SWIFTBUY

A PROJECT REPORT

Submitted By

Adithya Vinod (SNG22CS015)

Adwaid Manoj (SNG22CS016)

Ajay Das (SNG22CS021)

Anamika S (SNG22CS033)

To

The APJ Abdul Kalam Technological University

In partial fulfillment of the requirements for the award of the degree

Of

Bachelor of Technology

In

Computer Science and Engineering

Department of Computer Science and Engineering

Sree Narayana Gurukulam College of Engineering,

Kadayiruppu, 682311

APRIL,2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
SREE NARAYANA GURUKULAM COLLEGE OF ENGINEERING,
KADAYIRUPPU, 682311

(Affiliated to APJ Abdul Kalam Technological University &Approved by A.I.C.T.E)

CERTIFICATE

This is to certified that the project report, "**SwiftBuy**" submitted by ADWAID MANOJ,ADITHYA VINOD,AJAY DAS and ANAMIKA S to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision.This report in any form has not been submitted to any other University or Institute for any purpose

Dr. Smitha Suresh	**Archana P.S**	**Vinila V**
(Prof. CSE Dept)	(Asst Prof. CSE Dept)	(Asst Prof. CSE Dept)
HEAD OF THE DEPARTMENT	**COORDINATOR**	**GUIDE**

Submitted for the University Evaluation on……

University Register No……

Internal Examiner External Examiner

DECLARATION

We undersigned hereby declare that the project report "**SWIFTBUY**" submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Asst. Prof. Vinila V . This submission represents our ideas in our own words and where ideas or words of others have been included. We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Kadayiruppu

Date: 03/04/2025

Adwaid Manoj

Adithya Vinod

Ajay Das

Anamika S

COURSE OUTCOME AND PROGRAM OUTCOMES

COURSE OUTCOMES: After the completion of the course the student will be able to

CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools &advanced programming techniques (Cognitive Knowledge Level: Apply)
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)

Program outcomes
Engineering Graduates will be able to:
PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES (PSO's)
PSO1: Shall enhance the employability skills by finding innovative solutions for challenges and problems in various domains of CS.
PSO2: Shall apply the acquired knowledge to develop software solutions and innovative mobile applications for various problems.

CO PO PSO MAPPING

	PO1 (Engineering Knowledge)	PO2 (Problem Analysis)	PO3(Design/Development of Solution)	PO4 (Investigation of complex problem)	PO5 (Modern tool usage)	PO6 (The Engineer and Society)	PO7 (Environment and Sustainability)	PO8 (Ethics)	PO9 (Individual and team work)	PO10 (Communication)	PO11(Management and Finance)	PO12 (Life long learning)	PSO1 finding innovative solution	PSO2 Software envelopment
CO1	2	2	2	2		2	2	2	2	2	2	2	2	2
CO2	2	2	2	2	2	2		2	2	2	2	2	2	2
CO3	2	3	3	3	3	2	2	2	3	3	2	2	3	3
CO4	2	2	2	2	2			2	2	2	2	2		2
CO5	3	3	3	3	3	2	2	2	2		3	3	3	3
AVERAGE	2.2	2.4	2.4	2.4	2.5	2.0	2.0	2.0	2.2	2.25	2.2	2.2	2.5	2.4

PO PSO Attainment and Justification

PO	Attained Point (0/1/2/3)	Justification
PO1	3	The system applies engineering knowledge in software development and cybersecurity to improve e-commerce. It ensures secure payments and scalable architecture.
PO2	3	User preferences are analyzed to optimize search algorithms. Large-scale data analysis refines recommendations and enhances user interactions.
PO3	3	Advanced filtering and secure transactions improve online shopping. The system maintains efficiency and user-friendliness.
PO4	3	Real-time tracking and data analytics optimize operations and decision-making. Customer behavior analysis enhances satisfaction.
PO5	3	Modern web technologies and encrypted transactions enhance security and reliability. The system ensures seamless user engagement.
PO6	2	Secure transactions and data protection uphold ethical shopping. Customer privacy is safeguarded while preventing cyber threats.
PO7	1	Sustainability is not a core focus but can be promoted through eco-friendly filters and responsible brand recommendations.
PO8	3	Ethical considerations are maintained through data privacy policies and compliance with security standards in e-commerce.
PO9	3	Engineers, designers, and security experts collaborate to create an efficient and user-friendly shopping experience.
PO10	3	An intuitive UI, real-time notifications, and order tracking ensure clear communication and improve user engagement.
PO11	3	Financial transactions and project management sustain feature enhancements and long-term scalability.
PO12	3	The system evolves with new technologies, integrating innovations to meet e-commerce demands.
PSO1	3	The project enhances employability by showcasing expertise in data analytics and secure web development.
PSO2	3	Engineering principles are applied to create an integrated platform that ensures a smooth shopping experience.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Dedicating this project to the Almighty God whose abundant grace and mercy enabled its successful completion, we would like to express our profound gratitude to all the people who had inspired and motivated us to undertake this project.We wish to express our sincere thanks to our Head of the Department,**Assoc. Prof.(DR.) Smitha Suresh**, for providing us the opportunity to undertake this project. We are deeply indebted to our project guide **Asst. Prof. Vinila V** and project coordinator **Asst. Prof. Archana P.S** in the Department of Computer Science and Engineering for providing us with valuable advice and guidance during the course of the project. Finally, we would like to express our gratitude to Sree Narayana Gurukulam College of Engineering for providing us with all the required facilities without which the successful completion of the project would not have been possible.

ABSTRACT

SwiftBuy is a comprehensive e-commerce solution designed to deliver a seamless and personalized shopping experience. Offering a diverse range of product categories, including electronics, fashion, and groceries, **SwiftBuy** enables users to browse, purchase, and manage products with ease.

Key features include AI-driven personalized recommendations, advanced search and filtering options, secure payment gateways, and real-time order tracking from purchase to delivery. Users can compare products, read reviews, create wishlists, and receive notifications about exclusive deals and discounts.

With a strong focus on security, **SwiftBuy** ensures a robust checkout process and supports multiple payment methods for user convenience. Its intuitive interface enhances navigation, while features like product comparison and customer reviews empower shoppers to make informed decisions.

By integrating smart recommendations, secure transactions, and an enhanced user experience, **SwiftBuy** redefines online shopping, offering a safe, convenient, and engaging platform for all users.

Table of Figures

CHAPTER 1
INTRODUCTION

OVERVIEW

SwiftBuy is an advanced e-commerce solution designed to provide users with a seamless, convenient, and personalized shopping experience. It enables customers to browse, purchase, and manage a wide variety of products across categories such as electronics, fashion, and groceries.

SwiftBuy integrates cutting-edge technologies to enhance user engagement, including AI-driven product recommendations, advanced search and filtering options, and secure payment gateways. Users can compare products, read customer reviews, create wishlists, and track their orders in real time from purchase to delivery.

With a strong focus on security and accessibility, the platform ensures a smooth checkout process with multiple payment options. Its intuitive user interface makes navigation effortless, catering to both new and experienced online shoppers. By offering personalized experiences, secure transactions, and efficient order management, SwiftBuy revolutionizes online shopping, making it more user-friendly, secure, and enjoyable for all.

OBJECTIVE

The objective of **SwiftBuy** is to create a seamless, secure, and user-friendly e-commerce system that enables customers to browse, purchase, and manage products efficiently. By integrating AI-driven recommendations, advanced search and filtering options, secure payment gateways, and real-time order tracking, the platform enhances the overall shopping experience. It aims to provide users with personalized product suggestions, easy navigation, and multiple payment methods while ensuring a secure and efficient transaction process. Additionally, SwiftBuy focuses on accessibility and user engagement through features like product comparisons, customer reviews, and exclusive deals, making online shopping more convenient, reliable, and enjoyable for all users.

SCOPE

The project scope includes developing **SwiftBuy**, a comprehensive online shopping platform designed to provide a seamless and efficient shopping experience. It involves integrating AI-driven product recommendations, secure payment gateways, and real-time order tracking to enhance user convenience. Additionally, the project focuses on designing an intuitive and user-friendly web interface, implementing advanced search and filtering options, and ensuring a secure and smooth transaction process. Features such as product comparisons, customer reviews, wishlists, and personalized deals will also be incorporated to improve user engagement and satisfaction, making **SwiftBuy** a reliable and innovative e-commerce solution.

MOTIVATION

This project is motivated by the growing demand for a seamless, secure, and personalized online shopping experience. In today's digital era, consumers seek convenience, efficiency, and reliability when making purchases, yet many e-commerce platforms lack intuitive navigation, personalized recommendations, and robust security features. **SwiftBuy** aims to bridge this gap by integrating AI-driven product suggestions, secure payment options, and real-time order tracking to enhance user satisfaction. By providing advanced search filters, product comparisons, and customer reviews, the platform empowers shoppers to make informed purchasing decisions. With a user-friendly interface and accessibility for diverse users, **SwiftBuy** strives to revolutionize online shopping, ensuring a more engaging, secure, and efficient retail experience.

FEATURES OF PROJECT WORK

1. **Seamless Shopping Experience: SwiftBuy** provides a hassle-free shopping journey with an intuitive interface, allowing users to browse, search, and purchase products effortlessly.

2. **Secure Transactions: SwiftBuy** integrates robust security measures, including encrypted payment gateways and multi-layer authentication, ensuring safe and reliable transactions.

3. **Real-Time Order Tracking:** Users can monitor their orders from purchase to delivery with real-time tracking features, offering transparency and convenience.

4. **User-Centric Features:** The platform includes advanced search filters, product comparisons, customer reviews, and wishlists, empowering users to make informed purchasing decisions.

5. **Scalability and Performance Optimization:** Designed to handle high traffic efficiently, the platform ensures smooth performance, scalability, and minimal downtime for an optimal shopping experience.

CHAPTER 2

LITERATURE SURVEY

PAPER 1: Online Product Decision Support Using Sentiment Analysis and Fuzzy Cloud-based Multi-Criteria Model through Multiple E-Commerce Platforms

Authors: Yang Z., Li Q., Vincent C., Xu B., Gupta S. (2023)

This paper presents a novel approach to online product decision-making by integrating BiLSTM-CRF, sentiment analysis, and K-means clustering for effective product attribute mining. The study introduces the q-Rung Orthopair Fuzzy Cloud (q-ROFC) method to analyze sentiment errors and proposes a multi-platform online decision framework using fuzzy set theory. The results highlight improved accuracy in mining product sentiments and enhanced decision-making by leveraging cross-platform reviews. However, challenges include limited real-time scalability and potential biases from uneven review distribution and fake reviews. Future work suggests expanding applications beyond e-commerce and developing real-time systems for live consumer behavior analysis.

PAPER 2: A Static Machine Learning-Based Evaluation Method for Usability and Security Analysis in E-Commerce Websites

Authors: Kumar B., Roy S., Singh K.U., Pandey S.K., Kumar A., et al. (2023)

This study evaluates usability and security in e-commerce websites using Analytic Hierarchy Process (AHP), VIKOR, and TOPSIS for multi-criteria decision-making. Usability was assessed through user surveys and statistical analysis, while security evaluations relied on online malware scanners like SUCURI and Qualys. The proposed approach effectively combines usability and security metrics, offering insights into e-commerce website performance, particularly in rural regions. However, limitations include a small sample size, geographic constraints, and a lack of adaptation to dynamic usability trends. Future research aims to integrate real-time data for adaptive evaluation and expand the methodology to global e-commerce platforms for broader analysis.

PAPER 3: Sentiment Analysis in E-Commerce Platforms: A Review of Current Techniques and Future Directions

Authors: Huang Huang, Adeleh Asemi Zavareh, Mumtaz Begum Mustafa (2023)

This paper provides a comprehensive review of sentiment analysis techniques in e-commerce by

analyzing 54 experimental studies. It explores both machine learning approaches (e.g., SVM, Naïve Bayes) and deep learning models (e.g., RNN, LSTM, BERT) while categorizing data sources such as Amazon, Twitter, and IMDB. The review identifies key research gaps and future directions, such as sarcasm detection and aspect-level sentiment analysis. While the study offers a detailed comparison of existing techniques, limitations include a lack of focus on non-English datasets and no experimental validation of proposed research directions. Future research suggests developing universal models for multi-language and cross-domain sentiment analysis, with improvements in implicit aspect recognition, fine-grained sentiment classification, and sarcasm detection.

PAPER 4: Exploring the Landscape of Hybrid Recommendation Systems in E-Commerce

Authors: Kailash Chowdary Bodduluri, Francis Palma, Arianit Kurti, Ilir Jusufi, and Henrik Löwenadler (2024)

This paper presents a **systematic literature review** analyzing **48 studies** on hybrid recommendation systems in e-commerce, focusing on **data synthesis, quality assessment, and trend analysis**. The findings highlight improvements in **recommendation accuracy, scalability, robustness, and personalization**. However, persistent challenges include the **cold start problem, data sparsity, and complexity in evolving user behavior**. Future research suggests integrating **AI, explainable models, multimodal data fusion, and standardized metrics** to enhance transparency and personalization.

PAPER 5: Exploring the Impact of Computer Applications on Cross-Border E-Commerce Performance

Authors: Lijing Jin, Lin Chen (2024)

This study examines the role of **computer applications** in improving **cross-border e-commerce performance**, employing **Genetic Algorithm (GA) optimization** and **deep learning techniques (CNNs)** to enhance operational efficiency. The approach improves **predictive accuracy and customer satisfaction** through advanced **data analytics and machine learning**. However, challenges include **computational complexity, sensitivity to parameter settings, and regulatory compliance issues**. Future research aims to **explore blockchain for secure transactions, AI for dynamic customer personalization, and ethical AI for privacy and security considerations**.

PAPER 6: OntoCommerce: Incorporating Ontology and Sequential Pattern Mining for Personalized E-Commerce Recommendations

Authors: Ghulam Mustafa et al. (2024)

This study presents a hybrid recommender system that integrates ontology-based domain knowledge with Sequential Pattern Mining (SPM) to enhance personalized e-commerce recommendations. The approach effectively addresses cold-start and data sparsity issues, ensuring highly personalized user experiences. However, the model suffers from high computational complexity and relies heavily on accurate ontology construction. Future research aims to extend the approach to other domains and improve the scalability of hybrid systems.

PAPER 7: End-Cloud Collaboration Framework for Advanced AI Customer Service in E-Commerce

Authors: Liangyu Teng, Yang Liu, Jing Liu, Liang Song (2024)

This paper introduces an end-cloud collaboration framework that integrates cloud-based large models with end-device AI models to provide real-time, privacy-preserving AI customer service. The framework employs online evolutive learning, allowing adaptive improvements over time. Key advantages include computational efficiency, enhanced privacy, and adaptability, but challenges involve high initial setup costs for cloud-end collaboration and expensive training requirements. Future research suggests expanding deployment across industries and enhancing optimization strategies with larger training datasets.

PAPER 8: E-Commerce Image Enhancement Method Based on Instance Segmentation and Background Replacement

Authors: Qiang Gao, Huiping Hu, Wei Liu (2024)

This study proposes a computer vision-based image enhancement technique using Fast-SAM instance segmentation, background replacement, and realistic shadow generation to improve product visual appeal and recognition accuracy in e-commerce platforms. The method enhances image clarity and aesthetics while maintaining computational efficiency. However, challenges include limited generalizability to highly diverse product datasets and computational overhead in real-time applications. Future work aims to expand dataset diversity and explore applications in other fields like medical imaging and artwork analysis.

CHAPTER 3

PROBLEM STATEMENT AND OBJECTIVE

PROBLEM STATEMENT

With the rapid growth of e-commerce, consumers face challenges such as overwhelming product choices, lack of personalized recommendations, inefficient search functionality, and security concerns during transactions. Additionally, sellers struggle with optimizing customer engagement and managing inventory effectively. A seamless, secure, and intelligent online shopping platform is essential to enhance user experience, ensure smooth transactions, and improve decision-making for both buyers and sellers.

OBJECTIVE

The primary objective of SwiftBuy is to develop a feature-rich, user-friendly, and secure e-commerce solution that enhances the shopping experience through:

1. **Efficient Search and Navigation** – Advanced filtering, sorting, and keyword-based search functionalities.
2. **Seamless Transaction Processing** – Secure payment gateways for reliable and encrypted transactions.
3. **Real-Time Order Tracking** – Providing users with accurate updates on order status.
4. **Customer Engagement** – Enabling product reviews and ratings for informed decisions.
5. **Scalability & Security** – Ensuring system reliability, protection against fraud, and scalable infrastructure for growing user demand.

This project aims to deliver a next-generation **SwiftBuy** platform that enhances convenience, security, and efficiency for a superior online shopping experience.

CHAPTER 4

SYSTEM ANALYSIS

Here we are familiarizing with the requirements that the system needs. They are listed below by categories.

Functional Requirements

Functional requirements define the essential features that must be incorporated into the system to meet business needs and ensure user satisfaction. The system must include:

1. User Management – Enables account creation, authentication, and profile management.
2. Product Browsing and Search – Allows users to explore products using advanced search and filtering options.
3. Personalized Recommendations – Provides AI-driven product suggestions based on user preferences.
4. Shopping Cart and Wishlist – Enables users to add, remove, and save products for future purchases.
5. Order Management – Facilitates order placement, tracking, and history management.
6. Secure Payment Processing – Ensures safe transactions with multiple payment gateway integrations.
7. Real-Time Notifications – Sends updates on order status, promotions, and personalized deals.
8. Product Review and Comparison – Allows users to rate, review, and compare products for informed decision-making.

NON-FUNCTIONAL REQUIREMENTS

Non functional requirements are a description of features, characteristics and attribute of the system as well as any constraints that may limit the boundaries of the proposed system. Mainly based on performance, information, economy, control and security efficiency and services. Requirements are as follows:

1. Usability
2. Reliability
3. Maintainability
4. Accessibility

5. Scalability

Hardware Requirements

1. **Processor:** Intel i3/i5/i7 processors provide sufficient performance for the application.
2. **RAM:** A minimum of 8GB is recommended, with 16GB ensuring optimal performance.
3. **Storage:** A 256GB HDD is sufficient, but an SSD will significantly enhance performance.
4. **Network:** A high-speed internet connection is essential for smooth operation.

Software Requirements

1. **Windows:** Supports Windows 8 or above.
2. **Linux:** Compatible with Linux distributions.
3. **Mac:** Supports Mac operating systems.

CHAPTER 5

DESIGN

ARCHITECTURE DIAGRAM

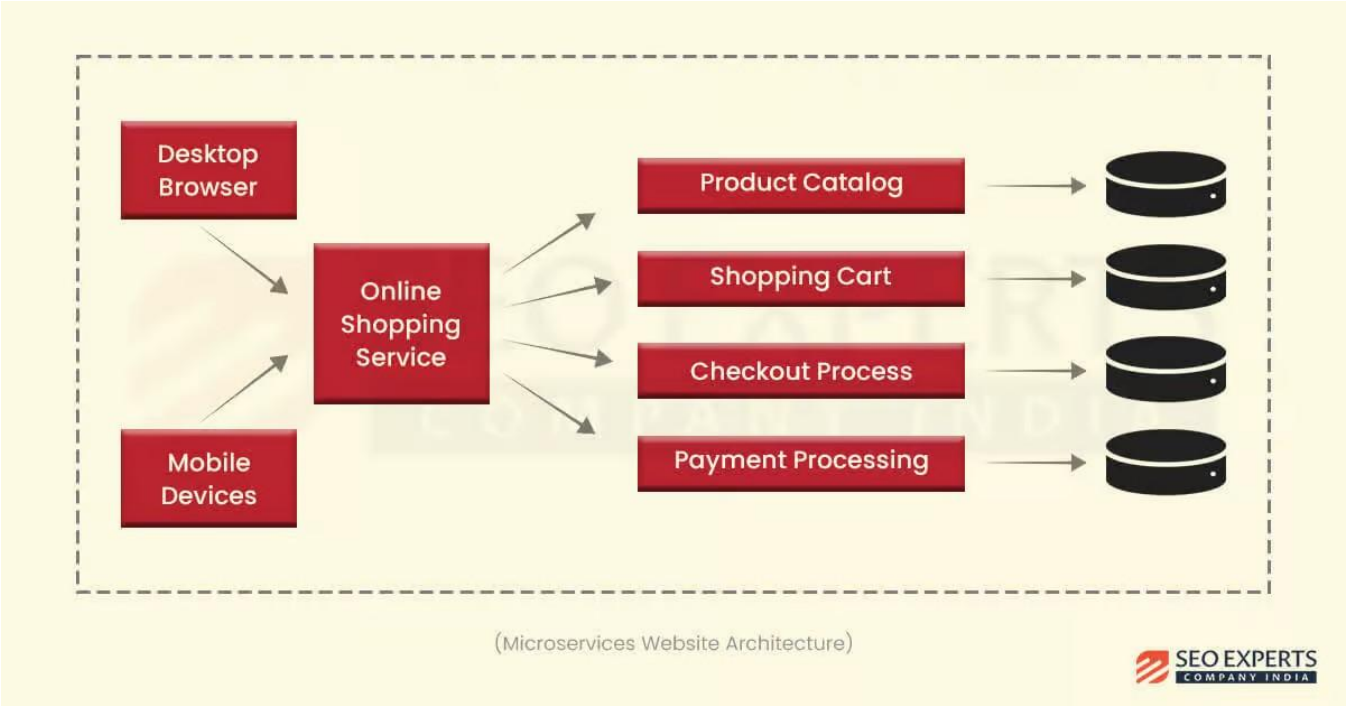

Fig 1 Architecture Diagram

An architecture diagram provides a visual representation of the structure and components of a system or software application. It illustrates the relationships between various elements, including modules, layers, interfaces, and dependencies. Architecture diagrams are used to communicate the design and organization of a system to stakeholders, developers, and other team members. They help in understanding how different parts of the system interact and work together to achieve the desired functionality. These diagrams can range from high-level overviews, showing the overall architecture, to detailed diagrams, depicting specific components and their interactions. Overall, architecture diagrams serve as a blueprint for designing, implementing, and maintaining complex systems.

USE-CASE DIAGRAM

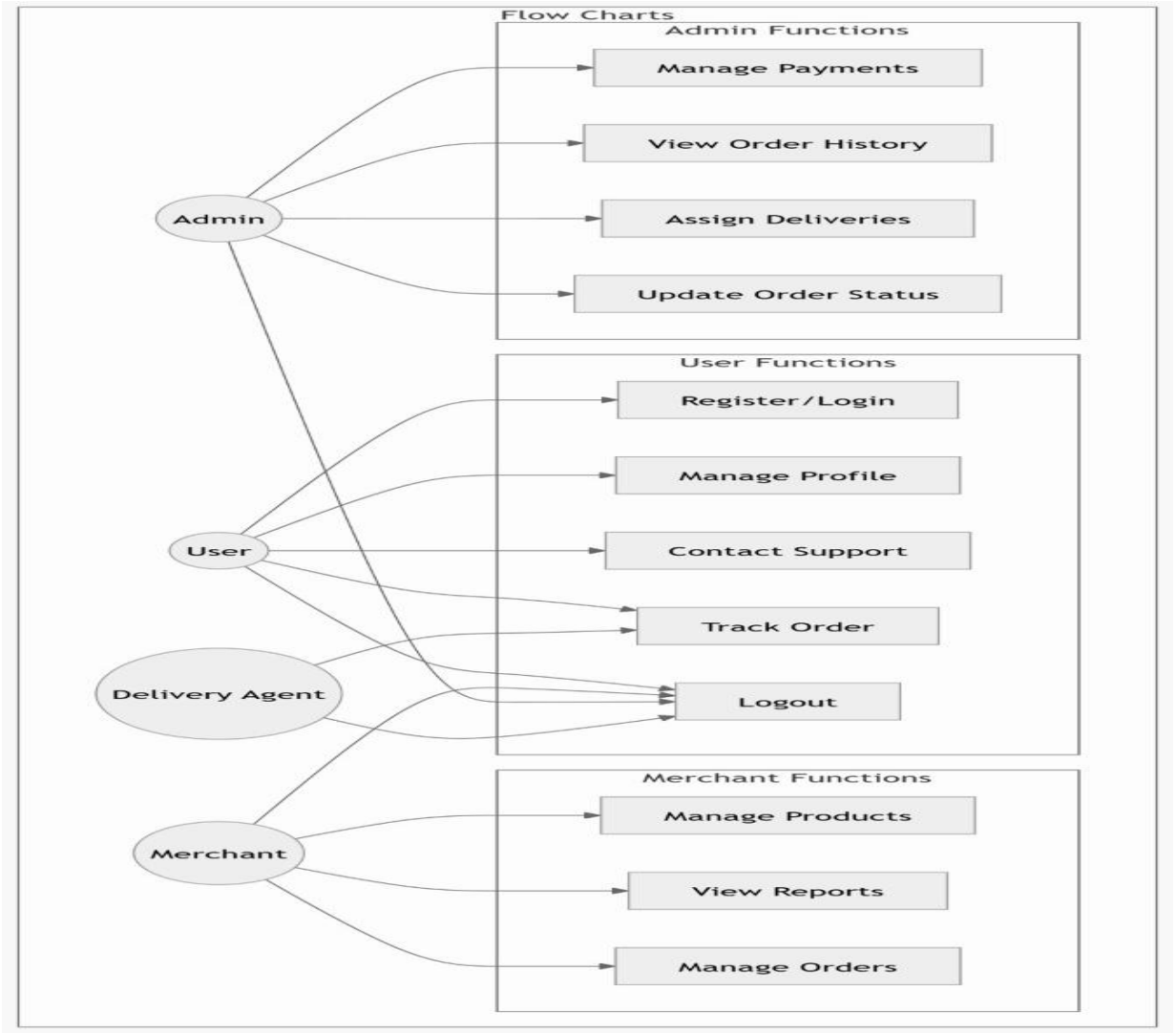

Fig 2 Use case Diagram

A use case diagram is a visual representation that illustrates the interactions between actors (users or external systems) and a system under consideration to achieve specific goals or tasks. It showcases the functionalities or features of a system from the perspective of its users. In a use case diagram, actors are represented as stick figures, while use cases are depicted as ovals. Lines connecting actors and use cases represent the interactions or relationships between them. Use case diagrams help stakeholders, including

developers, designers, and project managers, to understand the system's behavior, functionalities, and how users interact with it. They serve as a valuable tool for requirements elicitation, analysis, and validation during the software development lifecycle.

ER DIAGRAM

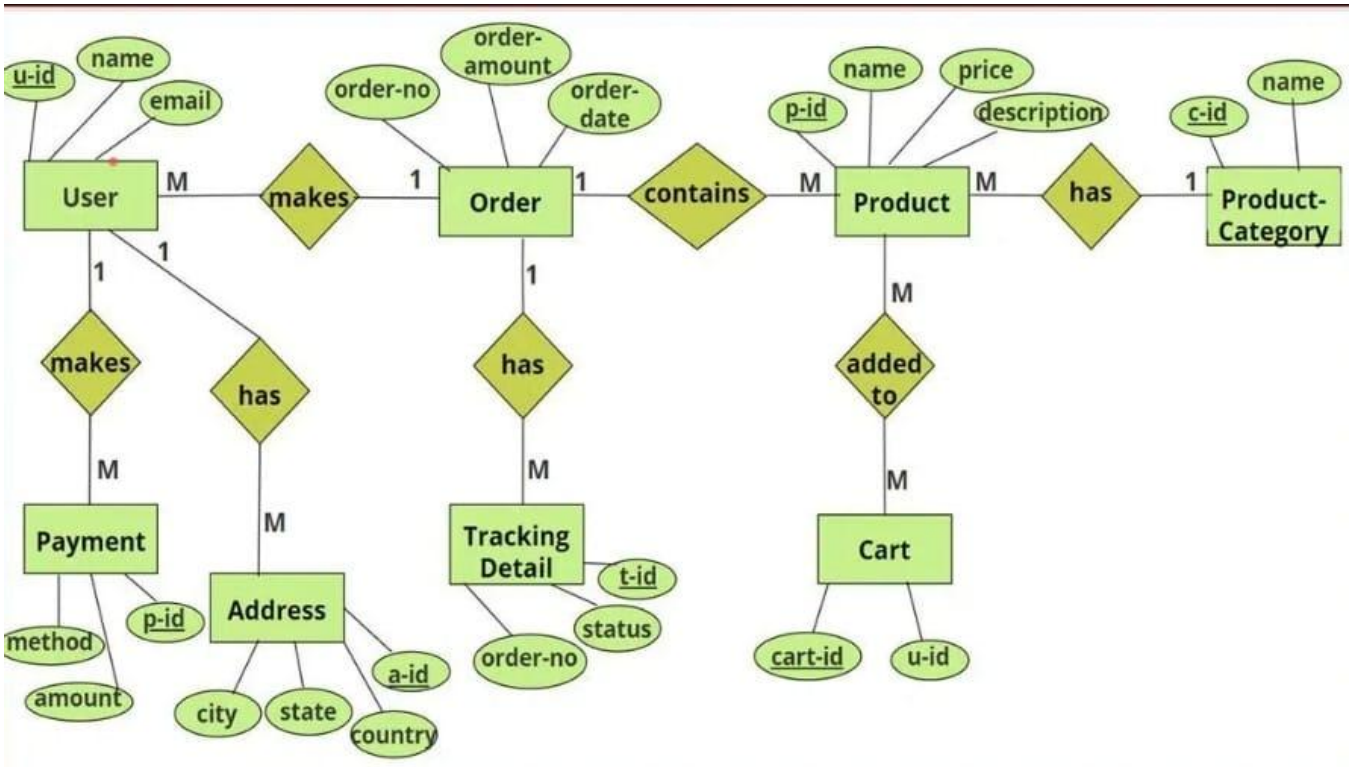

Fig 3 ER-Diagram

An Entity-Relationship (ER) diagram visually represents the data structure of a database by illustrating entities (objects or concepts), their attributes (properties), and relationships (associations) between them. It helps in designing and organizing databases by defining how data is stored, linked, and accessed. ER diagrams range from high-level conceptual models that outline major entities and relationships to detailed logical models specifying attributes, keys, and cardinality constraints. They serve as a blueprint for database design, aiding developers, database administrators, and stakeholders in understanding data flow, ensuring data integrity, and optimizing the system’s structure.

CHAPTER 6

IMPLEMENTATION

Programming Codes

FRONTEND

App.jsx

```
import { BrowserRouter, Route, Routes } from "react-router-dom";
import Login from "./components/Login";
import Signin from "./components/Signin";
import colors from "./colors";
import { AppContext } from "./AppContext";
import { useState } from "react";
import Profile from "./components/Profile";
import ProductAdd from "./components/ProductAdd";
import MyProduct from "./components/MyProduct";
import SearchProduct from "./components/SearchProduct";
import Users from "./components/Users";
import Products from "./components/Products";
import Dashboard from "./components/Dashboard";
import DetailedProduct from "./components/DetailedProduct";
import MyCart from "./components/MyCart";
import MyOrder from "./components/MyOrder";
import Wishlist from "./components/Wishlist";
import AdminTools from "./components/AdminTools";

function App() {
  const [data, setData] = useState({
    username: "",
    place: "",
    age: "",
    email: "",
    password: "",
    role: "",
```

```
  });
  document.body.style.backgroundColor = colors.backgroundcolor;

  return (
    <>
      <BrowserRouter>
        <AppContext.Provider value={{ data, setData }}>
          <Routes>
            <Route path="/" element={<Login />} />
            <Route path="/signin" element={<Signin />} />
            <Route path="/dashboard" element={<Dashboard />} />
            <Route path="/profile" element={<Profile />} />
            <Route path="/product/add" element={<ProductAdd />} />
            <Route path="/merchant/products" element={<MyProduct />} />
            <Route path="/search/products" element={<SearchProduct />} />
            <Route path="/admin/users" element={<Users />} />
            <Route path="/admin/products" element={<Products />} />
            <Route path="/detproduct" element={<DetailedProduct />} />
            <Route path="/mycart" element={<MyCart />} />
            <Route path="/user/orders" element={<MyOrder />} />
            <Route path="/user/wishlist" element={<Wishlist />} />
            <Route path="/admintools" element={<AdminTools />} />
          </Routes>
        </AppContext.Provider>
      </BrowserRouter>
    </>
  );
}

export default App;
```

.env

```
VITE_API_URL=http://localhost:3000
```

Login.jsx

```
import React, { useContext, useState } from "react";
import { Box, Button, TextField, Typography } from "@mui/material";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";
import styles from "../styles";
import { AppContext } from "../AppContext";

const Login = () => {
  const { setData } = useContext(AppContext);
  const [user, setUser] = useState({ email: "", password: "" });
  const [errors, setErrors] = useState({ email: false, password: false });
  const [generalError, setGeneralError] = useState("");
  const navigate = useNavigate();
  const api_url = import.meta.env.VITE_API_URL;

  const inputHandler = (e) => {
    setUser({ ...user, [e.target.name]: e.target.value });
    setErrors({ ...errors, [e.target.name]: false });
    setGeneralError("");
  };

  const validateFields = () => {
    const newErrors = {
      email: user.email === "",
      password: user.password === "",
    };
    setErrors(newErrors);
    return !newErrors.email && !newErrors.password;
  };
```

```
const submitHandler = async () => {
  if (validateFields()) {
    try {
      const login = await axios.get(
        `${api_url}/user/get/${user.email}/${user.password}`
      );
      const userData = {
        username: login.data.username,
        email: login.data.email,
        place: login.data.place,
        age: login.data.age,
        password: login.data.password,
        role: login.data.role,
        _id: login.data._id,
      };
      setData(userData);
      localStorage.setItem("userData", JSON.stringify(userData));
      navigate("/dashboard", { state: login.data });
    } catch (error) {
      console.error(error);
      setGeneralError("Invalid Email or Password");
    }
  }
};

return (
  <div>
    <Box
      sx={{
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
        height: "97vh",
      }}
    >
```

```
<Box sx={styles.box_style}>
  <img
    src="/logo1.png"
    alt="Login Icon"
    style={{
      width: "180px",
      marginBottom: "-1rem",
      marginTop: "-2.5rem",
    }}
  />
  <Typography
    fontFamily={"fantasy"}
    variant="h3"
    color="white"
    style={{
      marginBottom: "0px",
    }}
    gutterBottom
  >
    LOG-IN
  </Typography>
  <TextField
    required
    fullWidth
    name="email"
    label="Email"
    variant="outlined"
    margin="normal"
    value={user.email}
    onChange={inputHandler}
    error={errors.email}
    helperText={errors.email ? "Email is required" : generalError}
    FormHelperTextProps={{
      sx: { color: errors.email ? "red" : "red" },
    }}
    InputLabelProps={{ style: { color: "white" } }}
```

```
  InputProps={styles.textfield}
/>
<TextField
  required
  fullWidth
  name="password"
  type="password"
  label="Password"
  variant="outlined"
  margin="normal"
  value={user.password}
  onChange={inputHandler}
  error={errors.password}
  helperText={errors.password ? "Password is required" : ""}
  InputLabelProps={{ style: { color: "white" } }}
  InputProps={styles.textfield}
/>
<Button
  variant="contained"
  sx={{
    mt: 2,
    backgroundColor: "orange",
    "&:hover": { backgroundColor: "orange" },
  }}
  onClick={submitHandler}
>
  Log-in
</Button>
<Box mt={2}>
  <Typography style={{ color: "darkgray" }}>
    Don't have an Account? 
    <Link style={styles.link_style} to={"/signin"}>
      SignUp
    </Link>
  </Typography>
</Box>
```

```
        </Box>
      </Box>
    </div>
  );
};

export default Login;
```

MyCart.jsx

```
import React, { useEffect, useState } from "react";
import { Link, useLocation, useNavigate } from "react-router-dom";
import axios from "axios";
import {
  Box,
  Button,
  Container,
  Grid,
  Input,
  List,
  ListItem,
  Paper,
  Rating,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  Typography,
} from "@mui/material";
import Navbar from "./Navbar";

const MyCart = () => {
  const navigate = useNavigate();
  const [loading, setLoading] = useState(true);
  const [products, setProducts] = useState([]);
```

```
const [empty, setEmpty] = useState(true);
const api_url = import.meta.env.VITE_API_URL;
const data = JSON.parse(localStorage.getItem("userData"));
const [checkout, setCheckout] = useState(true);
var Price = 0;
var Index = 1;

useEffect(() => {
  const apiUrl = `${api_url}/user/getcart/${data._id}`;
  axios
    .get(apiUrl)
    .then((response) => {
      setProducts(response.data);
      console.log(response.data);
      setEmpty(response.data.length === 0);
    })
    .catch((error) => {
      console.error("Error fetching data:", error);
    })
    .finally(() => {
      setLoading(false);
    });
  // }
}, []);

useEffect(() => {
  products.forEach((product) => {
    if (product.product.stock < product.quantity) {
      setCheckout(false);
    }
  });
}, [products]);

return (
  <div>
    <Navbar />
```

```
{loading ? (
  <center>
    <br />
    <br />
    <br />
    <br />
    loading...
  </center>
) : empty ? (
  <center>
    <Typography style={{ fontSize: 17, marginTop: "50vh" }}>
      Your cart is empty
    </Typography>
  </center>
) : (
  <Box
    sx={{
      display: "flex",
      alignItems: "flex-start",
      marginTop: "9vh",
      width: "90%",
      justifyContent: "space-between",
    }}
  >
    <TableContainer>
      <Table>
        <TableHead>
          <TableRow>
            <TableCell
              sx={{
                fontFamily: "fantasy",
                color: "white",
                fontWeight: "bold",
                fontSize: "3vh",
                width: "600px",
              }}
```

```
    >
      ITEM
    </TableCell>
    <TableCell
      sx={{
        fontFamily: "fantasy",
        color: "white",
        fontWeight: "bold",
        fontSize: "3vh",
      }}
    >
      PRICE
    </TableCell>
    <TableCell
      sx={{
        fontFamily: "fantasy",
        color: "white",
        fontWeight: "bold",
        fontSize: "3vh",
      }}
    >
      QUANTITY
    </TableCell>
    <TableCell
      sx={{
        fontFamily: "fantasy",
        color: "white",
        fontWeight: "bold",
        fontSize: "3vh",
      }}
    >
      TOTAL
    </TableCell>
  </TableRow>
</TableHead>
<TableBody>
```

```
{products.map((product, index) => (
 <TableRow key={index}>
  <TableCell
   sx={{
    fontFamily: "cursive",
    color: "white",
    display: "flex",
    alignItems: "flex-start",
   }}
  >
   <Box>
    <img
     src={`${api_url}/${product.product.image}`}
     alt={product.product.name}
     style={{
      width: 150,
      height: "auto",
      cursor: "pointer",
     }}
     onClick={() => {
      navigate("/detproduct", { state: product.product });
     }}
    />
   </Box>
   <Box sx={{ mt: 11.8, ml: 6 }}>
    <Typography
     sx={{
      fontFamily: "fantasy",
      color: "white",
      fontSize: 32,
      mt: -10,
     }}
    >
     {product.product.name}
    </Typography>
    <Box sx={{ display: "flex", alignItems: "center" }}>
```

```
<Box
 sx={{
  mt: 1,
  display: "flex",
  alignItems: "center",
  backgroundColor: "#222",
  color: "white",
  borderRadius: "8px",
  padding: "4px 8px",
  fontSize: "14px",
  fontWeight: "bold",
  width: "fit-content",
 }}
>
 <Typography sx={{ mr: 0.5, color: "#FFAD18" }}>
  {parseFloat(product.product.rating.toFixed(1)) ||
   0}{" "}
  ⭐
 </Typography>
</Box>
<Typography
 sx={{
  color: "white",
  fontFamily: "cursive",
  ml: 1,
  mt: 0.8,
 }}
>
 {product.product.reviews.length === 0
  ? "No Rating"
  : product.product.reviews.length === 1
  ? "1 Rating"
  : `${product.product.reviews.length} Ratings`}
</Typography>
</Box>
<Button
```

```
      onClick={async () => {
        try {
          await axios.delete(
            `${api_url}/user/cart/delitem/${data._id}/${product.product._id}`
          );
          window.location.reload(true);
        } catch (error) {
          console.error(error);
        }
      }}
      variant=""
      sx={{
        ml: -2,
        mt: 1,
        color: "transparent",
        "&:hover .remove_cart": {
          color: "red",
        },
      }}
    >
      <Typography
        className="remove_cart"
        color="white"
        sx={{
          fontFamily: "fantasy",
          textTransform: "none",
        }}
      >
        Remove
      </Typography>
    </Button>
  </Box>
</TableCell>
<TableCell>
  <Typography
    sx={{
```

```
      mt: 1.5,
      fontFamily: "cursive",
      color: "yellow",
    }}
  >
    ₹{product.product.price}
  </Typography>
</TableCell>
<TableCell>
  <Box
    sx={{
      display: "flex",
      alignItems: "center",
      border: "2px solid white",
      borderRadius: "8px",
      width: "100px",
      height: "25px",
      overflow: "hidden",
    }}
  >
    <Button
      sx={{
        minWidth: "30px",
        fontSize: "20px",
        borderRight: "2px solid white",
        borderRadius: "0",
        color: "white",
      }}
      onClick={async () => {
        if (product.quantity === 1) {
          try {
            await axios.delete(
              `${api_url}/user/cart/delitem/${data._id}/${product.product._id}`
            );
            window.location.reload(true);
          } catch (error) {
```

```
                    console.error(error);
                   }
                  } else {
                   try {
                    setProducts((prevProducts) =>
                     prevProducts.map((p) =>
                      p.product._id === product.product._id
                       ? { ...p, quantity: p.quantity - 1 }
                       : p
                     )
                    );
                    product.quantity -= 1;
                    await axios.post(
                     `${api_url}/user/cart/updateitemquantity/${data._id}/${product.product._id}/${pro
duct.quantity}`
                    );
                    if (checkout === false) {
                     window.location.reload(true);
                    }
                   } catch (error) {
                    console.error(error);
                   }
                  }
                 }}
                >
                 -
                </Button>

                <Typography
                 type="text"
                 sx={{
                  width: "40px",
                  textAlign: "center",
                  fontSize: "16px",
                  background: "transparent",
                  color: "white",
```

```
  }}
>
  {product.quantity}
</Typography>

<Button
  sx={{
    minWidth: "30px",
    fontSize: "20px",
    borderLeft: "2px solid white",
    borderRadius: "0",
    color: "white",
  }}
  onClick={async () => {
    setProducts((prevProducts) =>
      prevProducts.map((p) =>
        p.product._id === product.product._id
          ? p.quantity >= p.product.stock
            ? { ...p, available: "Not available" }
            : {
                ...p,
                quantity: p.quantity + 1,
                available: "",
              }
          : p
      )
    );

    if (product.quantity < product.product.stock) {
      try {
        await axios.post(
          `${api_url}/user/cart/updateitemquantity/${
            data._id
          }/${product.product._id}/${
            product.quantity + 1
          }`
```

```
            );
          } catch (error) {
            console.error(error);
          }
        }
      }}
    >
      +
    </Button>
  </Box>
  <Typography
    sx={{
      color: "red",
      minHeight: "25px",
      mb: -4.5,
      mt: 0.5,
      ml: 0.5,
    }}
  >
    {product.available}
  </Typography>
  {product.product.stock < product.quantity && (
    <Typography
      sx={{
        color: "red",
        minHeight: "25px",
        mb: -4.5,
        mt: 1,
        ml: 0.5,
      }}
    >
      Not available
    </Typography>
  )}
</TableCell>
<TableCell>
```

```
            <Typography
              sx={{
                mt: 1.5,
                fontFamily: "cursive",
                color: "yellow",
              }}
            >
              ₹{`${product.product.price * product.quantity}`}
            </Typography>
          </TableCell>
        </TableRow>
      ))}
    </TableBody>
  </Table>
</TableContainer>
<Box
  sx={{
    width: "18%",
    height: "89vh",
    mt: "1vh",
    position: "sticky",
    top: "10vh",
    borderLeft: "1px solid white",
  }}
>
  <Box
    sx={{
      borderBottom: "1px solid white",
      width: "25.65vw",
    }}
  >
    <Typography
      sx={{
        ml: 3,
        mb: 1.4,
        mt: 0.25,
```

```
      fontFamily: "fantasy",
      color: "white",
      fontWeight: "bold",
      fontSize: "3vh",
      width: "300px",
    }}
  >
    SUMMARY
  </Typography>
</Box>

{products.forEach((product, index) => {
  Price += product.product.price * product.quantity;
  Index += index;
  console.log(product);
})}
<Box
  sx={{
    minWidth: 350,
    mt: 2,
    ml: 3,
  }}
>
  <Typography
    sx={{ fontFamily: "cursive", fontWeight: "bold", fontSize: 23 }}
  >
    Total Items :<span style={{ color: "orange" }}> {Index}</span>
    <br />
    <br />
    Delivery : Free
    <br />
    <br />
    Payment : Cash On Delivery
    <br />
    <br />
    Tax Payable :<span style={{ color: "orange" }}> ₹0</span>
```

```
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<br />
 Estimated Total :
<span style={{ color: "darkorange" }}> ₹{Price}</span>
<br />
{checkout ? (
 <Button
  variant="contained"
  sx={{
   mt: 3,
   ml: 1,
   fontSize: "20px",
   padding: "15px 30px",
   minWidth: "350px",
   backgroundColor: "orangered",
  }}
  onClick={async () => {
   try {
    for (const product of products) {
     await axios.post(
      `${api_url}/product/buy/${data._id}/${product.product._id}/${product.quantity}`
     );
    }

    for (const product of products) {
     await axios.delete(
      `${api_url}/user/cart/delitem/${data._id}/${product.product._id}`
     );
    }
```

```
                window.location.reload(true);
                console.log("Order confirmed");
              } catch (error) {
                console.error("Error ordering product:", error);
              }
            }}
          >
            CheckOut
          </Button>
        ) : (
          <Button
            variant="contained"
            sx={{
              mt: 3,
              ml: 1,
              fontSize: "20px",
              padding: "15px 30px",
              minWidth: "350px",
              backgroundColor: "gray",
              cursor: "not-allowed",
            }}
          >
            CheckOut
          </Button>
        )}
      </Typography>
    </Box>
  </Box>
</Box>
  )}
 </div>
 );
};

export default MyCart;
```

BACKEND

Connection.js

```
var mongoose = require("mongoose");

mongoose.connect("mongodb+srv://test:test@cluster0.cguda.mongodb.net/new1?retryWrites=true&w=
majority&appName=Cluster0").then(()=>{
   console.log("Connected!");
}).catch((error) => {
   console.log(error)
})
```

Index.js

```
var express = require("express");
var cors = require("cors");
var path = require('path');
var multer = require("multer");
var crypto = require('crypto')
var app = express();
var fs = require("fs");
require("./connection.js");

const URL = "http://localhost";
const PORT = 3000;

var CryptoJS = require('crypto-js');
var userModel = require("./models/user");
var productModel = require("./models/product");
var orderModel = require("./models/order")
var miscModel = require("./models/misc")

const storage = multer.diskStorage({
```

```
    destination: (req, file, cb) => {
        cb(null, 'images/products');
    },
    filename: (req, file, cb) => {
        cb(null, crypto.randomBytes(8).toString('hex').slice(0, 8) + path.extname(file.originalname));
    },
});
const upload = multer({storage: storage});

app.use(express.json());
app.use(cors());

app.get("/user/viewall", async (req, res) => {
    try {
        var data = await userModel.find();
        res.send(data)

    } catch (error) {
        console.log(error);
    }
});

app.post("/user/add", async (req, res) => {
    try {
        await userModel(req.body).save();
        res.send({message: "Data Added"});
    } catch (error) {
        console.log(error);
    }
});

app.get("/user/get/:email/:password", async (req, res) => {
    try {
        var email = req.params.email;
        var password = CryptoJS.SHA256(req.params.password).toString(CryptoJS.enc.Hex)
        var user = await userModel.findOne({email: email, password: password});
```

```
        if (user) {
            res.send(user);
        } else {
            res.status(404);
            res.send({message: "Invalid Email or Password"});
        }

    } catch (error) {
        console.log(error);
    }
});

app.post("/user/register/", async (req, res) => {
    try {
        var user = req.body;
        user.admin = false;
        user.merchant = false;
        user.cart = [];
        user.wishlist = [];
        user.password = CryptoJS.SHA256(user.password).toString(CryptoJS.enc.Hex)
        var existing_user = await userModel.findOne({email: user.email});
        if (!existing_user) {

            await userModel(user).save();
            res.send({message: "Account Registered"});

        } else {
            res.status(409);
            res.send({message: "Email Already Exists"});
        }

    } catch (error) {
        console.log(error);
    }
});
```

```
app.put("/user/edit/", async (req, res) => {
    try {
        var id = req.body._id;
        var user = req.body;
        var existing_user = await userModel.findOne({email: user.email});
        if (!existing_user) {
            await userModel.findByIdAndUpdate(id, user);
            res.send({message: "Profile Updated"})
        } else if (existing_user._id == user._id) {
            await userModel.findByIdAndUpdate(id, user);
            res.send({message: "Profile Updated"})
        } else {
            res.status(409);
            res.send({message: "Email Already Exists."})
        }

    } catch (error) {
        console.log(error);
    }
})

app.delete("/user/delete/", async (req, res) => {
    try {
        var id = req.body._id;
        var del = await userModel.findByIdAndDelete(id);
        if (del != null) {
            var delproducts = await productModel.find({owner: id})
            await productModel.deleteMany({owner: id});
            for (let i = 0; i < delproducts.length; i++) {
                fs.unlink(delproducts[i].image, () => {
                });
            }
            res.send({message: "Account Deleted"});
        } else {
            res.status(404);
            res.send({message: "Failed To Delete Account"});
```

```
        }

    } catch (error) {
        res.status(404);
        res.send({message: "Failed To Delete Account"});
    }
})

app.post("/product/add/", upload.single('file'), async (req, res) => {
    try {
        var product = req.body;
        product.reviews = [];
        product.rating = 0;
        product.featured = false;
        req.body.image = ""
        req.body.keywords = req.body.keywords.split(",")
        product = await productModel(product).save();
        var img_path = `${req.file.destination}/${product._id}${path.extname(req.file.filename)}`;
        fs.rename(req.file.path, img_path, () => {
        })
        product.image = `${img_path}`;
        product.save();
        res.send({message: "Product Added"})

    } catch (error) {
        console.log(error);
    }
});

app.get("/product/viewall", async (req, res) => {
    try {
        var data = await productModel.find();
        res.send(data)

    } catch (error) {
        console.log(error);
```

```
    }
});

app.get("/product/view/:pid", async (req, res) => {
    try {
        var id = req.params.pid
        var data = await productModel.findOne({_id: id});
        res.send(data)

    } catch (error) {
        console.log(error);
    }
});

app.get("/merchant/products/:id", async (req, res) => {
    try {
        var id = req.params.id;
        var data = await productModel.find({merchant_id: id});
        res.send(data)

    } catch (error) {
        console.log(error);
    }
});

app.delete("/product/delete/:id", async (req, res) => {
    try {
        var id = req.params.id;
        var del = await productModel.findByIdAndDelete(id);
        if (del != null) {
            fs.unlink(del.image, () => {
            });
            await userModel.updateMany(
                { "cart.product": id },
                { $pull: { cart: { product: id } } }
            );
```

```
            await userModel.updateMany(
                { "wishlist.product": id },
                { $pull: { wishlist: { product: id } } }
            );
            res.send({message: "Product Deleted"});
        } else {
            res.status(404);
            res.send({message: "Failed To Delete Product"});
        }

    } catch (error) {
        res.status(404);
        res.send({message: "Failed To Delete Product"});
    }
})
app.get("/product/search/:word?", async (req, res) => {
    try {
        var word="";
        word = req.params.word;
        var query = word && word.trim() !== "" ? { name: new RegExp(".*" + word + ".*", "i") } : {};
        var data = await productModel.find(query);

        res.send(data)

    } catch (error) {
        console.log(error);
    }
});

app.post("/product/addreview/:productId", async (req, res) => {
    try {
        var id = req.params.productId;
        var review = req.body;
        var product = await productModel.findById(id);
        product.reviews.unshift(review)
        let total = 0
```

```
        for (let i=0;i<product.reviews.length;i++){
            total += product.reviews[i].rating;
        }
        product.rating = total / product.reviews.length;
        await product.save();
        res.send({ message: "Review Added" })

    } catch (error) {
        console.log(error);
    }
});

app.delete("/product/delreview/:productId/:userId", async (req, res) => {
    try {
        var recid = req.params.productId;
        var userid = req.params.userId;
        var product = await productModel.findById(recid);
        product.reviews = product.reviews.filter(review => review.userId != userid)
        let total = 0
        for (let i=0;i<product.reviews.length;i++){
            total += product.reviews[i].rating;
        }
        product.rating = total / product.reviews.length;
        await product.save();
        res.send({ message: "Review Deleted" })

    } catch (error) {
        console.log(error);
    }
});

app.get("/product/getreviews/:productId", async (req, res) => {
    try {
        var id = req.params.productId;
        var product = await productModel.findById(id);
        res.send(product.reviews);
```

```
    } catch (error) {
        console.log(error);
    }
});

app.post("/product/addtocart/:userId/:productId", async (req, res) => {
    try {
        var userId = req.params.userId;
        var productId = req.params.productId;
        var user = await userModel.findById(userId);
        var cartItem = user.cart.find(item => item.product == productId);
        if (cartItem == undefined){
            user.cart.unshift({"product": productId, "quantity": 1});
        }
        else{
            cartItem.quantity += 1
        }

        await user.save();
        res.send({ message: "Product Added To Cart" })

    } catch (error) {
        console.log(error);
    }
});
app.get("/user/getcart/:userId", async (req, res) => {
    try {
        var userId = req.params.userId;
        var user = await userModel.findById(userId);
        await user.populate("cart.product")

        res.send(user.cart)

    } catch (error) {
        console.log(error);
```

```
    }
});

app.delete("/user/cart/delitem/:userId/:productId", async (req, res) => {
    try {
        var userId = req.params.userId;
        var productId = req.params.productId;
        var user = await userModel.findById(userId);
        user.cart = user.cart.filter(entry => entry.product != productId);
        await user.save();
        res.send({ message: "Product Deleted" })

    } catch (error) {
        console.log(error);
    }
});

app.post("/user/cart/updateitemquantity/:userId/:productId/:quantity", async (req, res) => {
    try {
        var userId = req.params.userId;
        var productId = req.params.productId;
        var quantity = parseInt(req.params.quantity);
        var user = await userModel.findById(userId);
        var item = user.cart.find(entry => entry.product == productId);
        item.quantity = quantity;
        await user.save();
        res.send({ message: "Quantity Updated" })

    } catch (error) {
        console.log(error);
    }
});

app.get("/orders/viewall", async (req, res) => {
    try {
        var data = await orderModel.find();
```

```
        res.send(data)

    } catch (error) {
        console.log(error);
    }
});

app.post("/product/buy/:userId/:productId/:quantity", async (req, res) => {
    try {
        var userId = req.params.userId;
        var productId = req.params.productId;
        var productDetails = await productModel.findById(productId)
        var quantity = parseInt(req.params.quantity);
        var user = await userModel.findById(userId);
        var order = {
            "name": productDetails.name,
            "userId": userId,
            "amount": productDetails.price * quantity,
            "quantity": quantity,
            "status": "Processing",
            "image": productDetails.image,
            "product": productId
        }
        if (productDetails.stock < quantity){
            res.send("Requested Quantity Not In Stock")
        }
        else{
            productDetails.stock -= quantity
            await productDetails.save()
            order = await orderModel(order).save()
            res.send({ message: "Product Order Placed" })
        }

    } catch (error) {
        console.log(error);
    }
```

```
});

app.post("/user/cartcheckout/:userId", async (req, res) => {
    try {
        var userId = req.params.userId;
        var productId = req.params.productId;
        var productDetails = await productModel.findById(productId)
        var quantity = parseInt(req.params.quantity);
        var user = await userModel.findById(userId);
        var order = {
            "name": productDetails.name,
            "userId": userId,
            "amount": productDetails.price * quantity,
            "quantity": quantity,
            "status": "Processing",
            "image": productDetails.image,
            "product": productId
        }
        if (productDetails.stock < quantity){
            res.send("Requested Quantity Not In Stock")
        }
        else{
            productDetails.stock -= quantity
            await productDetails.save()
            order = await orderModel(order).save()
            res.send({ message: "Product Order Placed" })
        }

    } catch (error) {
        console.log(error);
    }
});

app.get("/user/orders/:userId", async (req, res) => {
    try {
        var userId = req.params.userId
```

```
        var data = await orderModel.find({userId: userId})
        res.send(data)

    } catch (error) {
        console.log(error);
    }
});

app.post("/product/makefeatured/:productId", async (req, res) => {
    try {
        var productId = req.params.productId;
        var product = await productModel.findById(productId);
        product.featured = true;
        product.save();

        res.send({ message: "Added Product to Featured" })

    } catch (error) {
        console.log(error);
    }
});

app.post("/product/removefeatured/:productId", async (req, res) => {
    try {
        var productId = req.params.productId;
        var product = await productModel.findById(productId);
        product.featured = false;
        product.save();

        res.send({ message: "Removed Product From Featured" })

    } catch (error) {
        console.log(error);
    }
});
```

```
app.get("/product/getfeatured", async (req, res) => {
    try {
        var data = await productModel.find({ featured: true });
        res.send(data)

    } catch (error) {
        console.log(error);
    }
});

app.delete("/user/wishlist/delitem/:userId/:productId", async (req, res) => {
    try {
        var userId = req.params.userId;
        var productId = req.params.productId;
        var user = await userModel.findById(userId);
        user.wishlist = user.wishlist.filter(entry => entry.product != productId);
        await user.save();
        res.send({ message: "Product Removed From Wishlist" })

    } catch (error) {
        console.log(error);
    }
});

app.post("/misc/categories/add/:name", async (req, res) => {
    try {
        var name = req.params.name;
        var misc = await miscModel.findOne();
        if (misc.categories.filter(item => item === name).length == 0){
            misc.categories.push(name);
            await misc.save()
            res.send({message: "Category Added"});
        }
        else {
            res.send({message:"Category Already Exists"})
        }
```

```
    } catch (error) {
        console.log(error);
    }
});

app.delete("/misc/categories/delete/:name", async (req, res) => {
    try {
        var name = req.params.name;
        var misc = await miscModel.findOne();
        misc.categories = misc.categories.filter(item => item !== name)
        await misc.save()
        res.send({message: "Category Deleted"})
    } catch (error) {
        console.log(error);
    }
});

app.get("/misc/categories/get", async (req, res) => {
    try {
        res.send((await miscModel.findOne()).categories)
    } catch (error) {
        console.log(error);
    }
});

app.get("/misc", async (req, res) => {
    try {
        res.send(await miscModel.findOne())
    } catch (error) {
        console.log(error);
    }
});

app.use('/images/products', express.static(path.join(__dirname, 'images/products')));

app.listen(PORT, () => {
```

```
    console.log("Port is Up");
});
```

misc.js

```
var mongoose = require("mongoose");
var miscSchema = mongoose.Schema({
    categories: [String]
})

var miscModel = mongoose.model("misc", miscSchema);

module.exports = miscModel;
```

order.js

```
var mongoose = require("mongoose");
var orderSchema = mongoose.Schema({
    name: String,
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'user' },
    amount: Number,
    quantity: Number,
    status: String,
    image: String,
    product: { type: mongoose.Schema.Types.ObjectId, ref: 'product' },
    placedAt: { type: Date, default: () => new Date(Date.now() + 5.5 * 60 * 60 * 1000) }
})

var orderModel = mongoose.model("order", orderSchema);

module.exports = orderModel;
```

product.js

```
var mongoose = require("mongoose");
```

```
const reviewSchema = mongoose.Schema({
    userId: String,
    username: String,
    rating: Number,
    comment: String
});

var productSchema = mongoose.Schema({
    merchant_id: String,
    merchant_name: String,
    name: String,
    description: String,
    price: Number,
    stock: Number,
    category: String,
    keywords: [String],
    image: String,
    featured: Boolean,
    rating: Number,
    reviews: [reviewSchema]
})

var productModel = mongoose.model("product", productSchema);

module.exports = productModel;
```

user.js

```
var mongoose = require("mongoose");

const cartSchema = mongoose.Schema({
    product: {type: mongoose.Schema.Types.ObjectId, ref: "product"},
    quantity: Number
});

const wishlistSchema = mongoose.Schema({
    product: {type: mongoose.Schema.Types.ObjectId, ref: "product"}
```

```
});

var userSchema = mongoose.Schema({
    username: String,
    email: String,
    place: String,
    age: Number,
    password: String,
    role: String,
    cart: [cartSchema],
    wishlist: [wishlistSchema]
})

var userModel = mongoose.model("user", userSchema);

module.exports = userModel;
```

App ScreenShots

FRONTEND

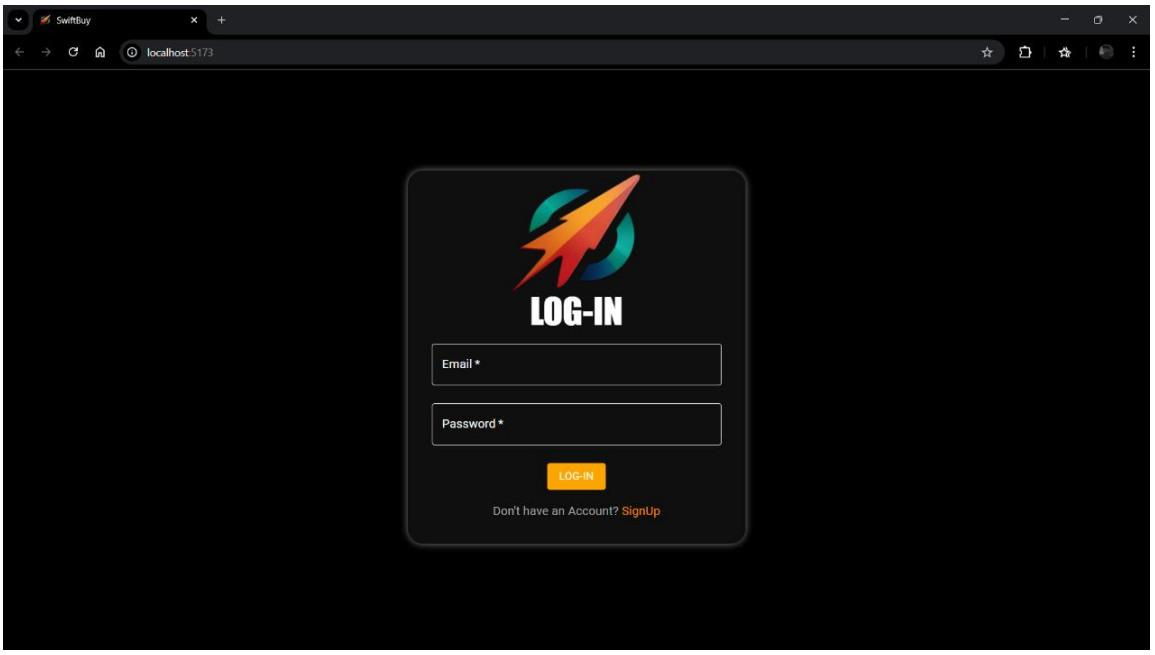

Fig 4 Login Page

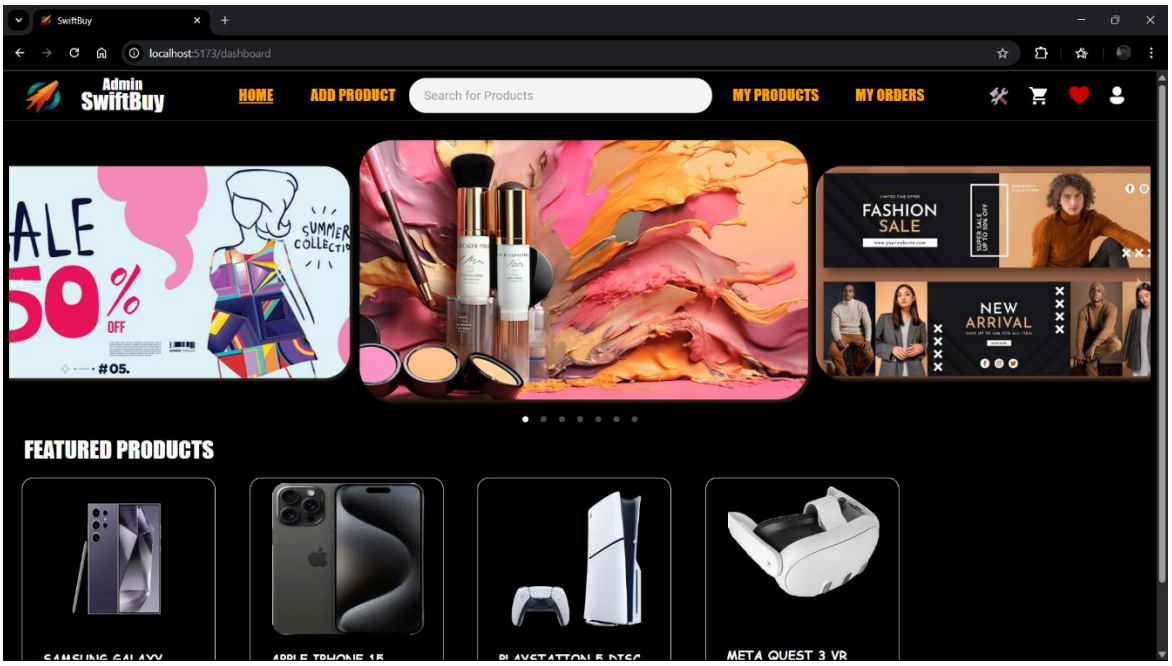

Fig 5 Home Page

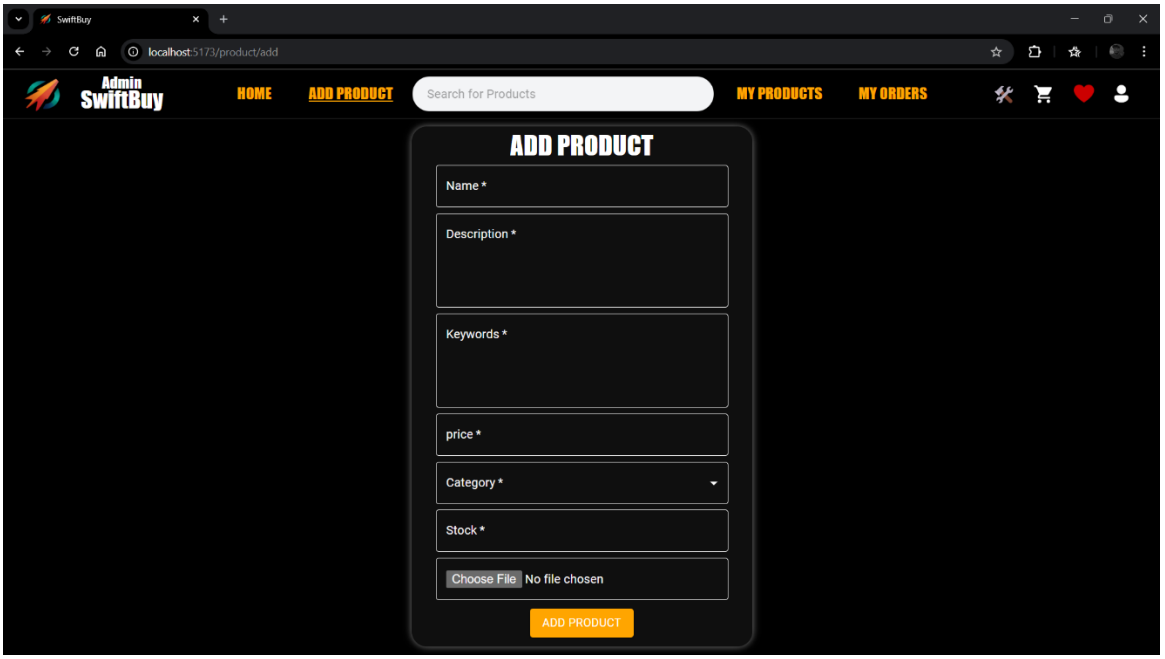

Fig 6 Add Product Page

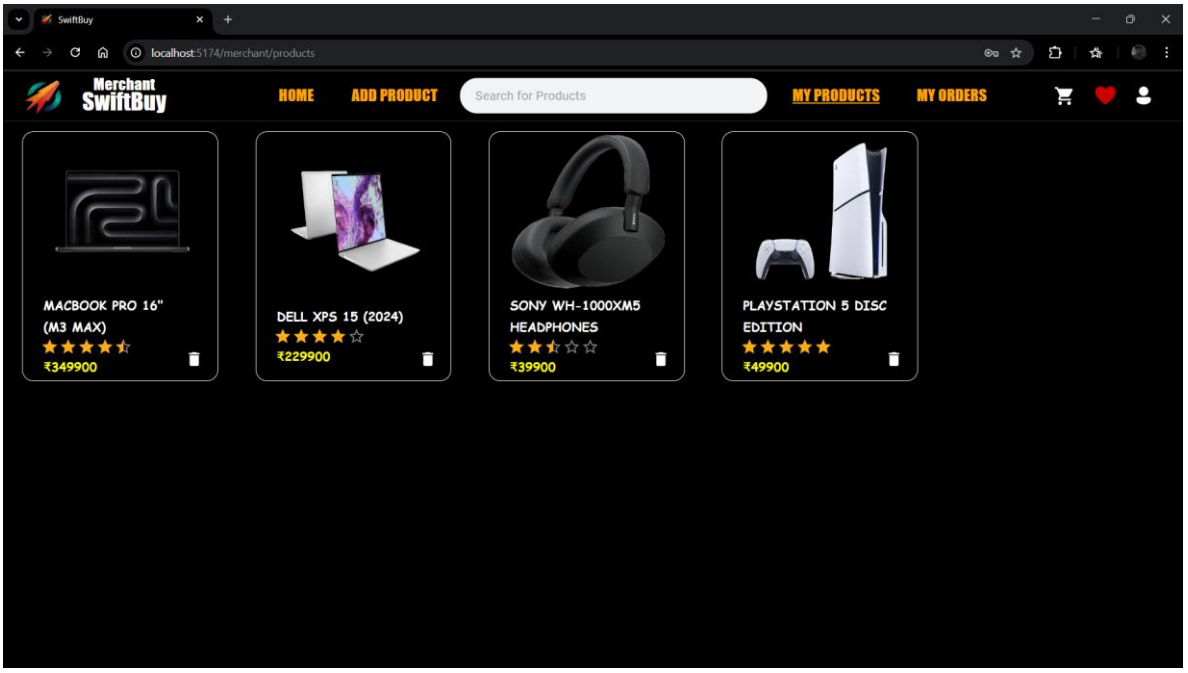

Fig 7 My Products Page

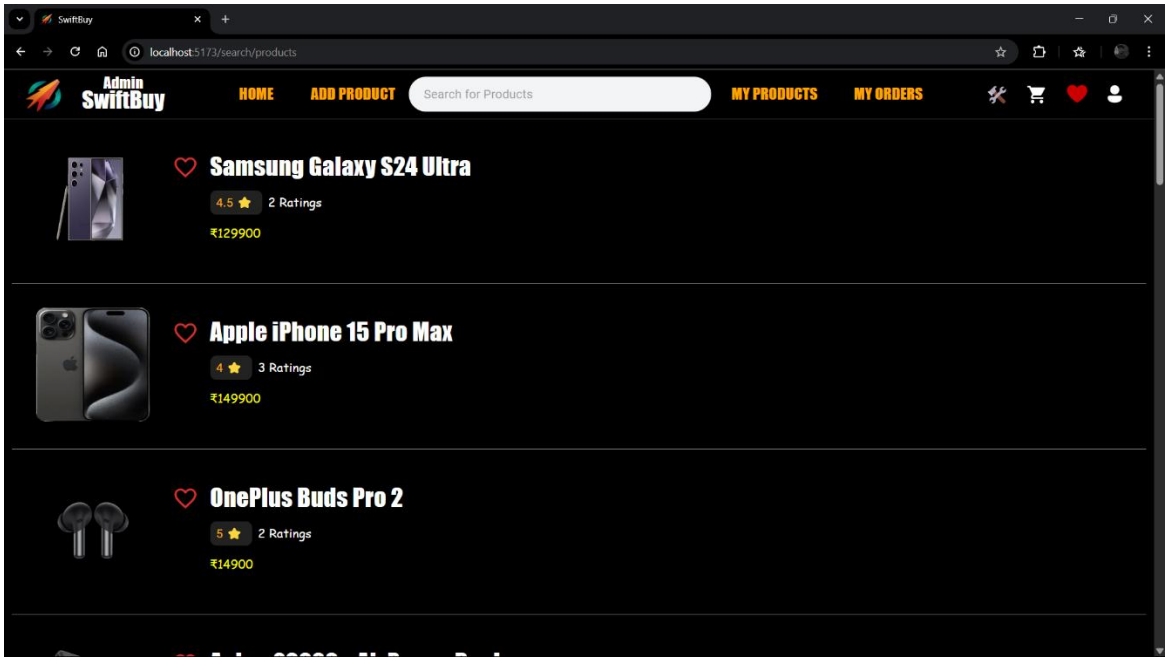

Fig 8 Search Product Page

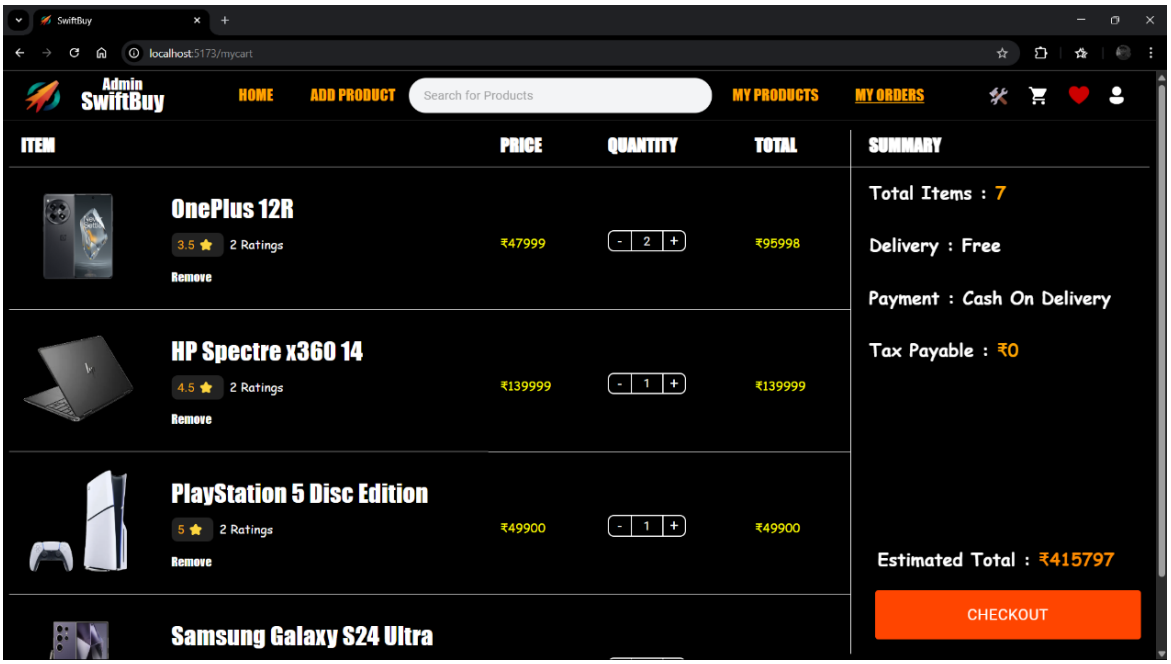

Fig 9 My Cart Page

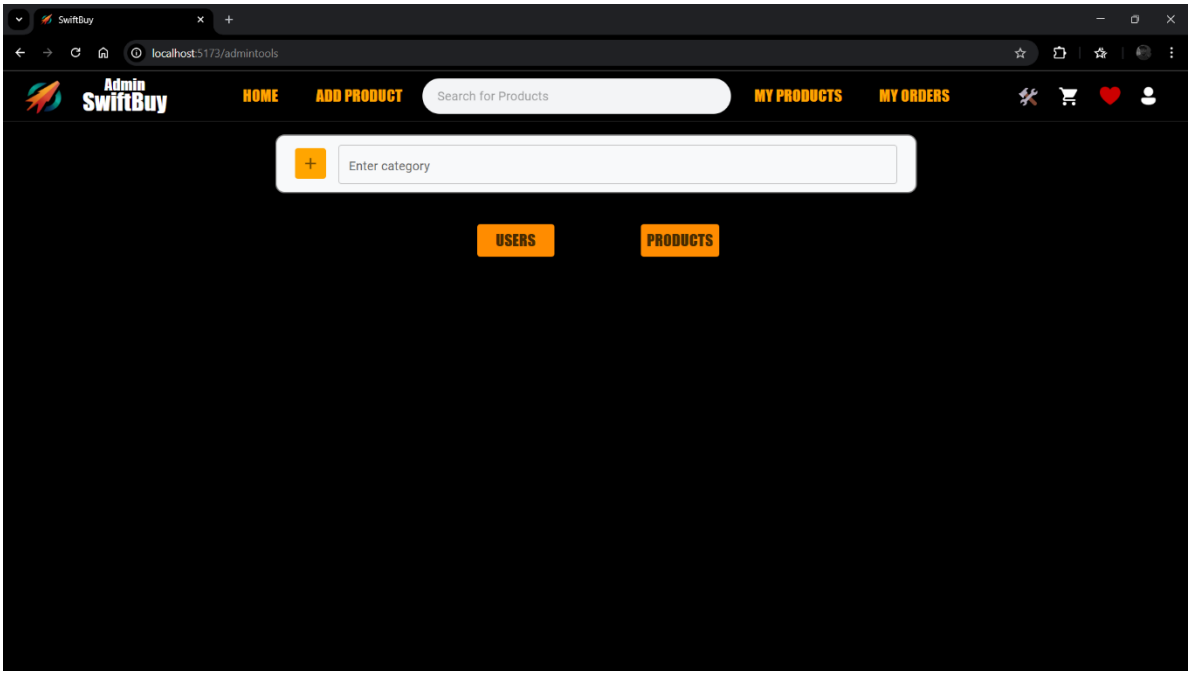

Fig 10 Admin Tools Page

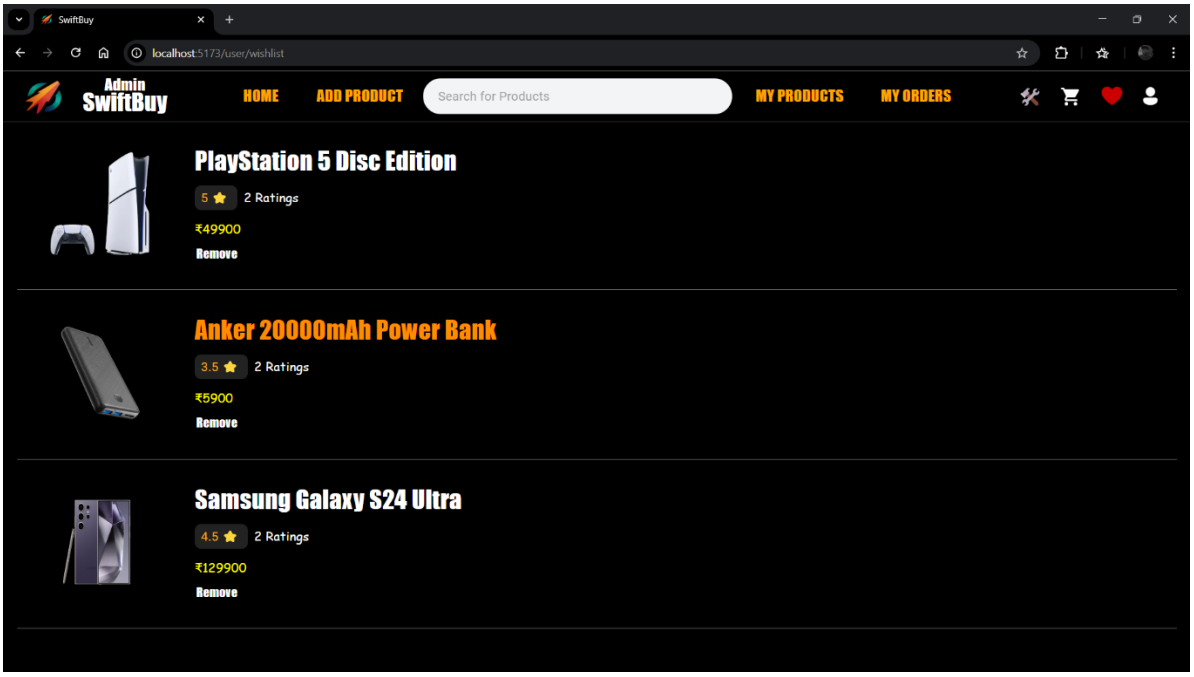

Fig 11 Wishlist Page

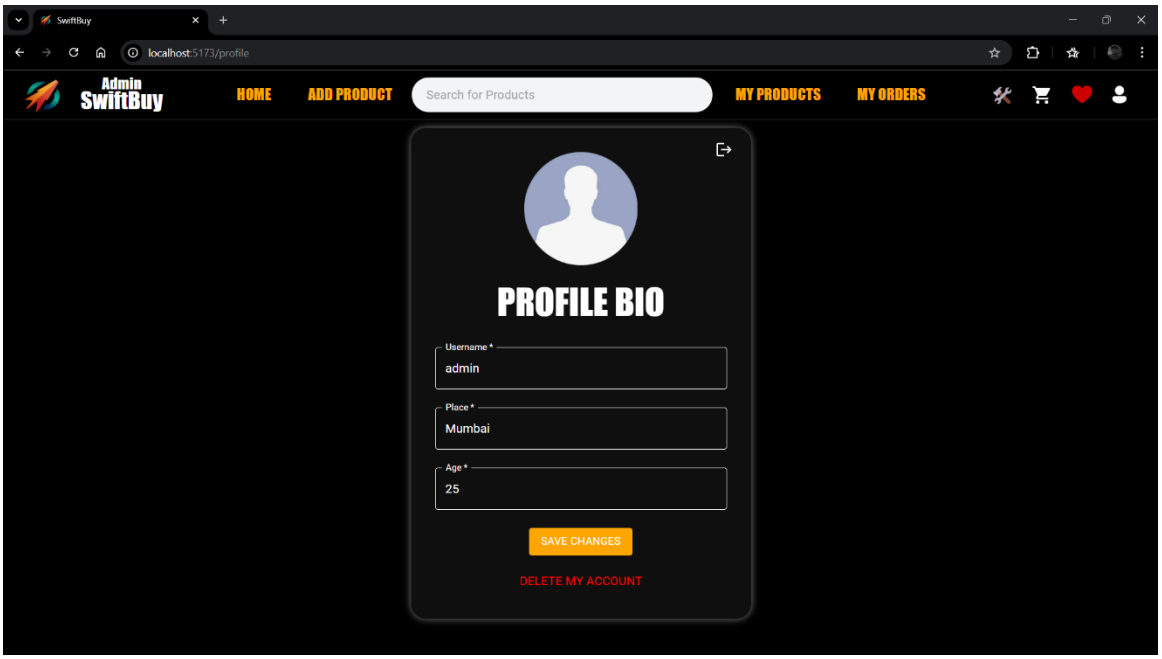

Fig 12 Profile Page

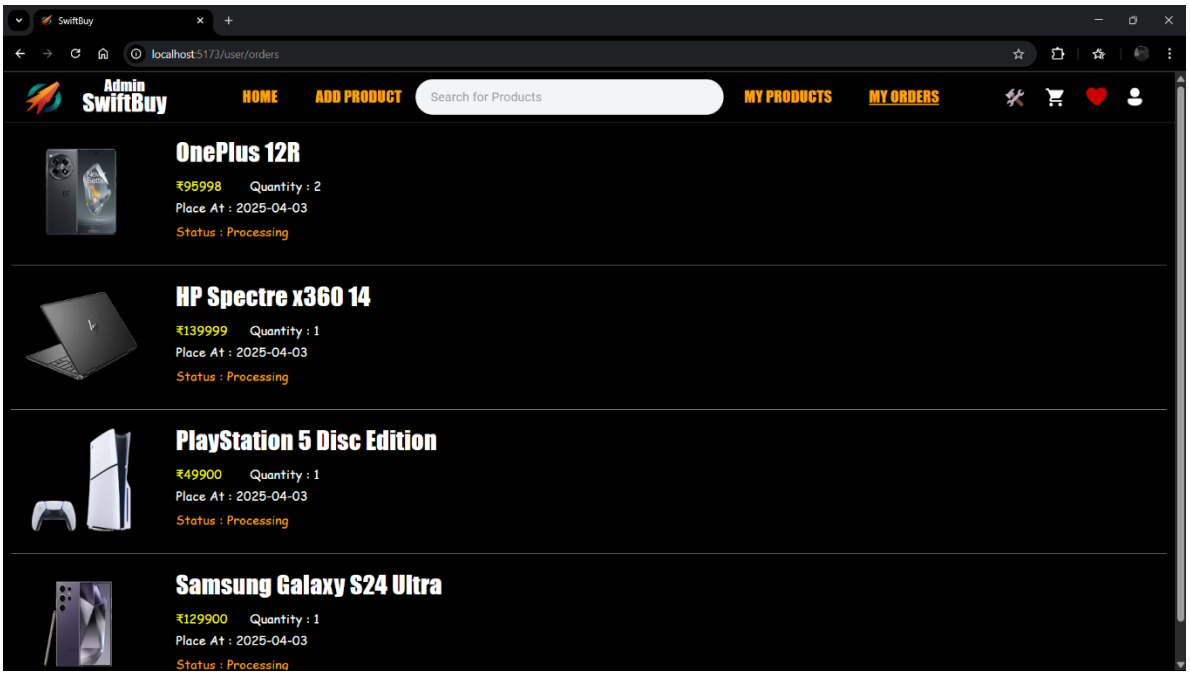

Fig 13 My Orders Page

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

In conclusion, the **SwiftBuy** project represents a transformative step in revolutionizing online shopping experiences. By integrating advanced technologies such as AI-driven recommendation systems, real-time order tracking, and secure payment gateways, **SwiftBuy** ensures a seamless, efficient, and user-centric shopping platform. With its intuitive interface and personalized features, it enhances customer engagement and simplifies the decision-making process. As e-commerce continues to evolve, **SwiftBuy** stands as a pioneering solution, redefining convenience and accessibility in digital retail.

Future Scope:

1. **AI-Powered Enhancements** – Implement advanced AI models for more accurate and personalized recommendations.
2. **Augmented Reality (AR) Integration** – Enable virtual product try-ons for a more immersive shopping experience.
3. **Blockchain Security** – Enhance transaction security and transparency using blockchain technology.
4. **Multilingual & Multicurrency Support** – Expand globally with support for various languages and currencies.
5. **Voice & Chatbot Assistance** – Incorporate AI-driven voice assistants and chatbots for real-time customer support.

REFERENCES

1. Vincent, C., et al. (2023). *Online Product Decision Support Using Sentiment Analysis and Fuzzy Cloud-based Multi-Criteria Model through Multiple E-Commerce Platforms*.
2. Kumar, B., Roy, S., Singh, K. U., Pandey, S. K., & Kumar, A. (2023). *A Static Machine Learning-Based Evaluation Method for Usability and Security Analysis in E-Commerce Websites*.
3. Huang, H., Zavareh, A. A., & Mustafa, M. B. (2023). *Sentiment Analysis in E-Commerce Platforms: A Review of Current Techniques and Future Directions*.
4. Bodduluri, K. C., Palma, F., Kurti, A., Jusufi, I., & Löwenadler, H. (2024). *Exploring the Landscape of Hybrid Recommendation Systems in E-Commerce: A Systematic Literature Review*.
5. Jin, L., & Chen, L. (2024). *Exploring the Impact of Computer Applications on Cross-Border E-Commerce Performance*.
6. Mustafa, G., et al. (2024). *OntoCommerce: Incorporating Ontology and Sequential Pattern Mining for Personalized E-Commerce Recommendations*.
7. Teng, L., Liu, Y., Liu, J., & Song, L. (2024). *End-Cloud Collaboration Framework for Advanced AI Customer Service in E-Commerce*.
8. Gao, Q., Hu, H., & Liu, W. (2024). *E-Commerce Image Enhancement Method Based on Instance Segmentation and Background Replacement*.