PROJECT REPORT

ON

ONLINE E-COMMERCE WEBSITE USING AGILE METHODOLOGY

Submitted By:

Nikhil Harwani

1. ABSTRACT

In the rapidly evolving digital marketplace, having a robust online presence is crucial for retail businesses. This project report details the development of a full-stack Online E-commerce Website utilizing the Agile software development methodology. The primary objective of this system is to provide a seamless platform for customers to browse products, manage shopping carts, and perform secure transactions, while offering administrators tools for inventory and order management.

The adoption of the Agile methodology allowed for iterative development and rapid adaptation to changing market requirements and user feedback. The system features modules for Product Catalog Browsing, User Account Management, Shopping Cart functionality, Secure Payment Gateway integration, and an Admin Dashboard for order fulfillment.

The resulting application provides a scalable, responsive, and user-friendly interface that enhances the online shopping experience. This report documents the entire software development lifecycle, highlighting how Agile practices facilitated the delivery of high-priority commerce features in manageable sprints.

2. INTRODUCTION

2.1 Introduction

E-commerce (Electronic Commerce) consists of the buying and selling of products or services over electronic systems such as the Internet. This project focuses on building a B2C (Business-to-Consumer) web application that allows a retailer to sell goods directly to end-users. The project leverages modern web technologies (MERN/PERN stack) and follows the Agile Scrum framework to ensure flexibility during development.

2.2 Problem Identification

Traditional brick-and-mortar retail faces several limitations that an online platform addresses:

- **Geographical Limitations:** Customer base is limited to the physical location of the shop.
- **Time Constraints:** Sales can only occur during business opening hours.
- **Inventory Management:** Manual tracking of stock is error-prone and inefficient.
- **Customer Reach:** Inability to easily market products to a wider audience.

2.3 Need of the Project

To overcome these limitations, a centralized E-commerce platform is required to:

1. Provide 24/7 availability for customers to browse and purchase goods.
2. Automate inventory updates immediately upon sale.
3. Securely process online payments through integrated gateways.
4. Provide an administrative interface to track orders and shipping status.

2.4 Project Scheduling (Agile Timeline)

The project was divided into 4 Sprints, each lasting two weeks:

Phase	**Duration**	**Deliverable**
Sprint 1	2 Weeks	Requirement Analysis, DB Design, Product Catalog Display
Sprint 2	2 Weeks	User Registration/Login, Shopping Cart Functionality
Sprint 3	2 Weeks	Payment Gateway Integration, Order Creation
Sprint 4	2 Weeks	Admin Dashboard (CRUD Products), Testing, Deployment

2.5 Objectives

- To design and develop a responsive E-commerce web application.
- To implement secure user authentication and authorization.
- To integrate a reliable third-party payment processor (e.g., Stripe or PayPal).
- To provide a backend management system for inventory control.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Purpose

The purpose of this SRS document is to outline the functional and non-functional requirements of the E-commerce website. It defines the scope for developers and QA testers to ensure the final product enables smooth online transactions.

3.2 Scope

The system covers the following modules: **Storefront** (Product browsing, search, cart management), **Customer Accounts** (Order history, profile management), and **Admin Panel** (Product CRUD, order status updates). It does not currently cover multi-vendor marketplace features or advanced AI product recommendations.

3.3 System Requirements

Hardware Requirements (Minimum/Client Side)

- **Processor:** Modern dual-core processor or equivalent smartphone CPU
- **RAM:** 4 GB
- **Internet Connection:** Stable broadband or 4G/5G connection
- **Monitor:** Any standard display (responsive design enabled)

Software Requirements

- **Operating System:** Windows, macOS, Linux, or mobile OS (Android/iOS)
- **Frontend:** HTML5, CSS3, JavaScript (React.js or Vue.js)
- **Backend:** Node.js / Express.js (or Python Django)
- **Database:** MongoDB (NoSQL) or PostgreSQL (SQL)
- **Payment Gateway:** Stripe API or PayPal Developer SDK

3.4 Tools Used

- **VS Code:** Integrated Development Environment.

- **Git/GitHub:** Version Control for codebase management.
- **Jira/Asana:** Agile Project Management (User story tracking).
- **Postman:** Testing RESTful API endpoints for cart and orders.

3.5 Software Process Model

Agile Model (Scrum): The project utilizes the Agile Scrum methodology. This allows for prioritizing features based on business value (e.g., prioritizing the checkout flow over advanced user profiles) and releasing functional increments of the store at the end of every sprint.

4. SYSTEM DESIGN

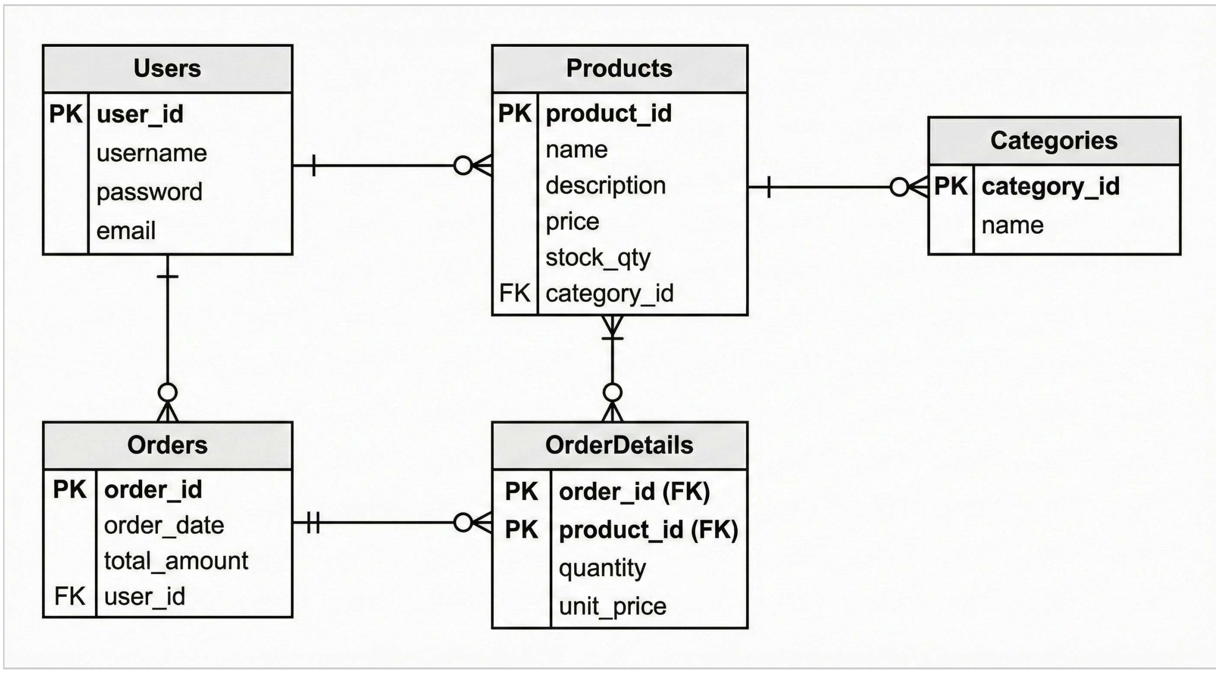
4.1 Data Dictionary

Table: Products

Field	**Type**	**Description**
product_id	INT (PK)	Unique Identifier for product
name	VARCHAR	Name of the item
price	DECIMAL(10,2)	Unit price of the item
stock_qty	INT	Current inventory level
category_id	INT (FK)	Reference to Category table

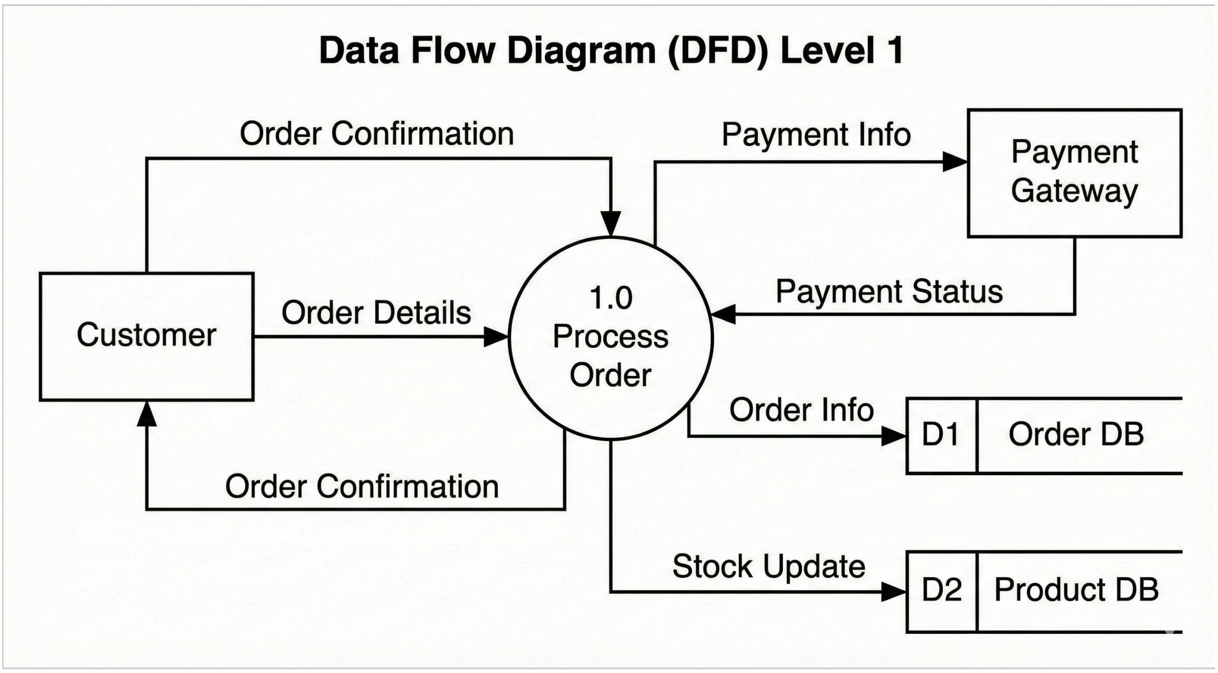
4.2 ER Diagram

Entity Relationship Diagram (E-commerce)



4.3 Data Flow Diagram (DFD)

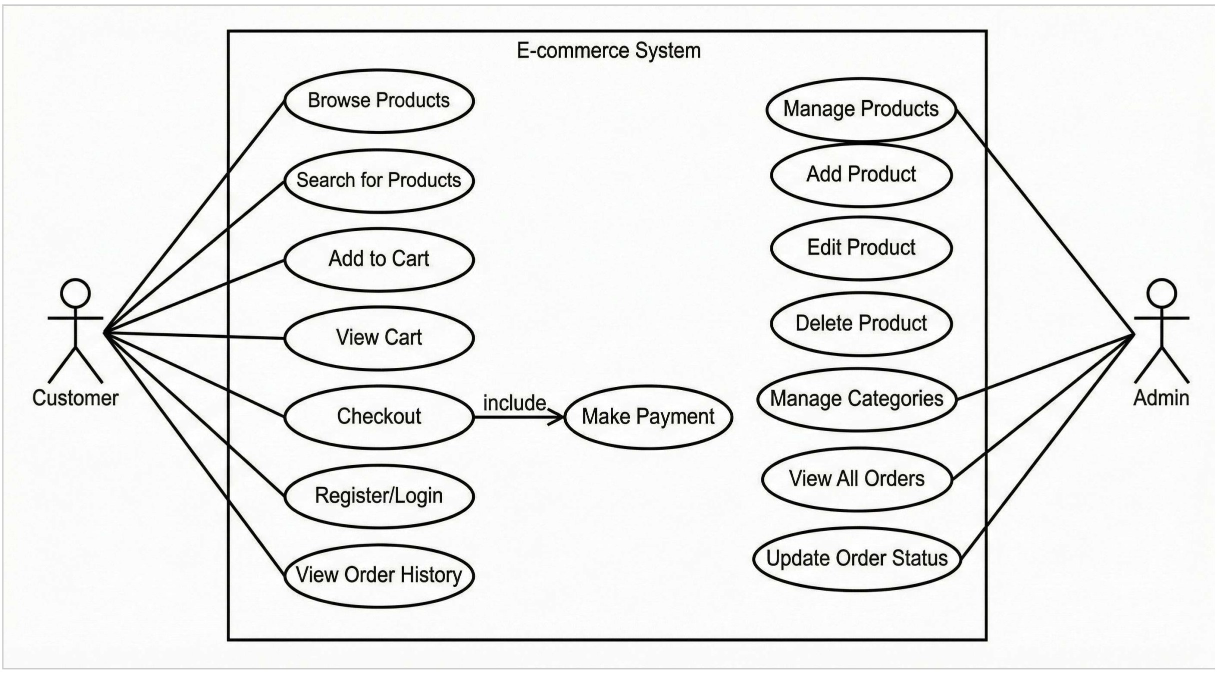
Data Flow Diagram (Order Process Level 1)



4.4 UML Diagrams

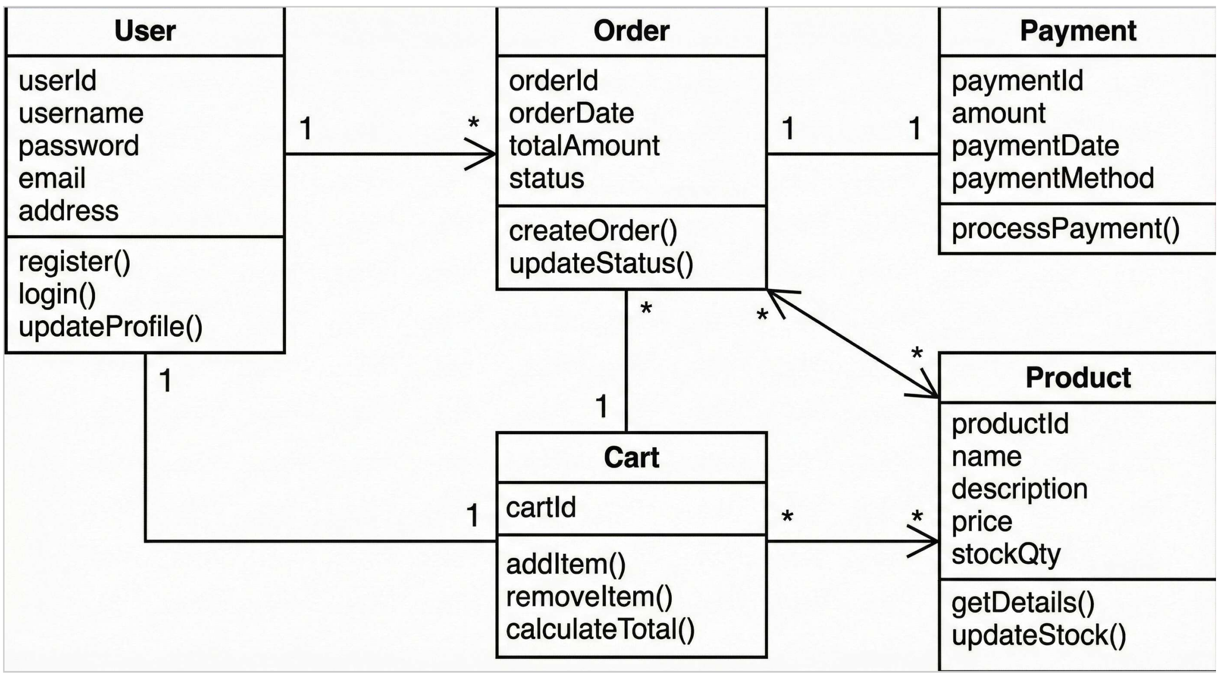
Use Case Diagram

Use Case Diagram (Customer & Admin)



Class Diagram

Class Diagram



5. IMPLEMENTATION

5.1 Program Code

Below are selected core snippets of the application logic related to e-commerce functionality.

Backend Controller (Product Catalog)

```
// ProductController.js
const Product = require('../models/Product');

// Get all products with pagination and filtering
exports.getProducts = async (req, res) => {
  try {
    const pageSize = 10;
    const page = Number(req.query.pageNumber) || 1;
    const keyword = req.query.keyword ? {
      name: { $regex: req.query.keyword, $options: 'i' }
    } : {};

    const count = await Product.countDocuments({ ...keyword });
    const products = await Product.find({ ...keyword })
      .limit(pageSize)
      .skip(pageSize * (page - 1));

    res.json({ products, page, pages: Math.ceil(count / pageSize) });
  } catch (error) {
    res.status(500).json({ message: "Server Error fetching products" });
  }
};
```

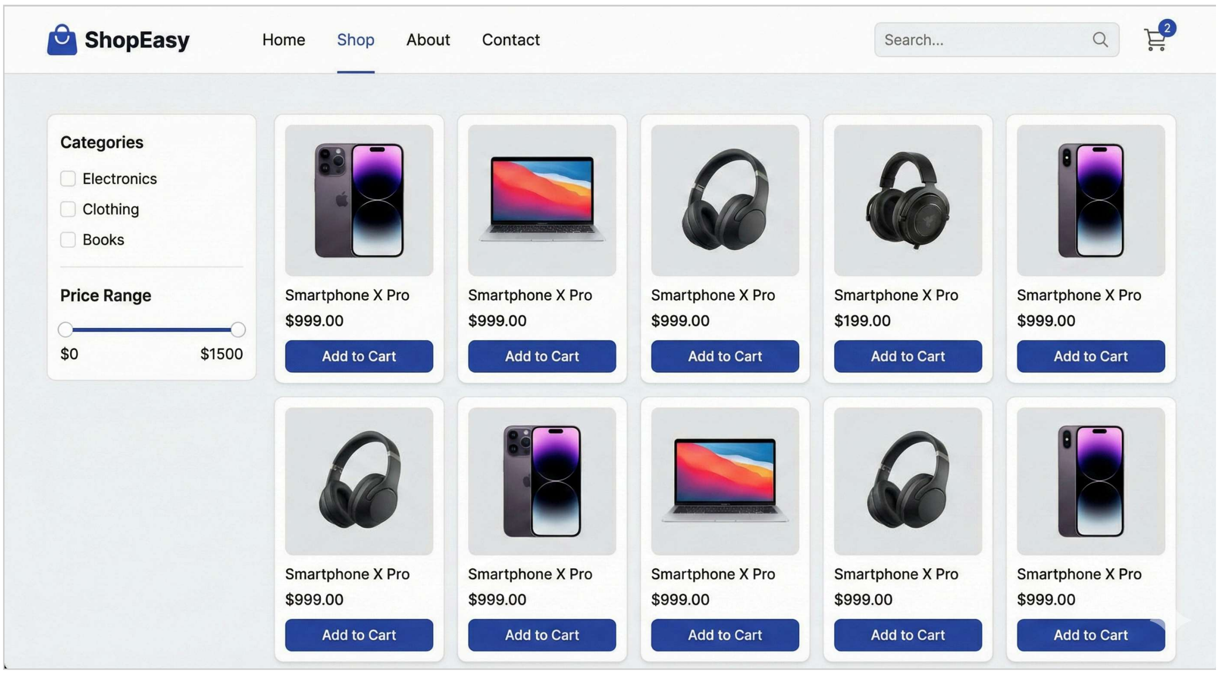
Backend Model (Order Schema)

```
// OrderModel.js
const mongoose = require('mongoose');

const orderSchema = mongoose.Schema({
  user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required:
true },
  orderItems: [{
    name: { type: String, required: true },
    qty: { type: Number, required: true },
    image: { type: String, required: true },
    price: { type: Number, required: true },
    product: { type: mongoose.Schema.Types.ObjectId, ref: 'Product',
required: true }
  }],
  shippingAddress: {
    address: { type: String, required: true },
    city: { type: String, required: true },
    postalCode: { type: String, required: true },
    country: { type: String, required: true }
  },
  paymentMethod: { type: String, required: true },
  totalPrice: { type: Number, required: true, default: 0.0 },
  isPaid: { type: Boolean, required: true, default: false },
  paidAt: { type: Date }
}, { timestamps: true });

module.exports = mongoose.model('Order', orderSchema);
```

5.2 Output Screens

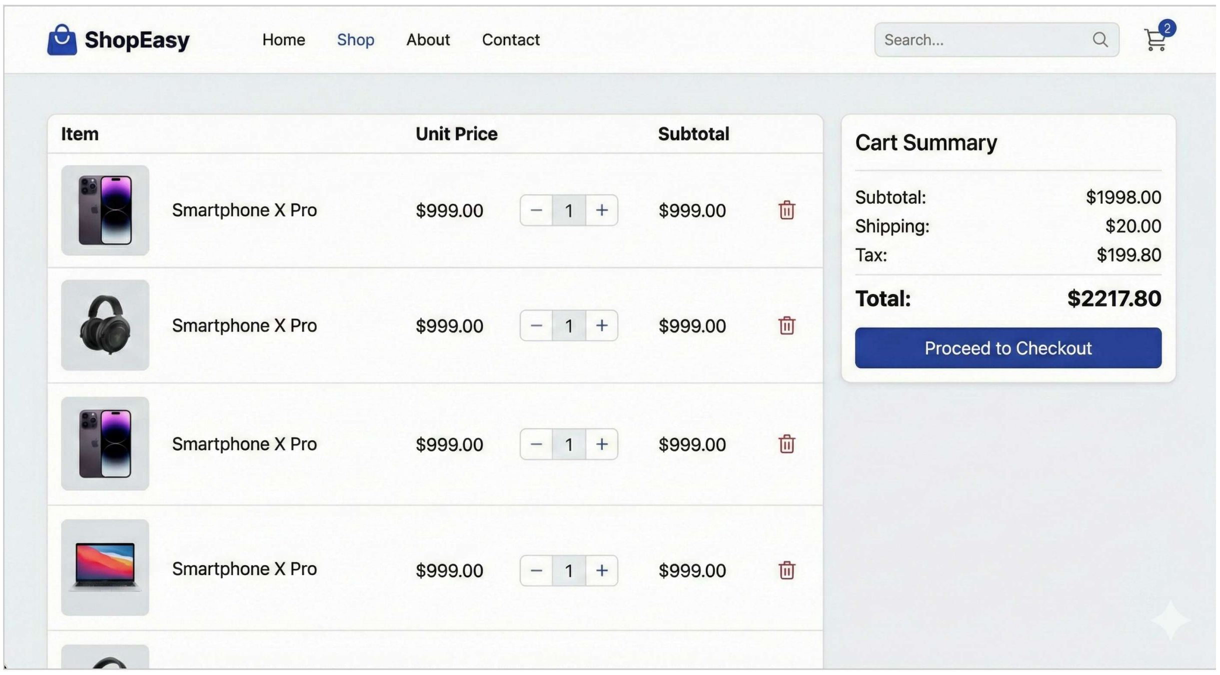
Home Page / Product Listing:

Product Catalog Screenshot



Shopping Cart & Checkout:

Shopping Cart Screenshot



Item	Unit Price		Subtotal
Smartphone X Pro	$999.00	− 1 +	$999.00
Smartphone X Pro	$999.00	− 1 +	$999.00
Smartphone X Pro	$999.00	− 1 +	$999.00
Smartphone X Pro	$999.00	− 1 +	$999.00

Cart Summary

Subtotal:	$1998.00
Shipping:	$20.00
Tax:	$199.80
Total:	**$2217.80**

6. TESTING

6.1 Test Data & Strategy

Testing was conducted continuously throughout the Agile sprints. The strategy included Unit Testing for individual components (e.g., cart price calculation logic) and Integration Testing for end-to-end flows (e.g., completing a purchase).

6.2 Test Results

Test Case ID	**Description**	**Input Data**	**Expected Output**	**Status**
TC_01	Product Search Functionality	Search Term: "Smartphone"	List displays only smartphone items	PASS
TC_02	Add Item to Cart	Click "Add to Cart" on Product A	Cart icon shows '1' item; Cart total updates	PASS
TC_03	Payment Gateway Processing	Valid Test Credit Card Details	Order confirmed; Redirect to Success Page	PASS
TC_04	Admin Stock Update	Admin changes stock of Item X from 10 to 5	Frontend shows "Only 5 left" for Item X	PASS

7. USER MANUAL

7.1 How to use project guidelines

1. **Installation:** Clone repository. Ensure Node.js and MongoDB exist. Create a `.env` file with database URI and Payment Gateway API keys.
2. **Setup:** Run `npm install` in both frontend and backend directories.
3. **Data Seeding:** Run `npm run data:import` to populate the database with initial sample products.
4. **Execution:** Run `npm run dev` to launch frontend and backend concurrently.
5. **Access:** Navigate to `http://localhost:3000` to browse the store.

7.2 Screen Layouts and Description

Navigation Bar: Contains links to Home, Shop Categories, Search bar, Cart icon (with item count), and User Profile/Login status.
Product Grid: Displays product thumbnails, titles, prices, and "Add to Cart" buttons.
Admin Sidebar: (Visible only to admin users) Provides access to Product Management, Order List, and User List.

8. PROJECT APPLICATIONS AND LIMITATIONS

Applications

- **Retail Businesses:** Small to medium businesses wanting to establish an online sales channel.
- **Digital Goods Sellers:** Platforms for selling items like e-books, software, or art.
- **Wholesale Distributors:** B2B portals for bulk ordering by pre-approved clients.

Limitations

- The application requires constant internet connectivity to process transactions.
- Currently only supports a single currency (USD) and language (English).
- Dependent on the uptime and availability of the third-party payment gateway provider.

9. CONCLUSION AND FUTURE ENHANCEMENT

The "Online E-commerce Website" project successfully demonstrated the effectiveness of the Agile methodology in developing complex transactional systems. By utilizing sprints, the team was able to deliver essential features like the product catalog and secure checkout early, allowing for testing and refinement. The final product provides a solid foundation for digital retail operations.

Future Enhancements:

- **Recommendation Engine:** Implementing AI/ML algorithms to suggest related products based on browsing history.
- **Multi-Language & Currency Support:** Localization to target a broader international audience.
- **Mobile Application:** Developing native iOS and Android apps using React Native for better mobile engagement.
- **Wishlist Functionality:** Allowing users to save items for future purchase consideration.

10. BIBLIOGRAPHY & REFERENCES

Books:

- Freeman, A. (2019). *Pro React 16*. Apress.
- Haverbeke, M. (2018). *Eloquent JavaScript, 3rd Edition*. No Starch Press.
- Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Websites & Documentation:

- React Documentation (https://reactjs.org/docs/getting-started.html)
- Stripe API Reference (https://stripe.com/docs/api)
- MongoDB Manual (https://docs.mongodb.com/)
- Agile Alliance (https://www.agilealliance.org/)