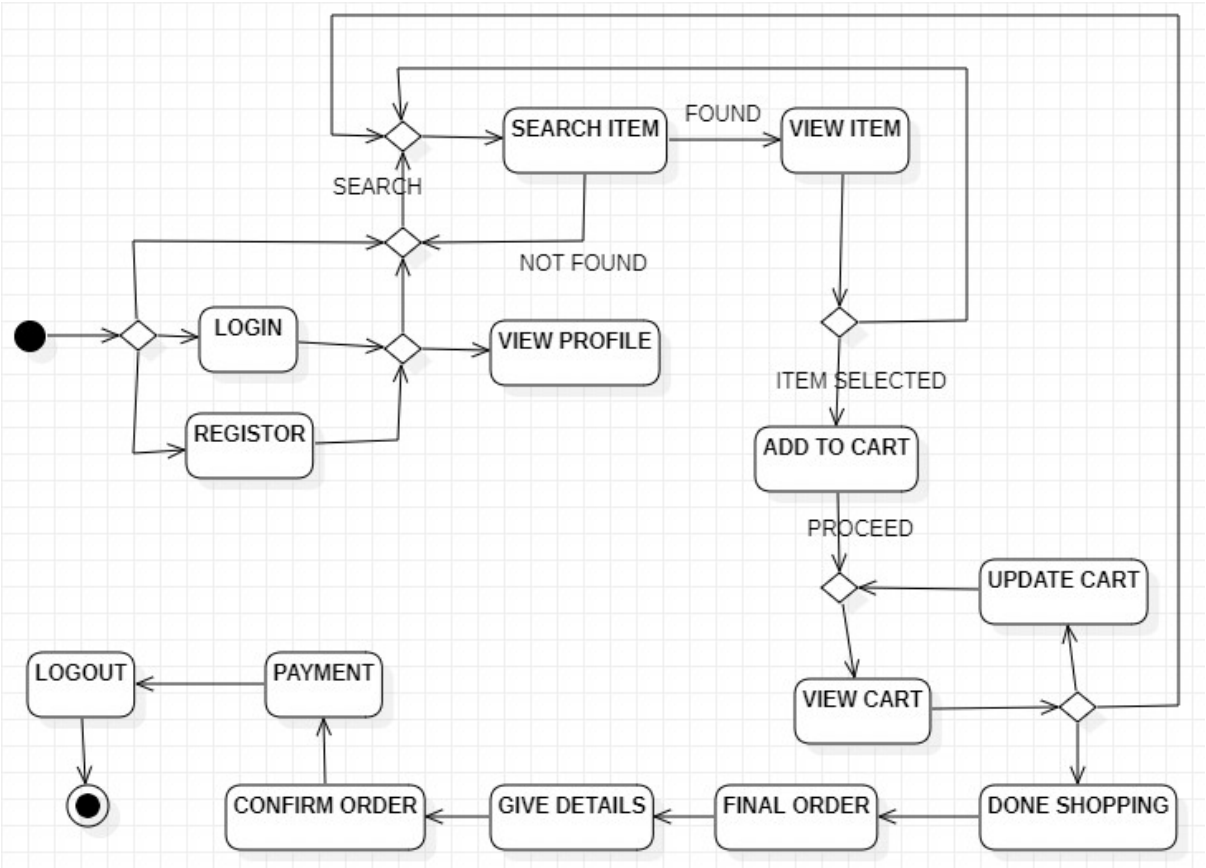
S. No.	Communication Checklist	Check(Y/N)
I	Name objects when referred to in messages	Yes
2	Name objects when several of same type exist	'Yes
3	Do not model return value when it is obvious	Yes
4	Model return value when you need to refer to elsewhere	Yes
5	Indicate return value when it is not clear	Yes
6	Indicate parameters when they are not clear	Yes
7	Depict arrow for each message	Yes
8	Indicate navigability sparingly	Yes
9	Prefer roles on links instead of within classes	Yes

State Diagram

State Chart Diagram

This diagram shows the process behind ordering a item and it shows the exact state of the system when a step is being executed. A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). In this context, a state defines a stage in the evolution or behavior of an object, which is a specific entity in a program or the unit of code representing that entity. A state diagram resembles a flowchart in nature; however, a flowchart shows the processes within a system that alters the state of an object rather than the actual state changes themselves.

First state starts with activity diagram Customer login which verifies into Verification activity diagram which connected to Re-login and Access Account with access to Manage Account Activity Add to cart and Cancel item activity diagrams are completed or cancelled and proceed to Order Processing Activity which processes to Transaction Activity Transaction if Transaction successfully connects to Order Summary activity diagram ends with Terminator symbol.



Checklist

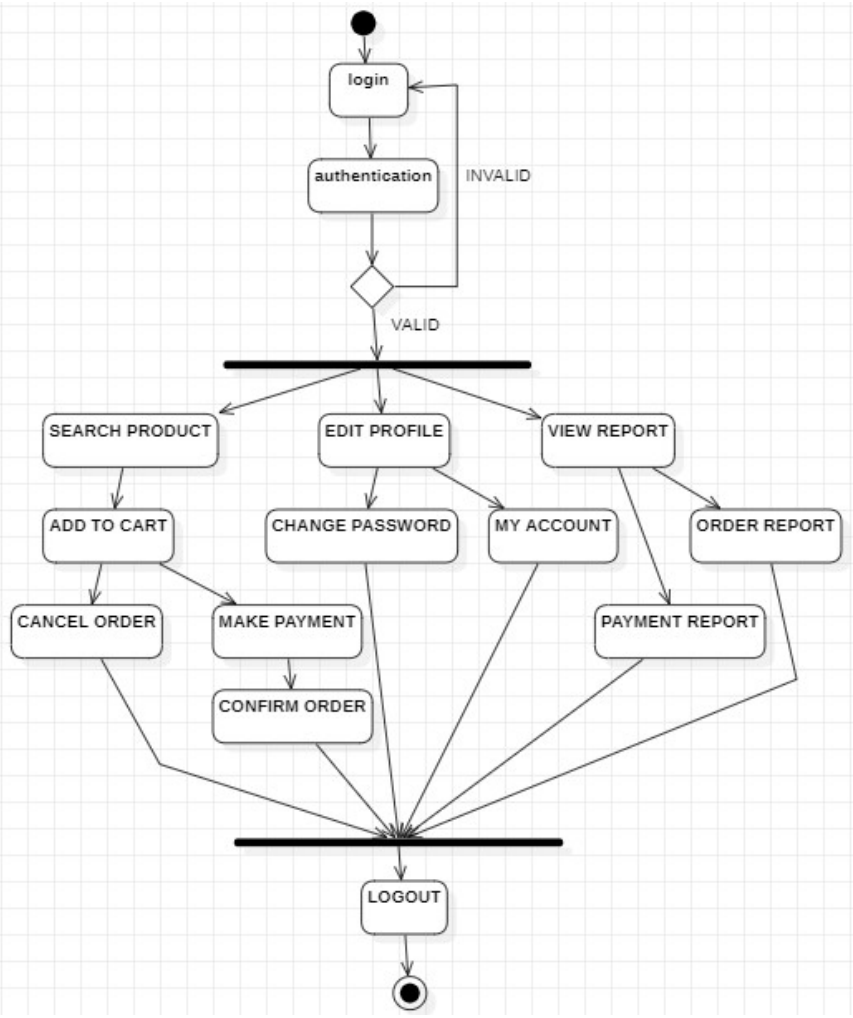
S.NO	State Chart Checklist	Check(Y/N)
1	Initial state on top left corner	Yes
2	Final state on bottom right corner	Yes
3	State names should be simple but descriptive	Yes
4	Model Sub states for targeted complexity	Yes
5	Aggregate common Sub state transitions	Yes
6	Top-level state machines always have initial and fin al states	Yes
7	Name software actions using implementation language naming conventions	Y e s
8	Name actors using prose	Yes
9	Indicate entry actions wherever applicable	Yes
10	Indicate exit actions wherever applicable	Yes
11	Model recursive transitions only when you want to exit and re-enter the state	Yes
12	Name transition event in past tense	Yes
13	Place transition label near source state	Yes
14	Place transition label based on source direction	Yes

Activity Diagram

Activity Diagram Report

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity 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 use Activity Diagrams to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram.

The activity diagram shows the necessary flow of data in the item shop inventory system. The user begins by logging in and the system checks the user permissions. If the user has adequate permissions the user can then access the system functions to perform the necessary tasks. Activity diagram is started with Activity login into Item shop inventory system. Check user level and permissions. It is connected to check permissions which are connected to Manage Items, Manage Sales, Manage Inventory, Manage Stocks, Manage Publications activity diagrams. Every activity is connected to Logic from the system activity and end.



Checklist

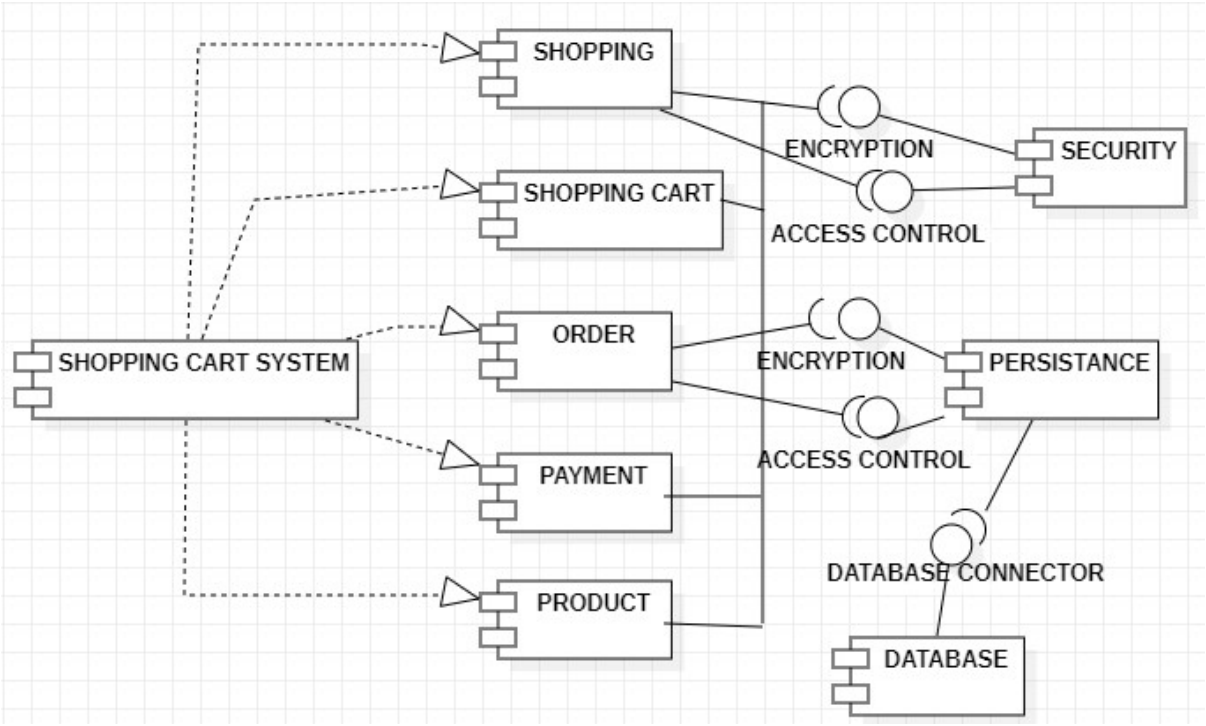
S.NO	Activity Checklist	Check(Y/N)
1	Place start point on top left corner	Yes
2	Always include end point	Yes
3	Simplify Flow charting operations	Yes
4	Decision points should reflect previous activity	Yes
5	Avoid Superfluous decision points	Yes
6	Each transition leaving point must have a guard	NIL
7	Guards should not overlap	NIL
8	Guards on decision points must forma complete set	NIL
9	Exit transition guards and Activity Invariants must form a complete set	NIL
10	A fork should have a corresponding join	NIL
11	Forks only have one entry transition	NIL
12	Joins only have one exit transition	NIL
13	Avoid superfluous forks	NIL
14	Order Swim lanes in logical manner	NIL
15	Apply swim lanes to linear processes	NIL
16	Have less than 5 swim lanes	NIL
17	Consider horizontal swim lanes for business processes	NIL

Component Diagram

Component Diagram

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node. It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

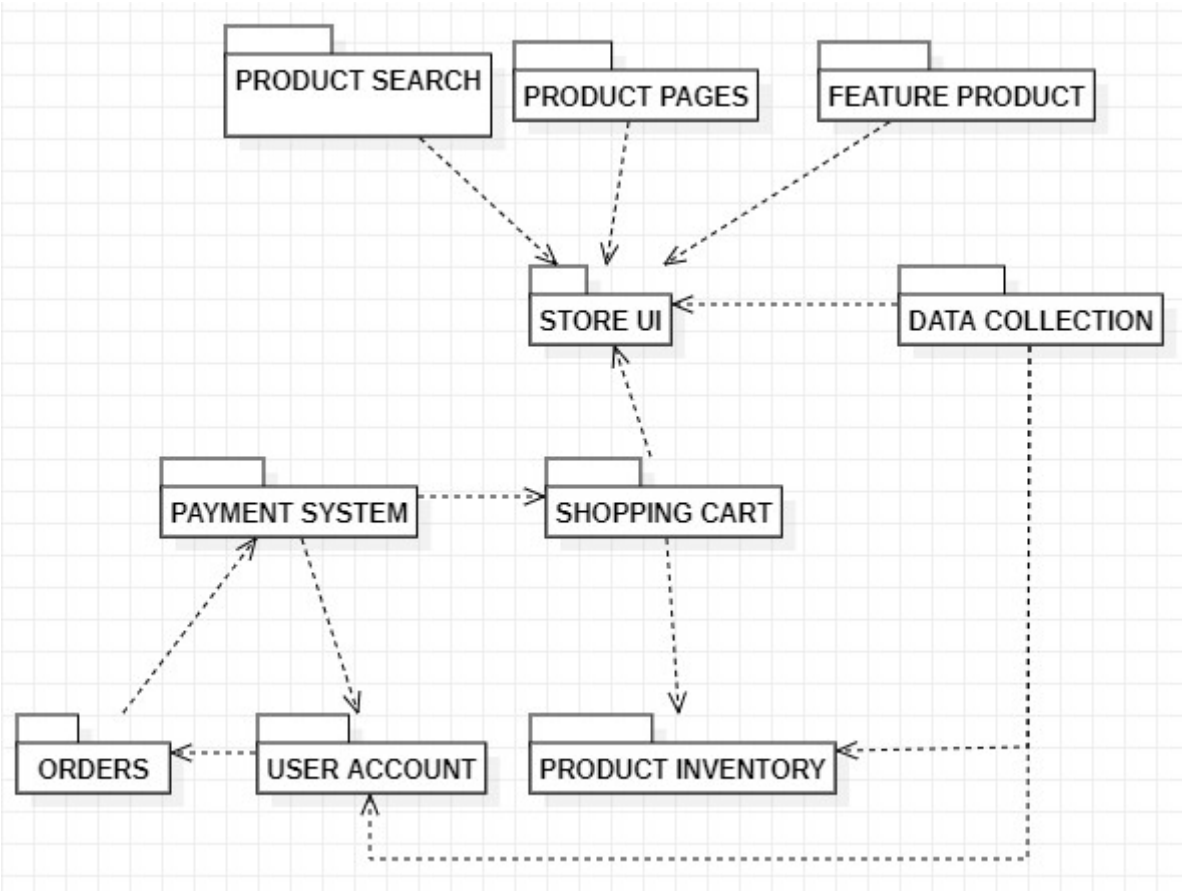
Item shop management system component is connected with the provided interface Interface1. Whole management system is dependent on Items, Sales, Inventory, Stock, Publication components which are connected with the provided interface data type. Components show the consumer behavior. Item, Inventory components are extended with required interface to get connected with provided interface of security, Persistence components. Those combinations are called Encryption, Access control. Persistence is connected with required interface which is connected with provided interface of Database component which is named database connector.



Package Diagram

Package Diagram

Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages. A package is a grouping of related UML elements, such as diagrams, documents, classes, or even other packages. Each element is nested within the package, which is depicted as a file folder within the diagram, then arranged hierarchically within the diagram. Package diagrams are most commonly used to provide a visual organization of the layered architecture within any UML classifier, such as a software system. Item shop management system package has dependency with many packages like Items, Users, Resources, Database. Items has sub package with Type, User has sub package with Specification, Resource with sub package as Stock and Database package with sub package Details of item. User package is dependent on Database. Database package is in dependency with Items package, User package, Resources package and Item shop management system package.

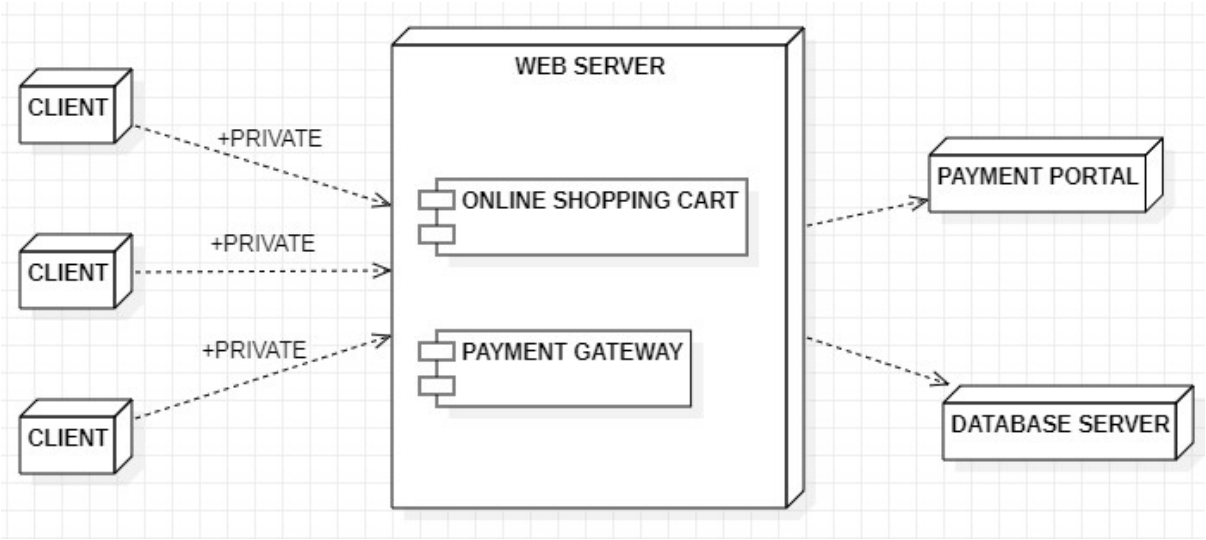


Deployment Diagram

Deployment Diagram

In UML, deployment diagrams model the physical architecture of a system. Deployment diagrams show the relationships between the software and hardware components in the system and the physical distribution of the processing. Deployment diagrams, which you typically prepare during the implementation phase of development, show the physical arrangement of the nodes in a distributed system, the artifacts that are stored on each node and the components and other elements that the artifacts implement. Nodes represent hardware devices such as computers, sensors, and printers, as well as other devices that support the runtime environment of a system. Communication paths and deploy relationships model the connections in the system.

Deployment diagram has nodes of the Item shop management system, Data server and Clients. Every node is connected with communication association type. Data server has Web server has items database component and user database component. Data server component is connected with Web server component. Item shop management is connected with client nodes.



Conclusion

The ‘Online Shopping’ is designed to provide a web based application that would make searching, viewing and selection of a product easier. The search engine provides an easy and convenient way to search for products where a user can Search for a product interactively and the search engine would refine the products available based on the user’s input. The user can then view the complete specification of each product. They can also view the product reviews and also write their own reviews. Use of Ajax components would make the application interactive and prevents annoying post backs. Its drag and drop feature would make it easy to use.

The overall idea of doing this project is to get a real time experience. Learn new technologies.

References

- Australian Computer Society, 2003, ASC Code of Ethics. Retrieved March 15,
- 2007, from http://www.acs.org.au.htm
- Elmasri, R. and Navathe, S. 2004. Enhanced Entity Relationship and UML. In
- Fundamentals of Database Systems, 3rd Edition
- Out Source 2 India n.d.: Why Do Software Projects Fail? Retrieved 22 March 2007 from
- http://www.outsource2india.com/software/SoftwareProjectFailure.asp
- Six Sigma n.d. : Applying Six Sigma to Software Implementation Projects Retrieved 22
- Sommerville, Ian 2004. Object Oriented Design Software Engineering, 7thEdition
- Start your journey the easy way n.d : Retrieved 4th February 2007 from
- http://www.liverpooljohnlennonairport.com/TravelServices/CarParking.php
- Ramakrishnan, R. and Gehrke, J. 2003. The Relational Model In Database
- Management Systems, 3rdEdition
- All about Microsoft controls in C# http://www.msdn.microsoft.com/
- Wikipedia for various diagrams & testing methods http://www.wikipedia.org/
- Cool text for Images and Buttons http://cooltext.com/
- K-State Research Exchange for samples in report writing http://krex.k-state.edu/dspace/handle/2097/959
- Smart Draw for drawing all the Diagrams used in this report. http://www.smartdraw.com/
- Sample Ecommerce Application http://www.NewEgg.com
- Ajax Toolkit controls http://asp.net/ajax