SWAPZONE – ONLINE CLOTH STORE

Mini Project Report

Submitted by

Preethy Ann Thomas

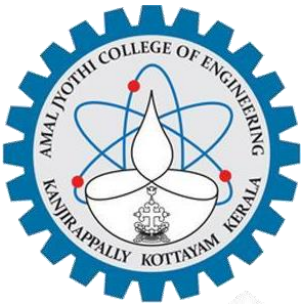
Reg. No.: AJC19MCA-I044

In Partial fulfillment for the Award of the Degree of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, **"SWAPZONE"** is the bona fide work of **PREETHY ANN THOMAS (Regno: AJC19MCA-I044)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Sruthimol Kurian
Internal Guide

Ms. Meera Rose Mathew
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report **"SWAPZONE"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Integrated Master of Computer Applications (INMCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: **PREETHY ANN THOMAS**

KANJIRAPPALLY **Reg: AJC19MCA-I044**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Sruthimol Kurian** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

PREETHY ANN THOMAS

ABSTRACT

.

SwapZone is a comprehensive e-commerce platform designed to cater to the needs of both sellers and customers within the clothing industry. It serves as a one-stop solution, offering sellers a platform to list and sell a wide range of clothing items, including both stitched and unstitched garments. Simultaneously, it provides customers with a seamless shopping experience, enabling them to effortlessly browse, search for, purchase, and even sell clothing items.

The platform boasts a user-friendly interface and intuitive navigation, ensuring that users can easily access an extensive selection of clothing items spanning various categories. Whether customers are looking for trendy fashion apparel or unique ethnic wear, SwapZone offers a diverse marketplace where they can find what they need.

One of the core principles at SwapZone is inclusivity, allowing both individuals and businesses to become sellers on the platform. Sellers have the flexibility to buy and sell clothing items within the same ecosystem. To maintain a high standard of quality and customer satisfaction, sellers are assigned credit points. In cases where a seller receives multiple complaints or negative feedback from customers, their selling privileges may be restricted or suspended, ensuring a reliable and trustworthy shopping environment.

The project contains mainly three modules:

- Admin
- Seller
- Customer

The admin module holds the administrative functionalities, enabling oversight and management of the platform. The seller module empowers individuals and businesses to list their clothing items sell their products efficiently. Lastly, the customer module empowers shoppers to explore, add to Wishlist, add to cart and even purchase clothes via online payment, contributing to a vibrant and dynamic online community for clothing enthusiasts.

CONTENT

List of Abbreviation

IDE -	Integrated Development Environment
HTML -	Hyper Text Markup Language
CSS -	Cascading Style Sheet
SQL -	Structured Query Language
UML -	Unified Modelling Language
PHP -	PHP: Hypertext Preprocessor
JS –	JavaScript
AJAX –	Asynchronous JavaScript and XML Environment
BDD-	Behavioural Driven Development

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

SwapZone is a cutting-edge online marketplace created to transform the buying and selling of apparel goods by providing a distinctive and convenient experience to both sellers and consumers. The mission of SwapZone is to make it simple for people to buy and sell apparel. Users may browse, list, and trade a wide range of clothing items on the site, including both sewn and unstitched clothes.

The idea of community-driven trade is at the core of SwapZone. Sellers, who are frequently private individuals or small-scale clothing enterprises, may readily put their clothing items for sale and interact with prospective customers directly. This strategy encourages sellers to feel empowered and supports local, sustainable business. Through a credit point system, where they accrue points for successful transactions, sellers are rewarded. Customer complaints are carefully investigated to safeguard the integrity of the site, however sellers who consistently have problems may experience temporary selling limitations.

Administrators, sellers, and consumers are just a few of the many user roles that SwapZone's extensive modules serve, each with specific functions. A wide variety of apparel goods from many categories are easily accessible thanks to the platform's simple navigation and user-friendly layout. The textile exchange market will be transformed by SwapZone, making it more open, centered on the community, and empowering for all players.

1.2 PROJECT SPECIFICATION

SwapZone is an online platform developed to facilitate the buying and selling of clothing items.
Designed for individual sellers and small clothing businesses to reach a broader audience.
Users can trade a variety of clothing, including stitched and unstitched garments.

There are mainly three actors in this project:

Administrators:

- Manage the platform, user accounts, and transactions.
- Oversee seller performance and address customer complaints.
- Ensure platform security and functionality.

Sellers:

- Register and create seller profiles.
- List clothing items for sale, including images, descriptions, and prices.
- Accumulate credit points for successful transactions.
- Receive customer orders and arrange product delivery.
- Respond to customer inquiries and issues.
- Face temporary selling restrictions for repeated customer complaints.

Customers:

- Create user accounts and profiles.
- Browse a wide range of clothing items in various categories.
- Place orders, make payments, and track deliveries.
- Rate and review sellers based on their buying experience.
- Lodge complaints and disputes for seller or product-related issues.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The SwapZone system research has shed important light on the goals, functioning, and design of this ground-breaking textile exchange platform. The goal of SwapZone is to build a vibrant marketplace that connects buyers and sellers in the clothes sector. The scope and direction of the project have been greatly influenced by this thorough investigation.

First of all, SwapZone is better prepared to accommodate the various demands of both vendors and buyers by knowing user wants and expectations. The study has discovered the needs and motives of potential customers through surveys, interviews, and market research. SwapZone will be able to provide a platform that is not just feature-rich but also catered to the tastes of its target audience thanks to this user-centric strategy.

Second, the system analysis has created the framework for deciding on the best technology stack and outlining important functionality. For a smooth user experience, strong security protocols, and effective operations, this is essential. SwapZone is well on its way to become a dependable and user-friendly cloth exchange platform with an emphasis on user-friendly interfaces and a fair credit point system for vendors.

2.2 EXISTING SYSTEM

The lack of a centralized, user-friendly platform in the current system of apparel exchange and commerce makes it fragmented and ineffective for both vendors and customers. It is now difficult to attract a larger audience since people trying to sell goods frequently turn to sporadic internet listings, local marketplaces, or social media channels. Similarly, buyers have a hard time finding a wide variety of clothing alternatives and sellers, which results in few selections. Additionally, there is no credit-based mechanism in place to uphold buyer and vendor confidence. Overall, the current system is uncoordinated, inaccessible, and devoid of defined procedures, which creates space for a more effective and inclusive solution like SwapZone to completely transform the world of clothes exchange.

2.2.1 NATURAL SYSTEM STUDIED

The primary component of the natural system being researched in the context of the online

marketplace and clothes exchange SwapZone is the dynamic and changing behavior of users, sellers, and buyers. This approach takes into account the inherently complicated nature of social interactions, personal preferences, and clothing-related decision-making. It takes into account elements like fashion trends, seasonality, and personal preferences, all of which have an impact on the platform's clothes exchange and purchasing patterns. For SwapZone to succeed, it is essential to comprehend and successfully utilize this natural system. The platform may customize suggestions for users, improve inventory management for vendors, and improve user experience overall by utilizing data analytics and machine learning algorithms. SwapZone is able to develop a dynamic and accommodating marketplace that satisfies the ever-changing preferences and demands of its consumers thanks to the study of this natural system.

2.2.2 DESIGNED SYSTEM STUDIED

The SwapZone system is a comprehensive and well-organized platform that acts as the foundation of this vibrant marketplace for apparel exchange. It features an intuitive online interface, strong security measures, efficient payment processors, and a huge product selection. Platform operations are made possible by essential modules including seller and customer administration, as well as an administrative dashboard.

The versatility of the system's architecture is one of its most notable qualities since it enables vendors to display both sewn and unstitched clothing products, giving prospective purchasers a wide range of options. In addition to these features, the system's architecture makes use of machine learning and data analytics to let seller extra information based on submitted fabric images.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Lack of Centralization: The existing system is fragmented, with sellers and buyers spread across multiple websites and platforms. This decentralization makes it challenging for users to browse and manage listings efficiently.

- Trust and Credibility Issues: Due to the absence of a structured feedback and rating system, users have limited means to assess the trustworthiness of sellers. This lack of transparency can result in unsatisfactory transactions.

- Incentive for Quality: The existing system lacks mechanisms to encourage responsible and

quality-driven seller behavior. Sellers may not have sufficient incentives to consistently provide high-quality products and service.

- Security Concerns: Security threats such as fraud and data breaches are prevalent in the current system. Without a comprehensive security framework, user information and financial transactions are at risk.

- Inefficient User Experience: Navigating through various platforms can be cumbersome for users. The lack of a user-friendly interface and simple navigation hinders the shopping experience.

- Limited Accountability: There is a lack of accountability mechanisms for sellers. Without a reward or penalty system, sellers may not be motivated to resolve disputes or maintain high standards.

- Absence of a Unified Platform: Users must resort to word-of-mouth or scattered online reviews to make purchase decisions. The absence of a centralized marketplace contributes to confusion and inefficiency.

2.4 PROPOSED SYSTEM

Through the provision of a centralized, user-friendly, and secure online platform, the proposed SwapZone system aims to transform the experience of buying and selling clothing. Users will have access to a thorough marketplace where they can easily explore, search, purchase, and sell cloth products. A sophisticated rating and feedback system will be implemented by SwapZone with a focus on credibility and trust, empowering users to make educated choices and encouraging a feeling of dependability within the community. Measures for quality control will encourage vendors to uphold high standards, guaranteeing that customers always obtain high-quality goods. The entire user experience will be further improved by improved security features, effective dispute resolution procedures, and visible user profiles. SwapZone intends to establish a vibrant online ecosystem by providing administrators, sellers, and clients with specific modules.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The proposed SwapZone system is a potential option for both sellers and buyers since it has a number of benefits over the current environment for buying and selling cloth:

1. Integrated and centralized platform: Users may discover a large variety of fabric products on SwapZone, which offers a consolidated online marketplace. By eliminating the need to visit several platforms, this convenience allows you to find the apparel you want faster and with less effort.

2. A user-friendly interface: No matter their level of technical experience, users of the system may quickly browse and utilize the platform thanks to its straightforward and user-friendly interface. The whole user experience is improved by this usability.

3. Credibility and Trustworthiness: A comprehensive rating and feedback system is included in SwapZone. By allowing buyers to base their judgments on the experiences of others, the community may become more credible and trusting. Positive feedback builds a reputation for dependability for sellers.

4. Quality Control: Through a rewards system for good comments and a punishment system for unresolved complaints, sellers are encouraged to uphold high standards. This guarantees that customers get the quality they deserve.

5. Transparency: Transparency is provided via comprehensive user profiles for both buyers and vendors. Users have access to transaction histories, ratings, and reviews, giving them the information they need to make wise choices.

6. Specific Modules: SwapZone provides separate modules for vendors, administrators, and customers. Each module streamlines various user groups' interactions with the platform by catering to the particular features and requirements of each.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

This assessment is a fundamental method for determining if the project will achieve the goals of the organization in relation to the resources, labour, and time invested in it. It aids the designer in determining the project's potential focus points and long-term outcomes. A possibility analysis must be done to determine whether a proposed framework is feasible and advantageous for advance examination. The ability of the proposed system to meet client demands, the association's influence, and resource efficiency are all evaluated as part of the potential consideration. Achievability analysis is thus usually carried out after approval for the creation of a new application. The extension's technical, budgetary, and operational justifications are roughly equivalent.

3.1.1 Economical Feasibility

The process of evaluating a new project's economic viability in terms of time and financial commitment is essential. It entails a careful examination of all elements that may have an impact on the initiative's outcome. Following a cost-benefit study, it was determined that the suggested solution, SwapZone, is both practical and cost-effective given the project's presumptive cost.

This system is economically feasible as it reduces a lot of manual works making it automatic and user friendly. Different cost categories, comprising labour costs, computer costs, equipment and supplies costs, expenses associated with implementing new software and computer equipment, costs for system evaluation, website coding, and database design, were evaluated to establish the development cost of the system. Typically, they are one-time expenses that cease to exist when the project is completed. Since this is an academic project all these costs are saved and as a result no expenses are incurred. We are able to make sure that the system's development is economically feasible and will produce a good return on investment by doing an exhaustive examination of these cost categories.

3.1.2 Technical Feasibility

Technical feasibility refers to the evaluation process that assesses if it is feasible to produce and deploy a product or service utilizing current technology and resources. The proposed plan's tools, materials, labour, logistics, and technology are examined as part of the technical feasibility analysis to gauge how successful it would be. Before beginning the task, it is important to identify

and handle any potential project concerns. Technical feasibility can help in visualizing the system's process by making a flowchart of the product or service's development.

This website is technically feasible as it uses latest web technologies to develop the system. The technologies used can be made accomplished to user requirements in the software within the allocated time and budget, and also new updates can be performed.

The technical requirements for the system are:

1) MySQL, PHP is used as the backend technology
2) HTML, CSS JavaScript, Bootstrap, AJAX as the frontend technology

The latest web technologies are used to develop the system. The technologies used can be made accomplished to user requirements in the software within the allocated time and budget, and also new updates can be performed.

1) Is the project feasible within the limits of current technology.
   Yes
2) Technical issues raised during the investigation are:
   Nothing
3) Can the technology be easily applied to current problems?
   Yes
4) Does the technology have the capacity to handle the solution?
   Yes

3.1.3 Behavioral Feasibility

The website's behavioural feasibility assessment includes determining the practicality and viability of its activities. The study's goal is to establish if the proposed website can be built, launched, and maintained efficiently. To provide a pleasant user experience, it also examines the integration of secure payment methods and user-friendly interfaces. The behavioural feasibility study gives crucial insights for decision-makers to determine the viability of building and sustaining a successful cloth e-commerce website by examining technological capabilities, staff skills, and probable problems. the organization is satisfied by the alternative solutions proposed by the software development team.

The site is behaviourally feasible because it allows for automated and streamlined processes, reducing the need for manual interventions in various aspects of the business. With advanced technologies and robust platforms, tasks such as inventory management, order processing, and payment transactions can be efficiently handled, improving behavioural efficiency and reducing human errors.

1) Is there sufficient support for the users?
   Yes
2) Will the proposed system cause harm?
   No

3.1.4 Feasibility Study Questionnaire

1) Project Overview?

SwapZone is an innovative e-commerce platform that revolutionizes the way users shop for clothing and fashion items online. With a diverse collection of clothing categories, styles, and accessories, SwapZone caters to various fashion preferences and trends. Users create accounts through a secure registration process, enjoying personalized shopping experiences and updates on the latest trends. The platform offers organized product categories, efficient search functionality, detailed product pages, high-quality images, and genuine user reviews and ratings. The platform ensures a smooth and secure online shopping process, with an integrated payment gateway and secure user information handling. The responsive website design allows users to shop seamlessly across various devices. SwapZone also offers a dedicated platform for sellers to showcase their products, manage profiles, and monitor inventory. The platform's excellent customer service is crucial for a thriving online clothing community. SwapZone features a transparent order tracking system, special discounts, and promotional offers during festive seasons or sales events. Overall, SwapZone is a fashion destination that celebrates individuality and self-expression.

2) To what extend the system is proposed for?

The system allows the customers to select the latest trend clothes based on their preferences from their own comfort zone itself in online mode. They can save hours roaming around looking for clothes at a physical store. They get a chance to compare the prizes with different sellers. Website ensures that only quality products are being sold, and hence, more reliable for customers

3) **Specify the Viewers/Public which is to be involved in the System?**

Potential customers looking to purchase the latest trends as well as small-scale vendors suffering from commission fees from outside vendors since they are unable to establish their own clothing stores

4) **List the Modules included in your System?**

Admin, users, sellers and delivery boy

5) **Identify the users in your project?**

Customer and small-scale vendors

6) **Who owns the system?**

Admin

7) **System is related to which firm/industry/organization?**

Textiles and E-commerce

8) **Details of person that you have contacted for data collection?**

Ambili Textiles, Mannar, Alappuzha

3.1.4.1 Questionnaire to collect details about the project?

1) **What types of textiles and fabrics do you specialize in selling at your physical store?**

We specialize in a wide range of fabrics, including cotton, silk, and synthetic materials.

2) **How do you decide on the pricing for the textiles and fabrics in your store?**

We consider factors such as the cost of materials, market prices, and demand when setting prices.

3) **What marketing strategies do you use to promote your textile shop and attract customers?**

We use local advertising, word-of-mouth, and occasional sales events to reach potential customers.

4) **How do you manage inventory and ensure a diverse selection of textiles for customers?**

We regularly update our inventory, analyze sales data, and reorder popular fabrics to ensure availability.

5) **What challenges do you face in operating a physical textile shop compared to online stores?**

Competition from online stores and managing physical store expenses are some challenges we encounter.

6) **How do you provide customer support and address inquiries from visitors to your textile store?**

We have well-trained staff who provide personalized assistance to customers and answer their queries.

7) **What feedback do you receive from customers, and how do you use it to improve your services?**

Customer feedback helps us understand their preferences and make adjustments to our product offerings.

8) **What payment methods do you accept at your textile store, and how do you ensure secure transactions?**

We accept cash and card payments, and we have secure POS systems in place for transactions.

9) **How do you handle product returns or exchanges for textiles and fabrics purchased from your store?**

We have a flexible return policy and assist customers with exchanges or refunds when needed.

10) **Do you offer any loyalty programs or discounts to reward repeat customers?**

Yes, we have a loyalty program that offers exclusive discounts and special offers to regular customers.

11) **What steps do you take to maintain the quality and authenticity of the textiles and fabrics you sell?**

We source our fabrics from trusted suppliers and conduct quality checks to ensure customer satisfaction.

12) **How do you market seasonal collections and festive textiles at your physical store?**

We create seasonal displays, run promotions, and advertise special collections during festivals.

13) Have you considered expanding your textile business to online platforms in addition to the physical store?

Yes, we are exploring the possibility of launching an online store to reach a wider audience.

14) How do you handle bulk orders from businesses or organizations that require textiles in large quantities?

15) We have a dedicated team to handle bulk orders and offer competitive pricing for such customers.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel Core i3

RAM - 4 G B

Hard disk - 1 T B

3.2.2 Software Specification

Front End	-	HTML, CSS
Back End	-	MYSQL, PHP
Database	-	MYSQL
Client on PC	-	Windows 7 and above.
Technologies used	-	JS, HTML5, AJAX, J Query, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 PHP

The programming language PHP may be used for a variety of projects, despite the fact that it was first designed for creating online applications. As of right moment, 2.1 million web servers and more than 244 million webpages use it. Rasmus Ledorf's 1995 PHP reference execution was developed by the PHP community. The PHP abbreviation now stands for "PHP: Hypertext Preprocessor," as opposed to the first meaning of "individual home page," as it did. The final web page may be produced by the PHP processor on the internet server by understanding the PHP code. Since PHP code may be precisely included into the raw HTML content, it is not necessary to call an external file to handle the data.

3.3.2 MySQL

The Prophet Organization claims and supports MySQL, a widely used open-source social database management system (RDBMS). It is widely used by web designers and is renowned for its unchanging quality, adaptability, and simplicity of use.

- **MySQL is a Database Management System**

A database is a structured collection of data that can be anything from a straightforward list of things to a complex system that can hold a massive quantity of data. In order to manage, retrieve, and alter data contained in a computer database, a database management system (DBMS), such as MySQL Server, is required. Due to computers' capacity to analyze and modify massive quantities of data, DBMS have grown to be an essential component of computing. They can be used independently or in conjunction with other software programs.

- **MySQL software is Open Source.**

"Open Source" in the context of software refers to software that is available for use and modification by anybody. Because MySQL is an open-source database management system, anybody may download and use the application without paying any fees. Additionally, customers have the option to examine the source code and modify the application to suit their own requirements. The GPL (GNU Common Open Permit), which outlines the terms and conditions for using and distributing the software, governs how it is released.

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**

Without the need for extensive maintenance, MySQL Server may be a reliable and effective database administration system suitable for use on desktop computers, tablets, or other mobile devices. It allows for adjustment of parameters to optimize execution according on available assets like Slam, CPU control, and I/O capacity. It is remarkably consistent with other apps and web servers. Client/server and added frameworks are supported by MySQL Server.

A multithreaded SQL server built into the MySQL database software can handle a large number of client programs, libraries, management tools, and multiple application programming interfaces (APIs). Expanding MySQL Server allows for the creation of a multi-threaded library that can be integrated with the software to create a self-contained framework that is smaller, faster, and more logical.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

The design phase is where any intended system or product begins to take shape. A well-executed design, which is a creative process, is essential to an effective system. It involves using a range of strategies and ideas to thoroughly define a process or system so that it may be put into practice. The design step in software engineering is crucial, regardless of the development methodology used. It serves as the technical backbone of the software engineering process and aims to create the architectural detail required to develop a system or product.

This software underwent a meticulous design phase that optimized every element of effectiveness, performance, and accuracy. During the design phase, a user-oriented document is transformed into a document for database or programmers.

4.2 UML DIAGRAM

Program frameworks are conceptualized, described, planned, and illustrated using a standardized language called Unified Modelling Language (UML). UML was developed by the Question Management Group (OMG), and the first draft of the UML 1.0 specification was published in January 1997. UML is distinct from programming languages like Java, C++, and COBOL. It could be a non-exclusive pictorial language used for program outlines as well as a visual showcasing language for computer program frameworks. Although UML is typically used to communicate with software frameworks, it may also be used for non-software frameworks like creating forms.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the relationships between customers and other external on-screen characters and the internal system components. Understanding, organizing, and coordinating a system's utilitarian requirements as viewed through the eyes of its users is the fundamental task of a use case diagram. Use case diagrams are typically created using the Unified Modelling Language (UML), a standard language for modeling real-world objects and systems.

Use cases may be used to accomplish a variety of framework goals, such as establishing fundamental requirements, validating equipment plans, testing and researching programs, developing online provide help references, or fulfilling client support duties. Customer service, product sourcing, catalog updating, and payment processing are practically a few examples of use cases in the context of item deals.

A use case diagram is made consisting of the system boundaries, actors, use cases, and their linkages. The system boundary determines the limits of the system in relation to its surroundings. Actors are frequently characterized in terms of the roles they take on and how those roles represent the individuals or systems that interact with the system. Use cases are the specific actions or behaviors that actors do when using the technology or while nearby. The image also depicts the relationships between actors and use cases in addition to the use cases themselves.

Use case diagrams are visual representations that are used to record a system's functional needs. It's crucial to adhere to these rules while creating a use case diagram to create an effective and efficient diagram:

- Pick names for use cases that are realistic representations of the functions they provide.
- Give actors proper names to make it easier to understand what part they play in the system.
- Verify that the diagram accurately depicts all links and dependencies.
- Since the major objective is to identify the fundamental criteria, refrain from listing every potential link.
- When required, take notes to help you remember key details.

By adhering to these rules, we may produce a use case diagram that is precise in capturing the functional needs of the system.

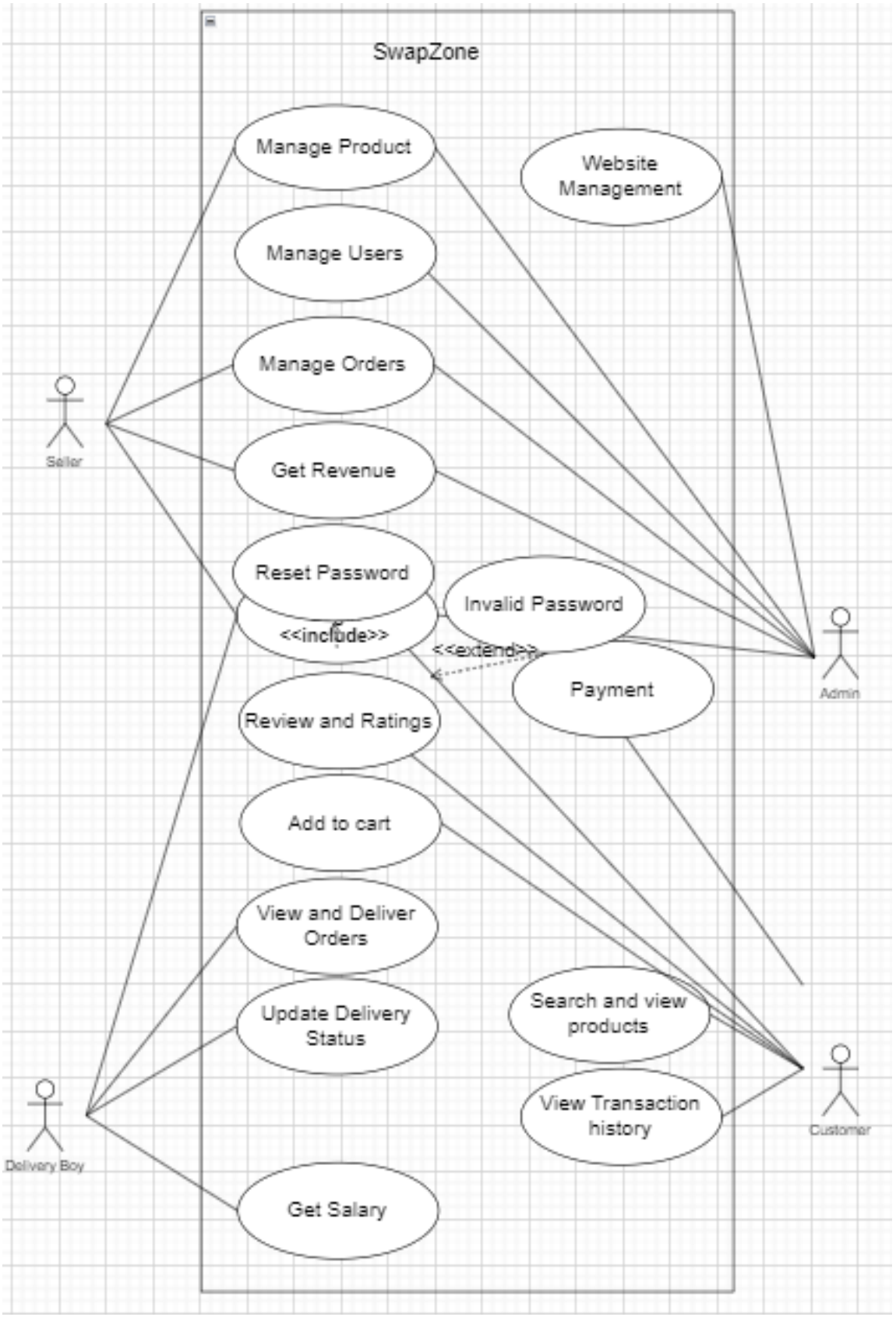


Fig 1: Use Case diagram for SwapZone

4.2.2 SEQUENCE DIAGRAM

A sequence diagram, which is a type of interaction diagram, displays the interactions between different system components in chronological order. It shows how several items may interact with one another through a string of messages. Event scenarios and event scenarios diagrams are other names for these pictures. Sequence diagrams are commonly used in software engineering to explain and understand the requirements of both new and legacy systems. They assist in the discovery of systemic problems and the display of object control linkages.

Sequence Diagram Notations –

i. **Actors:** An actor in UML represents a role that communicates with the system's objects. Actors frequently exist outside of the system represented by the UML diagram. A variety of roles, such as those of external subjects or human users, can be played by actors. Actors are shown in UML diagrams using a stick person notation. A sequence diagram may have several actors, depending on the circumstance being depicted.

ii. **Lifelines:** In a sequence diagram, a lifeline is a vertical dashed line that symbolizes an object's lifetime during the interaction. Each lifeline is tagged with the participant's name and serves to symbolize a specific participant in the series of events. The lifeline, which is shown as a vertical line running from the participant's activation point to its deactivation point, displays the participant's history of events.

iii. **Messages:** Messages, which describe the interactions and communication between objects or components in a system, are an important part of sequence diagrams. Synchronous and asynchronous communications, create and delete messages, self-messages, reply messages, found messages, and lost messages are some of the categories they fall under. Guards are often employed to simulate constraints on message flow.

iv. **Guard:** When a given condition is satisfied, guards—which are used to describe conditions in UML—are used to stop the flow of messages. This function is crucial for informing software developers of any restrictions or limitations related to a system or specific process.

Uses of sequence diagram –

- Modeling and illustrating the logic of difficult actions, processes, or functions.
- Displaying UML use case diagrams in depth.
- Recognizing the precise operation of present or upcoming systems.
- Making a visual representation of the flow of information and tasks inside a system.
- The flow of interactions between items in a system may be represented using sequence diagrams, which can be helpful for both businessmen and software developers in understanding and communicating system needs and behavior.

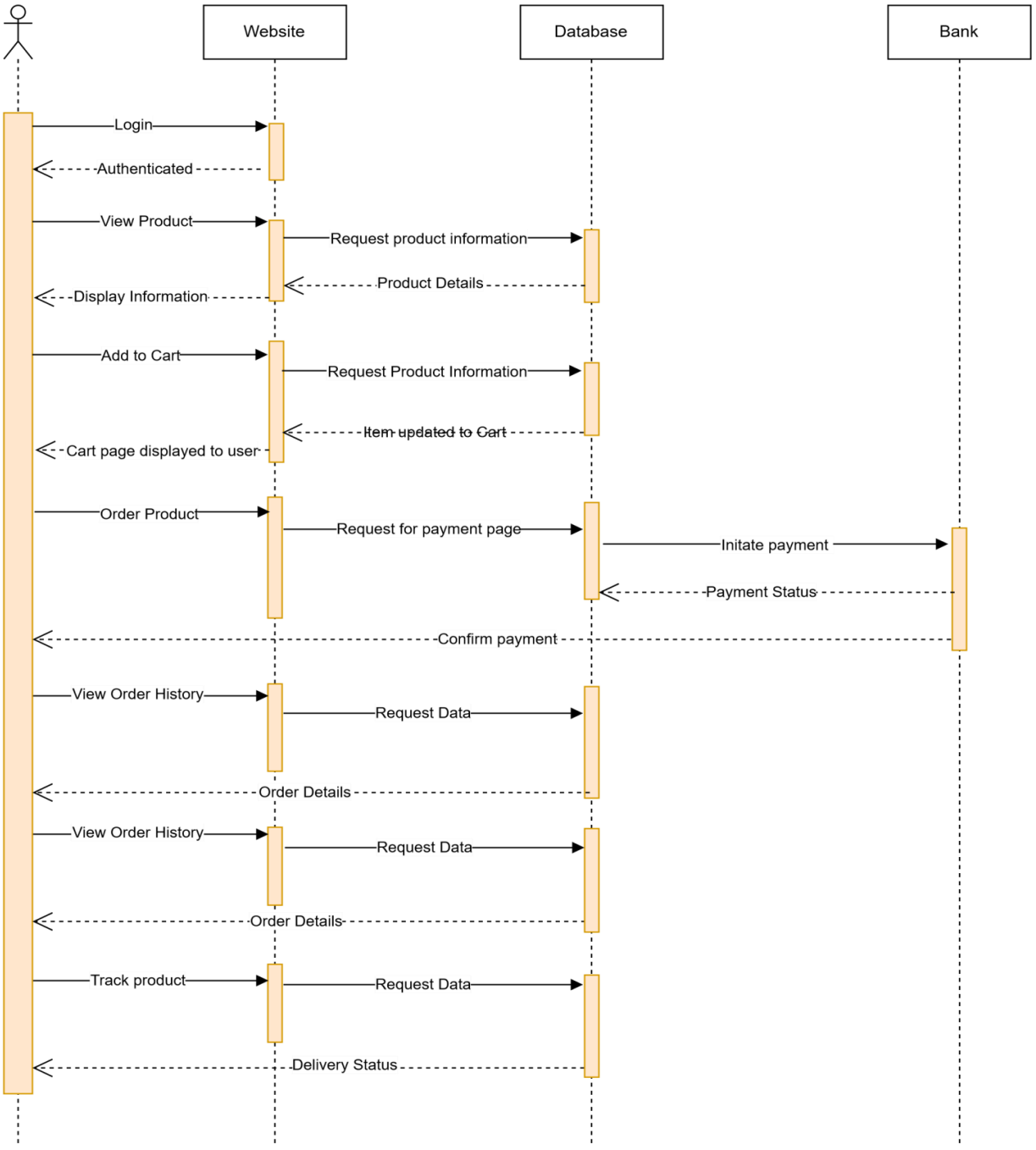


Fig 2: Sequence Diagram for SwapZone

4.2.3 State Chart Diagram

A state diagram is a picture that demonstrates the many states an item might be in and how it can change between those states. It is frequently made using the Unified Modeling Language (UML). State machine diagrams and state chart diagrams are other names for it.

The State Chart Diagram in UML is a behavioral diagram that illustrates how an object or system behaves over time. It contains a number of things, including:

- Initial State: A solid black circle is used to depict this state, which symbolizes the starting position of the system or item.
- State – This element, which is represented as a rectangle with rounded corners, depicts the status of the system or item at a certain moment in time.
- Transition - This element, which is represented by an arrow, depicts how the system or item changes from one state to another.
- Event and Action - A transition is triggered by an event, and the behavior or result of the transition is an action.
- Signal - A message or trigger that an event sends to a state, triggering a transition, is known as a signal.
- Final State - A complete black circle with a dot within is used to symbolize the Final State element at the end of the State Chart Diagram. It shows that the system's or object's behavior has ended.

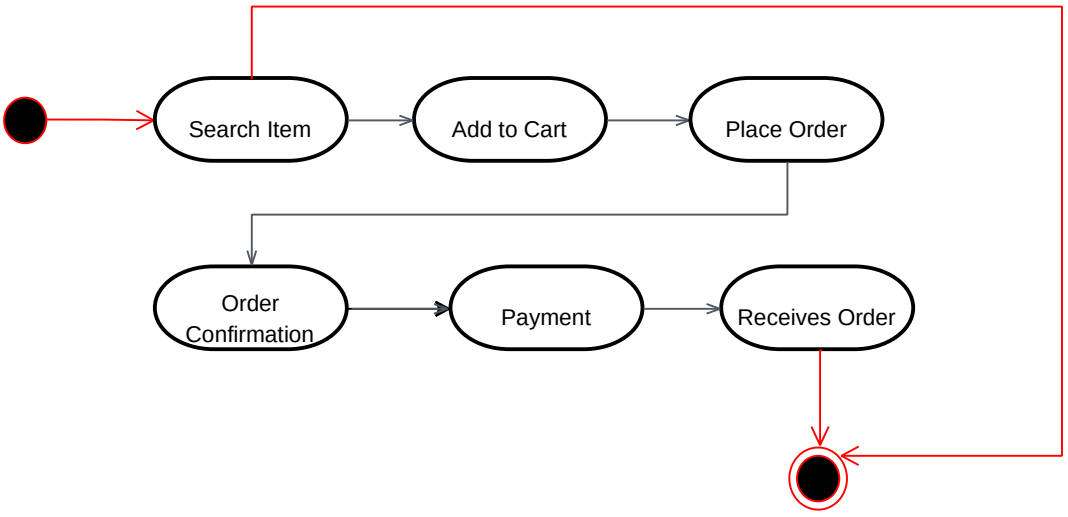


Fig 3: State Chart Diagram for SwapZone

4.2.4 Activity Diagram

A graphic depiction of a workflow that demonstrates how one activity flows into another is called an activity diagram. A system operation is an action, and in the control flow, one operation follows another. Activity diagrams employ multiple functions like branching, joining, etc. to manage all sorts of flow control. A flow might be parallel, concurrent, or branched. A sort of behavior diagram that depicts a system's behavior is an activity diagram. They display the control flow from the beginning to the finish of the activity as well as the many decision-making pathways that might be taken.

The key components of an activity diagram are:

- Initial node - A black circle designates the activity diagram's starting point.
- Activity - An action or task carried out by the entity or system, symbolized by a rectangle with rounded corners.
- Control flow: Depicted by an arrow, it shows the order in which a system or entity performs its various operations.
- Decision node - A diamond-shaped node that represents a choice or branching point in the activity flow.
- Merge node - Displayed as a diamond-shaped node with a plus sign inside it, this node is used to combine many activity flow branches into a single flow.
- Fork node—represented by a solid black circle with several arrows—is used to divide the activity flow into multiple parallel flows.
- Join node—shown as a solid black circle with several arrows pointing towards it—is used to combine multiple parallel flows into a single flow.
- Final node - The activity diagram's conclusion, represented by a black circle with a dot within.
- Object flow - A dashed arrow depicts the movement of items or data between activities.

Activity diagrams help stakeholders and project team members understand complicated processes, spot possible problems, and convey process flows.

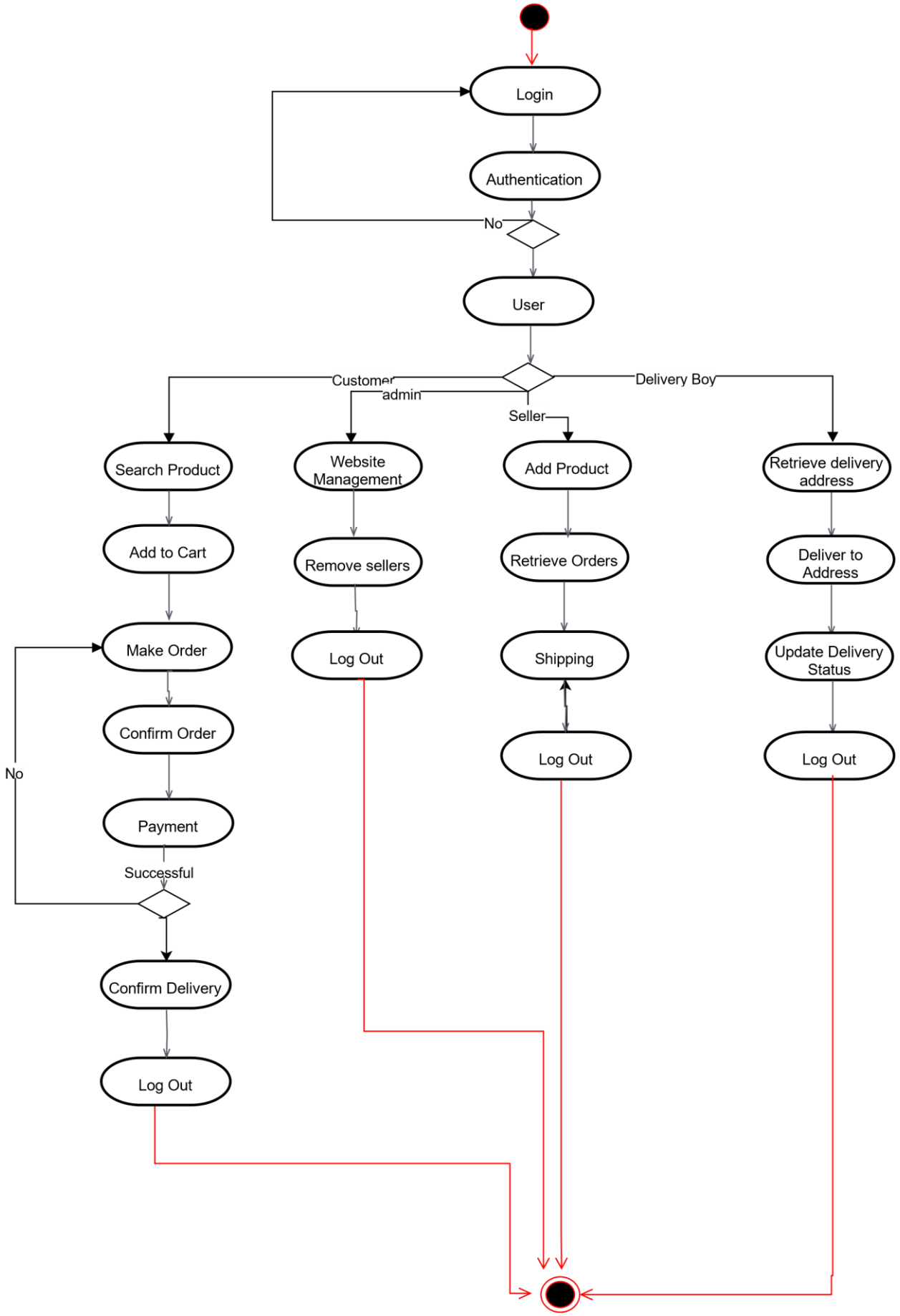


Fig 4: Activity Diagram for SwapZone

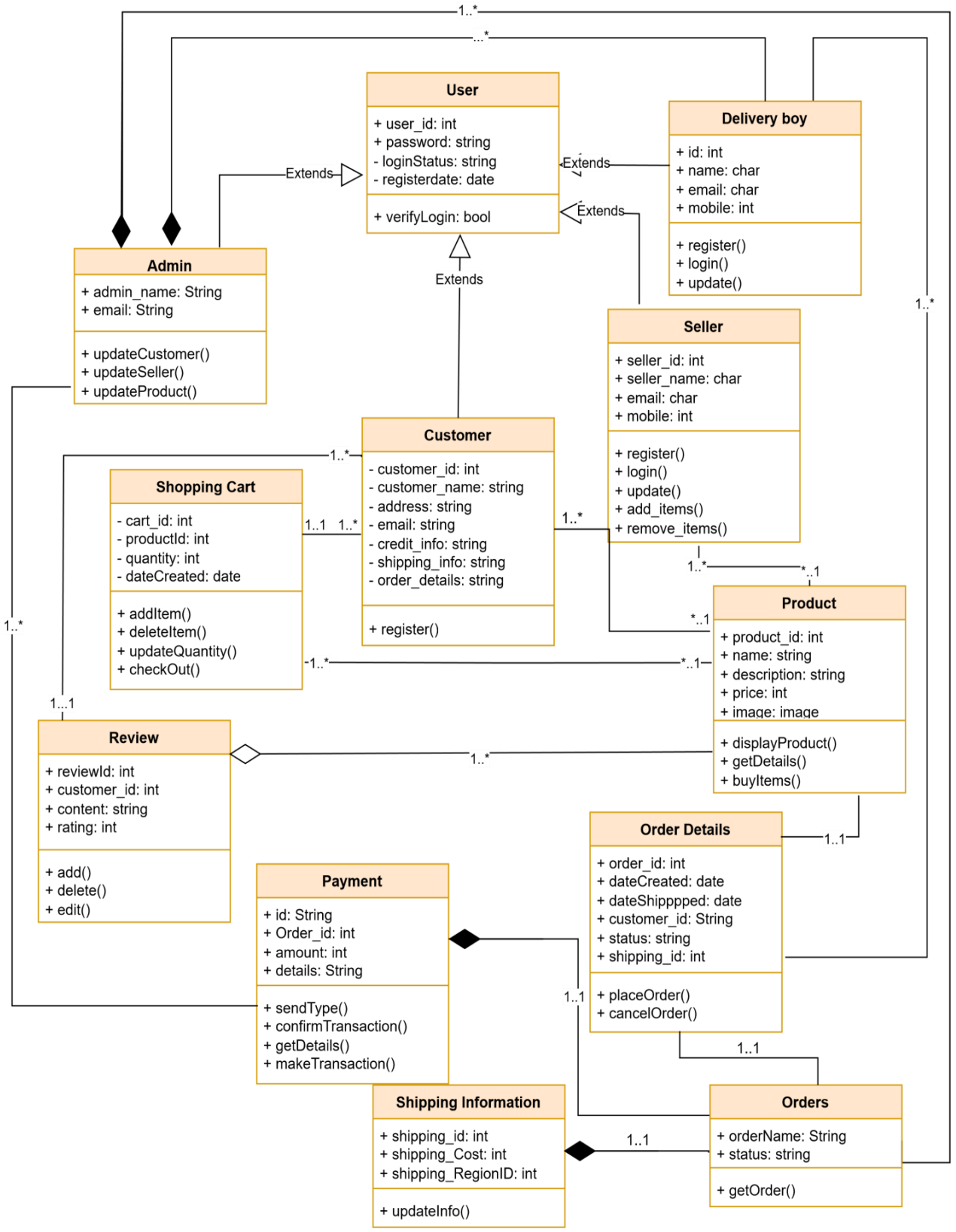
4.2.5 Class Diagram

The class diagram is a key element of object-oriented modeling and acts as the main tool for conceptually modeling an application's structure. Class diagrams can also be utilized for thorough modeling that can be converted into computer code. They may be used for data modeling as well. A key part of the UML is the class diagram, which is used to depict classes, objects, interfaces, and the connections between and characteristics of each in a system.

The following are some crucial elements of a class diagram:

- Class: A rectangle containing the class name, properties, and methods serves as a blueprint or template for building objects.
- Interface: An interface is a group of abstract methods that define the terms of a class's agreement with the outside world. It appears as a circle with the name of the interface within.
- Object: It is a state- and behavior-filled instance of a class. The object name is enclosed inside a rectangle for representation.
- Association: An association is a connection or link between two classes that is shown as a line with optional directionality, multiplicity, and role names.
- Aggregation: On the aggregator side, the connection is depicted as a diamond shape, with the whole (aggregator) made up of pieces (aggregates).
- Composition: On the aggregator side, it is shown as a filled diamond shape and is a stronger kind of aggregation where the components cannot exist without the whole.
- Inheritance: This relationship between a superclass and its subclasses is symbolized by a line with an open arrowhead pointing from the subclass to the superclass. It indicates a "is-a" relationship.
- Dependency: This connection, which is shown as a dashed line with an arrowhead pointing from the dependent class to the independent class, is one in which a change in one class may have an impact on the other class.
- Multiplicity: It represents the number of instances of a class that can be associated with another class and is represented as a range of values near the association or aggregation line.

Class diagrams are crucial for the design and modeling of object-oriented software systems because they give a clear visual depiction of the system's functionality, organizational structure, and object interactions. They make it easier to create and maintain software and to communicate with other team members.


User
+ user_id: int
+ password: string
- loginStatus: string
- registerdate: date
+ verifyLogin: bool
Delivery boy
+ id: int
+ name: char
+ email: char
+ mobile: int
+ register()
+ login()
+ update()
Admin
+ admin_name: String
+ email: String
+ updateCustomer()
+ updateSeller()
+ updateProduct()
Seller
+ seller_id: int
+ seller_name: char
+ email: char
+ mobile: int
+ register()
+ login()
+ update()
+ add_items()
+ remove_items()
Customer
- customer_id: int
- customer_name: string
- address: string
- email: string
- credit_info: string
- shipping_info: string
- order_details: string
+ register()
Shopping Cart
- cart_id: int
- productId: int
- quantity: int
- dateCreated: date
+ addItem()
+ deleteItem()
+ updateQuantity()
+ checkOut()
Product
+ product_id: int
+ name: string
+ description: string
+ price: int
+ image: image
+ displayProduct()
+ getDetails()
+ buyItems()
Review
+ reviewId: int
+ customer_id: int
+ content: string
+ rating: int
+ add()
+ delete()
+ edit()
Order Details
+ order_id: int
+ dateCreated: date
+ dateShipppped: date
+ customer_id: String
+ status: string
+ shipping_id: int
+ placeOrder()
+ cancelOrder()
Payment
+ id: String
+ Order_id: int
+ amount: int
+ details: String
+ sendType()
+ confirmTransaction()
+ getDetails()
+ makeTransaction()
Shipping Information
+ shipping_id: int
+ shipping_Cost: int
+ shipping_RegionID: int
+ updateInfo()
Orders
+ orderName: String
+ status: string
+ getOrder()
Extends
Extends
Extends
Extends
1..*
1..1
*..1

4.2.6 Object Diagram

In object-oriented modeling, class diagrams and object diagrams are closely connected. Class diagrams, which show a point in time snapshot of the system, are examples of object diagrams. The principles and notation used to illustrate a system's structure are the same in both kinds of diagrams. Object diagrams depict a collection of items and their connections at a certain moment in time, whereas class diagrams are used to model the structure of the system, including its classes, properties, and methods.

A sort of structure diagram used in UML called an object diagram displays instances of classes and their connections. The essential elements of an object diagram are as follows:

- Object: An object is an instance of a class that stands in for a particular system entity. The object name is enclosed inside a rectangle for representation.
- Class: A class is an outline or model for building objects that specifies the characteristics and methods of the object. The class name, properties, and methods are shown in three compartments inside a rectangle.
- Link: A link is a connection or association that exists between two items. It appears as a line joining two items with omitted labels.
- Attribute: A property or characteristic of an item that represents its state is called an attribute. It is shown inside the object rectangle as a name-value pair.
- Value: A value is a particular use or configuration of an attribute. Within the attribute name-value pair, it is shown as a value.
- Operation: An item can engage in a behavior or activity known as an operation. Inside the class rectangle, it appears as a method name.
- Multiplicity: The quantity of instances of one class that may be linked to another class is referred to as multiplicity. It is shown as a range of numbers close to the link between objects (e.g., 0..1, 1..*, etc.).

Object diagrams help to visualize the relationships between objects and their attributes in a system. They are useful for understanding the behavior of a system at a specific point in time and for identifying potential issues or inefficiencies in the system.

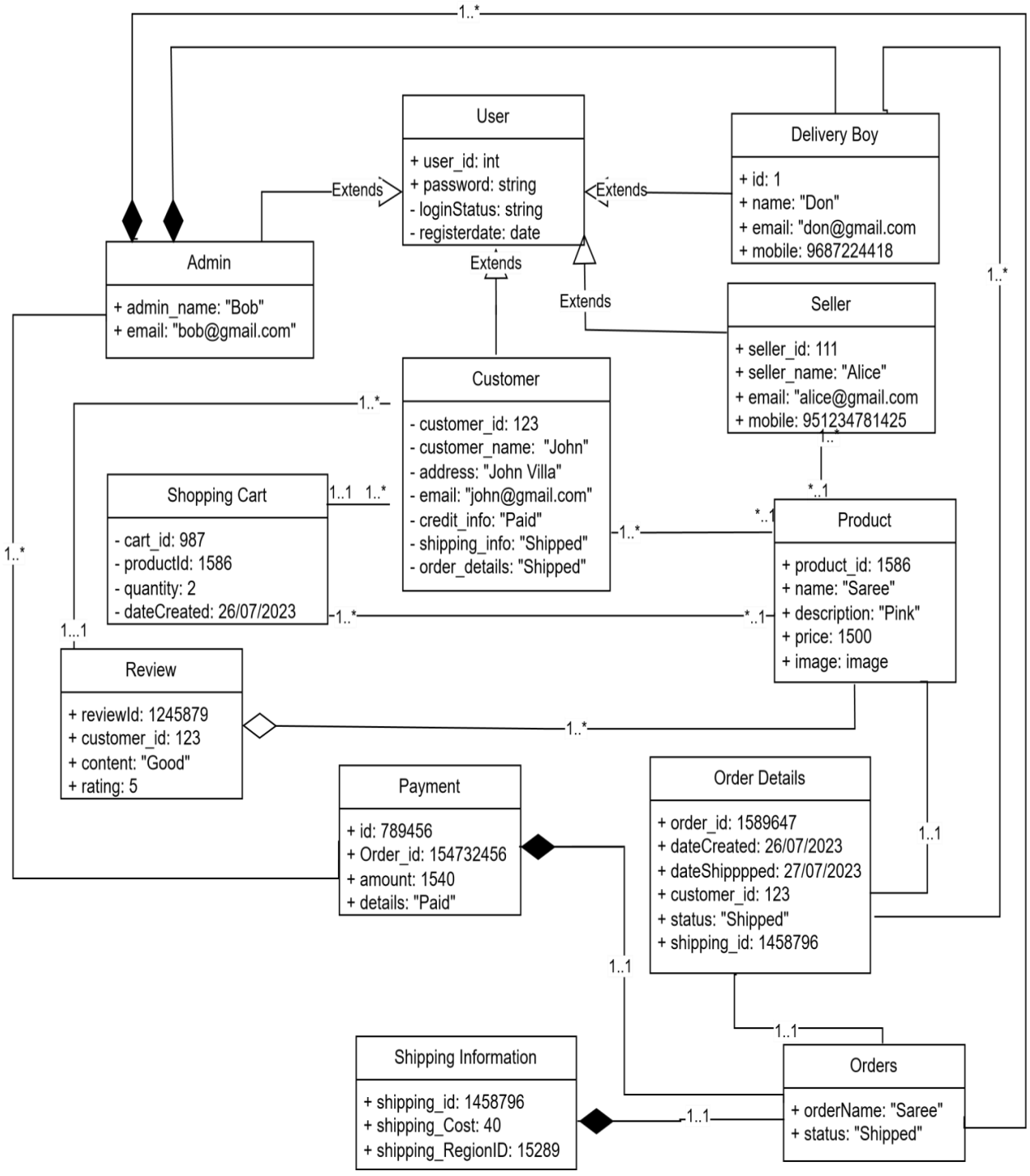


Fig 6: Object diagram for SwapZone

4.2.7 Component Diagram

In the UML, a component diagram shows how different components are linked together to form bigger components or software systems. It is a useful tool for illustrating the structure of intricate systems with numerous parts. Developers may quickly see the internal workings of a software system and comprehend how various components interact to complete a given job by utilizing component diagrams.

Its key components include:

- Component: A system's functionally modular and encapsulated unit that provides interfaces for interacting with other components. It is shown as a rectangle that contains the component name.
- Interface: An agreement between a component and its surroundings or other components that outlines a group of techniques that other components may utilize. It appears as a circle with the name of the interface within.
- Port: A port is a location where a component interacts with the environment or other components. It appears as a little square on a component's edge.
- Connector: An interface that allows communication or data exchange between two components. It is shown as a line with optional labels and adornments.
- Dependency is a connection between two components in which one depends on the other for functionality or implementation. A dashed line with an arrowhead pointing from the dependent component to the independent component serves as its visual representation.
- Association: A connection or link between two elements in a relationship. With optional directionality, multiplicity, and role names, it is shown as a line joining two components.
- Supplied/needed Interface: A supplied interface is one that a component provides to other components, whereas a needed interface is one that a component requires from other components in order for it to operate effectively. Half-circles and lollipops, respectively, are used to depict this.

When modeling the architecture of a software system, component diagrams are helpful because they may point out possible problems and suggest design changes. They may also be used to explain a system's behavior and structure to stakeholders, including programmers and project managers.

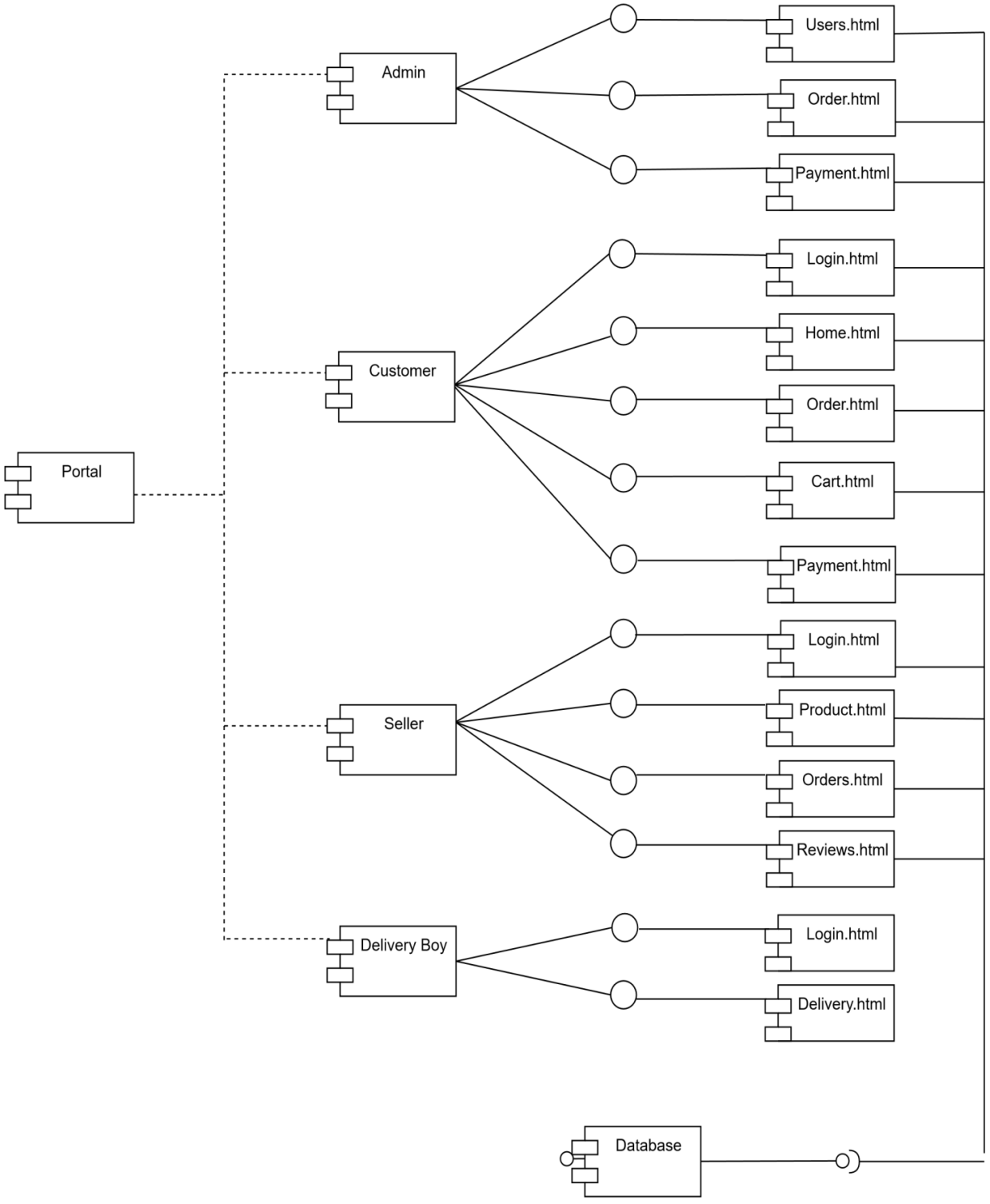


Fig 7: Component Diagram for SwapZone

4.2.8 Deployment Diagram

A particular kind of UML diagram that focuses on the actual hardware needed to distribute software is called a deployment diagram. It involves nodes and their connections and offers a static picture of a system's deployment. The deployment diagram illustrates how the program will be run on nodes by mapping the software architecture to the physical system architecture. The connections between the nodes are depicted using communication routes. The hardware topology is highlighted in the deployment diagram more so than in other UML diagram types, which emphasize a system's logical components.

The key components of a deployment diagram are:

- A component or artifact is deployed on a node, which is a real or virtual computer. It is shown as a box with the name of the node within.
- Component - A component is a piece of software that carries out a particular task or offers a particular service. It appears as a rectangle with the name of the component within.
- Artifact - An artifact is a tangible piece of data that a component uses or creates. It is symbolized as a rectangle with the name of the artifact within.
- Deployment Specification - A deployment specification outlines the steps involved in deploying a component or artifact on a node. It contains details about the component or artifact's location, version, and configuration settings.

The physical architecture of a system may be visualized using deployment diagrams, which can aid in spotting any possible problems or deployment process bottlenecks. Additionally, they support the deployment strategy planning and hardware resource optimization.

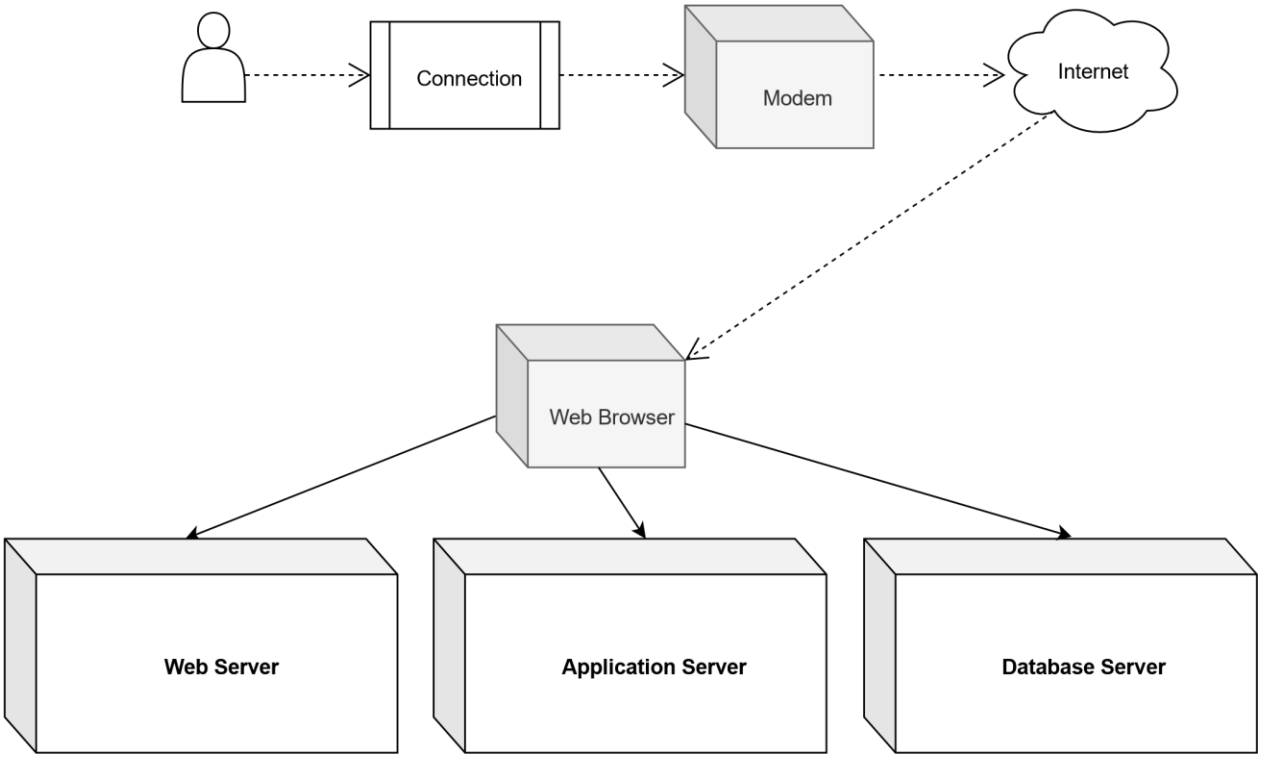


Fig 8: Deployment Diagram for SwapZone

4.2.9 Collaboration Diagram

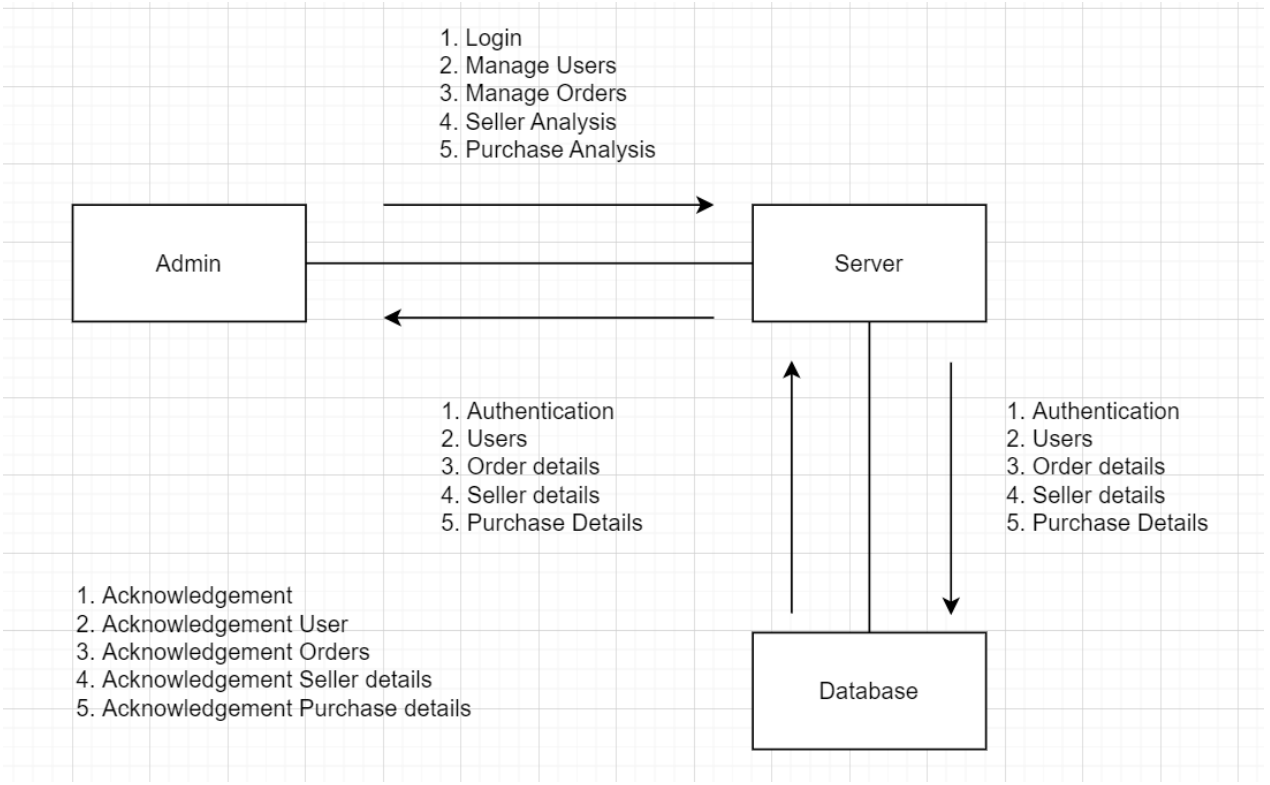
An illustration of the relationships between objects in a system is called a cooperation diagram. While it conveys the same knowledge in a different way, it is comparable to a sequence diagram. It illustrates the layout of the items in the system, not how messages flow between them. The reason for this is that object-oriented programming, on which collaboration diagrams are built, allows objects to have an assortment of properties and be related to one another. Collaboration diagrams, then, act as a visual depiction of a system's object architecture.

Key components in Collaboration Diagram include:

- Objects: Symbols for objects are designated by their name and class, which are separated by a colon. Objects are used to represent class instances in collaboration diagrams and provide information about their name and class. A class does not always need to have an object representation, and a class can contain more than one object. Classes are provided after objects have been formed. It is crucial to give items names in order to distinguish them from one another.
- Actors: Because they commence the interaction, actors are important components of the cooperation diagram. Every performer has a unique name and function. The use case is initiated

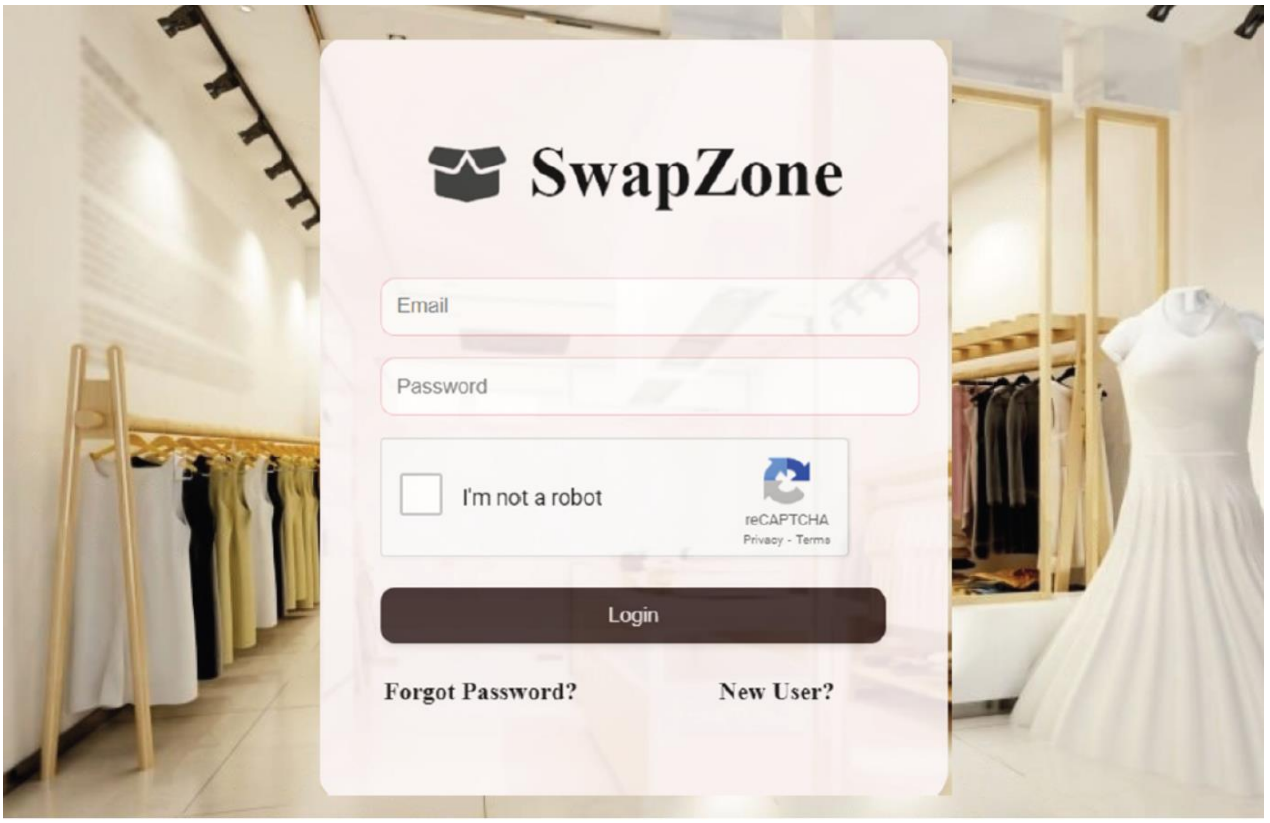
by one person in the diagram.

- Links: Objects and actors are connected by links, which are examples of association. They serve as an illustration of the connection between the items that transmit messages. Solid lines serve as links, facilitating object navigation to other items.
- Messages: Identified by a sequence number, messages are a means of communication between objects carrying information. They are transmitted from the sender to the recipient and shown as labeled arrows that are positioned close to the connection. The message must be understood by the recipient and the direction must be navigable in that particular direction.

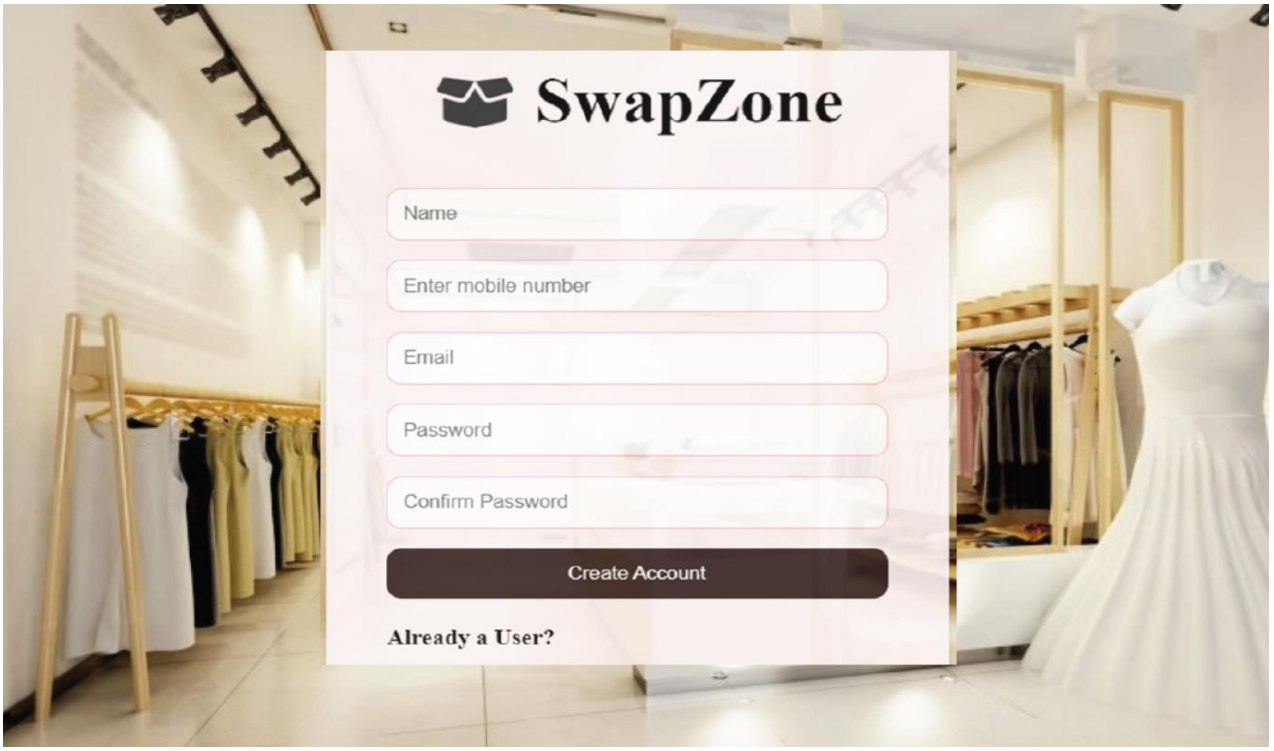


4.3 USER INTERFACE DESIGN USING FIGMA

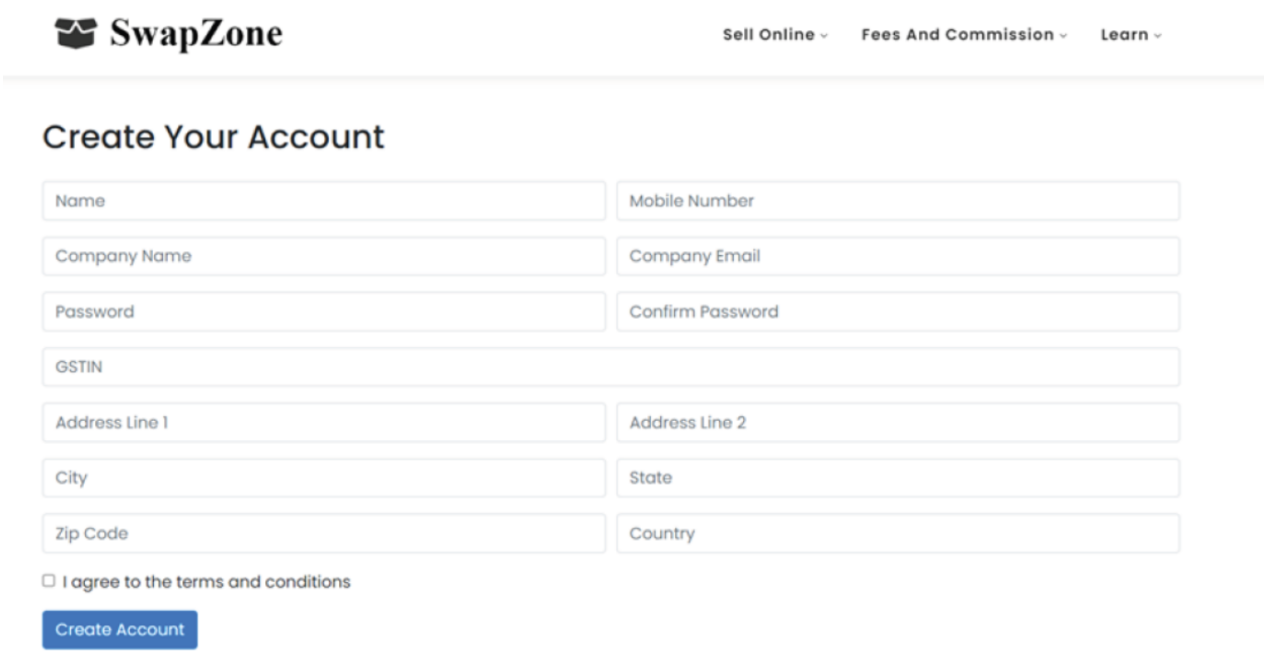
1) Form Name: Login Page



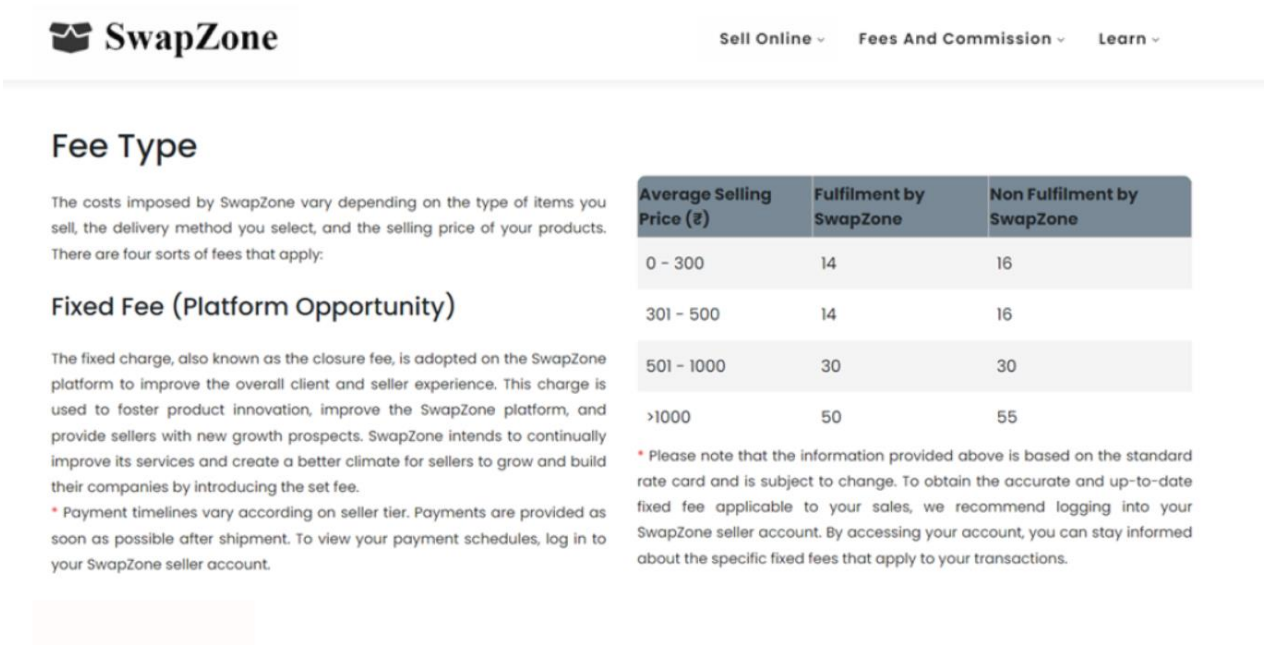
2) Form Name: Customer Registration Page



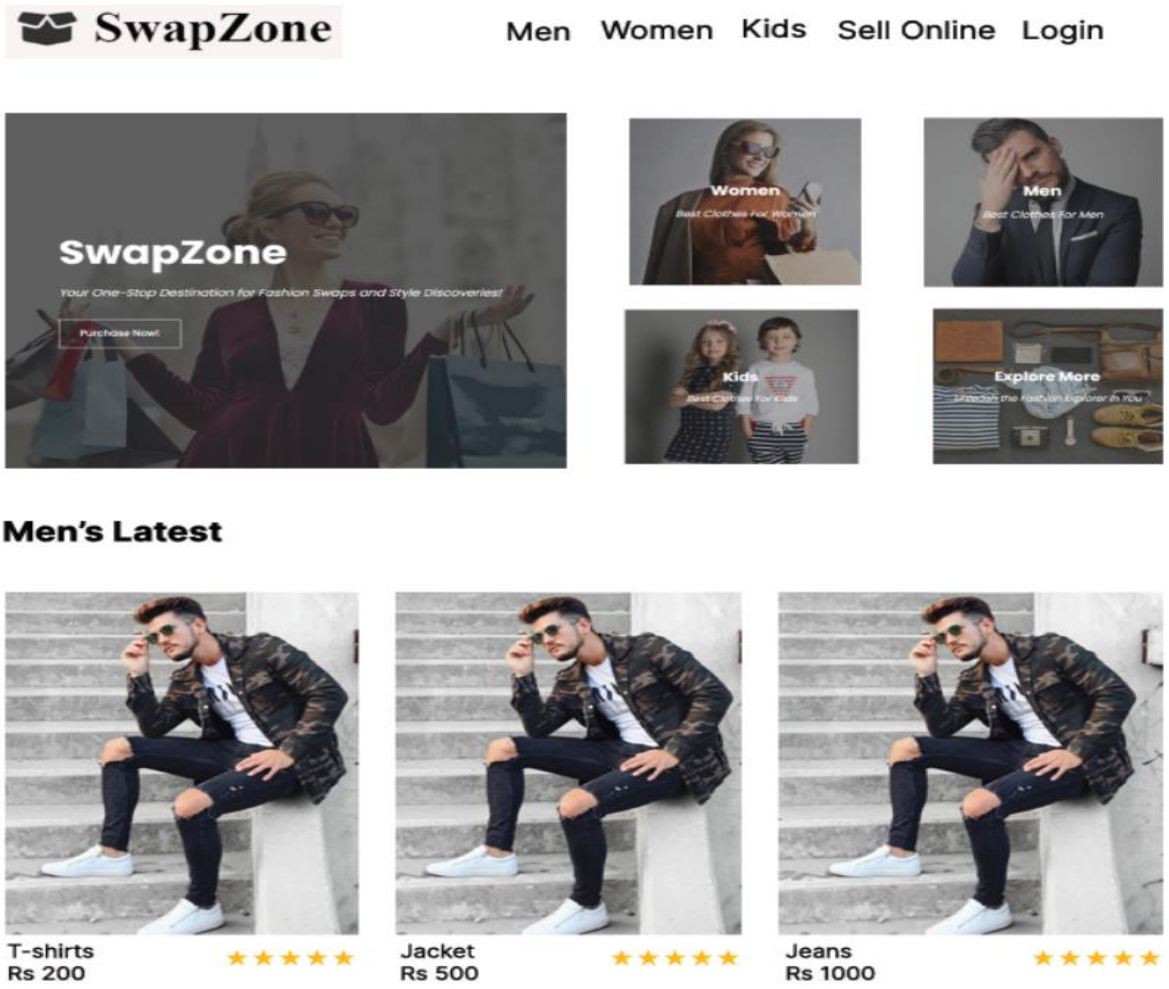
3) Form Name: Seller Registration Page



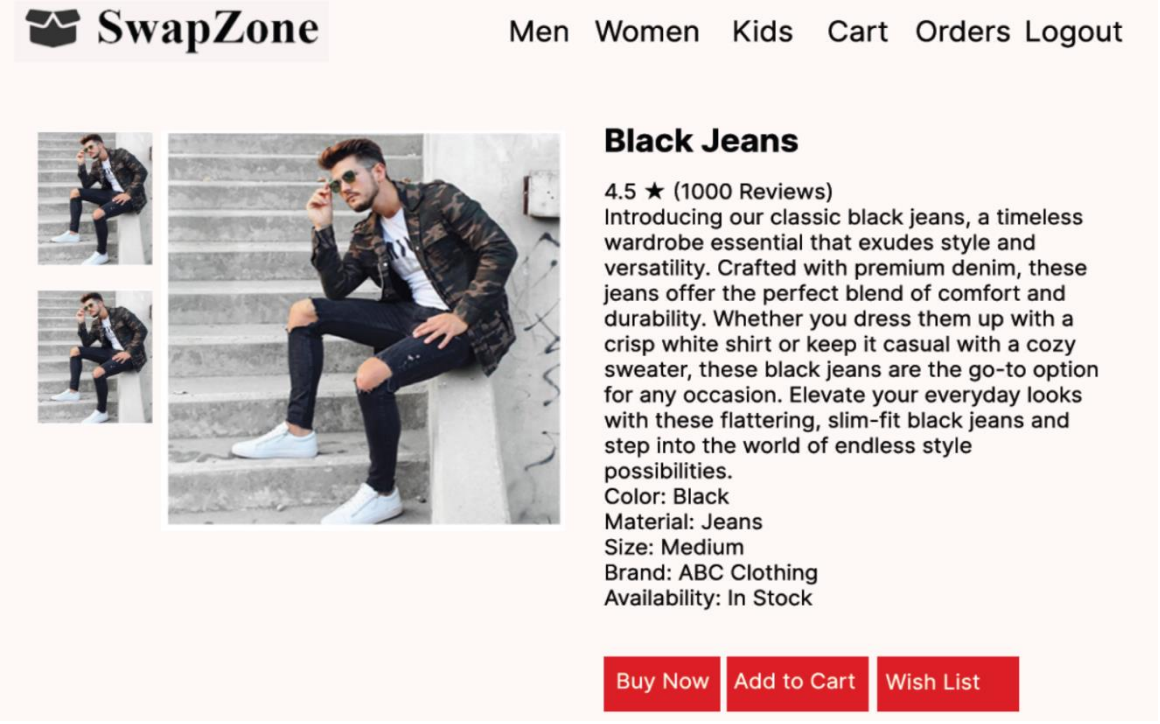
4) Form Name: Seller Index Page



5) Form Name: Customer Index Page

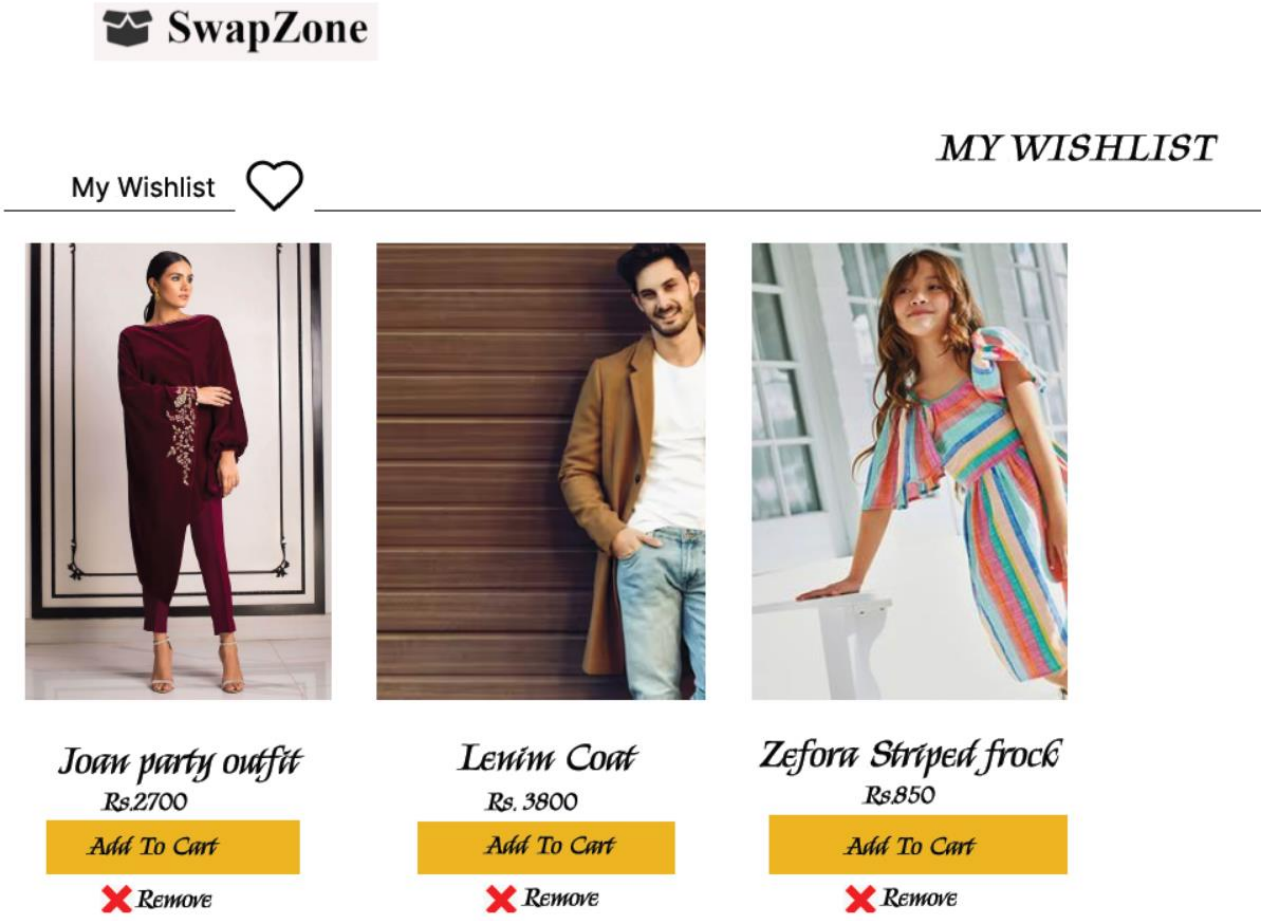


6) Form Name: Product Detail Page

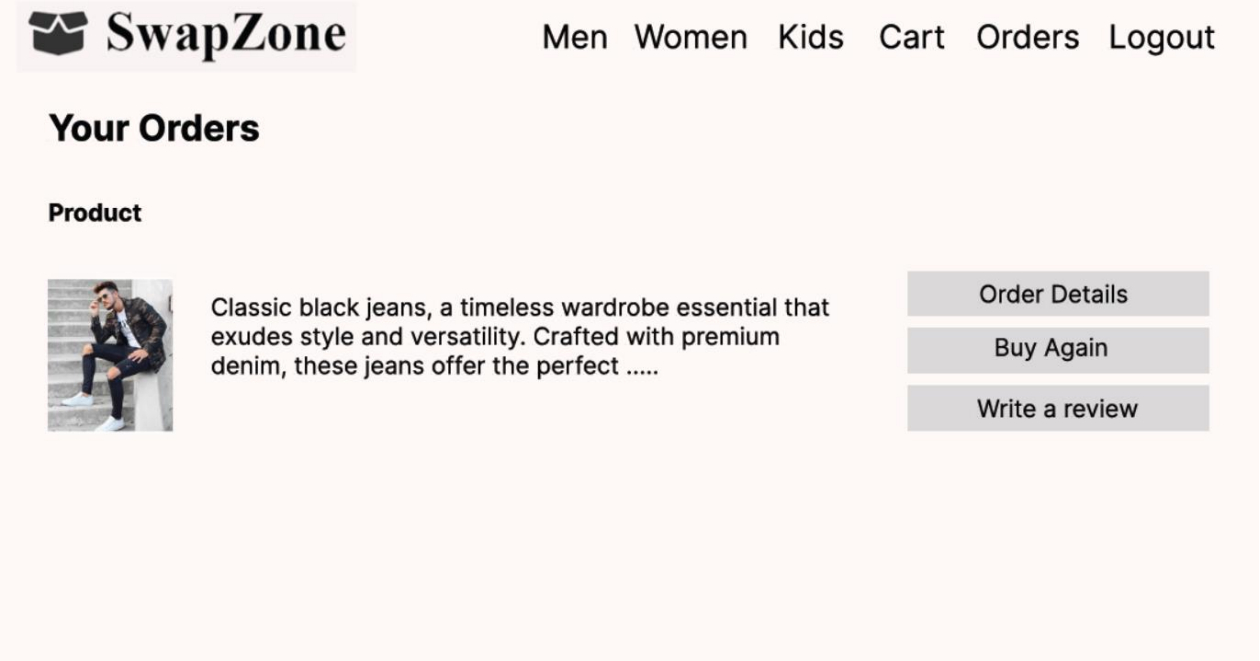


7) Form Name: Cart Page

8) Form Name: Wishlist Page



9) Form Name: Order Page



4.3 DATABASE DESIGN

A database is a set of information that has been arranged so that it can be easily accessed, managed, and updated. Any database might have information security as one of its primary goals. There are two stages in the database design process. To build a database that as clearly as possible satisfies user needs, user requirements are obtained in the first step. This is done independently of any DBMS and is referred to as information-level design. The design is changed from an information-level design to a particular DBMS design that will be utilized to build the system in the second step. The physical-level design stage is when the properties of the particular DBMS are taken into account. In addition to system design, database design also has two main objectives.

4.4.1 Relational Database Management System (RDBMS)

A common kind of database, known as a relational database management system (RDBMS), arranges data into tables to make connections with other stored data sets easier. The rows in tables, which are each referred to as records, can range in number from a few hundred to millions. A column heading is an attribute, a row is referred to as a tuple, and the table is referred to as a relation in formal relational model language. Multiple tables with unique names make up a relational database. A set of associated values is represented as a row in a table.

Connections between tables are pre-established in relational databases to guarantee the consistency of both referential and entity connections. Choosing a data type from which the domain's data values are derived is a popular technique to define a domain, which is a collection of atomic values. Giving the domain a name makes it simpler to comprehend the values it contains. A relation's values are all atomic and cannot be further subdivided.

Keys are used in relational databases to create table associations, with the main key and foreign key being the two most crucial types. These keys can be used to create connections between entity integrity and referential integrity. Referential integrity assures that every different foreign key value must have a corresponding primary key value in the same domain, whereas entity integrity ensures that no primary key may contain null values. Other key kinds include super keys and candidate keys, among others.

4.4.2 Normalization

To ensure that future modifications have little impact on the data structures, the data are organized into the smallest feasible groups. the methodical process of standardizing data structures in a way that promotes integrity and reduces redundancy. Superfluous fields are eliminated and a large table is reduced into multiple smaller ones using the normalizing process. Using it also prevents anomalies in insertion, deletion, and updating. The traditional approach of data modeling makes use of the concepts of keys and relationships. To ensure that future modifications have little impact on the data structures, the data are organized into the smallest feasible groups. the methodical process of standardizing data structures in a way that promotes integrity and reduces redundancy. Superfluous fields are eliminated and a large table is reduced into multiple smaller ones using the normalizing process. Using it also prevents anomalies in insertion, deletion, and updating. The traditional approach of data modeling makes use of the concepts of keys and relationships.

In database architecture, normalization is a technique that seeks to properly arrange data into tables and columns so that the user may readily connect it to the data. Redundancy in data is eliminated, which relieves pressure on computer resources. The key steps in normalizing are as follows:

- Normalizing the data
- Choosing appropriate names for tables and columns
- Choosing the correct names for the data

By following these steps, a developer can create a more efficient and organized database that is easier to manage and maintain.

First Normal Form

According to the First Normal Form (1NF), a table's attributes can only have atomic or indivisible values. The usage of nested relations, or relations inside relations, as attribute values in tuples is forbidden. Data must be transferred into distinct tables with data of a comparable kind in each table to meet 1NF, and each table must have a primary key or foreign key depending on the project's specifications. For each non-atomic property or nested relation, this technique generates new relations, eliminating repetitive sets of data. Only when a relation complies with the requirements requiring the main key to be present alone is it deemed to be in 1NF.

Second Normal Form

According to the second normal form (2NF) rule of database normalization, non-key properties in

a relation with a composite primary key should not be functionally dependent on just one portion of the primary key. To put it another way, every non-key characteristic need to be dependent on the complete main key, not just a portion of it. To do this, the table must be broken down, and new connections must be created for each subkey and the dependent attributes. The association with the original primary key and any properties that are completely functionally dependent on it must be preserved. A relation is only considered to be in 2NF if its primary key meets all 1NF requirements and all of its non-primary key attributes are solely dependent on the main key

Third Normal Form

A relation must not contain any non-key attributes that are functionally determined by another non-key attribute or combination of non-key attributes in order to satisfy the third normal form (3NF). As a result, the main key shouldn't have any transitive dependencies. To accomplish 3NF, we breakdown the relation into non-key characteristics that functionally determine other non-key attributes, and we set up a new relation based on this new relation. This aids in removing any dependencies that include more than simply the primary key. If a relation meets the requirements of 2NF and, in addition, has non-key qualities that are independent of one another, it is regarded as a relation in 3NF.

4.4.3 Sanitization

Sanitizing data involves taking out any illegitimate characters or values. Sanitizing user input is a typical operation in online applications to guard against security flaws. To clean and check many sorts of external input, including email addresses, URLs, IP addresses, and more, PHP has a built-in filter extension. These filters are intended to accelerate and simplify data sanitization. For instance, the PHP filter extension has a function that, depending on a flag, can filter out all characters other than letters, digits, and specific special characters (!#$%&'*+-=?_'|@.[]).

Web applications often receive external input from various sources, including user input from forms, cookies, web services data, server variables, and database query results. It is important to sanitize all external input to ensure that it is safe and does not contain any malicious code or values.

4.4.4 Indexing

An index is a database structure that enhances the speed of table operations. Indexes can be created on one or more columns to facilitate quick lookups and efficient ordering of records. When creating an index, it's important to consider which columns will be used in SQL queries and to create one or more

indexes on those columns. In practice, indexes are a type of table that store a primary key or index field and a pointer to each record in the actual table. Indexes are invisible to users and are only used by the database search engine to quickly locate records. The CREATE INDEX statement is used to create indexes in tables.

When tables have indexes, the INSERT and UPDATE statements take longer because the database needs to insert or update the index values as well. However, the SELECT statements become faster on those tables because the index allows the database to locate records more quickly.

4.5 TABLE DESIGN

1. tbl_user

Primary key: **user_id**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	user_id	INT	Primary Key, Auto Increment	User Id
2.	email	varchar (20)	Unique, Not Null	Email
3.	password	varchar (20)	Not Null	Password
4.	role	int (10)	Not Null	Admin/ Seller/ Customer
5.	status	int (10)	Not Null	Status-active/blocked

2. tbl_admin

Primary key: **admin_id**
Foreign key: **user_id** references **tbl_user**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	admin_id	INT	Primary Key, Auto Increment |	Admin Id
2.	user_id	INT	Foreign Key referencing Users	User Id
3.	email	VARCHAR(30)	Unique, Not Null	Email
4.	password	VARCHAR(30)	Not Null	Password

3. tbl_seller
Primary key: **seller_id**
Foreign key: **user_id** references **tbl_user**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	seller_id	INT	Primary Key, Foreign Key	Seller Id
2.	user_id	INT	Foreign Key referencing Users	User Id
3.	seller_name	VARCHAR(30)	Not Null	Seller Name
4.	company_name	VARCHAR(30)	Not Null	Company name
5.	contact_no	VARCHAR(20)	Not Null	Mobile Number

4. tbl_customer

Primary key: **customer_id**
Foreign key: **user_id** references **tbl_user**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	customer_id	INT	Primary Key, Foreign Key	Customer Id
2.	user_id	INT	Foreign Key referencing Users	User Id
3.	customer_name	VARCHAR(30)	Not Null	Customer name
4.	contact_no	VARCHAR(20)	Not Null	Mobile Number

5. tbl_address

Primary key: **address_id**
Foreign key: **user_id** references **tbl_user**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	address_id	INT	Primary Key, Auto Increment	Address Id
2.	user_id	INT	Foreign Key referencing Customer orSeller	User Id
3.	address_type	VARCHAR(20)	Not Null	Permanent or Residential
4.	address_line1	VARCHAR(30)	Not Null	Address line
5.	address_line2	VARCHAR(30)		Address line 2
6.	city	VARCHAR(50)	Not Null	City
7.	state	VARCHAR(50)	Not Null	State
8.	zip_code	VARCHAR(10)	Not Null	Zip code
9.	country	VARCHAR(50)	Not Null	Country
10.	default	INT	Not null	Set default address

6. tbl_categories
Primary key: **category_id**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	category_id	INT	Primary Key, Auto Increment	Category Id
2.	category_name	VARCHAR(30)	Not Null	Category name
3.	status	INT	Not Null	Active or deleted

7. tbl_address

Primary key: **subcategory_id**
Foreign key: **category_id** references **tbl_ category**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	subcategory_id	INT	Primary Key, Auto Increment	Subcategory Id
2.	subcategory_name	VARCHAR(30)	Not null	Subcategory name
3.	category_id	INT	Foreign Key referencing Categories	Category Id

8. tbl_product

Primary key: **product_id**
Foreign key: **seller_id** references **tbl_seller, subcategory_id** references **tbl_subcategory**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	product_id	INT	Primary Key, Auto Increment	Product Id
2.	seller_id	INT	Foreign Key referencing Seller	Seller Id
3.	subcategory_id	INT	Foreign Key referencing Subcategories	Sub Category name
4.	product_name	VARCHAR(30)	Not Null	Product name
5.	description	TEXT	Not Null	Description aboutproduct
6.	price	DECIMAL(10,2)	Not Null	Price of product
7.	stock_quantity	INT	Not Null	Total quantity
8.	is_available	BOOLEAN	Not Null	If product is available

9. tbl_order

Primary key: **order_id**
Foreign key: **customer_id** references **tbl_customer**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
10.	order_id	INT	Primary Key, Auto Increment	order Id
11.	customer_id	INT	Foreign Key referencing Customer	customer name
12.	order_date	DATETIME	Not Null	Order date
13.	total_amount	DECIMAL(10,2)	Not Null	amount
14.	status	VARCHAR(20)	Not Null	Pending, Shipped, Delivered

10. tbl_orderdetail

Primary key: **order_detail_id**
Foreign key: **order_id** references **tbl_order, product_id** references **tbl_product**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	order_detail_id	INT	Primary Key, Auto Increment	Order Detail Id
2.	order_id	INT	Foreign Key referencing Orders	Order Id
3.	product_id	INT	Foreign Key referencing Products	Product Id
4.	quantity	INT	Not Null	Quantity
5.	unit_price	Decimal(10,2)	Not null	Price per unit

11. tbl_review

Primary key: **review_id**
Foreign key: **customer_id** references **tbl_customer, product_id** references **tbl_product**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	review_id	INT	Primary Key, Auto Increment	Review Id
2.	product_id	INT	Foreign Key referencing Product	Product Id
3.	customer_id	INT	Foreign Key referencing Customer	Customer Id
4.	rating	DECIMAL(3,1)	Not Null	Rating by Customer
5.	review_text	TEXT		Text description

12. tbl_savelist

Primary key: **save_id**
Foreign key: **customer_id** references **tbl_customer, product_id** references **tbl_product**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	save_id	INT	Primary Key, Auto Increment	Save Id
2.	customer_id	INT	Foreign Key referencing Customer	Customer Id
3.	product_id	INT	Foreign Key referencing Product	Product Id

13. tbl_image

Primary key: **image_id**
Foreign key: **product_id** references **tbl_product**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	image_id	INT	Primary Key, Auto Increment	Image Id
2.	product_id	INT	Foreign Key referencing Product	Product Id
3.	image_url	VARCHAR(30)	Not Null	Url of Image

14. tbl_cart

Primary key: **cart_id**
Foreign key: **customer_id** references **tbl_customer, product_id** references **tbl_product**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	cart_id	INT	Primary Key, Auto Increment	Cart Id
2.	customer_id	INT	Foreign Key referencing Customer	Customer Id
3.	product_id	INT	Foreign Key referencing Product	Product Id
4.	quantity	INT	Not Null	Quantity

15. tbl_rating

Primary key: **rating_id**
Foreign key: **seller_id** references **tbl_seller**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	rating_id	INT	Primary Key, Auto Increment	Rating Id
2.	seller_id	INT	Foreign Key referencing Seller	Seller Id
3.	rating	DECIMAL(3,1)	Not Null	Rating given

16. tbl_payment

Primary key: **payment_id**
Foreign key: **order_id** references **tbl_seller**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	payment_id	INT	Primary Key, Auto Increment	Payment Id
2.	order_id	INT	Foreign Key referencing Order	Order Id
3.	payment_date	DATETIME	Not Null	Payment Date
4.	amount	DECIMAL(10,2)	Not Null	Total Amount
5.	payment_status	VARCHAR(20)	Not Null	Pending, Paid

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing is a crucial stage in the software development lifecycle and is sometimes referred to as system testing when done on an entire software system. Before being made available to end users, its main objective is to review the complete software program to make sure it complies with all necessary specifications, performs as intended, and is error- and defect-free. System testing focuses on checking the integrated system as a whole rather than just individual code components.

System testing is a thorough validation procedure that looks at many parts of the software system in the context of software testing. To ensure that the program carries out its intended activities precisely and effectively, functional requirements must be tested. To make sure the system satisfies expectations, non-functional elements like performance, security, and scalability are also closely examined. Testing various scenarios and boundary conditions is another aspect of system testing that helps to find weak points and potential vulnerabilities in the system. This in-depth investigation tries to find problems that might harm the user experience or jeopardize the integrity of the system.

Regression testing, which ensures that new updates or modifications don't cause new bugs or negatively affect current functionality, is an additional vital component of system testing. System testing ensures stakeholders that the software is dependable and stable while minimizing risks related to software failures via the use of rigorous testing processes. System testing's ultimate objectives are to raise the software's overall quality, increase user confidence, and produce a reliable system that meets users' needs and expectations.

5.2 TEST PLAN

A test plan is a detailed document that defines the full testing process for a software project and serves as a guide for actions related to quality control. It is essential for making sure that a piece of software has been extensively tested, adheres to specifications, and performs as intended. Several important elements are often included in the test plan.

The test plan, first and foremost, outlines the goals and parameters of the test. It describes the objectives of the testing team, the features and functionalities that will be tested, as well as any particular objectives or success criteria. This aids stakeholders in comprehending the aim of the testing and how it affects the general caliber of the project.

Second, the test plan details the testing technique and strategy. It outlines the several kinds of testing that will be done, such as functional, performance, security, and usability tests. It also describes the testing procedures, tools, and techniques that will be used. This section makes sure that the testing procedure fits the project's particular requirements and objectives.

A test plan also contains a thorough timetable and schedule for testing operations. It defines important checkpoints, due dates, and dependencies, assisting stakeholders and project managers in monitoring development and allocating resources wisely. A clearly established timetable guarantees that testing takes place at the appropriate phases of development and that any problems are dealt with quickly.

The different levels of testing include:

- Unit testing
- Integration testing
- Data validation testing
- Output testing

5.2.1 Unit Testing

A key component of software development is unit testing, which ensures the dependability and operation of certain parts or units inside a larger software system. Developers generate and run brief, isolated tests for certain sections of the codebase during unit testing, usually at the function or method level. The main goal is to make sure that each individual piece of code works as intended and generates the right outputs for the supplied inputs.

The majority of unit tests are automated, allowing for frequent execution during the development process and the early identification of errors and problems. Testing frameworks and libraries are frequently used by developers to speed up the preparation and execution of unit tests. Unit testing reduces the difficulty of debugging in the later phases of development by separating and testing individual units of code, enabling rapid identification and correction of issues.

The potential of unit testing to improve code maintainability and scalability is a big advantage. Unit tests serve as living documentation for programmers, giving them insight into how particular functions or methods should operate. As the codebase expands and new team members are added,

this documentation becomes increasingly important for preventing unintended side effects and functionality breaks from updates and modifications. Additionally, unit testing promotes sound coding techniques like modularization, separation of concerns, and adherence to design tenets like SOLID, which in turn results in software systems that are more durable and maintainable. The construction of high-quality, dependable, and maintainable software products is facilitated by unit testing, which is a core technique in software development.

5.2.2 Integration Testing

The evaluation of how various software application modules or components interact with one another when combined into a cohesive system is the primary goal of integration testing, an important stage in the software testing process. It acts as a link between unit testing, which isolates individual components, and system testing, which looks at the complete system as a whole. Integration testing tries to identify any difficulties, such as incompatible interfaces, data flow issues, or communication problems, that may occur as a result of the integration of various components.

Integrity testing's main goal is to make sure that the various components of the program perform as a cohesive whole to offer the desired functionality. Various scenarios in which components interact are simulated via test cases, and their results are assessed to see if they match the intended outcomes. This makes it easier to see any discrepancies or departures from the software's design guidelines.

There are several ways to approach integration testing, including top-down, bottom-up, and gradual approaches. Top-down integration testing gradually incorporates lower-level components after starting with the highest-level ones. Contrarily, bottom-up integration begins with the most basic components and progresses to increasingly intricate systems. Combining aspects of both, incremental testing involves gradually adding and testing components in small batches.

One of integration testing's key benefits is its capacity to identify problems early in the development process, decreasing the possibility of expensive mistakes emerging later on. By proactively resolving integration-related issues, it also improves the software's general dependability and resilience. Integration testing is crucial in ensuring that the finished product works as intended and lives up to user expectations by carefully examining how various system components interact. It helps to produce reliable and stable software applications and is a crucial component of software quality assurance.

5.2.3 Validation Testing or System Testing

A crucial step in the software testing process, validation testing, often referred to as system testing, focuses on assessing a whole system or software application to make sure it satisfies the requirements and operates well in the environment for which it is designed. This thorough testing strategy entails evaluating the system's general behavior, functionality, and performance to ensure that it meets the needs and expectations of the user. Validation testing ensures that the program achieves the required outcomes, runs without hiccups, and effectively manages a range of inputs, situations, and user interactions. It acts as the last checkpoint before the software is released, giving stakeholders the assurance that the system is prepared for use in production and can dependably carry out the activities it is designed to do while adhering to quality standards and compliance.

5.2.4 Output Testing or User Acceptance Testing

User acceptance testing (UAT) is a crucial stage in the software development lifecycle when the program or application is assessed from the viewpoint of the end-user to make sure it complies with the stated requirements and performs as expected. Representatives of the target audience test and validate the system's outputs, such as reports, screens, and results, during UAT to ensure that they meet their demands and corporate goals. During this testing phase, the main goal is to ensure that the program meets user expectations while delivering the intended results accurately. Successful UAT reduces the chance of errors or problems that might affect user satisfaction and company operations and helps ensuring that the software is ready for production deployment.

5.2.5 Automation Testing

Automation is the use of technology, software, or equipment to carry out activities, procedures, or tasks with the least amount of human involvement. It strives to improve efficiency, decrease mistakes, and increase production across many sectors by streamlining and optimizing procedures. Automated processes carry out routine, rule-based tasks, freeing up human workers to concentrate on more difficult, strategic, or creative duties. Manufacturing, software development, data analysis, customer service, and several other industries all use automation. It is essential for quickening corporate procedures, maintaining dependability, and attaining consistency, all of which result in reduced costs and better results.

5.2.6 Selenium Testing

Selenium is a well-liked open-source framework for online application automation that gives programmers the means to interface with web browsers. Its main component, Selenium WebDriver, enables the automation of browser functions including button clicks, form fills, and page navigation. Selenium is incredibly adaptable since it supports a broad spectrum of programming languages and enables developers to create automation scripts in the language of their choice. Selenium is frequently used for functional, regression, and cross-browser testing to make sure that web applications perform as intended across a variety of browsers. It is a solid option for thorough web application testing since it can be connected with Selenium Grid for parallel test execution and with other tools for load testing.

The Behavior-Driven Development (BDD) tool Cucumber, on the other hand, encourages collaborators from both technical and non-technical backgrounds to work together. This is achieved by allowing tests to be expressed in plain text using the Gherkin language, which is both legible by humans and simple to understand. Feature files, which each represent a distinct component of the functionality of the program, serve to structure test cases. To translate Gherkin stages into practical automation logic, Cucumber uses step definitions that are defined in code. Cucumber acts as a link among non-technical stakeholders who construct test scenarios and technical testers who employ the automation code when paired with Selenium. In addition to ensuring that web applications are adequately tested for functionality and quality, this combination also improves communication, test documentation, and maintainability.

Test Case 1: Customer - Login page

Code

```
package stepdefinition;
import org.openqa.selenium.By;
public class loginsteps {
    WebDriver driver=null;
    @Given("browser is open")
    public void browser_is_open() {
        System.setProperty("webdriver.gecko.marionette","D:\\Downloads\\Eclipse\\Assignment\\src\\test\\resources\\dr
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }
    @And("user is on login page")
    public void user_is_on_login_page() throws InterruptedException {
        driver.navigate().to("http://localhost/Project/login.php");
        Thread.sleep(1000);
    }
    @When("user enters username and password")
    public void user_enters_username_and_password() {
        driver.findElement(By.id("id_email")).sendKeys("thomaspunchodil@gmail.com");
        driver.findElement(By.id("id_password")).sendKeys("Ammu@123");
    }
    @And("User clicks on login")
    public void user_clicks_on_login() {
        driver.findElement(By.id("log")).click();
    }
    @Then("user is navigated to the home page")
    public void user_is_navigated_to_the_home_page() throws Exception {
        //driver.navigate().to("http://localhost/Project/home.php");
        //driver.findElement(By.id("logout")).isDisplayed();
        String expectedTitle = "SwapZone Home";
        String actualTitle = driver.getTitle();
        Assert.assertEquals(expectedTitle,actualTitle);
        Thread.sleep(500);
        driver.quit();
    }
}
```

Screenshot

```
  Then user is navigated to the home page1                # stepdefinition.loginsteps.user_is_navigated_to_the_home_page

1 Scenarios (1 passed)
5 Steps (5 passed)
0m12.239s

Share your Cucumber Report with your team at https://reports.cucumber.io
Activate publishing with one of the following:

src/test/resources/cucumber.properties:          cucumber.publish.enabled=true
src/test/resources/junit-platform.properties:    cucumber.publish.enabled=true
Environment variable:                            CUCUMBER_PUBLISH_ENABLED=true
JUnit:                                           @CucumberOptions(publish = true)

More information at https://cucumber.io/docs/cucumber/environment-variables/

Disable this message with one of the following:

src/test/resources/cucumber.properties:          cucumber.publish.quiet=true
src/test/resources/junit-platform.properties:    cucumber.publish.quiet=true
```

Test Report 1

Test Case 1

Project Name: SWAPZONE					
Login Test Case					
Test Case ID: Test_1			**Test Designed By:** Preethy Ann Thomas		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 09-10-2023		
Module Name: Login Screen			**Test Executed By :** Ms. Sruthimol Kurian		
Test Title : Customer Login			**Test Execution Date:** 09-10-2023		
Description: Verify login with valid mail and password					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid Email	Email: preethy@gmail.com	Customer should be able to login	Customer Logged in and navigated to Customer Dashboard with records	Pass
3	Provide Valid Password	Password: preethy@123			
4	Click on login button				
Post-Condition: Admin is validated with database and successfully login into account.					

Test Case 2: Admin – Edit Profile

Code

```
package stepdefinition;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.*;
import java.util.List;
public class edit {
    WebDriver driver;
    @Given("browser is open4")
    public void browser_is_open4() {
        System.setProperty("webdriver.gecko.marionette","D:\\Downloads\\Eclipse\\Assignment\\src\\test\\resources\\dr
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }
    @And("a user is logged in4")
    public void a_user_is_logged_in4() {
        driver.get("http://localhost/Project/login.php");
        driver.findElement(By.id("id_email")).sendKeys("admin@gmail.com");
        driver.findElement(By.id("id_password")).sendKeys("Admin@123");
        driver.findElement(By.id("log")).click();
    }
    @When("the user is on home page and edit button is clicked")
    public void the_user_is_on_home_page_and_edit_button_is_clicked() {
        driver.findElement(By.id("edit")).click();
    }
    @Then("user navigate to communication detail page")
    public void user_navigate_to_communication_detail_page() {}
    @Then("address is changed")
    public void address_is_changed() throws Exception{
        WebElement address = driver.findElement(By.id("address"));
        address.clear();
        address .sendKeys("Punchodil change");
        WebElement area = driver.findElement(By.id("area"));
        area.clear();
        area.sendKeys("Melpadom change");

       WebElement city = driver.findElement(By.id("city"));
       city.clear();
       city.sendKeys("Alappuzha change");
       Thread.sleep(500);
       driver.findElement(By.id("submit")).click();
    }

    @Then("is to be navigated to dashboard")
    public void is_to_be_navigated_to_dashboard() throws Exception {
        Thread.sleep(800);
        driver.quit();
    }
}
```

Screenshot

```
  And is to be navigated to dashboard                    # stepdefinition.edit.is_to_be_navigated_to_dashboard()

1 Scenarios (1 passed)
5 Steps (6 passed)
0m13.493s


Share your Cucumber Report with your team at https://reports.cucumber.io
Activate publishing with one of the following:

src/test/resources/cucumber.properties:          cucumber.publish.enabled=true
src/test/resources/junit-platform.properties:    cucumber.publish.enabled=true
Environment variable:                            CUCUMBER_PUBLISH_ENABLED=true
JUnit:                                           @CucumberOptions(publish = true)

More information at https://cucumber.io/docs/cucumber/environment-variables/

Disable this message with one of the following:

src/test/resources/cucumber.properties:          cucumber.publish.quiet=true
```

Test Report

Test Case 2

Project Name: SWAPZONE					
Admin Edit Profile Test Case					
Test Case ID: Test_2			**Test Designed By:** Preethy Ann Thomas		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 09-10-2023		
Module Name: Admin edit profile			**Test Executed By :** Ms. Sruthimol Kurian		
Test Title : Admin edit profile			**Test Execution Date:** 09-10-2023		
Description: Verify admin can edit profile					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Navigation to Login Page		Admin should be able to login	Admin Logged in and navigated to Admin Dashboard with records	Pass
3	Provide Valid Email	Email: admin@gmail.com			
4	Provide Valid Password	Password: admin@123			
5	Click on edit profile		Should be able to redirect to change profile page	Navigated to change profile page	Pass
6	Provide valid address fields	Address: Punchodil Change City: Melpadom Change Area: Alappuzha Change	Address field to be changed and redirect to dashboard	Profile is being edited and is redirected to dashboard with updated profile	Pass
7	Click on submit button				
Post-Condition: Admin profile is being edited and saved to database successfully.					

Test Case 3: Customer – Add to Wishlist

Code

```
package stepdefinition;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.*;
public class wishlist {
    WebDriver driver;
    @Given("browser is open2")
    public void browser_is_open2() {
        System.setProperty("webdriver.gecko.marionette","D:\\Downloads\\Eclipse\\Assignment\\src\\test\\resources\\drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }
    @And("a user is logged in")
    public void a_user_is_logged_in() {
        driver.get("http://localhost/Project/login.php");
        driver.findElement(By.id("id_email")).sendKeys("thomaspunchodil@gmail.com");
        driver.findElement(By.id("id_password")).sendKeys("Ammu@123");
        driver.findElement(By.id("log")).click();
    }
    @When("the user is on the product page")
    public void the_user_is_on_the_product_page() {
    }
    @Then("\"Add to Wishlist\" button is clicked for product with name {string}")
    public void Add_to_Wishlist_button_is_clicked_for_product_with_name(String productName) {
        // Locate the "Add to Wishlist" button for the specified product        WebElement addToWishlistButton = driver.findElement(By.xpath("//h5[
        WebElement addToWishlistButton = driver.findElement(By.xpath("//h5[contains(text(),\"A.T.U.N Girl's Dress\")]/following-sibling::div//buttc
        // Click the "Add to Wishlist" button
        addToWishlistButton.click();
        // Switch to the alert
        Alert alert = driver.switchTo().alert();
        // Click "OK" on the alert
        alert.accept();
    }
    @Then("is to be navigated to wishlist page")
    public void is_to_be_navigated_to_wishlist_page()throws Exception {

         // Assuming you want to navigate to the wishlist page
         driver.navigate().to("http://localhost/Project/wishList.php");
         Thread.sleep(500);
         driver.quit();
     }
}
```

Screenshot

```
Dynamically enable window occlusion 1
JavaScript error: http://localhost/Project/js/jquery-2.1.0.min.js, line 2: Error: Syntax error, unrecognized expression: productList.php?category_id=4
  And is to be navigated to wishlist page                                                    # stepdefinition.wishlist.is_to_be_navigated_to_wishlist_page

1 Scenarios (1 passed)
5 Steps (5 passed)
0m9.651s


Share your Cucumber Report with your team at https://reports.cucumber.io
Activate publishing with one of the following:

src/test/resources/cucumber.properties:          cucumber.publish.enabled=true
src/test/resources/junit-platform.properties:    cucumber.publish.enabled=true
Environment variable:                            CUCUMBER_PUBLISH_ENABLED=true
JUnit:                                           @CucumberOptions(publish = true)

More information at https://cucumber.io/docs/cucumber/environment-variables/

Disable this message with one of the following:

src/test/resources/cucumber.properties:          cucumber.publish.quiet=true
src/test/resources/junit-platform.properties:    cucumber.publish.quiet=true
```

Test Report

Test Case 3

Project Name: SWAPZONE					
Customer Add to Wishlist Test Case					
Test Case ID: Test_3			**Test Designed By:** Preethy Ann Thomas		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 09-10-2023		
Module Name: Customer Wishlist			**Test Executed By :** Ms. Sruthimol Kurian		
Test Title : Customer add to wishlist			**Test Execution Date:** 09-10-2023		
Description: Verify customer can add product to wishlist					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Navigation to Login Page		Customer should be able to login	Customer Logged in and navigated to home page containing various products	Pass
3	Provide Valid Email	Email: preethy@gmail.com			
4	Provide Valid Password	Password: preethy@123			
5	Click on Add to Wishlist button		Should be able to add product to wishlist	Product added to wishlist and alert box appears that is being handled	Pass
6	Change title to Wishlist		Navigate to wishlist page	Navigated to wishlist page containing newly added product	Pass
Post-Condition: Customer is able to add product to wishlist and is updated to database successfully.					

Test Case 4: Customer – Search for Products based on Colour

Code

```
package stepdefinition;

import org.openqa.selenium.By;

public class search {
    WebDriver driver;
    @Given("browser is open3")
    public void browser_is_open3() {
        System.setProperty("webdriver.gecko.marionette","D:\\Downloads\\Eclipse\\Assignment\\src\\test\\resources\\drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }
    @And("a user is logged in3")
    public void a_user_is_logged_in3() {
        driver.get("http://localhost/Project/login.php");
        driver.findElement(By.id("id_email")).sendKeys("thomaspunchodil@gmail.com");
        driver.findElement(By.id("id_password")).sendKeys("Ammu@123");
        driver.findElement(By.id("log")).click();
    }
    @When("the user enters {string} into the search bar")
    public void the_user_enters_keyword_into_the_search_bar(String keyword) {
        driver.findElement(By.id("search")).sendKeys("Pink");
    }

    @And("clicks the \"Search\" button")
    public void clicks_the_Search_button() {
        WebElement searchButton = driver.findElement(By.id("searchbutton"));

        // Use JavaScriptExecutor to click the "Search" button
        ((JavascriptExecutor) driver).executeScript("arguments[0].click();", searchButton);
    }

    @Then("the user should see a list of products related to {string}")
    public void the_user_should_see_related_products(String keyword) throws Exception {
        Thread.sleep(800);
        driver.quit();
    }
}
```

Screenshot

```
Dynamically enable window occlusion 1
  Then the user should see a list of products related to "Pink" # stepdefinition.search.the_user_should_see_related_products(java.lang.String)

1 Scenarios (1 passed)
5 Steps (5 passed)
0m9.283s


Share your Cucumber Report with your team at https://reports.cucumber.io
Activate publishing with one of the following:

src/test/resources/cucumber.properties:          cucumber.publish.enabled=true
src/test/resources/junit-platform.properties:    cucumber.publish.enabled=true
Environment variable:                            CUCUMBER_PUBLISH_ENABLED=true
JUnit:                                           @CucumberOptions(publish = true)

More information at https://cucumber.io/docs/cucumber/environment-variables/

Disable this message with one of the following:

src/test/resources/cucumber.properties:          cucumber.publish.quiet=true
src/test/resources/junit-platform.properties:    cucumber.publish.quiet=true
```

Test Report

Test Case 4

Project Name: SWAPZONE					
Customer Search for Products Test Case					
Test Case ID: Test_3			**Test Designed By:** Preethy Ann Thomas		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 09-10-2023		
Module Name: Customer Product Search			**Test Executed By :** Ms. Sruthimol Kurian		
Test Title : Customer add to wishlist			**Test Execution Date:** 09-10-2023		
Description: Verify customer can search for a product					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Navigation to Login Page		Customer should be able to login	Customer Logged in and navigated to home page containing various products	Pass
3	Provide Valid Email	Email: preethy@gmail.com			
4	Provide Valid Password	Password: preethy@123			
5	Click on search box and enter valid string	String: Pink	Should be able to see all pink clothes	Pink products are being displayed on screen	Pass
Post-Condition: Customer is able to see all clothes based on the string provided successfully.					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The SwapZone system's deployment is a crucial step toward making the idea of a vibrant online cloth marketplace a reality. This implementation process is prepared to bring to life the features and functions described in the project specifications thanks to a precise strategy and a committed team of developers, designers, and testers. Creating a fully working, user-friendly website that smoothly combines the purchasing and selling of clothing is the goal of this phase.

The process of implementation will include a number of crucial elements, such as but not restricted to database design, user interface development, security integration, payment gateways, and administrative controls. In order to guarantee the best performance, scalability, and reliability, each component will be meticulously designed. Throughout the implementation phase, the system will be improved and refined using rigorous testing, quality assurance, and user input. The goal of the project is to offer a cutting-edge and reliable platform for cloth aficionados while fostering business prospects for vendors. SwapZone seeks to establish new benchmarks in the online textile marketplace sector as it takes shape during its deployment.

6.2 IMPLEMENTATION PROCEDURES

The process of putting the program in its actual context and confirming that it fulfills the intended usage and performs as expected is known as software implementation. In certain companies, the project to build the software could be ordered by someone who won't be utilizing it themself. Initial skepticism of the program is possible, but it's crucial to prevent opposition from mounting. You may accomplish this by:

- Ensuring active users are informed of the new system's advantages, increasing their trust in the program.
- Ensuring that users receive the right instructions so they feel confident using the program.

Users need to understand that the server software has to be operating on the server before they can access the system. The planned procedure won't occur if the server object isn't active and functioning.

6.2.1 User Training

The purpose of user training is to get the user ready to test and modify the system. The individuals who will be engaging must have faith in their ability to contribute to the goal and advantages anticipated from the computer-based system. Training is more necessary as systems get more complicated. The user learns how to input data, handle error warnings, query the database, call up routines that will generate reports, and execute other important tasks via user training.

6.2.2 Training on the Application Software

The user must undergo training on the new application software after receiving the requisite foundational instruction on computer awareness. This instruction should cover the basic principles of using the newly implemented system, as well as the way the screens work together, how they are designed, what kind of help is available, the kinds of errors that might occur when entering data, the checks that should be made for validating each entry, and how to correct the data after it has been entered. Additionally, for effective use of the system or a component of the system, the training should contain knowledge that is particular to the user or group. It is crucial to remember that this training may vary depending on the user groups and hierarchical levels.

6.2.3 System Maintenance

As the period when the program is really used and carries out its planned duties, the maintenance phase is a vital component of the software development cycle. To maintain the system's functionality, dependability, and adaptability to changes in the system environment, proper maintenance is crucial. Maintenance activities involve more than just finding and repairing flaws or problems in the system. Updates to the program, adjustments to its features, and speed improvements are just a few instances of what it may entail. Software maintenance is essentially an ongoing process that calls for constant surveillance, assessment, and enhancement of the system to satisfy shifting user demands and requirements.

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

A dynamic and cutting-edge online platform designed to satisfy the requirements of consumers as well as sellers in the textile market, SwapZone stands for. SwapZone wants to transform the way textile transactions are carried out in the digital era with its intuitive layout, extensive features, and variety of capabilities. SwapZone provides a flexible marketplace that accommodates a range of client preferences by letting vendors to display both stitched and unstitched clothing. The system analysis for the project has carefully evaluated the landscape of the current textile market, highlighting its shortcomings. SwapZone has been established to address these issues and offer an effective, secure, and entertaining forum for cloth aficionados through a natural and planned system study.

Numerous benefits, including a simplified user experience, enhanced security, and an easy-to-use interface, are introduced by the suggested approach. SwapZone aims to establish a dependable and trustworthy cloth marketplace by emphasizing enhancing user interactions, streamlining database administration, and laying in place strong security measures. SwapZone will follow a disciplined development process that includes database design, user interface development, feature development, security integration, and other tasks as it moves closer to realization. The platform will be held to the highest standards through meticulous testing, the inclusion of user input, and ongoing oversight. In a nutshell SwapZone imagines a time where all people may conduct textile transactions in a way that is simple, safe, and pleasurable. SwapZone is excited to transform the experience of buying and selling textiles with its dedication to offering a top-tier online marketplace.

7.2 FUTURE SCOPE

The future scope of SwapZone holds significant potential for growth and expansion in various dimensions which include:

1. Diversified Product Categories: In addition to apparel, accessories, footwear, and fashion-related goods can be added to SwapZone's product categories. A greater variety of consumers and sellers would be drawn to the expansion, increasing the platform's adaptability.

2. Mobile App: Creating a mobile app specifically for SwapZone can improve user ease and

accessibility. Mobile applications offer a more individualized and convenient purchasing experience, which is becoming more and more well-liked by consumers.

3. Advanced Analytics: Using machine learning algorithms and advanced data analytics, it is possible to forecast market patterns, enhance user engagement, and provide users with individualized suggestions.

4. Social media integration: SwapZone may increase user engagement and promote items through social networking by integrating with well-known social media networks. Integration with social network may make login and registration simple.

5. Green Initiatives: In keeping with environmental awareness, SwapZone may encourage vendors and buyers to use sustainable methods. Promoting eco-friendly packaging, assisting retailers of repurposed clothing, and supporting carbon offset schemes are a few examples of this.

6. Integration of user feedback: Gathering and acting on user feedback continuously is crucial for improvement. To better understand customer requirements and preferences, the platform can include user-friendly feedback instruments and do surveys.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- PankajJalote, *"Software engineering: a precise approach"*
- Roger S Pressman, *"Software Engineering"*
- IEEE Std 1016, *"Recommended Practice for Software Design Descriptions"*

WEBSITES:

- www.w3schools.com
- www.bootstrap.com
- www.amazon.in
- www.jquery.com

CHAPTER 9

APPENDIX

9.1 Sample Code

Login

```
<?php
require_once('test_login/auth.php');
require_once('vendor/autoload.php');
include("connection.php");

if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

$clientID = "95908132454-43ct561tga2rk82a6bku1e1llekgfemv.apps.googleusercontent.com"; // Replace with your Google Client ID
$secret = "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"; // Replace with your Google Client Secret

// Google API Client
$gclient = new Google_Client();

$gclient->setClientId($clientID);
$gclient->setClientSecret($secret);
$gclient->setRedirectUri('http://localhost/Project/login.php');

$gclient->addScope('email');
$gclient->addScope('profile');



// Check if the user clicks the Google sign-in button
if (isset($_GET['code'])) {
    // Get Token
    $token = $gclient->fetchAccessTokenWithAuthCode($_GET['code']);

    // Check if fetching token did not return any errors
    if (!isset($token['error'])) {
        // Setting Access token
        $gclient->setAccessToken($token['access_token']);

        // Store access token
        $_SESSION['access_token'] = $token['access_token'];

        // Get Account Profile using Google Service
        $gservice = new Google_Service_Oauth2($gclient);

        // Get User Data
        $udata = $gservice->userinfo->get();
```

```
        foreach ($udata as $k => $v) {
            $_SESSION['login_' . $k] = $v;
        }
        $_SESSION['ucode'] = $_GET['code'];
        $email = $_SESSION['login_email'];

        if (isset($_SESSION['ucode']) && !empty($_SESSION['ucode'])) {
            // Query the database to check if the email exists
            $query = "SELECT * FROM tbl_user WHERE email = '" .
mysqli_real_escape_string($conn, $email) . "'";
            $result = mysqli_query($conn, $query);

            if ($result && mysqli_num_rows($result) > 0) {
                $user_data = mysqli_fetch_assoc($result);
                $role = $user_data['role'];

                switch ($role) {
                    case 0:
                        // Admin role
                        $_SESSION['login_role'] = 0;
                        $_SESSION['user_id'] = $user_data['user_id'];
                        header("Location: admin/adminProfile.php");
                        exit;
                    case 1:
                        $_SESSION['login_role'] = 1;
                        $_SESSION['user_id'] = $user_data['user_id'];
                        // User role
                        header("Location: home.php");
                        exit;
                    case 2:
                        $_SESSION['login_role'] = 2;
                        $_SESSION['user_id'] = $user_data['user_id'];
                        // Seller role
                        header("Location: seller/sellerProfile.php");
                        exit;
                    default:
                        // Invalid role
                        echo '<script>alert("Invalid role!")</script>';
                        exit;
                }
            } else {
                // Email not found in the database
                echo '<script>alert("Email not found!")</script>';
            }
        } else {
            echo '<script>alert("Invalid session code!")</script>';
        }
    } else {
        echo '<script>alert("Token retrieval error!")</script>';
```

```
    }
}
?>
<?php
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}
include("connection.php");

if (isset($_POST['login'])) {
    $name = $_POST["email"];
    $pass = $_POST["password"];
    $pass = md5($pass);

    /* if (!empty($name) && !empty($pass) && !is_numeric($name)) {
        $recaptchaSecretKey = "6Legj5MnAAAAAEURNcMOtZQAacUKNiRwRtoc1gjj";
        $recaptchaResponse = $_POST['g-recaptcha-response'];

        $verifyResponse =
file_get_contents("https://www.google.com/recaptcha/api/siteverify?secret=$recaptchaSec
retKey&response=$recaptchaResponse");
        $responseData = json_decode($verifyResponse);

        if ($responseData->success) { */
            $query = "SELECT * FROM tbl_user WHERE email = '$name' AND password =
'$pass'";
            $result = mysqli_query($conn, $query);

            if ($result && mysqli_num_rows($result) > 0) {
                $user_data = mysqli_fetch_assoc($result);

                if ($user_data['password'] === $pass) {
                    $_SESSION['user_id'] = $user_data['user_id'];
                    $role = $user_data['role'];

                    switch ($role) {
                        case 0:
                            header("Location: admin/adminProfile.php");
                            break;
                        case 1:
                            header("Location: home.php");
                            break;
                        case 2:
                            header("Location: seller/sellerProfile.php");
                            break;
                        default:
                            echo '<script>alert("Invalid role!")</script>';
                            break;
                    }
```

```
            }
        } /* else {
            echo '<script>alert("Wrong username or password!")</script>';
        }
    } else {
        echo '<script>alert("reCAPTCHA verification failed!")</script>';
    }
} */else {
    echo '<script>alert("Wrong username or password!")</script>';
}
}
?>
<!DOCTYPE html>
<html>
<head>
    <script src="https://www.google.com/recaptcha/api.js" async defer></script>
    <script src="https://code.jquery.com/jquery-3.6.1.min.js" integrity="sha256-
o88AwQnZB+VDvE9tvIXrMQaPlFFSUTR+nldQm1LuPXQ="
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
    <style>
    body {
        background-color: #f8f9fa;
        background-image: url("images/bg2.jpg");
        background-repeat: no-repeat;
        background-size: cover;
        height: 100vh;
    }
    .google-icon {
            width: 18px;
            height: 18px;
        }

        /* Sign in with Google button styling */
        .sign-in-button {
            background-color: #E74C3C; /* Red background color */
            color: #ffffff; /* White text color */
            border: none;
            border-radius: 50px; /* Make the button round */
            padding: 10px 20px; /* Adjust padding as needed */
            cursor: pointer;
            display: flex;
            align-items: center; /* Vertically center icon and text */
            width:328px;
        }
```

```
    /* White circle surrounding the icon */
    .icon-circle {
        background-color: #ffffff; /* White circle background color */
        border-radius: 60%; /* Make it a circle */
        padding: 5px; /* Adjust padding as needed */
        margin-right: 10px; /* Adjust the spacing between icon and text */
        margin-left:60px;
    }

    /* Hover effect */
    .sign-in-button:hover {
        background-color: #C0392B; /* Slightly darker red on hover */
    }
    .hr-with-text {
        position: relative;
        border: none;
        height: 1px; /* Adjust the height of the line as needed */
        background-color: #ccc; /* Adjust the color of the line as needed */
    }

    /* Style for the "or" text */
    .or-text {
        position: absolute;
        top: -10px; /* Adjust the vertical position of the text */
        left: 50%;
        background-color: #fff; /* Background color to hide part of the line */
        padding: 0 10px; /* Adjust padding as needed */
        transform: translateX(-50%); /* Center the text horizontally */
    }
.container {
    margin-top: 130px;
    border-radius: 20px;
}

.login-container {
    max-width: 326px;
    margin: 0 auto;
    padding: 40px;
    background-color: rgb(251, 243, 243);
    border-radius: 20px;
    opacity: 0.9;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.login-container h2 {
    text-align: center;
    margin-bottom: 30px;
}
```

```
.form-group {
    margin-bottom: 20px;
}

.form-group label {
    font-weight: bold;
}

.form-group input {
    width: 100%;
    padding: 10px;
    border: 1px solid pink;
    border-radius: 10px;
}

.btn-login {
    display: block;
    width: 100%;
    padding: 10px;
    color: #ffffff;
    background: #392626;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    transition: 0.4s linear;
    border: none;
    border-radius: 10px;
    border-color: #131312;
    text-align: center;
    text-decoration: none;
}

.btn-login:hover {
    color: #ffffff;
    background: #080808;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    transition: 0.4s linear;
}
.bold-on-hover {
    font-weight: normal;
    transition: font-weight 0.2s ease;
}
.bold-on-hover:hover,
.bold-on-hover:active,
.bold-on-hover:visited {
    font-weight: bold;
    text-decoration: none;
}
.error {
    color: red;
    font-size: 14px;
```

```
      margin-top: 5px;
   }
</style>
</head>

<body>
   <div class="container">
      <div class="row">
         <div class="col-sm-9 col-md-7 col-lg-5 mx-auto">
            <div class="login-container">
               <h2><img src="images/logo.png" style="height: 60px;"></h2>
               <div class="form-group">
                  <form id="loginForm" method="post" name = "login">
                     <input type="email" name="email" id="id_email" placeholder="Email">
                     <p id="emailError" class="error"></p>
                     <input type="password" name="password" id="id_password"
placeholder="Password">
                     <p id="passwordError" class="error"></p>
                    <!-- <div class="g-recaptcha" data-
sitekey="6Legj5MnAAAAAMy8KZAXmeroGPQFM7hdMFE_E9zL"
                          style="align-items: center;width:330px;"></div><br> -->
                     <button type="submit" name = "login" class="btn btn-login" id="log"
style="color:#ede5e5;">Login</button>
                     <hr class="hr-with-text">
   <span class="or-text">or</span>
   <button class="sign-in-button">
      <div class="icon-circle">
      <img src="https://www.gstatic.com/firebasejs/ui/2.0.0/images/auth/google.svg" alt="Google
icon" class="google-icon">
   </a>
</div>
      <a href="<?= $gclient->createAuthUrl() ?>" class="btn btn btn-primary btn-flat rounded-0"
style = "text-decoration:none; color:white;">Login with Google</a>
   </button>
                     <br>
                     <table>
                        <tr>
                           <td>
                              <b><a href="recover_psw.php" style="text-decoration: none;
color:black;">Forgot Password?</a></b>
                           </td>
                           <td>
                              <b><a href="registration.php" style="text-decoration: none; color:black;
text-align:right;">       New User?</a></b>
                           </td>
                        </tr>
                     </table>
                  </form>
               </div>
```

```
            </div>
          </div>
       </div>
   </div>
</body>
</html>
```

Customer Profile

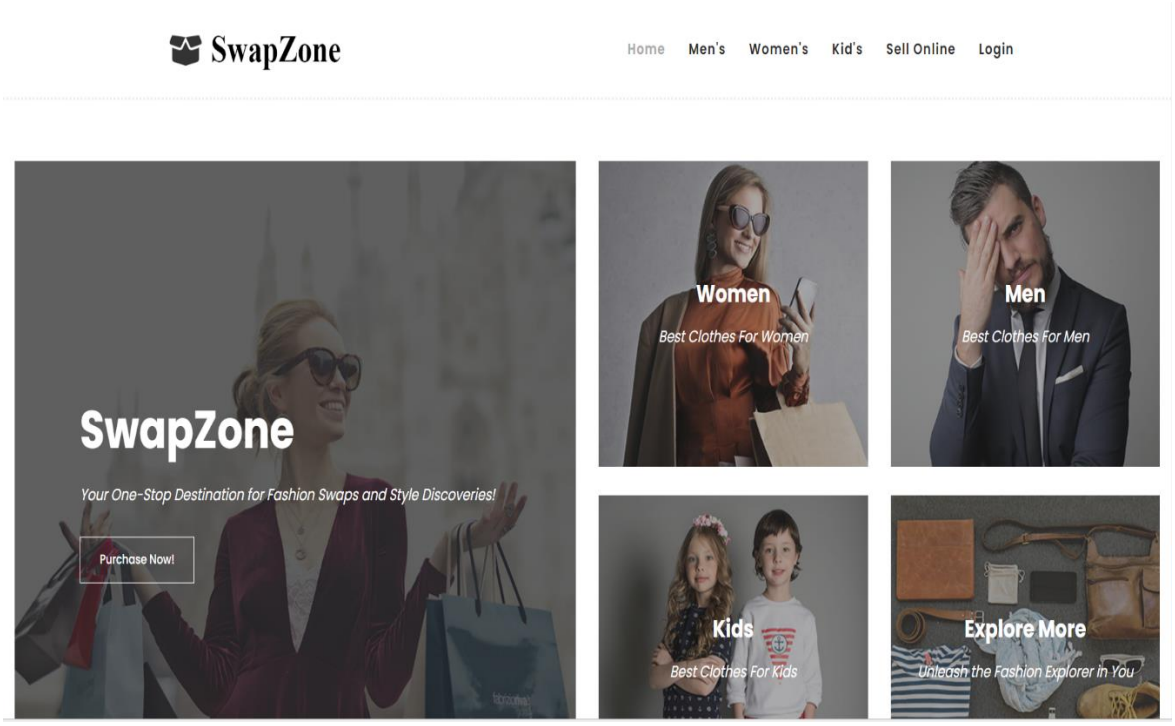
```
<?php
session_start(); // Start the session at the very beginning
if (!isset($_SESSION['user_id'])) {
   header('location:./');
   exit(); // Ensure that you exit after a header redirect
}
require 'connection.php'; // Require the connection script here to ensure it's loaded
?>
<!DOCTYPE html>
<html lang="en">
<html>
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Profile</title>
   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
   <style>
      .align-right {
      text-align: left;
   }
   </style>
</head>
<body>
   <?php include('customer_header.php') ?>
   <section><br><br>
   <div class="container mt-5">
      <div class="row">
         <br>
         </div>
        <br>
         <?php
         require 'connection.php';
         $user_id = $_SESSION['user_id'];
         $query2 = "SELECT * FROM tbl_user WHERE user_id = '$user_id'";
         $result2 = mysqli_query($conn, $query2);
         $row2 = mysqli_fetch_assoc($result2);
         $mail = $row2['email'];
```

```
        $user_id = $_SESSION['user_id'];
        $query1 = "SELECT * FROM tbl_customer WHERE user_id = '$user_id'";
        $result1 = mysqli_query($conn, $query1);
        while ($row1 = mysqli_fetch_assoc($result1) ) {

            echo '<div class="col-lg-6 mb-4">';
            echo '</div>';
            echo '<div class="card">';
            echo '<div class="card-body">';
            echo '<div >';
            echo '<p class="card-text"><h3>Personal Details <a href =
"updateProfile.php">        
            &e
msp;       <i class="fas fa-edit" style="color:
#89a9e1;height:25px;"></i></a></h3></p>';
            echo '<p class="card-text"><br><table><tr><td>Name: </td><td  class="align-right">' .
$row1['customer_name'] . '</td></p>';
            echo '<p class="card-text"><tr><td>Mail id: </td><td class="align-right">' .  $mail .
'</td></p>';
            echo '<p class="card-text"><tr><td>Contact number:      </td><td
class="align-right">' . $row1['contact_no'] . '</td></p>';
            echo '<p class="card-text"><tr><td>Password: </td><td class="align-
right">**********</td></table></p></div>';
           echo '<a href = "changePassword.php">Change Password</a>';
            echo '</div>';
        }
        mysqli_close($conn); // Close the database connection
        ?>
      </div>
   </div>
   </section>
   <section>

   <div class="container mt-5">

      <div class="row">
         <div class="col-lg-4 mb-4">
                <a href="addAddress.php"><img src = "images/addAddress.png" style =
"height:280px;width:320px;"></a>
         </div>
         <?php
         require 'connection.php';
         $user_id = $_SESSION['user_id'];
         $query = "SELECT * FROM tbl_address WHERE user_id = '$user_id'";
         $result = mysqli_query($conn, $query);
         while ($row = mysqli_fetch_assoc($result)) {

            echo '<div class="col-lg-4 mb-4">';
            echo '<div class="card">';
```

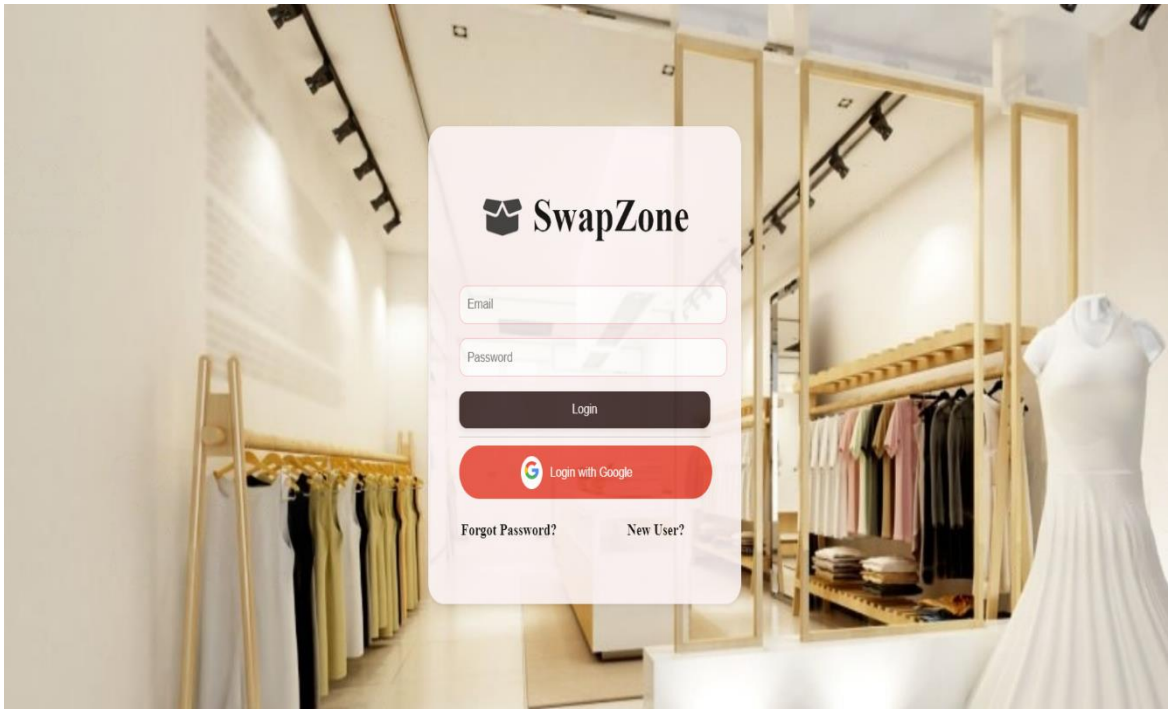
```
            echo '<div class="card-body">';
            echo '<h5 class="card-title">' . $row['name'] . '</h5>';
            echo '<p class="card-text">' . $row['mobile_number'] . '</br> '. $row['address'] . '</br> '
. $row['city'] . '</br> ' . $row['area'] . '</br> ' . $row['state'] . ' </br>' . $row['pincode'] . '</p><br>';
            echo '<a href="edit_address.php?address_id=' . $row['address_id'] . '" class="text-
primary"><small>Edit</small></a>';
            echo ' | ';
            echo '<a href="remove_address.php?address_id=' . $row['address_id'] . '" class="text-
primary"><small>Remove</small></a>';

            if ($row['is_default'] == 1) {
                echo ' |<small> Default </small>';
            } else {
                echo ' | <a href="set_default_address.php?address_id=' . $row['address_id'] . '"
class="text-primary"><small>Set as Default</small></a>';
            }
            echo '</div>';
            echo '</div>';
            echo '</div>';
        }

        mysqli_close($conn);
        ?>
      </div>
   </div>
   </section>
   <?php include('customer_footer.php') ?>
   <script src="js/myscripts.js"> </script>
   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
   <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js"></script>
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

9.1 Screen Shots

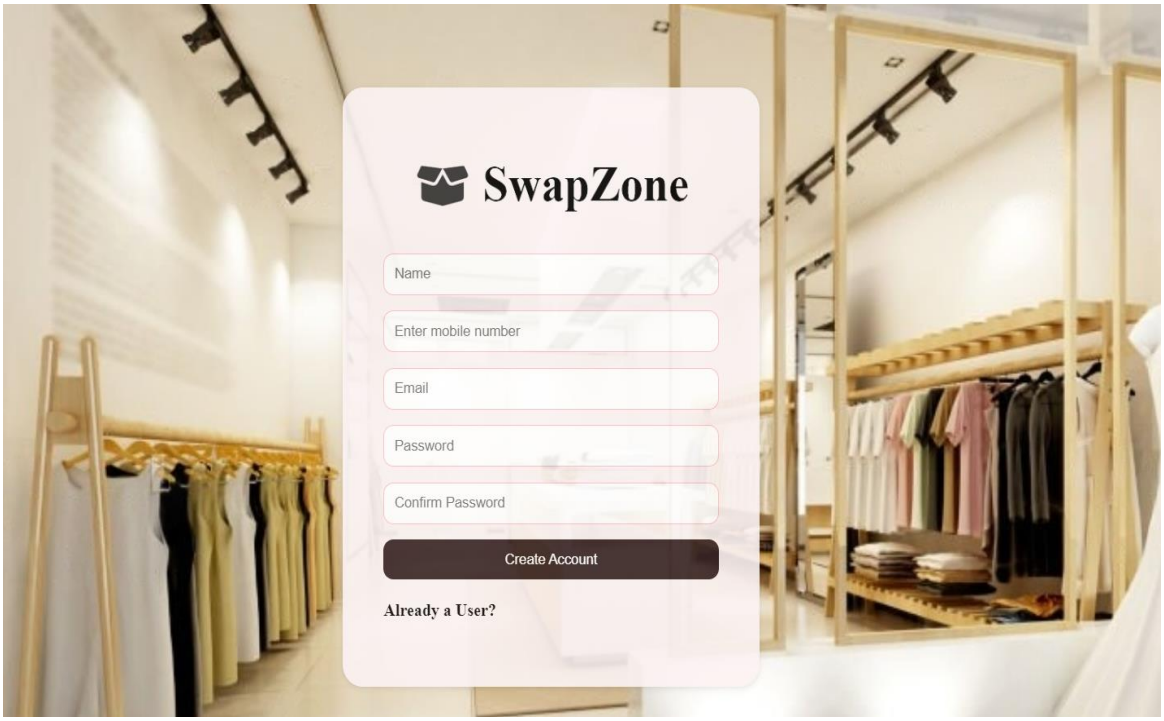
I. Index page



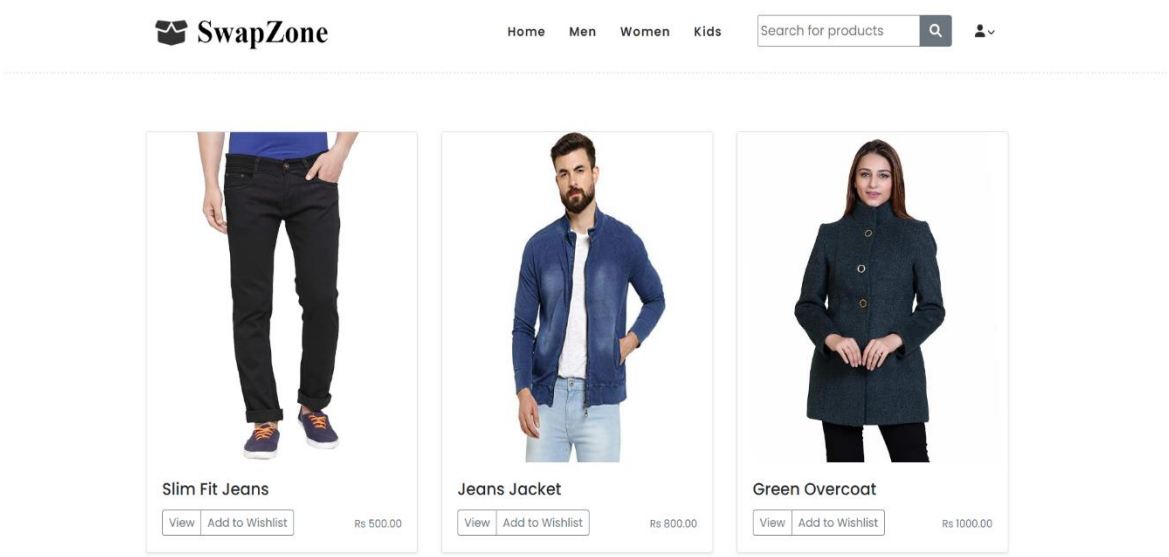
II. Login page



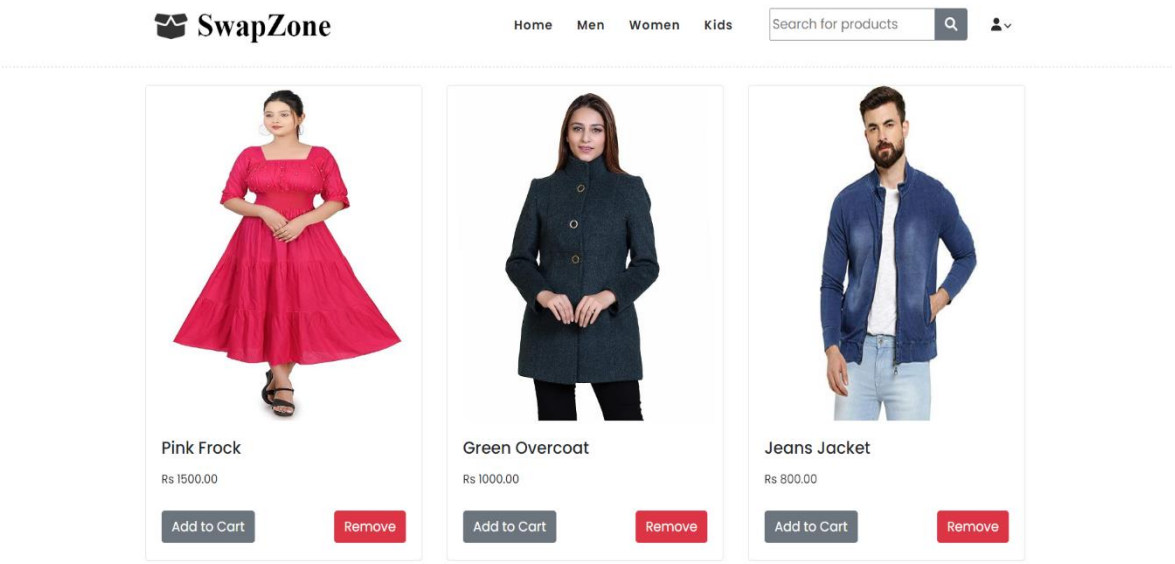
III. Customer Registration page



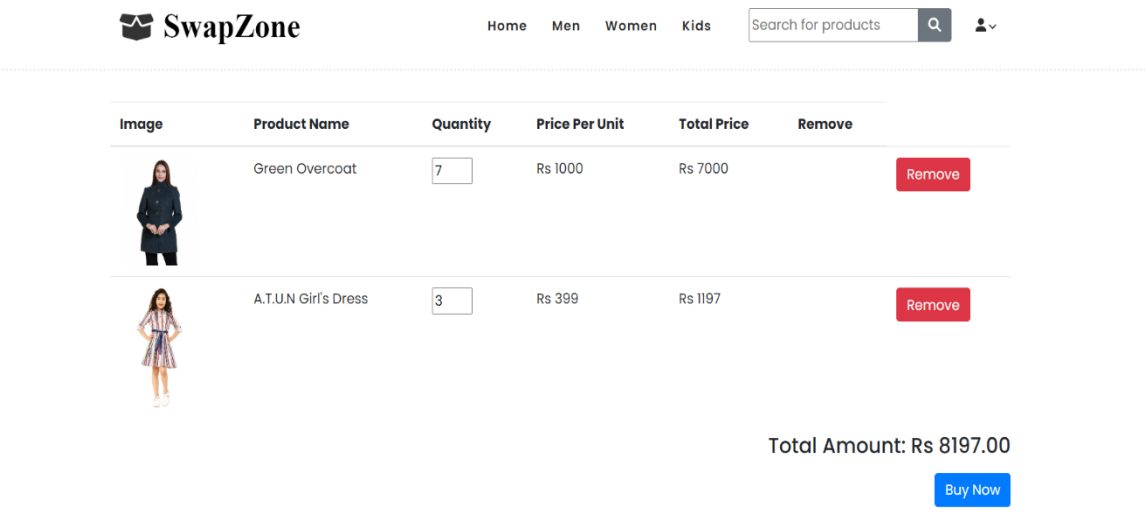
IV. Customer home page



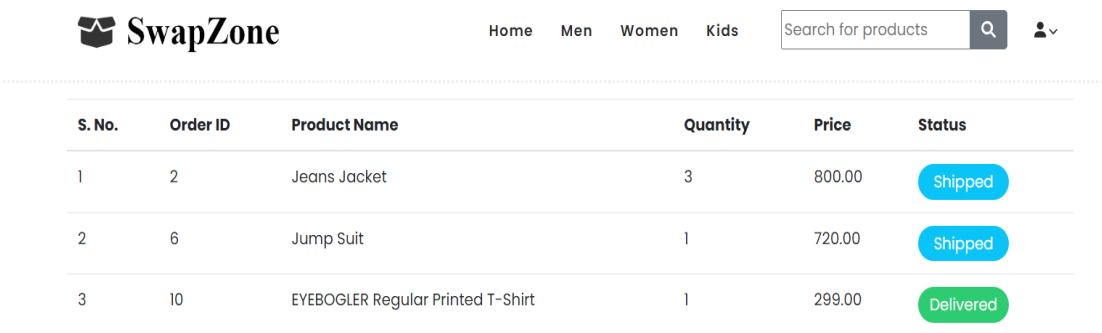
I. Wishlist page



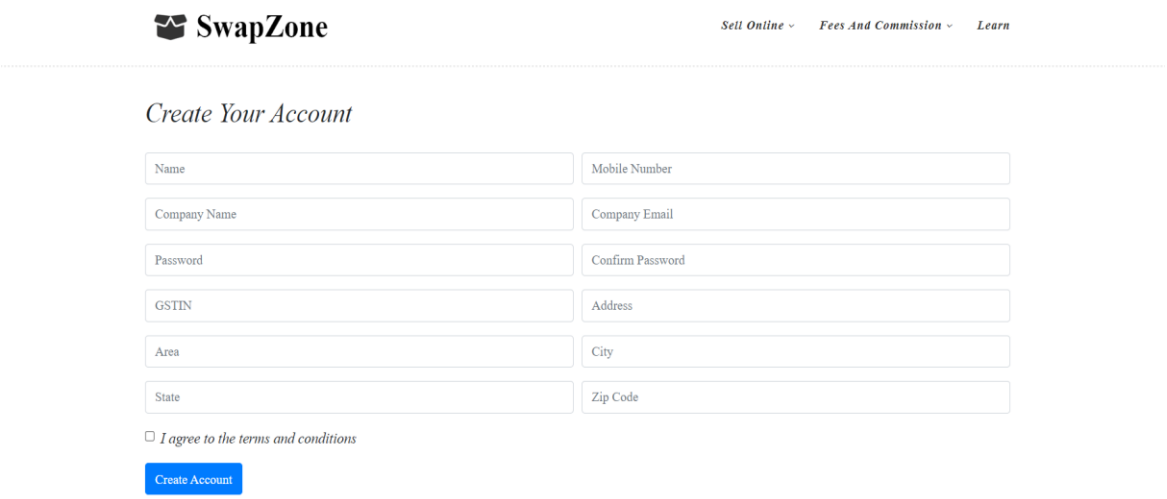
II. Cart page



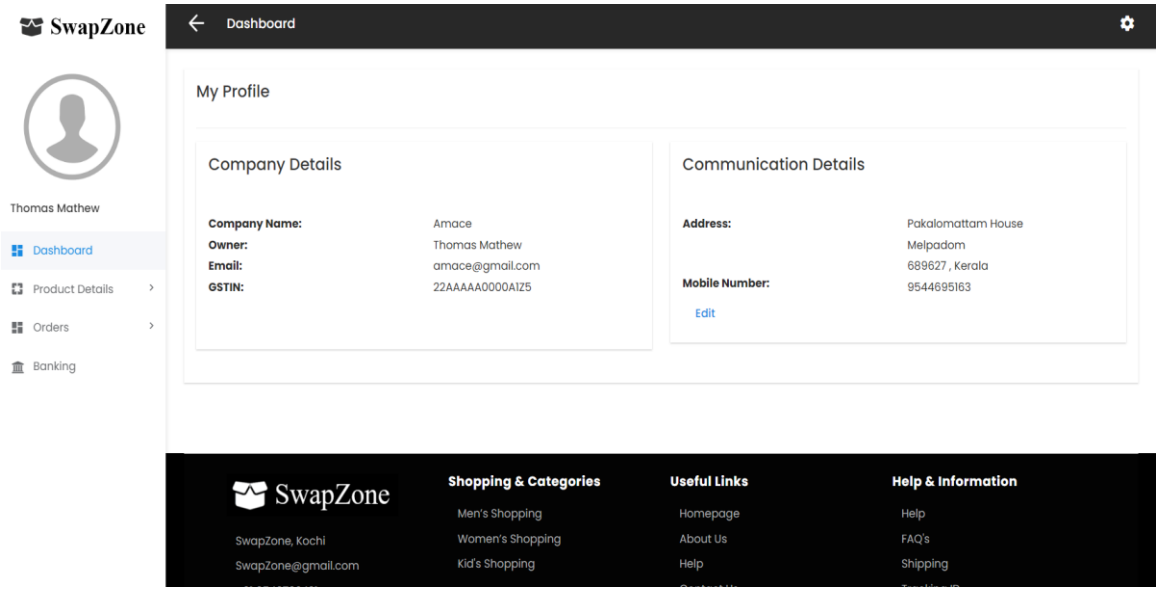
III. Order page



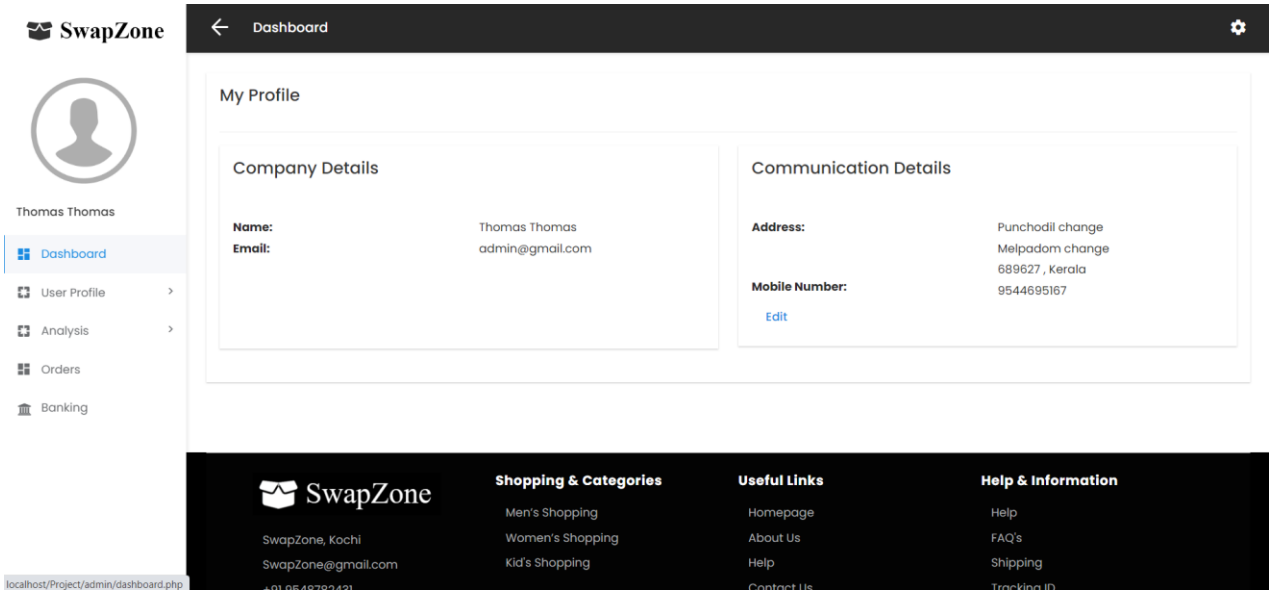
IV. Seller Registration



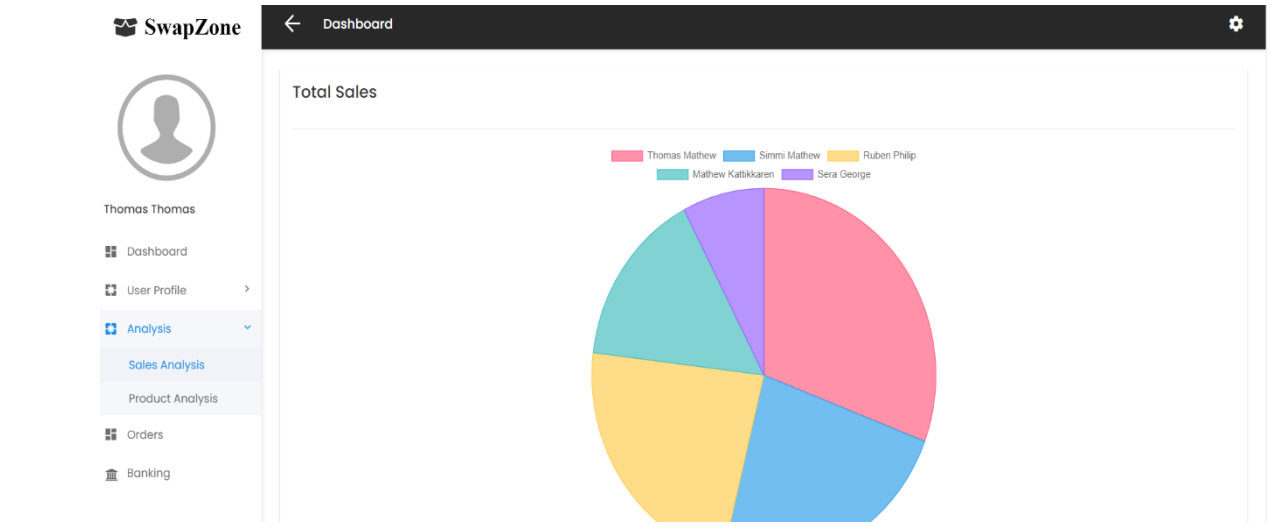
V. Seller Dashboard



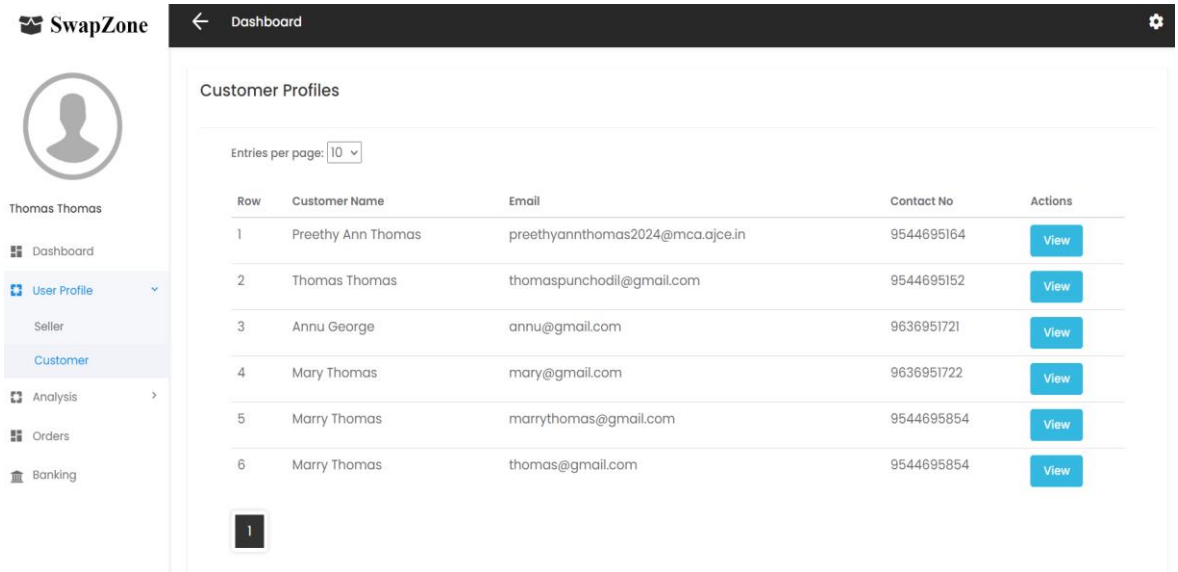
I. Admin Dashboard



II. Admin Sales Analysis



III. Admin customer view



IV. Admin seller view