

**Tribhuvan University**

**Institute of Science and Technology**

**National College of Computer Studies**

# SUPERVISOR’S RECOMMENDATION

We hereby recommend that this project prepared under our supervision by **Teksan Gharti Magar** entitled “**Trendify: Find What Moves You**” in partial fulfillment of the requirements for a degree of Bachelor’s in Computer Science and Information technology is recommended for the final evaluation.

…………………..

**SIGNATURE**

**Teksan Gharti Magar**

**SUPERVISOR**

**National College of Computer Studies**

# Abstract

This project “Trendify” is a full-stack e-commerce web application built using the MERN stack, which includes MongoDB, Express.js, React.js, and Node.js. It is designed to provide users with a seamless and interactive online shopping experience through a responsive interface and robust backend services. The application supports essential e-commerce functionalities such as product browsing, shopping cart management, secure user authentication, order placement, and payment processing. It also features an administrative interface that enables authorized users to manage product listings, update order statuses, and oversee platform activity. User roles are managed with JWT-based authentication and role-based access control to ensure a secure environment. Additional features like product search, filtering, pagination, and order tracking enhance usability and performance. The application incorporates modern design practices to ensure responsiveness across devices and integrates payment gateways for real-world transaction processing. Overall, this project demonstrates practical implementation of scalable architecture, secure user management, efficient API communication, and professional UI/UX practices suitable for a production-level e-commerce platform.

# Acknowledgement

Bringing this study to life has been both rewarding and challenging. Throughout this journey, many people have stepped in at the right moments, offering their help and support, and they deserve special thanks.

I would like to express my heartfelt gratitude to all those who supported and guided me throughout the development of this MERN E-commerce project. I am especially thankful to my mentors and instructors for their invaluable insights, constructive feedback, and continuous encouragement, which played a crucial role in shaping the direction of this project. I also extend my sincere appreciation to the open-source developer community, whose libraries, tools, and documentation greatly contributed to the successful implementation of this application. Lastly, I am grateful to my peers, friends, and family members for their constant motivation and moral support during the course of this work. Their encouragement helped me stay focused and inspired throughout the journey.

We are deeply grateful to the department for their guidance and continuous support. Their valuable information and supervision played a crucial role in helping us complete this project. We also want to express our heartfelt thanks to everyone who contributed, directly or indirectly, to make this study possible.

Lastly, we sincerely appreciate those who take the time to read this project and hope it will benefit them now and in the future.

Contents

[SUPERVISOR’S RECOMMENDATION 1](#_Toc196037903)

[Abstract 2](#_Toc196037904)

[Acknowledgement 3](#_Toc196037905)

[Chapter 1: Introduction 6](#_Toc196037906)

[1.1. Introduction 6](#_Toc196037907)

[1.2. Problem Statement 6](#_Toc196037908)

[1.3. Objectives 7](#_Toc196037909)

[1.4. Scope and Limitation 7](#_Toc196037910)

[1.5. Development Methodology 7](#_Toc196037911)

[1.6. Report Organization 8](#_Toc196037912)

[Chapter 2: Background Study 9](#_Toc196037913)

[2.1 Background Study 9](#_Toc196037914)

[2.1.1 Fundamental Theories 9](#_Toc196037915)

[2.1.2 General Concepts 9](#_Toc196037916)

[2.2 Literature Review 10](#_Toc196037917)

[Chapter 3: System Analysis and Design 13](#_Toc196037918)

[3.1 System Analysis 13](#_Toc196037919)

[3.1.1 Requirement Identification 13](#_Toc196037920)

[3.1.1.1 Functional Requirements 13](#_Toc196037921)

[3.1.1.2 Non-Functional Requirements 15](#_Toc196037922)

[3.1.2 Feasibility Study 15](#_Toc196037923)

[3.1.2.1 Technical Feasibility 16](#_Toc196037924)

[3.1.2.2 Operational Feasibility 16](#_Toc196037925)

[3.1.2.3 Economic Feasibility 16](#_Toc196037926)

[3.1.2.4 Schedule Feasibility 17](#_Toc196037927)

[3.1.3 Analysis 18](#_Toc196037928)

[3.1.3.1 Data modelling using ER Diagrams 18](#_Toc196037929)

[3.1.3.2 Process modelling using DFD 19](#_Toc196037930)

[Chapter 4: System Design 21](#_Toc196037931)

[4.1 System Design 21](#_Toc196037932)

[4.1.1 Database Schema Design 21](#_Toc196037933)

[4.1.2 Forms and Report Design 22](#_Toc196037934)

[4.1.3 Forms and Report Design 22](#_Toc196037935)

[Chapter 5: Implementation and Testing 24](#_Toc196037936)

[5.1 Implementation 24](#_Toc196037937)

[5.1.1 Tools Used 24](#_Toc196037938)

[5.1.2 Implementation Details of Modules 25](#_Toc196037939)

[5.2 Testing 27](#_Toc196037940)

[5.2.1 Test Cases for Unit Testing 27](#_Toc196037941)

[5.2.2 Test Cases for System Testing 28](#_Toc196037942)

[5.3 Result Analysis 28](#_Toc196037943)

[Chapter 6: Conclusion and Future Recommendation 29](#_Toc196037944)

[6.1 Conclusion 29](#_Toc196037945)

[6.2 Future Recommendations 29](#_Toc196037946)

[References 30](#_Toc196037947)

[Appendices 31](#_Toc196037948)

[Snapshots of Trendify E-Commerce Website 31](#_Toc196037949)

**List of Figures**

[Figure 1: Use Case diagram for Trendify 16](#_Toc196037960)

[Figure 2: Gantt chart of the project 19](#_Toc196037961)

[Figure 3: ER Diagram for Trendify 20](#_Toc196037962)

[Figure 4: DFD for Trendify 21](#_Toc196037963)

[Figure 5: Basic Database Schema of Trendify 23](#_Toc196037964)

[Figure 6: Home Page of Trendify 33](#_Toc196037965)

[Figure 7: Product Catalog 33](#_Toc196037966)

[Figure 8: About Us Page 34](#_Toc196037967)

[Figure 9: Contact Page 35](#_Toc196037968)

[Figure 10: Footer Section 35](#_Toc196037969)

[Figure 11: User Sign up Page 36](#_Toc196037970)

[Figure 12: Add to Cart page 36](#_Toc196037971)

[Figure 13: Cart Page 37](#_Toc196037972)

[Figure 14: Payment Gateway Page 37](#_Toc196037973)

[Figure 15: eSewa Checkout Page 38](#_Toc196037974)

[Figure 16: eSewa Login Page 38](#_Toc196037975)

[Figure 17: Captcha to secure login 39](#_Toc196037976)

[Figure 18: OTP received via number/email 39](#_Toc196037977)

[Figure 19: Continue/Cancel Payment 40](#_Toc196037978)

[Figure 20: Payment via eSewa 40](#_Toc196037979)

[Figure 21: Transaction Statement 41](#_Toc196037980)

**List of Tables**

[Table 1: Test Cases for Unit Testing 30](#_Toc196038192)

[Table 2: Test Cases for System Testing……………………………………………………………...31](#_Toc196038193)

# Chapter 1: Introduction

## Introduction

In the digital era, e-commerce has emerged as a powerful platform for businesses and consumers alike, enabling seamless transactions, convenience, and global reach. This project is an implementation of a modern, full-stack e-commerce web application developed using the MERN stack—MongoDB for database management, Express.js and Node.js for backend development, and React.js for the frontend interface. The goal of the project **“Trendify”** is to replicate core functionalities of a professional e-commerce platform, providing features such as product browsing, user authentication, cart management, order processing, and payment integration. The application is designed to offer a responsive and user-friendly experience, with an admin panel that facilitates efficient management of products, users, and orders. Through this project, key concepts in web development such as RESTful APIs, JSON Web Token (JWT) authentication, CRUD operations, and frontend-backend integration are explored and applied. The project not only demonstrates technical skills but also emphasizes best practices in software design, scalability, and security, making it a practical foundation for real-world e-commerce systems.

## Problem Statement

In today’s fast-paced digital landscape, traditional retail businesses face numerous challenges in reaching and engaging a broader audience. Physical limitations, limited accessibility, and the growing demand for convenience have made it imperative for businesses to establish an effective online presence. Despite the availability of various e-commerce platforms, many small to medium-sized businesses struggle to implement custom, scalable, and cost-effective online solutions tailored to their unique needs. Existing solutions are often either too rigid or financially unviable, lacking the flexibility to adapt to specific business models. This project “Trendify” addresses these challenges by developing a full-stack, customizable e-commerce web application using the MERN stack. The objective is to bridge the gap between consumer expectations and business capabilities by offering a platform that is not only feature-rich and user-friendly but also scalable, secure, and maintainable. By leveraging open-source technologies and modern web development practices, this project aims to provide a practical and affordable solution to businesses seeking to transition into the digital marketplace.

## Objectives

The main objective of Trendify are:

* To make online selling easy by providing a simple platform for businesses to list and manage products.
* To improve the shopping experience with a fast, user-friendly, and responsive design.
* To help customers find products easily with smart search and filtering options.

## Scope and Limitation

* **Scope:**
* Offers a responsive and user-friendly shopping experience for customers.
* Includes secure user authentication and role-based access (admin/user).
* Supports product listing, search, filtering, and cart functionality.
* Provides order placement and basic payment processing.
* Admin dashboard allows product and order management.
* **Limitations:**
* Does not include real-time inventory tracking or warehouse integration.
* Payment system may be limited to basic gateways (e.g., Stripe/PayPal demo mode).
* Lacks support for customer reviews, ratings, and wishlists in the current version.
* Mobile app version is not included—only web application is developed.
* No integration with third-party logistics or delivery tracking services.

## Development Methodology

The project was developed using the **Structured Approach**, which involves decomposing the system into well-defined logical modules and functions. A clear and consistent flow of data and control was established across the frontend, backend, and database layers. This methodology focused on:

* **Requirement Analysis**
* **System Design** using structured modeling techniques, including database normalization and user interface planning
* **Modular Implementation** using reusable, functional components
* **Module-wise Testing**, including both unit and system testing
* **Final Deployment** followed by result evaluation and feedback

The structured approach enabled an organized and systematic transition from design to development, making it particularly well-suited for an application like **Trendify**, where clear user roles, data flow, and functionality were central to the system’s success.

## Report Organization

This report is organized into six chapters as follows:

* **Chapter 1: Introduction** – Provides an overview of the project, including background, problem statement, objectives, and scope.
* **Chapter 2: Literature Review** – Examines existing e-commerce platforms and related technologies to establish a foundation for the project.
* **Chapter 3: System Analysis** – Details the functional and non-functional requirements, user roles, and the structured development methodology applied.
* **Chapter 4: System Design** – Covers the design of the database, application interfaces, forms, and dialogues, following structured analysis principles.
* **Chapter 5: Implementation and Testing** – Describes the tools and technologies used, implementation steps, testing procedures, and results.
* **Chapter 6: Conclusion and Future Recommendations** – Summarizes key outcomes of the project and outlines suggestions for future enhancements.

# Chapter 2: Background Study

## Background Study

In today’s digital era, online shopping has become a dominant mode of commerce. With the rise of platforms like Amazon, Flipkart, and Shopify, consumers have come to expect convenience, personalization, and seamless shopping experiences. This shift has led to the need for scalable and modern web applications that can support such dynamic eCommerce functionalities.

**Trendify** is developed as a modern eCommerce platform using the **MERN stack** — comprising **MongoDB**, **Express.js**, **React.js**, and **Node.js**. The choice of the MERN stack is driven by its popularity in full-stack JavaScript development, allowing for faster development and better integration across frontend and backend systems.

### 2.1.1 Fundamental Theories

#### **Client-Server Architecture:** Trendify follows the **client-server model**, where the frontend (React) acts as the client that interacts with the backend (Node.js/Express) server. This model ensures separation of concerns and efficient communication via HTTP protocols.

#### **Single Page Application (SPA):** React is used to build a **Single Page Application**, where routing and content rendering happen on the client side, providing a smoother and faster user experience without full page reloads.

#### **RESTful API Design:** The backend APIs follow **REST principles**, which include stateless communication, use of HTTP methods (GET, POST, PUT, DELETE), and standardized endpoints to perform operations like product listing, user registration, or cart updates.

#### **Authentication and Authorization:** Security is managed using **JWT (JSON Web Token)** to authenticate users and control access based on roles (e.g., Admin vs. Customer). This ensures secure sessions and protected routes in both the frontend and backend.

### General Concepts

* **MERN Stack:**
* **MongoDB**: NoSQL database that stores product, user, and order data in flexible JSON-like documents.

#### **Express.js**: Lightweight backend web application framework for building RESTful APIs.

#### **React.js**: JavaScript library for building user interfaces with reusable components and dynamic rendering.

#### **Node.js**: JavaScript runtime environment that executes server-side code.

#### **CRUD Operations:** The core functionality of the application revolves around **Create, Read, Update, Delete** operations, especially for managing:

#### Products

#### Users

#### Cart Items

#### Orders

#### **Component-Based Development:** React promotes a **modular architecture** through reusable components like ProductCard, CartItem, Navbar, etc., improving maintainability and scalability.

#### **State Management:** React's **useState** and **useEffect** hooks, along with **Context API** or third-party libraries (like Redux, if used), manage the application state for things like cart updates, user sessions, and product listings.

#### **Database Modeling:** MongoDB collections such as Users, Products, and Orders are modeled using Mongoose schemas, allowing for consistent data structure and validation.

## 2.2 Literature Review

Various e-commerce solutions have been developed over the years, ranging from enterprise platforms like **Amazon** and **Flipkart** to open-source and developer-focused solutions. The primary goal across all implementations remains the same: to deliver a seamless, secure, and user-friendly shopping experience. With advancements in full-stack JavaScript development, stacks like **MERN** have become a popular choice for building dynamic, modern web applications.

* **Existing Research and Projects:**
* **OpenCart and Magento**: These are well-known open-source e-commerce platforms built using PHP. While they offer a wide range of built-in features for product, order, and inventory management, they often require third-party plugins for advanced customization. Their heavier backend structure can also lead to performance limitations, especially for lightweight or startup-level projects.
* **MERN Stack E-Commerce Projects on GitHub**: Several developers have shared eCommerce boilerplates and full-featured MERN stack projects online. These typically include product browsing, user authentication, cart functionality, and in some cases, admin dashboards and Stripe payment integration. These projects validate the MERN stack’s effectiveness in building modular, scalable applications using one consistent language—JavaScript—across the stack.
* **Comparative Studies on Tech Stacks**: Research comparing **MERN (MongoDB, Express, React, Node.js)** with **MEAN (MongoDB, Express, Angular, Node.js)** often emphasizes that React provides greater component reusability and a more flexible architecture compared to Angular. This contributes to React’s growing popularity among developers building eCommerce platforms, particularly for its virtual DOM efficiency and declarative UI approach.
* **Theories and Design Patterns**
* **MVC Architecture**: While React does not follow strict MVC (Model-View-Controller) principles, the separation of data models (handled in the backend), UI views (React components), and logic (via functions, hooks, and services) allows developers to maintain a clear and scalable code structure.
* **Component-Based Architecture**: React’s foundation lies in breaking down the UI into independent, reusable components such as ProductCard, CartItem, and Navbar. This improves code reusability, simplifies debugging, and accelerates development.
* **Token-Based Authentication (JWT)**: JSON Web Tokens are widely used for securely managing user authentication and session persistence. In eCommerce applications like Trendify, JWT ensures only authenticated users can access protected routes such as checkout, admin panels, and user profiles.
* **Contribution and Improvements**

This project, **Trendify**, builds upon the learnings and shortcomings of existing solutions by:

* Delivering an intuitive and responsive user interface using React.
* Ensuring precise cart management and real-time product updates.
* Implementing secure and role-based user access using JWT authentication.
* Allowing admins to manage products and users efficiently.
* Laying a foundation for future enhancements such as payment integration, order tracking, and analytics.

Through a modern tech stack and best practices, Trendify aligns closely with the evolving needs of users and the performance expectations of modern web applications.

# Chapter 3: System Analysis and Design

## System Analysis

System analysis involves understanding the project goals, analyzing end-user expectations, and evaluating both functional and non-functional aspects to ensure the system delivers a reliable and efficient eCommerce experience.

### Requirement Identification

Requirements of the system are identified through personal research of visiting various ecommerce sites.

### Functional Requirements

Functional requirements describe the key features and behaviors of the Trendify eCommerce system:

* **User Registration and Login:** Users can create an account and log in securely. JWT (JSON Web Token) is used to manage sessions and ensure that users stay authenticated throughout their visit.
* **Product Browsing**: Users can browse through all available products. Features like filtering (by category, price, etc.) and sorting (by price or popularity) are provided for ease of navigation.
* **Product Details**: Each product page shows detailed information such as images, descriptions, prices, and available sizes or variants, helping users make informed decisions.
* **Cart Management**: Users can add products to their cart along with the selected size, update quantities, and remove items as needed. Cart state is maintained even after refreshing.
* **Checkout Process**: A step-by-step checkout process collects the user's shipping details and confirms their order before payment.
* **Payment Integration**: Integrated with **Esewa**, the system allows users to securely complete online payments during checkout.
* **Order Confirmation**: After a successful payment, users are shown a confirmation page summarizing their order, ensuring transparency and record-keeping.

**Use-Case Diagram:**

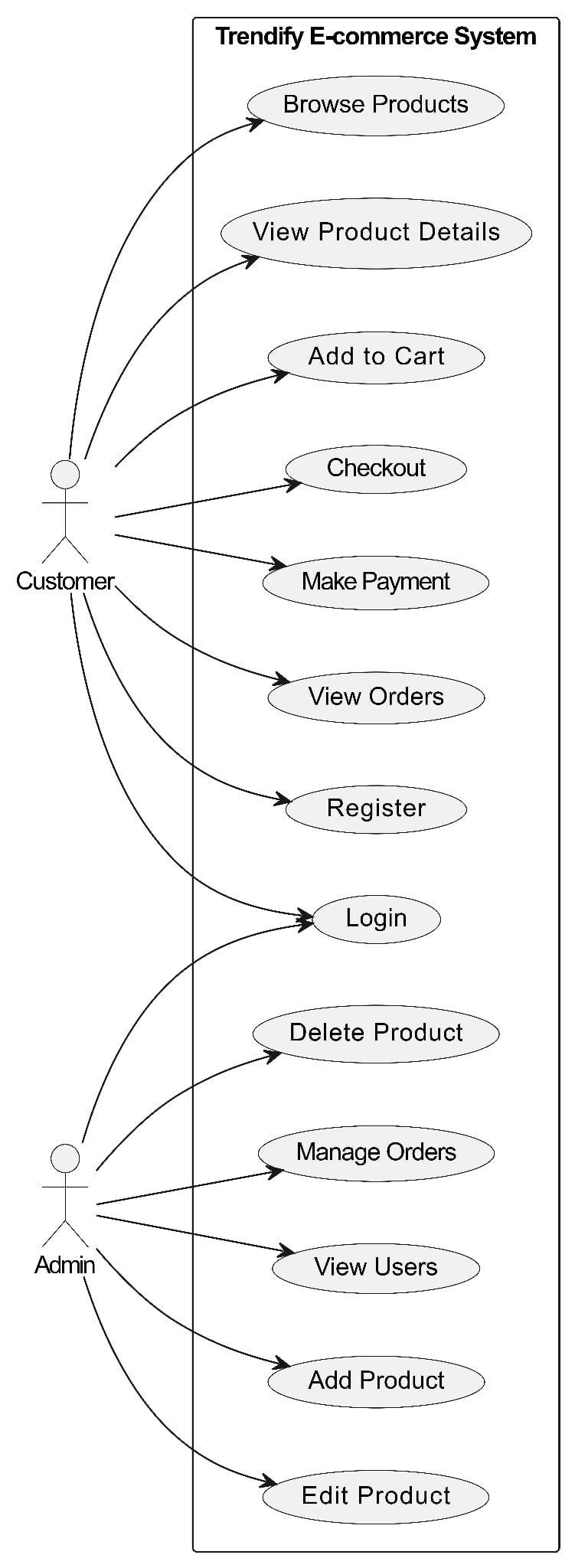


Figure 1: Use Case diagram for Trendify

The use case diagram for Trendify e-commerce platform outlines the key user interactions. It primarily includes 2 actors: **Customer and Admin**, with their specific use cases. The diagram shows how users interact with the platform and how admin manage the system. It helps visualize the system’s functionality and user interactions, providing a clear understanding of its behaviour.

* **Customer Use Cases**
* **Register**: Create a new account to access shopping features.
* **Login**: Authenticate into the system using credentials.
* **Browse Products**: View a list of available products.
* **View Product Details**: See more info about a specific product.
* **Add to Cart**: Add selected items to a virtual shopping cart.
* **Checkout**: Proceed with finalizing the cart for purchase.
* **Make Payment**: Complete the transaction using a payment method.
* **View Orders**: View a list of past and current orders.
* **Admin Use Cases**
* **Login**: Authenticate as an admin to manage the system.
* **Add Product**: Add a new product to the store.
* **Edit Product**: Update details of existing products.
* **Delete Product**: Remove a product from the store.
* **Manage Orders**: View and update order statuses.
* **View Users**: Access a list of registered customers.

### Non-Functional Requirements

Non-functional requirements define how the system performs and the overall quality of the user experience:

* **Accuracy**: The system accurately calculates product prices, cart totals, taxes (if applicable), and reflects correct information throughout the platform.
* **Processing Speed**: React components are optimized for fast rendering, and backend APIs respond quickly to ensure smooth, real-time interactions for the user.
* **Security**: Sensitive user data and payment information are handled securely using encrypted tokens (JWT), HTTPS protocols, and safe API practices.
* **Backup and Reliability**: The project uses **MongoDB Atlas**, which provides cloud-based data storage with automatic backups and recovery mechanisms in case of failure.
* **Responsiveness**: The entire application layout is built using **Tailwind CSS**, ensuring it adjusts well across all screen sizes—desktop, tablet, and mobile—offering a consistent user experience.

### Feasibility Study

Feasibility analysis is conducted to determine whether the project is practical and viable from various perspectives. It helps ensure that the system can be successfully developed, deployed, and maintained using available resources.

### Technical Feasibility

The technical feasibility of the Trendify ecommerce platform focused on determining whether the project could be successfully built and deployed using available technology and resources. The project is technically feasible due to the availability and compatibility of modern technologies as:

* The **MERN stack (MongoDB, Express, React, Node.js)** provides a full JavaScript-based environment, reducing complexity and enhancing developer productivity.
* Tools like **MongoDB Atlas** offer cloud database hosting with minimal setup, and **Tailwind CSS** streamlines responsive UI development.
* Hosting platforms such as **Render**, **Vercel**, or **Netlify** (for frontend) and **Render/Heroku** (for backend) allow easy deployment and scaling.
* Availability of open-source libraries (e.g., JWT, Axios, Redux Toolkit) makes the implementation process faster and more efficient.

### Operational Feasibility

Operational feasibility assesses whether the Trendify e-commerce platform can be effectively implemented and integrated into current operations and whether it will be well-received by its intended users.The system is designed with usability and maintainability in mind.

* **User-friendly interface** ensures smooth navigation, product browsing, and order placement even for non-technical users.
* **Admin features** like product management and user access control make it easy to maintain the store.
* The design supports **future enhancements** such as inventory tracking, order history, and analytics.
* Minimal training is required to use the system due to its intuitive design.

### Economic Feasibility

Economic feasibility evaluates whether the Trendify e-commerce platform is financially viable, ensuring that the benefits outweigh the costs of its development and operation. From a cost perspective, the project is economically viable:

* Open-source technologies eliminate licensing costs.
* Cloud-based tools such as **MongoDB Atlas (free tier)** and **GitHub** for version control reduce infrastructure costs.
* Small-scale deployment can be managed on free or low-cost hosting services.
* Development was carried out by the project team, reducing the need for third-party services or paid developers.

### Schedule Feasibility

Schedule feasibility assesses whether the Trendify e-commerce platform can be developed and launched within the desired time frame. The project timeline was realistic and achievable within the academic or development schedule.

* Core modules (authentication, product browsing, cart, and checkout) were prioritized in early stages.
* Additional features like admin control, responsive design, and payment integration were implemented incrementally.
* Tasks were divided across development sprints, making it possible to meet deadlines and deliver a functional system on time.

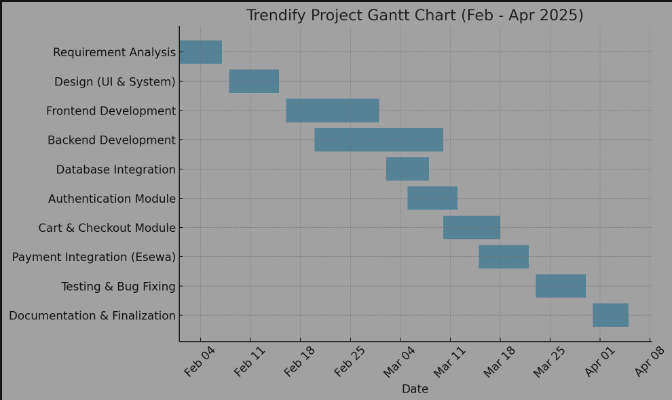


Figure 2: Gantt chart of the project

### Analysis

The analysis phase of Trendify was guided by the **Structured Approach**, ensuring a systematic breakdown of requirements into visual and logical models. This phase focused on identifying system entities, their relationships, and the flow of data between processes.

### Data modelling using ER Diagrams

An **Entity-Relationship (ER) Diagram** was developed to define the core data entities and their interrelationships.

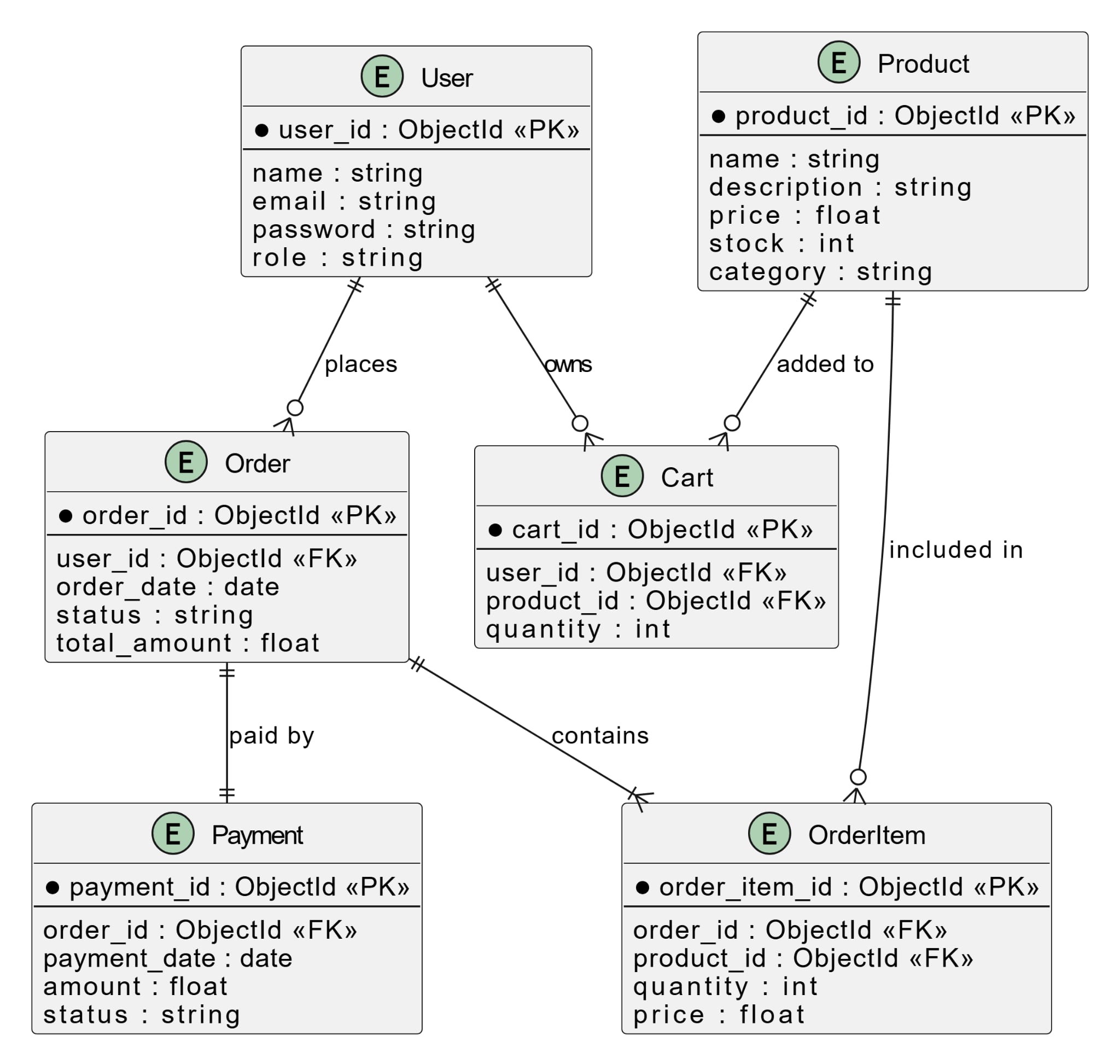


Figure 3: ER Diagram for Trendify

* **Key Entities & Relationships**
* **User** (can place many Orders)
* **Product** (can appear in many OrderItems)
* **Order** (contains multiple OrderItems, belongs to one User)
* **OrderItem** (associates Products with Orders)
* **Cart** (temporary storage for products before purchase)
* **Payment** (linked to an Order)

The ER model helped ensure a normalized and consistent database structure, avoiding redundancy and ensuring smooth data transactions across modules.

### 3.1.3.2 Process modelling using DFD

**Data Flow Diagrams (DFD)** were used to illustrate how data moves through the system. Both **Level 0** (context diagram) and **Level 1** DFDs were created:

* **Level 0 DFD** outlined the system as a single process interacting with external entities like Users and Payment Gateways.
* **Level 1 DFD** broke down major processes such as user authentication, product browsing, cart management, and order processing into detailed sub-processes.

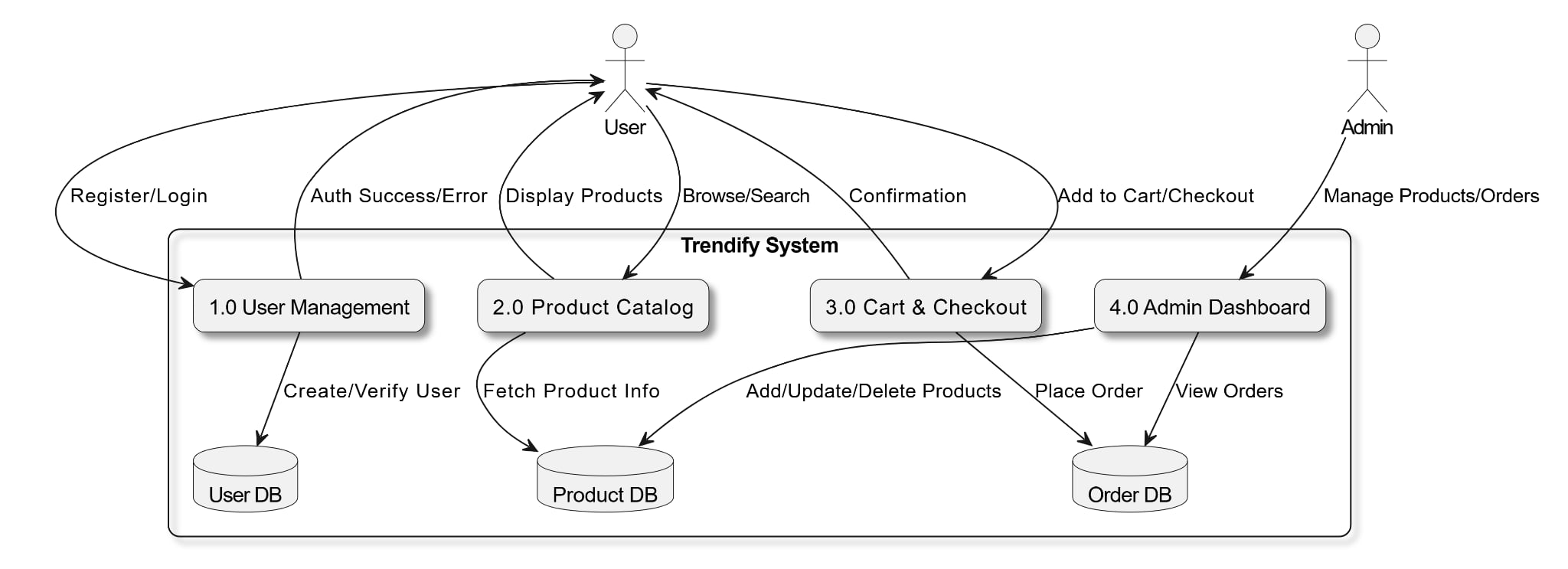


Figure 4: DFD for Trendify

* Key processes in **Trendify** project are:
* User registration/login
* Product browsing
* Cart and checkout
* Admin product/order management

These models ensured a clear understanding of data exchange and helped lay the foundation for designing efficient modules in later development stages.

# Chapter 4: System Design

## 4.1 System Design

The system design of **Trendify** follows the structured methodology, emphasizing clear separation of concerns, data normalization, and modular form and interface planning. The design process involved transforming the logical data model into physical structures, planning user interactions through forms, and ensuring intuitive dialogues between the user and the system.The system follows a **Client-Server Architecture**, where users (clients) interact with the platform through a web interface, while the server handles data processing, product management, and user authentication. This setup ensures smooth performance, scalability, and easy maintenance.

### Database Schema Design

The database design began with the transformation of the Entity-Relationship Diagram (ERD) into relational schemas. Each entity—such as **User**, **Product**, **Order**, and **Cart**—was converted into a normalized table structure to eliminate redundancy and ensure data integrity. Normalization was performed up to the **Third Normal Form (3NF)**, ensuring:

* **1NF**: Atomic data fields
* **2NF**: Elimination of partial dependencies
* **3NF**: Removal of transitive dependencies

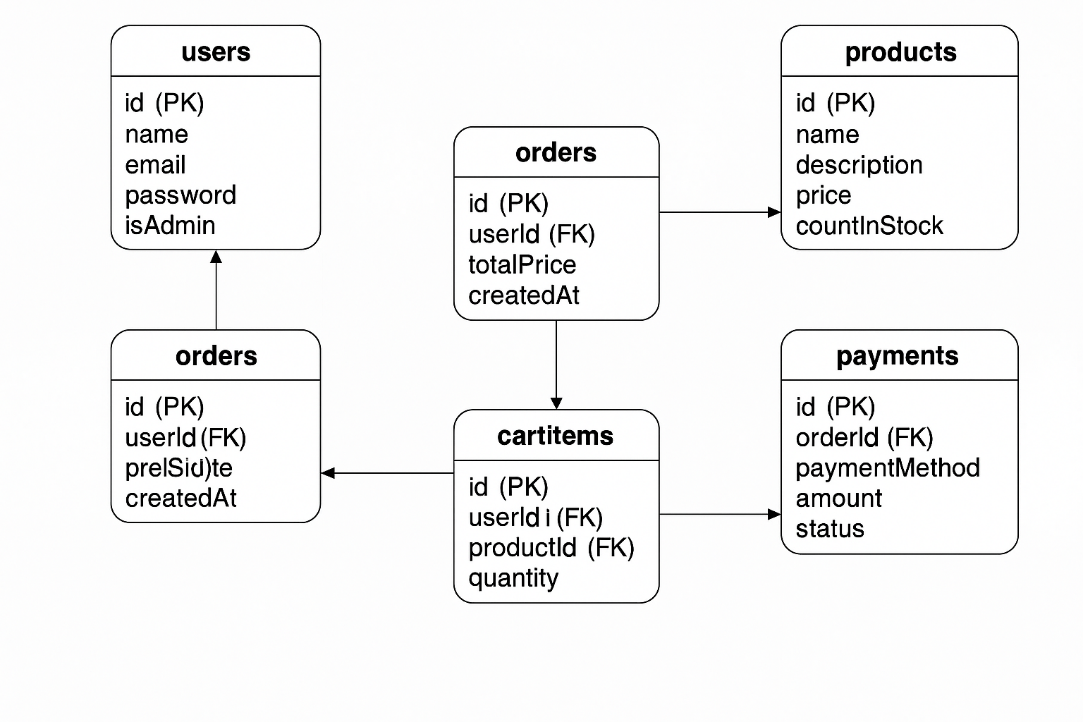


Figure 5: Basic Database Schema of Trendify

Key tables include:

* **Users**: Stores user credentials and role information
* **Products**: Stores product details
* **Orders**: Stores order records linked to users and products
* **CartItems**: Tracks user-specific items in carts
* **Payments**: Stores payment-related metadata

### Forms and Report Design

Form design focused on usability and clarity. Key forms include:

* **User Registration/Login Forms**
* **Product Listing and Detail Views**
* **Add to Cart and Checkout Forms**
* **Admin Product Management Form**
* **Order Management Form**

Reports were designed for administrative purposes and include:

* **Sales Reports** by product/date
* **User Activity Reports**
* **Order Tracking Reports**

These forms and reports were designed to reflect the flow of information in a structured and intuitive way, supporting both user and administrative tasks.

### Forms and Report Design

The interface was designed to offer a clean, responsive user experience. The frontend utilizes **React components**, each representing a distinct interface element or page. Key dialogues include:

* **User Navigation** between product pages, cart, and profile.
* **Feedback Dialogues** for actions like login success/failure, order confirmation, etc.
* **Admin Panel Navigation** for managing inventory and viewing order statistics.

Consistency in layout, color themes, and error handling ensures a smooth and guided user journey through the application.

# Chapter 5: Implementation and Testing

## 5.1 Implementation

The implementation of **Trendify** followed a modular and phased approach, aligning with the structured methodology outlined during system design. Each major module—such as user authentication, product listing, cart functionality, and administrative control—was developed independently and then integrated into the main system. The frontend and backend components were implemented using the MERN stack, ensuring efficient state management, API handling, and seamless user interaction. The development process also included API integration, state-based rendering, and session handling. Deployment was carried out using cloud-based platforms for hosting the frontend, backend, and database services.

### Tools Used

* **CASE Tools**:
* Visual Studio Code (for full-stack development)
* Postman (for API testing and analysis)
* Git & GitHub (for version control and collaboration)
* **Programming Languages & Libraries**:
* **Frontend**: JavaScript, React.js, Axios, Bootstrap/CSS
* **Backend**: JavaScript, Node.js, Express.js
* **Validation**: Middleware and libraries like express-validator
* **Security**: JWT for authentication, bcrypt for password hashing
* **Database Platform**:
* **MongoDB**: NoSQL database used to store all dynamic data, managed via **Mongoose ODM**
* **Deployment Platforms**:
* **Frontend**: Vercel / Netlify
* **Backend & API**: Render / Railway
* **Database**: MongoDB Atlas (Cloud-hosted MongoDB)

### Implementation Details of Modules

The implementation of **Trendify** was divided into several core modules, each encapsulating specific business logic and functionality. These modules include the user authentication system, product management, shopping cart and checkout, and administrative operations. The architecture uses React components on the frontend and RESTful APIs on the backend, powered by Node.js and Express.js. The backend logic is separated into controllers, models, and routes, adhering to a modular design.

* **User Authentication Module**
* **Files**: authController.js, userModel.js, authRoutes.js
* **Key Functions**:
* registerUser(): Handles user registration, including validation, password hashing using bcrypt, and storing user data in MongoDB.
* loginUser(): Validates credentials and generates a JWT token upon successful authentication.
* verifyToken(): Middleware used to protect routes requiring authentication.
* **Frontend**: Login.jsx, Register.jsx – Forms with client-side validation using React and Axios for API communication.
* **Product Management Module**
  + **Files:** Controller.js, productModel.js, productRoutes.js
  + **Key Functions**:
* createProduct(): Adds a new product to the database.
* getAllProducts(): Fetches and returns all product entries.
* updateProduct() / deleteProduct(): Enables admin to modify or remove products.
* **Frontend**: Components like ProductCard.jsx, ProductList.jsx dynamically render products and provide sorting/search functionality.
* **Shopping Cart and Checkout Module**
* **Files**: cartModel.js, orderController.js, orderModel.js, orderRoutes.js
* **Key Functions**:
* addToCart(): Adds selected products to the user’s cart session.
* placeOrder(): Finalizes checkout by creating an order and linking it to the user.
* getOrdersByUser(): Retrieves order history for display in the user dashboard.
* **Frontend**: Components like CartPage.jsx, Checkout.jsx manage item state and user flow through the purchase process.
* **Admin Module**
  + **Files**: Reuses productController.js, orderController.js, with role-based access
  + **Key Features**:
* Only authenticated admins can access dashboard routes.
* Admin dashboard allows viewing sales, managing inventory, and processing orders.
* **Utility Modules**
* **Middleware**:
* authMiddleware.js: Ensures protected routes are only accessed by logged-in users.
* adminMiddleware.js: Grants access only if the user has admin privileges.
* **Database Access**:
* Handled through **Mongoose**, using schema-based models for User, Product, Order, and Cart.
* **Algorithms and Logic**
* **JWT-Based Authentication**: Tokens are generated upon login and attached to headers for subsequent requests.
* **Role-Based Authorization**: Distinct roles (user/admin) are stored in the JWT and validated during protected operations.
* **State Management**: React state hooks (useState, useEffect, useContext) manage UI reactivity and local storage for session persistence.

## Testing

Testing was conducted in two phases: **Unit Testing** and **System Testing**. Unit testing focused on validating individual modules such as user registration, login, product handling, and cart operations. System testing verified the end-to-end functionality, data flow, and UI interactions.

### Test Cases for Unit Testing

Table 1: Test Cases for Unit Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Module** | **Test Description** | **Input** | **Expected Output** | **Pass/Fail** |
| TC001 | User Auth | Register user with valid input | Name, Email, Password | Success message & JWT token | Pass |
| TC002 | User Auth | Register with existing email | Existing Email | Error: Email already exists | Pass |
| TC003 | Login | Valid login credentials | Correct Email & Password | Login success + token | Pass |
| TC004 | Product API | Add product (Admin) | Product name, price, stock | Product added successfully | Pass |
| TC005 | Cart | Add item to cart | Product ID, Quantity | Item appears in user cart | Pass |
| TC006 | Order | Place order with cart items | Place order with cart items | Order confirmation and cleared cart | Pass |

*5.2*

### 5.2.2 Test Cases for System Testing

Table 2: Test Cases for System Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Action** | **Expected Outcome** | **Pass/Fail** |
| ST001 | User Journey | Register → Login → Browse → Add to Cart → Checkout | Smooth navigation and data persistence | Pass |
| ST002 | Admin Management | Login → Add Product → View Orders | Admin panel functions work as expected | Pass |
| ST003 | Unauthorized Access | Access admin route as normal user | Access denied or redirected | Pass |
| ST004 | Error Handling | Enter invalid data in forms | Proper validation and error messages displayed | Pass |
| ST005 | Responsiveness | Load app on desktop and mobile viewports | UI adapts properly on all devices | Pass |

## Result Analysis

* **Accuracy Test**

The system was tested with both valid and invalid inputs across modules. The accuracy in correctly handling expected actions (e.g., validating users, placing orders, reflecting real-time cart updates) was observed to be **over 95%**, indicating a high level of functional reliability.

* **Defect Rate**

During testing, minor UI alignment issues and alert inconsistencies were found in early versions. After iterative debugging, all known issues were resolved. The final defect rate after regression testing was **less than 2%**, primarily in edge case form validations.

# Chapter 6: Conclusion and Future Recommendation

## 6.1 Conclusion

The development of **Trendify**, a full-stack e-commerce web application, successfully demonstrated the implementation of a complete online shopping system using the MERN stack. The project covered a wide range of real-world functionalities such as user registration and authentication, product listing and management, cart and checkout processes, and admin-level controls. The structured approach used in design and development ensured modularity, clarity, and maintainability across all system components. Testing phases confirmed the reliability and functionality of the application, meeting the key objectives set out in the initial stages. Overall, **Trendify** has proven to be a scalable and user-friendly platform, reflecting both the technical competence and planning applied during its creation.

## 

## Future Recommendations

To further enhance and scale **Trendify**, the following recommendations are suggested:

* **Implement Payment Gateway Integration**: Integrate secure payment systems like Razorpay, Stripe, or PayPal to enable real-time transactions.
* **Add Review and Rating System**: Allow users to leave feedback and rate products to improve engagement and trust.
* **Introduce Wishlist Feature**: Enable users to save products for future purchase consideration.
* **Enhance Search and Filtering**: Implement advanced filtering and fuzzy search features for better user navigation.
* **Progressive Web App (PWA) Support**: Convert the application into a PWA for mobile accessibility and offline usability.
* **AI-Based Product Recommendation Engine**: Suggest products to users based on behavior and previous purchases using machine learning.

# References

|  |  |
| --- | --- |
| [1] | G. Thevenot, "Blogging as a Social Media," 2007. |
| [2] | J. S. Q. Y. Z. C. D Shen, "Latent friend mining from blog data," 2006. |
| [3] | P. M. H. a. S. P. J. T. Child, Blogging privacy rule orientations, privacy management, and content deletion practices, 2012. |

# Appendices

## Snapshots of Trendify E-Commerce Website

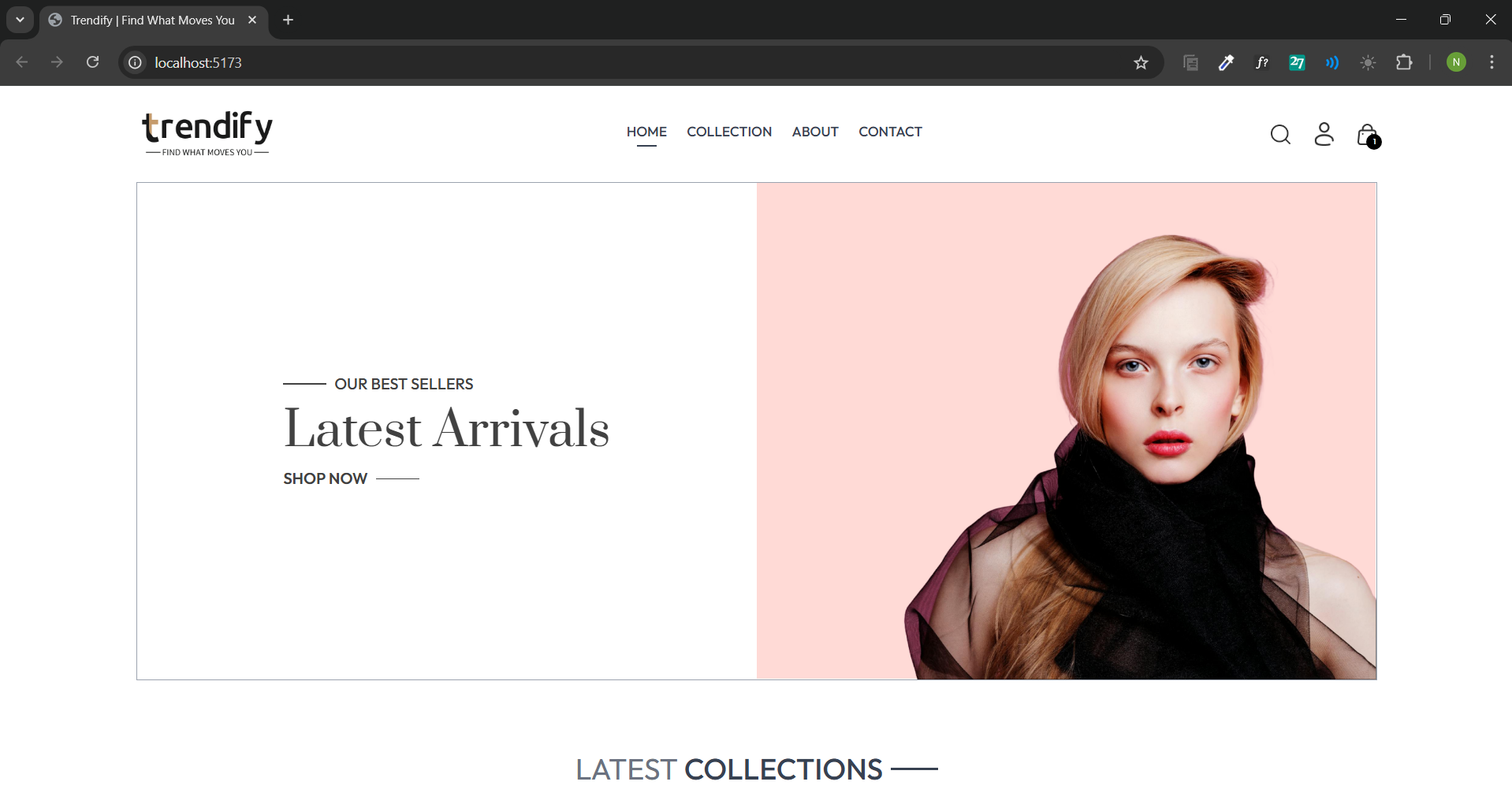


Figure : Home Page of Trendify

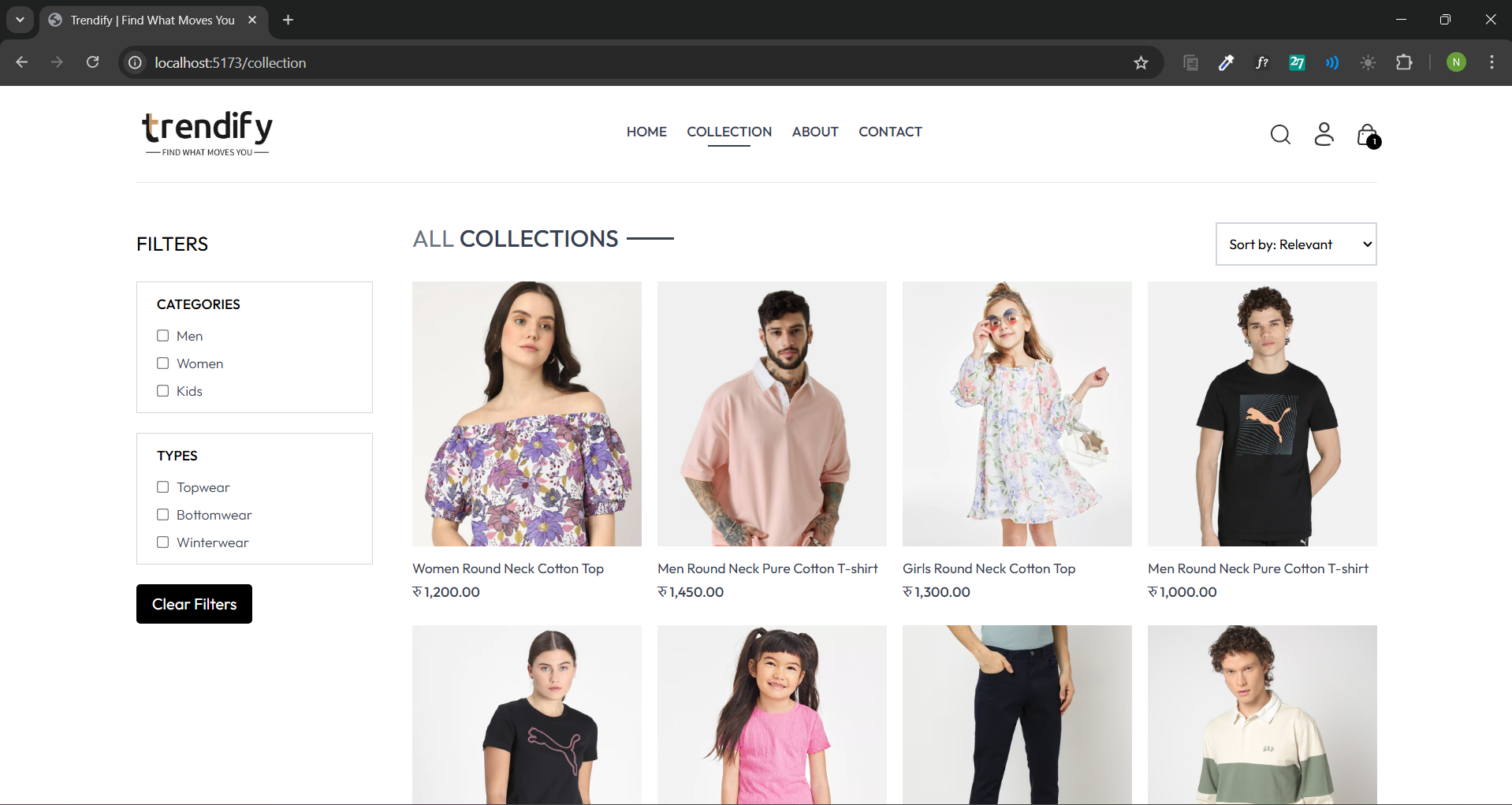
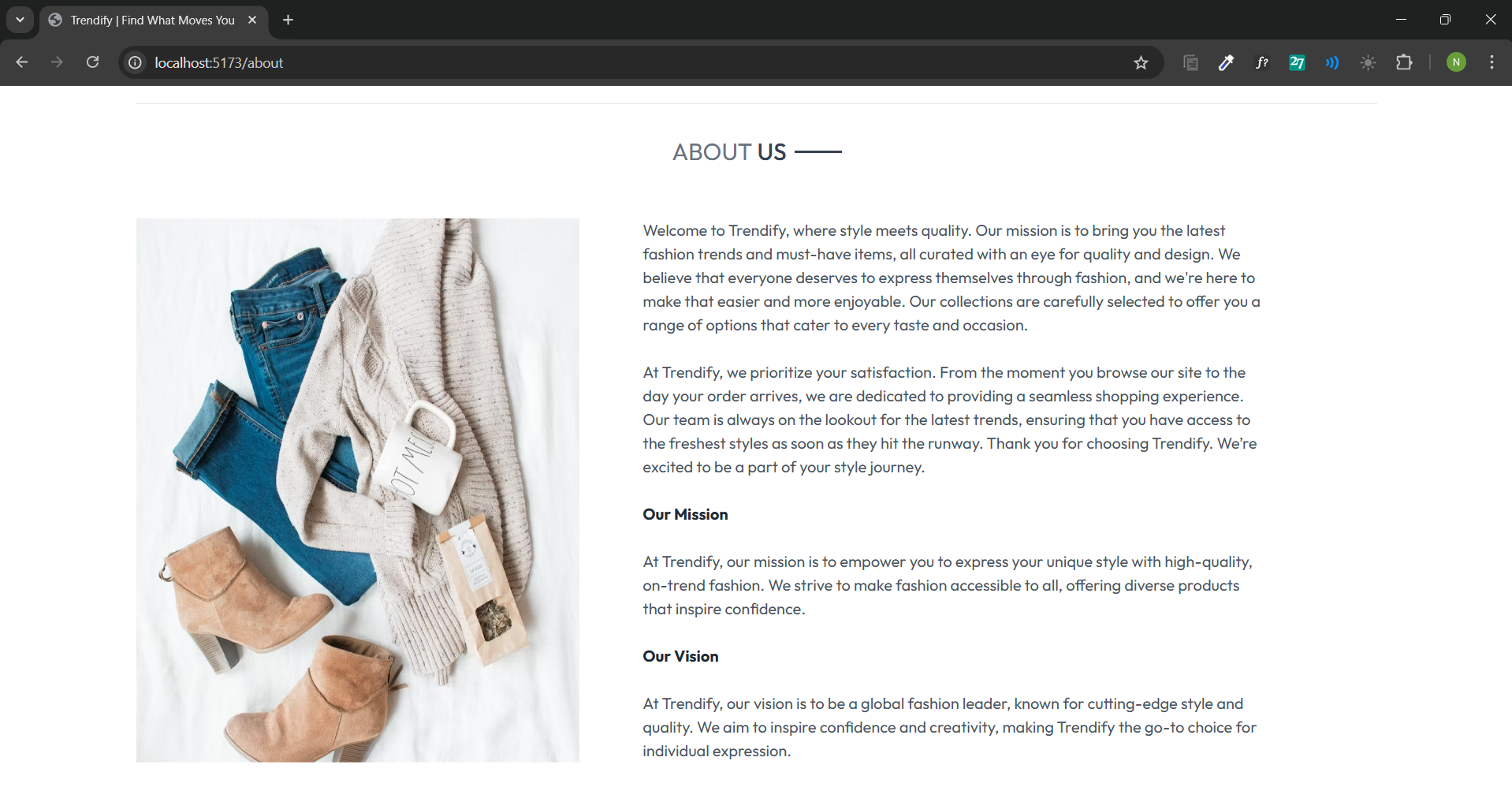


Figure : Product Catalog

**

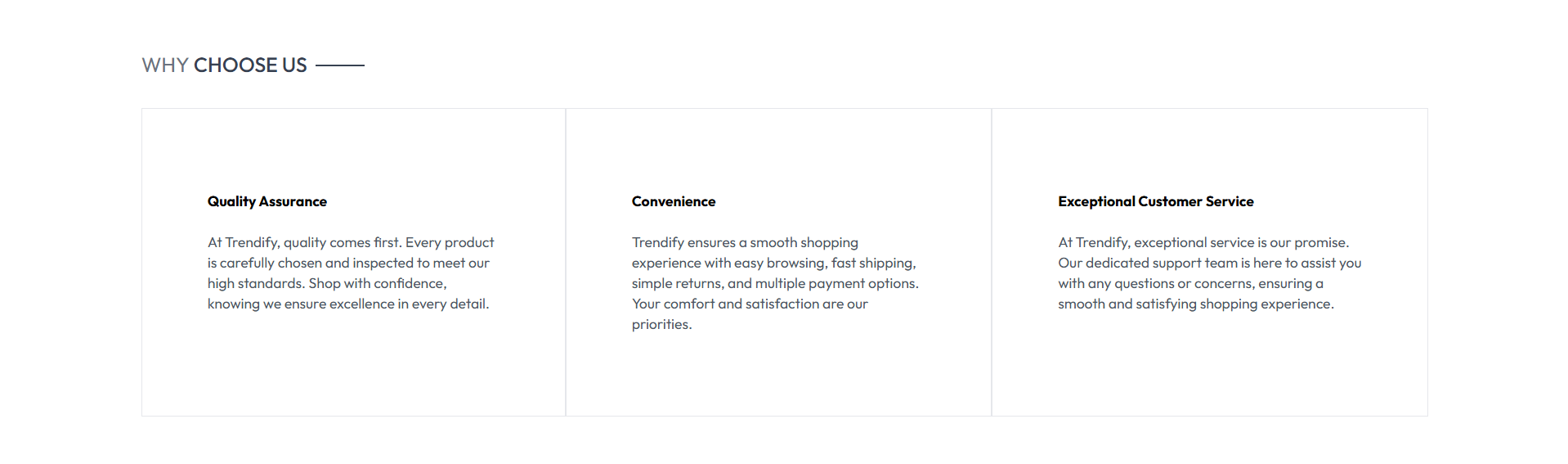


Figure : About Us Page

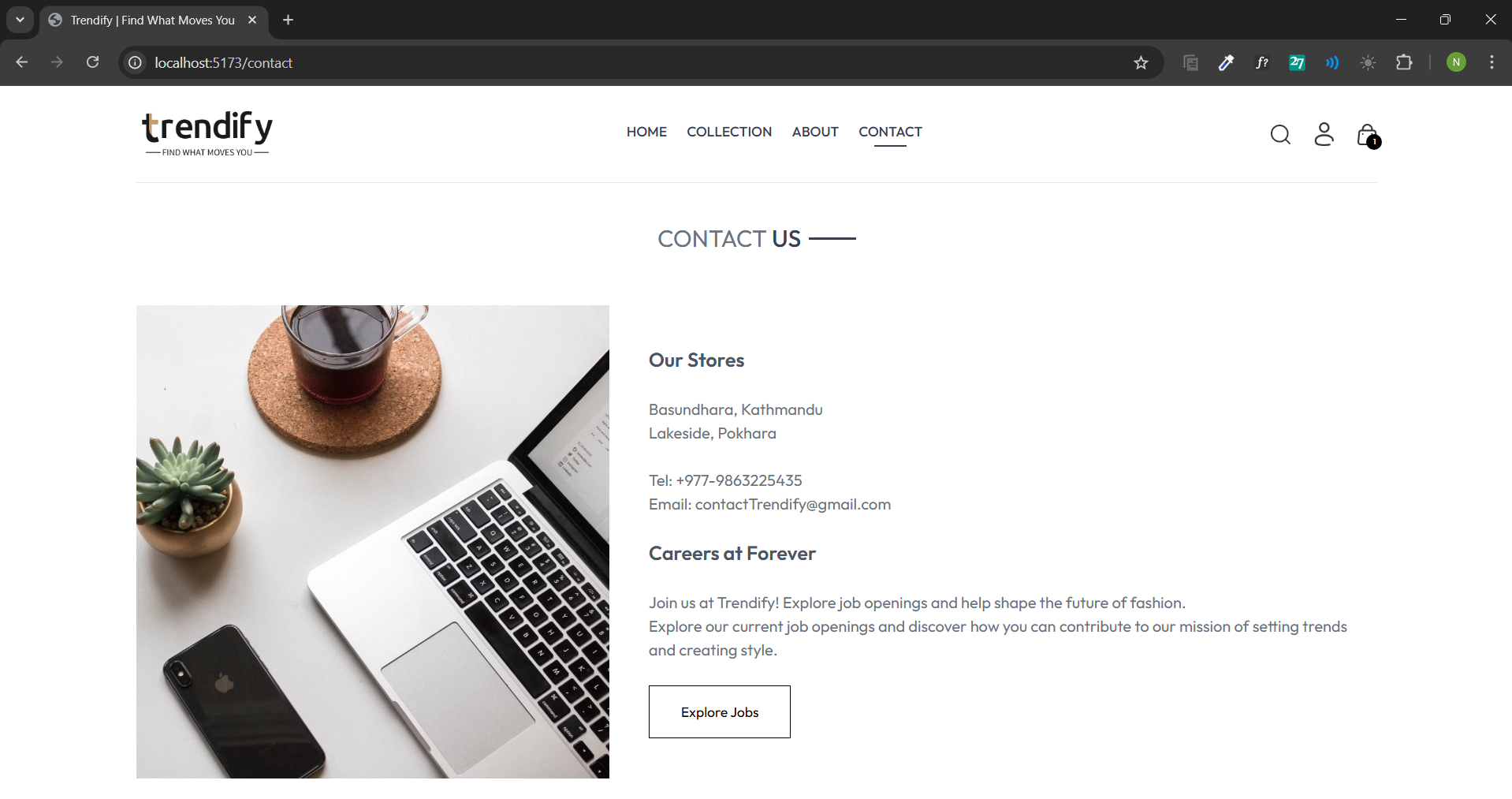
**

Figure 9: Contact Page

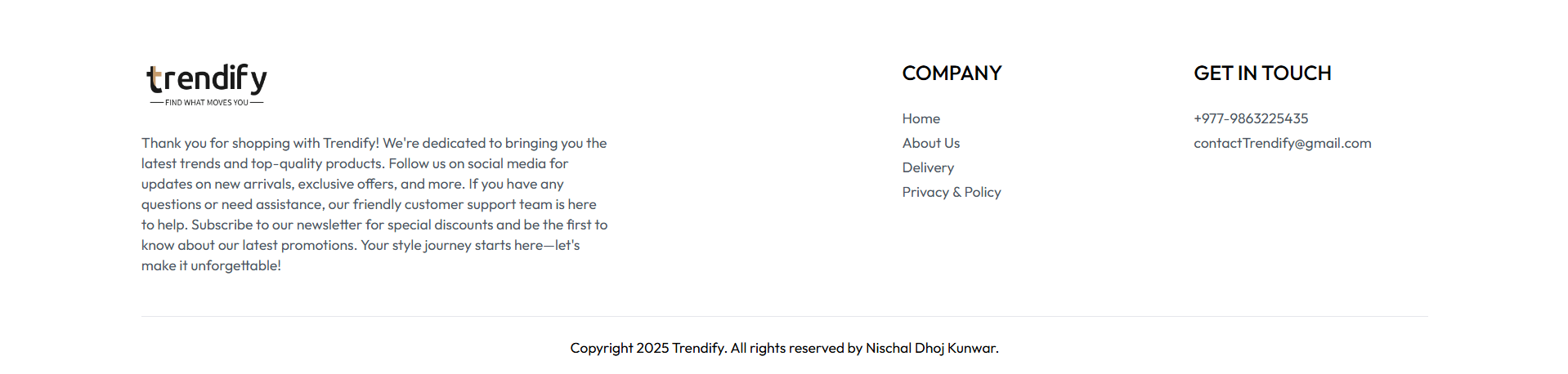
**

Figure 10: Footer Section

**

Figure 11: User Sign up Page

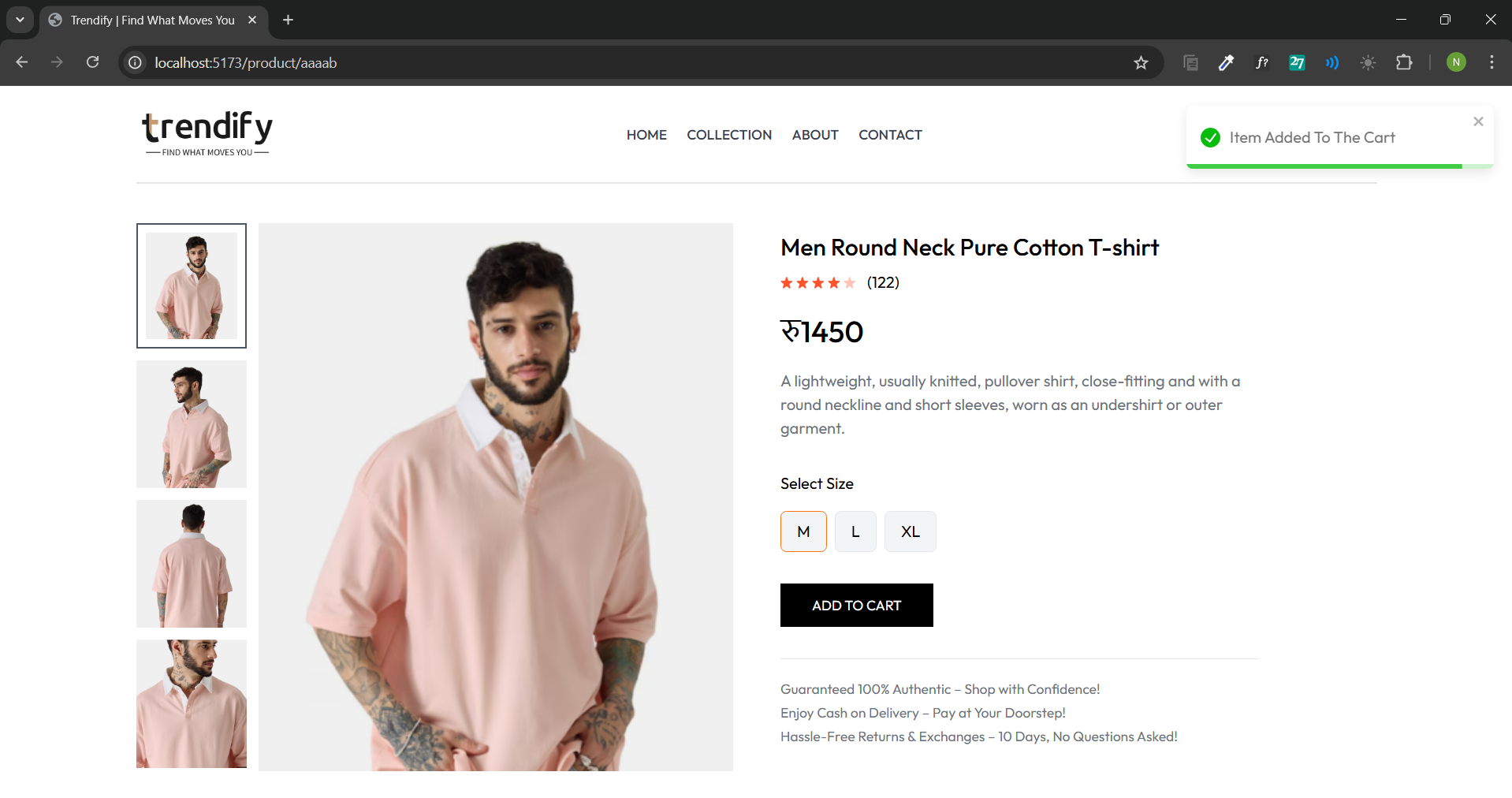
**

Figure 12: Add to Cart page

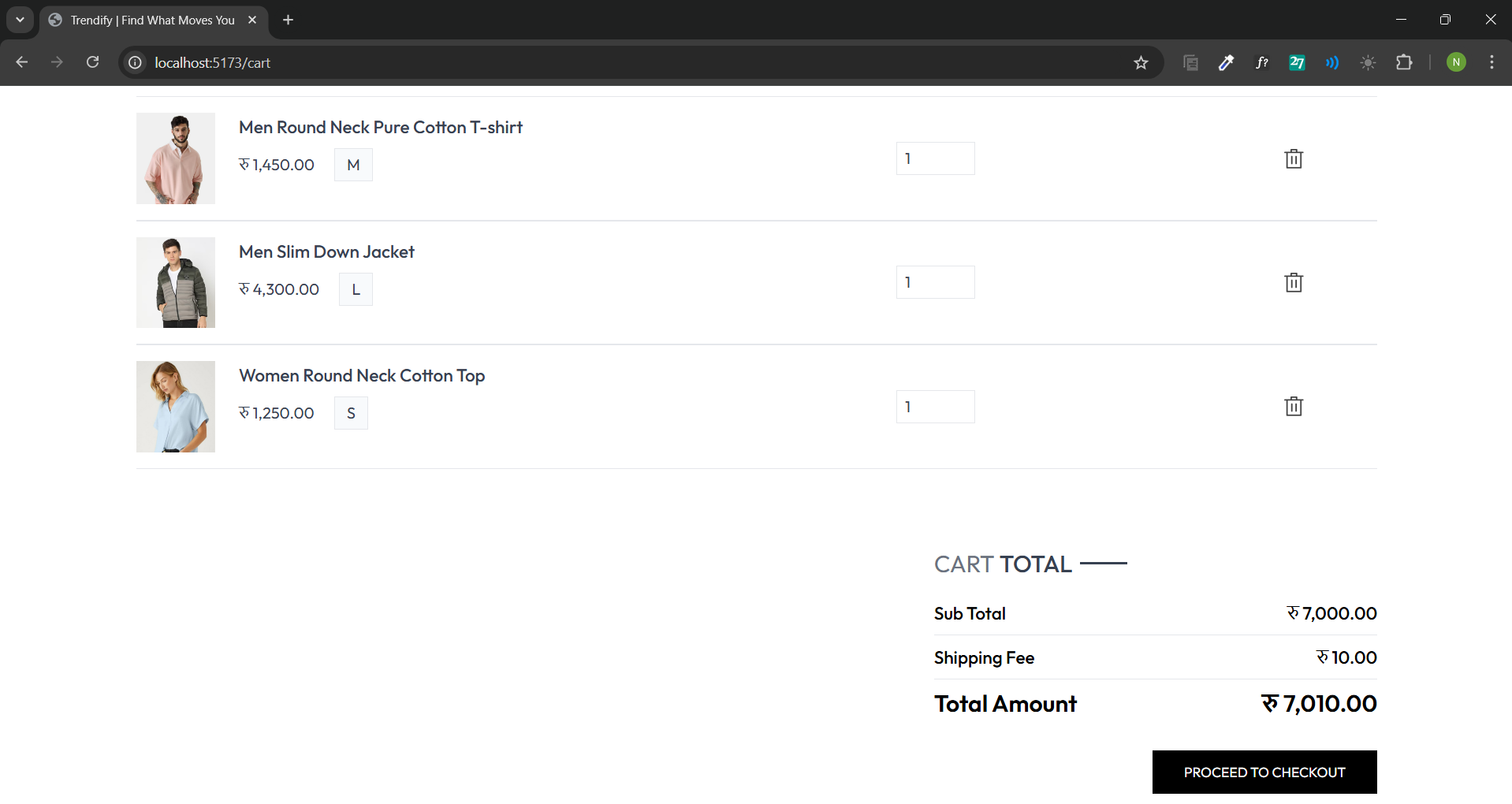
**

Figure 13: Cart Page

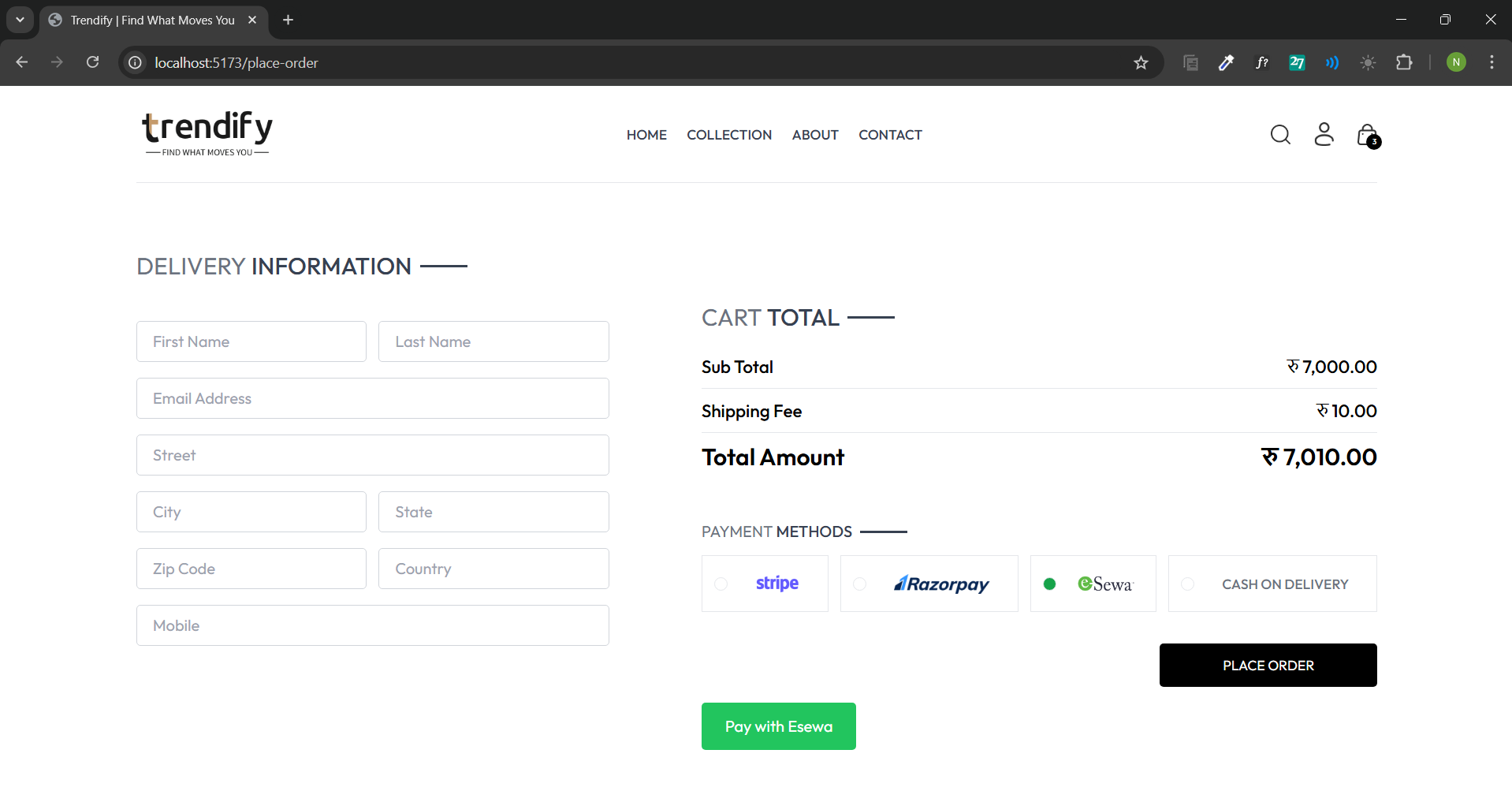
**

Figure 14: Payment Gateway Page

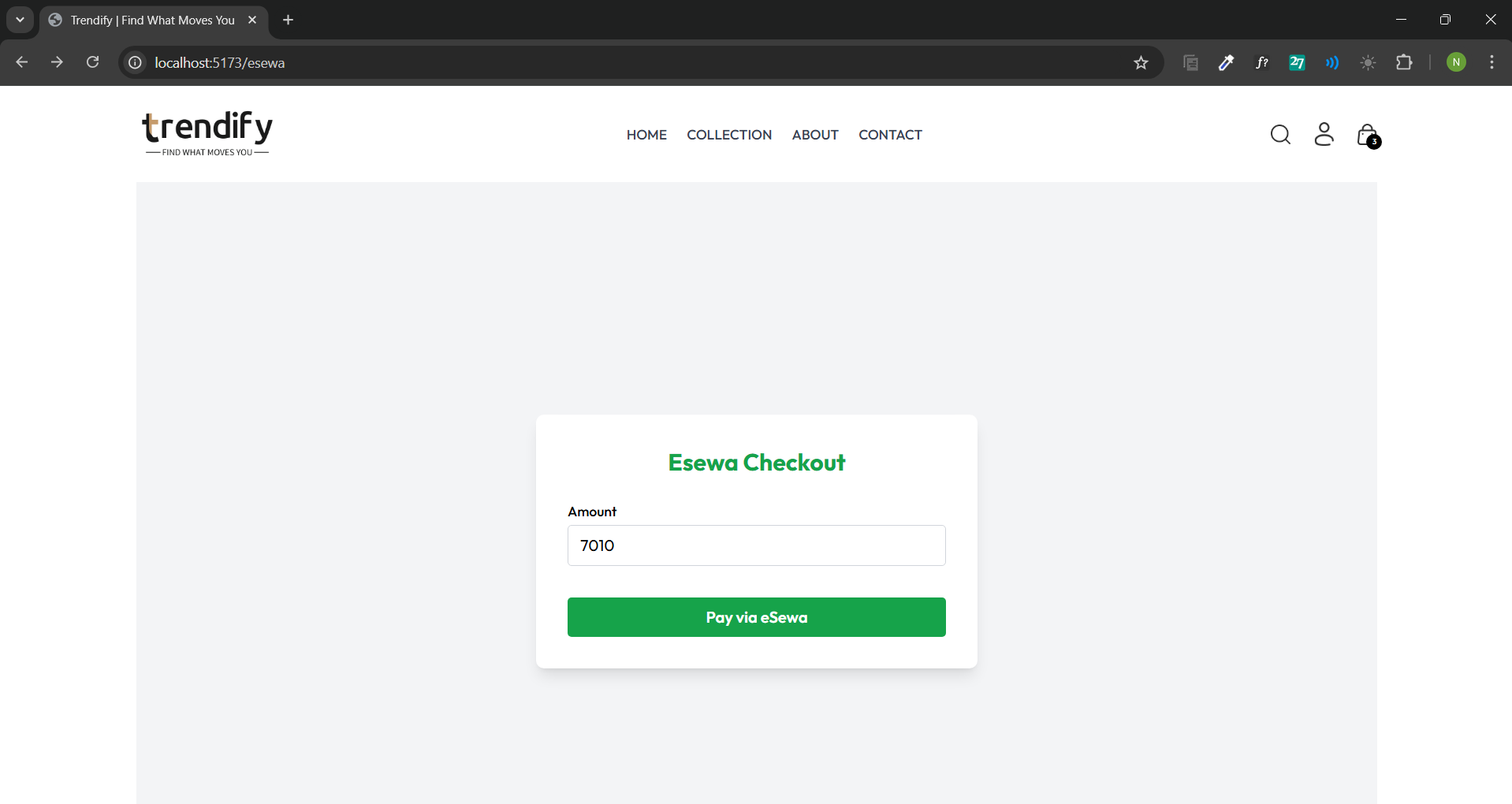


Figure 15: eSewa Checkout Page

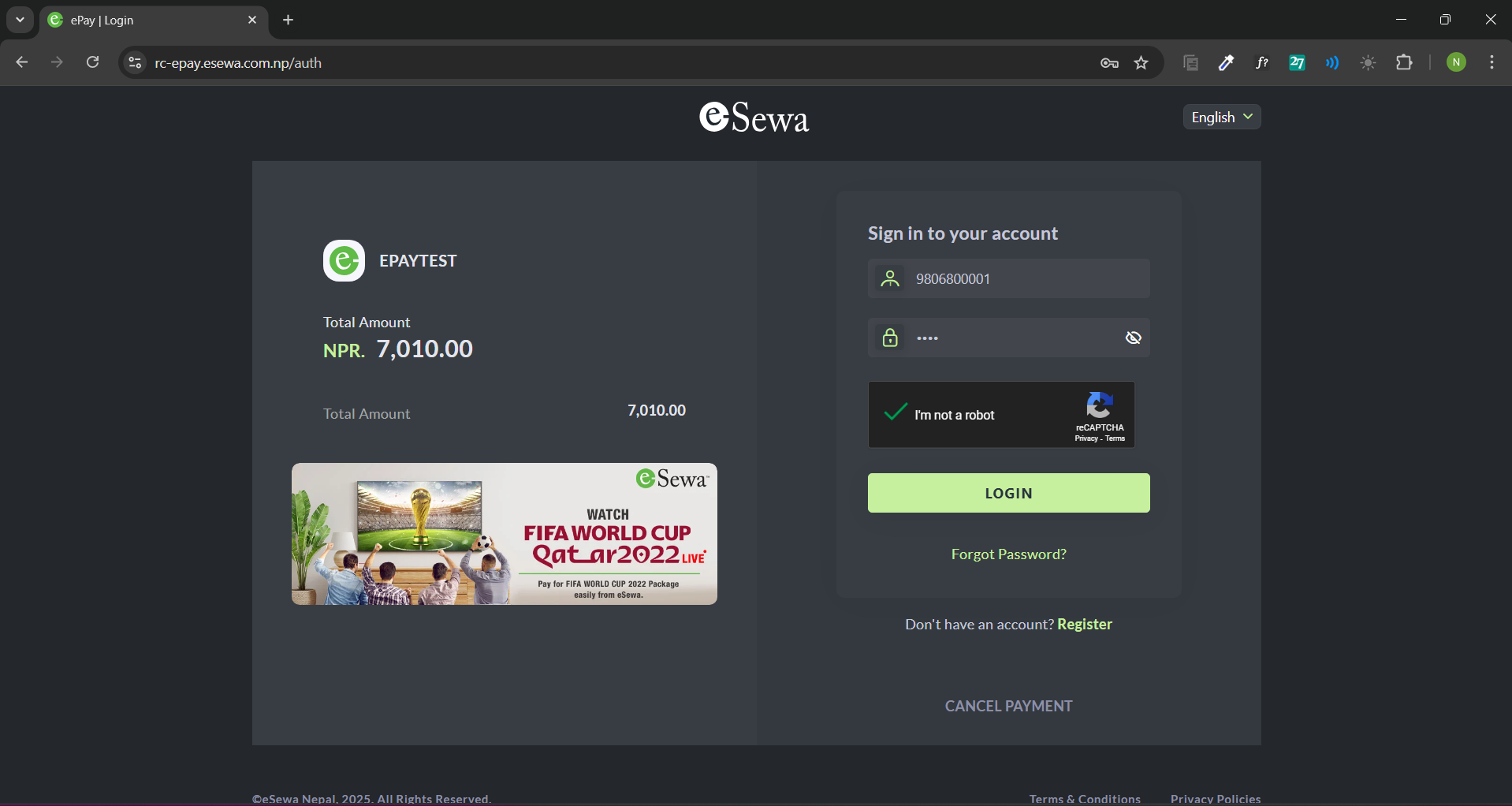


Figure 16: eSewa Login Page

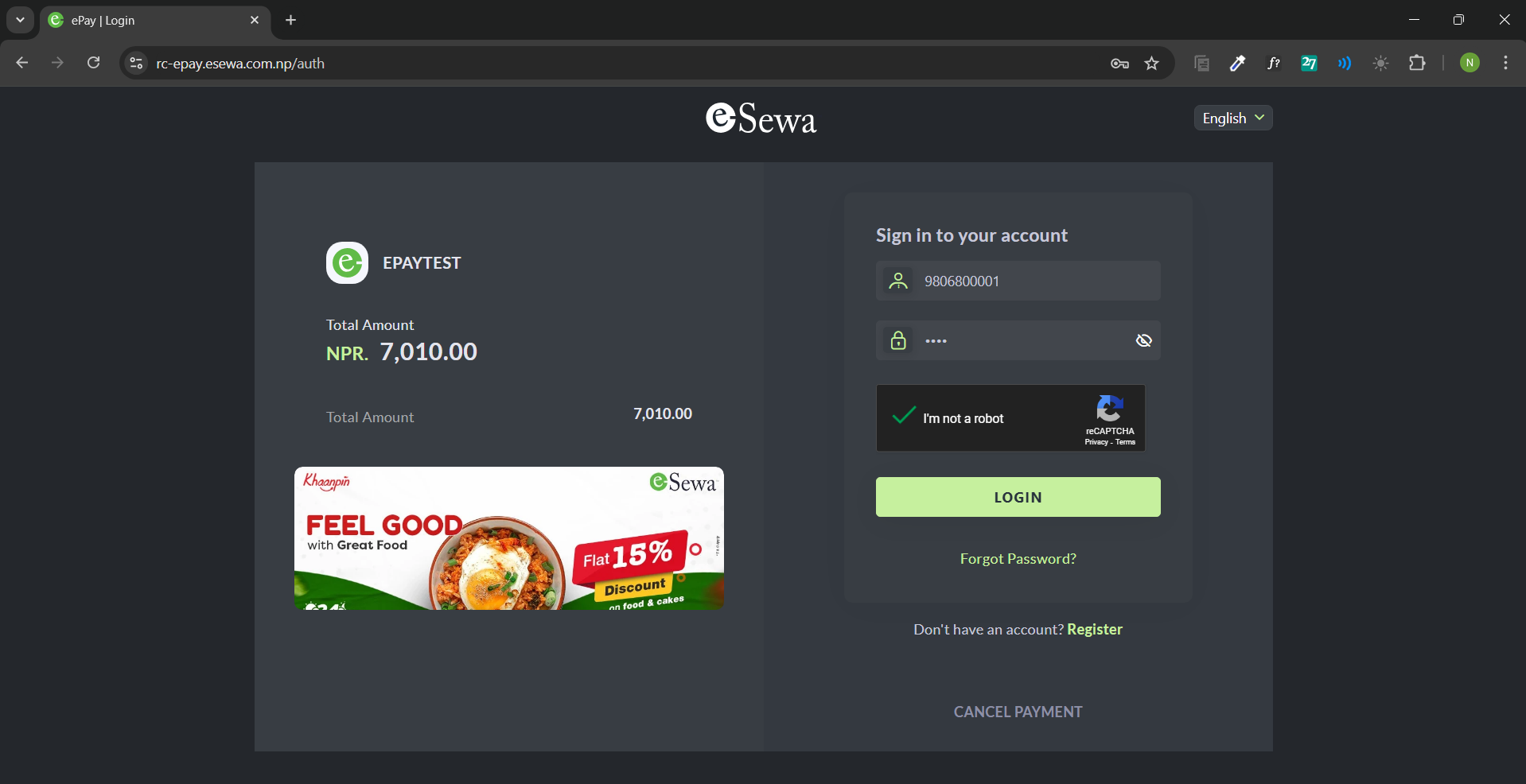


Figure 17: Captcha to secure login

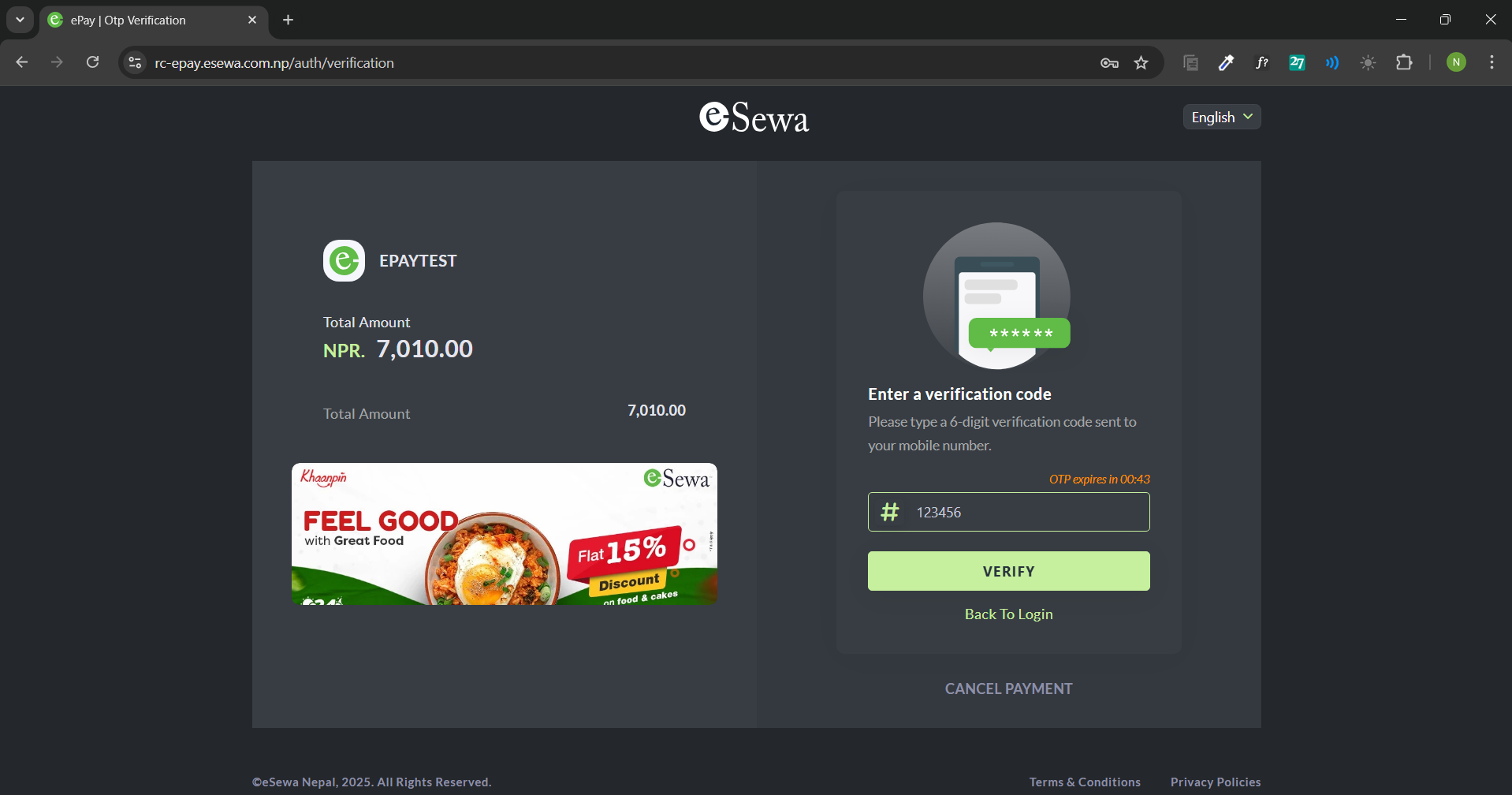


Figure 18: OTP received via number/email

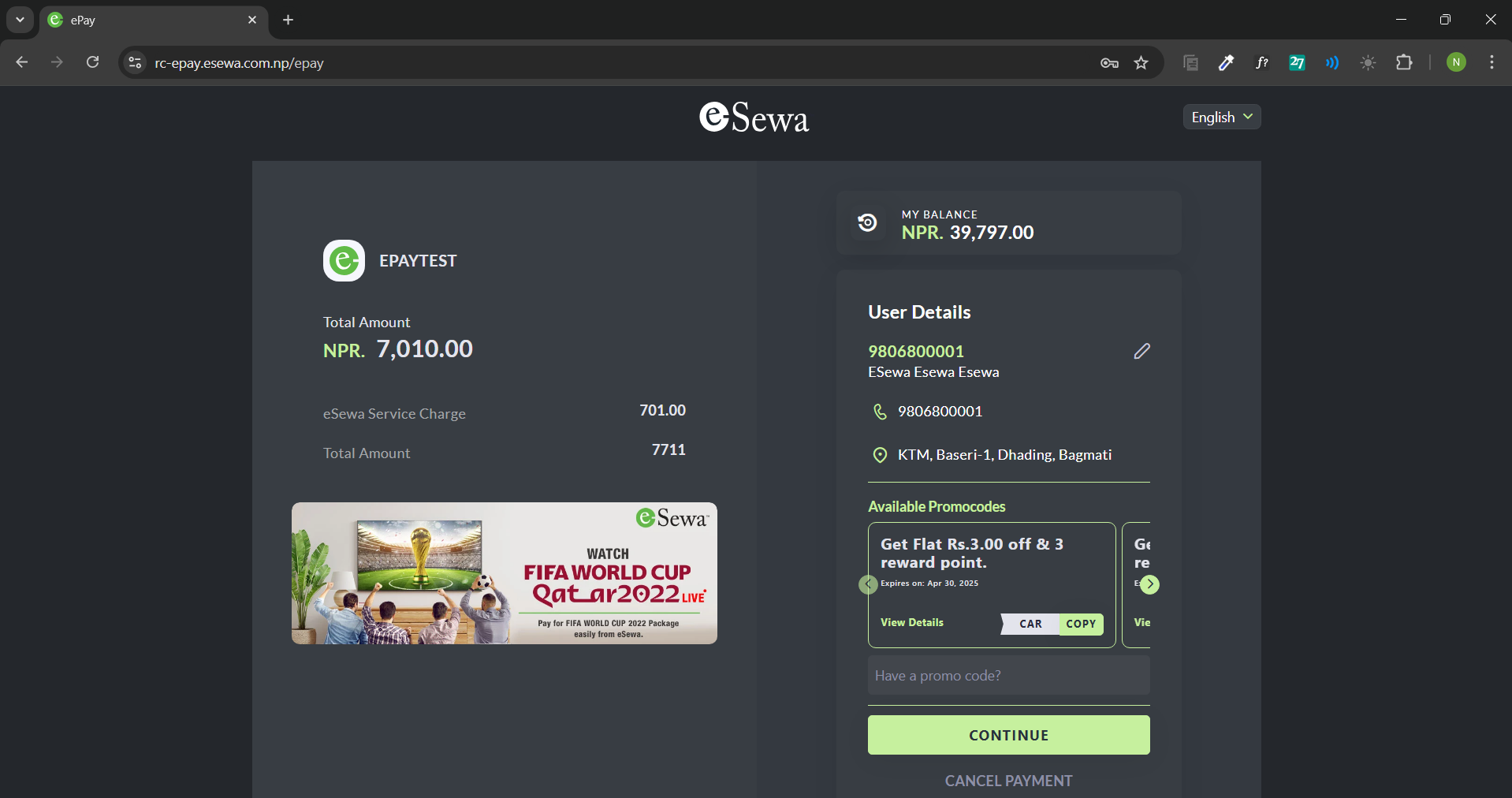
**

Figure 19: Continue/Cancel Payment

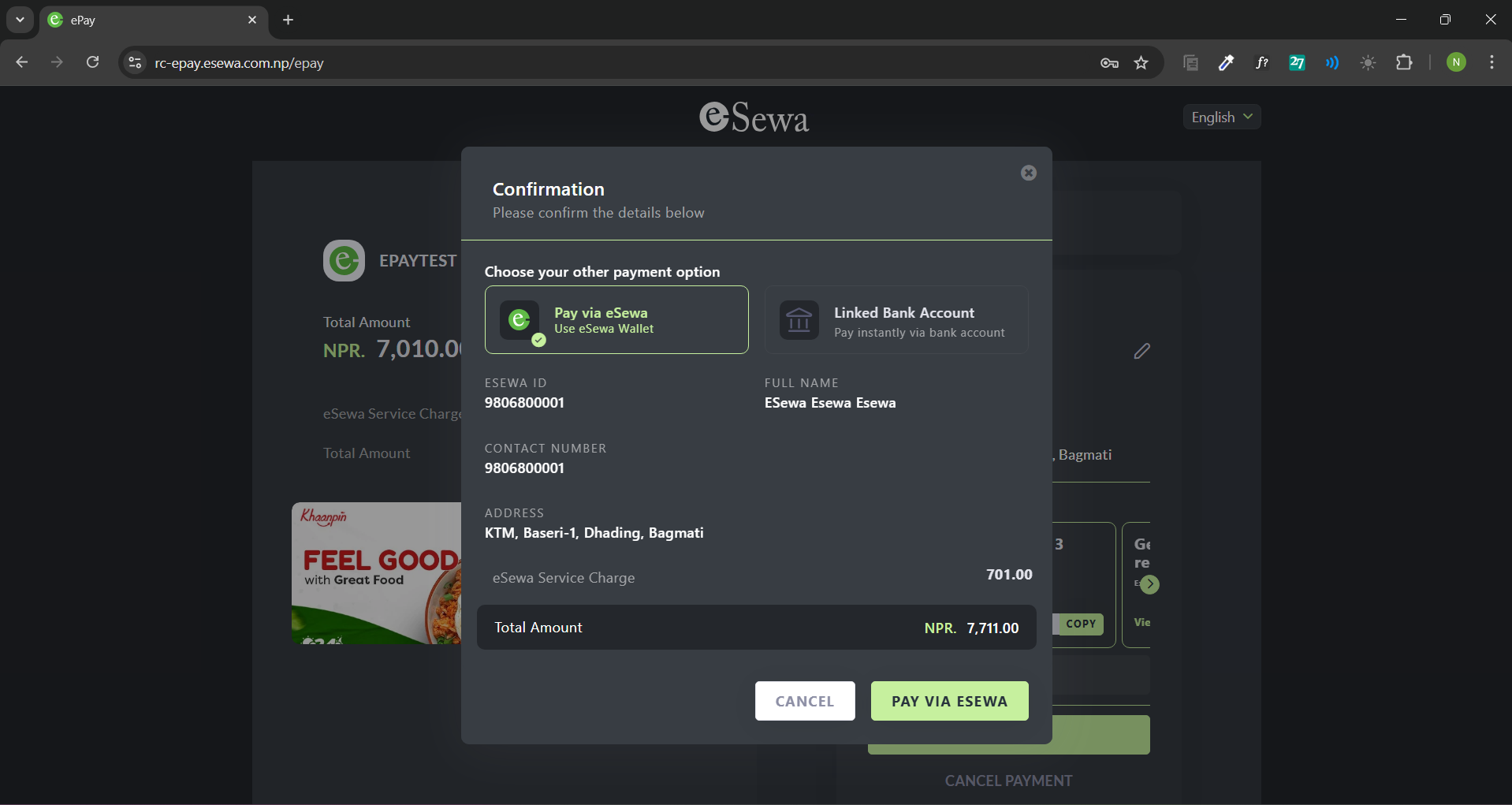
**

Figure 20: Payment via eSewa

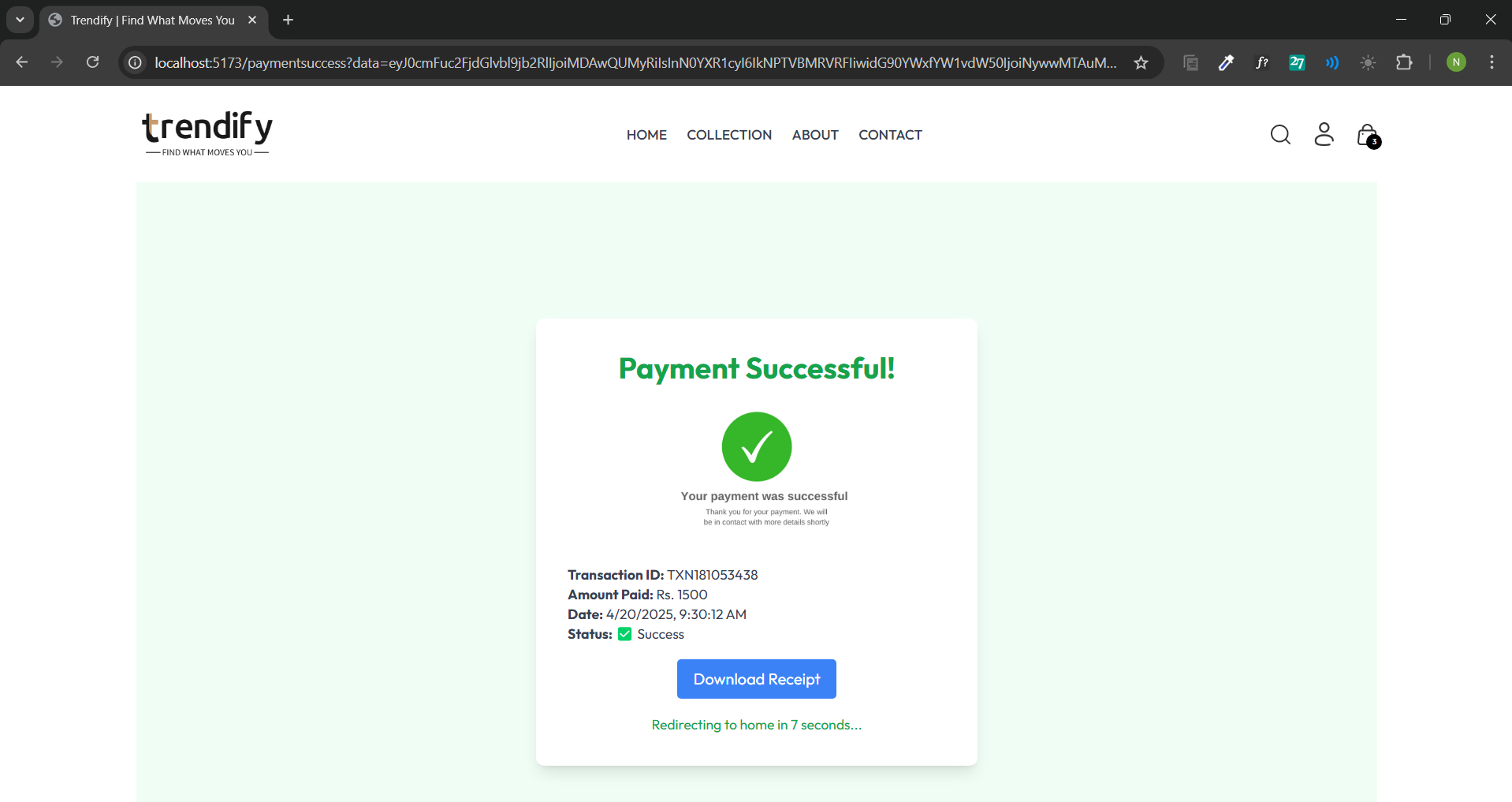
**

Figure 21: Transaction Statement