

# **Tech Trove**

## **An online mobile store**

*Mini Project Report*

*Submitted by*

**Manjari Jayan**

**Reg. No.: AJC19MCA-I036**

*In Partial fulfillment for the Award of the Degree of*

**INTEGRATED MASTER OF COMPUTER APPLICATIONS**

**(INMCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



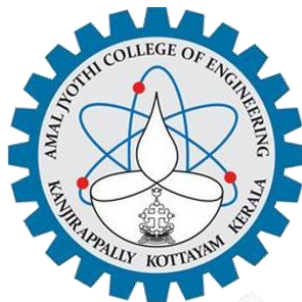
**AMAL JYOTHI COLLEGE OF ENGINEERING**

**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, “**TECHTROVE**” is the bona fide work of **MANJARI JAYAN (Regno: AJC19MCA-I036)** 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. Jetty Benjamin**

**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 “**TECHTROVE**” 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 (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

**Date: 30/10/2023**

**MANJARI JAYAN**

**KANJIRAPPALLY**

**Reg: AJC19MCA-I036**

## 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 his 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. Jetty Benjamin** 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.

MANJARI JAYAN

# ABSTRACT

‘TechTrove’ is an online mobile shopping platform that intends to provide a quick and user friendly platform for users to explore, select, and purchase a wide choice of mobile devices from various brands and categories. The website is intended to improve the whole shopping experience by providing a streamlined interface, secure transactions, and tailored recommendations. This system is intended to handle all admin, user and delivery person activities.

The shopping site has an effective product listing and search system that allows users to easily select mobile devices based on criteria such as brand, price range, features, and customer ratings. This improves the user experience and allows for more efficient product discovery.

Furthermore, the website employs responsive design concepts, ensuring interoperability across various devices such as PCs, tablets, and mobile phones. Users can now access the site from multiple platforms and enjoy a consistent and optimized shopping experience.

There are mainly three modules

- Admin
- User
- Delivery Person

Each module plays a vital role in providing a user-friendly and comprehensive online mobile shopping experience, allowing efficient management of products, user accounts, and order deliveries, ultimately enhancing customer satisfaction and convenience.

In conclusion, the online mobile shopping site aspires to be a comprehensive and user friendly platform for purchasing mobile gadgets. The website attempts to improve the whole shopping experience for customers in the digital era by providing a user-friendly layout, customized profiles, secure payment choices, and rapid order tracking.

# CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	01
1.1	PROJECT OVERVIEW	02
1.2	PROJECT SPECIFICATION	02
2	SYSTEM STUDY	03
2.1	INTRODUCTION	04
2.2	EXISTING SYSTEM	04
2.3	DRAWBACKS OF EXISTING SYSTEM	05
2.4	PROPOSED SYSTEM	06
2.5	ADVANTAGES OF PROPOSED SYSTEM	07
3	REQUIREMENT ANALYSIS	08
3.1	FEASIBILITY STUDY	09
3.1.1	ECONOMICAL FEASIBILITY	09
3.1.2	TECHNICAL FEASIBILITY	09
3.1.3	BEHAVIORAL FEASIBILITY	10
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	10
3.2	SYSTEM SPECIFICATION	12
3.2.1	HARDWARE SPECIFICATION	12
3.2.2	SOFTWARE SPECIFICATION	12
3.3	SOFTWARE DESCRIPTION	13
3.3.1	DJANGO	13
3.3.2	SQLITE	13
4	SYSTEM DESIGN	14
4.1	INTRODUCTION	15
4.2	UML DIAGRAM	15
4.2.1	USE CASE DIAGRAM	16
4.2.2	SEQUENCE DIAGRAM	17
4.2.3	STATE CHART DIAGRAM	19
4.2.4	ACTIVITY DIAGRAM	20
4.2.5	CLASS DIAGRAM	21
4.2.6	OBJECT DIAGRAM	23
4.2.7	COMPONENT DIAGRAM	25

<b>4.2.8</b>	<b>DEPLOYMENT DIAGRAM</b>	<b>26</b>
<b>4.3</b>	<b>USER INTERFACE DESIGN USING FIGMA</b>	<b>28</b>
<b>4.4</b>	<b>DATABASE DESIGN</b>	<b>30</b>
<b>5</b>	<b>SYSTEM TESTING</b>	<b>37</b>
<b>5.1</b>	<b>INTRODUCTION</b>	<b>38</b>
<b>5.2</b>	<b>TEST PLAN</b>	<b>38</b>
<b>5.2.1</b>	<b>UNIT TESTING</b>	<b>39</b>
<b>5.2.2</b>	<b>INTEGRATION TESTING</b>	<b>39</b>
<b>5.2.3</b>	<b>VALIDATION TESTING</b>	<b>40</b>
<b>5.2.4</b>	<b>USER ACCEPTANCE TESTING</b>	<b>40</b>
<b>5.2.5</b>	<b>AUTOMATION TESTING</b>	<b>40</b>
<b>5.2.6</b>	<b>SELENIUM TESTING</b>	<b>41</b>
<b>6</b>	<b>IMPLEMENTATION</b>	<b>51</b>
<b>6.1</b>	<b>INTRODUCTION</b>	<b>52</b>
<b>6.2</b>	<b>IMPLEMENTATION PROCEDURE</b>	<b>53</b>
<b>6.2.1</b>	<b>USER TRAINING</b>	<b>53</b>
<b>6.2.2</b>	<b>TRAINING ON APPLICATION SOFTWARE</b>	<b>53</b>
<b>6.2.3</b>	<b>SYSTEM MAINTENANCE</b>	<b>53</b>
<b>7</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>54</b>
<b>7.1</b>	<b>CONCLUSION</b>	<b>55</b>
<b>7.2</b>	<b>FUTURE SCOPE</b>	<b>55</b>
<b>8</b>	<b>BIBLIOGRAPHY</b>	<b>57</b>
<b>9</b>	<b>APPENDIX</b>	<b>59</b>
<b>9.1</b>	<b>SAMPLE CODE</b>	<b>60</b>
<b>9.2</b>	<b>SCREEN SHOTS</b>	<b>67</b>

## List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language.
CSS	-	Cascading Style Sheet
UI	-	User Interface
UML	-	Unified Modelling Language
PostgreSQL	-	Postgres Structured Query Language
1NF	-	First Normal Form
2NF	-	Second Normal Form
3NF	-	Third Normal Form
PK	-	Primary Key
FK	-	Foreign Key
JSON	-	JavaScript Object Notation
DJANGO	-	Web Framework
JS	-	JavaScript
AJAX	-	Asynchronous JavaScript and XML Environment



# **CHAPTER 1**

## **INTRODUCTION**

## 1.1 PROJECT OVERVIEW

'TechTrove' is an online mobile shopping platform that aims to provide a rapid and userfriendly platform for consumers to explore, select, and purchase a wide range of mobile devices from various brands and categories. The website's goal is to improve the whole shopping experience by providing a streamlined interface, safe transactions, and personalized recommendations. This system is designed to handle the admin, user, and delivery person tasks. The shopping platform includes an effective product listing and search engine that allows consumers to readily select mobile devices based on factors such as brand, price range, features, and customer ratings. This improves the user experience and enables for more efficient product discovery.

## 1.2 PROJECT SPECIFICATION

This is a website where users can purchase a variety of mobile phones to meet their specific needs and preferences. The system consists of 3 actors. They are:

### **Admin:**

1. Secure login for administrators.
2. Management of product listings, including adding, updating, and deleting products
3. Inventory management to ensure accurate product information
4. Assignment and oversight of delivery personnel.
5. Real-time view of customer profiles, shopping carts and orders.
6. Approve or deny registration of delivery personnel.
7. Tracking of delivery status updates.

### **User:**

1. Account creation and management for users
2. Browsing and searching for products with advanced filters
3. Detailed product specifications for informed decision-making
4. 3D product viewing and product comparison
5. Shopping cart and wishlist management
6. Voice search and translation features for enhanced accessibility
7. Seamless integration of payment gateways for secure transactions
8. Order tracking and shipment status updates
9. Ability to provide reviews and ratings
10. Integration of a user-friendly ChatBot

**Delivery Person:**

1. Secure registration and login for delivery personnel.
2. Real-time access to assigned delivery orders
3. Visibility of delivery locations and customer contact information.
4. Ability to update delivery status for efficient order deliveries.

## **CHAPTER 2**

### **SYSTEM STUDY**

## 2.1 INTRODUCTION

A crucial stage, in the development of a system is system analysis, which entails collecting and evaluating data to pinpoint issues and offer remedies. In this phase it is crucial to have communication, between the users of the system and the developers. It is always important for any process of system development to start with an analysis of the system. The system analyst takes on the role of an investigator thoroughly assessing the performance of the existing system. This involves identifying the inputs and outputs of the system as establishing how its processes relate to the outcomes achieved within the organization.

Various methods, such, as surveys and interviews are employed to gather information. The main goals include comprehending how the system operates, identifying areas of concern and proposing solutions to address the business challenges. The designer takes on the role of problem solver. Thoroughly compares the suggested fixes with the existing system. Once the best option is selected the user is given an opportunity to accept or reject the advice. This process continues until the user expresses satisfaction after evaluating their input alongside the proposed idea.

The phase of gathering and evaluating data, for system study is referred to as a preliminary study. It is crucial to carry out a preliminary study to guarantee the triumph of a system development endeavor.

## 2.1 EXISTING SYSTEM

Customers must allot time for travel, parking, and shopping in the current system. This might be tough for people who have hectic schedules, employment responsibilities, or other time restraints that make it difficult to find devoted time for shopping. Physical stores have specified opening hours and frequently stick to a strict schedule. These hours may not correspond to customers' availability, particularly for those who work late shifts, have irregular schedules, or prefer to shop outside of typical store hours. Some physical stores have delivery services, so offline management of each order and delivery person is a common duty. It takes extra workers and a supervisor to collect updates from the delivery person and manually assign fresh deliveries.

### **2.2.1 NATURAL SYSTEM STUDIED**

In the current mobile selling system, customers typically engage with physical retail stores. They visit these brick-and-mortar shops to explore and purchase the mobile devices they desire. However, these stores often have limited stock or may not have the specific mobile device customers are looking for. This can lead to the inconvenience of having to visit multiple stores, which can be time-consuming and require considerable effort.

A practical and efficient solution to overcome these challenges is the introduction of a mobile device e-commerce platform. This digital platform allows customers to browse, compare, and purchase a wide range of mobile devices from the comfort of their own homes. By doing so, it eliminates the need for extensive travel and provides access to a broader selection of mobile products. Customers also benefit from transparent pricing, detailed product descriptions, and a variety of payment options, all of which enhance the overall mobile buying experience.

### **2.2.2 DESIGNED SYSTEM STUDIED**

To address the shortcomings of offline mobile phone purchases, this system enables users to comfortably select mobile phones and related items online, tailored to their preferences. In this project, users can choose mobile phones based on their interests and requirements. The system facilitates customers in placing orders by collecting their contact information, preferred phone specifications, special service requests, and order quantities. Upon order submission, a confirmation report is generated and sent to the customer for their review.

## **2.2 DRAWBACKS OF EXISTING SYSTEM**

- **Time and travel constraints:** Offline mobile phone purchasing necessitates customers to personally visit the store, leading to potential time and convenience challenges. Customers must set aside time for travel, parking, and browsing through the store's phone selection. This can pose challenges, especially for individuals with hectic schedules or those residing far from mobile phone retailers, resulting in time and effort inefficiencies.
- **Lack of detailed information:** Offline mobile phone purchasing may not offer the level of comprehensive and detailed information available on online platforms. In physical stores, product information may be limited to the packaging or signage, potentially

lacking in-depth details about phone specifications, features, or specific usage recommendations. This

lack of information can hinder customers' ability to make informed decisions about their mobile phone purchases.

- **Dependence on store hours and availability:** Physical mobile phone stores have fixed operating hours, requiring customers to visit within those time constraints. This limitation can be inconvenient for individuals with rigid schedules or those who work during the store's operating hours. Moreover, mobile phone stock availability may fluctuate, potentially leading to disappointment if the specific phone models they seek are out of stock or temporarily unavailable.
- **Limited selection:** Offline mobile phone purchasing can sometimes be constrained in terms of the variety and choices available. Physical stores may have limited display space and might not provide the same extensive range of mobile phone models as online platforms. This limitation can narrow the options accessible to customers, making it more challenging to find specific or unique phone models they may be seeking.

## **2.3 PROPOSED SYSTEM**

The proposed mobile buying system envisions a comprehensive online platform where customers can effortlessly explore and select from a diverse range of mobile devices from the comfort of their own spaces. By offering an extensive array of mobile products, detailed product information, user reviews, and competitive pricing, this approach aims to overcome the limitations associated with traditional brick-and-mortar mobile retailers. The platform will also prioritize user-friendly design and straightforward navigation to ensure a seamless mobile shopping experience. To enhance customer satisfaction and build trust, this system will integrate secure payment options and reliable delivery services. Additionally, the implementation of a responsive customer support system will promptly address any inquiries or concerns, thereby improving the overall effectiveness and efficiency of the proposed online platform for mobile devices.

## 2.4 ADVANTAGES OF PROPOSED SYSTEM

- **Detailed product information:** The proposed system can offer comprehensive product details for each mobile phone model. Customers can access in-depth descriptions, including specifications, features, operating instructions, and potential applications. This information empowers customers to make informed decisions and select mobile phones that align with their specific technological needs and preferences.
- **Convenience:** The proposed system provides a high level of convenience to customers. They can access the system anytime and anywhere with an internet connection, allowing them to browse, select, and purchase mobile phones at their own convenience. This eliminates the need for physical travel, saving time and effort.
- **Expanded product availability:** The proposed system can offer a wider range of mobile phone options compared to traditional offline retailers. It can connect customers with various mobile phone suppliers, both local and international, thereby increasing the availability of different phone models and brands. This expanded product availability provides customers with a greater selection and more opportunities to find the specific mobile phones they are looking for.
- **Efficient search options:** The proposed system can incorporate advanced search and filtering features to streamline mobile phone selection. Customers can use specific criteria such as brand, operating system, price range, or desired features to refine their search and locate their preferred mobile phones more efficiently. This not only saves time but also assists customers in finding phones that precisely match their individual requirements.
- **Efficient search options:** The proposed system can integrate advanced search and filtering capabilities to simplify the process of selecting mobile phones. Customers can utilize specific criteria such as brand, operating system, price range, or desired features to fine-tune their search and quickly identify their preferred mobile devices. This not only reduces the time spent searching but also aids customers in discovering phones that precisely align with their unique needs and preferences.



## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**

### **3.1 FEASIBILITY STUDY**

A feasibility study is a thorough examination of a proposed project, initiative, or business proposal to determine its viability and potential for success. It entails acquiring and analyzing important data in order to evaluate many variables such as technical, economic, legal, operational, and scheduling factors. A feasibility study's major goal is to give stakeholders with important insights into the project's viability and dangers, allowing them to make educated decisions. A feasibility study is an important tool for establishing whether a proposed project is viable, realistic, and aligned with the organization's goals and resources by identifying potential challenges, estimating costs and benefits, and comparing alternatives.

#### **3.1.1 Economical Feasibility**

The economic feasibility analysis for 'TechTrove' website entails a thorough examination of its financial sustainability. To determine whether a project will be beneficial and viable, its predicted cost, prospective sources of revenue, and expected return on investment must be evaluated. Some the questions addressed during the inquiry are the following:

##### **1.What is the cost of the hardware and software?**

All the resources are readily available.

##### **2. How will the TechTrove's ongoing operational costs be controlled and sustained?**

As the proposed system is being created as part of a academic project, it will focus on the use of open-source platforms and free tools for website building and hosting. Inventory may consist of virtual or sample products. As a result, the 'TechTrove' has no ongoing operating costs.

#### **3.1.2 Technical Feasibility**

The technical feasibility study is critical in determining the practicability and achievability of a proposed system's technical aspects. The analysis assures that the project can be completed effectively within the academic restrictions by examining the availability of essential resources, technical capabilities, and compatibility with existing infrastructure.

**1. What technical skills are required to develop and implement the proposed system?**

The proposed framework requires knowledge of web development tools such as HTML, CSS, JavaScript, Ajax as well as a backend language such as Python Django. Database management systems like SQLite may also be required.

**2. Is it necessary for users to be familiar with the technology?**

No

**3. How well does the proposed system function with various web browsers and devices?**

The system is supported by all modern browsers and devices.

**3.1.3 Behavioral Feasibility**

The behavioral feasibility is critical since it focuses on understanding user behavior, preferences, and concerns. The project team obtains vital insights about the target population by performing this assessment, allowing them to design the website to match users' individual requirements and expectations.

**1. How will the proposed system ensure user adoption and engagement?**

User adoption and engagement will be prioritized through a simple and user-friendly design, personalized product recommendations, and interactive features such as customer reviews and ratings.

**1. How will user trust and credibility be ensured on the website?**

By displaying customer reviews and secure payment icons. The project would be beneficial since it would meet the objectives when designed and implemented. All behavioral concerns are carefully studied, and it is determined that the project is behaviorally feasible.

### **3.1.4 Feasibility Study Questionnaire**

#### **1. Project Overview?**

'TechTrove' is an online mobile shopping platform that aims to provide a rapid and userfriendly platform for consumers to explore, select, and purchase a wide range of mobile devices from various brands and categories. The website's goal is to improve the whole shopping experience by providing a streamlined interface, safe transactions, and personalized recommendations. This system is designed to handle the admin, user, and delivery person tasks. The shopping platform includes an effective product listing and search engine that allows consumers to readily select mobile devices based on factors such as brand, price range, features, and customer ratings. This improves the user experience and enables for more efficient product discovery.

#### **2. To what extend the system is proposed for?**

'TechTrove' provides a seamless and time saving shopping experience for customers, streamlining the process of acquiring mobile phones and related products. It gives consumers access to premium products, easy pricing comparisons and convenient order tracking, all while providing efficient management for administrators and delivery staff. The system is intended to provide a satisfactory and enjoyable buying experience for all users involved in the online mobile store.

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

Customers, Guest Users and Online shoppers

#### **4. List the Modules included in your System?**

Admin, User (Customer) and Delivery Person

#### **5. Identify the users in your project?**

Guest Users & Customers

#### **6. Who owns the system?**

Admin

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

E-Commerce Industry

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

Dennis Jose (Konattu Group, Kottayam)

**9. Questionnaire to collect details about the project? (min 10 questions, include descriptive answers, attach additional docs (e.g. Bill receipts, certificate models), if any?)****1. Which brands are the most popular in your store?**

Apple, Samsung, Xiaomi, OPPO, Realme, Motorola.

**2. Which type of payment method is used in your store?**

Cash or credit card payments are accepted.

**3. How do you incorporate customer review into your service improvements?**

We can alter our product offers based on customer review to better meet their needs.

**4. Do you keep track of your inventory using inventory management software or systems?**

No. We have a printed recorder. When a product is sold or restricted, update the inventory record.

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

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

**6. How do you market your business and attract new customers?**

By reward programs.

**7. How do you handle orders from customers?**

We handle it by offering both telephone support and in-person assistance.

**8. What are the biggest difficulties you have when managing your mobile store?**

My biggest difficulties are expanding my consumer base, effectively managing my inventory, and ensuring the legitimacy of my products.

**9. How do you manage product shipments?**

We occasionally work with reputable courier firms in addition to in-house delivery for specific deliveries.

**10. How do you deal with customer questions?**

For customer inquiries, we provide both telephone support and in-person assistance.

## 3.1 SYSTEM SPECIFICATION

### 3.2.1 Hardware Specification

Processor - Intel Core i5

RAM - 8 G B

Hard disk - 2 5 6 G B

### 3.2.2 Software Specification

Front End - HTML, CSS

Back End - DJANGO

Database - SQLite

Client on PC - Windows 11

Technologies used - HTML5, CSS , JS, JQuery, DJANGO

## **3.3 SOFTWARE DESCRIPTION**

### **3.3.1 Python**

Python is a high-level, dynamically-typed, and interpreted programming language known for its readability and ease of use. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python has a vast standard library, providing modules and packages for a wide range of tasks, from web development to scientific computing. Its syntax, using indentation for code blocks, promotes clean and consistent coding practices. Python's extensive community support and third-party libraries make it a versatile choice for various applications, including web development, data analysis, machine learning, and automation.

### **3.3.2 Django**

Django is a high-level, open-source web framework written in Python, designed for building robust and scalable web applications. It follows the Model-View-Controller (MVC) architectural pattern, promoting clean and maintainable code. With built-in features like an Object-Relational Mapping (ORM) system, Django simplifies database interaction. Its extensive and well-documented library, including authentication, URL routing, and templating, accelerates development. Furthermore, Django emphasizes security, protecting against common web vulnerabilities and providing a strong foundation for building secure web applications.

### **3.3.3 SQLite**

SQLite is a lightweight, self-contained, serverless, and open-source relational database management system (RDBMS). It is designed for embedded and local data storage, making it ideal for mobile and desktop applications. SQLite operates as a single-user database engine, eliminating the need for complex setup or configuration. Despite its small footprint, it supports standard SQL and offers transactional integrity for data consistency. Developers often choose SQLite for its speed, simplicity, and versatility in scenarios where a full-scale RDBMS is not required, such as mobile apps, browsers, and small-scale applications.



## **CHAPTER 4**

## **SYSTEM DESIGN**

## 4.1 INTRODUCTION

The development of any designed system or product begins with the design phase. A executed design is crucial, for creating a system as it involves a creative process. It involves using methods and ideas to describe a process or system, in detail so that it can be effectively implemented. Regardless of the chosen development model the design phase holds importance in the field of software engineering.

It aims to generate the specifications, for constructing a system or product and acts as the fundamental component of the software engineering procedure. This program has undergone a design process that maximizes effectiveness, performance and accuracy. During the design phase, a document that is originally intended for users is transformed into one that caters to the needs of programmers or database employees.

## 4.2 UML DIAGRAM

A standardized dialect called Unified Modelling Language (UML) is utilized to conceptualize, characterize, plan, and depict program frameworks. The Question Administration Gather (OMG) was dependable for creating UML, and the primary draft of the UML 1.0 definition was discharged in January 1997. Programming dialects like Java, C++, and COBOL are not the same as UML. It could be a nonexclusive visual demonstrating dialect utilized for computer program frameworks and a pictorial dialect utilized for program outlines. UML may be utilized for non-software frameworks, such as fabricating forms, indeed though it is generally utilized to speak to program frameworks.

- 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 representation that shows how users and other external characters interact with the components of a system. A utilize case diagram's essential work is to perceive, layout, and orchestrate a system's utilitarian needs as seen through the eyes of its clients. The Unified Modelling Language (UML), a standard language for modelling actual things and systems, is frequently used to construct use case diagrams.

Use cases have applications, in achieving system objectives. These include defining requirements, validating hardware designs, testing and debugging software creating help resources or fulfilling customer support responsibilities. Customer support, product obtaining, catalogue overhauling, and payment processing are as it were a couple of illustrations of use cases within the setting of item deals. The system boundaries, actors, use cases, and their connections together make up a use case diagram.

The system boundary defines the limits of the system, in relation, to its environment. Actors are often defined depending on the roles they play and reflect the people or systems that interact with the system. The precise activities or behaviors that actors carry out within or close to the system are known as use cases. Finally, the graphic shows the connections between actors and use cases as well as the use cases themselves.

Use case diagrams are graphical representations used to capture the functional requirements of a system. When drawing a use case diagram, it is important to follow these guidelines to ensure an efficient and effective diagram:

- Choose descriptive names for use cases that accurately reflect the functionalities they perform.
- Assign appropriate names to actors to help identify their roles in the system.
- Ensure that relationships and dependencies are clearly depicted in the diagram.
- Avoid including every possible relationship, as the main goal is to identify the essential requirements.
- Use notes when necessary to clarify important points.

By following these guidelines, we can create a clear and concise use case diagram that accurately represents the functional requirements of the system.

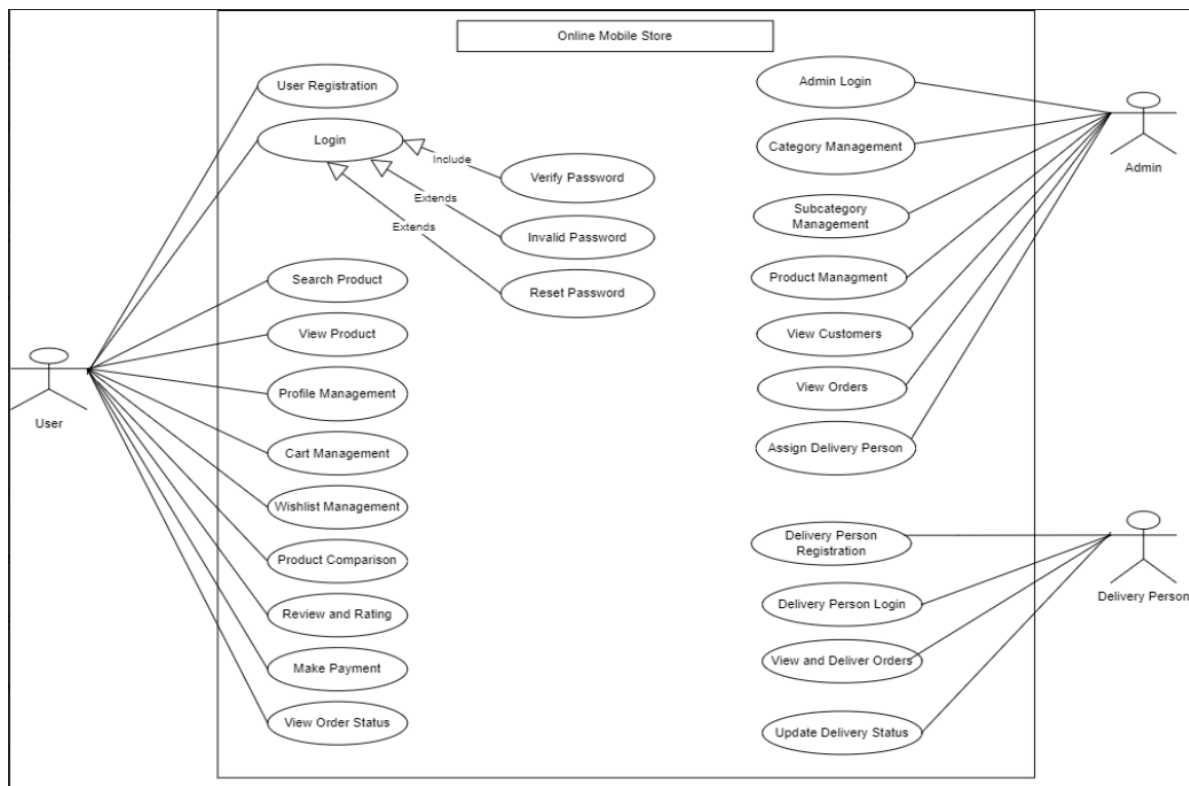


Fig 1: Use case Diagram for TechTrove Ecommerce Website

#### 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 business people and software engineers better understand and communicate system requirements and behavior.

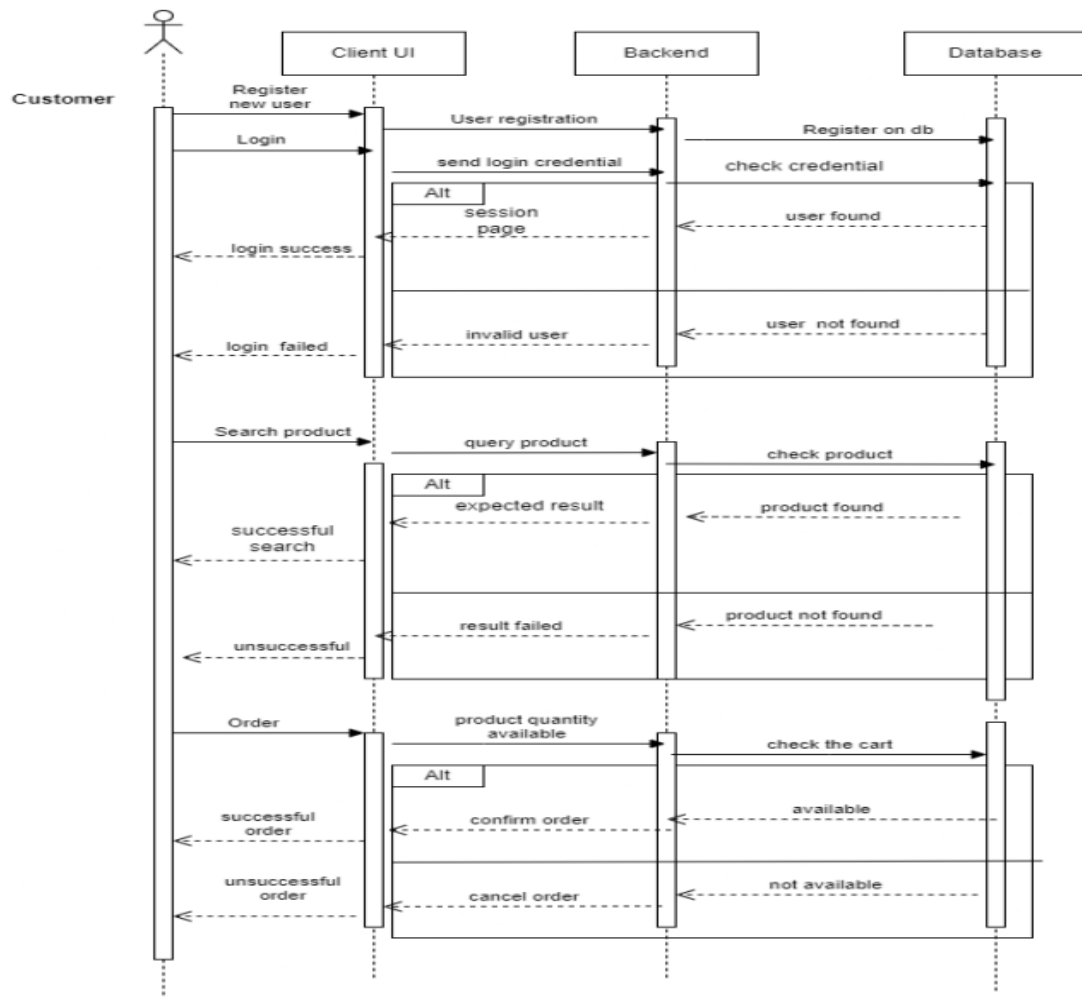


Fig 2: Sequence Diagram for TechTrove Ecommerce Website

#### 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 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.

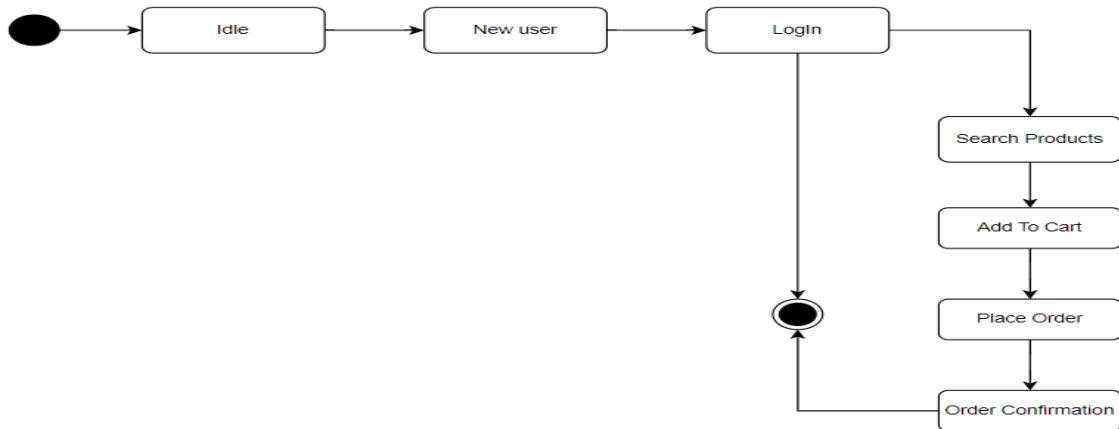


Fig: 3 State Chart Diagram for TechTrove Ecommerce Website

#### 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.

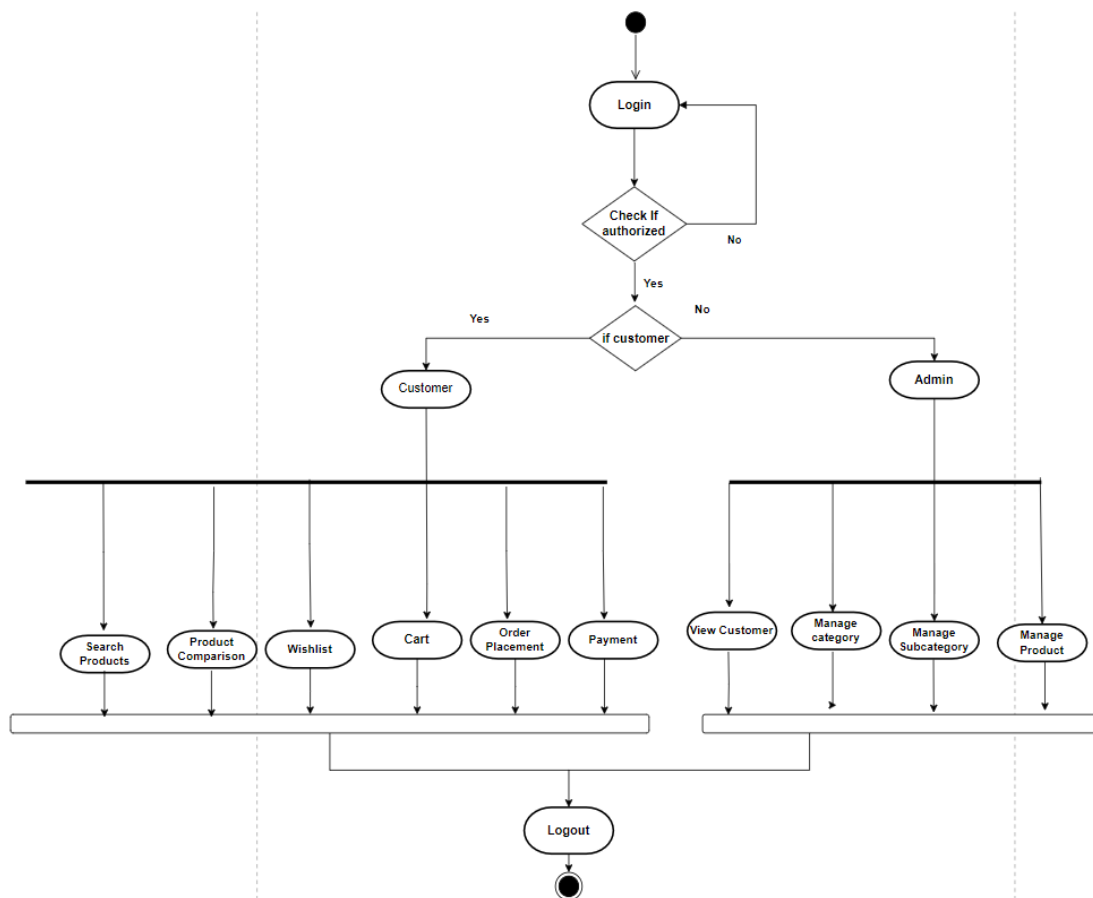


Fig :4 Activity Chart Diagram for TechTrove Ecommerce Website



### 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.

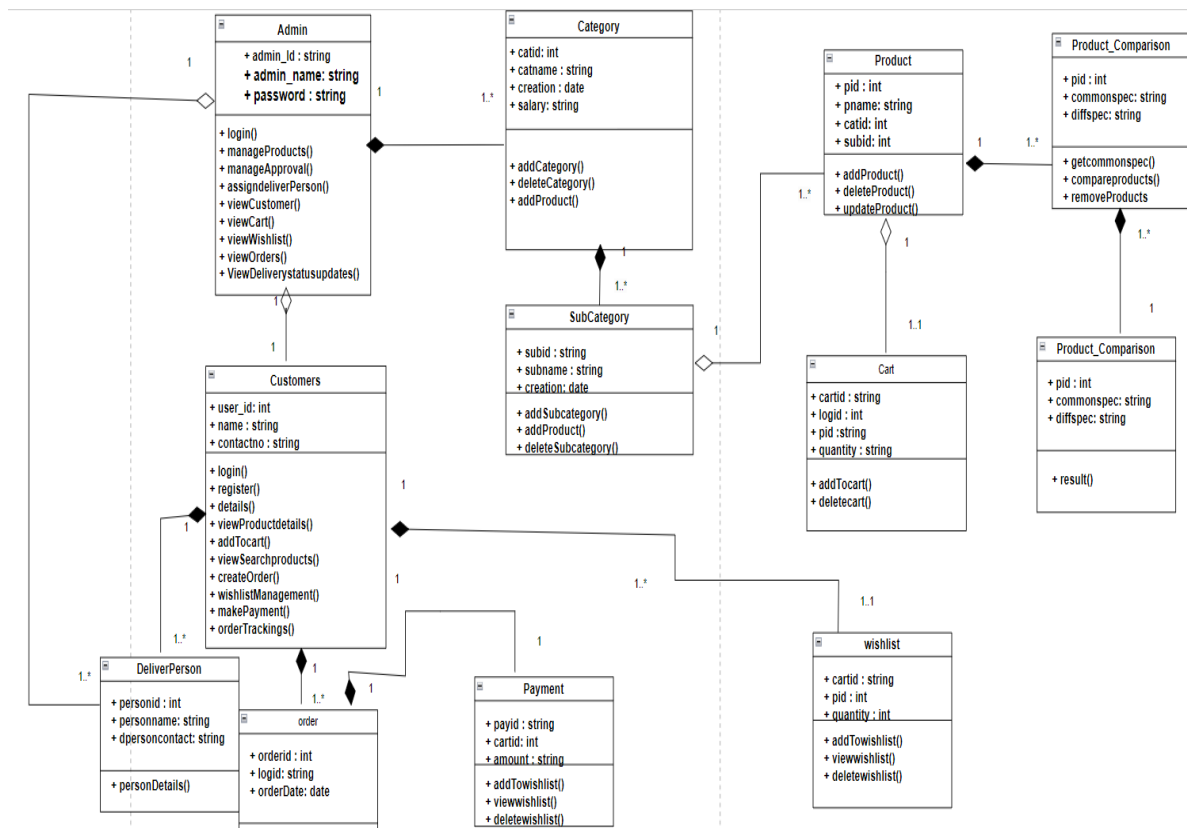


Fig:5 Class Diagram for TechTrove Ecommerce Website

#### 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.

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

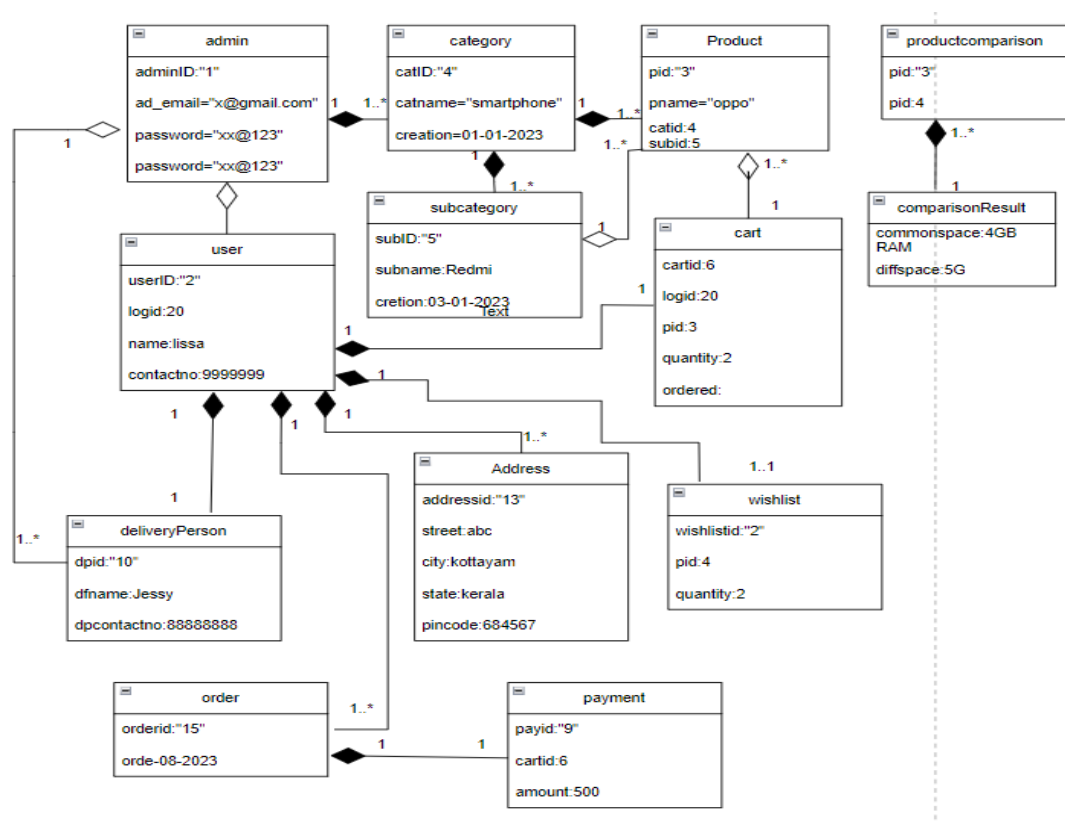


Fig:6 Object Diagram for TechTrove Ecommerce Website

#### 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.

Component diagrams are useful for modeling the architecture of a software system, and can help identify potential issues and improvements in the design. They can also be used to communicate the structure and behavior of a system to stakeholders, such as developers and project managers.

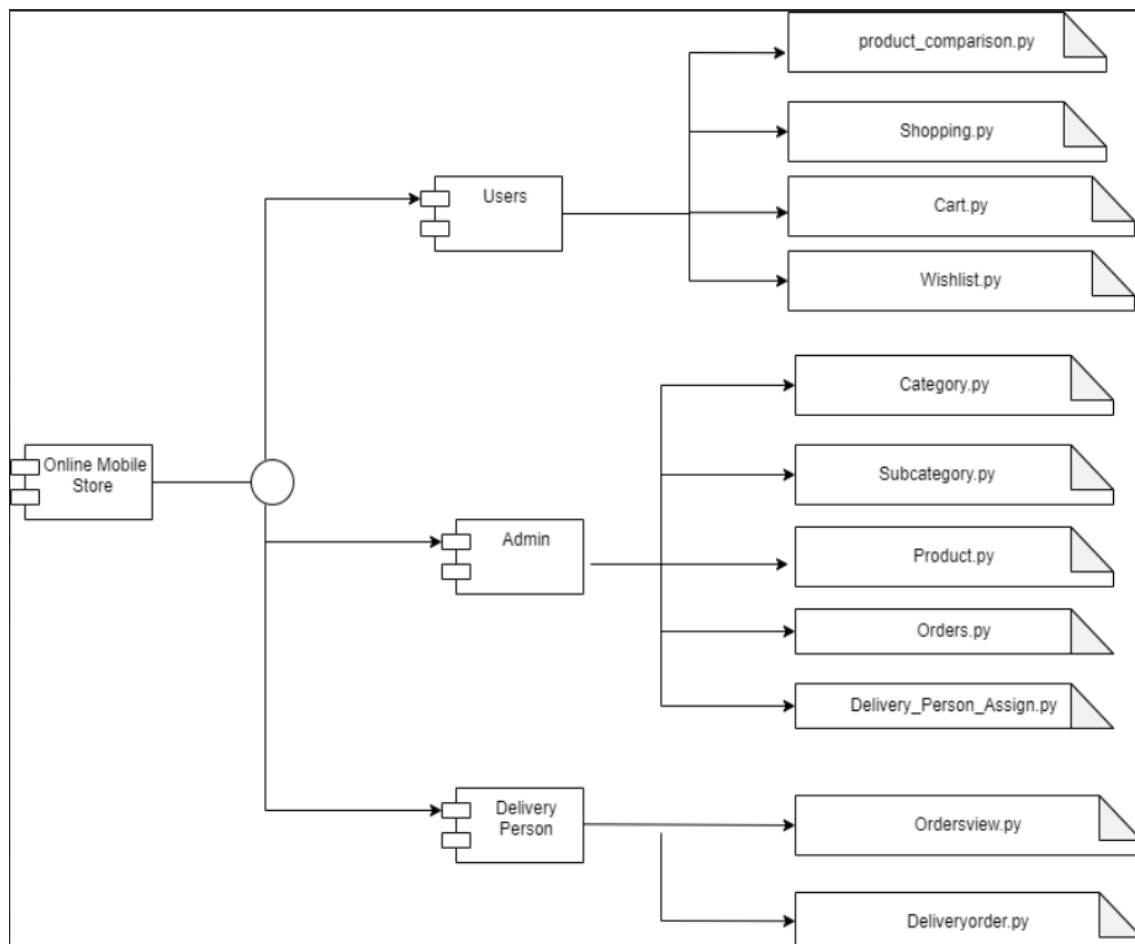


Fig:7 Component Diagram for TechTrove Ecommerce Website

#### 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.

Deployment diagrams help in visualizing the physical architecture of a system and identifying any potential issues or bottlenecks in the deployment process. They also aid in planning the deployment strategy and optimizing the use of hardware resources.

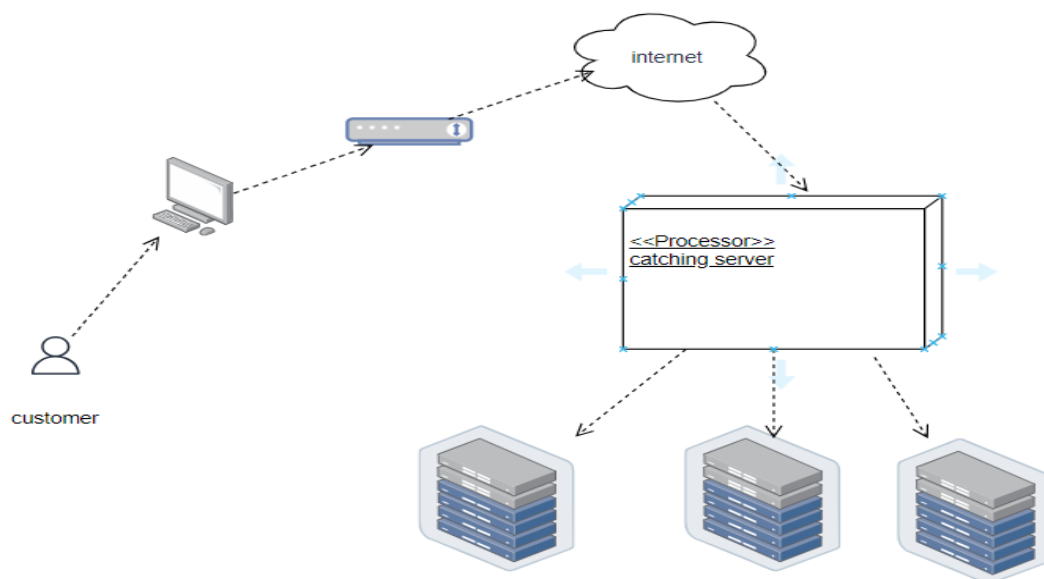
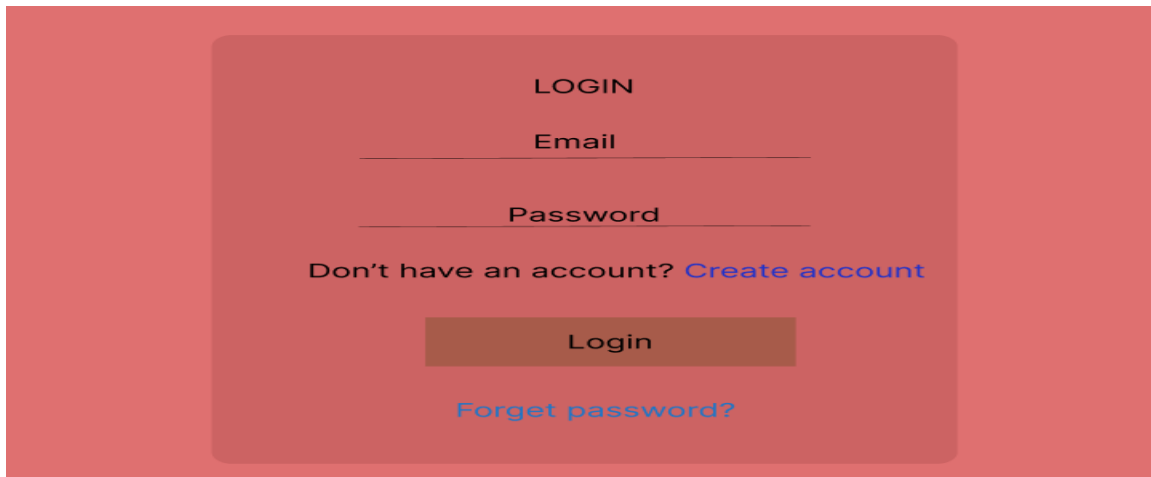


Fig: 8 Deployment Diagram for TechTrove E-commerce Website

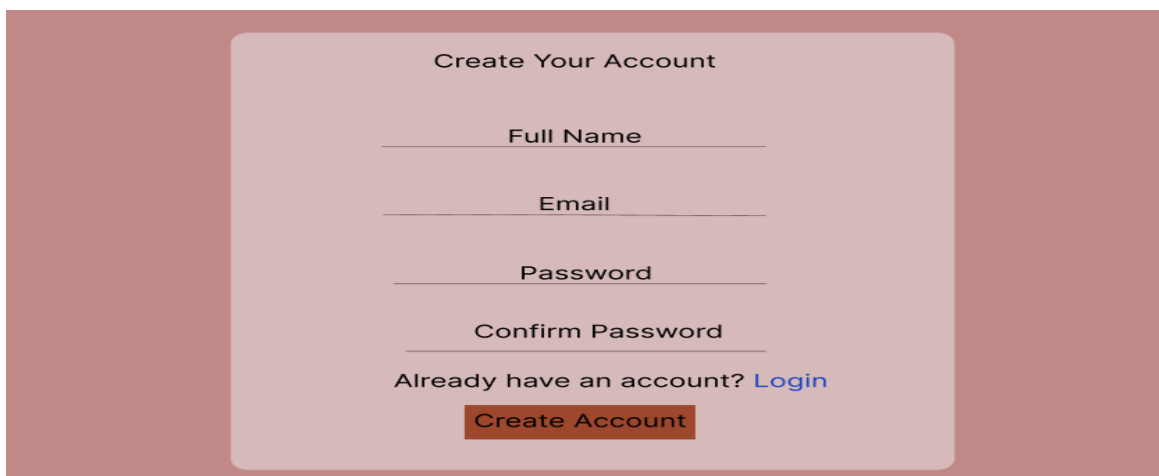
## 4.3 USER INTERFACE DESIGN USING FIGMA

### 1. Form Name: Login Form



A login form UI design on a red background. The form is a light red rounded rectangle. It contains the text "LOGIN" at the top, followed by "Email" and "Password" labels above their respective input fields. Below the fields is the text "Don't have an account? [Create account](#)". At the bottom of the form is a brown "Login" button and a blue "[Forget password?](#)" link.

### 2. Form Name: Registration Form

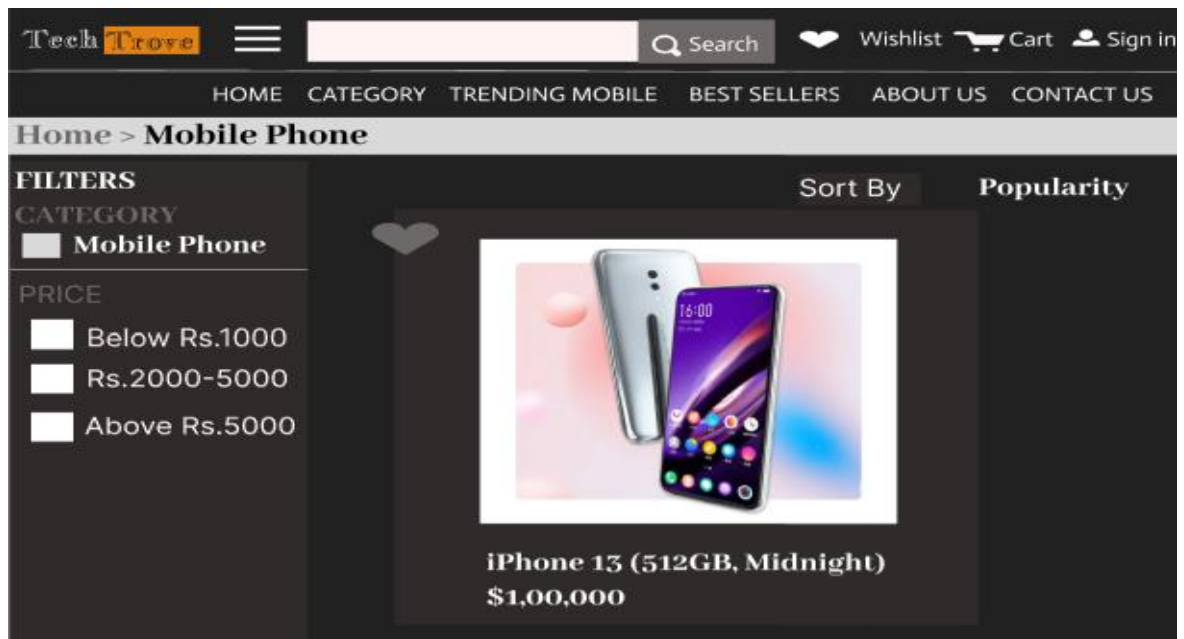


A registration form UI design on a red background. The form is a light red rounded rectangle. It contains the text "Create Your Account" at the top, followed by "Full Name", "Email", "Password", and "Confirm Password" labels above their respective input fields. Below the fields is the text "Already have an account? [Login](#)". At the bottom of the form is a brown "Create Account" button.

### 3. Form Name: Index Page



#### 4. Form Name: Home Page



#### 5. Form Name: Admin Dashboard Page





## 4.4 DATABASE DESIGN

### 4.4.1 Relational Database Management System (RDBMS)

A relational database management system (RDBMS) is software specifically designed for the storage, organization, querying, and retrieval of data within a relational database. It acts as an intermediary between users and applications and the underlying database, offering essential administrative capabilities for overseeing data storage, access, and system performance.

### 4.4.2 Normalization

Normalization is the procedure of structuring data within a database. This involves the creation of tables and the establishment of relationships between those tables, all in accordance with predefined principles. The objective is twofold: to safeguard data integrity and enhance database flexibility by eliminating redundancy and inconsistent dependencies. The normalization process consists of three primary forms:

#### First Normal Form:

- Eliminate repeating groups in individual tables.
  - Create separate tables for each set of related data.
  - Identify each set of related data with a primary key.

#### Second Normal Form:

- Create separate tables for sets of values that apply to multiple records.
- Establish relationships between these tables using a foreign key.

#### Third Normal Form:

- Eliminate fields that do not depend on the key.

#### Normal Form (3NF):

- In 3.5NF, the focus is on eliminating transitive dependencies between non-prime attributes (attributes that are not part of the primary key) in a relation.
- It ensures that there are no indirect relationships between non-prime attributes, which can lead to data anomalies.
- Achieving 3.5NF involves further breaking down tables and creating additional relationships to remove any non-prime attribute dependencies that are not directly related to the primary key.

### **3.5 Normal Form (3.5NF):**

- 3.5NF is a more advanced stage of normalization that builds upon the principles of 3 NF. This process of normalization ensures that data is structured efficiently, avoiding duplication and ensuring the database's stability and flexibility.

### **4.4.3 Sanitization**

Validation involves assessing whether input data adheres to a defined set of criteria, ensuring it meets specific requirements, while sanitization is the process of modifying input to guarantee its validity. Combining these two techniques provides a comprehensive defense for your application. For instance, you may transform all single quotation marks into double quotation marks (sanitize) and subsequently validate that all quotation marks have been successfully changed to double quotation marks. Validation checks encompass various aspects, including length, format, range, and allowable characters. For instance, in cases where an application anticipates positive integer input, validation is necessary to confirm that any string input exclusively contains the digits 0 through 9. These practices collectively bolster data quality and security within your application.

### **4.4.4 Indexing**

Indexing is a method employed to enhance the performance of a database by reducing the number of disk accesses needed during query processing. It's a data structure technique designed for swift data retrieval in a database. Indexes are constructed using specific database columns, typically comprising two main components:

1. **Search Key:** This is the first column in the index and holds a copy of the primary key or candidate key of the table. These values are organized in sorted order, facilitating rapid access to corresponding data. It's worth noting that the data may or may not be stored in sorted order.
2. **Data Reference or Pointer:** The second column contains a set of pointers that store the addresses of disk blocks where specific key values can be located. Indexes are a critical part of database optimization, allowing for efficient data retrieval and query processing.

## 4.5 TABLE DESIGN

### 1. tbl\_login

Primary key: **login\_id**

Foreign key: **login\_id** references table **tbl\_login**, **tbl\_userregistration**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	login_id	Int(10)	PRIMARY KEY	Login Id
2.	email	Varchar(20)	NULL	Email
3.	db_username	Varchar(20)	NULL	Username of delivery boy
4.	password	Varchar(20)	NOT NULL	Password
5.	role	Int (10)	NOT NULL	Role of actors
6.	status	Int (10)	NOT NULL	Status-active/blocked

### 2. tbl\_userregistration

Primary key: **userid**

Foreign key: **login\_id** references table **tbl\_login**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	userid	Int(10)	PRIMARY KEY	Users Registration Id
2.	name	Varchar(20)	NOT NULL	Users name
3.	contactno.	Varchar(20)	NOT NULL	Users contact number
4.	login_id	Int (10))	FOREIGN KEY	Login id
5.	userreg_data	Timestamp	NOT NULL	Users' registration time
6.	status	Int (10)	NOT NULL	Status-active/blocked

### 3. tbl\_category

Primary key: **cat\_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	category_id	Int(10)	PRIMARY KEY	Category Id
2.	categoryName	Varchar(20)	NOT NULL	Category Name
3.	categoryDescription	Varchar(20)	NOT NULL	Category Description
4.	creation-date	Timestamp	NOT NULL	Category Creation time
5.	updatetime	Timestamp	NOT NULL	Category Updatetime
6.	status	Int (10)	NOT NULL	Status-active/blocked

### 4. tbl\_brand

Primary key: **subcat\_id**

Foreign key: category\_id references table **tbl\_category**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	bid	Int(10)	PRIMARY KEY	Brand Id
2.	category_id	Int(10)	FOREIGN KEY	Category Id
3.	b_name	Varchar(20)	NOT NULL	Brand Name
4.	creation-date	Timestamp	NOT NULL	Brand Creation time
5.	updatetime	Timestamp	NOT NULL	Brand Updatetime
6.	status	Int (10)	NOT NULL	Status-active/blocked

## 5. tbl\_series

Primary key: **series\_id**

Foreign key: **cat\_id** references table **tbl\_category**,

**bid** reference table **tbl\_brand**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	series_id	Int(10)	PRIMARY KEY	Brand Id
2.	category_id	Int(10)	FOREIGN KEY	Category Id
3.	bid	Int(10)	FOREIGN KEY	Brand Id
4.	series_name	Varchar(20)	NOT NULL	Series Name
5.	creation-date	Timestamp	NOT NULL	Brand Creation time
6.	updation_date	Timestamp	NOT NULL	Brand Updation time
7.	status	Int (10)	NOT NULL	Status-active/blocked

## 6. tbl-order

Primary key: **order\_id**

Foreign key: **pid** references table **tbl\_product**,

**userid** references table **tbl\_userregistration**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	order_id	Int(10)	PRIMARY KEY	Order Id
2.	userid	Int(10)	FOREIGN KEY	Users Registration Id
3.	p_id	Int(10)	FOREIGN KEY	Product Id
4.	amount	Int (100)	NOT NULL	Amount
5.	creation-date	Timestamp	NOT NULL	Brand Creation time
6.	updation_date	Timestamp	NOT NULL	Brand Updation time
7.	status	Int (10)	NOT NULL	Status-active/blocked
8.	item_quantity	Int (10)	NOT NULL	Quantity of Item
9.	order_date	Timestamp	NOT NULL	Order Date

## 7. tbl\_product

Primary key: **pid**

Foreign key: **series\_id** references table **tbl\_series**,

**bid** references table **tbl\_brand**,

**category\_id** references table **tbl\_category**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	p_id	Int(10)	PRIMARY KEY	Primary key
2.	pname	Varchar(20)	NOT NULL	Product Name
3.	series_id	Int(10)	FOREIGN KEY	Series Id
4.	bid	Int(10)	FOREIGN KEY	Brand Id
5.	category_id	Int(10)	FOREIGN KEY	Category Id
6.	quantity	Int(10)	NOT NULL	Quantity of product
7.	status	Int (10)	NOT NULL	Status-active/blocked

## 8. tbl\_image

Primary key: **img\_id**

Foreign key: **pid** references table **tbl\_product**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	img_id	Int(10)	PRIMARY KEY	Image Id
2.	pid	Int(10)	FOREIGN KEY	Product Id
3.	img1	varchar (225)	NOT NULL	Image 1
4.	img2	varchar (225)	NOT NULL	Image 2
5.	img3	varchar (225)	NOT NULL	Image 3

## 9. tbl\_cart

Primary key: **cart\_id**

Foreign key: **pid** references table **tbl\_product**,

**pid** references table **tbl\_product**,

**login\_id** references table **tbl\_login**,

**orderid** references table **tbl\_orders**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	Cart_id	Int(10)	PRIMARY KEY	Cart Id
2.	pid	Int(10)	FOREIGN KEY	Product Id
3.	login_id	Int(10)	FOREIGN KEY	Login Id
4.	orderid	Int(10)	FOREIGN KEY	Order Id
5.	date	Timestamp	NOT NULL	Date of product added to cart
6.	quantity	Int(10)	NOT NULL	No: of quantity added to cart

## 10. tbl\_wishlist

Primary key: **wish\_id**

Foreign Key: **pid** references table **tbl\_product**,

**login\_id** references table **tbl\_login**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	wish_id	Int(10)	PRIMARY KEY	Wishlist Id
2.	pid	Int(10)	FOREIGN KEY	Product Id
3.	login_id	Int(10)	FOREIGN KEY	Login Id
4.	Status	Varchar (30)	NOT NULL	Status

## **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.

### 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 anytime 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. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. 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

### Code:

```
package stepDef;
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 testingLog {
    WebDriver driver=null;
    @Given("browser is open")
    public void browser_is_open() {
        System.out.println("Inside step browser");

System.out.printf("webdriver.gecko.marionette", "D:\\sample1\\sample1\\src\\test\\resources\\driver\\geckodriver.exe");
        driver=new FirefoxDriver();
    }
    @And("user is on login page")
    public void user_is_on_login_page() throws Exception {

        driver.navigate().to("http://127.0.0.1:8000/userlogin/");
    }
    @When("user enter username and password")
    public void user_enter_username_and_password() throws Exception {

        driver.findElement(By.id("username")).sendKeys("manjarijayan2001@gmail.com");
        driver.findElement(By.id("password")).sendKeys("Manjari@123");
    }
    @And("user clicks on login")
    public void user_clicks_on_login() throws Exception {

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

        driver.findElement(By.className("container d-flex align-items-center justify-content-lg-between")).isDisplayed();
        Thread.sleep(2000);
        driver.close();
        driver.quit();
    }
}
```

**Screenshot:**

```

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
And user is on login page
When 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
1698155877058 RemoteAgent INFO Perform WebSocket upgrade for incoming connection from 127.0.0.1:50655

```

**Test Report**

Test Case 1					
Project Name: TechTrove					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Manjari Jayan		
Test Priority(Low/Medium/High): High			Test Designed Date: 18/09/2023		
Module Name: Login Screen			Test Executed By : Ms. Jetty Benjamin		
Test Title : User Login			Test Execution Date: 18/09/2023		
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		Home page should be displayed	Login page displayed	Pass
2	Provide Valid Email	Email: manjarijayan2024@mca.aaje.in	User should be able to Login	User Logged in and navigated to Home Page	Pass
3	Provide Valid Password	Manjari @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 stepDef;
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 testingLog {
        WebDriver driver=null;
        @Given("browser is open")
        public void browser_is_open() {
            System.out.println("Inside step browser");

System.out.printf("webdriver.gecko.marionette","D:\\sample1\\sample1\\src\\test\\resources\\driver\\geckodriver.exe");
            driver=new FirefoxDriver();
        }
        @And("admin is on login page")
        public void admin_is_on_login_page() throws Exception {

            driver.navigate().to("http://127.0.0.1:8000/userlogin/");
        }
        @When("admin enter username and password")
        public void admin_enter_username_and_password() throws Exception {

            driver.findElement(By.id("username")).sendKeys("mj");
            driver.findElement(By.id("password")).sendKeys("mj");
            driver.findElement(By.id("submit")).click();
            Thread.sleep(3000);
        }
        @And("admin navigates to the admin dashboard page")
        public void admin_navigates_to_the_admin_page()throws Exception
        {

            Thread.sleep(3000);
        }
        @And("admin clicks on the category ")
        public void admin_clicks_on_the_manage_button()throws Exception {
            // Click on the category
            driver.findElement(By.xpath("//*[@id=\"navbardrop\"]")).click();
            Thread.sleep(3000);
        }
        @And("admin selects Add Category from the dropdown")
        public void admin_selects_category_from_the_dropdown() throws Exception
        {
            driver.findElement(By.xpath("//*[@id=\"sidebar\"]/ul/li[3]/div/a[1]")).click();
            Thread.sleep(3000);
        }
    }
}
```

```

    }
    @And("admin navigates to the add category page")
    public void user_navigates_to_the_add_category_page() throws Exception
    {
        Thread.sleep(3000);
    }
    @And("admin enters category details")
    public void user_enters_category_details() throws Exception
    {
        // Enter category details
        driver.findElement(By.name("name")).sendKeys("Smart Phones");
        Thread.sleep(3000);
        // Add any other details related to category creation
    }
    @And("admin clicks on the submit button")
    public void user_clicks_on_the_add_category_button() {
        // Click on the "Add category" button
        driver.findElement(By.cssSelector("#content > div > div > div > form > input.m-2.px-3.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.xpath("//*[@id='sidebar']/ul/li[3]/div/a[2]")).isDisplayed();
    }
}

```

## Screenshot:

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: TechTrove	
Add Category Test Case	
Test Case ID: Test_2	Test Designed By: Manjari Jayan
Test Priority(Low/Medium/High): High	Test Designed Date: 18/09/2023



<b>Module Name:</b> Add Category Screen			<b>Test Executed By :</b> Ms. Jetty Benjamin		
<b>Test Title :</b> Admin Add Catgeory			<b>Test Execution Date:</b> 18/09/2023		
<b>Description:</b> Add Catgeory					
<b>Pre-Condition :</b> User has valid username and password					
Step	Test Step	Test Data	ExpectedResult	Actual Result	Status(Pass/ Fail)
1	Navigation to Login Page Login button		Dashboar d should be displayed	Dashboard displayed	Pass
2	Provide Valid Email	Email: mj	Admin should be able to Login	Admin Logged in and navigated to Home Page	Pass
3	Provide Valid Password	mj			
4	Click on Login button				
5	Click on Catgeory Dropdown		Add Category and View Category Options should be displayed	Add Category and View Category Options displayed	Pass
6	Select Add Category		Add Category page should be displayed	Add Category page displayed	Pass
<b>Post-Condition:</b> Admin is validated with database and successfully login into account. The Account session details are logged in database. Admin has accessed the Add Category page, and the Category options have been displayed					

### Test Case 3: Admin View Category

```

public class testingLog {
    WebDriver driver=null;
    @Given("browser is open")
    public void browser_is_open() {
        System.out.println("Inside step browser");

        System.out.printf("webdriver.gecko.marionette", "D:\\sample1\\sample1\\src\\test\\resources\\driver\\geckodriver.exe");
    }
}

```

```

driver=new FirefoxDriver();
}
@And("admin is on login page")
public void admin_is_on_login_page() throws Exception {

    driver.navigate().to("http://127.0.0.1:8000/userlogin/");
}
@When("admin enter username and password")
public void admin_enter_username_and_password() throws Exception {

    driver.findElement(By.id("username")).sendKeys("mj");
    driver.findElement(By.id("password")).sendKeys("mj");
    driver.findElement(By.id("submit")).click();
    Thread.sleep(3000);
}
@And("admin navigates to the admin dashboard page")
public void admin_navigates_to_the_admin_page()throws Exception
{

    Thread.sleep(3000);
}
@And("admin clicks on the brand ")
public void admin_clicks_on_the_category_dropdown()throws Exception {
// Click on the category
driver.findElement(By.xpath("//*[@id=\"sidebar\"]/ul/li[4]")).click();
Thread.sleep(3000);
}
@And("admin selects view category from the dropdown")
public void admin_selects_viewcategory_from_the_dropdown() throws Exception
{
driver.findElement(By.xpath("//*[@id=\"\"navbardrop\"]")).click();
Thread.sleep(3000);
}
@Then("admin navigates to the view category page")
public void user_navigates_to_the_view_category_page() throws Exception
{
driver.findElement(By.xpath("//*[@id=\"example\"]tbody/tr[1]")).click();
Thread.sleep(3000);
}
}

```

### Screenshot:

```

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 3					
Project Name: TechTrove					
View Category Test Case					
Test Case ID: Test_3			Test Designed By: Manjari Jayan		
Test Priority(Low/Medium/High): High			Test Designed Date: 18/09/2023		
Module Name: View Category Screen			Test Executed By : Ms. Jetty Benjamin		
Test Title : Admin View Catgeory			Test Execution Date: 18/09/2023		
Description: View Catgeory					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	ExpectedResult	Actual Result	Status(Pass/ Fail)
1	Navigation to Login Page Login button		Dashboard should be displayed	Dashboard displayed	Pass
2	Provide Valid Email	Email: mj	Admin should be able to Login	Admin Logged in and navigated to Home Page	Pass
3	Provide Valid Password	mj			
4	Click on Login button				
5	Click on Catgeory Dropdown		Add Category and View Category Options should be displayed	Add Category and View Category Options displayed	Pass
6	Select View Category		View Category page should be displayed	View Category page displayed	Pass
Post-Condition: Admin is validated with database and successfully login into account. The Account session details are logged in database. Admin has accessed the Add Category page, and the Category options have been displayed					

**Test Case 4: Wishlist Page**

```
public class wishlist {  
    WebDriver driver=null;  
    @Given("browser is open")  
    public void browser_is_open() {  
        System.out.println("Inside step browser");  
  
        System.out.printf("webdriver.gecko.marionette","D:\\sample1\\sample1\\src\\test\\resources\\  
\\driver\\geckodriver.exe");  
  
        driver=new FirefoxDriver();  
    }  
    @And("user is on login page")  
    public void user_is_on_login_page() throws Exception {  
        driver.navigate().to("http://127.0.0.1:8000/userlogin/");  
    }  
    @When("user enter username and password1")  
    public void user_enter_username_and_password() throws Exception {  
  
        driver.findElement(By.id("username")).sendKeys("manjarijayan2024@mca.ajce.in");  
        driver.findElement(By.id("password")).sendKeys("Manjari@1234");  
    }  
    @And("user clicks on login")  
    public void user_clicks_on_login() throws Exception {  
  
        driver.findElement(By.id("submit")).click();  
    }  
  
    @Then("user is navigated to the home page")  
    public void user_is_navigated_to_the_home_page() throws Exception {  
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10)); //  
Wait up to 10 seconds  
        By elementLocator = By.id("wi");
```

```

        // Wait for the element to be present

        WebElement element =
wait.until(ExpectedConditions.presenceOfElementLocated(elementLocator));

        // Perform the action (e.g., click on the element)

        element.click();

    }

}

```

## Test Report

Test Case 4					
Project Name: TechTrove					
View Category Test Case					
Test Case ID: Test_4			Test Designed By: Manjari Jayan		
Test Priority(Low/Medium/High): High			Test Designed Date: 18/09/2023		
Module Name: Wishlist Screen			Test Executed By : Ms. Jett Benjamin		
Test Title : User View Wishlist			Test Execution Date: 18/09/2023		
Description: View Wishlist					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Navigation to Login Page Login button		Dashboard should be displayed	Dashboard displayed	Pass
2	Provide Valid Email	Email: mj	Home Page should be able to Login	User Logged in and navigated to Home Page	Pass
3	Provide Valid Password	mj			
4	Click on Login button				

5	Click on		Wishlist should be displayed	Wishlist displayed	Pass
<b>Post-Condition:</b> User is validated with database and successfully login into account. The Account session details are logged in database. User has accessed the Wishlist page.					

## **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. System personnel evaluate the feasibility of the system. Implementation requires extensive effort in three main areas: education and training, system testing, and changeover. 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 mobile e-commerce website represent a significant milestone in our commitment to providing customers with a convenient, secure, and efficient platform for their mobile device shopping needs. The project's core objectives revolved around creating a user-friendly interface, offering a diverse product range, ensuring secure payment processing, providing personalized shopping experiences, optimizing order fulfillment, and establishing robust customer support. We are proud to report that these objectives have been successfully achieved, and our website is poised to deliver a seamless shopping experience for mobile device enthusiasts.

One of our standout achievements is the design of a user-friendly interface that centers on enhancing the customer's journey. The website's layout is clean and intuitive, making the process of discovering and selecting mobile devices straightforward. With a focus on clear product categorization and an efficient search function, customers can easily find the mobile devices they seek. We've included comprehensive product descriptions, high-quality images, and user reviews to empower customers to make well-informed purchasing decisions.

Security is paramount in the realm of mobile e-commerce, and we have dedicated significant resources to implement robust security measures. Our website integrates secure payment gateways and encryption protocols to protect customer data and financial transactions. This ensures that customers can shop with confidence, knowing that their sensitive information is always safeguarded.

We've harnessed the potential of data analytics and machine learning to offer a unique and personalized shopping experience. Through a sophisticated recommendation system, customers receive tailored product suggestions based on their browsing and purchasing history. This not only enhances the shopping experience but also serves as a potent tool for boosting sales and fostering customer loyalty.

Efficiency is crucial for success in mobile e-commerce. To this end, our project has seamlessly integrated the order fulfillment process with an advanced inventory management system. This guarantees that customers' orders are processed swiftly and accurately, resulting in timely deliveries and a reduced risk of product unavailability, ultimately enhancing customer

---

satisfaction.

## **7.2 FUTURE SCOPE**

The future scope for TechTrove, our online mobile shopping platform, is brimming with potential for further growth and enhancement. As we move forward, we envision the continuous expansion of our product offerings, including an even wider selection of mobile devices from various brands and categories to cater to the evolving preferences of our customers. Our commitment to user satisfaction and convenience will drive the implementation of advanced features such as voice search, translation capabilities, and AI-powered ChatBot assistance to make the shopping experience more personalized and seamless.

Furthermore, we aim to stay at the forefront of responsive design, ensuring a consistent and optimized user experience across all devices. On the administrative front, we will explore the integration of advanced analytics tools to gain deeper insights into customer behavior and preferences. These insights will inform our efforts to further refine product listings and enhance the overall user experience. The future of TechTrove is marked by a dedication to innovation, adaptability, and an unwavering focus on providing a cutting-edge mobile shopping experience for our valued users.

## **CHAPTER 8**

## **BIBLIOGRAPHY**

**REFERENCES:**

- Roger S Pressman, “Software Engineering”
- Ken Schwaber, Mike Beedle, Agile Software Development with Scrum\_, Pearson(2008)
- PankajJalote, “Software engineering: a precise approach”
- IEEE Std 1016 Recommended Practice for Software Design Descriptions

**WEBSITES:**

- [www.w3schools.com](http://www.w3schools.com)
- [www.bootstrap.com](http://www.bootstrap.com)
- <https://www.geeksforgeeks.org/>
- [www.guru99.com](http://www.guru99.com)

## **CHAPTER 9**

## **APPENDIX**

## 9.1 Sample Code

### Registration Page

```
def registration(request):
    if request.method == "POST":
        fname = request.POST['fname']
        lname = request.POST['lname']
        email = request.POST['email']
        password = request.POST['password']
        address = request.POST['address']
        mobile = request.POST['mobile']
        image = request.FILES['image']

        # Check if a user with the same email already exists
        if User.objects.filter(email=email).exists():
            messages.error(request, "A user with this email already exists.")
        else:
            # Create a new user if the email is unique
            user = User.objects.create_user(username=email, first_name=fname,
last_name=lname, email=email, password=password)
            UserProfile.objects.create(user=user, mobile=mobile, address=address,
image=image)
            messages.success(request, "Registration Successful")
            return redirect('userlogin')
    return render(request, 'registration.html', locals())
```

### Login Page

```
@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def userlogin(request):
    if request.user.is_authenticated:
        return redirect('home')
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user:
            if(user.is_staff):
                login(request, user)
                messages.success(request, "User login successfully")
                return redirect('admindashboard')
            else:
                login(request, user)
                return redirect('home')
        else:
            messages.success(request, "Invalid Credentials")
    return render(request, 'login.html', locals())
```



### Home Page

```
@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def home(request):
    allcategory = Category.objects.all()
    nested_categories = []
    for category in allcategory:
        category_dict = {'category': category, 'subcategories': []}

        subcategories = Subcategory.objects.filter(category=category)

        for subcategory in subcategories:
            subcategory_dict = {'subcategory': subcategory, 'series': []}

            # Retrieve series for this subcategory
            series = Series.objects.filter(brand=subcategory)

            for s in series:
                subcategory_dict['series'].append(s)

            category_dict['subcategories'].append(subcategory_dict)
        nested_categories.append(category_dict)

    for c in nested_categories:
        for s in c.values():
            print(s)
    products = Product.objects.all()
    return render(request, 'home.html', {'nested_categories': nested_categories, 'products':
products})
```

### Profile Page

```
@login_required
def profile(request):
    user = request.user
    try:
        user_profile = UserProfile.objects.get(user=user)
    except UserProfile.DoesNotExist:
        # If UserProfile doesn't exist, create a new one for the user
        user_profile = UserProfile(user=user)
        user_profile.save()

    if request.method == "POST":
        fname = request.POST.get('fname')
        lname = request.POST.get('lname')

        mobile = request.POST.get('mobile')
        address = request.POST.get('address')
        image = request.FILES.get('image')
        if fname:
```

```
        user.first_name = fname
    if lname:
        user.last_name = lname
    if mobile:
        user_profile.mobile = mobile
    if address:
        user_profile.address = address
    if image:
        user_profile.image = image

    # Save changes to user and user profile
    user.save()
    user_profile.save()

    messages.success(request, "Profile updated")
    return redirect('profile')

return render(request, 'profile.html', {'user_profile': user_profile, 'user': user})
```

### Product Details Page

```
def product_detail(request, pid):
    try:
        product = Product.objects.get(id=pid)
        latest_product = Product.objects.filter().exclude(id=pid).order_by('-id')[:10]
        allcategory = Category.objects.all()
        nested_categories = []
        for category in allcategory:
            category_dict = {'category': category, 'subcategories': []}

            subcategories = Subcategory.objects.filter(category=category)

            for subcategory in subcategories:
                subcategory_dict = {'subcategory': subcategory, 'series': []}

                # Retrieve series for this subcategory
                series = Series.objects.filter(branch=subcategory)

                for s in series:
                    subcategory_dict['series'].append(s)

                category_dict['subcategories'].append(subcategory_dict)
            nested_categories.append(category_dict)

        for c in nested_categories:
            for s in c.values():
                print(s)
        products = Product.objects.all()
        in_stock = product.stock > 0
```

```
if request.method == "POST" and in_stock:
    quantity = int(request.POST.get('qty', 1))
    if quantity > 0 and quantity <= product.stock:
        # Decrease the stock count
        product.stock -= quantity
        product.save()

    return render(request, "product_detail.html")
else:
    # Handle the case where the quantity is invalid or exceeds the stock.
    return render(request, "product_detail.html", {
        'product': product,
        'latest_product': latest_product,
        'nested_categories': nested_categories,
        'in_stock': in_stock,
        'error_message': "Invalid quantity or insufficient stock.",
    })

context = {
    'product': product,
    'latest_product': latest_product,
    'nested_categories': nested_categories,
    'in_stock': in_stock,
}

return render(request, "product_detail.html", context)

except Product.DoesNotExist:
    return HttpResponse("Product not found", status=404)
```

### Wishlist Page

```
@login_required
def view_wishlist(request):
    if request.user.is_authenticated:
        wishlist_items = WishlistItem.objects.filter(user=request.user)

    # Retrieve the categories and nested categories again for the wishlist page
    allcategory = Category.objects.all()
    nested_categories = []
    for category in allcategory:
        category_dict = {'category': category, 'subcategories': []}

        subcategories = Subcategory.objects.filter(category=category)

        for subcategory in subcategories:
```

```
        subcategory_dict = {'subcategory': subcategory, 'series': []}

        # Retrieve series for this subcategory
        series = Series.objects.filter(brand=subcategory)

        for s in series:
            subcategory_dict['series'].append(s)

        category_dict['subcategories'].append(subcategory_dict)
        nested_categories.append(category_dict)

    for c in nested_categories:
        for s in c.values():
            print(s)
    products = Product.objects.all()
    return render(request, 'wishlist.html', {
        'wishlist_items': wishlist_items,
        'nested_categories': nested_categories, # Pass the categories to the wishlist template
    })
else:

    return redirect('login')
```

### Product Comparison Page

```
@login_required
def product_comparison(request):
    try:
        # comparison_list = ComparisonList.objects.get(id=comparison_id)
        user = User.objects.get(id=request.user.id)
        products_to_compare = ComparisonList.objects.all().filter(user=user)
        products = []
        allcategory = Category.objects.all()
        nested_categories = []
        for category in allcategory:
            category_dict = {'category': category, 'subcategories': []}

            subcategories = Subcategory.objects.filter(category=category)

            for subcategory in subcategories:
                subcategory_dict = {'subcategory': subcategory, 'series': []}

                # Retrieve series for this subcategory
                series = Series.objects.filter(brand=subcategory)

                for s in series:
                    subcategory_dict['series'].append(s)

                category_dict['subcategories'].append(subcategory_dict)
            nested_categories.append(category_dict)
```

```
# Loop through the comparison_lists and add the products to the products list
for comparison_list in products_to_compare:
    products.extend(comparison_list.products.all())
print(products)
# Debugging: Print the products related to the comparison list
print('Products in Comparison :', products_to_compare.values())
    return render(request, 'product_comparison.html', {'selectedProducts':
products,'nested_categories': nested_categories})
except ComparisonList.DoesNotExist:
    # Handle the case when the comparison list does not exist
    # Redirect or display an error message as needed
    return HttpResponseRedirect('Comparison list not found')
```

### Add Category Page

```
def add_category(request):
    if request.method == "POST":
        name = request.POST['name']

        # Check if the category already exists
        if Category.objects.filter(name=name).exists():
            category_exist_msg = "Category already exists"
        else:
            # Category doesn't exist, create it
            Category.objects.create(name=name)
            category_added_msg = "Category added successfully"

    return render(request, 'add_category.html', locals())
```

### Add Brand Page

```
def brand_product(request,pid):

    subcategory = Subcategory.objects.get(id=pid)
    product = Product.objects.filter(brand=subcategory)

    allcategory = Category.objects.all()
    nested_categories = []
    for category in allcategory:
        category_dict = {'category': category, 'subcategories': []}

        subcategories = Subcategory.objects.filter(category=category)

        for subcategory in subcategories:
            subcategory_dict = {'subcategory': subcategory, 'series': []}

            # Retrieve series for this subcategory
            series = Series.objects.filter(brand=subcategory)
```

```
        for s in series:
            subcategory_dict['series'].append(s)

        category_dict['subcategories'].append(subcategory_dict)
    nested_categories.append(category_dict)
    products = Product.objects.all()
    return render(request, "brand.html", locals())
```

### Add Product Page

```
def add_product(request):
    category = Category.objects.all()
    brand = Subcategory.objects.all()
    series = Series.objects.all()

    if request.method == "POST":
        name = request.POST['name']
        price = request.POST['price']
        cat = request.POST['category']
        brandid = request.POST['brand']
        seriesid = request.POST['series']
        discount = request.POST['discount']
        desc = request.POST['desc']
        stock = request.POST['stock']
        image = request.FILES['image']

        # New fields
        processor = request.POST['processor']
        front_camera = request.POST['front_camera']
        back_camera = request.POST['back_camera']
        ram = request.POST['ram']
        rom = request.POST['rom']

        catobj = Category.objects.get(id=cat)
        brandobj = Subcategory.objects.get(id=brandid)
        seriesobj = Series.objects.get(id=seriesid)

        # Create a new Product object with the additional fields
        product = Product.objects.create(
            name=name,
            price=price,
            discount=discount,
            category=catobj,
            brand=brandobj,
            series=seriesobj,
            description=desc,
            image=image,
            stock=stock,
            processor=processor,
            front_camera=front_camera,
```

```
        back_camera=back_camera,  
        ram=ram,  
        rom=rom  
    )  
  
    messages.success(request, "Product added")  
  
    return render(request, 'add_product.html', locals())
```

## 9.1 Screen Shots

### Registration Page


#### Create an Account

First Name	Last Name
<input type="text" value="Enter First Name"/>	<input type="text" value="Enter Last Name"/>
Email ID	Contact
<input type="text" value="Email Address"/>	<input type="text" value="Contact Number"/>
Password	Confirm Password
<input type="text" value="Enter Password"/>	<input type="text" value="Enter Password Again"/>
Upload PIC	
<input type="button" value="Choose File"/> No file chosen	
Address	
<input type="text" value="Full Address"/>	
<input type="button" value="Submit"/>	

### Login Page

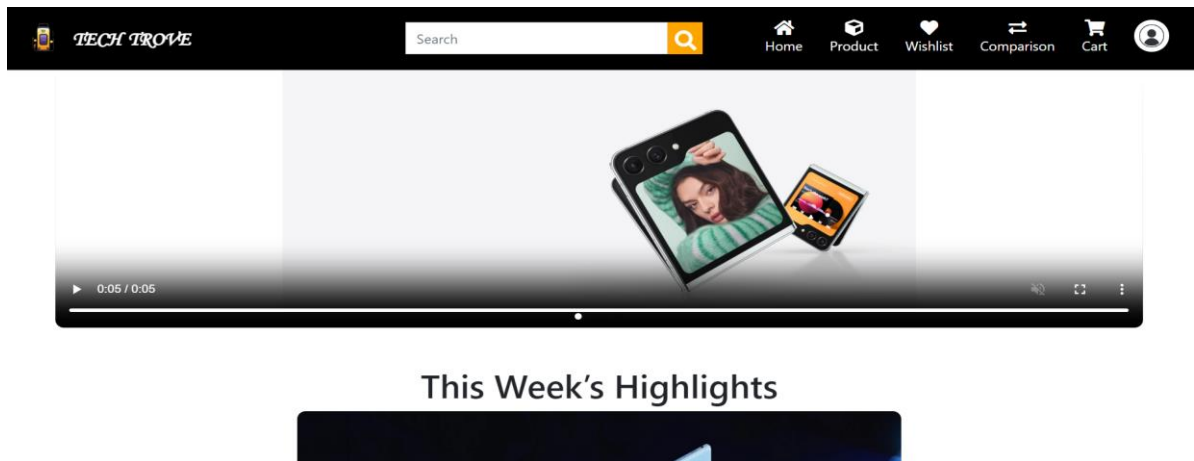
 **TECH TROVE**

[Home](#) [Product](#) [Cart](#) [Wishlist](#) [Comparison](#) [Login](#)

  
**User Login**

[Create Account >>](#)

### Home Page




## Profile Page

**My Profile**

First Name:  Last Name:

Contact:

Address:

User PIC:  No file chosen 

[Edit](#)

## Product Detail Page



### Nokia 130 Music

M.R.P. : 2499- **Rs. 1759.2**

Add Quantity

Description About this product:-

Built-in Powerful Loud Speaker

**Processor:** one-core

**Front Camera:** No

**Back Camera:** 0.3 MP

**Ram:** 4MB

**Rom:** 4GB

Stock left: 10

[Buy Now](#)

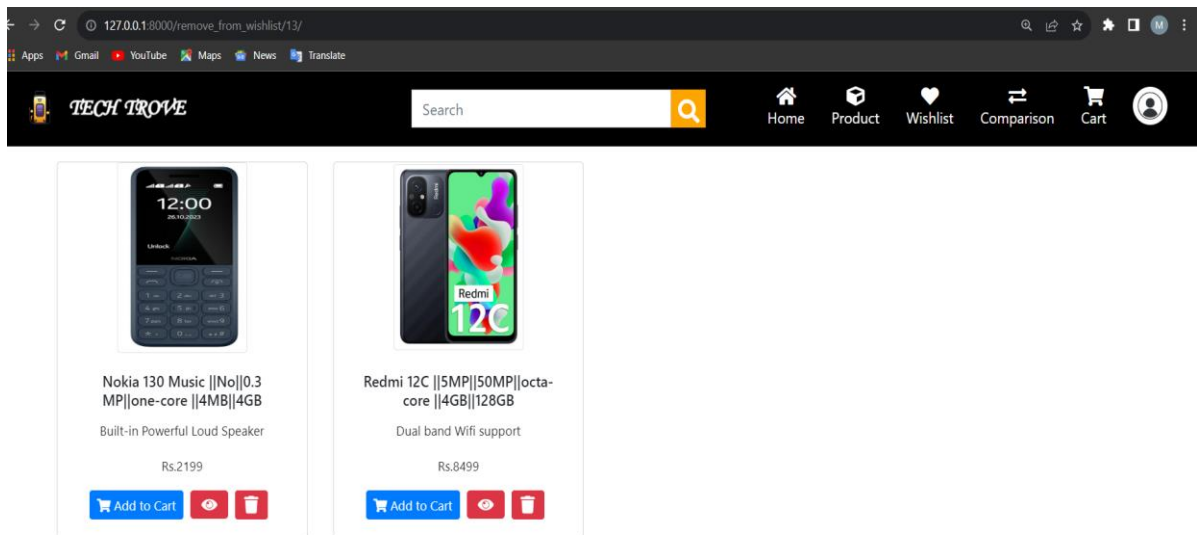
[Add to cart](#)

[Compare](#)

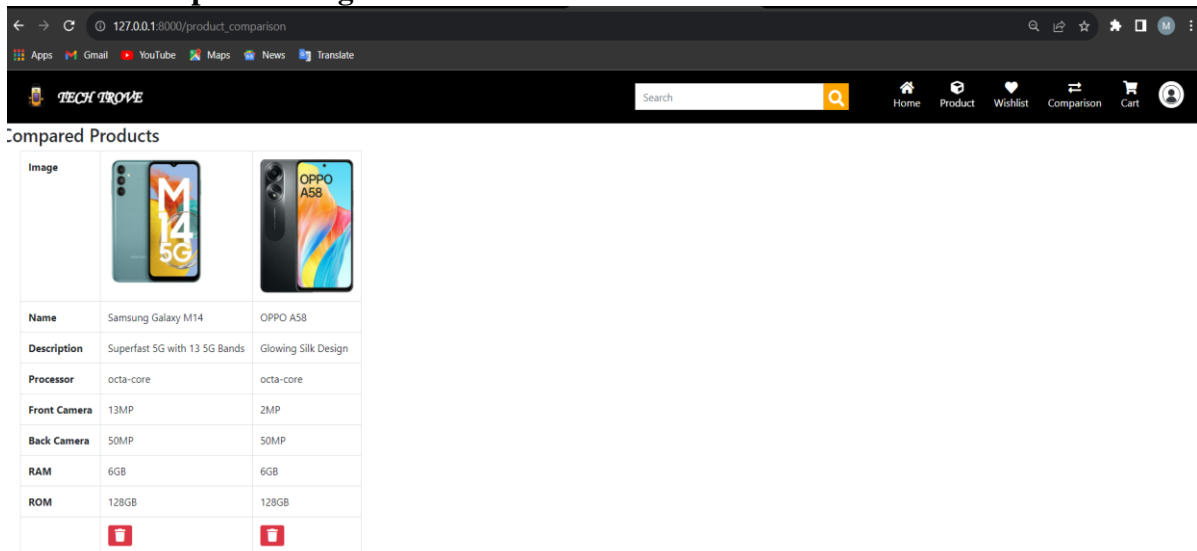


## Wishlist Page

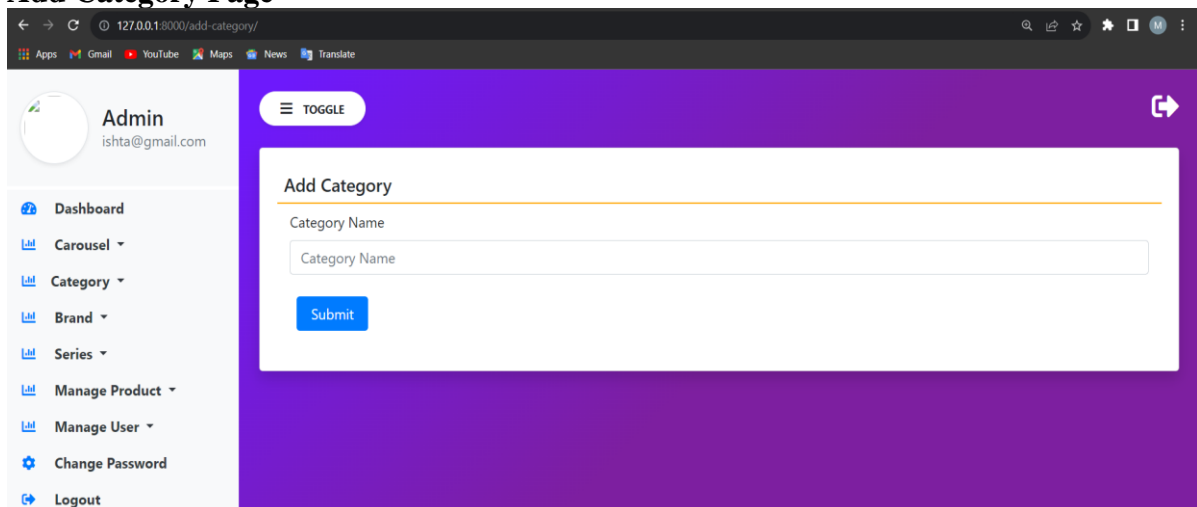




## Product Comparison Page



## Add Category Page



## Add Brand Page

The screenshot shows the 'Add Brands' page in the TechTrove admin dashboard. The page has a purple header and a sidebar on the left. The sidebar contains a user profile for 'Admin' (ishta@gmail.com) and a list of navigation links: Dashboard, Carousel, Category, Brand, Series, Manage Product, Manage User, Change Password, and Logout. The main content area is titled 'Add Brands' and contains a form with two input fields: 'Product Category' (a dropdown menu with 'Choose Category' selected) and 'Brand Name' (a text input field). A blue 'Submit' button is located below the 'Brand Name' field.

## Add Product Page

The screenshot shows the 'Add Product' page in the TechTrove admin dashboard. The page has a purple header and a sidebar on the left. The sidebar contains a user profile for 'Admin' (ishta@gmail.com) and a list of navigation links: Dashboard, Carousel, Category, Brand, Series, Manage Product, Manage User, Change Password, and Logout. The main content area is titled 'Add Product' and contains a form with multiple input fields arranged in two columns. The left column includes: 'Product Category' (dropdown), 'Product Series' (dropdown), 'Product Price' (text input), 'Stock' (text input), 'Description' (text area), 'front camera' (text input), and 'RAM' (text input). The right column includes: 'Product Subcategory' (dropdown), 'Product Name' (text input), 'Discount' (text input), 'Product Image' (file upload button labeled 'Choose File' with 'No file chosen' text), 'Processor' (text input), 'Back Camera' (text input), and 'Rom' (text input). A blue 'Submit' button is located at the bottom left of the form.

