Noakhali Science & Technology University
Department of Computer Science & Telecommunication Engineering

A Project Report on

Design and Development of "ShopVerse : Online Shopping System"

by

Mohammad Borhan Uddin
ID: ASH2101008M
Mohammad Billal Hossain
ID: MUH2101028M

Supervised by
Ratnadip Kuri
Assistant Professor
Department of Computer Science & Telecommunication Engineering

Date : 18th September, 2025

DECLARATION

We hereby declare that this project has been completed by us under the supervision of Mr. Ratnadip Kuri, Assistant Professor, Department of Computer Science and Telecommunication Engineering, Noakhali Science and Technology University. We also affirm that neither this project nor any part of it has been submitted elsewhere for the award of any degree.

Mohammad Borhan Uddin
ID: ASH2101008M

Mohammad Billal Hossain
ID: MUH2101028M

Abstract

The **ShopVerse** project is an online shopping system designed to enable organizations and shops to sell their products digitally, replacing traditional offline sales methods with a fully automated, efficient, and scalable platform. The system provides a seamless shopping experience for customers, allowing them to browse, search, add products to their cart, and make secure payments via the Stripe payment gateway. Administrators can efficiently manage products, including adding, updating, and deleting items organized under nested categories, while maintaining accurate inventory and order tracking. Product images are securely stored and handled using the Pinata cloud storage service. The platform implements role-based access control to distinguish between admin and customer functionalities, ensuring secure and restricted access. Built using Next.js for both frontend and backend operations, Prisma ORM for database management, MySQL for relational data storage, and ShadCN for user interface components, the system is designed for scalability, performance, and maintainability. Overall, ShopVerse provides a reliable foundation for digital commerce while allowing future enhancements such as multi-vendor support, advanced analytics, and mobile application integration.

Acknowledgment

We, the members of this project team, would like to express our sincere gratitude to all those who have contributed to the successful completion of this endeavor.

First and foremost, we extend our deepest appreciation to our esteemed Assistant Professor, Mr. Ratnadip Kuri, for her unwavering guidance and mentorship throughout this project. Her insightful feedback, constant encouragement, and expert direction were instrumental in shaping the direction and quality of our work.

We also acknowledge the invaluable contributions of our team members, Mohammad Borhan Uddin (ASH2101008M) and Mohammad Billal Hossain (MUH2101028M). The dedication, collaborative spirit, and collective effort of the team were key to overcoming the challenges encountered during the development process.

Additionally, we would like to extend our heartfelt thanks to the NSTU Administration and the Department of Computer Science and Telecommunication Engineering (CSTE) for providing the necessary resources, including lab support, and for ensuring the availability of the ACM lab 24/7, which greatly contributed to the successful completion of this project. Their cooperation in facilitating our work was vital in making this project a reality.

Finally, we express our gratitude to our families and friends for their constant encouragement, understanding, and moral support. Their patience and motivation have been a source of strength and inspiration throughout the course of this project. Above all, we are profoundly grateful to Almighty Allah for His countless blessings, guidance, and support, which have enabled us to complete this project successfully.

Contents

1 Introduction

1.1 Introduction to ShopVerse: Online Shopping System

In the modern era of digital transformation, online shopping systems have become a vital component of the retail and business ecosystem. Traditional physical stores are limited by geographical boundaries, time constraints, and manual processes that often reduce efficiency and customer satisfaction. With the rapid growth of internet accessibility, customers increasingly prefer purchasing products online due to convenience, availability of information, and secure payment options.

The proposed system, **ShopVerse: Online Shopping System**, is specifically designed for an organization or shop that wants to sell its products online through a centralized digital platform. Unlike large-scale marketplaces that cater to multiple vendors, this system focuses on serving the needs of a single shop or organization, allowing it to manage its inventory, categorize products, and handle customer orders seamlessly.

One of the unique features of ShopVerse is the support for nested categories, which ensures a well-structured and hierarchical arrangement of products, making it easier for customers to browse and search. Additionally, the system integrates with a cloud storage service (Pinata) for secure and efficient product image management, and a payment gateway (Stripe) to ensure safe and reliable financial transactions.

By combining effective database management, user-friendly design, and secure online payments, ShopVerse ensures an efficient and reliable shopping experience for customers while simplifying management tasks for the shop.

1.2 Objectives

The main objectives of the ShopVerse are:

1. Digitization of Sales Process: Replace traditional manual sales with a fully automated online shopping system.
2. Efficient Product Management: Allow the admin to add, update, and delete products under nested categories for easier navigation.
3. Seamless Shopping Experience for Customers: Enable browsing, searching, adding to cart, placing orders, and making secure payments online.
4. Order and Inventory Management: Track customer orders, update order status, and maintain accurate product stock levels.
5. Secure Payment Integration: Ensure safe and reliable transactions through Stripe payment gateway.

6. Cloud-Based Image Handling: Store and manage product images securely using Pinata.
7. Role-Based Access Control: Maintain Admin and Customer roles with restricted access to system functionalities.
8. Scalability and Flexibility: Design the system to support future expansion like multi-vendor integration, analytics, and mobile apps.

2 Features and System Highlights

2.1 Key Features

The **ShopVerse: Online Shopping System** is designed to provide a seamless online shopping experience for both admins and customers. Its main features include:

- **User Registration and Authentication**: Secure sign-up and login process for both admins and customers, ensuring safe access to the system.
- **Product Management**: Admins can add, update, and delete products, organize them under nested categories, and manage stock levels efficiently.
- **Shopping Cart and Checkout**: Customers can add products to the cart, modify quantities, and place orders with a smooth checkout process.
- **Secure Payment Integration**: Stripe is used to process payments securely and reliably.
- **Order Management**: Admins can track and update order status (Pending, Processing, Shipped, Delivered, Cancelled), while customers can monitor their orders.
- **Cloud-Based Image Handling**: Product images are stored and managed securely using Pinata, ensuring high availability and scalability.
- **Responsive and Intuitive UI**: The interface is developed with Next.js, ShadCN components, and Tailwind CSS, providing a visually appealing and responsive design across devices.
- **Role-Based Access Control**: Two distinct roles (Admin and Customer) maintain system security and restrict access to sensitive functionalities.
- **Scalability and Extensibility**: The system architecture allows future enhancements such as multi-vendor support, analytics dashboards, and mobile applications.

2.2 Additional Capabilities

- **Search and Filtering**: Basic search and category-based product filtering are implemented, with room for advanced search capabilities in future updates.
- **Data Integrity and Reliability**: Use of MySQL with Prisma ORM ensures consistent and reliable handling of user, product, and order data.
- **SEO and Performance Optimization**: Next.js enables server-side rendering (SSR) and static site generation (SSG), improving page load speed and SEO.
- **Future-Proof Design**: Modular architecture allows easy addition of new features without disrupting existing functionalities.

3 Methodology and Database Design

3.1 Definition

Methodology refers to a structured set of principles, methods, and processes used to guide the development and execution of a project or system.

3.2 Project Selection and Planning

We identified the need for an efficient online shopping system to enable organizations and shops to sell products digitally. We assessed the feasibility of the project and defined its scope, ensuring it addressed both admin and customer requirements effectively.

3.3 Requirement Gathering and Analysis

The first phase of the project involved gathering detailed requirements from key stakeholders, including shop owners, administrators, and customers.

- Surveys and interviews were conducted to understand users' needs, challenges, and desired features for the online shopping platform.
- Functional requirements included product management (add, update, delete), category management, browsing and searching products, cart management, order placement, and secure payment processing via Stripe.

- Non-functional requirements such as system security, scalability, responsiveness, and cloud-based image handling using Pinata were also considered.
- Role-based access control was implemented to differentiate between Admin and Customer functionalities.

3.4 Use Case Modeling

A use case model describes user interactions and activities within a system, outlining the actions performed by each type of user. The key elements include an actor, an event, and a use case.

In the ShopVerse project, there are two main actors:

- **Admin**
- **Customer**

The admin manages product and category data, views and updates orders, and oversees overall system operations. Customers can browse products, add them to the cart, place orders, and make payments.

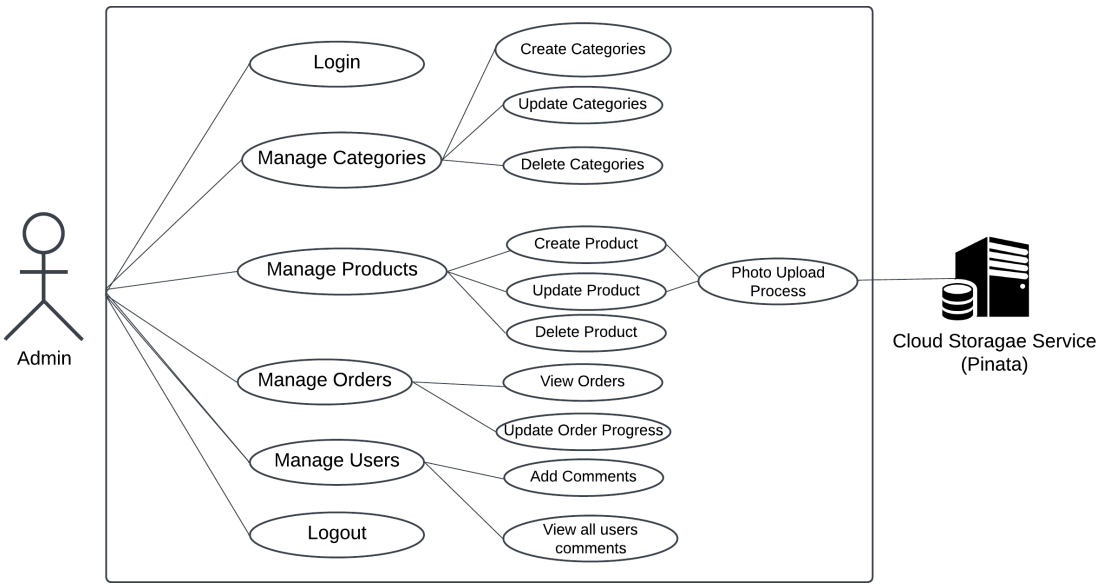


Figure 1: Use Case Diagram of Admin

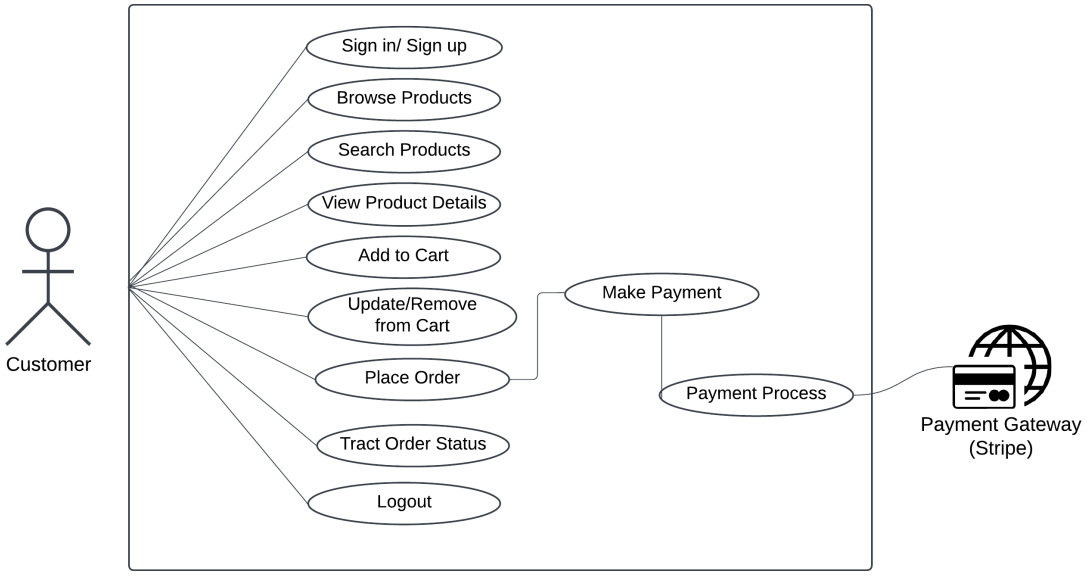


Figure 2: Use Case Diagram of Customer

4 Database Design and Development

We designed a relational database to store product data, categories, user information, orders, and payment details. Tables were created with proper relationships to ensure data integrity and scalability, allowing the system to efficiently handle multiple products, customers, and orders. The database structure supports future expansion, including additional features such as analytics, multi-vendor integration, and mobile application support.

4.0.1 Database Table Description

User Table

- **id**: VARCHAR(191) – Unique user ID (Primary Key)
- **name**: VARCHAR(191) – Name of the user
- **email**: VARCHAR(191) – User email (Unique)
- **password**: VARCHAR(191) – Encrypted password
- **address**: VARCHAR(191) – Shipping address (nullable)
- **createdAt**: DATETIME(3) – Account creation timestamp
- **updatedAt**: DATETIME(3) – Last account update timestamp
- **role**: ENUM('USER','ADMIN') – Role of the user, default 'USER'

Category Table

- **id**: INT(11) – Unique category ID (Primary Key, Auto Increment)
- **name**: VARCHAR(191) – Name of the category (Unique)
- **slug**: VARCHAR(191) – URL-friendly name (Unique)
- **parentId**: INT(11) – Parent category ID (0 for root categories)
- **createdAt**: DATETIME(3) – Timestamp when category was created
- **updatedAt**: DATETIME(3) – Timestamp when category was last updated

Product Table

- **id**: INT(11) – Unique product ID (Primary Key, Auto Increment)
- **name**: VARCHAR(191) – Product name
- **slug**: VARCHAR(191) – URL-friendly name (Unique)
- **description**: VARCHAR(191) – Short description of the product
- **price**: INT(11) – Price of the product
- **stock**: INT(11) – Available quantity
- **categoryId**: INT(11) – Foreign Key referencing Category ID
- **createdAt**: DATETIME(3) – Timestamp when product was created
- **updatedAt**: DATETIME(3) – Timestamp when product was last updated
- **image**: VARCHAR(191) – Image file reference stored on Pinata

Cart Table

- **userId**: VARCHAR(191) – Foreign Key referencing User ID
- **productId**: INT(11) – Foreign Key referencing Product ID
- **quantity**: INT(11) – Number of units, default 1
- **createdAt**: DATETIME(3) – Timestamp when item added to cart
- **updatedAt**: DATETIME(3) – Timestamp when item was last updated

Order Table

- **id**: INT(11) – Unique order ID (Primary Key, Auto Increment)
- **userId**: VARCHAR(191) – Foreign Key referencing User ID
- **total**: DOUBLE – Total order amount
- **status**: ENUM('PENDING','PROCESSING','SHIPPED','DELIVERED','CANCELLED') – Order status, default 'PENDING'
- **createdAt**: DATETIME(3) – Timestamp when order was created
- **updatedAt**: DATETIME(3) – Timestamp when order was last updated
- **address**: VARCHAR(191) – Shipping address
- **paymentId**: VARCHAR(191) – Stripe payment ID (nullable)

OrderItem Table

- **id**: INT(11) – Unique ID for the order item (Primary Key, Auto Increment)
- **orderId**: INT(11) – Foreign Key referencing Order ID
- **productId**: INT(11) – Foreign Key referencing Product ID
- **quantity**: INT(11) – Number of units ordered, default 1
- **price**: DOUBLE – Price per unit at the time of order

4.0.2 ER Diagram

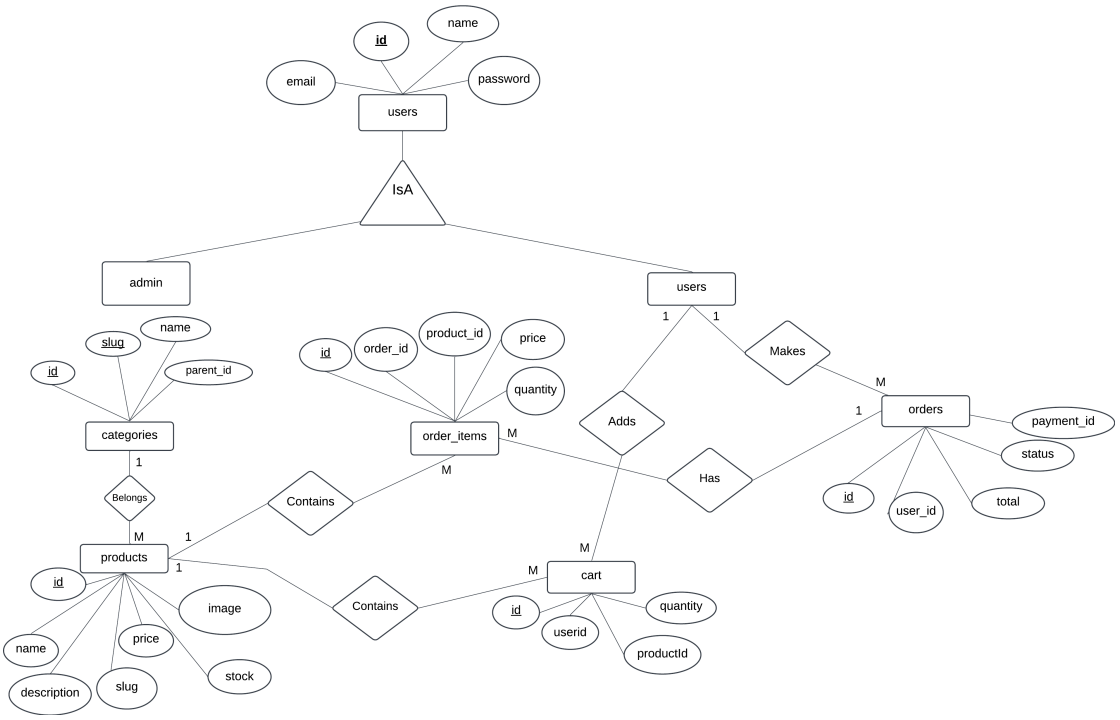


Figure 3: ER Diagram

4.0.3 Relationships of tables

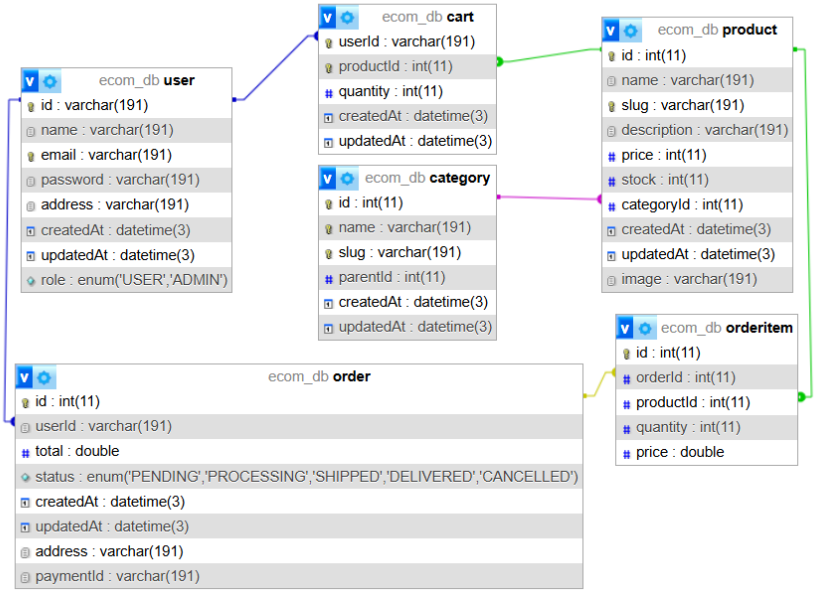


Figure 4: Relationship of tables

4.1 Technology and Programming Language Selection

Selecting the right technologies is crucial for ensuring system scalability, performance, and user experience. In this project, we chose tools that align with the project's goals, providing an efficient, maintainable, and user-friendly solution.

4.1.1 Frontend

We selected **Next.js** for front-end development due to its server-side rendering capabilities, optimized performance, and SEO friendliness. Next.js combines React's component-based architecture with powerful rendering strategies:

- **Component-based UI development:** Utilizes React for building interactive and dynamic user interfaces.
- **Client-Side Rendering (CSR):** Renders parts of the application directly in the user's browser after the initial page load, providing smooth interactions.
- **Static Site Generation (SSG):** Pre-renders pages at build time, generating fast-loading static HTML files for improved performance and SEO.

To enhance the UI, we integrated **ShadCN** for pre-built, modern components and **Tailwind CSS** for flexible styling and responsive layouts. **Zod** was used for client-side schema validation, ensuring reliable data handling and form validation.

4.1.2 Backend (via Next.js)

The backend functionality is fully handled by **Next.js**, eliminating the need for separate server frameworks:

- **Server-Side Rendering (SSR):** Renders pages on the server for each request, improving initial load performance and SEO.
- **API Routes:** Creates server-side API endpoints within the Next.js project to handle requests, manage routing, and interact with the database.
- **Middleware:** Executes code on the server before a request reaches its intended handler, useful for authentication, logging, or request pre-processing.

Prisma ORM was used to communicate with the database, enabling type-safe, efficient, and maintainable database queries.

4.1.3 Databases

We selected **MySQL** as the relational database for storing structured data such as users, products, categories, orders, and payments. MySQL ensures reliability, data consistency, and scalability, effectively managing complex relationships between tables while maintaining data integrity.

4.1.4 Payment and Cloud Services

For secure and reliable transactions, we integrated the **Stripe** payment gateway. Product images are stored and managed using **Pinata**, a cloud-based IPFS service, ensuring security, scalability, and easy retrieval.

4.2 Hardware and Software Requirements

The Requirement Analysis phase is a critical part of software development, where the technical needs for hardware, software, and tools are identified and defined.

4.2.1 Hardware Requirements

- Processor : Intel(R) core(TM) i5-5005 U CPU @ 2.00 GHZ 2.00GHZ
- RAM : 4GB
- Hard disk : 1TB
- Monitor : VGA/SVGA

4.2.2 Software Requirements

- Operating System : Windows 10
- IDE : Visual Studio Code
- Browser : Chrome, Edge
- Version Control : Git and GitHub
- Package Manager : NPM
- Database : MySql

4.3 System Implementation, Development, and Testing

The implementation phase involved translating design specifications into functional code, using suitable programming languages and frameworks. Comprehensive testing was conducted throughout the development lifecycle to ensure that the system met functional requirements and performed reliably under various condition

4.4 Deployment and Final Presentation

The completed system was deployed to a cloud server for production use. After deployment, we presented the system to the stakeholders, demonstrating its features and collecting feedback for future improvements. Additionally, a comprehensive project report was prepared, documenting the system's design, development process, and key functionalities, providing stakeholders with a detailed overview of the project.

5 Payment Gateway Activity Flow

This section illustrates the activity flow of the Stripe payment gateway in the ShopVerse application. The process begins when a customer clicks the **"Buy Now"** button and proceeds through the web server, Stripe, and database interactions.

5.1 Activity Flow Description

1. **Customer Action:** The customer clicks the *"Buy Now"* button on the product page.
2. **Site / Frontend Processing:** The site collects product and customer information and sends a payment request to the web server.
3. **Web Server Processing:** The server communicates with Stripe API, passing transaction details (amount, currency, and customer info).
4. **Stripe Payment Gateway:** Stripe validates and processes the payment, returning a success or failure status to the web server.
5. **Database Update:** The server updates the order status in the database based on the payment result.
6. **Feedback to Customer:** The customer is notified of the payment outcome via a confirmation page or error message.

5.2 Activity Diagram

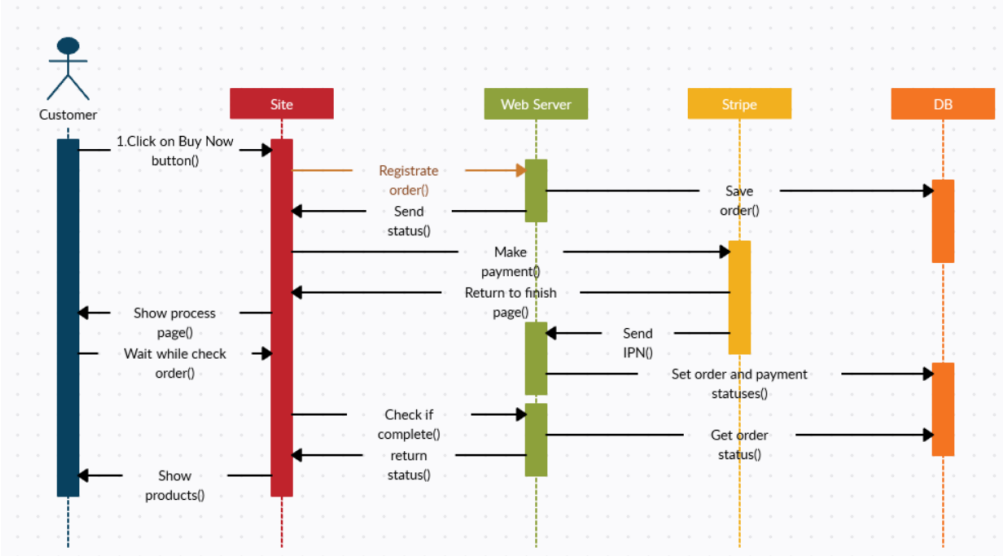


Figure 5: Activity Diagram for Stripe Payment Gateway Flow in ShopVerse

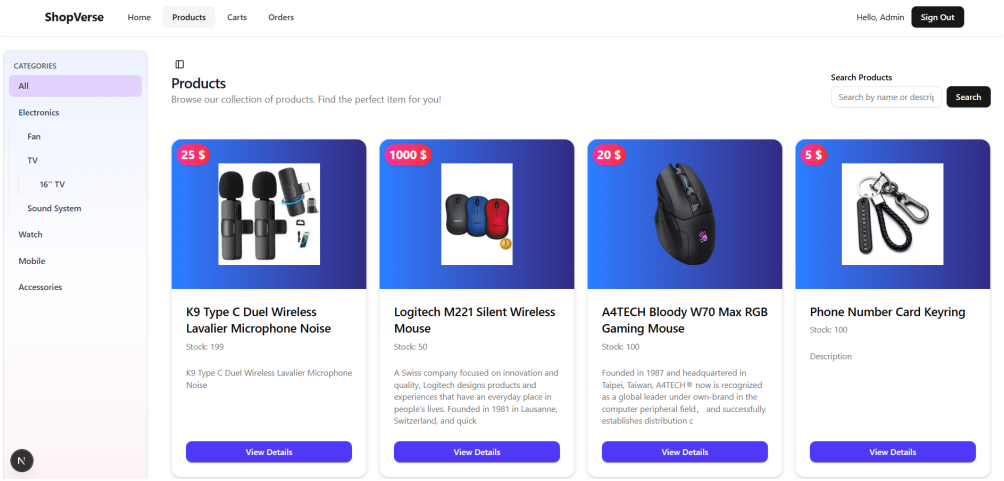
6 Software Design

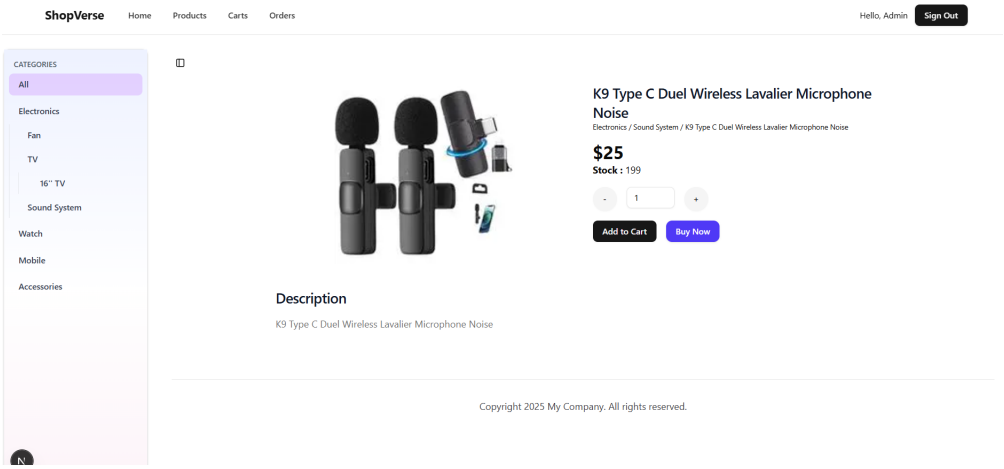
6.1 Definition

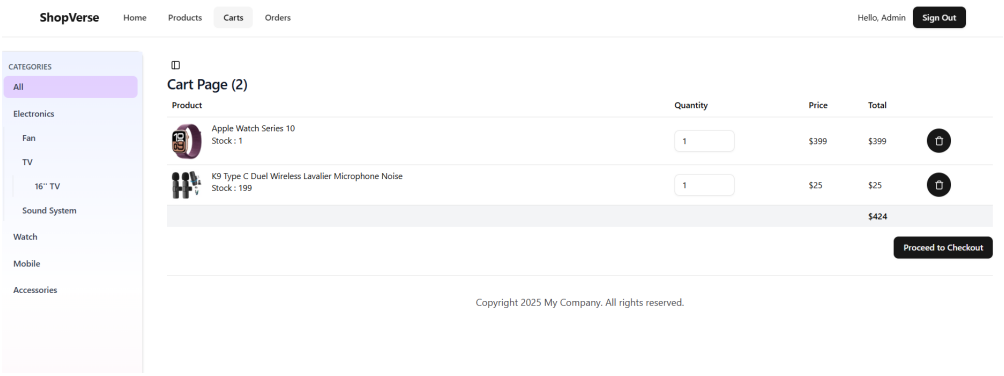
Software design is an iterative process that translates requirements into a blueprint for building software. Initially, the design provides a high-level view, aligning with the system's objectives and detailed data, functional, and behavioral requirements.

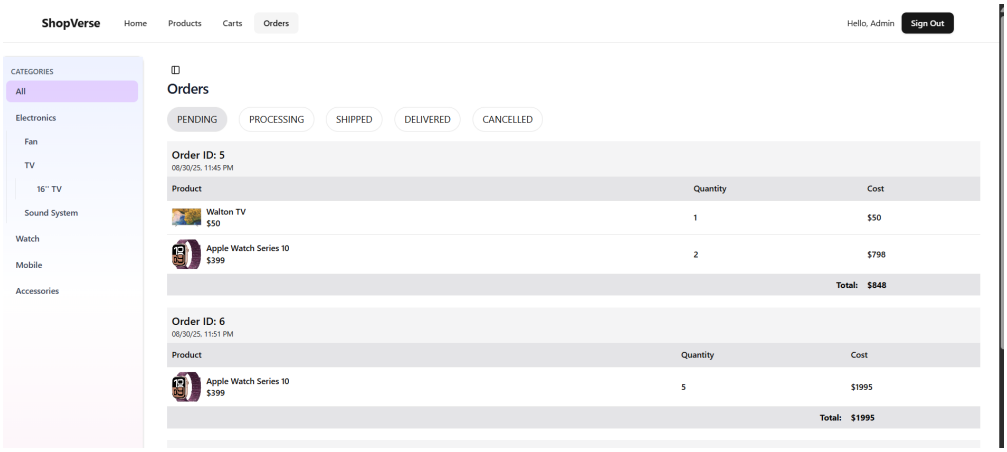
6.2 A Closer Look at Our Web App

Customer

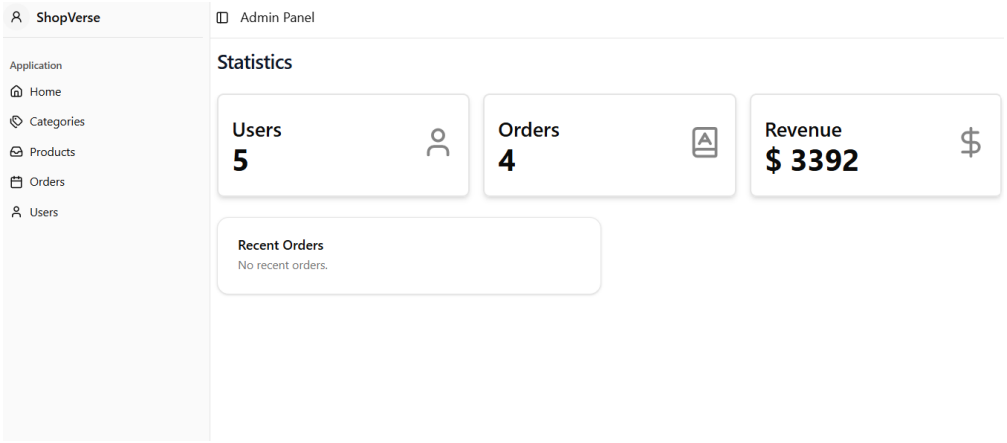




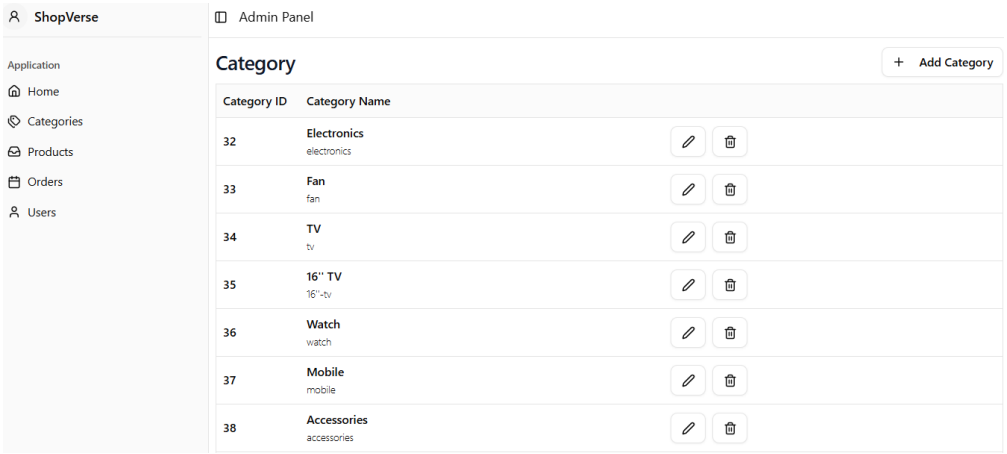


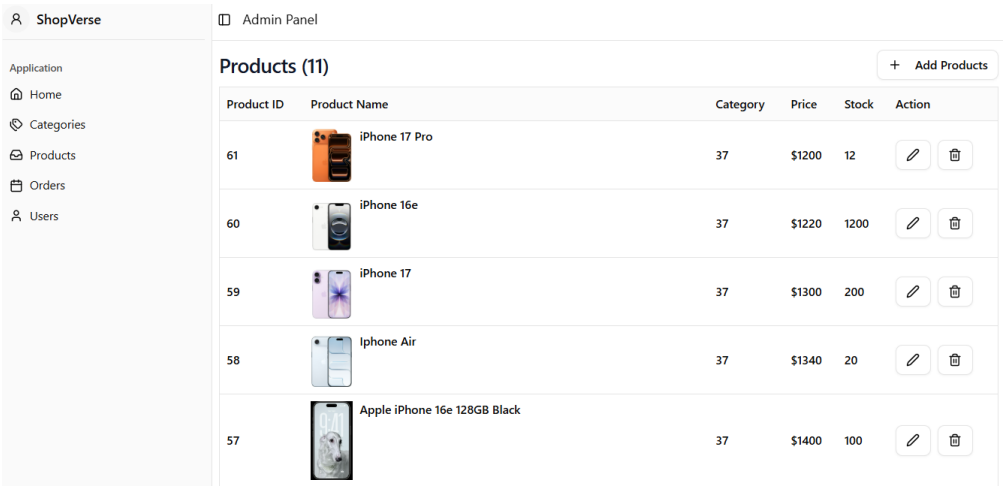


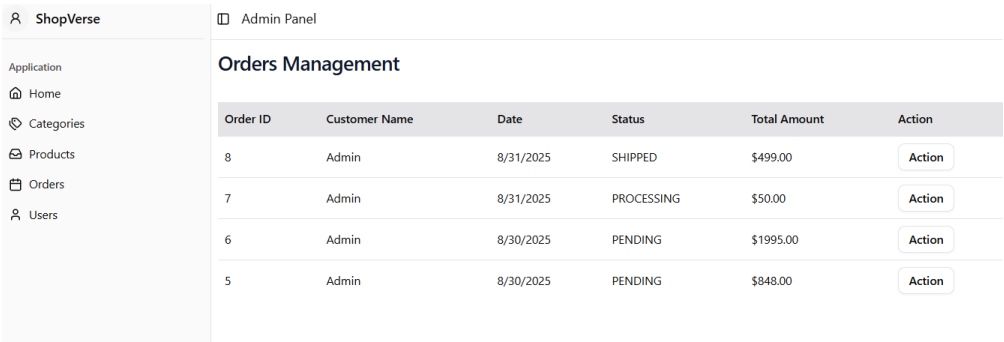
Admin Panel



Department







6.3 Github Repository

https://github.com/borhan008/complain_nstu

7 Conclusion and Recommendations

7.1 Conclusion

In conclusion, the **ShopVerse: Online Shopping System** web application serves as an efficient platform for organizations to manage their products and for customers to shop online seamlessly. The system provides functionalities such as product browsing, cart management, order placement, and secure payment processing through Stripe, while cloud-based image handling via Pinata ensures reliable media storage.

Although the system meets the core requirements, it has some limitations, including basic search functionality, limited analytics, and the absence of multi-vendor support. Future improvements could include a dedicated mobile application, advanced filtering and search features, enhanced admin control, and expanded

analytics capabilities. Despite these limitations, the project establishes a strong foundation for further enhancements and scalable growth.

7.2 Recommendations

The software is designed to be user-friendly and scalable. For future development and optimization, the following recommendations are suggested:

- **Expand User Roles**: Introduce additional roles such as Vendor, Moderator, or Guest to provide granular access control and better manage system responsibilities.
- **Implement Advanced Search and Filtering**: Enhance product search capabilities by supporting category-based filtering, keyword search, and price range selection.
- **Optimize Scalability**: Apply performance optimizations such as database indexing, caching strategies, and cloud-based resources to efficiently handle growing data and user traffic.
- **Enhance Security Measures**: Strengthen security through features like multi-factor authentication (MFA), secure password policies, and regular security audits to protect user and transaction data.
- **Integrate Analytics and Reporting**: Incorporate dashboards and reports for admins to monitor sales, user behavior, and inventory trends for informed decision-making.

8 References

1. Next.js Documentation. *The React Framework for Production.* Retrieved from `https://nextjs.org/docs`
2. ReactJS. *React - A JavaScript Library for Building User Interfaces.* Retrieved from `https://react.dev/`
3. Stripe. *Stripe Payment Integration Documentation.* Retrieved from `https://stripe.com/docs`
4. Pinata. *Pinata Cloud - IPFS File Hosting and Management.* Retrieved from `https://docs.pinata.cloud/`
5. MySQL Documentation. *MySQL Reference Manual.* Retrieved from `https://dev.mysql.com/doc/`
6. Prisma. *Prisma ORM Documentation.* Retrieved from `https://www.prisma.io/docs/`
7. Tailwind CSS. *Utility-First CSS Framework.* Retrieved from `https://tailwindcss.com/docs`
8. ShadCN. *shadcn/ui Component Library.* Retrieved from `https://ui.shadcn.com/`
9. Mozilla Developer Network. *JavaScript Functions.* Retrieved from `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions`
10. W3C. *Web Application Security.* Retrieved from `https://www.w3.org/Security`
11. Ahmad, M., & Raza, S. (2020). *Building Scalable Web Applications: A Practical Approach.* Journal of Web Development, 15(2), 45-56.
12. Smith, R., Doe, J., & Brown, K. (2019). *User Authentication in Web Applications.* In *Proceedings of the International Conference on Web Security* (pp. 35-41). Springer.