

SLEEK MART

Home Appliances Ecommerce Website

Mini Project Report

Submitted by

Ishta Rachel Mathew

Reg. No.: AJC19MCA-I027

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, “**SLEEK MART**” is the bona fide work of **ISHTA RACHEL MATHEW (Regno: AJC19MCA-I027)** 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. Anit James

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 “**SLEEK MART**” 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: 31-10-2023

ISHTA RACHEL MATHEW

KANJIRAPPALLY

Reg: AJC19MCA-I027

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. Anit James** 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.

ISHTA RACHEL MATHEW

ABSTRACT

Modern conveniences like home appliances are made to make life easier and more enjoyable. They include a wide variety of mechanical and electronic appliances used for cleaning, amusement, climate control, and other purposes. This equipment save time and effort, from refrigerators storing food to washing machines simplifying laundry tasks. Innovation is still being driven by technological development, making appliances smarter and more energy-efficient to fit changing lifestyles. This project provides a wide variety of contemporary solutions for daily life. My Sleek Mart aspires to be the go-to place for those looking for top-notch home appliances, promoting a more organized and pleasurable way of living.

Three modules make up the bulk of this project:

- Admin
- Customers
- Sellers

To access the website, customers must create an account using their name, password and other details. Once registered, they can choose from a variety of budget-friendly options across different categories.

Admin has access to the details of customers and sellers who engage with the website. Admins also manage product categories, ensuring an organized catalog. Additionally, they oversee user accounts, including user registrations, account approvals, and user roles. Customers can select products, add them to their cart, make online payments. Sellers can register and create accounts on the platform. They can list their home appliances for sale, manage their product inventory, update product information, and monitor the needs of their customers.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	2
1.1	PROJECT OVERVIEW	3
1.2	PROJECT SPECIFICATION	3
2	SYSTEM STUDY	5
2.1	INTRODUCTION	6
2.2	EXISTING SYSTEM	6
2.3	DRAWBACKS OF EXISTING SYSTEM	7
2.4	PROPOSED SYSTEM	8
2.5	ADVANTAGES OF PROPOSED SYSTEM	8
3	REQUIREMENT ANALYSIS	10
3.1	FEASIBILITY STUDY	11
3.1.1	ECONOMICAL FEASIBILITY	11
3.1.2	TECHNICAL FEASIBILITY	11
3.1.3	BEHAVIORAL FEASIBILITY	12
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	13
3.2	SYSTEM SPECIFICATION	15
3.2.1	HARDWARE SPECIFICATION	15
3.2.2	SOFTWARE SPECIFICATION	15
3.3	SOFTWARE DESCRIPTION	15
3.3.1	PYTHON	15
3.3.2	DJANGO	16
3.3.3	SQLITE	16
4	SYSTEM DESIGN	18
4.1	INTRODUCTION	19
4.2	UML DIAGRAM	19
4.2.1	USE CASE DIAGRAM	20
4.2.2	SEQUENCE DIAGRAM	22
4.2.3	STATE CHART DIAGRAM	24
4.2.4	ACTIVITY DIAGRAM	25
4.2.5	CLASS DIAGRAM	26
4.2.6	OBJECT DIAGRAM	28

4.2.7	COMPONENT DIAGRAM	30
4.2.8	DEPLOYMENT DIAGRAM	31
4.3	USER INTERFACE DESIGN USING FIGMA	33
4.4	DATABASE DESIGN	36
5	SYSTEM TESTING	46
5.1	INTRODUCTION	47
5.2	TEST PLAN	47
5.2.1	UNIT TESTING	48
5.2.2	INTEGRATION TESTING	49
5.2.3	VALIDATION TESTING	49
5.2.4	USER ACCEPTANCE TESTING	49
5.2.5	AUTOMATION TESTING	50
5.2.6	SELENIUM TESTING	50
6	IMPLEMENTATION	67
6.1	INTRODUCTION	68
6.2	IMPLEMENTATION PROCEDURE	68
6.2.1	USER TRAINING	69
6.2.2	TRAINING ON APPLICATION SOFTWARE	69
6.2.3	SYSTEM MAINTENANCE	69
7	CONCLUSION & FUTURE SCOPE	70
7.1	CONCLUSION	71
7.2	FUTURE SCOPE	71
8	BIBLIOGRAPHY	73
9	APPENDIX	75
9.1	SAMPLE CODE	76
9.2	SCREEN SHOTS	95

List of Abbreviation

HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheet
SQLite	–	Structured Query Language Lite
UML	–	Unified Modelling Language
JS	–	JavaScript
AJAX	–	Asynchronous JavaScript and XML Environment
RDBMS	–	Relational Database Management System
IDE	–	Integrated Development Environment
ML	–	Machine Learning
BDD	–	Behavioral-Driven Development
UAT	–	User Acceptance Test
URL	–	Uniform Resource Locator
IP	–	Internet Protocol
QMG	–	Question Administration Gather

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The "Sleek Mart" project is an advanced online store precisely created to provide customers looking for premium home appliances with an amazing shopping experience. The way we manage everyday duties has been revolutionized by home appliances, which straddle the line between contemporary convenience and lifestyle improvement. These gadgets have become essential, from kitchen appliances that streamline cooking tasks to smart home technology that redefines living quarters. A wide range of users, including visitors, registered users, administrators, vendors, and delivery agents, are catered for by the platform's intelligent design. Visitors are welcome to browse a thorough product catalog and learn important details about various home appliances. However, registered users are granted several benefits, including account setup, profile maintenance, and customized purchasing experience. A seamless purchasing procedure is further enhanced by adding items to the shopping cart and safely completing transactions through linked payment gateways. The platform's effective operation depends on administrators, who oversee user administration, categories, and product inventory. A key player in the ecosystem, sellers can sign up, offer their appliances for sale, and actively monitor product listings.

The primary concern in achieving these objectives is to improve the entire user experience through logical design, easy navigation, and thorough product information. By putting in place strong security measures and trustworthy communication channels, we are committed to establishing trust and maintaining a secure environment. To foster a sense of community and transparency, we work to encourage meaningful connections between vendors and buyers. One of our main goals is to operate the platform effectively, which includes streamlining order processing and inventory procedures.

1.2 PROJECT SPECIFICATION

This is a website in which users can purchase several types of home appliances on this website.

There are 3 actors in the system. They are as follows:

- **Admin.**
 1. After the sellers have registered, accept, or reject them.
 2. Accept or reject a delivery person's registration.
 3. Managing product categories

4. Organizing the catalog
5. Handling user accounts

- **Sellers**

1. Seller Registration
2. Profile Management
3. Product Listing
4. Product Management
5. Order Management
6. Sales Report Generation

- **Customers**

1. Can purchase the products and can make payment.
2. Products can be added to the cart and wish list.
3. They can view product reviews and ratings.
4. They can view their order history.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis, which involves acquiring and examining data to pinpoint issues and offer solutions, is an essential stage of system development. Effective communication between system users and developers is crucial during this stage. A system analysis should always be the first step in any system development process. By closely analyzing the performance of the current system, the system analyst assumes the position of an investigator. Included are the identification of the system's inputs and outputs as well as a link between its operations and organizational outcomes.

Information is gathered through a variety of ways, including surveys and interviews. The goals are to comprehend how the system works, find problem areas, and recommend solutions to the problems the company is experiencing. The role of the problem-solver is taken on by the designer, and the proposed fixes are thoroughly compared with the existing system. After the best option is selected, the user is given the option to accept or reject the recommendation. Once the proposal has been considered considering their feedback, the process is repeated until the user is satisfied.

A preliminary study is the procedure of collecting and assessing data for upcoming system investigations. To guarantee the success of a system development project, a comprehensive preliminary investigation must be conducted.

2.2 EXISTING SYSTEM

It takes time to travel to many places to evaluate goods and costs, which makes it unlikely for those with hectic schedules. Additionally, the selection of products that are offered may be constrained to what is in stock or accessible at a specific location, limiting the options for clients. Geographical restrictions can also be a big problem, especially for people who live in distant areas or locations with few physical retailers nearby. Customers may struggle to locate appliances in these circumstances or may need to drive far to go to a store. This not only adds to the time and effort needed, but it also raises the expense of transportation.

2.2.1 NATURAL SYSTEM STUDIED

Customers generally interact with brick-and-mortar retailers in the current home appliance purchasing system. They go to these actual shops or showrooms to look around and get the appliances they require. Due to a store's limited selection or availability, customers frequently struggle to locate specific appliances. This requires going to several stores, which takes time and effort.

An effective and practical solution to these drawbacks is an e-commerce platform for household appliances. Customers may browse, compare, and buy a variety of appliances from the convenience of their homes thanks to this digital platform. It removes the need for protracted travel and offers access to a wider selection of goods. Customers can also gain from open pricing, thorough product descriptions, and a variety of payment alternatives, all of which improve their entire buying experience.

2.2.2 DESIGNED SYSTEM STUDIED

The method created for buying appliances for the home is an online store offering a variety of home goods. It is accessible to users through a user-friendly website or mobile app with well-organized product categories and simple navigation. Each product comes with thorough descriptions, pictures, and prices. The technology makes it possible to conduct effective searches and to enjoy easy shopping, from adding items to the cart to making safe online payments. It provides support services, user feedback, and purchase tracking.

The total buying experience is improved by special discounts and a wish list option, making it convenient and client focused. Users gain trust and confidence as a result of the seamless and safe transaction process it ensures while making online transactions. The system also includes a feedback feature that enables users to provide reviews and ratings that help other people make informed judgments.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Limited Accessibility:** Physically visiting brick-and-mortar establishments is required for customers, which can be problematic, especially for those with hectic schedules.
- **Restricted Choice:** Customers' options may be limited in physical stores due to a

smaller selection of products than on online marketplaces.

- **Lack of Information:** It may be difficult for customers to make well-informed purchasing selections in traditional stores since they may not have access to comprehensive product information, specifications, or user reviews.
- **Price Comparisons:** It might be challenging for customers to compare features and costs across several stores, which could lead to overpaying for an item.
- **Time-consuming:** Visiting several physical stores to evaluate appliances takes time, which may discourage buyers and lead them to make a less-than-ideal decision.
- **Geographical restrictions:** Customers may have fewer options due to limited access to brick-and-mortar stores in rural or less populated locations.
- **Inconvenient Returns and Exchanges:** Compared to online platforms with simple return procedures, returning or exchanging appliances purchased from a physical store might be more difficult and time-consuming.

2.4 PROPOSED SYSTEM

The suggested system for buying appliances for the home envisions a comprehensive online platform where clients can easily browse and choose a variety of appliances from the convenience of their homes. By offering a wide variety of appliances, comprehensive product information, customer feedback, and affordable price, this approach will get beyond the drawbacks of the current brick-and-mortar retailers. The platform will also provide a user-friendly layout and simple navigation, guaranteeing a smooth shopping experience. This system attempts to increase client pleasure and foster trust by incorporating safe payment channels and dependable delivery services. Additionally, putting in place an attentive customer support system will swiftly address any questions or issues, increasing the overall efficacy and efficiency of the suggested online platform for home appliances.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- **Convenience:** The proposed system offers a high level of convenience to customers. They can access the platform anytime and anywhere with an internet connection, allowing them to browse, select, and purchase home appliances at their own

convenience. This eliminates the need for physical travel, saving time and effort.

- **Expanded Product Availability:** The proposed system can provide a broader range of home appliance options compared to traditional offline stores. It can connect customers with various appliance suppliers, including local and international sources, increasing the availability of different appliance models. This expanded product availability gives customers more choices and opportunities to find specific appliances they are looking for.
- **Detailed Product Information:** The proposed system can provide detailed product information for each appliance. Customers can access comprehensive descriptions, including details about the appliance's specifications, features, dimensions, and potential uses. This information allows customers to make informed decisions and select appliances that align with their specific needs and preferences.
- **User Reviews and Ratings:** The proposed system can incorporate user reviews and ratings for appliances. Customers can read feedback and experiences shared by other buyers who have previously purchased and used the appliances. User reviews can provide valuable insights into the quality, performance, and durability of the appliances, helping customers make more informed purchasing decisions.
- **Efficient Search Options:** The proposed system can include advanced search and filtering options to facilitate appliance selection. Customers can use specific criteria such as appliance type, size, energy efficiency, or preferred features to narrow down their options and find the desired appliances more efficiently. This saves time and helps customers find appliances that meet their specific requirements.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

This analysis is a fundamental step in determining if a project will meet the association's goals in relation to the resources, effort, and time invested in it. It aids the designer in determining the project's potential focus points and long-term outcomes. To find out if a given framework is feasible and advantageous for further research, it is necessary to consider all options, impact of the proposed system on the association, assessments of resource efficiency, client satisfaction, and capabilities to meet client requests all form part of the contemplate the prospect. Consequently, an achievability analysis is conducted on a regular basis. Recently, approval was provided for the creation of a contemporary application. The specific, monetary, and operational justification of the scope is as it were.

3.1.1 Economic Feasibility

To assess a new project's value in terms of time and money commitment, it is essential to conduct an economic feasibility analysis. It entails a careful examination of every issue that could affect the initiative's outcome. The suggested system, Sleek Mart, has undergone cost-benefit analysis and is proven to be both practical and cost effective given the project's presumptive cost. Various cost categories were evaluated, including labor costs, computer costs, supplies and equipment costs, charges for implementing new software and computer equipment, system analysis, website coding costs, and database design costs, to establish the system's development cost. These are often one-time costs that won't be incurred again once the project is complete. These cost categories can be thoroughly examined to make sure the system's development is feasible from an economic standpoint and will provide a profit.

- The cost conducts a full system investigation?

The proposed system is developed as part of the project work, there is no manual cost to spend for the proposed system.

- The cost of hardware and software?

All the resources are already available.

3.1.2 Technical Feasibility

Technical feasibility is the process of evaluating whether it is feasible to create and deploy a good or service with the technology and resources currently in use. The proposed plan's tools, materials, labor, logistics, and technology are examined as part of the technical

feasibility analysis to gauge how effective 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.

The Sleek Mart is simple to use and doesn't need much instruction because it is self-explanatory. Even for first-time users, the application is simple to use. The technology is easily accessible, saving clients' money, with only the time spent online being wasted. It is technically feasible due to reliable eCommerce frameworks, secure payment gateways, responsive design, and efficient inventory and user management. Continuous improvement using feedback and analytics further enhances its viability as a successful online home appliance store.

- Is the project feasible within the limits of current technology?
Yes
- Technical issues raised during investigation are:
Nothing
- Can technology be easily applied to current problem?
Yes
- Does the technology have the capacity to handle the solution?
Yes

3.1.3 Behavioral Feasibility

Behavioral feasibility refers to the assessment of whether a proposed project or system aligns with the organization's culture, existing processes, and the willingness of users to adapt and adapt to the changes brought about by the project. It evaluates the human and behavioral aspects of implementing the project and addresses potential challenges related to acceptance, resistance, and support from stakeholders and end-users.

Two crucial considerations have been made to guarantee the system's success:

- (1) if users will have enough assistance, and
- (2) whether the system will be harmful.

To make sure that the system will be useful after implementation, these issues were carefully examined. Additionally, to make sure the project is behaviorally feasible, all behavioral

elements were considered throughout the feasibility assessment. Overall, it is anticipated that the proposed system will be highly successful in achieving its goals. It is behaviorally feasible due to its user-friendly interface, convenient shopping experience, efficient order processing, and seller engagement.

- Is there sufficient support for the users?
Yes
- Will the proposed system cause harm?
No

3.1.4 Feasibility Study Questionnaire

1. Project Overview?

This project aims to provide users with a convenient online platform to shop for a wide range of high-quality home appliances. With the website, customers can easily find and purchase appliances that suit their specific needs, enhancing their daily routines and adding efficiency to their homes.

The website offers a diverse selection of appliances at affordable prices. To use the website, customers need to register with their name and password, gaining access to various features and personalized recommendations. The admin oversees user interactions and seller details, ensuring a secure and reliable shopping environment.

Customers can explore different appliance categories, add their desired products to the cart, make secure online payments, and have their orders delivered to their doorstep. Sellers can update their product listings, ensuring they meet the evolving needs of customers.

2. To what extent is the system proposed for?

This project aims to offer a seamless and timesaving shopping experience, eliminating the need to visit multiple stores and simplifying the process of finding the perfect home appliances. By providing online access to premium products and easy price comparisons, the website ensures users have a convenient and satisfying shopping journey from start to finish.

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

Potential Customers, General Visitors, Online Shoppers

4. List the Modules included in your System?

Admin, Guest Users, Customers, Sellers, and Delivery Boy

5. Identify the users in your project?

Guest Users, Customers

6. Who owns the system?

Administrator

7. System is related to which firm/industry/organization?

Home Appliances Industry

8. Details of person that you have contacted for data collection.

Thomas Abraham(White QMart, Chalapally)

9. Questionnaire to collect details about the project?**1. How do you buy appliances for the shop?**

Buying from the distributors

2. How the payment is collected from the user?

By cash or credit card facility

3. Is there any replacement for the product delivered?

Yes. Replacement of the delivered product is available only if any damage is found.

4. Do you use any inventory management software or systems to keep track inventory?

No. Maintain a manual inventory system where each appliance is recorded in a physical logbook. Update the logbook whenever a product is sold or restocked.

5. How do you decide on the pricing of your appliances?

The pricing model is based on a combination of factors, including the manufacturer's suggested retail price (MSRP), our purchasing costs.

6. Do you offer any special discounts or promotions?

Offer special discounts during festive seasons.

7. How do you handle customer inquiries and orders?

Use a combination of phone support and in-person assistance for customer inquiries and

orders.

8. What measures do you take to ensure product quality and customer satisfaction?

The shop offers warranty coverage on most appliances. Assist customers with warranty claims and ensure a smooth process for any repairs or replacements.

9. How do you handle product deliveries?

In addition to in-house delivery, also partner with reliable courier services for certain deliveries.

10. How do you market your shop and attract new customers?

The Shop utilizes loyalty programs.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - AMD Ryzen 7 7735HS

RAM - 16.0 GB

Hard disk - 1 TB

3.2.2 Software Specification

Front End - HTML, CSS

Back End - Python, Django

Database - SQLite

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, jQuery, Python, Django, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 Python

Python is a versatile and widely used programming language, prized for its readability and ease of use. Its simple syntax and extensive library support, including frameworks like Django and Flask, expedite development. Python finds applications across web development, data analysis, machine learning, and more. Its platform independence and scalability make it suitable for various operating systems and evolving project needs. In the

proposed system, Python forms the foundation for backend development, core functionality, data processing, and integration with different tools and frameworks, enabling the creation of an efficient online platform for purchasing home appliances.

3.3.2 Django

Django is a high-level Python web framework known for its simplicity and efficiency in web application development. It follows the "batteries-included" philosophy, offering a wide range of built-in features and tools that empower developers to create robust, scalable, and secure web applications. Django promotes the DRY (Don't Repeat Yourself) principle, making it easy to build complex applications with clean, maintainable code. It includes an Object-Relational Mapping (ORM) system for database interactions, a templating engine for designing user interfaces, and a secure authentication system. Django's built-in admin interface simplifies content management and data handling. With a strong community, extensive documentation, and a rich ecosystem of third-party packages, Django remains a top choice for web developers aiming to streamline the development process and deliver high-quality web applications.

3.3.3 SQLite

SQLite is a lightweight and self-contained relational database management system. It's known for its simplicity, speed, and ease of integration, making it a popular choice for embedded systems and applications. SQLite operates without a separate server process and allows direct access to the database using a simple and efficient query language. It's ideal for small to medium-sized applications, especially those that need a local data store. In the proposed system, SQLite serves as the backend database, efficiently managing and storing essential data related to home appliances, user accounts, transactions, and more, ensuring a seamless and reliable user experience.

- **Self-Contained:** SQLite is a self-contained DBMS, meaning it doesn't require a separate server process. The entire database is stored in a single file on the disk.
- **Zero Configuration:** Unlike many other database systems, SQLite requires minimal to no configuration. You can start using it by just including the library in your project.
- **Serverless Architecture:** It operates without a central server, allowing applications to access the database directly, simplifying setup and reducing

latency.

- ACID Compliant: SQLite ensures data reliability through ACID (Atomicity, Consistency, Isolation, Durability) compliance, guaranteeing that transactions are processed reliably.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The design phase is the first step in the development of any designed system or product. 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. In software engineering, the design phase is crucial regardless of the development paradigm selected. It serves as the technical backbone of the software engineering process and aims to create the architectural detail required to build a system or product.

This program underwent a careful design phase to maximize every area of efficiency, performance, and accuracy. During the design phase, a user-oriented document is transformed into a document for programmers or database staff.

4.2 UML DIAGRAM

The Unified Modelling Language (UML) is a standard dialect used for planning, conceptualizing, describing, and visualizing program frameworks. The Question Administration Gather (QMG) proved to be a reliable source for UML creation; the UML 1.0 definition's first draft was released in January 1997. UML is not the same as programming dialects like Java, C++, and COBOL. It could be a pictorial language used for program blueprints and a nonexclusive visual exhibiting language used for computer program frameworks. Although UML is generally used to talk to program 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 that shows how users and other external characters interact with the internal parts of a system. The fundamental task of a usage case diagram is to identify, organize, and plan out a system's functional requirements as perceived by its users. Use case diagrams are commonly created using the Unified Modelling Language (UML), a standard language for modelling real objects and systems.

Use cases can be employed to accomplish a variety of framework goals, such as establishing basic requirements, verifying equipment plans, testing and analyzing programs, generating online help references, or carrying out client support duties. In the context of item deals, customer service, product acquisition, catalog revamping, and payment processing are essentially a few examples of use cases.

A use case diagram is composed of the system boundaries, actors, use cases, and their relationships. The boundaries of the system are established with respect to its surroundings by the system boundary. The roles that actors play and how they represent the individuals or systems that interact with the system are frequently used to characterize them. Lastly, the graphics display both the use cases and the relationships between the actors and the cases.

Use case diagrams are visual aids that help convey a system's functional needs. To produce an efficient and effective diagram, it's crucial to adhere to following rules while creating a use case diagram:

- Give use cases names that appropriately describe the functions they carry out.
- Give actors the proper names to assist them understand their duties within the system.
- Verify that the diagram accurately illustrates the links and interdependence.
- Refrain from mentioning every relationship that might exist because the main objective is to determine the necessary conditions.
- When needed, go to your notes to reaffirm key topics.

By adhering to these recommendations, we can produce a use case diagram that accurately depicts the system's functional needs and is clear and simple.

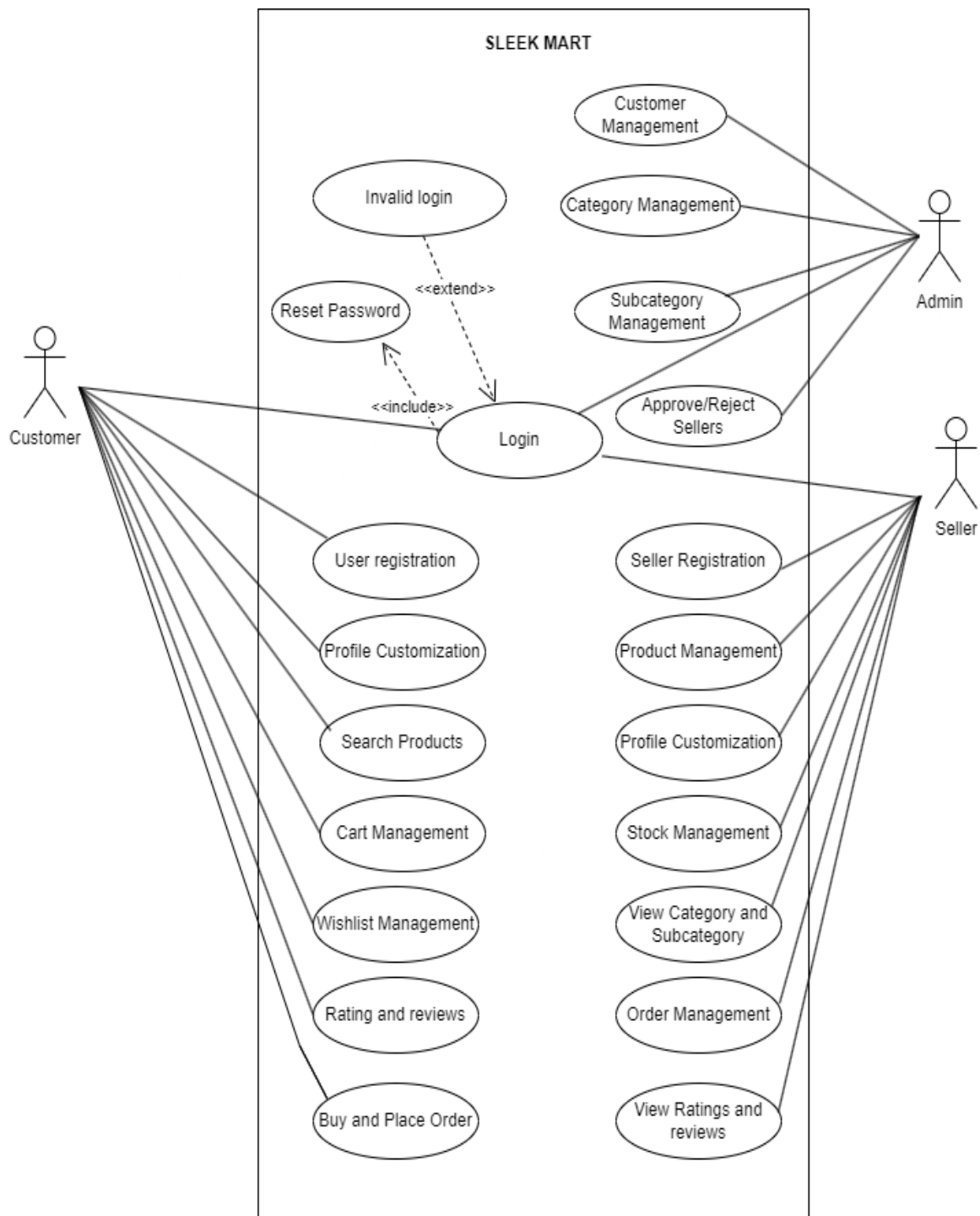


Fig 1: Use case diagram for Sleek Mart

4.2.2 SEQUENCE DIAGRAM

The chronological order of interactions between various system components is shown in a sequence diagram, a form of interaction diagram. It demonstrates how several things communicate with one another over the course of a series of messages. These images are sometimes referred to as event scenarios or event scenarios diagrams. In software engineering, sequence diagrams are frequently used to describe and comprehend the needs of both new and old systems. They support the visualization of object control relationships and the detection of systemic issues.

Sequence Diagram Notations –

i. Actors – In UML, a role that interacts with the system and its objects is represented by an actor. Actors frequently exist outside of the system that the UML diagram is intended to portray. Actors can play a variety of roles, including those of external topics or human users. A stick person notation is used in UML diagrams to represent actors. Depending on the situation that is being modelled, a sequence diagram may have more than one actor.

ii. Lifelines – A lifeline in a sequence diagram is a vertical dashed line that represents the lifespan of an object participating in the interaction. Each lifeline represents an individual participant in the sequence of events and is labeled with the name of the participant. The lifeline shows the timeline of events for the participant and is drawn as a vertical line extending from the participant's activation point to its deactivation point.

iii. Messages - Messages are a key component of sequence diagrams, representing the interactions and communication between objects or components in a system. They can be categorized into synchronous and asynchronous messages, create and delete messages, self-messages, reply messages, found messages, and lost messages. Guards are also used to model conditions and restrictions on message flow.

iv. Guards- Guards in UML are used to model conditions and are employed to restrict the flow of messages when a certain condition is met. This feature is essential for letting software developers know about any constraints or limitations associated with a system or a particular process.

Uses of sequence diagram –

- Modeling and visualizing the logic of complex functions, operations, or procedures.
- Showing details of UML use case diagrams.
- Understanding the detailed functionality of current or future systems.
- Visualizing how messages and tasks move between objects or components in a system.
- Overall, sequence diagrams are useful for representing the flow of interactions between objects in a system, and can help both businesspeople and software engineers better understand and communicate system requirements and behavior.

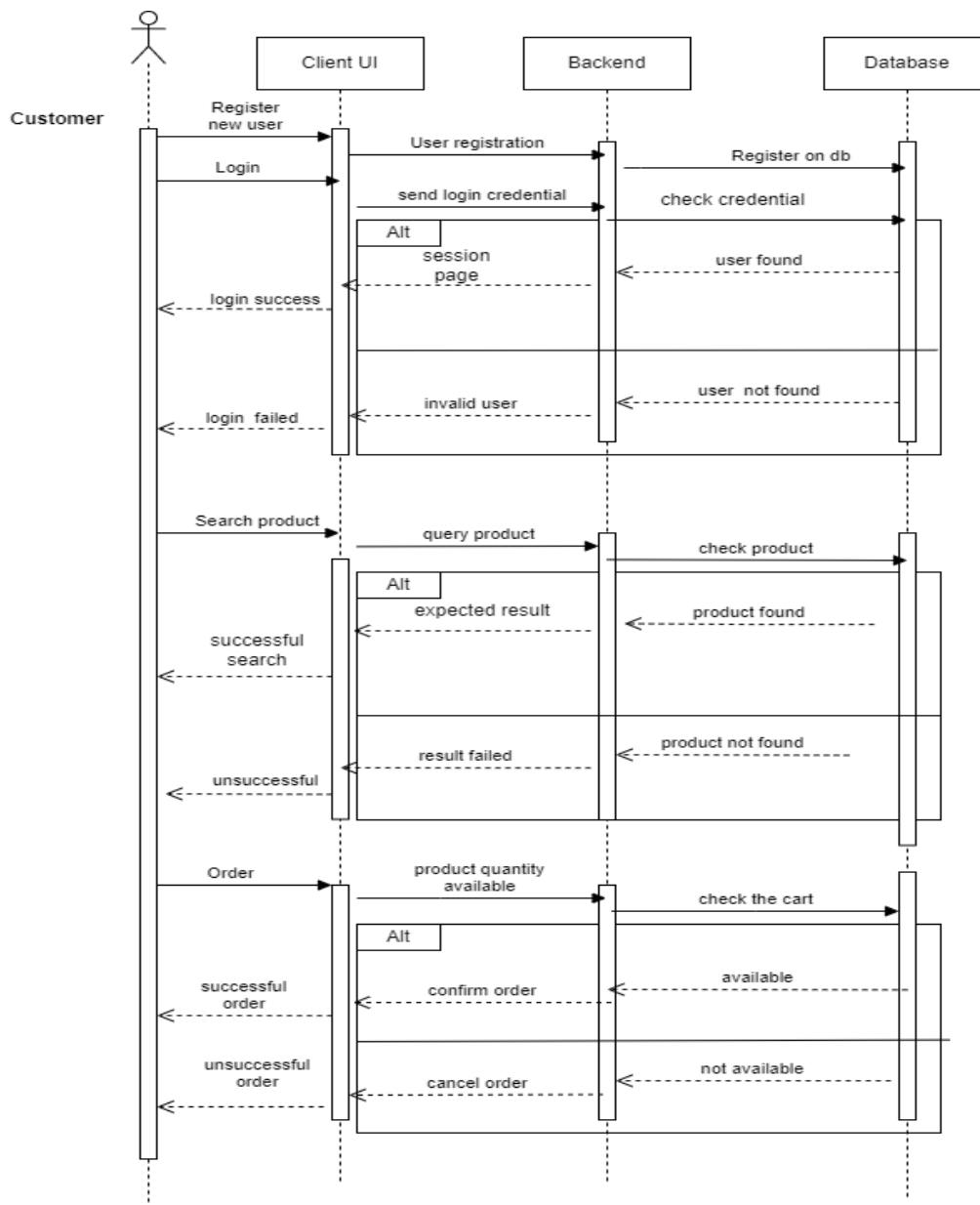


Fig 2: Sequence diagram for SLEEK Mart

4.2.3 State Chart Diagram

A state diagram is a visual representation, often created using the Unified Modeling Language (UML), that shows the different states that an object can exist in and how it can transition between those states. It is also referred to as a state machine diagram or state chart diagram.

The State Chart Diagram is a behavioral diagram in UML that describes the behavior of a system or object over time. It includes various elements such as:

- Initial State - This state represents the starting point of the system or object and is denoted by a solid black circle.
- State - This element describes the current state of the system or object at a specific point in time and is represented by a rectangle with rounded corners.
- Transition - This element shows the movement of the system or object from one state to another and is represented by an arrow.
- Event and Action - An event is a trigger that causes a transition to occur, and an action is the behavior or effect of the transition.
- Signal - A message or trigger caused by an event that is sent to a state, causing a transition to occur.
- Final State - The State Chart Diagram ends with a Final State element, which is represented by a solid black circle with a dot inside. It indicates that the behavior of the system or object has completed.

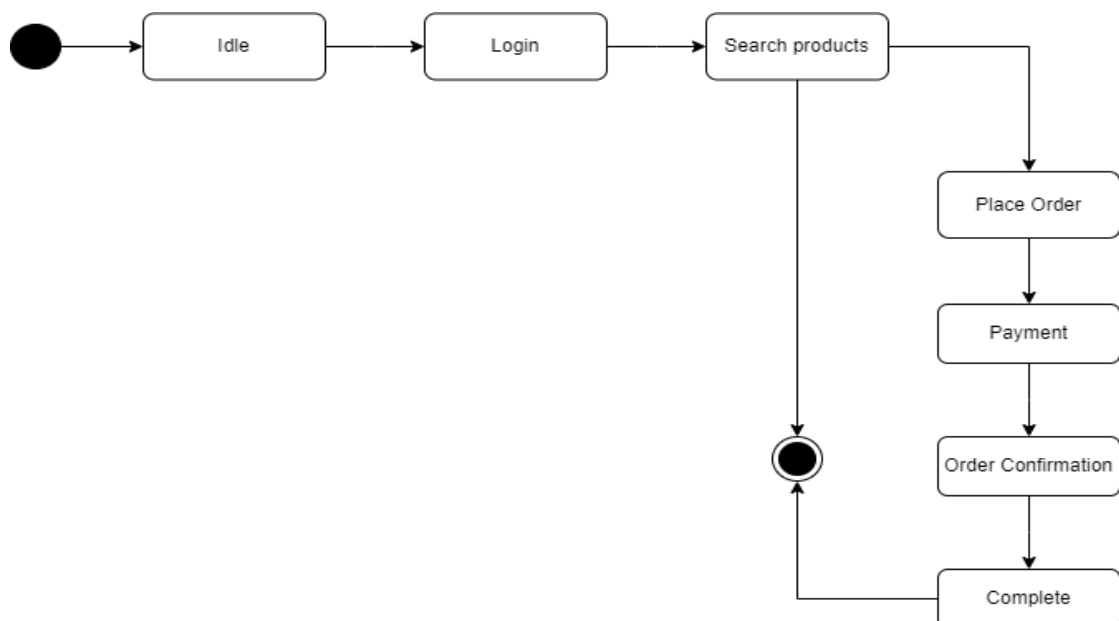


Fig 3: State Chart diagram for Slek Mart

4.2.4 Activity Diagram

An activity diagram is a visual representation of a workflow that shows how one activity leads to another. An activity is referred to as a system operation, and one operation leads to another in the control flow. A flow can be parallel, concurrent, or branched, and activity diagrams use various functions such as branching, joining, etc., to manage all types of flow control. Activity diagrams are a type of behavior diagram that shows the behavior of a system. They show the flow of control from the start point to the end point and show the different decision paths that exist during the execution of the activity.

The key components of an activity diagram are:

- Initial node - A starting point of the activity diagram, denoted by a black circle.
- Activity - A task or action performed by the system or entity, represented by a rectangle with rounded corners.
- Control flow - It represents the sequence of activities or actions performed by the system or entity, represented by an arrow.
- Decision node - A decision or branching point in the activity flow, denoted by a diamond shape.
- Merge node - Used to merge multiple branches of the activity flow into a single flow, represented by a diamond shape with a plus sign inside.
- Fork node - Used to split the activity flow into multiple parallel flows, represented by a solid black circle with multiple arrows.
- Join node - Used to join multiple parallel flows back into a single flow, represented by a solid black circle with multiple arrows pointing towards it.
- Final node - The end point of the activity diagram, denoted by a black circle with a dot inside.
- Object flow - Represents the flow of objects or data between activities, represented by a dashed arrow.

Activity diagrams are useful in clarifying complex processes, identifying potential issues, and communicating process flows to stakeholders and project team members.

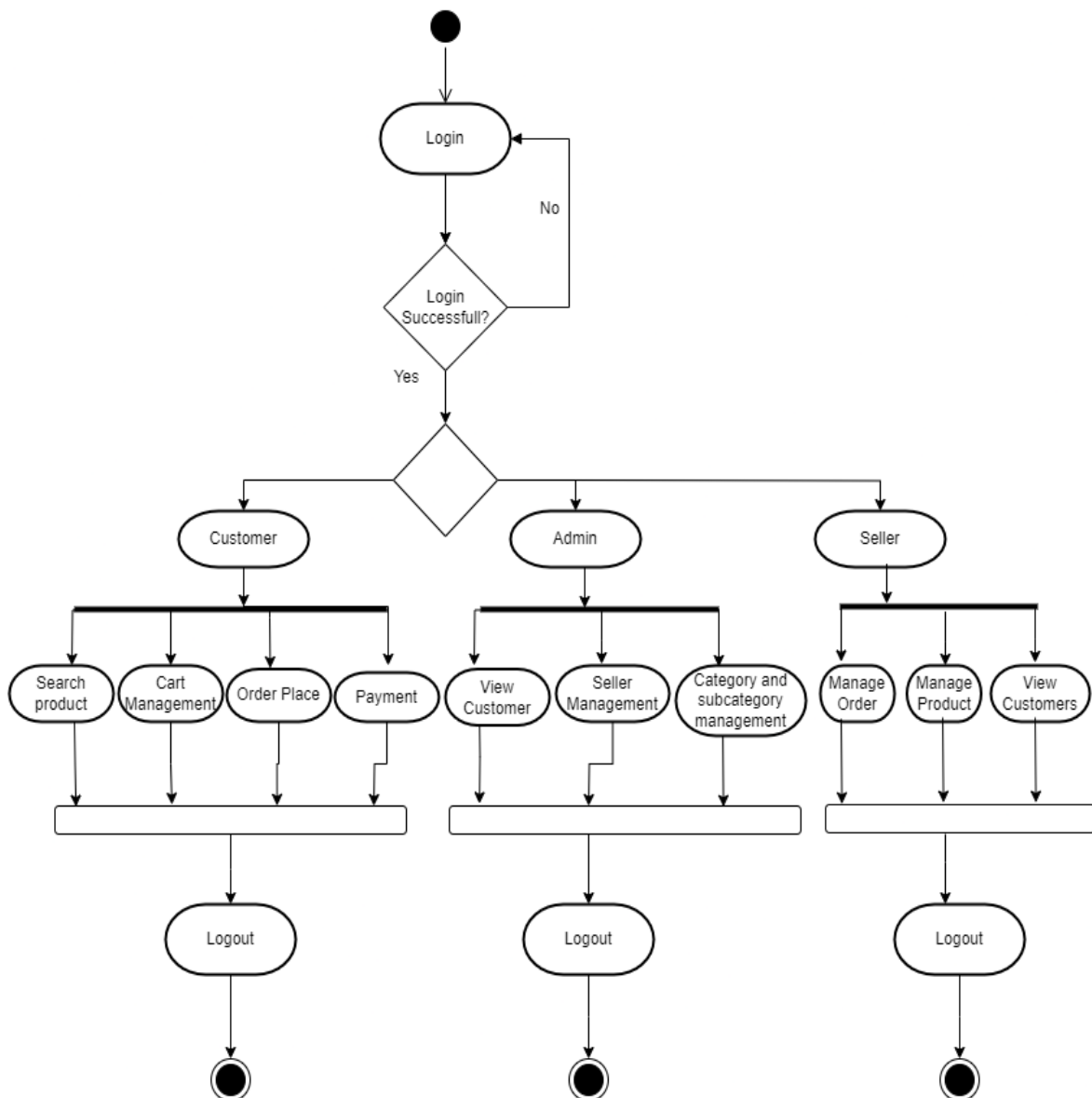


Fig 4: Activity diagram for SLEEK MART

4.2.5 Class Diagram

The class diagram is a fundamental component of object-oriented modeling and serves as the primary means of conceptual modeling for the structure of an application. Additionally, class diagrams can be used for detailed modeling that can be translated into programming code. They can also be employed for data modeling purposes. Class diagrams are a crucial component of UML used to represent classes, objects, interfaces, and their relationships and attributes in a system.

Some important components of a class diagram are:

- **Class:** It is a blueprint or template for creating objects and is represented as a rectangle with the class name, attributes, and methods.
- **Interface:** It is a collection of abstract methods that specify a contract between a class and the outside world. It is represented as a circle with the interface name inside.
- **Object:** It is an instance of a class with state and behavior. It is represented as a rectangle with the object name inside.
- **Association:** It is a relationship between two classes that represents a connection or link and is represented as a line with optional directionality, multiplicity, and role names.
- **Aggregation:** It is a part-whole relationship where the whole (aggregator) is composed of parts (aggregates) and is represented as a diamond shape on the aggregator side.
- **Composition:** It is a stronger form of aggregation where the parts cannot exist without the whole and is represented as a filled diamond shape on the aggregator side.
- **Inheritance:** It is a relationship between a superclass and its subclasses that represents an "is-a" relationship and is represented as a line with an open arrowhead pointing from the subclass to the superclass.
- **Dependency:** It is a relationship where a change in one class may affect the other class and is represented as a dashed line with an arrowhead pointing from the dependent class to the independent 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 essential in designing and modeling object-oriented software systems as they provide a visual representation of the system's structure, its functionality, and the relationships between its objects. They facilitate software development, maintenance, and improve communication among team members.

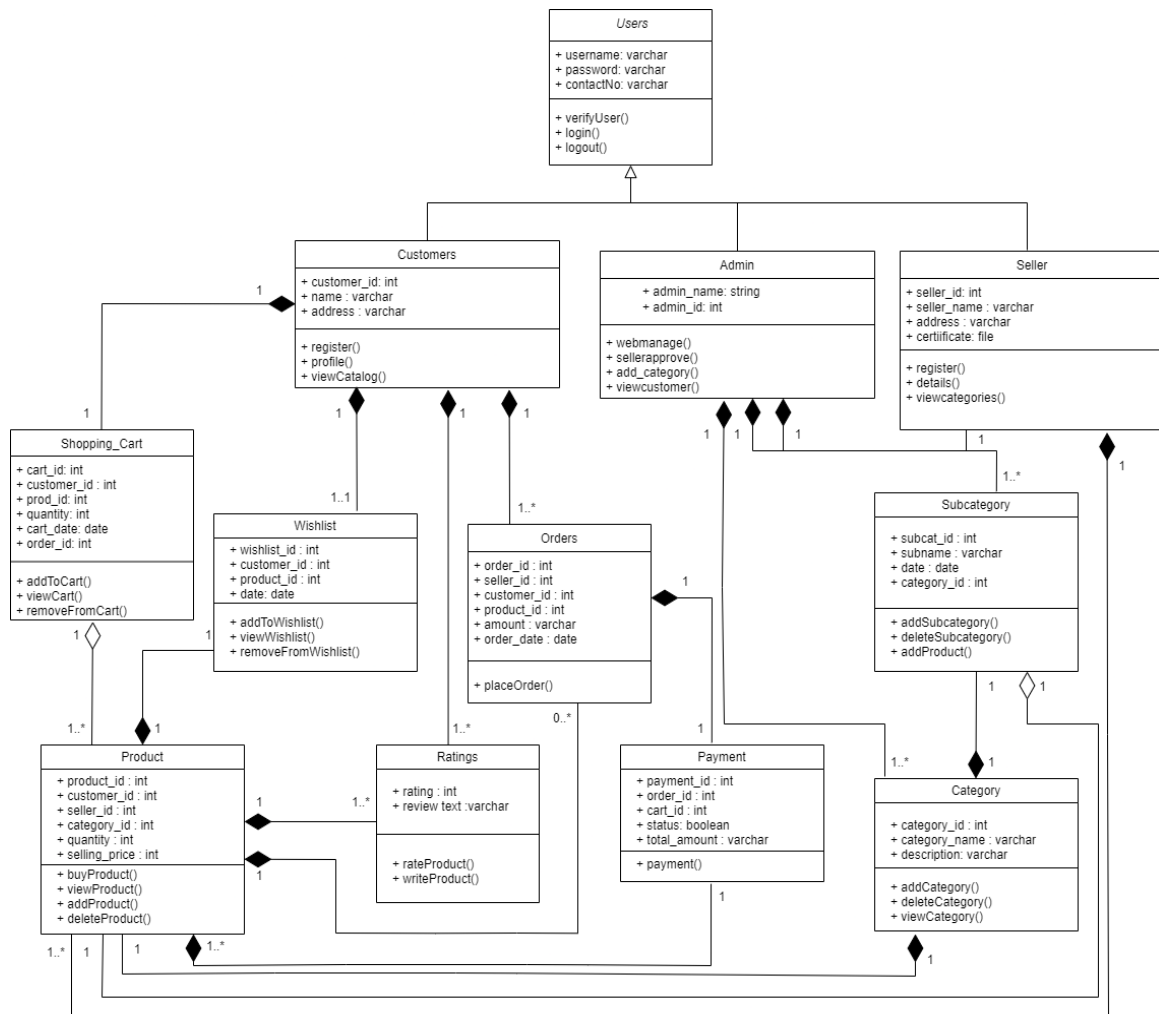


Fig 5: Class diagram for SLEEK MART

4.2.6 Object Diagram

Class diagrams and object diagrams are closely related in object-oriented modeling. Object diagrams are instances of class diagrams, which represent a snapshot of the system at a given moment in time. Both types of diagrams use the same concepts and notation to represent the structure of a system. While class diagrams are used to model the structure of the system, including its classes, attributes, and methods, object diagrams represent a group of objects and their connections at a specific point in time.

An object diagram is a type of structural diagram in UML that shows instances of classes and their relationships. The main components of an object diagram include:

- **Object:** An object is an instance of a class that represents a specific entity in the system. It is represented as a rectangle with the object name inside.

- **Class:** A class is a blueprint or template for creating objects that defines its attributes and methods. It is represented as a rectangle with three compartments for the class name, attributes, and methods.
- **Link:** A link is a relationship between two objects that represents a connection or association. It is represented as a line connecting two objects with optional labels.
- **Attribute:** An attribute is a property or characteristic of an object that describes its state. It is represented as a name-value pair inside the object rectangle.
- **Value:** A value is a specific instance or setting of an attribute. It is represented as a value inside the attribute name-value pair.
- **Operation:** An operation is a behavior or action that an object can perform. It is represented as a method name inside the class rectangle.
- **Multiplicity:** Multiplicity represents the number of instances of a class that can be associated with another class. It is represented as a range of values (e.g. 0..1, 1..*, etc.) near the link between objects.

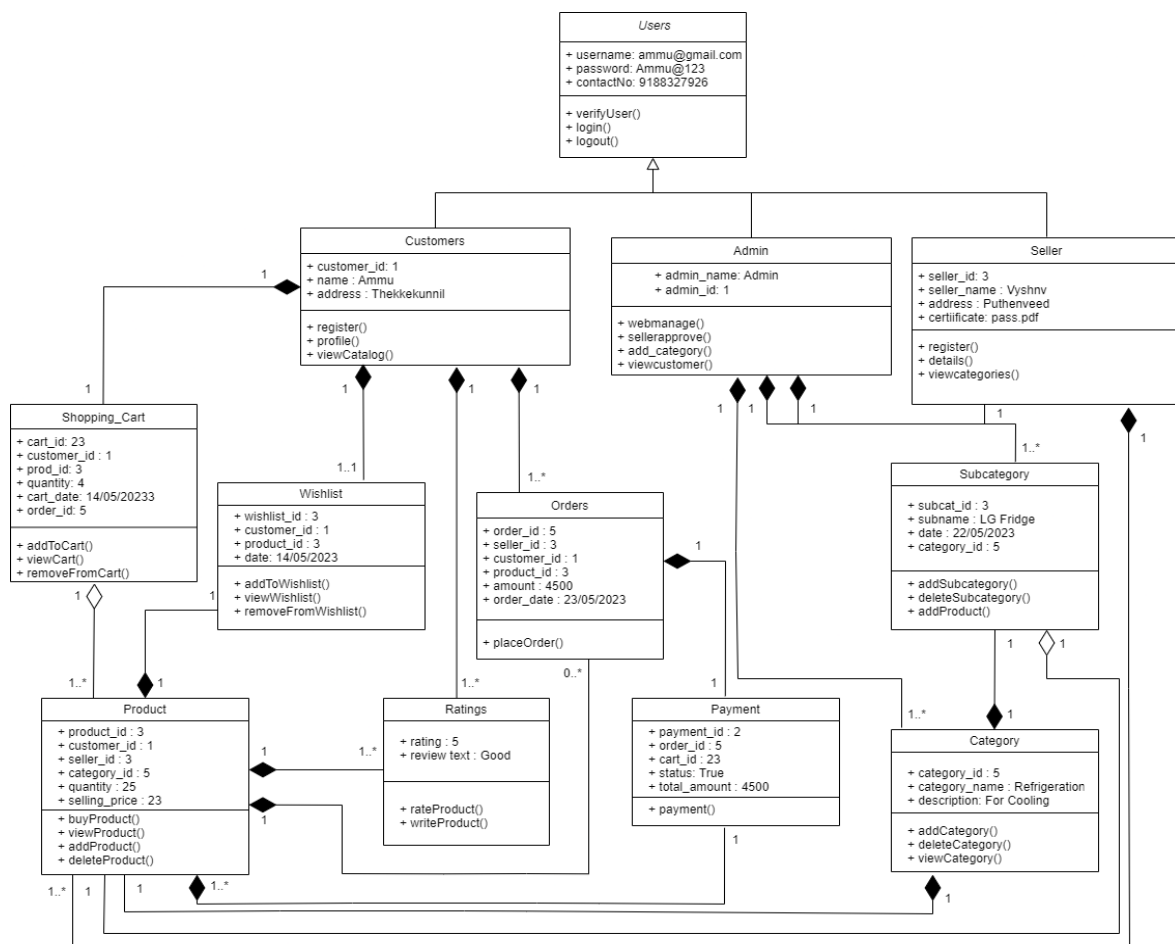


Fig 6: Object Diagram for SLEEK MART

4.2.7 Component Diagram

A component diagram in UML illustrates how various components are interconnected to create larger components or software systems. It is an effective tool for representing the structure of complex systems with multiple components. By using component diagrams, developers can easily visualize the internal structure of a software system and understand how different components work together to accomplish a specific task.

Its key components include:

- **Component:** A modular and encapsulated unit of functionality in a system that offers interfaces to interact with other components. It is represented as a rectangle with the component name inside.
- **Interface:** A contract between a component and its environment or other components, specifying a set of methods that can be used by other components. It is represented as a circle with the interface name inside.
- **Port:** A point of interaction between a component and its environment or other components. It is represented as a small square on the boundary of a component.
- **Connector:** A link between two components that enables communication or data exchange. It is represented as a line with optional adornments and labels.
- **Dependency:** A relationship between two components where one component depends on another for its implementation or functionality. It is represented as a dashed line with an arrowhead pointing from the dependent component to the independent component.
- **Association:** A relationship between two components that represents a connection or link. It is represented as a line connecting two components with optional directionality, multiplicity, and role names.
- **Provided/Required Interface:** A provided interface is an interface that a component offers to other components, while a required interface is an interface that a component needs from other components to function properly. These are represented by lollipops and half-circles respectively.

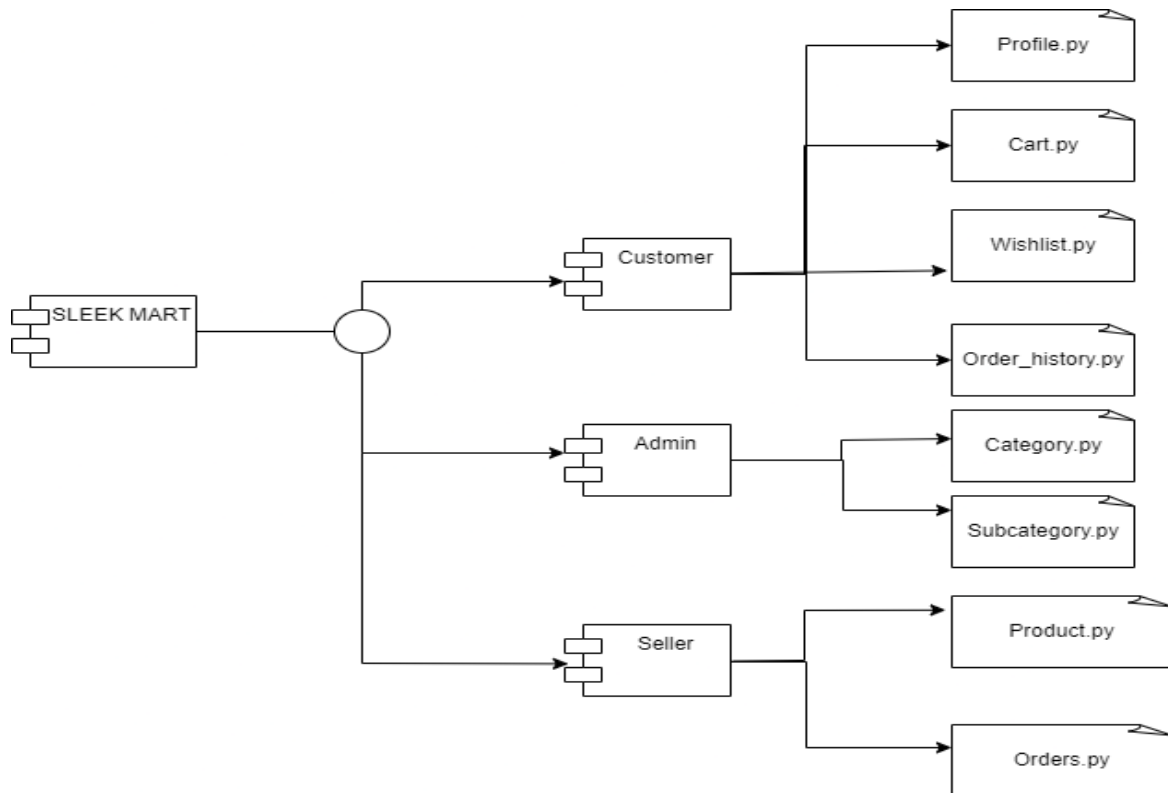


Fig 7: Component Diagram for Sleek Mart

4.2.8 Deployment Diagram

A deployment diagram is a type of UML diagram that focuses on the physical hardware used to deploy software. It provides a static view of a system's deployment and involves nodes and their relationships. The deployment diagram maps the software architecture to the physical system architecture, showing how the software will be executed on nodes. Communication paths are used to illustrate the relationships between the nodes. Unlike other UML diagram types, which focus on the logical components of a system, the deployment diagram emphasizes the hardware topology.

The key components of a deployment diagram are:

- **Node** - A node is a physical or virtual machine on which a component or artifact is deployed. It is represented by a box with the node's name inside.
- **Component** - A component is a software element that performs a specific function or provides a specific service. It is represented by a rectangle with the component's name inside.
- **Artifact** - An artifact is a physical piece of data that is used or produced by a component. It is represented by a rectangle with the artifact's name inside.

- **Deployment Specification** - A deployment specification describes how a component or artifact is deployed on a node. It includes information about the location, version, and configuration parameters of the component or artifact.
- **Association** - An association is a relationship between a node and a component or artifact that represents a deployment dependency. It is represented by a line connecting the two components with optional directionality, multiplicity, and role names.
- **Communication Path** - A communication path represents the connection between nodes, such as a network connection or communication channel. It is represented by a line with optional labels and adornments.

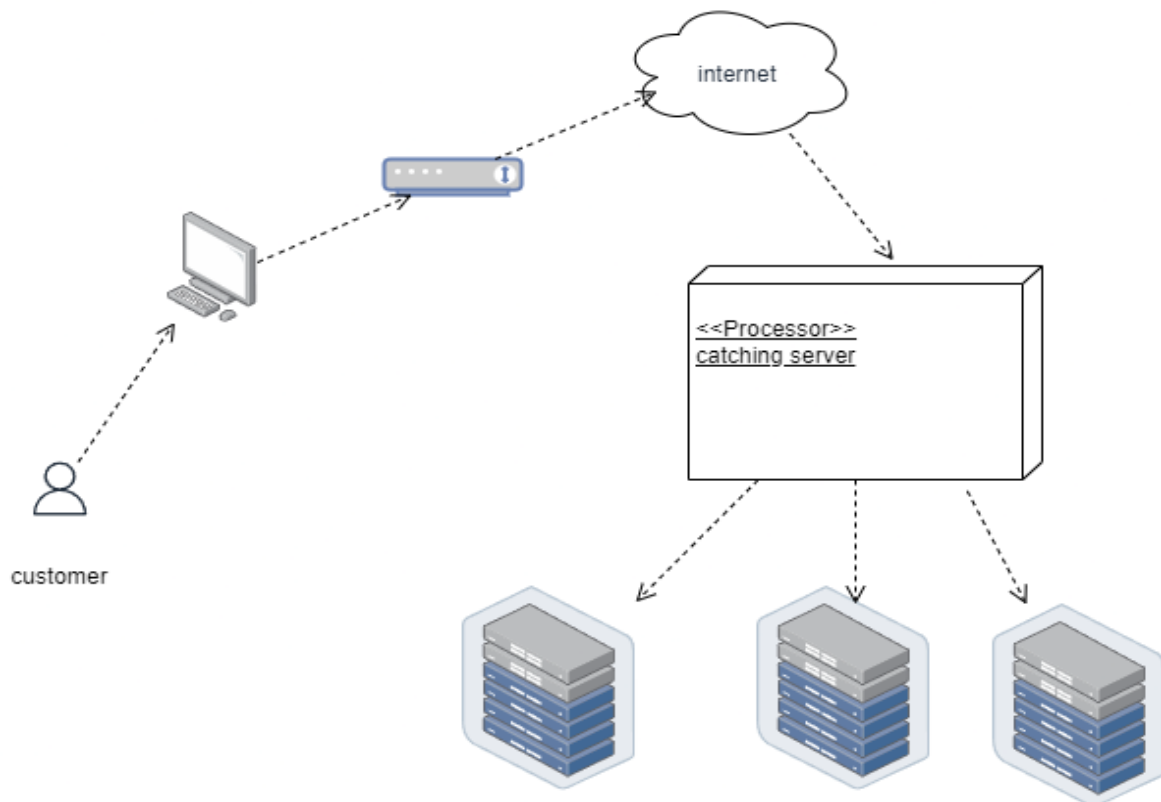


Fig 8: Deployment Diagram for Sleek Mart

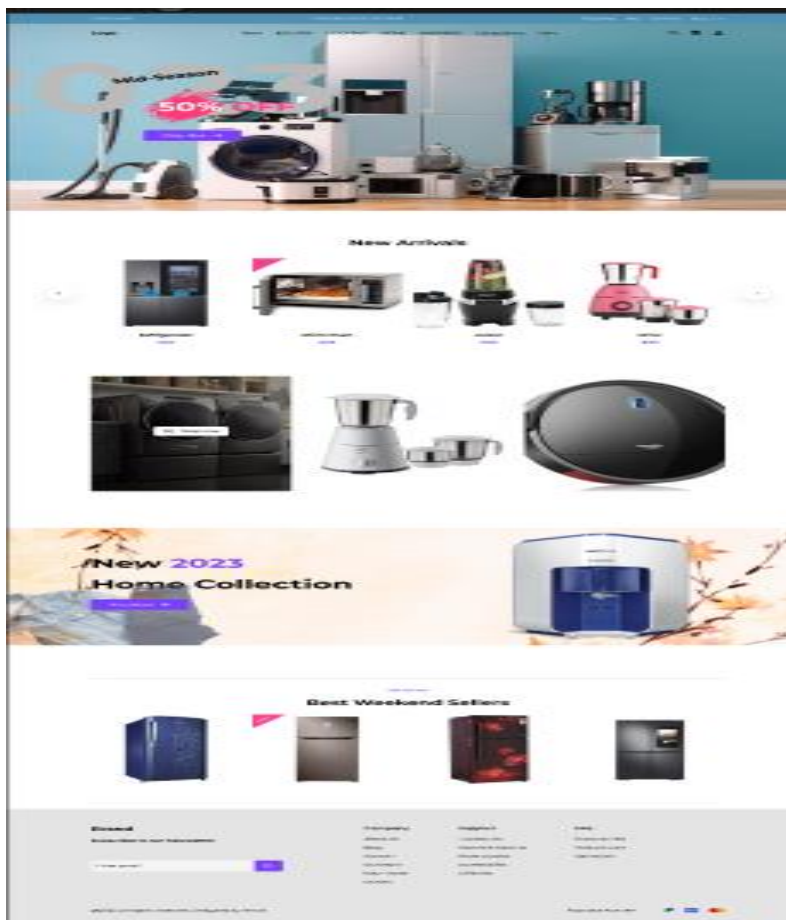
4.3 USER INTERFACE DESIGN USING FIGMA

1. Form Name: SignUp Form

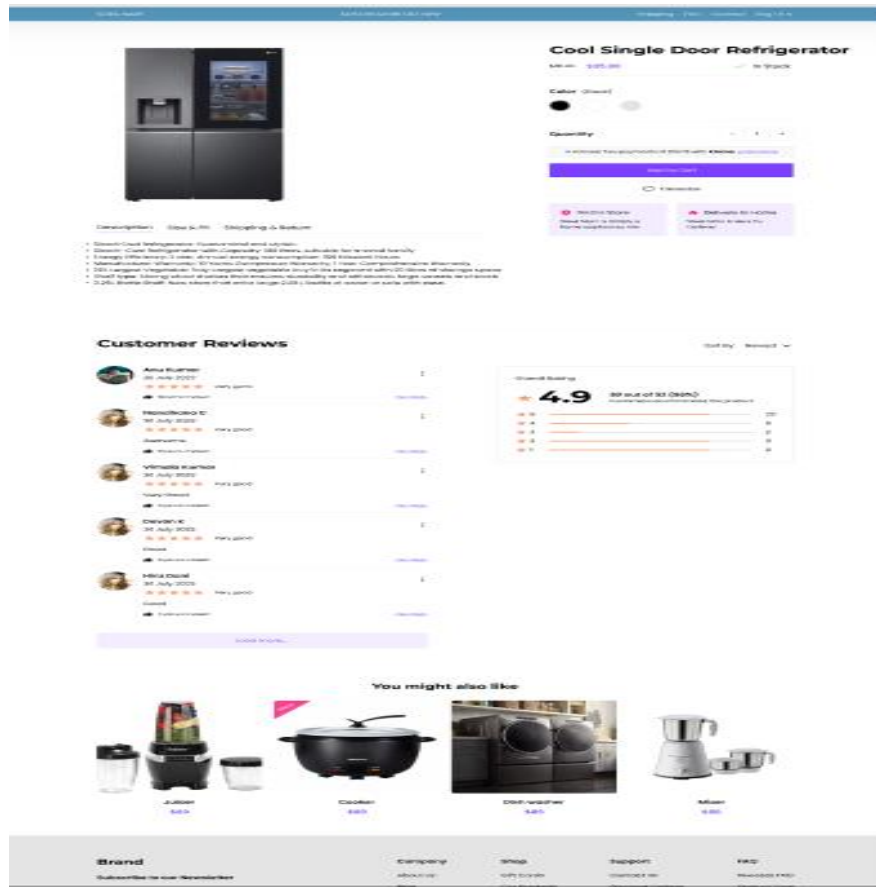


The SignUp Form UI design features a teal sidebar on the left with the 'SM' logo and the text 'Sign Up' and 'We do not share your personal details with anyone'. The main form area is white and contains input fields for 'FIRST NAME', 'LAST NAME', 'USERNAME', 'PASSWORD', and 'EMAIL ID'. Below these fields are two black buttons: 'Continue' and 'Continue with Google'. A horizontal line with the word 'OR' is positioned between the buttons. At the bottom, there is a link that says 'Existing User? Log In'.

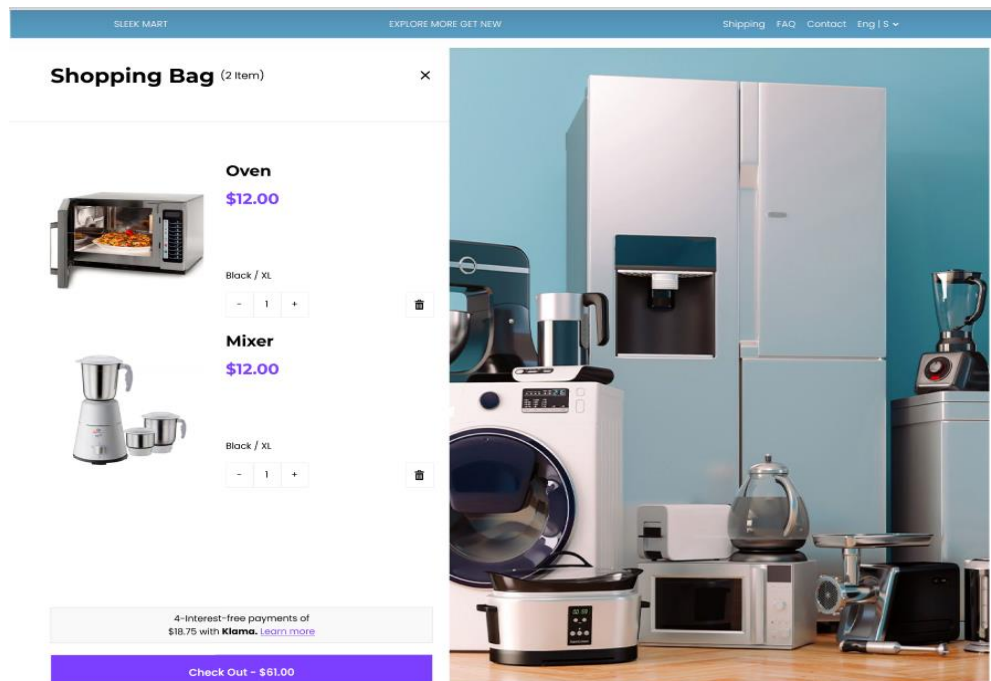
2. Form Name: Home Page



3. Form Name: Product Details Page



4. Form Name: Shopping Bag Page




5. Form Name: Purchase Page

SLEEK MART

X

You're 35.00 Away from free Shipping



Oven


\$12.00


Black / XL

-

1

+





Mixer


\$12.00


Black / XL

-

1

+





Refrigerator


\$12.00

Black / XL

-

1

+



Gift card or discount code

Enter gift card or discount code

Apply

Subtotal

\$49.40

Tax

\$0.00

Shipping

\$0.00

Total

\$62.40

6. Form Name: Category Page

SLEEK MART

Home Appliances

Shopping Cart (0)

Account

Help

New Arrivals

Filter

Department

Appliances

Countertop

Brands

Blender

Mixer

Oven

Refrigerator

Color

Black

White

Grey

Red

Blue

Green

Brand

Samsung

LG

Whirlpool

Haier

Hisense

Sharp

Electrolux

Category

Refrigerator

Oven

Mixer

Blender

Brand

Subscribe to our Newsletter

Enter email

Go

Company

About Us

Blog

Privacy Policy

Terms & Conditions

Contact Us

Shop

Home Appliances

Electronics

Books

Gifts

Support

FAQ

Help Center

Feedback

FAQ

Home Appliances

Electronics

Books

Gifts

Amal Jyothi College of Engineering, Kanjirappally

Department of Computer Applications

4.4 DATABASE DESIGN

A database is an organized collection of information that's organized to enable easy accessibility, administration, and overhauls. The security of information could be an essential objective of any database. The database design process comprises of two stages. In the first stage, user requirements are gathered to create a database that meets those requirements as clearly as possible. This is known as information-level design and is carried out independently of any DBMS. In the second stage, the design is converted from an information-level design to a specific DBMS design that will be used to construct the system. This stage is known as physical-level design, where the characteristics of the specific DBMS are considered. Alongside system design, there is also database design, which aims to achieve two main goals: data integrity and data independence.

4.4.1 Relational Database Management System (RDBMS)

A relational database management system (RDBMS) is a popular type of database that organizes data into tables to facilitate relationships with other stored data sets. Tables can contain vast amounts of data, ranging from hundreds to millions of rows, each of which are referred to as records. In formal relational model language, a row is called a tuple, a column heading is an attribute, and the table is a relation. A relational database consists of multiple tables, each with its own name. Each row in a table represents a set of related values.

In a relational database, relationships are already established between tables to ensure the integrity of both referential and entity relationships. A domain D is a group of atomic values, and a common way to define a domain is by choosing a data type from which the domain's data values are derived. It is helpful to give the domain a name to make it easier to understand the values it contains. Each value in a relation is atomic and cannot be further divided.

In a relational database, table relationships are established using keys, with primary key and foreign key being the two most important ones. Entity integrity and referential integrity relationships can be established with these keys. Entity integrity ensures that no primary key can have null values, while referential integrity ensures that each distinct foreign key value must have a matching primary key value in the same domain. Additionally, there are other types of keys such as super keys and candidate keys.

4.4.2 Normalization

The simplest possible grouping of data is used to put them together so that future changes can be made with little influence on the data structures. The formal process of normalizing data structures in a way that reduces duplication and fosters integrity. Using the normalization technique, superfluous fields are removed and a huge table is divided into several smaller ones. Anomalies in insertion, deletion, and updating are also prevented by using it. Keys and relationships are two notions used in the standard form of data modelling. A row in a table is uniquely identified by a key. Primary keys and foreign keys are two different kinds of keys. Primary key is an element, or set of components, in a table that serves as a means of distinguishing between records from the same table. A column in a table known as a foreign key is used to uniquely identify records from other tables. Up to the third normal form, all tables have been normalized.

Normalization is a process in database design that aims to organize data into proper tables and columns, making it easily correlated to the data by the user. This process eliminates data redundancy that can be a burden on computer resources. The main steps involved in normalization include:

- 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

The First Normal Form (1NF) requires that each attribute in a table must contain only atomic or indivisible values. It prohibits the use of nested relations or relations within relations as attribute values within tuples. To satisfy 1NF, data must be moved into separate tables where the data is of similar type in each table, and each table should have a primary key or foreign key as required by the project. This process eliminates repeating groups of data and creates new relations for each non-atomic attribute or nested relation. A relation is in 1NF only if it satisfies the constraints that contain the primary key only.

	id	business_name	business_address	business_website	seller_proof	is_approved	user_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	SleekMart	SleekMart Street	https://sleekmart.in	seller_proofs/...	approved	3
2	3	WMart	WMart, Street 12	https://wmart.org	seller_proofs/...	approved	7

Second Normal Form

Second normal form (2NF) is a rule in database normalization that states that non-key attributes should not be functionally dependent on only the part of the primary key in a relation that has a composite primary key. In other words, each non-key attribute should depend on the entire primary key, not just a part of it. To achieve this, we need to decompose the table and create new relationships for each subkey along with their dependent attributes. It is important to maintain the relationship with the original primary key and all attributes that are fully functionally dependent on it. A relation is said to be in 2NF only if it satisfies all the 1NF conditions for the primary key and every non-primary key attribute of the relation is fully dependent only on the primary key.

	id	shipname	shipaddress	shipcity	shippingpincode	user_id
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Ishta Rachel Mathew	Uthinilkunnathil(H)...	Mallapally	629592	8
2	2	Adrishya Maria Abraham	Vettuvellil(H)...	Kottayam	689293	5
3	3	Nandhana C Reghu	Karvelil(H)...	Manimala	689596	4

Third Normal Form

Third normal form (3NF) requires that a relation have no non-key attribute that is functionally determined by another non-key attribute or set of non-key attributes. This means that there should be no transitive dependency on the primary key. To achieve 3NF, we decompose the relation and set up a new relation that includes non-key attributes that functionally determine other non-key attributes. This helps eliminate any dependencies that don't just rely on the primary key. A relation is considered a relation in 3NF if it satisfies the conditions of 2NF and, moreover, the non-key attributes of the relation are not dependent on any other non-key attribute.

	id	slug	name	image	status	trending	description	created_date
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	refrigeration-appliances	Refrigeration Appliances	category_images/...	0	0	Refrigeration appliances enable us to...	2023-09-16 18:05:22.430266
2	2	cooking-appliances	Cooking Appliances	category_images/cooking-...	0	0	Cooking appliances always top the ...	2023-09-16 18:05:55.814946
3	3	washing-and-drying-appliances	Washing And Drying Appliances	category_images/...	0	0	Washing Appliances	2023-09-16 18:07:03.823236

id	name	description	image	created_date	updated_date	status	trending	category_id
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1 Refrigerators	There are hardly any households that...	subcategory_images/...	2023-09-16 18:10:17.047314	2023-09-18 04:49:15.732068	0	0	1
2	2 Freezers	Freezers are essential for preserving ...	subcategory_images/...	2023-09-16 18:10:47.975057	2023-09-16 18:10:47.975057	0	0	1
3	3 Water Coolers	Having a water cooler is more ...	subcategory_images/watercooler2.jpg	2023-09-16 18:11:19.461983	2023-09-18 04:49:50.126883	0	0	1
4	4 Kitchen Stoves	Even with the advent of many new-a...	subcategory_images/kitchen-...	2023-09-16 18:12:31.509081	2023-09-16 18:12:31.509081	0	0	2
5	5 Rice Cookers	The easiest way to cook rice is by ...	subcategory_images/rice-...	2023-09-16 18:12:57.892134	2023-09-16 18:12:57.892134	0	0	2
6	6 Roti Makers	After the rice cooker, kitchen ...	subcategory_images/rotimaker.png	2023-09-16 18:13:44.746272	2023-09-16 18:13:44.746272	0	0	2
7	7 Washing Machines	Washing machine – an appliance that...	subcategory_images/...	2023-09-16 18:14:31.424705	2023-09-26 09:00:12.962350	0	0	3
8	8 Clothes Dryer	Dryers can effectively dry your clothe...	subcategory_images/Cloth-Dryer.jpg	2023-09-16 18:15:06.670338	2023-09-26 09:00:09.179890	0	0	3
9	9 Dishwashers	Homemakers vouch for this home ...	subcategory_images/dishwashers.jpg	2023-09-18 19:52:02.309771	2023-09-26 09:00:02.470451	0	0	3

4.4.3 Sanitization

Data sanitization is the process of removing any illegal characters or values from data. In web applications, sanitizing user input is a common task to prevent security vulnerabilities. Python provides a built-in filter extension that can be used to sanitize and validate various types of external input such as email addresses, URLs, IP addresses, and more. These filters are designed to make data sanitization easier and faster. For example, the Python filter extension has a function that can remove all characters except letters, digits, and certain special characters (!#\$%&'*+-=?_`{|}~@.[]), as specified by a flag.

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_Users

Primary key: **userid**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	userid	int (10)	PRIMARY KEY	Login Id
2	email	varchar (20)	NOT NULL, UNIQUE	Username of actors
3	password	varchar (20)	NOT NULL	Password
4	name	varchar (20)	NOT NULL	Name of actors
5	mobile	varchar (20)	NOT NULL	Contact number of actors
6	profile_pic	varchar (20)	NOT NULL	Profile picture
7	role	varchar (20)	NOT NULL	Role of actors
8	status	int (10)	NOT NULL	Status-active/blocked

2. Tbl_Sellers

Primary key: **sellerid**

Foreign key: **userid** references table **Tbl_Users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	sellerid	int (10)	PRIMARY KEY	Id of Seller
2	business_name	varchar (20)	NOT NULL, UNIQUE	Business name of seller
3	business_address	varchar (20)	NOT NULL	Business address of seller
4	business_website	varchar (20)	NOT NULL	Business website
5	seller_proof	varchar (20)	NOT NULL	Certificate proof of seller
6	is_approved	varchar (20)	NOT NULL	Approved/rejected
8	userid	int (10)	FOREIGN KEY	Login Id

3. Tbl_CategoryPrimary key: **categoryid**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	categoryid	int (10)	PRIMARY KEY	Id of category
2	name	varchar (20)	NOT NULL	Name of category
3	image	varchar (20)	NOT NULL	Image of category
4	status	int (10)	NOT NULL	Status – active/inactive
5	trending	int (10)	NOT NULL	Trending
6	description	varchar (20)	NOT NULL	Description of category
8	created_date	date	NOT NULL	Category creation date
9	updated_date	date	NOT NULL	Category updation date

4. Tbl_SubcategoryPrimary key: **subcategoryid**Foreign key: **categoryid** references table **Tbl_Category**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	subcategoryid	int (10)	PRIMARY KEY	Id of Subcategory
2	name	char (20)	NOT NULL	Name of Subcategory
3	image	char (20)	NOT NULL	Image of Subcategory
4	status	int (10)	NOT NULL	Status – active/inactive
5	trending	int (10)	NOT NULL	Trending
6	description	char (20)	NOT NULL	Description of Subcategory
8	created_date	date	NOT NULL	SubCategory creation date
9	updated_date	date	NOT NULL	SubCategory updation date
10	categoryid	int (10)	FOREIGN KEY	Id of Category

4. Tbl_Product

Primary key: **productid**

Foreign key: **sellerid** references table **Tbl_Sellers**

subcategyid references table **Tbl_Subcategory**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	productid	int (10)	PRIMARY KEY	Id of Product
2	name	char (20)	NOT NULL	Name of Product
3	product_image	char (20)	NOT NULL	Image of Product
4	status	int (10)	NOT NULL	Status – active/inactive
5	trending	int (10)	NOT NULL	Trending
6	description	char (20)	NOT NULL	Description of Product
7	quantity	int (10)	NOT NULL	Product Quantity
8	originalprice	int (10)	NOT NULL	Original price of product
9	sellingprice	int (10)	NOT NULL	Selling price of product
10	created_date	date	NOT NULL	SubCategory creation date
11	updated_date	date	NOT NULL	SubCategory updation date
12	subcategoryid	int (10)	FOREIGN KEY	Id of Subcategory
13	sellerid	int (10)	FOREIGN KEY	Id of Seller

4. Tbl_AddWishlist

Primary key: **wishlistid**

Foreign key: **userid** references table **Tbl_Users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	wishlistid	int (10)	PRIMARY KEY	Id of Wishlist
2	wishlist_date	date	NOT NULL	Wishlist Created Date
3	user_id	int (10)	FOREIGN KEY	User Id

5. Tbl_WishlistItems

Primary key: **wishlistitemsid**

Foreign key: **productid** references table **Tbl_Product**

wishlistid references table **Tbl_AddWishlist**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	wishlistitemsid	int (10)	PRIMARY KEY	Id of WishlistItem
2	productid	int (10)	FOREIGN KEY	Id of Product
3	wishlistid	int (10)	FOREIGN KEY	Id of Wishlist

6. Tbl_Order

Primary key: **orderid**

Foreign key: **userid** references table **Tbl_Users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	orderid	int (10)	PRIMARY KEY	Id of Order
2	totalprice	char (20)	NOT NULL	Total Price For Order
3	orderdate	date	NOT NULL	Order Created Date
4	razorpay_order_id	Int (20)	NOT NULL	Payment Id
5	payment_status	char (20)	NOT NULL	Payment status – confirmed/cancelled/pending
6	user_id	Int (10)	FOREIGN KEY	User Id

7. Tbl_OrderItem

Primary key: **orderitemid**

Foreign key: **orderid** references table **Tbl_Order**, **productid** references table **Tbl_Product**, **sellerid** references table **Tbl_Sellers**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	orderitemid	int (10)	PRIMARY KEY	Id of OrderItem
2	totalprice	int (20)	NOT NULL	Total Price For OrderItems
3	price	int (20)	NOT NULL	Price Of OrderItems

4	quantity	int (20)	NOT NULL	Quantity of ordereditems
5	orderid	int (20)	FOREIGN KEY	Order Id
6	productid	int (20)	FOREIGN KEY	Product Id
7	sellerid	int (20)	FOREIGN KEY	Seller Id

8. Tbl_AddCart

Primary key: **addcartid**

Foreign key: **userid** references table **Tbl_Users**, **orderid** references table **Tbl_Order**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	addcartid	int (10)	PRIMARY KEY	Id of Addcart
2	cart_date	date	NOT NULL	Cart creation date
3	userid	int (20)	FOREIGN KEY	Price Of OrderItems
4	orderid	int (20)	FOREIGN KEY	Quantity of ordereditems

9. Tbl_CartItems

Primary key: **cartitemsid**

Foreign key: **addcartid** references table **Tbl_AddCart**, **productid** references table **Tbl_Product**, **orderid** references table **Tbl_Order**, **orderitemid** references table **Tbl_OrderItem**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	cartitemsid	int (10)	PRIMARY KEY	Id of CartItems
2	cartitem_date	date	NOT NULL	CartItems creation date
3	quantity	int (20)	NOT NULL	Quantity of cartitems
4	totalprice	int (20)	NOT NULL	Total price of cartitems
5	addcartid	int (20)	FOREIGN KEY	Cart id
6	productid	int (20)	FOREIGN KEY	Product id
7	orderid	int (20)	FOREIGN KEY	Order Id
8	orderitemid	int (20)	FOREIGN KEY	OrderItem id

10. Tbl_Review

Primary key: **reviewid**

Foreign key: **productid** references table **Tbl_Product**, **userid** references table **Tbl_Users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	reviewid	int (10)	PRIMARY KEY	Id of Review
2	review_date	date	NOT NULL	Review creation date
3	productid	int (20)	FOREIGN KEY	Product id
4	userid	int (20)	FOREIGN KEY	User Id

11. Tbl_ReviewRating

Primary key: **reviewratingid**

Foreign key: **reviewid** references table **Tbl_Review**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	reviewratingid	int (10)	PRIMARY KEY	Id of Reviewrating
2	rating	int (5)	NOT NULL	Rating for product
3	comment	char (20)	NOT NULL	Comments for product
8	reviewid	int (20)	FOREIGN KEY	Review id

12. Tbl_Shippings

Primary key: **shippingid**

Foreign key: **userid** references table **Tbl_Users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	shippingid	int (10)	PRIMARY KEY	Id of Shipping
2	shipname	char (25)	NOT NULL	Name of shipping user
3	shipaddress	char (20)	NOT NULL	Shipping Address
4	shipcity	char (20)	NOT NULL	Shipping City
5	shippingpincode	int (10)	NOT NULL	Shipping Place Pincode
6	userid	Int(10)	FOREIGN KEY	User Id

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing involves executing a software program in a controlled manner to determine if it behaves as intended, often using verification and validation methods. Validation involves evaluating a product to ensure it complies with specifications, while verification can involve reviews, analyses, inspections, and walkthroughs. Static analysis examines the software's source code to identify issues, while dynamic analysis examines its behavior during runtime to gather information like execution traces, timing profiles, and test coverage details.

Testing involves a series of planned and systematic activities that start with individual modules and progress to the integration of the entire computer-based system. The objectives of testing include identifying errors and bugs in the software, ensuring that the software functions according to its specifications, and verifying that it meets performance requirements. Testing can be performed to assess correctness, implementation efficiency, and computational complexity.

A successful test is one that detects an undiscovered error, and a good test case has a high probability of uncovering such errors. Testing is crucial to achieving system testing objectives and can involve various techniques such as functional testing, performance testing, and security testing.

5.2 TEST PLAN

A test plan is a document that outlines the required steps to complete various testing methodologies. It provides guidance on the activities that need to be performed during testing. Software developers create computer programs, documentation, and associated data structures. They are responsible for testing each component of the program to ensure it meets the intended purpose. To address issues with self-evaluation, an independent test group (ITG) is often established.

Testing objectives should be stated in quantifiable language, such as mean time to failure, cost to find and fix defects, remaining defect density or frequency of occurrence, and test work-hours per regression test.

The different levels of testing include:

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

5.2.1 Unit Testing

Unit testing is a software testing technique that focuses on verifying individual components or modules of the software design. The purpose of unit testing is to test the smallest unit of software design and ensure that it performs as intended. Unit testing is typically white-box focused, and multiple components can be tested simultaneously. The component-level design description is used as a guide during testing to identify critical control paths and potential faults within the module's perimeter.

During unit testing, the modular interface is tested to ensure that data enters and exits the software unit under test properly. The local data structure is inspected to ensure that data temporarily stored retains its integrity during each step of an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once, and all error handling paths are tested to ensure that the software can handle errors correctly.

Before any other testing can take place, it is essential to test data flow over a module interface. If data cannot enter and exit the system properly, all other tests are irrelevant. Another crucial duty during unit testing is the selective examination of execution pathways to anticipate potential errors and ensure that error handling paths are set up to reroute or halt work when an error occurs. Finally, boundary testing is conducted to ensure that the software operates correctly at its limits.

In this system, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. Unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome.

5.2.2 Integration Testing

Integration testing is a systematic approach that involves creating the program structure while simultaneously conducting tests to identify interface issues. The objective is to construct a program structure based on the design that uses unit-tested components. The entire program is then tested. Correcting errors in integration testing can be challenging due to the size of the overall program, which makes it difficult to isolate the causes of the errors. As soon as one set of errors is fixed, new ones may arise, and the process may continue in an apparently endless cycle.

Once unit testing is complete for all modules in the system, they are integrated to check for any interface inconsistencies. Any discrepancies in program structures are resolved, and a unique program structure is developed.

5.2.3 Validation Testing or System Testing

The final stage of the testing process involves testing the entire software system as a whole, including all forms, code, modules, and class modules. This is commonly referred to as system testing or black box testing. The focus of black box testing is on testing the functional requirements of the software. A software engineer can use this approach to create input conditions that will fully test each program requirement. The main types of errors targeted by black box testing include incorrect or missing functions, interface errors, errors in data structure or external data access, performance errors, initialization errors, and termination errors.

5.2.4 Output Testing or User Acceptance Testing

User acceptance testing is performed to ensure that the system meets the business requirements and user needs. It is important to involve the end users during the development process to ensure that the software aligns with their needs and expectations. During user acceptance testing, the input and output screen designs are tested with different types of test data. The preparation of test data is critical to ensure comprehensive testing of the system. Any errors identified during testing are addressed and corrected, and the corrections are noted for future reference.

5.2.5 Automation Testing

Automation testing is a software testing approach that employs specialized automated testing software tools to execute a suite of test cases. Its primary purpose is to verify that the software or equipment operates precisely as intended. Automation testing identifies defects, bugs, and other issues that may arise during product development.

While some types of testing, such as functional or regression testing, can be performed manually, there are numerous benefits to automating the process. Automation testing can be executed at any time of day and uses scripted sequences to evaluate the software. The results are reported, and this information can be compared to previous test runs. Automation developers typically write code in programming languages such as C#, JavaScript, and Ruby.

5.2.6 Selenium Testing

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors.

In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It allows for the creation of executable specifications in a human-readable format called Gherkin. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals.

Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

Test Case 1: User Login

Code

```
package CucumberJava;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class Cucumberclass {

    WebDriver driver = null;

    @Given("browser is open")
    public void browser_is_open() {
        System.out.println("Inside Step - Browser Is Open");
        System.setProperty("webdriver.gecko.marionette",
"C:\\\\Users\\\\ISHTA RACHEL MATHEW\\\\eclipse-
workspace\\\\new_artifact\\\\src\\\\test\\\\resources\\\\Drivers\\\\geckodriver.exe")
;
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("user is on login page")
    public void user_is_on_login_page() throws Exception{
        driver.navigate().to("http://127.0.0.1:8000/login_page/");
        Thread.sleep(3000);
    }

    @When("user enters username and password")
    public void user_enters_username_and_password() throws Throwable {

driver.findElement(By.name("email")).sendKeys("ishtarachelmthw2000@gmail.com");
        driver.findElement(By.name("password")).sendKeys("IshtaRMM@123");
        Thread.sleep(3000);
    }

    @And("User click on login")
    public void user_click_on_login() {
        driver.findElement(By.className("btn-primary")).click();
    }

    @Then("user is navigated to the home page")
    public void user_is_navigated_to_the_home_page() throws Exception
    {
        driver.findElement(By.id("top")).isDisplayed();

        Thread.sleep(2000);

        driver.close();

        driver.quit();
    }
}
```

Screenshot

```
Oct 24, 2023 7:27:53 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check login is successfull with valid credential # src/test/resources/feature/Cucumberfile.feature:3
Inside Step - Browser Is Open
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1698155874313 geckodriver INFO Listening on 127.0.0.1:6921
1698155874560 mozrunner:runner INFO Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "--remote-d
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698155874897 Marionette INFO Marionette enabled
Dynamically enable window occlusion 0
1698155874966 Marionette INFO Listening on port 50600
WebDriver BiDi listening on ws://127.0.0.1:26723
Read port: 50600
1698155875174 RemoteAgent WARN TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:26723/devtools/browser/95d337f8-90c9-400e-b1b7-75e9247fd83f
Given browser is open # CucumberJava.Cucumberclass.browser_is_open()
And user is on login page # CucumberJava.Cucumberclass.user_is_on_login_page()
When user enters username and password # CucumberJava.Cucumberclass.user_enters_username_and_password()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/login_page/, line 1: ReferenceError: validateLoginForm is not defined
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
And User click on login # CucumberJava.Cucumberclass.user_click_on_login()
1698155887058 RemoteAgent INFO Perform WebSocket upgrade for incoming connection from 127.0.0.1:50655
1698155887062 CDP WARN Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fission.bfcacheInPare
```

Test Report

Test Case 1					
Project Name: SLEEK MART					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Ishta Rachel Mathew		
Test Priority(Low/Medium/High):High			Test Designed Date: 15/09/2023		
Module Name: Login Screen			Test Executed By : Ms. Anit James		
Test Title : User Login			Test Execution Date: 15/09/2023		
Description: Verify Login with valid email 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 Login button		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid Email	Email: ishtarachelmathew2000@gmail.com	User should be able to Login	User Logged in and navigated to User Dashboard	Pass
3	Provide Valid Password	Password: Ishta@123			
4	Click on Login button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2: Admin – Add Category**Code**

```
package Categoryjava;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class Categoryfile {
    WebDriver driver = null;

    @Given("browser is open")
    public void browser_is_open() {
        System.out.println("Inside Step - Browser Is Open");
        System.setProperty("webdriver.gecko.marionette", "C:\\Users\\ISHTA RACHEL
MATHEW\\eclipse-
workspace\\new_artifact\\src\\test\\resources\\Drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("admin is on the login page")
    public void admin_is_on_the_login_page() throws Exception{
        driver.navigate().to("http://127.0.0.1:8000/login_page/");
        Thread.sleep(3000);
    }

    @When("admin enters admin credentials and logs in")
```

```
public void admin_enters_admin_credentials_and_logs_in()throws Throwable {  
    driver.findElement(By.name("email")).sendKeys("admin@gmail.com");  
    driver.findElement(By.name("password")).sendKeys("123");  
    driver.findElement(By.className("btn-primary")).click();  
    Thread.sleep(3000);  
}
```

```
@And("admin navigates to the admin page")  
public void admin_navigates_to_the_admin_page()throws Exception  
{
```

```
    Thread.sleep(3000);  
}
```

```
@And("admin clicks on the \"Manage\" button")  
public void admin_clicks_on_the_manage_button()throws Exception {
```

```
    // Click on the "Manage" button  
    driver.findElement(By.xpath("//a[contains(@href, 'productlist')]")).click();  
    Thread.sleep(3000);  
}
```

```
@And("admin selects \"Category\" from the dropdown")  
public void admin_selects_category_from_the_dropdown() throws Exception  
{
```

```
    driver.findElement(By.xpath("//a[contains(@href, 'categories')]")).click();  
    Thread.sleep(3000);  
}
```

```
@And("user navigates to the category page")  
public void user_navigates_to_the_category_page() throws Exception  
{
```

```
    Thread.sleep(3000);  
}
```

```
@And("user enters category details")  
public void user_enters_category_details() throws Exception  
{
```

```

        // Enter category details
        driver.findElement(By.name("name")).sendKeys("Washing    And    Drying
Appliances");
        driver.findElement(By.name("slug")).sendKeys("washing-and-drying-
appliances");
        driver.findElement(By.name("catdescription")).sendKeys("Washing
Appliances");
        // Upload an image (if needed)
        // Assuming you have the file path to an image to upload
        String                                imagePath                                =
"D:\\DemoTesting\\MainProject\\SleekMart\\static\\images\\washing$dryingappliance
s.jpg";
        driver.findElement(By.name("image")).sendKeys(imagePath);
        Thread.sleep(3000);
        // Add any other details related to category creation
    }
    @And("user clicks on the \"Add category\" button")
    public void user_clicks_on_the_add_category_button() {
        // Click on the "Add category" button
        driver.findElement(By.cssSelector("button.btn.btn-primary")).click();
    }
    @Then("category should be added and displayed on the category page")
    public void category_should_be_added_and_displayed_on_the_category_page()throws Exception
    {
        driver.findElement(By.id("top")).isDisplayed();

        Thread.sleep(2000);
        driver.close();
        driver.quit();
    }
}

```

Screenshot

```
Oct 24, 2023 7:45:08 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Adding a category as an admin # src/test/resources/Categoryfeature/Categoryfile.feature:3
Inside Step - Browser Is Open
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1698156909767 geckodriver INFO Listening on 127.0.0.1:36509
1698156910014 mozrunner::runner INFO Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "--remote-d
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698156910288 Marionette INFO Marionette enabled
Dynamically enable window occlusion 0
1698156910340 Marionette INFO Listening on port 50722
WebDriver BiDi listening on ws://127.0.0.1:24412
Read port: 50722
1698156910440 RemoteAgent WARN TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:24412/devtools/browser/8f87a2a8-723f-4a38-b94a-02f4e400cc58
Given browser is open # Categoryjava.Categoryfile.browser_is_open()
And admin is on the login page # Categoryjava.Categoryfile.admin_is_on_the_login_page()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/login_page/, line 1: ReferenceError: validateLoginForm is not defined
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
```

Test Report

Test Case 2					
Project Name: SLEEK MART					
Add Category Test Case					
Test Case ID: Test_2			Test Designed By: Ishta Rachel Mathew		
Test Priority(Low/Medium/High):High			Test Designed Date: 15/09/2023		
Module Name: Add Category			Test Executed By : Ms. Anit James		
Test Title : Admin Add Category			Test Execution Date: 15/09/2023		
Description: Create new category and its description by Admin					
Pre-Condition : Admin 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: admin@gmail.com	Admin should be able to Login	Admin Logged in and navigated to Dashboard with Records	Pass
3	Provide Valid Password	Password: Admin@123			

4	Click on Login button				
5	Click on add new category		Add category page should be displayed	Admin navigated to add category page	Pass
6	Provide new category name	Category name: Washing and drying appliances			
7	Provide description	Description: Best in market			
8	Click on Create Category Button		Admin should be able to add new category	Admin created new category	Pass
9	Navigation to homepage		Admin should be able to view category list	Admin navigated to home page and view categories added	Pass
Post-Condition: New category is successfully created by admin					

Test Case 3: Admin – Product Display

Code

```
package productjava;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class Productfile {

    WebDriver driver = null;
```



```
@Given("browser is open")
public void browser_is_open() {
    System.out.println("Inside Step - Browser Is Open");
    System.setProperty("webdriver.gecko.marionette", "C:\\Users\\ISHTA RACHEL
MATHEW\\eclipse-
workspace\\new_artifact\\src\\test\\resources\\Drivers\\geckodriver.exe");
    driver = new FirefoxDriver();
    driver.manage().window().maximize();
}

@And("seller is on the login page")
public void seller_is_on_the_login_page() throws Exception{
    driver.navigate().to("http://127.0.0.1:8000/login_page/");
    Thread.sleep(3000);
}

@When("seller enters seller credentials and logs in")
public void seller_enters_seller_credentials_and_logs_in()throws Throwable {

    driver.findElement(By.name("email")).sendKeys("ishtarachelmathew2024@m
ca.ajce.in");
    driver.findElement(By.name("password")).sendKeys("Ishta@123");
    driver.findElement(By.className("btn-primary")).click();
    Thread.sleep(3000);
}

@And("seller navigates to the seller page")
public void seller_navigates_to_the_seller_page()throws Exception
{

    Thread.sleep(3000);
}

@And("seller clicks on the \"Manage\" button")
public void seller_clicks_on_the_manage_button()throws Exception {
```

```

        // Click on the "Manage" button
        driver.findElement(By.xpath("//a[contains(@href, 'productlist')]")).click();
        Thread.sleep(3000);
    }

    @And("seller selects \"Product\" from the dropdown")
    public void seller_selects_product_from_the_dropdown() throws Exception
    {
        driver.findElement(By.xpath("//a[text()='Product List']")).click();
        Thread.sleep(3000);
    }

    @And("user clicks on the \"Create New\" button")
    public void user_clicks_on_the_create_new_button()throws Exception {

        driver.findElement(By.cssSelector("a[href*='addproduct']+[class*='btn-
primary']")).click();
        Thread.sleep(3000);
    }

    @And("user navigates to the product page")
    public void user_navigates_to_the_product_page() throws Exception
    {
        Thread.sleep(3000);
    }

    @And("user enters product details")
    public void user_enters_product_details() throws Exception
    {
        Select categoryDropdown = new
Select(driver.findElement(By.id("product_category"))));
        categoryDropdown.selectByVisibleText("Refrigeration Appliances"); // Replace
with the actual category name

```

```

        Select subcategoryDropdown = new
Select(driver.findElement(By.id("product_subcategory")));
        subcategoryDropdown.selectByVisibleText("Refrigerators");

        driver.findElement(By.name("name")).sendKeys("Washing And Drying
Appliances");
        driver.findElement(By.name("description")).sendKeys("Washing");
        driver.findElement(By.name("original_price")).sendKeys("610");
        // Upload an image (if needed)
        // Assuming you have the file path to an image to upload
        String imagePath =
"D:\\DemoTesting\\MainProject\\SleekMart\\static\\images\\washing$dryingappliance
s.jpg";
        driver.findElement(By.name("image")).sendKeys(imagePath);
        Thread.sleep(3000);
        // Add any other details related to category creation
    }
    @And("user clicks on the \"Add category\" button")
    public void user_clicks_on_the_add_category_button() {
        // Click on the "Add category" button
        driver.findElement(By.cssSelector("button.btn.btn-primary")).click();
    }
    @Then("category should be added and displayed on the category page")
    public void category_should_be_added_and_displayed_on_the_category_page()throws Exception
    {
        driver.findElement(By.id("top")).isDisplayed();
        Thread.sleep(2000);
        driver.close();
        driver.quit();
    }
}

```

Screenshot

```

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1698157209045   geckodriver      INFO    Listening on 127.0.0.1:23999
1698157209289   mozrunner::runner  INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "--re
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698157209565   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698157209618   Marionette      INFO    Listening on port 50798
WebDriver BiDi listening on ws://127.0.0.1:47622
Read port: 50798
1698157209658   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:47622/devtools/browser/72be9044-f805-4e56-ad67-e50f0dfdba28
  Given browser is open                                # productjava.Productfile.browser_is_open()
  And seller is on the login page                       # productjava.Productfile.seller_is_on_the_login_page()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/login_page/, line 1: ReferenceError: validateLoginForm is not defined
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
  When seller enters seller credentials and logs in     # productjava.Productfile.seller_enters_seller_credentials_and_logs_in
  And seller navigates to the seller page               # productjava.Productfile.seller_navigates_to_the_seller_page()
  And seller clicks on the "Manage" button              # productjava.Productfile.seller_clicks_on_the_manage_button()
  And seller selects "Product" from the dropdown        # productjava.Productfile.seller_selects_product_from_the_dropdown()
  And user navigates to the product page                # productjava.Productfile.user_navigates_to_the_product_page()
  And user clicks on the "Create New" button            # productjava.Productfile.user_clicks_on_the_create_new_button()
  And user enters product details                      # productjava.Productfile.user_enters_product_details()

```

Test Report

Test Case 3					
Project Name: SLEEK MART					
Add Product Test Case					
Test Case ID: Test_3			Test Designed By: Ishta Rachel Mathew		
Test Priority(Low/Medium/High):High			Test Designed Date: 15/09/2023		
Module Name: Add Product			Test Executed By : Ms. Anit James		
Test Title : Seller Add Product			Test Execution Date: 15/09/2023		
Description: Adding a new product through the seller interface					
Pre-Condition : Seller has a 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: mariajacob@gmail.com	Seller should be able to Login	Seller Logged in and navigated to Dashboard with Records	Pass
3	Provide Valid Password	Password: Maria@123			
4	Click on Login button				

5	Click on add new product		Add product page should be displayed	Seller navigated to add product page	Pass
6	Provide new product name	Product name: Refrigeration Appliances			
7	Provide description	Description: Best in market			
8	Click on Create Product Button		Seller should be able to add new product	Seller created new product	Pass
9	Navigation to homepage		Seller should be able to view product list	Seller navigated to home page and view product added	Pass
Post-Condition: New product is successfully created by seller					

Test Case 4: Add to Cart

Code

```

package Cartjava;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
public class Cart {
    WebDriver driver = null;
    @Given("browser is open")
    public void browser_is_open() {
        System.out.println("Inside Step - Browser Is Open");
        System.setProperty("webdriver.gecko.marionette", "C:\\Users\\ISHTA RACHEL
MATHEW\\eclipse-
workspace\\new_artifact\\src\\test\\resources\\Drivers\\geckodriver.exe");

```

```
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("customer is on the login page")
    public void customer_is_on_the_login_page() throws Exception {
        driver.navigate().to("http://127.0.0.1:8000/login_page/");
        Thread.sleep(3000);
    }

    @When("customer enters customer credentials and logs in")
    public void customer_enters_customer_credentials_and_logs_in() throws Throwable {

        driver.findElement(By.name("email")).sendKeys("ishtarachelmathew2000@gmail.
com");
        driver.findElement(By.name("password")).sendKeys("IshtaRMM@123");
        driver.findElement(By.className("btn-primary")).click();
        Thread.sleep(3000);
    }

    @And("customer navigates to the customer page")
    public void customer_navigates_to_the_customer_page() throws Exception {
        Thread.sleep(3000);
    }

    @And("customer clicks on the \"Category\" button")
    public void customer_clicks_on_the_category_button() throws Exception {
        driver.findElement(By.xpath("//a[contains(text(), 'Shop By Categories')]")).click();
        Thread.sleep(3000);
    }

    @And("customer selects \"Refrigeration Appliances\" from the dropdown")
    public void customer_selects_refrigeration_appliances_from_the_dropdown() throws
Exception {
```

```
        driver.findElement(By.xpath("//a[contains(text(), 'Refrigeration
Appliances')]")).click();
        Thread.sleep(3000);
    }

    @And("user navigates to the category page")
    public void user_navigates_to_the_category_page() throws Exception {
        Thread.sleep(3000);
    }

    @When("the customer selects a product from the category page")
    public void the_customer_selects_a_product_from_the_category_page() throws
Exception {
        // Implement code to select a product
        driver.findElement(By.cssSelector(".thumb a")).click();
        String productAlt = "Frestec 3.1 CU' Mini Fridge";
        driver.findElement(By.cssSelector("img[alt='" + productAlt + "']")).click();
    }

    @And("the customer clicks on the \"Add to Cart\" button")
    public void the_customer_clicks_on_the_add_to_cart_button() {
        // Click on the "Add to Cart" button
        driver.findElement(By.cssSelector(".add-to-cart-button")).click();
    }

    @Then("the product should be added to the cart")
    public void the_product_should_be_added_to_the_cart() throws Exception {
        // Implement code to check if the product is added to the cart
        // You can use assertions here to verify the product is in the cart
        Thread.sleep(3000);
        driver.close();
        driver.quit();
    }
}
```

Screenshot

```

Oct 30, 2023 11:45:35 AM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario:                                     # src/test/resources/cartfeature/cart.feature:3
Inside Step - Browser Is Open
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1698646537251  geckodriver      INFO    Listening on 127.0.0.1:48053
1698646537489  mozrunner::runner      INFO    Running command: "C:\Program Files\Mozilla Firefox\firefox.exe" "--marionette" "--remote-d
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698646537761  Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698646537813  Marionette      INFO    Listening on port 51459
WebDriver BiDi listening on ws://127.0.0.1:46221
Read port: 51459
1698646537932  RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:46221/devtools/browser/12f52d89-029e-416f-a6ff-e13e20a86ad7
  Given browser is open                      # Cartjava.Cart.browser_is_open()
  And customer is on the login page          # Cartjava.Cart.customer_is_on_the_login_page()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/login_page/, line 1: ReferenceError: validateLoginForm is not defined
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."

```

Test Report

Test Case 4					
Project Name: SLEEK MART					
Add to Cart Test Case					
Test Case ID: Test_4			Test Designed By: Ishta Rachel Mathew		
Test Priority(Low/Medium/High):High			Test Designed Date: 15/09/2023		
Module Name: Add to Cart			Test Executed By : Ms. Anit James		
Test Title : Add to Cart			Test Execution Date: 15/09/2023		
Description: Adding product to cart					
Pre-Condition : User has a 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: mariajacob@gmail.com	User should be able to Login	User Logged in and navigated to Dashboard with Records	Pass
3	Provide Valid Password	Password: Maria@123			

4	Click on Login button				
5	Click on product		Product page should be displayed	User navigated to product page	Pass
6	Click on add to cart button		Product should be added to cart page		
7	Navigation to Cart page		User should be able to view product list in cart	User navigated to cart page and view product added	Pass
Post-Condition: Product is successfully added to cart by user					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The implementation phase of a project is where the design is transformed into a functional system. It is a crucial stage in ensuring the success of the new system, as it requires gaining user confidence that the system will work effectively and accurately. User training and documentation are key concerns during this phase. Conversion may occur concurrently with user training or at a later stage. Implementation involves the conversion of a newly revised system design into an operational system.

During this stage, the user department bears the primary workload, experiences the most significant upheaval, and has the most substantial impact on the existing system. Poorly planned or controlled implementation can cause confusion and chaos. Whether the new system is entirely new, replaces an existing manual or automated system, or modifies an existing system, proper implementation is essential to meet the organization's needs. System implementation involves all activities required to convert from the old to the new system. The system can only be implemented after thorough testing is done and found to be working according to specifications. The implementation phase involves careful planning, investigating system and constraints, and designing methods to achieve changeover.

6.2 IMPLEMENTATION PROCEDURES

Software implementation is the process of installing the software in its actual environment and ensuring that it satisfies the intended use and operates as expected. In some organizations, the software development project may be commissioned by someone who will not be using the software themselves. During the initial stages, there may be doubts about the software, but it's important to ensure that resistance does not build up. This can be achieved by:

- Ensuring that active users are aware of the benefits of the new system, building their confidence in the software.
- Providing proper guidance to the users so that they are comfortable using the application. Before viewing the system, users should know that the server program must be running on the server. Without the server object up and running, the intended process will not take place.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness, it is essential to provide training on the new application software to the user. This training should include the underlying philosophy of using the new system, such as the flow of screens, screen design, the type of help available on the screen, the types of errors that may occur while entering data, and the corresponding validation checks for each entry, and ways to correct the data entered. Additionally, the training should cover information specific to the user or group, which is necessary to use the system or part of the system effectively. It is important to note that this training may differ across different user groups and levels of hierarchy.

6.2.3 System Maintenance

The maintenance phase is a crucial aspect of the software development cycle, as it is the time when the software is actually put to use and performs its intended functions. Proper maintenance is essential to ensure that the system remains functional, reliable, and adaptable to changes in the system environment. Maintenance activities go beyond simply identifying and fixing errors or bugs in the system. It may involve updates to the software, modifications to its functionalities, and enhancements to its performance, among other things. In essence, software maintenance is an ongoing process that requires continuous monitoring, evaluation, and improvement of the system to meet changing user needs and requirements.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In summary, the development and launch of our home appliances e-commerce website mark a significant milestone in our pursuit of delivering a convenient, secure, and efficient platform for customers to shop for their home appliance needs. This project's core objectives were to create a user-friendly interface, provide a diverse product catalog, ensure secure payment processing, offer personalized shopping experiences, streamline order fulfillment, and establish robust customer support. These objectives have been successfully met, and our website is poised to provide a seamless shopping experience for our customers.

One of our primary accomplishments lies in the creation of a user-friendly interface that prioritizes the customer's experience. The website's design boasts a clean and intuitive layout, making product discovery and selection a straightforward process. With an emphasis on clear product categorization and an effective search function, customers can easily locate the home appliances they require. The inclusion of rich product descriptions, high-quality images, and customer reviews empowers our customers to make well-informed purchasing decisions.

7.2 FUTURE SCOPE

In our upcoming developments, the system is committed to providing an enriched experience for both our valued customers and dedicated sellers. Registered users, encompassing customers and sellers, can anticipate an array of advanced features including functionalities like enhanced product comparisons, personalized product recommendations, the ability to explore products through Augmented Reality (AR), and the option to visualize products in 3D. They can also look forward to exclusive deals and special offers tailored to their interests, making their interactions with our platform more engaging and rewarding.

On the other hand, our administrators and delivery agents will play pivotal roles in ensuring the seamless operation of our platform. Administrators will continue to manage promotions, discounts, and special offers, keeping an eye on the performance of the website through in-depth reports and analytics.

Additionally, the system has exciting plans for implementing machine learning algorithms to forecast appliance sales and predict customer purchases more accurately. In parallel,

the dedicated delivery agents will uphold the commitment to prompt and accurate deliveries, keeping customers informed about any possible delays and managing the order confirmation process efficiently. In case of returns or exchanges, these agents will coordinate item pickups and ensure that the order status is updated accordingly. These ongoing efforts will ensure an exceptional user experience and further enhance the services we offer.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- PankajJalote, “Software engineering: a precise approach”
- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009
- Ken Schwaber, Mike Beedle, Agile Software Development with Scrum, Pearson (2008)
- Roger S Pressman, “Software Engineering”
- IEEE Std 1016 Recommended Practice for Software Design Descriptions
- Steve McConnell, "Code Complete: A Practical Handbook of Software Construction"
- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software"
- Martin Fowler, "Refactoring: Improving the Design of Existing Code"
- Grady Booch, James Rumbaugh, Ivar Jacobson, "Unified Modeling Language User Guide"
- Alistair Cockburn, "Writing Effective Use Cases"

WEBSITES:

- <https://data-flair.training>
- <https://www.w3schools.com>
- <https://www.tutorialspoint.com>
- <https://www.stackoverflow.com>
- <https://jquery.com/>
- <https://www.programiz.com>
- [/https://chat.openai.com/chat](https://chat.openai.com/chat)
- <https://getbootstrap.com/>

CHAPTER 9

APPENDIX

9.1 Sample Code :

Login Page

```

10 {% load static %}
11 {% load socialaccount %}
12
13 <!DOCTYPE html>
14 <html lang="en">
15 <head>
16     <link rel="stylesheet" href="{% static 'css/login.css' %}">
17     <link rel="stylesheet"
18         href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
19         beta3/css/all.min.css">
20
21 </head>
22 <body>
23     <div class="login-container">
24         <div class="login-content">
25             <div class="login-form">
26                 <h2>Login</h2>
27                 <a href="{% url 'dashboard_home' %}"> <button
28                     class="close-button" onclick="closeLoginForm()"><i class="fas fa-
29                     times"></i></button></a>
30                 {% for messages in messages %}
31                 <h3 style="color: blue">{{ messages }}</h3>
32                 {% endfor %}
33                 <form action="#" method="post" onsubmit="return
34                     validateLoginForm()">
35                     {% csrf_token %}
36                     <div class="form-group">
37                         <label for="username">Email</label>
38                         <input type="username" id="username"
39                             name="email" required>
40                         <span id="usernameError"
41                             class="error"></span>
42                     </div>
43                     <div class="form-group">
44                         <label for="password">Password</label>
45                         <input type="password" id="password"
46                             name="password" required>
47                         <span id="passwordError"
48                             class="error"></span>
49                     </div>
50                     <button type="submit" class="btn btn-
51                         primary"><b>Login</b></button><br><br>
52                     <button class="btn btn-primary">

```

```

44         <a href="{% provider_login_url 'google'
    {%}?next=/">
45             <i class="fab fa-google"></i>
46             Login With Google
47         </a>
48     </button>
49 </form>
50     <form action="{% url 'password_reset' %}"
    method="post">
51         {% csrf_token %}
52         <div class="form-group">
53             <a href="{% url 'password_reset' %}"
    class="forgot-password">Forgot Password?</a>
54         </div>
55     </form>
56     <div class="signup-link">
57         <a href="{% url 'register' %}"> Don't have an
    account? Sign Up</a>
58     </div>
59     <div class="signup-link">
60         <a href="{% url 'sellregister' %}"> Don't have
    an Seller account? Sign Up</a>
61     </div>
62 </div>
63 </div>
64 </div>
65 </body>
66 </html>
67

```

Add Category

```

    {% load static %}
<!DOCTYPE HTML>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>SleekMart Dashboard</title>
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta property="og:title" content="">
    <meta property="og:type" content="">
    <meta property="og:url" content="">
    <meta property="og:image" content="">
    <link rel="shortcut icon" type="image/x-icon" href="{% static
'assets/imgs/theme/favicon.svg' %}">

```

```

    <link href="{% static 'assets/css/main.css' %}" rel="stylesheet"
type="text/css" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.
js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js">
</script>
    <style>
        .error {
            color: red;
            font-size: 0.8rem;
        }
    </style>
</head>

<body>
    <div class="screen-overlay"></div>
    <aside class="navbar-aside" id="offcanvas_aside">
        <div class="aside-top">
            <a href="index.html" class="brand-wrap">
                
            </a>
            <div>
                <button class="btn btn-icon btn-aside-minimize"> <i
class="text-muted material-icons md-menu_open"></i> </button>
            </div>
        </div>
        <nav>
            <ul class="menu-aside">
                <li class="menu-item">
                    <a class="menu-link" href="{% url 'index' %}"> <i
class="icon material-icons md-home"></i>
                        <span class="text">Dashboard</span>
                    </a>
                </li>
                <li class="menu-item has-submenu active">
                    <a class="menu-link" href="page-products-list.html"> <i
class="icon material-icons md-shopping_bag"></i>
                        <span class="text">Manage</span>
                    </a>
                    <div class="submenu">
                        <a href="{% url 'categories' %}"
class="active">Categories</a>
                        <a href="{% url 'subcategories' %}">Sub
Categories</a>
                        <a href="{% url 'productlist' %}">Product List</a>
                    </div>
                </li>
            </ul>
        </nav>
    </aside>
</body>
</html>

```

```

        </div>
    </li>
    <li class="menu-item has-submenu">
        <a class="menu-link" href="page-orders-1.html"> <i
class="icon material-icons md-shopping_cart"></i>
            <span class="text">Orders</span>
        </a>
        <div class="submenu">
            <a href="">Order detail</a>
            <a href="">Order tracking</a>
        <a href="">Invoice</a>
        </div>
    </li>
    <li class="menu-item has-submenu">
        <a class="menu-link" href="page-sellers-cards.html"> <i
class="icon material-icons md-store"></i>
            <span class="text">Sellers</span>
        </a>
        <div class="submenu">
            <a href="{% url 'sellerslist' %}">Sellers list</a>
        </div>
    </li>
    <li class="menu-item has-submenu">
        <a class="menu-link" href="page-categories.html#"> <i
class="icon material-icons md-person"></i>
            <span class="text">Account</span>
        </a>
        <div class="submenu">
            <a href="{% url 'view_customer' %}">User Details</a>
        </div>
    </li>
    <li class="menu-item">
        <a class="menu-link" href=""> <i class="icon material-
icons md-comment"></i>
            <span class="text">Reviews</span>
        </a>
    </li>
</ul>
<hr>
<ul class="menu-aside">
    <li class="menu-item has-submenu">
        <a class="menu-link" href="page-categories.html#"> <i
class="icon material-icons md-settings"></i>
            <span class="text">Settings</span>
        </a>
    </li>
</ul>

```

```

        <li class="menu-item">
        </li>
    </ul>
    <br>
    <br>
</nav>
</aside>
<main class="main-wrap">
    <header class="main-header navbar">
        <div class="col-search">
            <form class="searchform">
                <div class="input-group">
                    <input list="search_terms" type="text" class="form-
control" placeholder="Search term" id="top">
                    <button class="btn btn-light bg" type="button"> <i
class="material-icons md-search"></i></button>
                </div>
                <datalist id="search_terms">
                    <option value="Products">
                    <option value="New orders">
                    <option value="Apple iphone">
                    <option value="Ahmed Hassan">
                </datalist>
            </form>
        </div>
        <div class="col-nav">
            <button class="btn btn-icon btn-mobile me-auto" data-
trigger="#offcanvas_aside"> <i class="material-icons md-apps"></i> </button>
            <ul class="nav">
                <li class="nav-item">
                    <a class="nav-link btn-icon" href="">
                        <i class="material-icons md-notifications
animation-shake"></i>
                        <span class="badge rounded-pill">0</span>
                    </a>
                </li>
                <li class="nav-item">
                    <a class="nav-link btn-icon darkmode" href="#"> <i
class="material-icons md-nights_stay"></i> </a>
                </li>
                <li class="nav-item">
                    <a href="#" class="requestfullscreen nav-link btn-
icon"><i class="material-icons md-cast"></i></a>
                </li>
                <li class="dropdown nav-item">
                    <a class="dropdown-toggle" data-bs-toggle="dropdown"
href="#" id="dropdownAccount" aria-expanded="false"> </a>

```

```

        <div class="dropdown-menu dropdown-menu-end" aria-
labelledby="dropdownAccount">
            <div class="dropdown-divider"></div>
            <a class="dropdown-item text-danger" href="{%
url 'logout' %}"><i class="material-icons md-exit_to_app"></i>Logout</a>
        </div>
    </li>
</ul>
</div>
</header>
<section class="content-main">
    <div class="content-header">
        <div>
            <h2 class="content-title card-title">Categories </h2>
            <p>Add, edit or delete a category</p>
        </div>

        </div>
        {% if messages %}
<div class="messages text-center">
    {% for message in messages %}
        <div class="alert alert-{{ message.tags }}">
            {{ message }}
        </div>
    {% endfor %}
</div>
{% endif %}

    <div class="card">
        <div class="card-body">
            <div class="row">
                <div class="col-md-3">
                    <form method="post" enctype="multipart/form-
data" >

                        {% csrf_token %}
                        <div class="mb-4">
                            <label for="product_name" class="form-
label">Name</label>

                            <input type="text" name="name"
placeholder="Type here" class="form-control" id="product_name" required>
                            <span id="nameError"
class="error"></span>

                        </div>
                        <div class="mb-4">
                            <label for="product_slug" class="form-
label">Slug</label>

                            <input type="text" name="slug"
placeholder="Type here" class="form-control" id="product_slug" required>
                            <span id="slugError"
class="error"></span>

```



```

        </div>
        <div class="mb-4">
            <label class="form-
label">Description</label>
            <input type="text" placeholder="Type
here" name="catdescription" class="form-control" required></textarea>
            <span id="descriptionError"
class="error"></span>
        </div>
        <div class="card-body">
            <div class="input-upload">
                
                <input class="form-control"
type="file" name="image" accept=".jpeg, .jpg, .png" required>
                <span id="imageError"
class="error"></span>
            </div>
        </div>
        <div class="d-grid">
            <button class="btn btn-primary">Create
category</button>
        </div>
    </form>

</div>
<div class="col-md-9">
    <div class="table-responsive">
        <table class="table table-hover">
            <thead>
                <tr>
                    <th class="text-center">
                        <div class="form-check">

                            </div>
                    </th>
                    <th>Name</th>
                    <th>Description</th>
                    <th>Slug</th>
                    <th>Image</th>

                    <th class="text-end">Action</th>
                </tr>
            </thead>
            <tbody>
                {% for category in categories %}
                <tr>
                    <td class="text-center">
                        <div class="form-check">

```

```

        </div>
    </td>
    <td>{{ category.name }}</td>
    <td><b>{{ category.description
}}</b></td>

    <td>{{ category.slug }}</td>
    <td></td>

    <td class="text-end">
        <div class="dropdown">
            <a href="#" data-bs-
toggle="dropdown" class="btn btn-light rounded btn-sm font-sm"> <i
class="material-icons md-more_horiz"></i> </a>

            <div class="dropdown-
menu">

                <a class="dropdown-
item" href="#">View detail</a>

                <a class="dropdown-
item" href="{% url 'update_category' category.slug %}">Edit info</a>
                <a class="dropdown-
item text-danger" href="{% url 'delete_category' category.slug
%}">Delete</a>

            </div>
        </div>
    </td>
</tr>
{% endfor %}
</tbody>

</table>
</div>
</div>
</div>
</div>
</div>
</div>
</section>
<footer class="main-footer font-xs">
    <div class="row pb-30 pt-15">
        <div class="col-sm-6">
            <script>
                document.write(new Date().getFullYear())
            </script> ©, SleekMart - A Home Appliances Ecommerce
Website .

        </div>
        <div class="col-sm-6">
            <div class="text-sm-end">
                All rights reserved

```

```

        </div>
    </div>
</div>
</footer>
</main>
<script src="{% static 'assets/js/vendors/jquery-3.6.0.min.js'
%}"></script>
<script src="{% static 'assets/js/vendors/bootstrap.bundle.min.js'
%}"></script>
<script src="{% static 'assets/js/vendors/select2.min.js' %}"></script>
<script src="{% static 'assets/js/vendors/perfect-scrollbar.js'
%}"></script>
<script src="{% static 'assets/js/vendors/jquery.fullscreen.min.js'
%}"></script>
<script src="{% static 'assets/js/main.js' %}"
type="text/javascript"></script>
<script>

    $(document).ready(function () {
        $("#category-select").change(function () {
            $("#category-filter-form").submit();
        });
    });

    var yourForm = document.getElementById("yourForm");
    var nameInput = document.getElementById("product_name");
    var nameError = document.getElementById("nameError");
    yourForm.addEventListener('submit', function(event) {
        if (!validateName()) {
            event.preventDefault();
        }
    });
    nameInput.addEventListener("input", function () {
        validateName();
    });

    var descriptionInput =
document.querySelector("textarea[name='description']");
    var descriptionError = document.getElementById("descriptionError");
    descriptionInput.addEventListener("input", function () {
        validateDescription();
    });

    var imageInput = document.querySelector("input[type='file']");
    var imageError = document.getElementById("imageError");
    imageInput.addEventListener("input", function () {
        validateImage();
    });

```

```

});

function validateName() {
    var nameValue = nameInput.value.trim();
    nameError.textContent = "";
    if (nameValue === "") {
        nameError.textContent = "Name is required.";
        return false
    } else if (nameValue.length < 3) {
        nameError.textContent = "Name should be at least 3
characters long.";
        return false
    } else if (!/^[a-zA-Z\s&]+$/.test(nameValue)) {
        nameError.textContent = "Name should only contain letters
and spaces.";
        return false
    } else if (!/^[A-Z&][a-zA-Z\s&]*$/.test(nameValue)) {
        nameError.textContent = "Name should start with a capital
letter.";
        return false
    } else if (!/^[^0-9]+$/.test(nameValue)) {
        nameError.textContent = "Name should not contain numbers.";
        return false
    }
    else if (!/^(?!.*(\w)\1{2,})[A-Z&][a-zA-
Z\s&]*$/.test(nameValue)) {
        nameError.textContent = "Name should not contain more than 2
consecutive identical letters.";
        return false;
    }

    else if (isAscendingOrDescendingSequence(nameValue)) {
        nameError.textContent = "Ascending or descending sequences
are not allowed";
        return false
    }
    else if (!/^[^\s]+(\s+[^\s]+)*$/.test(nameValue)) {
        nameError.textContent = "Name should not start or end with
spaces and should not have consecutive spaces.";
        return false
    }
    else{
        nameError.textContent = ""
        return true
    }
}

function isAscendingOrDescendingSequence(str) {
    const ascending = 'abcdefghijklmnopqrstuvwxyz';

```

```

const descending = 'zyxwvutsrqponmlkjihgfedcba';
return ascending.includes(str.toLowerCase()) ||
descending.includes(str.toLowerCase());
}

function validateDescription() {
    var descriptionValue = descriptionInput.value.trim();
    descriptionError.textContent = "";
    if (descriptionValue === "") {
        descriptionError.textContent = "Description is required.";
    }
    else if (!/^[A-Z][A-Za-z\s,.'&-]*(\s?[.]\s?[A-Z][A-Za-z\s,.'&-]*)*$/g.test(descriptionValue)) {
        descriptionError.textContent = "Description should start with a capital letter, contain only letters, spaces, commas, ampersands, hyphens, apostrophes, and each sentence should start with a capital letter and end with a period.";
    }
    else if (/^[0-9]+$/.test(descriptionValue)) {
        descriptionError.textContent = "Description cannot consist of numbers only.";
    }
    else if (!/^(?!.*(\w)\1{2,})[A-Z][a-zA-Z\s,.'-]*$/g.test(descriptionValue)) {
        descriptionError.textContent = "Description should not contain more than 2 consecutive identical letters.";
        return false;
    }
    else if (/((\.)\1{4,}|(,)\2{4,}|(%)\3{4,})/.test(descriptionValue)) {
        descriptionError.textContent = "Description should not contain consecutive repeated characters.";
        return false;
    }
    else if (isAscendingOrDescendingSequence(descriptionValue)) {
        descriptionError.textContent = "Ascending or descending sequences are not allowed";
    }
    else if (!/[a-zA-Z]/.test(descriptionValue)) {
        descriptionError.textContent = "Description should contain at least one letter.";
    }
    else if (descriptionValue.length < 10) {
        descriptionError.textContent = "Description should be at least 10 characters long.";
    }
    else if (descriptionValue.length > 200) {
        descriptionError.textContent = "Description is too long. Maximum 200 characters allowed.";
    }
}

```

```
    }
    function isAscendingOrDescendingSequence(str) {
        const ascending = 'abcdefghijklmnopqrstuvwxyz';
        const descending = 'zyxwvutsrqponmlkjihgfedcba';
        return ascending.includes(str.toLowerCase()) ||
descending.includes(str.toLowerCase());
    }
    function validateImage() {
        imageError.textContent = "";

var allowedExtensions = ["jpg", "jpeg", "png", "gif"];
var maxFileSizeMB = 5;

if (!imageInput.files || imageInput.files.length === 0) {
    imageError.textContent = "Please select an image file.";
} else {
    var imageFile = imageInput.files[0];
    var fileName = imageFile.name;
    var fileExtension = fileName.split('.').pop().toLowerCase();

    if (allowedExtensions.indexOf(fileExtension) === -1) {
        imageError.textContent = "Invalid file format. Allowed formats:
" + allowedExtensions.join(", ");
    } else if (imageFile.size > maxFileSizeMB * 1024 * 1024) {
        imageError.textContent = "File size exceeds the maximum limit of
" + maxFileSizeMB + "MB.";
    }
}
}

    function validateProductForm()
    {
        return true
    }
    $(document).ready(function() {
        $.ajaxSetup({ cache: false });
    });
    window.onpageshow = function(event) {

        if (event.persisted) {
            window.location.reload();
        }
    };
</script>
</body>

</html>
```

Seller Details

```
{% load static %}
<!DOCTYPE HTML>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Admin Dashboard</title>
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta property="og:title" content="">
    <meta property="og:type" content="">
    <meta property="og:url" content="">
    <meta property="og:image" content="">
    <link rel="shortcut icon" type="image/x-icon" href="{% static
'assets/imgs/theme/favicon.svg' %}">
    <link href="{% static 'assets/css/main.css' %}" rel="stylesheet"
type="text/css" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.
js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js">
</script>
<style>
.bg-primary {
    background-color: rgba(8, 129, 120, 0.2) !important;
}
</style>
</head>

<body>
    <div class="screen-overlay"></div>
    <aside class="navbar-aside" id="offcanvas_aside">
        <div class="aside-top">
            <a href="index.html" class="brand-wrap">
                
            </a>
        </div>
        <button class="btn btn-icon btn-aside-minimize" <i
class="text-muted material-icons md-menu_open"></i> </button>
    </div>
    <div>
        <nav>
            <ul class="menu-aside">
                <li class="menu-item active">
```

```

        <a class="menu-link" href="{% url 'index' %}"> <i
class="icon material-icons md-home"></i>
        <span class="text">Dashboard</span>
    </a>
</li>
<li class="menu-item has-submenu">
    <a class="menu-link" href="{% url 'productlist' %}"> <i
class="icon material-icons md-shopping_bag"></i>
        <span class="text">Manage</span>
    </a>
    <div class="submenu">
        <a href="{% url 'categories' %}">Categories</a>
        <a href="{% url 'subcategories' %}">Sub
Categories</a>
        <a href="{% url 'productlist' %}">Product List</a>

    </div>
</li>
<li class="menu-item has-submenu">
    <a class="menu-link" href="page-orders-1.html"> <i
class="icon material-icons md-shopping_cart"></i>
        <span class="text">Orders</span>
    </a>
    <div class="submenu">
        <a href="">Order detail</a>
        <a href="">Order tracking</a>
<a href="">Invoice</a>
    </div>
</li>
<li class="menu-item has-submenu">
    <a class="menu-link" href="page-sellers-cards.html"> <i
class="icon material-icons md-store"></i>
        <span class="text">Sellers</span>
    </a>
    <div class="submenu">

        <a href="{% url 'sellerslist' %}">Sellers list</a>

    </div>
</li>
<li class="menu-item has-submenu">
    <a class="menu-link" href="index.html#"> <i class="icon
material-icons md-person"></i>
        <span class="text">Account</span>
    </a>
    <div class="submenu">
        <a href="{% url 'view_customer' %}">User Details</a>

    </div>

```



```

        </li>
    </ul>
    <hr>
    <ul class="menu-aside">
        <li class="menu-item has-submenu">
            <a class="menu-link" href=""> <i class="icon material-
icons md-settings"></i>
                <span class="text">Settings</span>
            </a>
            <div class="submenu">
            </div>
        </li>
        <li class="menu-item">
        </li>
    </ul>
    <br>
    <br>
</nav>
</aside>
<main class="main-wrap">
    <header class="main-header navbar">
        <div class="col-search">
            <form class="searchform">
                <div class="input-group">
                    <input list="search_terms" type="text" class="form-
control" placeholder="Search term">
                    <button class="btn btn-light bg" type="button"> <i
class="material-icons md-search"></i></button>
                </div>
                <datalist id="search_terms">
                    <option value="Products">
                    <option value="New orders">
                    <option value="Apple iphone">
                    <option value="Ahmed Hassan">
                </datalist>
            </form>
        </div>
        <div class="col-nav">
            <button class="btn btn-icon btn-mobile me-auto" data-
trigger="#offcanvas_aside"> <i class="material-icons md-apps"></i> </button>
            <ul class="nav">
                <li class="nav-item">
                    <a class="nav-link btn-icon" href="#">
                        <i class="material-icons md-notifications
animation-shake"></i>
                        <span class="badge rounded-pill">0</span>
                    </a>
                </li>
                <li class="nav-item">

```

```

        <a class="nav-link btn-icon darkmode" href="#"> <i
class="material-icons md-nights_stay"></i> </a>
    </li>
    <li class="nav-item">
        <a href="{% url 'index' %}" class="requestfullscreen
nav-link btn-icon"><i class="material-icons md-cast"></i></a>
    </li>
    <li class="dropdown nav-item">
        <a class="dropdown-toggle" data-bs-toggle="dropdown"
href="#" id="dropdownAccount" aria-expanded="false"> </a>
        <div class="dropdown-menu dropdown-menu-end" aria-
labelledby="dropdownAccount">

            <div class="dropdown-divider"></div>
            <a class="dropdown-item text-danger" href="{%
url 'logout' %}"><i class="material-icons md-exit_to_app"></i>Logout</a>
        </div>
    </li>
</ul>
</div>
</header>
{% if messages %}
<div class="messages text-center">
    {% for message in messages %}
        <div class="alert alert-{{ message.tags }}">
            {{ message }}
        </div>
    {% endfor %}
</div>
{% endif %}
<section class="content-main">
    <div class="content-header">
        <a href="javascript:history.back()"><i class="material-icons
md-arrow_back"></i> Go back </a>
    </div>
    <div class="card mb-4">
        <div class="card-header bg-primary" style="height:150px">
        </div>
        <div class="card-body">
            <div class="row">
                <div class="col-xl col-lg flex-grow-0" style="flex-
basis:230px">

                    <div class="img-thumbnail shadow w-100 bg-white
position-relative text-center" style="height:190px; width:200px; margin-
top:-120px">

                        {% if seller.user.profile_pic %}

```

```

        
        {% else %}

            
        {% endif %}
    </div>

</div> <!-- col.// -->
<div class="col-xl col-lg">
    <h3>{{ seller.user.name }}</h3>
    <p>{{ seller.business_name }}</p>
</div>
</div>
<hr class="my-4">
<div class="row g-4">
    <div class="col-md-12 col-lg-4 col-xl-2">
        <article class="box">
            <p class="mb-0 text-muted">Total
Products:</p>

            <h5 class="text-success">{{ product_count
    }}</h5>

            <p class="mb-0 text-muted">Revenue:</p>
            <h5 class="text-success mb-0">${0}</h5>
        </article>
    </div>
    <div class="col-sm-6 col-lg-4 col-xl-3">
        <h6>Contacts</h6>
        <p>
            Name: {{ seller.user.name }}<br>
            Email: {{ seller.user.email }}<br>
            Mobile: {{ seller.user.mobile }}<br>
        </p>
    </div>
    <div class="col-sm-6 col-lg-4 col-xl-3">
        <h6>Address</h6>
        <p>
            {{ seller.user.address }}<br>
        </p>
    </div>
</div>
</div>
</div>
<div class="card mb-4">
    <div class="card-body">
        <h5 class="card-title">Products by seller</h5>
        <div class="row">
            {% if seller.product_set.all %}

```

```

        {% for product in seller.product_set.all %}
        {% if not product.status %}
        <div class="col-xl-2 col-lg-3 col-md-6"
style="width:350px;">
            <div class="card card-product-grid">
                <a href="" class="img-wrap">  </a>
                <div class="info-wrap">
                    <a href="page-seller-detail.html#"
class="title">{{ product.name }}</a>
                    <div class="price mt-1">${{
product.selling_price }}</div>
                </div>
            </div>
        {% endif %}
        {% endfor %}
        {% else %}
        <p>No Products</p>
        {% endif %}
    </div>
</div>
<div class="pagination-area mt-30 mb-50">
    <nav aria-label="Page navigation example">
        <ul class="pagination justify-content-start">
            <li class="page-item active"><a class="page-link"
href="page-seller-detail.html#">01</a></li>
            <li class="page-item"><a class="page-link"
href="page-seller-detail.html#">02</a></li>
            <li class="page-item"><a class="page-link"
href="page-seller-detail.html#">03</a></li>
            <li class="page-item"><a class="page-link dot"
href="page-seller-detail.html#">...</a></li>
            <li class="page-item"><a class="page-link"
href="page-seller-detail.html#">16</a></li>
            <li class="page-item"><a class="page-link"
href="page-seller-detail.html#"><i class="material-icons md-
chevron_right"></i></a></li>
        </ul>
    </nav>
</div>
</section>
<footer class="main-footer font-xs">
    <div class="row pb-30 pt-15">
        <div class="col-sm-6">
            <script>
                document.write(new Date().getFullYear())

```

```

        </script> ©, SleekMart - A Home appliance Ecommerce
Website .

        </div>
        <div class="col-sm-6">
            <div class="text-sm-end">
                All rights reserved
            </div>
        </div>
    </div>
</footer>
</main>
<script src="{% static 'assets/js/vendors/jquery-3.6.0.min.js'
%}"></script>
    <script src="{% static 'assets/js/vendors/bootstrap.bundle.min.js'
%}"></script>
    <script src="{% static 'assets/js/vendors/select2.min.js' %}"></script>
    <script src="{% static 'assets/js/vendors/perfect-scrollbar.js'
%}"></script>
    <script src="{% static 'assets/js/vendors/jquery.fullscreen.min.js'
%}"></script>
    <script src="{% static 'assets/js/vendors/chart.js' %}"></script>
    <script src="{% static 'assets/js/main.js' %}"
type="text/javascript"></script>
    <script src="{% static 'assets/js/custom-chart.js' %}"
type="text/javascript"></script>
    <script>
        $(document).ready(function() {

            $.ajaxSetup({ cache: false });

        });
        window.onpageshow = function(event) {

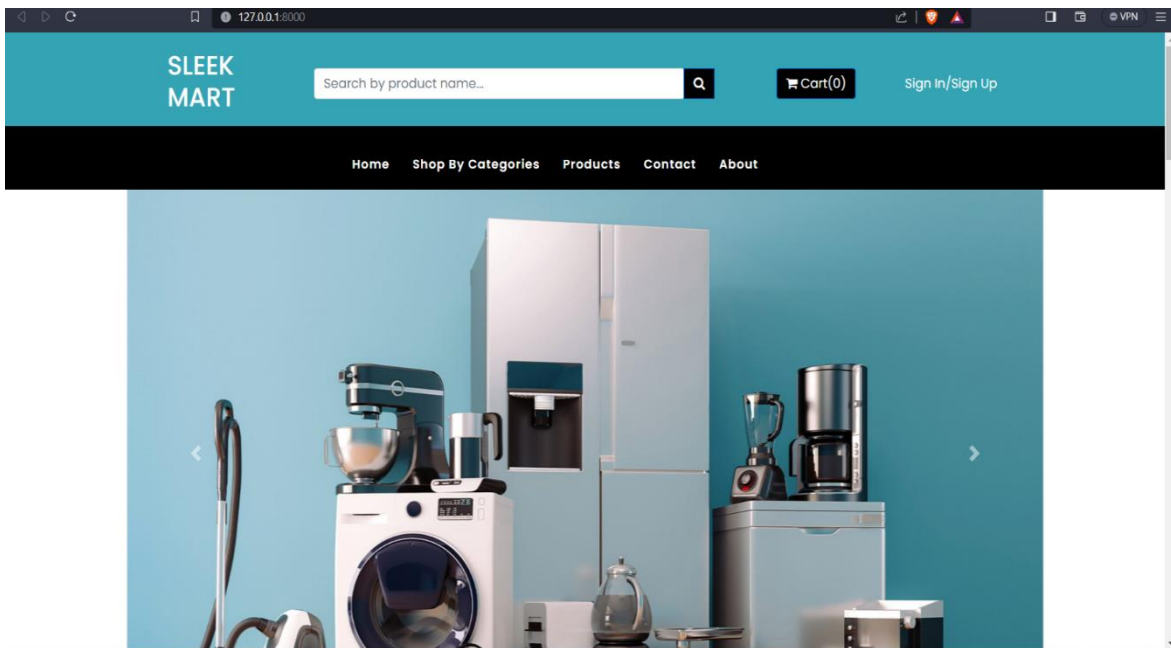
            if (event.persisted) {
                window.location.reload();
            }
        };
    </script>
</body>

</html>

```

9.2 Screen Shots

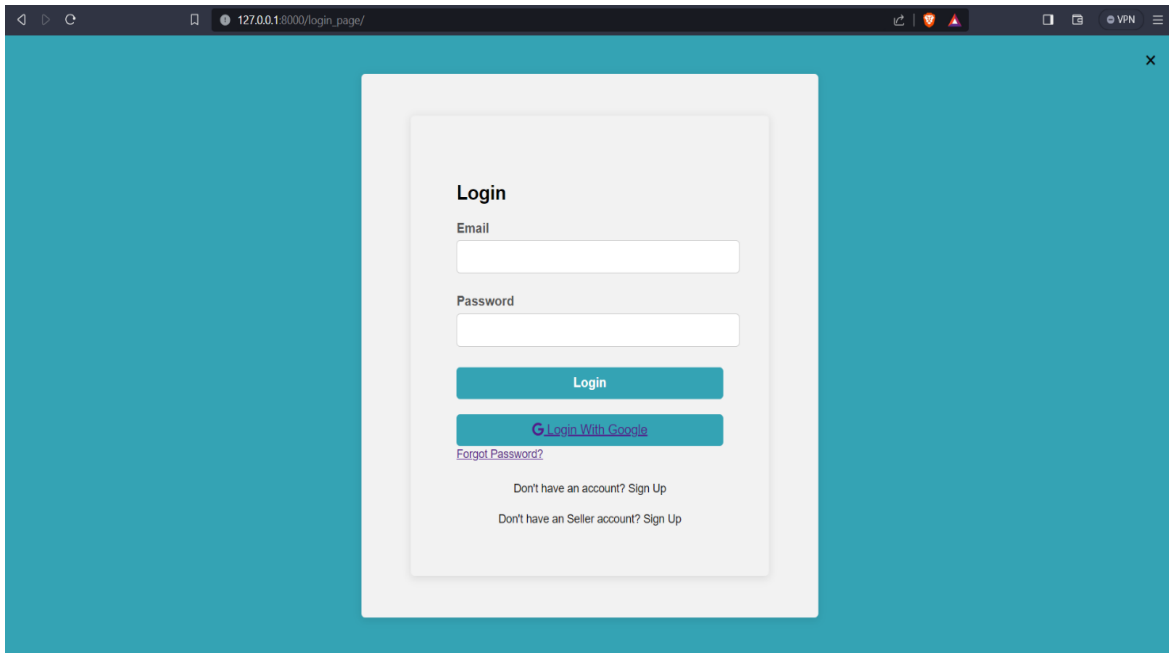
1. Index Page:



2. Customer Registration Form

A screenshot of the customer registration form on the SLEEK MART website. The form is titled 'Registration' and is centered on a teal background. It contains the following fields: 'Enter Your Name', 'Mobile Number', 'Email', 'Password', and 'Confirm Password'. Each field is represented by a white input box. Below the input boxes is a teal 'Submit' button. At the bottom of the form, there is a link that says 'Already a user? Login'. The browser's address bar shows the URL '127.0.0.1:8000/register/'.

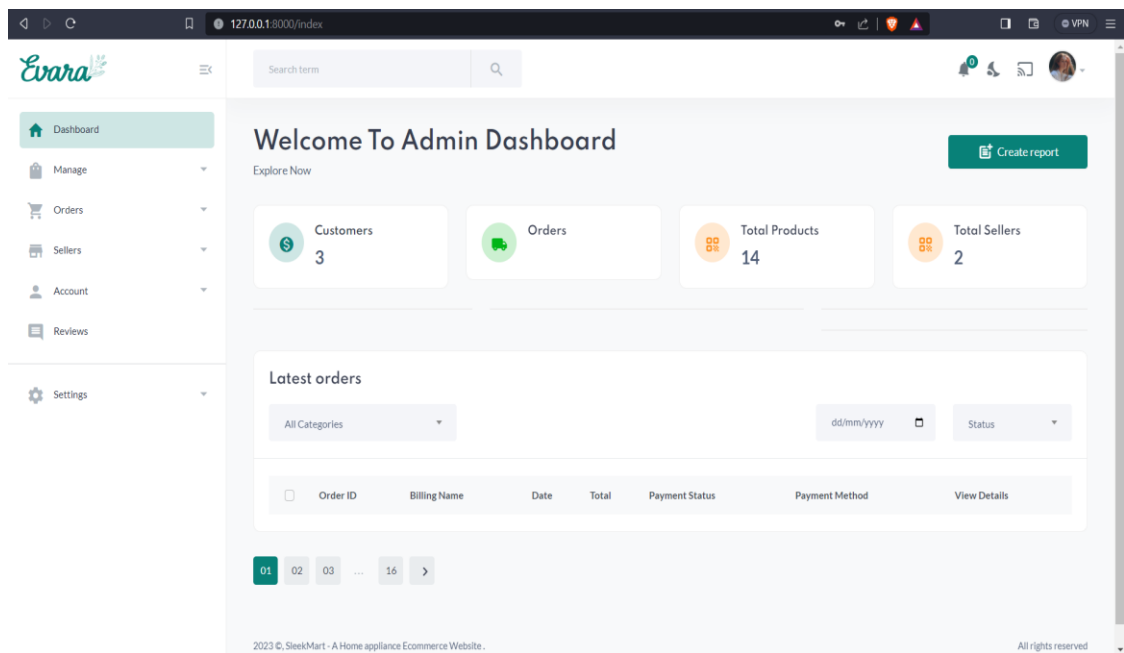
3. Login Form



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/login_page/". The page has a teal background. In the center, there is a white login form with the following elements:

- Login** (Section Header)
- Email** (Label) with a text input field.
- Password** (Label) with a text input field.
- Login** (Button)
- [Login With Google](#) (Link)
- [Forgot Password?](#) (Link)
- Don't have an account? [Sign Up](#) (Text)
- Don't have an Seller account? [Sign Up](#) (Text)

4. Admin Dashboard



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/index". The page has a light gray background. The dashboard includes the following elements:

- Header:** Logo "Evara" on the left, a search bar in the center, and user profile icons on the right.
- Left Sidebar:** A list of navigation items: Dashboard, Manage, Orders, Sellers, Account, Reviews, and Settings.
- Main Content Area:**
 - Welcome To Admin Dashboard** (Section Header)
 - Explore Now** (Text)
 - Create report** (Button)
 - Summary Cards:** Four cards showing "Customers: 3", "Orders", "Total Products: 14", and "Total Sellers: 2".
 - Latest orders** (Section Header)
 - Filters:** "All Categories" (dropdown), "dd/mm/yyyy" (date picker), and "Status" (dropdown).
 - Table:** A table with columns: Order ID, Billing Name, Date, Total, Payment Status, Payment Method, and View Details.
 - Pagination:** A row of buttons: 01, 02, 03, ..., 16, >.
- Footer:** "2023 ©, SlekMart - A Home appliance Ecommerce Website." and "All rights reserved".

5. Admin Category Page

Categories
Add, edit or delete a category

Name
Type here

Slug
Type here

Description
Type here

Choose File No file chosen

Create category

Name	Description	Slug	Image	Action
Refrigeration Appliances	Refrigeration appliances enable us to store and preserve food items and beverages. We also use refrigeration appliances for cooling purposes like making ice cubes and ice creams.	refrigeration-appliances		
Cooking Appliances	Cooking appliances always top the home appliances list of restaurants and eateries. Without the aid of cooking appliances, producing food on a large scale is next to impossible.	cooking-appliances		
Washing And Drying Appliances	Washing Appliances	washing-and-drying-appliances		

6. Seller Page

Welcome To Seller Dashboard
Explore Now

Create report

Categories 3

Orders 12

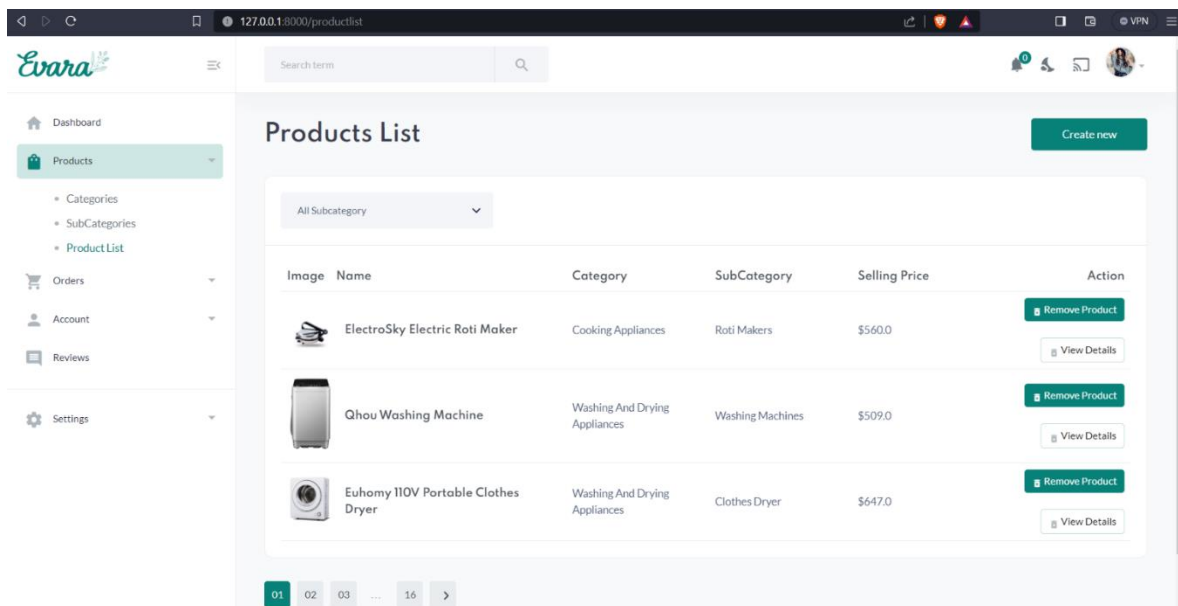
Products 3

Latest orders

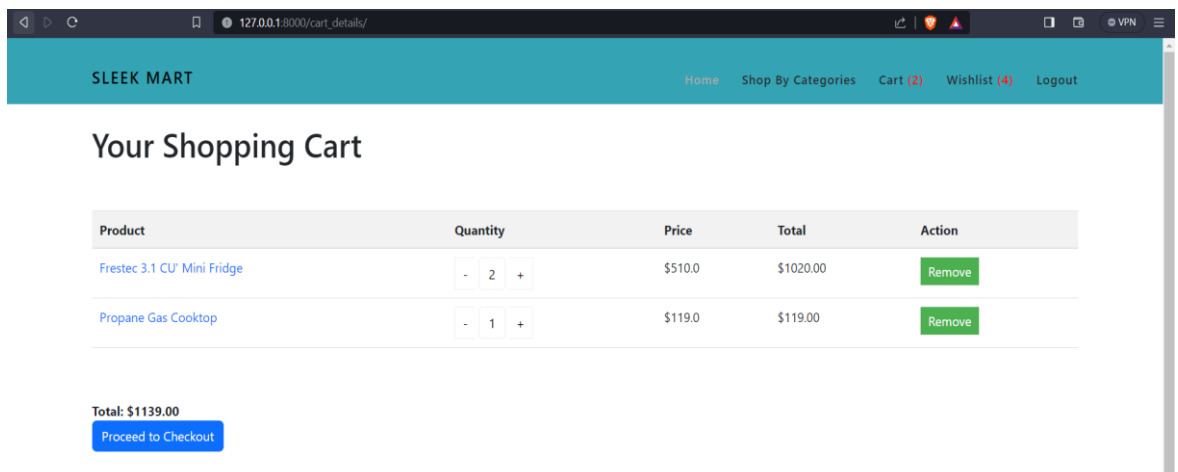
All Categories dd/mm/yyyy Status

Order ID	Billing Address	Date	Total	View Details
592	Chilampara(H), Kanjirapally PO, 623578	Oct. 13, 2023, 5:15 a.m.	560.00	View Details
622	Chilampara(H), Kanjirapally PO, 623578	Oct. 13, 2023, 5:36 p.m.	4480.00	View Details
564	Vettuvelli(H), Kanjirapally PO, 689492	Oct. 11, 2023, 4:57 p.m.	3563.00	View Details
568	Vettuvelli(H), Kanjirapally PO, 689492	Oct. 11, 2023, 4:59 p.m.	3563.00	View Details

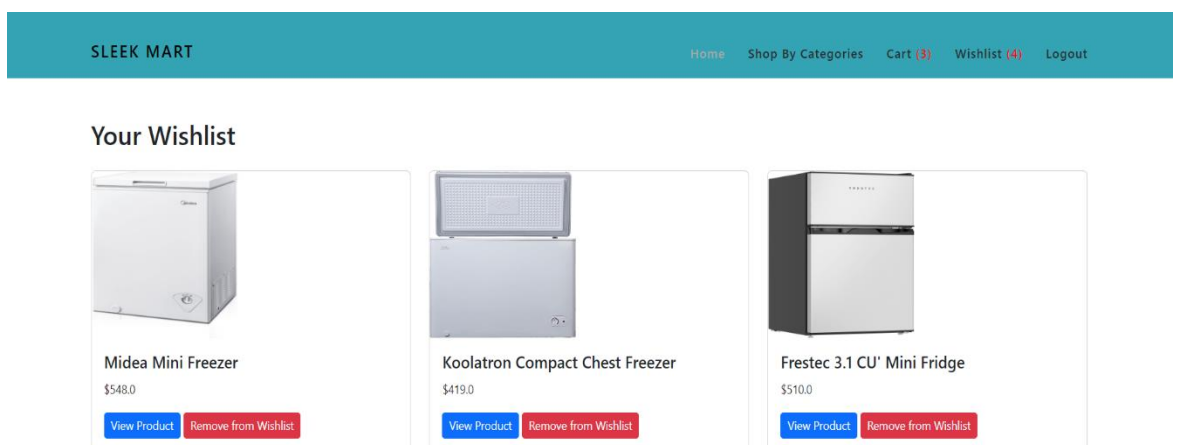
7. Product Page



8. Cart Page



9. Wishlist Page



10. Payment Page

