@doorz – A Multi-Vendor Website

By

MUHAMMAD ABUBAKAR SIDDIQUE	2022-GCUF-05733
NAILA SHAFIQUE	2022-GCUF-05736
SAUD AHMAD	2022-GCUF-05737

Project submitted in partial fulfilment

Of the requirements for the degree of

BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

Government College University Faisalabad

2024

Acknowledgement

In the name of Allah, the Most Merciful and the Most Gracious, we are incredibly grateful to Allah Almighty for giving us the courage, perseverance, and knowledge necessary to start and finish this project. His immense knowledge and guidance have served as the cornerstone of our academic career and have been essential in helping us overcome the obstacles we have faced.

We are grateful to our supervisor, Dr. Uzma Jameel, for her constant support, eloquent advice, and invaluable comments throughout this endeavor. Her knowledge and support have been instrumental in helping us improve our work and ensure that we meet the highest requirements for academic qualifications. His guidance has improved our technical expertise and inspired us to drive innovation and ongoing development.

We would like to express our sincere gratitude to the instructors and staff of the Department of Computer Science, Government College University, Faisalabad, who provided us with the tools and supportive environment that helped us complete this project. His commitment to fostering a culture of research and learning has greatly aided our academic and personal growth.

We also want to express our sincere gratitude to our families, who have been our greatest source of support with their love, tolerance, and sacrifices. Their unwavering encouragement and support have served as the foundation around which our academic success has been built. This endeavor would never have been possible without their undying faith in our abilities.

We would especially like to thank our friends and co-workers who have inspired us and offered us invaluable advice and support during this endeavor. This journey has been memorable and enriching because of their friendship and spirit of cooperation.

Finally, we thank Allah for His blessings, including our good health, our ability to persevere, and the opportunity to learn and grow. We pray that He will continue to guide and help us in all that we do in the future.

Muhammad Abubakar Siddique

Roll # 226690

Signature_______________

Saud Ahmad

Roll # 226695

Signature_______________

Naila Shafique

Roll # 226694

Signature_______________

Dedication

We dedicate this effort to Almighty Allah, the source of all wisdom and knowledge, whose immense mercies and blessings have made it possible for us to reach this important achievement. We have faced many difficulties along the way, but with His heavenly guidance, we have been able to overcome them and come out strong and determined.

Our success is largely due to the love, sacrifices, and perseverance of our parents, to whom we dedicate this effort. Our motivation comes from their unwavering support and belief in our abilities. This success is a testament to their commitment and the morals they have taught us. We are incredibly grateful to them for being our rock and for their endless tolerance and compassion.

We dedicate this effort to our siblings and extended family as a thank you for their unwavering encouragement and support. Their belief in us has motivated us to strive for excellence in all that we do and has been a source of inspiration.

This commitment also goes to our wonderful supervisor, Dr. Uzma Jameel, whose guidance was very helpful. His advice, tolerance, and insightful criticism have been invaluable to the success of our project. We sincerely appreciate the help and expertise that he has shared with us.

Additionally, we dedicate this initiative to our friends and co-workers who have supported us along the way. Their cooperation, support, and helpful criticism have been invaluable in bringing this idea to fruition. It has been a rewarding and enriching experience because of their friendship and support.

Finally, we dedicate this work to all the teachers, mentors, and friends who have helped us in our academic careers. His commitment to education and his love of learning have inspired us to work hard and honor our goals. This project is an expression of their impact on our lives and our commitment to preserve knowledge and creative heritage.

Muhammad Abubakar Siddique

Roll # 226690

Signature_______________

Saud Ahmad

Roll # 226695

Signature_______________

Naila Shafique

Roll # 226694

Signature_______________

CERTIFICATE BY SUPERVISORY COMMITTEE

This is to certify that the project report entitled "**@doorz – A multi vendor website**" which is submitted by **Muhammad Abubakar Siddique** Registration No. **2022-GCUF-05733**, **Saud Ahmad** Registration No. **2022-GCUF-05737** and **Naila Shafique** Registration No. **2022-GCUF-05736** has been examined by the undersigned as a part of the examination for the award of the degree of Bachelor of Computer Science from Government College University Faisalabad, Pakistan.

Supervisor:

____

Dr. Uzma Jameel
Assistant Professor

Chairperson:

____

Dr. Muhammad Kashif Hanif
Associate Professor
Department of Computer Science
Government College University, Faisalabad.

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction to the Problem

1.1 **Introduction**

Due to the rapid growth of the online industry, e-commerce platforms are now essential for both customer delight and business expansion. The need for a stable and dynamic multi-vendor e-commerce platform has never been greater. This need is met by the @doorz initiative, which creates a vast online marketplace aimed at promoting seamless communication between buyers and sellers. The project uses Redux for state management in conjunction with the MERN stack, which consists of MongoDB, Express.js, React, and Node.js. These technological advances guarantee that the platform is user-friendly, scalable, and efficient, meeting the demands of both modern suppliers and consumers.

@doorz aims to revolutionize the online shopping experience by providing a flexible and welcoming environment that fosters the growth of small and medium-sized enterprises (SMEs). It is not just another e-commerce platform. @doorz will bridge the gap between suppliers and buyers using innovative technology solutions and a simple and efficient marketplace that promotes company expansion and enhances customer happiness.

1.2 Background

E-commerce has emerged as a major sales channel as a result of the fundamental transformation of the retail industry as a result of the rapid spread of mobile and Internet technologies. Online marketplaces are gradually replacing or complementing traditional brick-and-mortar retailers as they provide unparalleled accessibility and convenience. As they can bring together a wide range of sellers and products, multi-vendor e-commerce platforms have become increasingly popular in this context, providing consumers with a one-stop shop for all their shopping needs.

However, there are often issues with scalability, security, and user experience with the multivendor platforms that now exist. Many small and medium-sized businesses (SMEs) find it difficult to build a solid online presence due to their expensive operating costs and lack of technical know-how. By offering SMEs an easy-to-use, scalable, and secure platform,

@doorz aims to solve these problems while also improving the overall shopping experience for customers.

1.3 Purpose

The primary goal of the @doorz project is to use the MERN stack (MongoDB, Express, React, and Node.js) as well as Redux for state management to design and build a robust multi-vendor e-commerce platform. The platform strives to:

- **Empower Vendors:** Give SMEs the tools and assets they need to monitor their stores, monitor revenue, and grow their customer base with minimal overhead.
- **Improve the Customer experience:** Provide a secure checkout procedure, an intuitive user interface, and advanced search features to guarantee client happiness.
- **Leverage Advanced technologies:** Build a scalable, responsive application with real-time updates and the ability to handle large levels of traffic using the MERN stack.

1.4 Scope

The scope of the @doorz project includes building an e-commerce platform with multiple vendors that has the following features:

- **Vendor Dashboard:** Vendor dashboards are comprehensive control panels that allow vendors to monitor orders, inventory, and sales data.
- **Customer Accounts:** Personalized shopping experiences with wish lists, transaction histories, and order monitoring are available through customer accounts.
- **Product Listings:** Comprehensive product pages that include dynamic pricing options and multimedia support.
- **Search and Filters:** Advanced search features and filters are provided to help users find products quickly and efficiently.
- **Checkout and Shopping Cart:** Simple, secure procedures that help customers complete their transactions.
- **Payment Gateway Integration:** To enable smooth transactions, several secure payment methods are provided.

- **Order Management:** Reliable back-end systems that allow suppliers to handle orders, track status updates, and monitor shipping.
- **Review and Rating System:** Mechanisms for users to rate and review products and vendors, fostering community trust and engagement.
- **Responsive Design:** Ensuring compatibility with various devices, including desktops, laptops, tablets, and smartphones.

1.5 Objective

The objectives of the @doorz project are as follows:

- **To create an efficient, scalable, multi-vendor e-commerce platform** that facilitates the expansion of small and medium enterprises (SMEs) by offering a comprehensive, user-friendly environment for managing their online presence. does.
- **To improve the online shopping experience for consumers** by providing a secure transaction processing system, advanced search features, and an easy-to-use interface.
- **Strong security mechanisms** should be in place to protect user information and guarantee the integrity of transactions, instilling trust and reliability in the process.
- **To give suppliers access to comprehensive analytics tools** that will help them understand market trends, consumer behavior, and sales success. This will allow them to make data-driven decisions and improve their business.
- **Offering a long-term and innovative solution** for online sales to guarantee that the platform is flexible enough for future expansion as well as advancements in technology.

1.6 Intended Audience and Reading Suggestions

The target audience for this paper is broad and includes:

- **Project Supervisor**: Charged with monitoring and ensuring that the project meets technical and academic requirements.
- **Developers**: Developers are charged with implementing the platform's technical framework. They will discover the comprehensive technical specifications, schematics, and installation instructions necessary to build the platform.
- **Graphic Designers**: Graphic designers are tasked with creating user interfaces and user experiences. The document will be used by designers to understand the functional specifications and design principles that guide the creation of the platform's interactions and visual components.
- **Project Manager:** responsible for overall project coordination, resource allocation, risk management, and planning. The document will serve as the project manager's reference for comprehensive project schedules, resource requirements, and risk mitigation strategies.

Reading Suggestions:

- Overview: Provides all readers with a basic understanding of the project's purpose by providing a quick explanation of the project's vision, objectives, and impact
- Project Description: For most project managers and developers, this outlines goals, expected results, and techniques.
- Design and Development: Describes specific processes, tools, and design philosophies that developers and designers use.
- Project Management: For most project managers, this focuses on resource allocation, risk management, and project timeframes.
- Testing and Quality Assurance: For most developers and project managers, this section covers specific techniques for assuring software quality and reliability.

1.7 Documentation Conventions

This document follows specific conventions to ensure clarity, consistency, and professionalism. The conventions used throughout the document are as follows:

- **Font and formatting:** Times New Roman is used throughout the work, with body text set at 12 points, main headings set at 16 points, and subheadings set at 14 points. Section and subsection headings are distinguished by bolding and capitalization.

- **Lists:** Products, features, and benefits are presented using bulleted lists. Sequences of actions or systematic steps are represented using numbered lists.
- **Emphasis:** Key words and phrases are underlined in the text to draw attention to them. Instructions and important parts can also be underlined for emphasis.
- **Tables and Figures:** All tables and figures in the paper have a sequential numbering system and are properly referenced in the text. This guarantees that every visual aid can be referenced and easily identified.

Chapter 2:

Software Requirements Specification

2.1 Overall Description

The @doorz platform is a multi-vendor e-commerce solution designed to cater to the needs of both vendors and consumers. Unlike traditional e-commerce websites, @doorz allows multiple vendors to manage their storefronts, products, and sales under a single platform. This project leverages the MERN stack—comprising MongoDB, Express.js, React, and Node.js—along with Redux for state management. The choice of this technology stack ensures a scalable, flexible, and efficient online marketplace that can handle a high volume of transactions and user interactions seamlessly.

The platform's architecture is designed to facilitate easy integration with third-party services such as payment gateways and shipping providers, ensuring secure transactions and real-time tracking capabilities. Additionally, the responsive design ensures that the platform is accessible and user-friendly across various devices and screen sizes, from desktops to smartphones.

2.1.1 Product Perspective

The @doorz project seeks to provide an innovative, multi-vendor e-commerce platform that improves the user experience by leveraging contemporary web technologies and addressing the shortcomings of existing solutions. The platform will act as a dynamic marketplace where customers can have a seamless and secure shopping experience and vendors can manage their products, track sales, and interact with customers.

- **Market Positioning:** With an emphasis on scalability, security, and user-friendliness, @doorz aims to be a preeminent multi-vendor platform. Its target market includes small and medium-sized businesses (SMEs) as well as a wide range of consumers looking for a variety of products.
- **Stakeholder Participation:** Key stakeholders involved are:
    - **Vendors:** business owners who need a stable platform to manage and list their goods.
    - **Consumers:** end users looking for a safe and convenient way to shop online.
    - **Administrators:** platform managers in charge of managing user complaints, ensuring compliance, and monitoring operations.

    - **Developers:** The technical group in charge of building and managing the platform.

- **Technical integration:** The platform guarantees a unified development process by leveraging the MERN stack. React provides a dynamic user interface, Express.js is a robust server framework, MongoDB provides a configurable database solution, and Node.js facilitates efficient server-side operations. Redux optimizes state management to guarantee a reliable and consistent user experience.
- **Scalability and adaptability:** The platform architecture is designed to withstand increasing traffic and data volumes. Its modular design allows for future improvements and the easy inclusion of new features.

2.1.2 Product Features

1. Vendor Dashboard:

A comprehensive system that enables vendors to manage their profiles, product listings, and orders efficiently.

- **Overview:** The vendor dashboard is a centralized control panel where vendors can manage their entire store. This includes inventory management, order processing, sales tracking, and customer interactions.
- **Sales Analytics:** Vendors can access detailed analytics, including sales reports, product performance metrics, customer demographics, and behavior patterns. This data helps vendors make informed decisions to optimize their sales strategies.
- **Order Management:** Vendors can view and manage all orders, from new orders to processing, shipping, and completed orders. The system supports order status updates, handling returns, and managing refunds.
- **Inventory Management**: Vendors can add new products, update existing product details, manage stock levels, and set dynamic pricing options. The system supports bulk product uploads through CSV files for ease of management.
- **Communication Tools:** Vendors have a built-in messaging system to communicate with customers directly, addressing their queries and resolving issues promptly.

2. **Customer Accounts:**

A secure and personalized space for customers to manage their information, orders, and interactions with vendors.

- **Registration and Profile Management:** Customers can create accounts, update personal information, and manage their profiles. The registration process includes email verification for added security.
- **Order Tracking:** Customers can track the status of their orders in real-time, from the moment they place an order until it is delivered.
- **Wishlist:** Customers can add products to their wishlist for future reference and easy access.
- **Reviews and Ratings:** Customers can rate and review products they have purchased, providing feedback to other customers and vendors. Reviews are moderated to ensure quality and relevance.

3. **Product Listing:**

A robust system that allows vendors to create, update, and manage their product listings with ease.

- **Multimedia Support:** Vendors can upload high-resolution images, videos, and detailed descriptions for each product. This includes support for 360-degree product views and augmented reality previews.
- **Dynamic Pricing:** Vendors can set regular prices, discounts, promotional prices, and schedule price changes in advance.
- Categorization and Tags: Products can be categorized into multiple categories and tagged with relevant keywords to enhance searchability and navigation.

4. **Search:**

An intuitive search engine with advanced options to help customers find products quickly and easily.

- **Advanced Search:** The search engine supports full-text search capabilities with real-time auto-complete suggestions based on user input.

5. **Shopping Cart and Checkout:**

A seamless process for customers to add products to their cart and complete purchases securely.

- **Cart Management:** Customers can add, remove, and modify products in their shopping cart. The cart is saved across sessions, allowing users to continue shopping at their convenience.
- **Multiple Checkout Steps:** The checkout process is divided into multiple steps: entering shipping information, selecting a payment method, and reviewing the order before final confirmation.
- **Payment Methods:** The platform integrates with multiple payment gateways, including Stripe for credit/debit card payments, PayPal, and Cash on Delivery (COD) for local purchases.
- **Order Confirmation:** Customers receive email notifications upon successful order placement, including order details and estimated delivery times.

6. **Payment Gateway Integration**

Secure and flexible payment gateway integration to facilitate various payment methods.

- **Support for Major Payment Gateways:** Integration with widely-used payment gateways such as Stripe, PayPal, and traditional banking APIs.
- **Encryption and Security Compliance:** All transactions are encrypted using SSL/TLS protocols to ensure data security and compliance with industry standards.
- **Real-time Payment Confirmation:** Immediate confirmation of payment status, allowing both customers and vendors to see real-time updates on transaction statuses.
- **Multiple Payment Options**: Support for various payment methods including credit/debit cards, PayPal, and COD.

7. **Order Management and Fulfillment**

Efficient back-end system for processing orders and managing fulfillment.

- **Order Receipt and Confirmation:** Automated systems generate and send order confirmations to customers and vendors.
- **Status Updates:** Real-time updates on order status including processing, shipping, and delivery. Notifications are sent at each stage.
- **Returns and Refunds:** Streamlined process for handling returns and refunds, including automated updates to stock levels and financial records.

8. **Review and Feedback Mechanism**

Allows users to leave reviews and ratings for products and vendors, fostering trust and community.

- **Product Reviews:** Customers can leave detailed reviews and ratings for products they have purchased. Reviews can include a rating system, written feedback, and the option to upload images.
- **Vendor Feedback:** Customers can provide feedback on their shopping experience with specific vendors. This helps build trust and credibility on the platform.

9. **Responsive Design**

Ensuring the platform is accessible and user-friendly across various devices and screen sizes.

- **Cross-Device Compatibility:** The platform is designed to work seamlessly on desktops, laptops, tablets, and smartphones, providing a consistent user experience across devices.
- **Adaptive UI/UX:** The user interface adapts to different screen sizes and resolutions, ensuring optimal usability regardless of the device used.
- **Touch-Friendly Interfaces:** The platform includes touch-friendly interfaces for mobile devices, making it easy to navigate and interact with on smartphones and tablets.

10. **Administrative Control Panel**

A central dashboard for administrators to manage the platform, including users, vendors, and content.

- **User and Vendor Management:** Tools for managing user and vendor accounts, including approval processes, role assignments, and access controls.
- **Content Moderation:** Systems for monitoring and moderating product listings, reviews, and user interactions to ensure compliance with platform policies.
- **System Configuration:** Settings for configuring platform-wide features, including payment options, shipping methods, and site policies.

2.1.3 Design and Implementation Constraints

Several limitations affect the design and implementation of the @doorz platform:

1. **Technology Stack:**
   - The application is built using the MERN stack (MongoDB, Express.js, React, Node.js), which may restrict the use of certain third-party tools and libraries not compatible with these technologies.
   - Redux is used for state management across the application to maintain a predictable and consistent state.
2. **Deployment and Scalability:**
   - The initial deployment will focus on a specific market or region with plans to expand geographically in subsequent phases.
   - The system must be designed to scale efficiently to handle increased traffic and data load without compromising performance.
3. **Security and Compliance:**
   - Strict security measures need to be implemented to protect user data, including encryption for data at rest and in transit, and compliance with industry standards like GDPR.
   - Role-based access control is essential to ensure users only have access to their permitted features and data.
4. **Performance Requirements:**
   - The application should support a high number of concurrent users (e.g., 1,000 vendors and 10,000 customers) without performance degradation.
   - Page load times should be optimized to ensure a smooth user experience, targeting load times of less than 3 seconds for key pages.
5. **Cross-Platform Compatibility:**
   - The front-end must be responsive, providing a seamless user experience across various devices and screen sizes, including desktops, tablets, and smartphones.
6. **Database Management:**
   - MongoDB is used for database management, requiring careful schema design to handle the hierarchical data structure efficiently.

7. **Integration with Third-Party Services:**
    - The system relies on third-party services for payment processing (Stripe, PayPal) and shipping integrations, which necessitates reliable and secure API integrations.
    - Any changes or outages in these third-party services could impact the platform's functionality.
8. **User Interface and Experience:**
    - The UI should be intuitive and user-friendly to accommodate users with varying levels of technical proficiency.
    - Consistency in design and navigation across the platform is crucial to enhance usability.
9. **Testing and Quality Assurance:**
    - Comprehensive testing, including unit tests, integration tests, and user acceptance tests, must be conducted to ensure the reliability and quality of the application.
    - Automated testing tools should be employed where possible to streamline the testing process.
10. **Maintenance and Updates:**
    - The codebase should follow clean code principles and be well-documented to facilitate easy updates and maintenance.
    - A version control system (e.g., Git) should be used to manage code changes and collaboration among developers.

2.1.4 Assumptions and Dependencies

1. **User Familiarity:**
    - It is assumed that target users (vendors and customers) have a basic understanding of online shopping and e-commerce platforms.
2. **Technical Infrastructure:**
    - The performance and scalability of the platform depend on the underlying cloud hosting services (e.g., AWS, Azure) used for deployment.
    - Reliable internet connectivity is required for seamless operation and access to the platform.
3. **Third-Party Service Availability:**
    - The platform's functionality depends on the availability and reliability of third-party services such as payment gateways (Stripe, PayPal) and shipping providers.

- Continuous updates and support from these services are assumed for uninterrupted platform operations**.**

4. **Regulatory Compliance:**
   - The platform must comply with relevant data protection and privacy laws (e.g., GDPR), and it is assumed that necessary legal consultations will be undertaken to ensure compliance.
   - Regular audits and updates may be required to stay compliant with changing regulations.
5. **User Engagement:**
   - The platform's success relies on active engagement from both vendors and customers. It is assumed that there will be sufficient marketing and outreach efforts to attract and retain users.
   - User feedback mechanisms will be in place to continuously improve the platform based on user needs and preferences.
6. **Development Resources:**
   - The project assumes availability of skilled developers proficient in the MERN stack and Redux for both initial development and ongoing maintenance.
   - Adequate resources (time, budget, personnel) will be allocated to meet project milestones and deadlines.
7. **Scalability Planning:**

   - It is assumed that the initial architecture will be designed with scalability in mind, allowing for seamless expansion as user numbers grow.
   - Future scalability plans will include provisions for additional server resources, database optimization, and load balancing solutions.

These sections address the necessary constraints and assumptions for the design and implementation of the multi-vendor platform, ensuring a robust, scalable, and user-friendly solution.

2.2 System Features

2.2.1 System Feature 1: Vendor Management System

Description: A comprehensive system that allows vendors to create and manage their profiles, products, and orders.

Features:

- **Vendor Registration and Login:** Vendors can create accounts, log in, and manage their profiles.
- **Dashboard:** Vendors have access to a dashboard to manage their inventory, track sales, and view analytics.
- **Product Management:** Vendors can add, update, and delete product listings with detailed descriptions, images, and pricing.
- **Order Management:** Vendors can view and manage orders, update order status, and handle returns and refunds.
- **Communication Tools:** Vendors can communicate with customers and site administrators via integrated messaging.

2.2.2 System Feature 2: Product Management System

Description: This feature allows vendors to create, update, and manage their product listings, ensuring accurate and detailed product information.

Features:

- **Product Listings:** Vendors can create product listings with comprehensive details, including titles, descriptions, images, and prices.
- **Pricing Options:** Support for dynamic pricing, discounts, and promotional offers.
- **Categorization:** Products can be categorized for easy navigation and searchability.
- **SEO Optimization:** Fields for meta titles, descriptions, and keywords to enhance search engine visibility.

2.2.3 System Feature 3: Customer Account Management

Description: A secure and personalized space for customers to manage their profiles, orders, and communications with vendors.

Features:

- **Customer Registration and Login:** Customers can create accounts, log in, and manage their profiles.

- **Order Tracking:** Customers can view their order history, track order status, and manage returns.
- **Wishlist:** Customers can save products to their wishlist for future purchases.
- **Product Reviews and Ratings:** Customers can leave reviews and ratings for purchased products.
- **Profile Management:** Customers can update their personal information, addresses, and payment methods.

2.2.4 System Feature 4: Search and Navigation

Description: An intuitive search engine that allows users to find products quickly using keywords, categories, and filters.

Features:

- **Keyword Search:** Users can search for products using keywords, with auto-complete suggestions.

2.2.5 System Feature 5: Shopping Cart and Checkout Process

Description: A seamless process that enables customers to review their selections, make changes, and complete purchases securely.

Features:

- **Add to Cart:** Users can add products to their shopping cart from product listings or detail pages.
- **Cart Summary:** A summary of the cart with options to edit quantities or remove items.
- **Checkout Process:** A multi-step checkout process including shipping details, payment options, and order review.
- **Payment Methods:** Support for multiple payment options including credit/debit cards, PayPal, and cash on delivery.
- **Order Confirmation:** Email notifications for order confirmation and updates.

2.2.6 System Feature 6: Payment System Integration

Description: A secure payment gateway system that supports various payment methods and ensures secure transaction processing.

Features:

- **Multiple Payment Gateways:** Integration with major payment gateways like Stripe and PayPal.

- **Transaction Security:** Encryption of payment data to ensure secure transactions.
- **Payment Confirmation**: Immediate confirmation of payments with detailed receipts sent to customers.

Refund Processing: Efficient processing of refunds for returned or canceled orders.

2.2.7 System Feature 7: Order Management and Fulfillment

Description: A backend system for vendors to process orders, update status, and manage fulfillment efficiently.

Features:

- **Order Processing:** Vendors can view new orders, update their status, and manage shipping details.
- **Shipping Integration:** Integration with shipping carriers for real-time tracking and shipping label generation.
- **Order History:** Detailed order history for vendors and customers, including tracking information and delivery status.
- **Return and Refund Management:** Tools for handling returns, exchanges, and refunds.

2.2.8 System Feature 8: Review and Feedback Mechanism

Description: A feature that allows users to leave reviews and ratings for products and vendors, fostering trust and community.

Features:

- **Product Reviews:** Customers can leave reviews and star ratings for products they have purchased.
- **Vendor Ratings:** Vendors can be rated based on their service quality and product offerings.
- **Feedback Moderation:** Tools for administrators to moderate and manage reviews and feedback.

2.2.9 System Feature 9: Administrative Control Panel

Description: A central dashboard for administrators to monitor the entire platform, manage users, and make site-wide changes.

Features:

- **User Management:** Tools to manage vendor and customer accounts, including approvals and suspensions.

- **Content Moderation:** Moderation of product listings, reviews, and other user-generated content.
- **System Configuration:** Settings for managing site-wide configurations, such as payment options and shipping settings.

2.2.10 System Feature 10: Event Management System

Description: A feature that allows vendors to create and manage promotional events and offers to attract customers.

Features:

- **Event Creation:** Vendors can create events and promotional offers for specific products or categories.
- **Event Scheduling:** Ability to schedule events for specific dates and times.
- **Event Management:** Tools to update, edit, or cancel events as needed.

2.2.11 System Feature 11: Communication and Messaging

Description: A robust communication system that allows vendors, customers, and administrators to interact seamlessly.

Features:

- **Vendor-Customer Messaging:** Direct messaging between vendors and customers for inquiries and support.
- **Support Tickets:** A ticketing system for customers to raise support issues, which vendors and admins can track and resolve.
- **Chat Integration:** Real-time chat support for immediate assistance.

2.2.12 System Feature 13: Inventory Management

Description: A feature for vendors to manage their inventory efficiently, ensuring products are always in stock.

Features:

- **Stock Levels:** Real-time tracking of product stock levels.

2.2.13 System Feature 14: Shipping and Delivery Management

Description: An integrated system to manage shipping and delivery processes, ensuring timely and accurate deliveries.

Features:

- **Shipping Options:** Multiple shipping options for customers, including standard and expedited shipping.

- **Shipping Label Generation:** Automatic generation of shipping labels based on carrier requirements.
- **Real-Time Tracking:** Integration with shipping carriers to provide real-time tracking information to customers and vendors.
- **Delivery Notifications:** Automated notifications for customers about shipping status and delivery updates.

2.2.14 System Feature 15: User Activation and Verification

Description: A feature to ensure the authenticity and security of user accounts through email verification and activation.

Features:

- **Email Verification:** Automated emails sent to users for account verification upon registration.
- **Activation Links:** Unique activation links for users to verify their email addresses and activate their accounts.

2.2.15 System Feature 16: Wishlist Management

Description: A convenient feature for customers to save products they are interested in purchasing later.

Features:

- **Add to Wishlist:** Customers can add products to their Wishlist directly from product listings or detail pages.
- **Manage Wishlist:** Tools for customers to view, edit, and remove items from their Wishlist.
- **Share Wishlist:** Option to share Wishlist with others via email or social media.

These system features comprehensively cover the functional aspects of the multi-vendor platform, ensuring a robust, user-friendly, and efficient e-commerce experience for vendors, customers, and administrators. The implementation of these features will leverage the strengths of the MERN stack and Redux to deliver a scalable and high-performing application.

2.3 External Interface Requirements

2.3.1 User Interfaces

The user interfaces are critical to ensuring a smooth and intuitive interaction with the system. The main user interfaces include:

- **Vendor Interface:** A web-based dashboard allowing vendors to manage their storefronts, products, orders, and view analytics.
- **Customer Interface:** A user-friendly front-end for customers to browse products, manage their accounts, place orders, and track order status.
- **Admin Interface:** An administrative portal for site management, user and content moderation, and accessing various reports and analytics.

2.3.2 Hardware Interfaces

The system is designed to be compatible with various hardware interfaces to ensure accessibility and functionality across different devices:

- **Responsive Design:** Ensures compatibility with desktops, laptops, tablets, and smartphones.

2.3.3 Software Interfaces

The software interfaces define the interaction between different software components within the system and external services:

- **Database:** MongoDB is used for storing user data, product lists, orders, etc.
- **Server:** The backend application server is built using Express.js running on Node.js.
- **Client-side:** The front-end user interface is built using React with Redux for state management.
- **Payment Gateway:** Integration with external payment services such as PayPal, Stripe, and traditional banking APIs for secure transactions.

2.3.4 Communications Interfaces

Effective communication interfaces are necessary for smooth operations and user interactions:

- **Email System:** Integration with an email service to send order confirmations, newsletters, and marketing materials.
- **Social Media Integration**: Interfaces with social media platforms for login, sharing, and promotional purposes.
- **RESTful API**: Backend services are exposed through a RESTful API for integration with third-party applications and services.

2.4 Other Nonfunctional Requirements

2.4.1 Performance Requirements

Performance requirements ensure that the system operates efficiently under expected conditions:

- **Load Time:** Web pages should load within less than 3 seconds to ensure a smooth user experience.
- **Concurrent Users:** The system must support up to 1,000 vendors and 10,000 customers simultaneously without performance degradation.
- **Scalability:** The system architecture should support scaling to accommodate an increasing number of users and data.

2.4.2 Safety Requirements

Safety requirements are essential to protect data integrity and system stability:

Transaction Rollback: The ability to rollback transactions in case of failure to ensure data integrity.

2.4.3 Security Requirements

Security requirements ensure that the system protects user data and transactions from unauthorized access and breaches:

Authentication: Strong authentication mechanisms to prevent unauthorized access for all users.

Authorization: Role-based access control to ensure users can only perform tasks they are authorized to do.

Data Encryption: Use encryption for sensitive data both at rest and in transit using industry-standard methods.

Chapter 3

Analysis (Use Case Model)

3.1 Identifying Actors and Use Cases Using Textual Analysis

Actors

1. Admin
   - **Role:** The admin oversees the entire system, ensuring smooth operations, managing user accounts, and maintaining platform integrity.
   - **Responsibilities:**
     - ➢ Manage user accounts, including sellers and customers.
     - ➢ Oversee and manage products listed on the platform.
     - ➢ Handle order approvals, disapprovals, and monitor transactions.
     - ➢ Access and manage site-wide settings
2. Seller
   - **Role:** The Seller manages their shop on the platform, listing products, handling orders, and interacting with customers.
   - **Responsibilities:**
     - ➢ Create and manage shop details.
     - ➢ List and manage products, including descriptions, prices, stock levels, and images.
     - ➢ Handle orders, including processing and fulfilling them.
     - ➢ Manage refunds and communicate with customers regarding orders and products.
     - ➢ Create and manage promotional events for products.
     - ➢ Manage financial transactions and withdrawal requests from the platform.
3. Customer/User
   - **Role:** The Customer interacts with the platform to browse products, make purchases, and manage their account.
   - **Responsibilities:**
     - ➢ Browse products and events listed by various sellers.
     - ➢ Add products to the cart and perform the checkout process.
     - ➢ Manage personal profile, including updating information and viewing order history.

        - Engage with the platform through contact forms and leave reviews for products.

4. System
    - **Role:** The System automates various processes, ensuring secure transactions, managing user sessions, and sending notifications.
    - **Responsibilities:**
        - Send notifications and emails for account verifications, order updates, and other communications.
        - Process payments through integrated payment gateways.
        - Manage user sessions, ensuring secure authentication and access control.

Use Cases

1. Admin Use Cases:
    - **Manage Dashboard:** Admin views earnings, events, orders, and user details on a centralized dashboard.
    - **Approve/Disapprove Withdrawal Requests:** Admin handles financial transactions of sellers, approving or disapproving withdrawal requests.
    - **Monitor Events and Products:** Admin oversees all events and products listed by sellers, ensuring compliance with platform policies.
    - **User Management:** Admin handles user accounts, including deletion of accounts.
    - **View Reports:** Admin accesses detailed analytics and reports on platform activities, including sales, user engagement, and system performance.

2. Seller Use Cases:
    - **Create Shop:** Seller fills out necessary details to create a new shop on the platform, including shop name, email, phone number, address, password, and logo.
    - **List Products:** Seller adds, edits, or deletes products with details such as name, description, price, sale price, stock levels, and images.
    - **Manage Orders:** Seller views, processes, and fulfills orders placed by customers.
    - **Handle Refunds:** Seller processes refund requests from customers, managing the status of refunds.
    - **Create Events**: Seller organizes and manages promotional events for specific products, offering discounts or special deals.

- **Communication:** Seller responds to customer queries and messages, managing interactions through the seller dashboard.

3. Customer Use Cases
    - **Browse Products**: Customer searches for products using keywords, filters, and categories.
    - **Add to Cart:** Customer adds desired products to the shopping cart.
    - **Checkout:** Customer completes the purchase process using various payment methods, including card payment, PayPal, and COD.
    - **Manage Profile:** Customer updates personal information, such as name, email, and address, and views order history.
    - **Contact Support:** Customer uses the contact form to send queries or issues to the admin.
    - **Write Reviews:** Customer leaves feedback and ratings for purchased products, helping other users make informed decisions.

4. System Use Cases:
    - **Send Notifications:** System sends email notifications for account verification, order confirmations, and updates.
    - **Process Payments:** System handles payment transactions securely through integrated payment gateways like Stripe and PayPal.
    - **Manage Sessions:** System manages user sessions, ensuring secure login and access control, maintaining session integrity.

3.2 Forming Use Case Diagram with Candidate and Use Cases

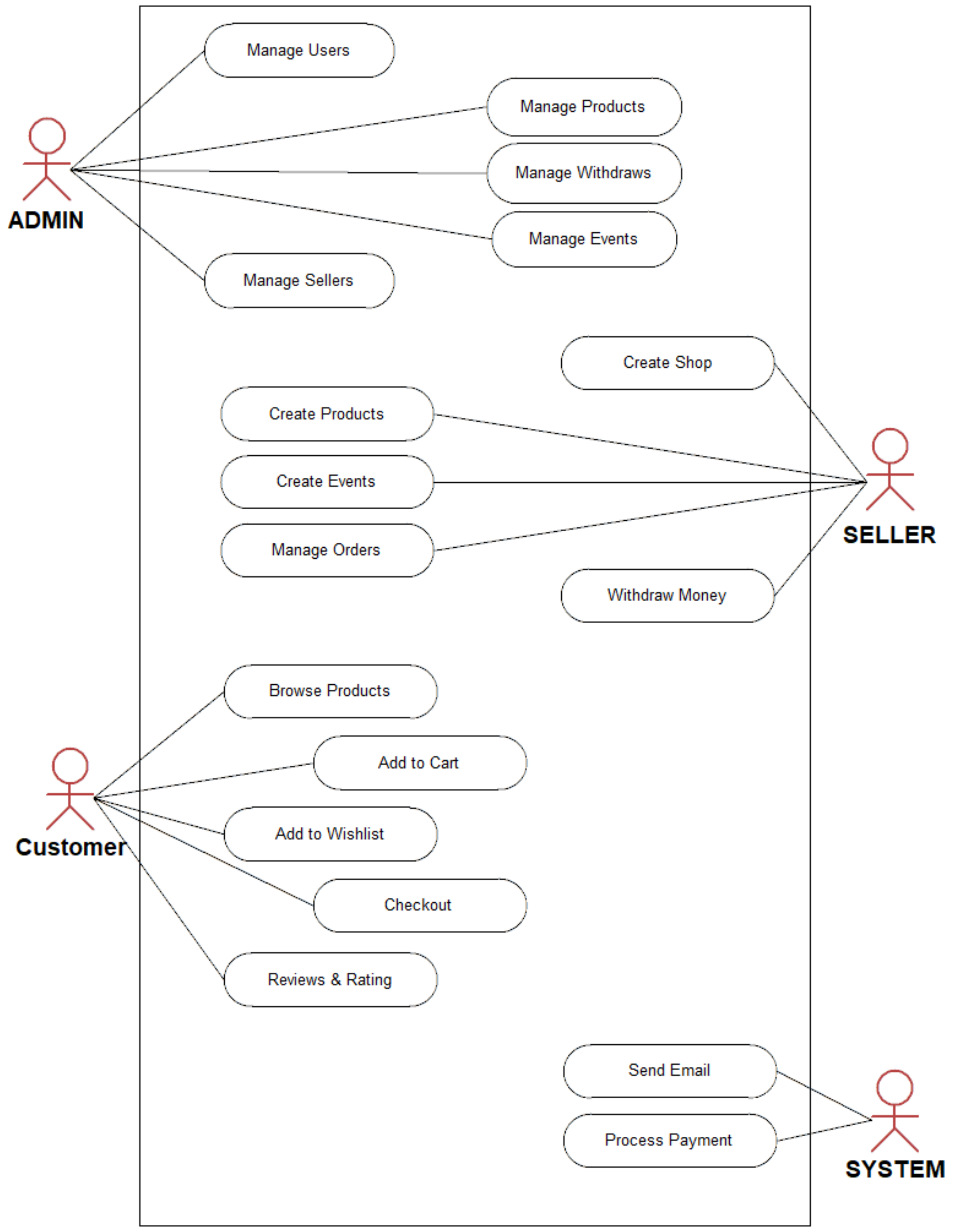


Figure 1: Forming use case diagram and candidate and use cases

3.3 Describe the Events Flow for Use Case

1. Admin Manages Users:
   - **Trigger:** Admin logs into the system.
   - **Preconditions:** Admin has appropriate access rights and is authenticated.
   - **Flow:**
     - ➢ Admin navigates to the user management section.
     - ➢ Admin views a list of all users, including sellers and customers.
     - ➢ Admin selects a user to view detailed information.

        - Admin performs actions such as editing user details, approving or suspending accounts, and deleting accounts.
        - Admin confirms the action, and the system updates the user information accordingly.
    - **Postconditions:** User information is updated in the database, and relevant actions are logged for auditing purposes.

2. Seller Lists Products:
    - **Trigger:** Seller logs into the seller dashboard.
    - **Preconditions:** Seller account is verified and active.
    - **Flow:**
        - Seller navigates to the product listing section.
        - Seller clicks 'Add New Product.'
        - Seller enters product details, including name, description, price, sale price, stock levels, and uploads images.
        - Seller submits the product listing.
        - The system validates the information and adds the product to the database.
        - The product becomes visible to customers on the platform.
    - **Postconditions:** Product is successfully added to the database, and a confirmation is sent to the seller.

3. Customer Completes Checkout:
    - **Trigger:** Customer has items in the cart and decides to proceed to checkout.
    - **Preconditions:** Customer is logged in and has a valid payment method.
    - **Flow:**
        - Customer navigates to the cart page and reviews the items added to the cart.
        - Customer clicks on 'Checkout' to proceed with the purchase.
        - Customer enters shipping information, including address, contact details, and any special instructions.
        - Customer selects a payment method (e.g., credit card via Stripe, PayPal, or COD).
        - Customer reviews the order summary and confirms the order.
        - The system processes the payment through the selected gateway.

        - Upon successful payment, the system updates the order status to 'Confirmed.'
        - Customer receives an order confirmation email with details of the purchase.
    - **Postconditions:** Order is processed, payment is completed, and customer receives confirmation. Order details are updated in the system for the seller to fulfill.

4. Processes Payments:
    - **Trigger:** Customer initiates a payment during checkout.
    - **Preconditions:** Payment gateway integrations are functional, and the customer has sufficient funds.
    - **Flow:**
        - System collects payment details from the customer.
        - System sends the payment details to the selected payment gateway (e.g., Stripe or PayPal).
        - Payment gateway processes the transaction and returns a response to the system.
        - If the payment is successful, the system updates the order status to 'Paid.'
        - If the payment fails, the system notifies the customer and prompts for retry or alternative payment method.
        - System logs the transaction details for record-keeping and auditing.
    - **Postconditions:** Payment is processed, and the order status is updated. Customer and seller are notified of the payment result.

Chapter 4

Design

4.1 Architecture Diagram

The architecture diagram illustrates the main components of the Smart Stock Manager application and their interactions. It provides a high-level view of the system's structure, helping to visualize the organization and flow of data within the application.

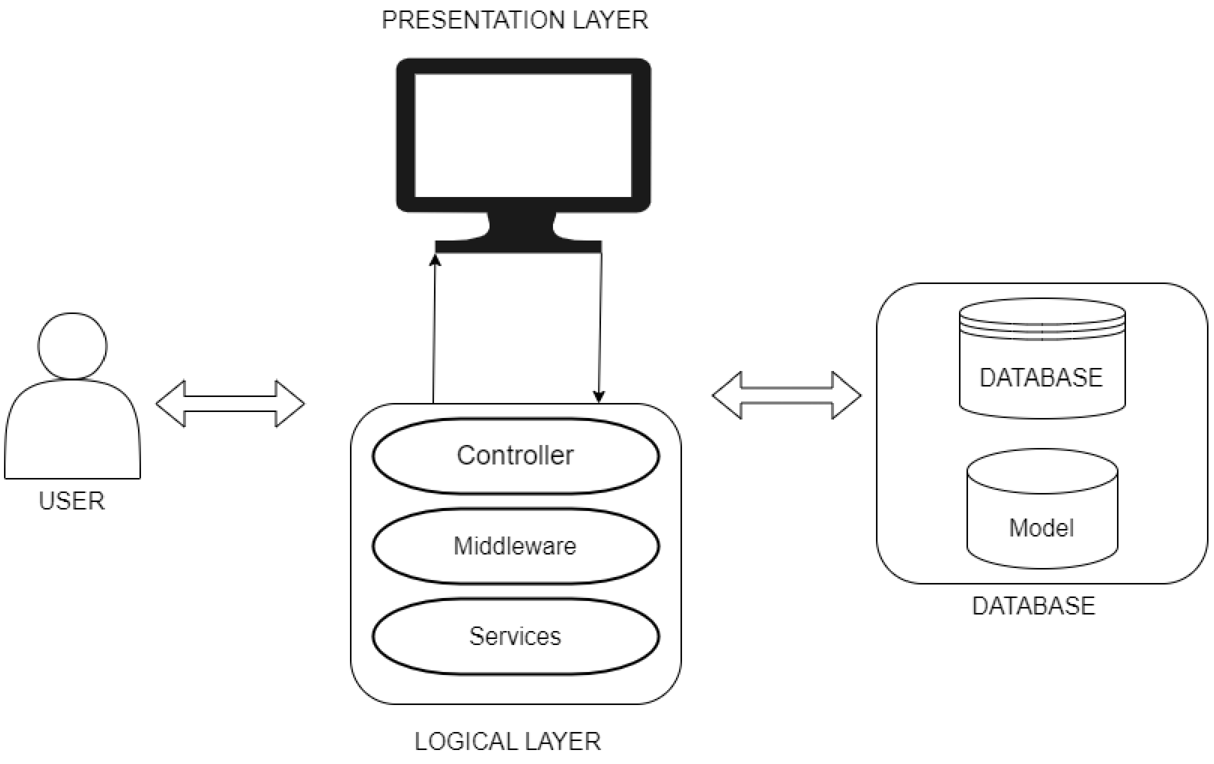


Figure 2: Architecture Diagram

User:

- Represents the end-users interacting with the system through the user interface.

Presentation Layer:

- **Frontend:** Contains the user interface and client-side logic, responsible for presenting data to the user and capturing user inputs.

Business Logic Layer:

- **Controllers:** Handle incoming requests, process them, and return appropriate responses. They act as intermediaries between the user interface and the business logic.
- **Middleware**: Includes various middleware functions such as authentication, error handling, and request validation.

- **Services:** Encapsulate the core business logic and application services, managing business rules and operations.

Data Layer:

- **Models:** Define the data structures and database schemas used by the application. They represent the entities within the system and their relationships.
- **Database:** The actual storage where data is persisted. It includes the database server and data storage mechanisms.

4.2 ERD with Data Dictionary

This section presents the ERD and data dictionary for the MongoDB collections used in the Smart Stock Manager application. The ERD visually represents the entities and their relationships, while the data dictionary provides detailed information about the attributes of each entity.

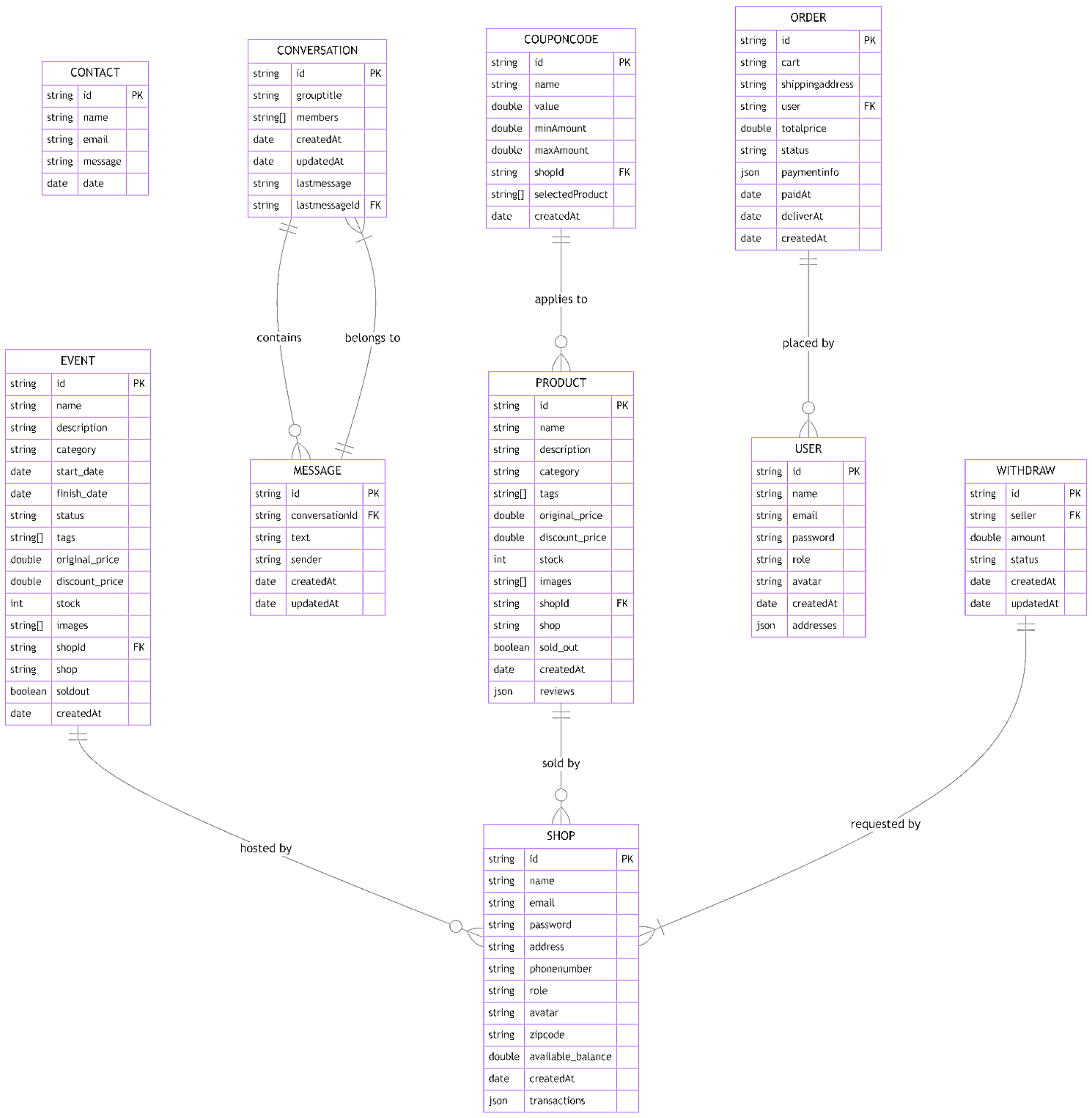


Figure 3: ER Diagram

Data Dictionary

1. **CONTACT**

- **id (string, PK):** Unique identifier for the contact.
- **name (string):** Name of the contact.

- **email (string):** Email address of the contact.
- **message (string):** Message sent by the contact.
- **date (date):** Date when the contact was made.

2. CONVERSATION
   - **id (string, PK):** Unique identifier for the conversation.
   - **grouptitle (string):** Title of the group conversation.
   - **members (string[]):** List of members in the conversation.
   - **createdAt (date):** Date when the conversation was created.
   - **updatedAt (date):** Date when the conversation was last updated.
   - **lastmessage (string):** Content of the last message in the conversation.
   - **lastmessageId (string, FK):** Unique identifier for the last message.
3. EVENT
   - **id (string, PK):** Unique identifier for the event.
   - **name (string):** Name of the event.
   - **description (string):** Description of the event.
   - **category (string):** Category of the event.
   - **start_date (date):** Start date of the event.
   - **finish_date (date**): End date of the event.
   - **status (string):** Status of the event.
   - **tags (string[]):** Tags associated with the event.
   - **original_price (double):** Original price of the event.
   - **discount_price (double):** Discounted price of the event.
   - **stock (int):** Stock quantity available for the event.
   - **images (string[]):** Images associated with the event.
   - **shopId (string, FK):** Unique identifier for the shop hosting the event.
   - **shop (string):** Name of the shop hosting the event.
   - **soldout (boolean):** Indicates if the event is sold out.
   - **createdAt (date**): Date when the event was created.
4. MESSAGE
   - **id (string, PK):** Unique identifier for the message.
   - **conversationId (string, FK):** Unique identifier for the conversation the message belongs to.
   - **text (string):** Content of the message.
   - **sender (string):** Sender of the message.
   - **createdAt (date):** Date when the message was created.

- **updatedAt (date):** Date when the message was last updated.

5. COUPONCODE
   - **id (string, PK):** Unique identifier for the coupon code.
   - **name (string):** Name of the coupon code.
   - **value (double):** Value of the discount.
   - **minAmount (double):** Minimum amount required to apply the coupon.
   - **maxAmount (double):** Maximum discount amount.
   - **shopId (string, FK):** Unique identifier for the shop offering the coupon.
   - **selectedProduct (string[]):** List of products the coupon applies to.
   - **createdAt (date):** Date when the coupon was created.
6. ORDER
   - **id (string, PK):** Unique identifier for the order.
   - **cart (string):** Details of the cart items.
   - **shippingaddress (string):** Shipping address for the order.
   - **user (string, FK):** Unique identifier for the user who placed the order.
   - **totalprice (double):** Total price of the order.
   - **status (string):** Status of the order.
   - **paymentinfo (json):** Payment information for the order.
   - **id (string):** Unique identifier for the payment.
   - **status (string):** Status of the payment.
   - **type (string):** Type of the payment method.
   - **paidAt (date):** Date when the order was paid.
   - **deliverAt (date):** Date when the order was delivered.
   - **createdAt (date):** Date when the order was created.
7. PRODUCT
   - **id (string, PK):** Unique identifier for the product.
   - **name (string):** Name of the product.
   - **description (string):** Description of the product.
   - **category (string):** Category of the product.
   - **tags (string[]):** Tags associated with the product.
   - **original_price (double):** Original price of the product.
   - **discount_price (double):** Discounted price of the product.
   - **stock (int):** Stock quantity available for the product.
   - **images (string[]):** Images associated with the product.
   - **shopId (string, FK):** Unique identifier for the shop selling the product.

- **shop (string):** Name of the shop selling the product.
- **sold_out (boolean):** Indicates if the product is sold out.
- **createdAt (date):** Date when the product was created.
- **reviews (json):** Reviews for the product.

8. SHOP
    - **id (string, PK):** Unique identifier for the shop.
    - **name (string):** Name of the shop.
    - **email (string):** Email address of the shop.
    - **password (string):** Password for the shop's account.
    - **address (string):** Address of the shop.
    - **phonenumber (string):** Phone number of the shop.
    - **role (string):** Role of the shop.
    - **avatar (string):** Avatar image of the shop.
    - **zipcode (string):** Zip code of the shop's location.
    - **available_balance (double):** Available balance of the shop.
    - **createdAt (date):** Date when the shop was created.
    - **transactions (json):** Transactions associated with the shop.
9. USER
    - **id (string, PK):** Unique identifier for the user.
    - **name (string):** Name of the user.
    - **email (string):** Email address of the user.
    - **password (string):** Password for the user's account.
    - **role (string):** Role of the user.
    - **avatar (string):** Avatar image of the user.
    - **createdAt (date):** Date when the user was created.
    - **addresses (json):** Addresses associated with the user.
    - **country (string):** Country of the address.
    - **city (string):** City of the address.
    - **shipping_address (string):** Shipping address of the user.
    - **billing_address (string):** Billing address of the user.
    - **zipCode (string):** Zip code of the address.
    - **addressType (string):** Type of the address (shipping or billing

4.3 Data Flow Diagram (Level 0 and Level 1)

Data Flow Diagrams (DFDs) provide a visual representation of the flow of data within a system. They show how data enters, moves through, and exits the system, highlighting the processes involved. We'll create a Level 0 (Context) and Level 1 DFD for the @Doorz – A Multi-Vendor Website

Level 0 DFD (Context Diagram)

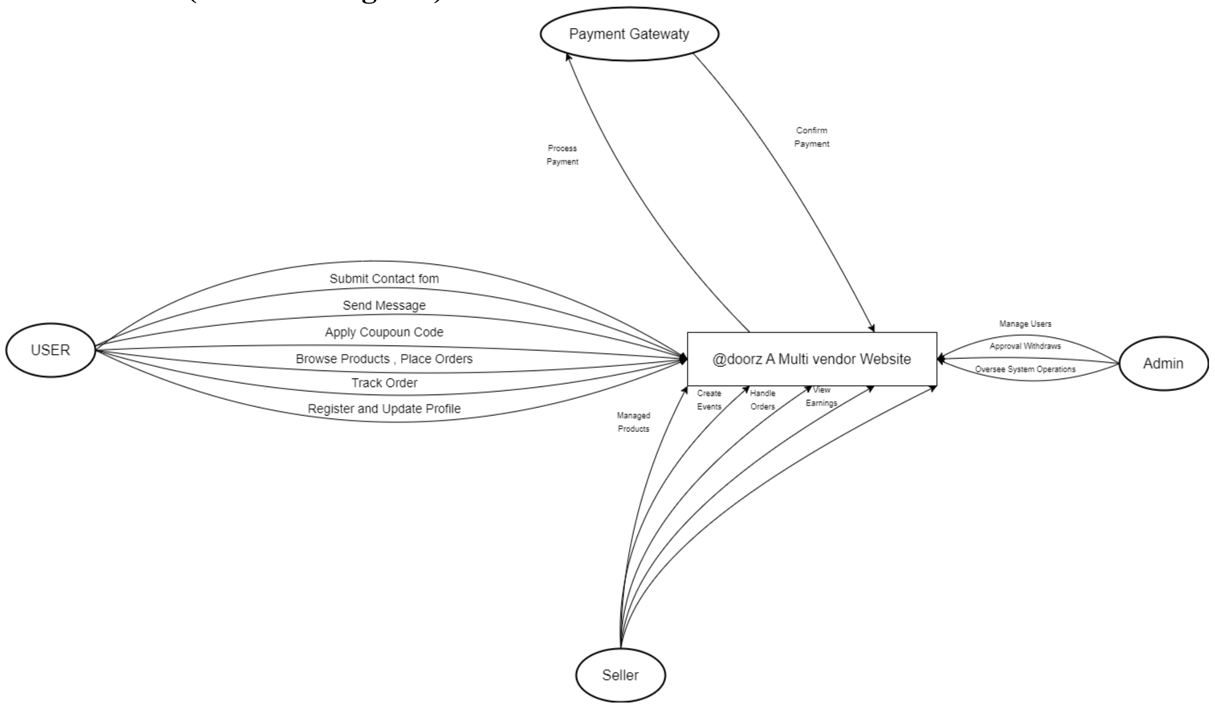


Figure 4: Level 0 DFD

Level 1 DFD

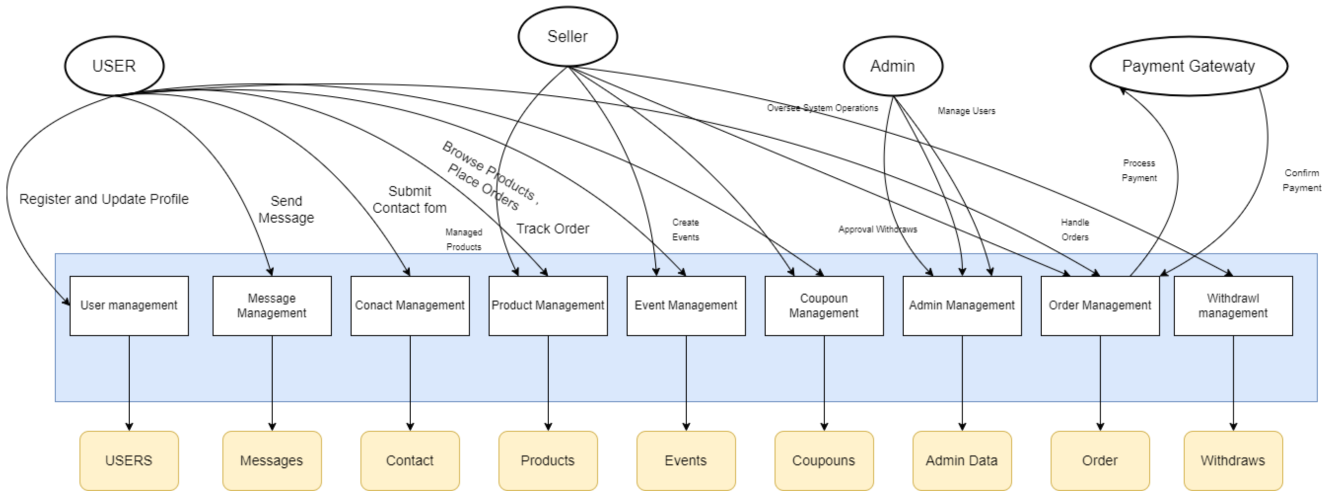


Figure 5: Level 1 DFD

4.4 Class Diagram

The class diagram for this project represents various entities and their relationships within the system. The main classes include Contact, Conversation, Event, Message, CouponCode, Order, Product, Shop, User, and Withdraw. Each class contains attributes relevant to their function, such as id, name, email, createdAt, and status. The diagram also illustrates the relationships between these classes, highlighting interactions such as users sending messages (Contact to User), orders containing products (Order to Product), and shops creating events (Event to Shop). This structured representation ensures a clear understanding of the system's architecture and data flow.

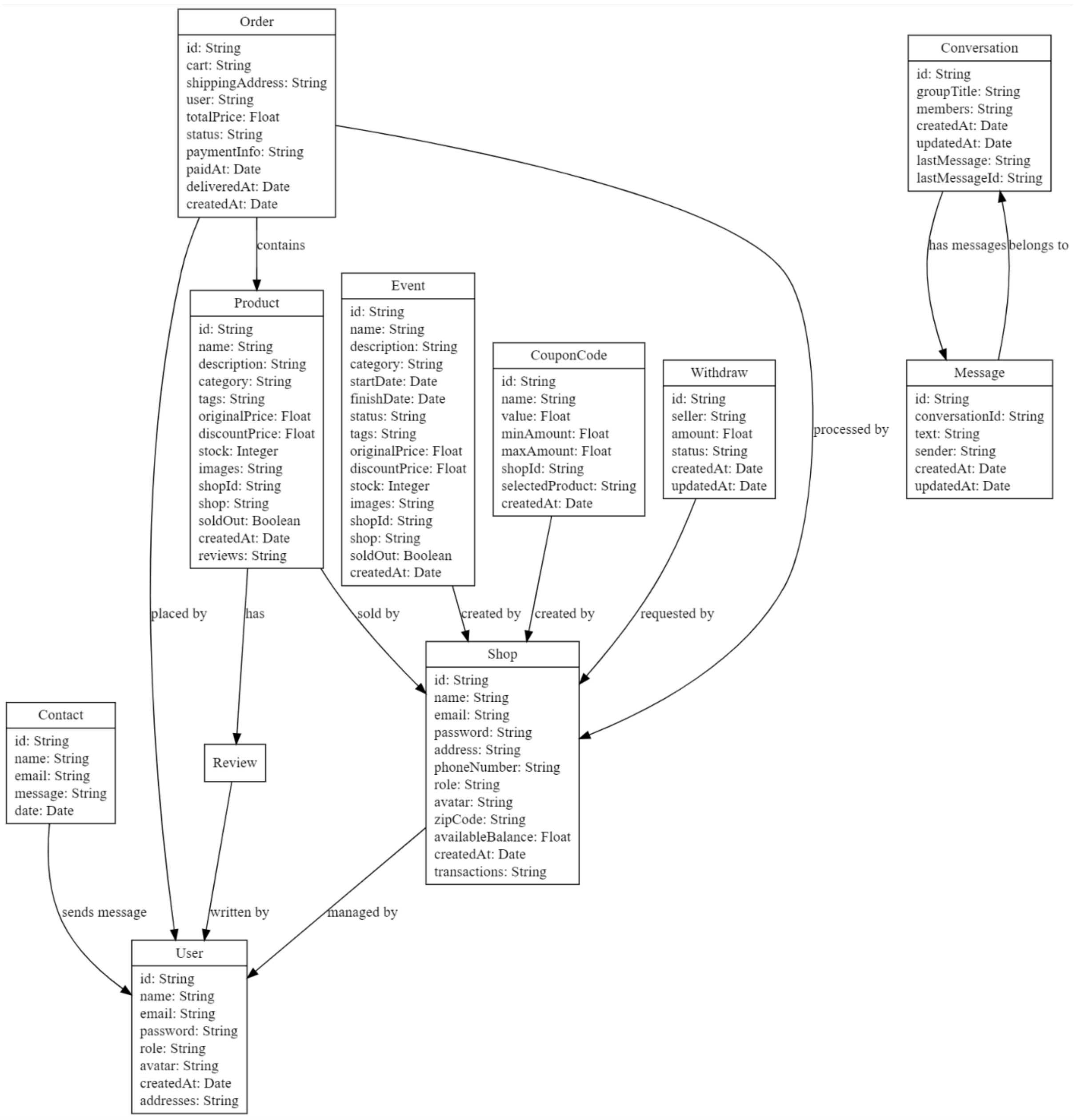


Figure 6: Class Diagram

4.5 Object Diagram

An object diagram shows a snapshot of the system at a particular point in time, highlighting instances of classes and their relationships.

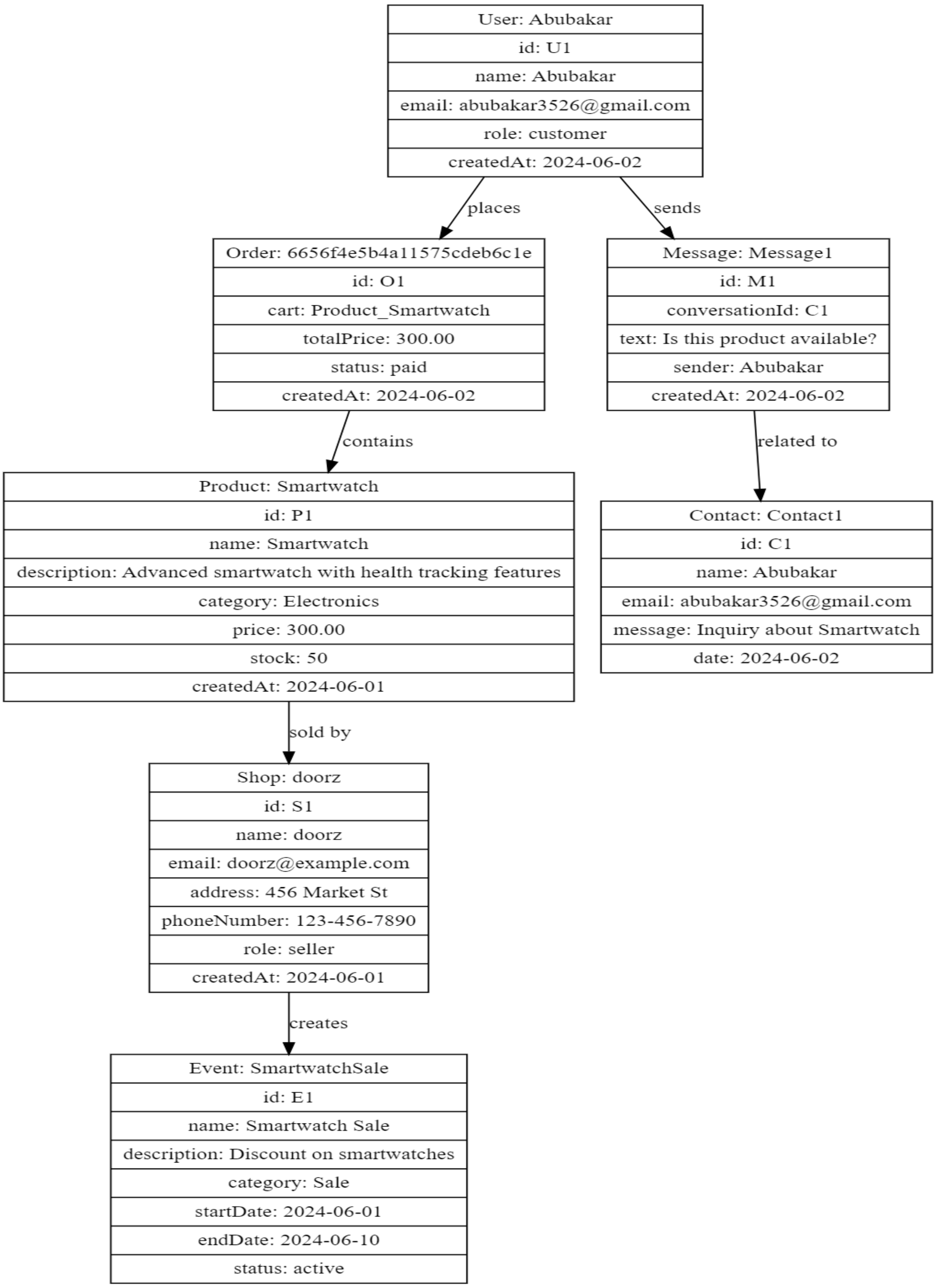
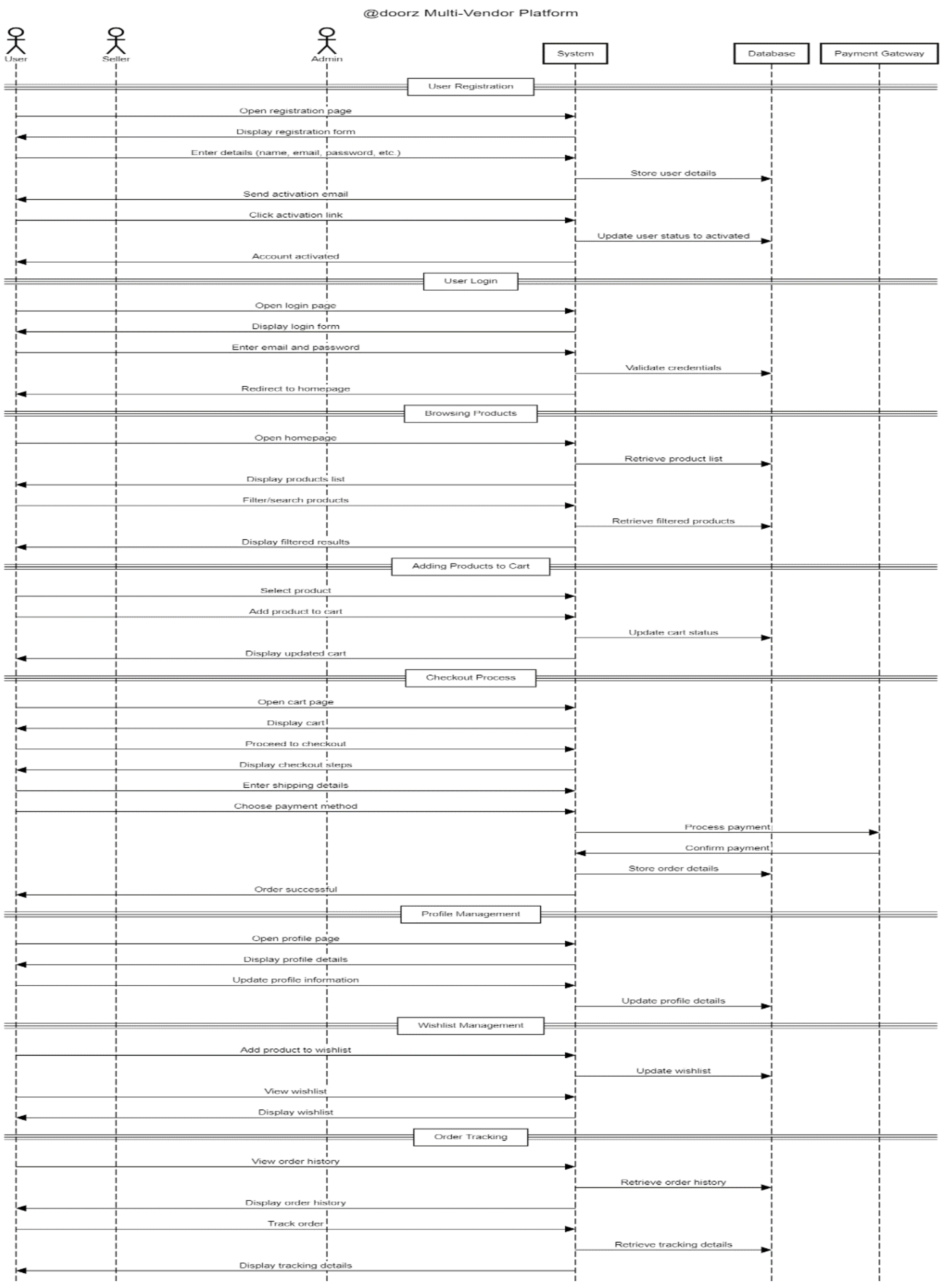


Figure 7: Object Diagram

The object diagram for this project represents specific instances of classes and their relationships at a given point in time. It showcases individual objects such as a particular User, Shop, Product, Order, Event, Message, and Contact. For example, a user object named JohnDoe is linked to an order object Order123 containing a product object Product456, processed by a shop object Shop789. Similarly, an event object Event1 is created by the same shop. This snapshot provides a clear view of how instances interact within the system at a particular moment.

4.6 Sequence Diagram



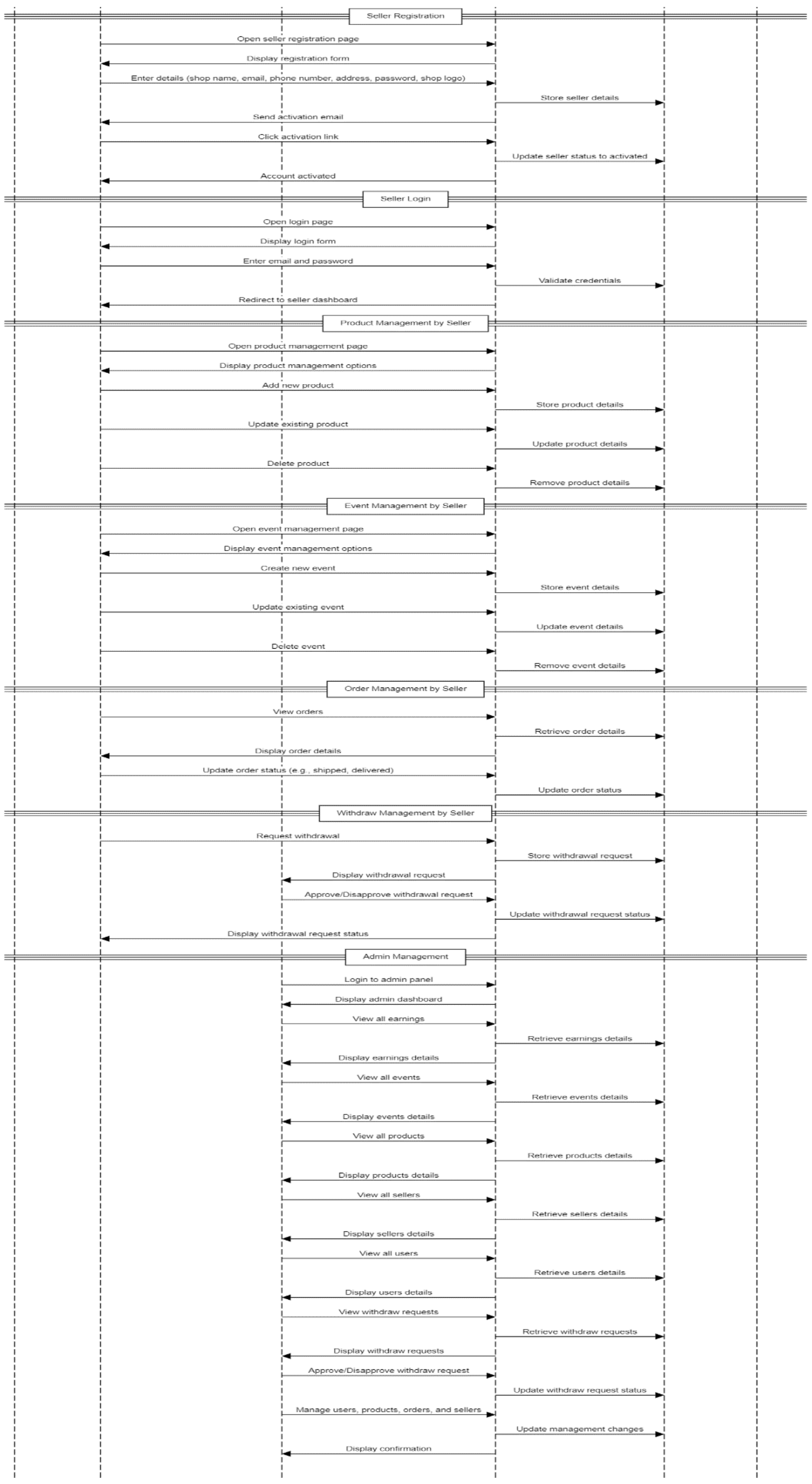


Figure 8: Sequence Diagram

A sequence diagram illustrates how objects interact in a particular scenario of a use case. Below is an outline of the sequence diagram for your project based on the details provided:

1. User Registration

Description: The user opens the registration page, enters their details, and activates their account via an email link.

Actors: User, System, Database

Steps:

- User opens the registration page.
- System displays the registration form.
- User enters details (name, email, password, etc.).
- System stores user details in the database.
- System sends an activation email to the user.
- User clicks the activation link.
- System updates the user status to activated in the database.
- System notifies the user that their account is activated.

2. User Login

Description: The user logs into the system using their credentials.

Actors: User, System, Database

Steps:

- User opens the login page.
- System displays the login form.
- User enters email and password.
- System validates credentials with the database.
- System redirects the user to the homepage.

3. Browsing Products

Description: The user browses and searches for products on the platform.

Actors: User, System, Database

Steps:

- User opens the homepage.
- System retrieves the product list from the database.
- System displays the product list.
- User filters or searches for products.

- System retrieves filtered products from the database.
- System displays the filtered results.

4. Adding Products to Cart

Description: The user adds selected products to their shopping cart.

Actors: User, System, Database

Steps:

- User selects a product.
- User adds the product to the cart.
- System updates the cart status in the database.
- System displays the updated cart to the user.

5. Checkout Process

Description: The user proceeds through the checkout process to complete their purchase.

Actors: User, System, Payment Gateway, Database

Steps:

- User opens the cart page.
- System displays the cart contents.
- User proceeds to checkout.
- System displays checkout steps.
- User enters shipping details.
- User chooses a payment method.
- System processes the payment through the payment gateway.
- Payment gateway confirms the payment to the system.
- System stores the order details in the database.
- System notifies the user of successful order placement.

6. Profile Management

Description: The user manages their profile information.

Actors: User, System, Database

Steps:

- User opens the profile page.
- System displays profile details.
- User updates profile information.
- System updates profile details in the database.

7. Wishlist Management

Description: The user manages their wishlist of products.

Actors: User, System, Database

Steps:

- User adds a product to the wishlist.
- System updates the wishlist in the database.
- User views the wishlist.
- System displays the wishlist to the user.

8. Order Tracking

Description: The user tracks their past and current orders.

Actors: User, System, Database

Steps:

- User views order history.
- System retrieves order history from the database.
- System displays the order history to the user.
- User tracks a specific order.
- System retrieves tracking details from the database.
- System displays tracking details to the user.

9. Seller Registration

Description: The seller registers on the platform and activates their account.

Actors: Seller, System, Database

Steps:

- Seller opens the seller registration page.
- System displays the registration form.
- Seller enters details (shop name, email, phone number, address, password, shop logo).
- System stores seller details in the database.
- System sends an activation email to the seller.
- Seller clicks the activation link.
- System updates the seller status to activated in the database.
- System notifies the seller that their account is activated.

10. Seller Login

Description: The seller logs into the system using their credentials.

Actors: Seller, System, Database

Steps:

- Seller opens the login page.
- System displays the login form.
- Seller enters email and password.
- System validates credentials with the database.
- System redirects the seller to the seller dashboard.

11. Product Management by Seller

Description: The seller manages their products on the platform.

Actors: Seller, System, Database

Steps:

- Seller opens the product management page.
- System displays product management options.
- Seller adds a new product.
- System stores product details in the database.
- Seller updates an existing product.
- System updates product details in the database.
- Seller deletes a product.
- System removes product details from the database.

12. Event Management by Seller

Description: The seller manages events associated with their products.

Actors: Seller, System, Database

Steps:

- Seller opens the event management page.
- System displays event management options.
- Seller creates a new event.
- System stores event details in the database.
- Seller updates an existing event.
- System updates event details in the database.
- Seller deletes an event.

- System removes event details from the database.

13. Order Management by Seller

Description: The seller manages orders placed for their products.

Actors: Seller, System, Database

Steps:

- Seller views orders.
- System retrieves order details from the database.
- System displays order details to the seller.
- Seller updates order status (e.g., shipped, delivered).
- System updates order status in the database.

14. Withdraw Management by Seller

Description: The seller requests withdrawals and manages withdrawal status.

Actors: Seller, System, Database, Admin

Steps:

- Seller requests a withdrawal.
- System stores the withdrawal request in the database.
- System displays the withdrawal request to the admin.
- Admin approves/disapproves the withdrawal request.
- System updates the withdrawal request status in the database.
- System notifies the seller of the withdrawal request status.

15. Admin Management

Description: The admin manages the platform, including earnings, events, products, sellers, users, and withdrawal requests.

Actors: Admin, System, Database

Steps:

- Admin logs into the admin panel.
- System displays the admin dashboard.

View Earnings:

- Admin views all earnings.
- System retrieves earnings details from the database.
- System displays earnings details to the admin.

View Events:

- Admin views all events.
- System retrieves events details from the database.
- System displays events details to the admin.

View Products:

- Admin views all products.
- System retrieves products details from the database.
- System displays products details to the admin.

View Sellers:

- Admin views all sellers.
- System retrieves seller's details from the database.
- System displays seller's details to the admin.

View Users:

- Admin views all users.
- System retrieves user's details from the database.
- System displays user's details to the admin.

Manage Withdrawal Requests:

- Admin views withdrawal requests.
- System retrieves withdrawal requests from the database.
- System displays withdrawal requests to the admin.
- Admin approves/disapproves the withdrawal request.
- System updates the withdrawal request status in the database.

Manage Platform:

- Admin manages users, products, orders, and sellers.
- System updates the management changes in the database.
- System displays confirmation to the admin.

4.7 Activity Diagram

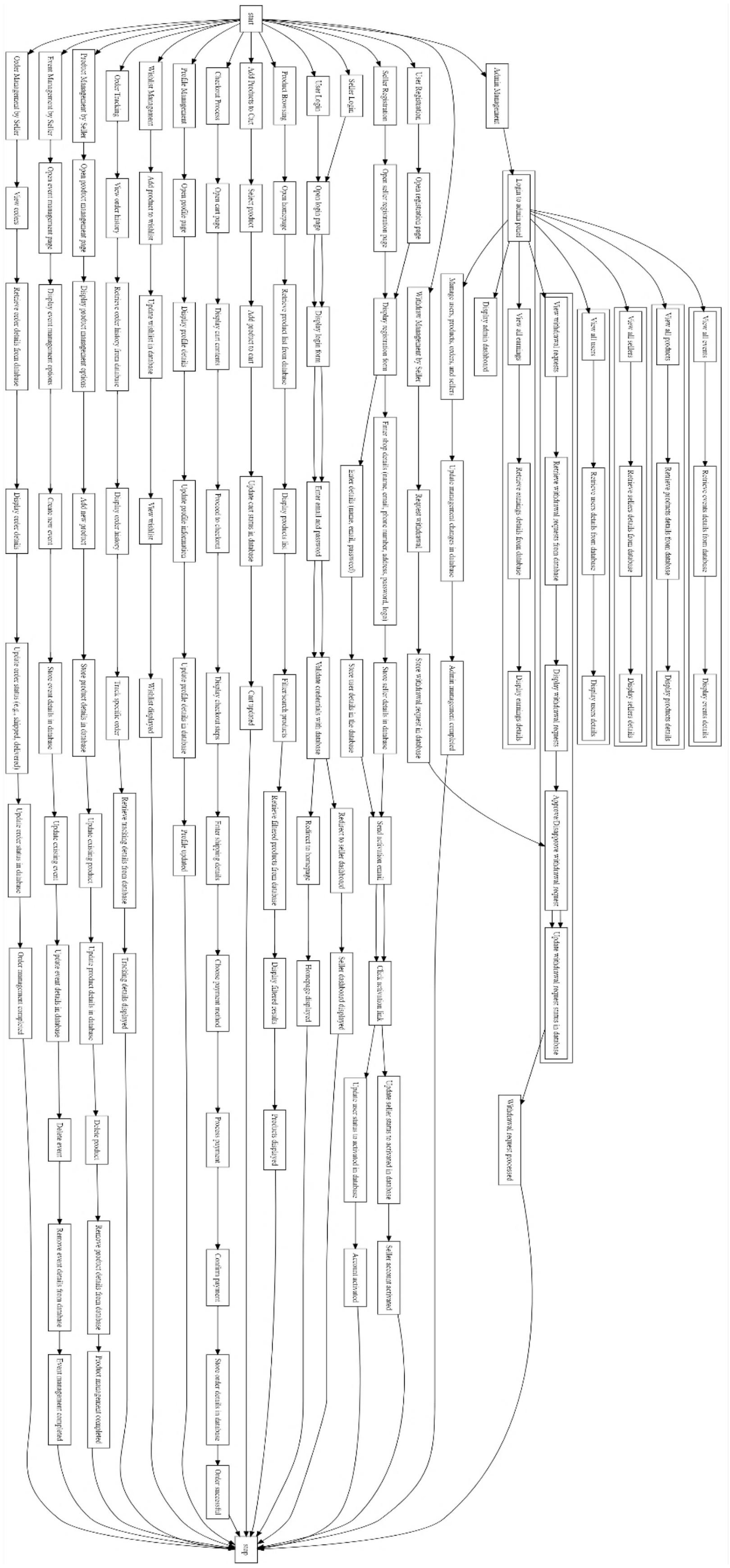

start
Order Management by Seller
View orders
Retrieve order details from database
Display order details
Update order status (e.g., shipped, delivered)
Update order status in database
Order management completed
Event Management by Seller
Open event management page
Display event management options
Create new event
Store event details in database
Update existing event
Update event details in database
Delete event
Remove event details from database
Event management completed
Product Management by Seller
Open product management page
Display product management options
Add new product
Store product details in database
Update existing product
Update product details in database
Delete product
Retrieve product details from database
Product management completed
Order Tracking
View order history
Retrieve order history from database
Display order history
Track specific order
Retrieve tracking details from database
Tracking details displayed
Wishlist Management
Add product to wishlist
Update wishlist in database
View wishlist
Wishlist displayed
Profile Management
Open profile page
Display profile details
Update profile information
Update profile details in database
Profile updated
Checkout Process
Open cart page
Display cart contents
Proceed to checkout
Display checkout steps
Enter shipping details
Choose payment method
Process payment
Confirm payment
Store order details in database
Order successful
Add Products to Cart
Select product
Add product to cart
Update cart items in database
Cart updated
Product Browsing
Open homepage
Retrieve product list from database
Display products list
Filter/search products
Retrieve filtered products from database
Display filtered results
Products displayed
User Login
Seller Login
Open login page
Display login form
Enter email and password
Validate credentials with database
Redirect to homepage
Homepage displayed
Redirect to seller dashboard
Seller dashboard displayed
Seller Registration
Open seller registration page
Display registration form
Enter shop details (name, email, phone number, address, password, logo)
Store seller details in database
Send activation email
Click activation link
Update seller status to activated in database
Seller account activated
User Registration
Open registration page
Enter user details (name, email, password)
Store user details in the database
Update user status to activated in database
Account activated
Withdrawal Management by Seller
Request withdrawal
Store withdrawal request in database
Admin Management
Login to admin panel
Manage users, products, orders, and sellers
Update management changes in database
Admin management completed
Display admin dashboard
View all earnings
Retrieve earnings details from database
Display earnings details
View withdrawal requests
Retrieve withdrawal requests from database
Display withdrawal requests
Approve/Disapprove withdrawal request
Update withdrawal request status in database
Withdrawal request processed
View all users
Retrieve users details from database
Display users details
View all sellers
Retrieve sellers details from database
Display sellers details
View all products
Retrieve products details from database
Display products details
View all events
Retrieve events details from database
Display events details
stop

Figure 9: Activity Diagram

The activity diagram will cover the main processes including user registration, user login, product browsing, adding products to cart, checkout process, order management by sellers, and admin management.

Explanation:

- **User Registration:** Flow from opening the registration page to account activation.
- **User Login:** Steps from opening the login page to accessing the homepage.
- **Product Browsing:** Steps involved in browsing and filtering products.
- **Adding Products to Cart:** Sequence of selecting and adding products to the cart.
- **Checkout Process:** Detailed steps from viewing the cart to order completion.
- **Profile Management:** Steps for managing and updating user profile.
- **Wishlist Management:** Process of adding products to the wishlist and viewing it.
- **Order Tracking:** Steps to view and track order history.
- **Seller Registration**: Flow from seller registration to account activation.
- **Seller Login:** Steps from seller login to accessing the seller dashboard.
- **Product Management by Seller:** Sequence of managing products by the seller.
- **Event Management by Seller:** Steps for creating, updating, and deleting events.
- **Order Management by Seller**: Steps for managing orders by the seller.
- **Withdraw Management by Seller:** Process of requesting and processing withdrawals.
- **Admin Management:** Comprehensive steps for admin management including viewing earnings, events, products, sellers, users, and handling withdrawal requests.

4.8 Collaboration Diagram

A collaboration diagram, also known as a communication diagram, shows the interactions between objects or parts of the system. Here, I'll create a collaboration diagram based on the scenarios for users, sellers, and admins.

Explanation:

- **User Interactions:** Shows how the user interacts with the system for registration, login, browsing products, adding to cart, checkout, managing profile, viewing wishlist, and tracking orders.
- **Seller Interactions:** Illustrates the seller's interactions with the system for registration, login, managing products, managing events, viewing orders, and requesting withdrawals.

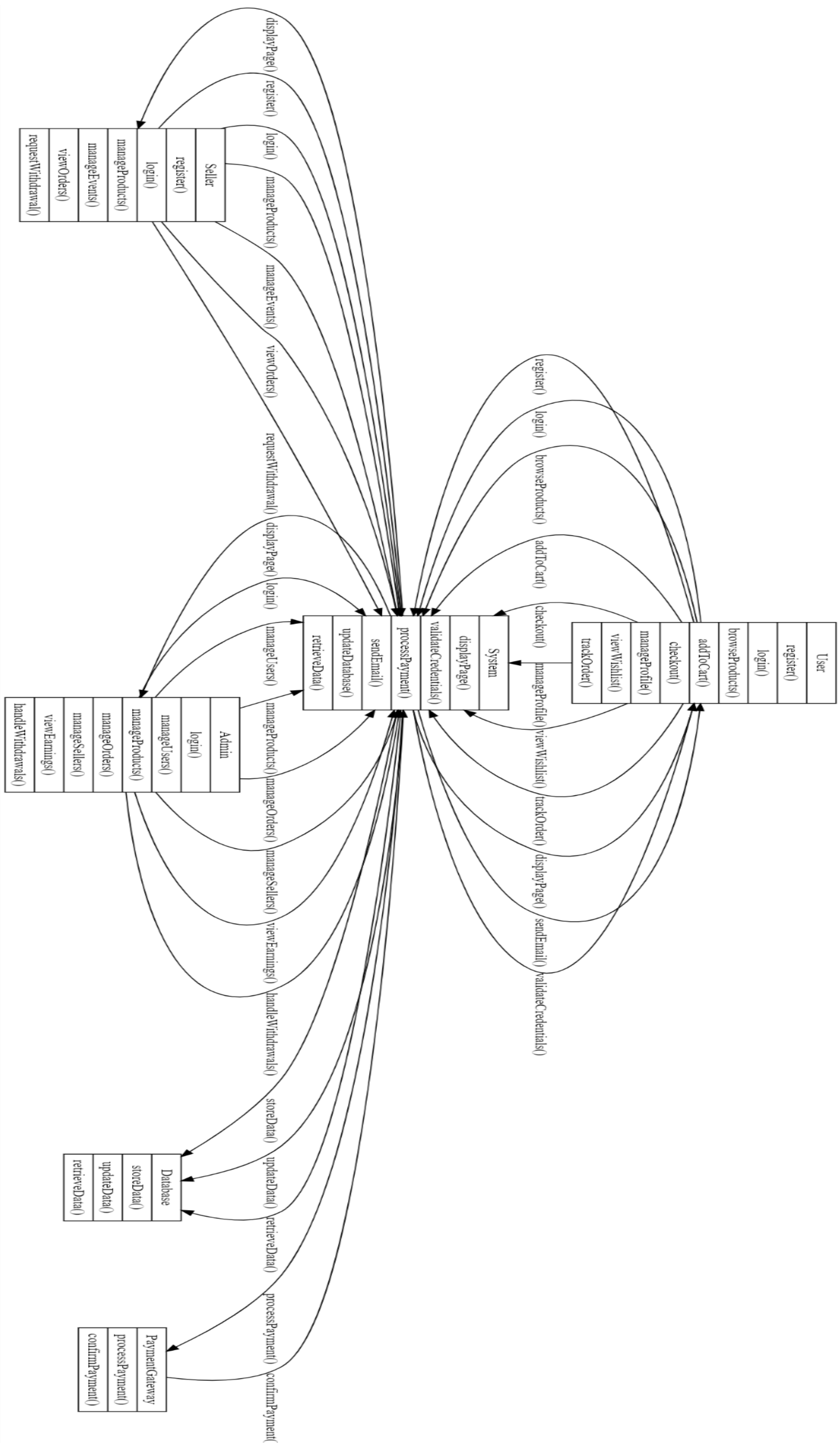


Figure 10: Collaboration Diagram

- **Admin Interactions:** Details the admin's interactions with the system for login, managing users, products, orders, and sellers, viewing earnings, and handling withdrawals.
- **System Interactions with Database**: Depicts the system's operations with the database, such as storing, updating, and retrieving data.
- **System Interactions with Payment Gateway**: Shows how the system processes and confirms payments through the payment gateway.
- **System Interactions with User, Seller, and Admin:** Displays pages, validates credentials, and sends emails to the user, seller, and admin.

4.9 State Transition Diagram

A state transition diagram illustrates the states an object can occupy and the transitions between those states.

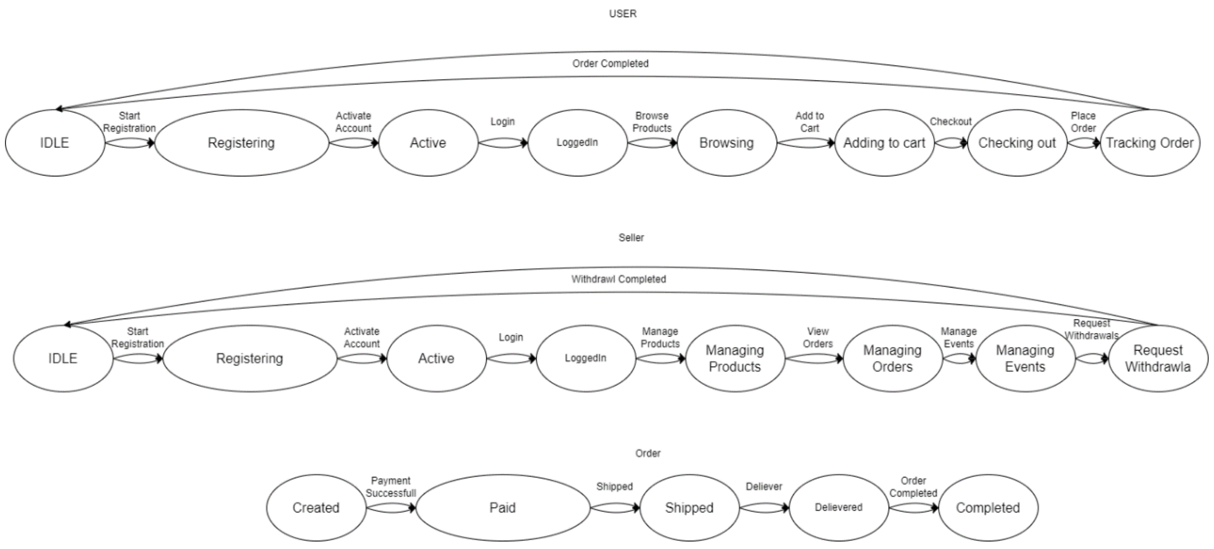


Figure 11: State Transition Diagram

Explanation:

User States:

- Idle: Initial state when the user is not interacting with the system.
- Registering: State when the user is in the process of registering.
- Active: State after the user has activated their account.
- Logged In: State when the user is logged into the system.
- Browsing: State when the user is browsing products.
- Adding to Cart: State when the user is adding products to the cart.

- Checking Out: State when the user is checking out.
- Tracking Order: State when the user is tracking their order.

Seller States:

- Idle: Initial state when the seller is not interacting with the system.
- Registering: State when the seller is in the process of registering.
- Active: State after the seller has activated their account.
- Logged In: State when the seller is logged into the system.
- Managing Products: State when the seller is managing their products.
- Managing Orders: State when the seller is managing their orders.
- Managing Events: State when the seller is managing their events.
- Requesting Withdrawal: State when the seller is requesting a withdrawal.

Order States:

- Created: Initial state when an order is created.
- Paid: State when the order payment is successful.
- Shipped: State when the order is shipped.
- Delivered: State when the order is delivered to the customer.
- Completed: Final state when the order is completed.

Chapter 5

Implementation

The implementation of the multi-vendor e-commerce platform, @doorz, is structured into three main layers following a 3-tier architecture: Presentation Layer, Logic Layer, and Data Layer. This structured approach ensures scalability, maintainability, and efficient separation of concerns.

5.1 Component Diagram

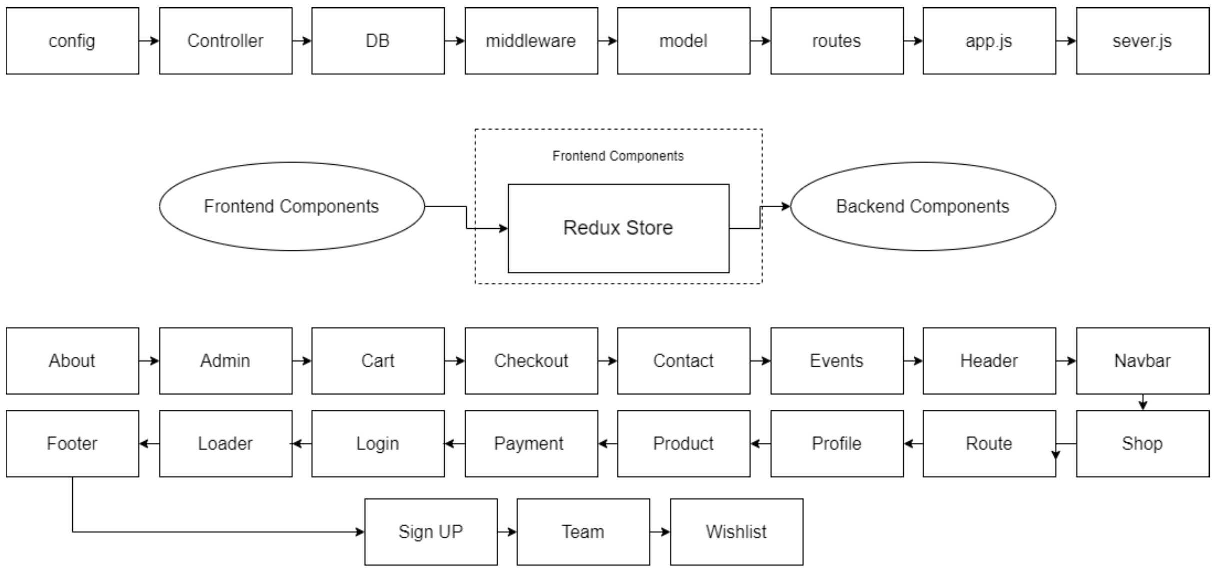


Figure 12: Component Diagram

The component diagram will now include the socket server to handle real-time events.

Backend Components:

- **config**: Manages environment variables.
- **controller:** Handles business logic.
- **db:** Manages database connections.
- **middleware:** Includes authentication and error handling.
- **model:** Contains Mongoose schemas.
- **routes:** Defines API endpoints.
- **app.js:** Initializes and configures the application.
- **server.js**: Starts the server.
- **socketServer.js:** Manages socket connections for real-time communication.

Frontend Components:

src/components:

About: Contains components for the about page.

- **Admin**: Includes components for admin dashboard and functionalities.
- **Cart:** Manages the shopping cart.
- **Checkout:** Handles multi-step checkout process.
- **Contact:** Contains the contact form.
- **Events:** Displays events managed by sellers.
- **Header, Navbar, Footer, Loader:** Common UI components.
- **Login:** Manages user login.
- **Payment:** Handles payment processing.
- **Product:** Displays product details, ratings, etc.
- **Profile:** Manages user profiles.
- **Route:** Manages routing components like best deals, categories, etc.
- **Shop:** Manages seller dashboard functionalities.
- **Signup:** Manages user signup.
- **Team:** Displays team information.
- **Wishlist:** Manages customer wishlist.

Redux Components:

- **actions:** Defines actions for different modules like cart, events, orders, etc.
- **reducers:** Defines reducers for different modules.
- **store.js:** Configures the Redux store.

5.2 Deployment Diagram

The deployment diagram shows the physical arrangement of hardware and software in the system. It includes the client node, server node, socket server node, and database node.

- **Client Node**: Runs the frontend application (React.js), providing a dynamic user interface for customers and vendors.
- **Server Node:** Hosts the backend application (Node.js, Express.js), handling business logic and API requests.
- **Socket Server Node**: Manages real-time events and communication using WebSockets, enhancing features like live chat and notifications.

- **Database Node:** Runs the MongoDB database, storing all collections and data.

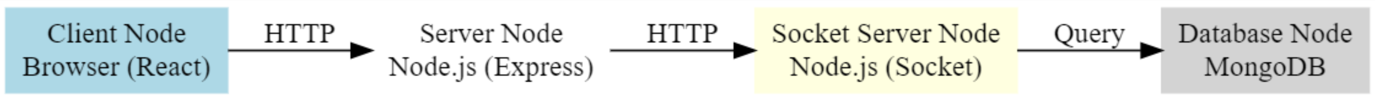


Figure 13: Deployment Diagram

5.3 Database Architecture (1-Tier, 2-Tier, 3-Tier Architecture)

The database architecture used for @doorz is a 3-Tier architecture, which provides a clear separation between the presentation, logic, and data layers.

1-Tier Architecture: Direct client-database interaction, suitable for small applications.

2-Tier Architecture: Client interacts with the database through an intermediate server, suitable for medium-sized applications.

3-Tier Architecture: Separation into three layers: Presentation Layer (UI), Logic Layer (Server), and Data Layer (Database), suitable for large, scalable applications.

For @doorz, the 3-Tier architecture is implemented as follows:

- **Presentation Layer:** The frontend application is developed using React.js, providing a dynamic and responsive user interface for customers and vendors.
- **Logic Layer:** The backend application is built using Node.js and Express.js, managing the business logic, API requests, and socket connections for real-time communication.
- **Data Layer**: The MongoDB database stores all collections and data, ensuring efficient data management and retrieval.

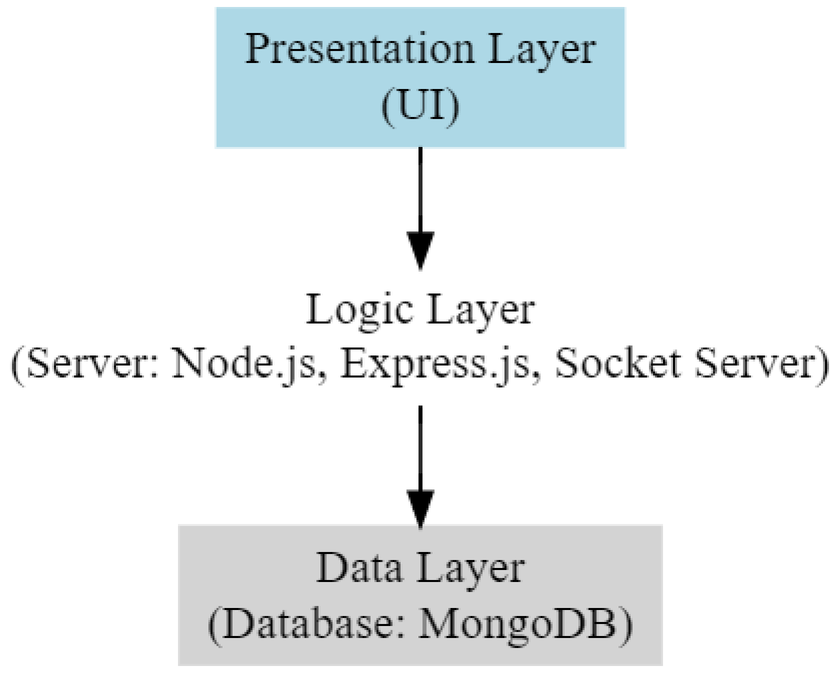


Figure 14: 3-Layer Architecture

This architecture ensures that the application is scalable, maintainable, and efficient, with a clear separation of concerns between the different layers.

Database Collections

The MongoDB database for @doorz includes the following collections with respective fields:

- **contacts:** { id, name, email, message, date }
- **conversations:** { id, groupTitle, members, createdAt, updatedAt, lastMessage, lastMessageId }
- **events:** { id, name, description, category, startDate, finishDate, status, tags, originalPrice, discountPrice, stock, images, shopId, shop, soldOut, createdAt }
- **messages:** { id, conversationId, text, sender, createdAt, updatedAt }
- **couponCodes:** { id, name, value, minAmount, maxAmount, shopId, selectedProduct, createdAt }
- **orders:** { id, cart, shippingAddress, user, totalPrice, status, paymentInfo: { id, status, type }, paidAt, deliveredAt, createdAt }
- **products:** { id, name, description, category, tags, originalPrice, discountPrice, stock, images, shopId, shop, soldOut, createdAt, reviews }
- **shops:** { id, name, email, password, address, phoneNumber, role, avatar, zipCode, availableBalance, createdAt, transactions }
- **users:** { id, name, email, password, role, avatar, createdAt, addresses: { country, city, shippingAddress, billingAddress, zipCode, addressType } }
- **withdrawals:** { id, seller, amount, status, createdAt, updatedAt }

Chapter 6

Testing (Software Quality Attributes)

Testing is a critical phase in the software development lifecycle, ensuring that the application meets its requirements and functions as expected. This chapter outlines the testing strategies and techniques used to validate the @doorz multi-vendor e-commerce platform.

6.1 Test Case Specification

Test cases are designed to cover all aspects of the application's functionality. Each test case includes a description, preconditions, steps to execute, expected results, and postconditions. Below is a template for the test case specification.

Test Case Template:

- **Test Case ID:** Unique identifier for the test case.
- **Test Case Description:** Brief description of the test case.
- **Preconditions:** Any prerequisites that must be met before executing the test.
- **Test Steps:** Detailed steps to execute the test.
- **Expected Result:** The expected outcome of the test.
- **Actual Result:** The actual outcome of the test (filled in after execution).
- **Status:** Pass or Fail (filled in after execution).
- **Comments:** Any additional notes or observations.

Example Test Case:

- **Test Case ID: TC-001**
- **Test Case Description:** Verify user login functionality.
- **Preconditions:** User account must exist.
- **Test Steps:**
    1. Navigate to the login page.
    2. Enter valid username and password.
    3. Click the login button.
- **Expected Result:** User should be redirected to the dashboard.
- **Actual Result:** (To be filled after test execution)
- **Status:** (To be filled after test execution)
- **Comments:** (To be filled after test execution)

6.2 Black Box Test Cases

Black box testing involves testing the application's functionality without knowledge of the internal code structure. It focuses on input/output and user interactions.

6.2.1 Boundary Value Analysis (BVA)

BVA tests the boundaries of input values to ensure the application handles edge cases correctly.

Example: User Registration Password Length

- Test Case ID: BVA-001
- Test Case Description: Verify password length boundary values.
- Preconditions: Registration form is accessible.
- Test Steps:
    1. Enter a password with minimum length (e.g.,8 characters).
    2. Enter a password with maximum length (e.g., 20 characters).
    3. Enter a password with one character less than minimum length (e.g., 5 characters).
    4. Enter a password with one character more than maximum length (e.g., 21 characters).
- Expected Result:
- Step 1 and 2: Registration should succeed.
- Step 3 and 4: Registration should fail with an appropriate error message.

6.2.2 Equivalence Class Partitioning

Equivalence class partitioning divides input data into valid and invalid classes, ensuring that each class is tested at least once.

Example: Email Validation

- Test Case ID: ECP-001
- Test Case Description: Verify email field with valid and invalid inputs.
- Preconditions: Registration form is accessible.
- Test Steps:
    1. Enter a valid email (e.g., user@example.com).
    2. Enter an invalid email (e.g., user@example).
- Expected Result:
    - ➢ Step 1: Registration should succeed.

➢ Step 2: Registration should fail with an appropriate error message.

6.2.3 State Transition Testing

State transition testing checks the application's behavior when transitioning between different states.

Example: Order Status Transition

- Test Case ID: STT-001
- Test Case Description: Verify order status transitions.
- Preconditions: Order is placed and accessible in the system.
- Test Steps:
    1. Change order status from "Pending" to "Processing".
    2. Change order status from "Processing" to "Shipped".
    3. Change order status from "Shipped" to "Delivered".
- Expected Result: Order status should update correctly at each step.

6.2.4 Decision Table Testing

Decision table testing uses a tabular representation of inputs and corresponding actions to ensure all combinations are tested.

Example: Payment Method Selection

Table 1: Payment Method Selection

Condition	Card Payment	PayPal	COD	Expected Result
Valid	Yes	No	No	Process Card Payment
Valid	No	Yes	No	Process PayPal Payment
Valid	No	No	Yes	Process COD
Invalid	Yes	Yes	Yes	Show Error

6.2.5 Graph Based Testing

Graph-based testing uses graph theory to model the flow of the application and ensure all paths are tested.

Example: User Navigation

- Test Case ID: GBT-001
- Test Case Description: Verify user can navigate from homepage to product detail page and back.
- Preconditions: User is on the homepage.
- Test Steps:

1. Click on a product link.
2. Verify product detail page loads.
3. Click the back button.

- Expected Result: User should return to the homepage.

6.3 White Box Testing

White box testing involves testing the internal structures or workings of the application, ensuring code quality and coverage.

6.3.1 Statement Coverage

Statement coverage ensures that every statement in the code is executed at least once.

Example: User Login Function

- Test Case ID: SC-001
- Test Case Description: Verify all statements in the login function are executed.
- Preconditions: User account must exist.
- Test Steps:
    1. Enter valid credentials.
    2. Enter invalid credentials.
- Expected Result: All code paths should be executed at least once.

6.3.2 Branch Coverage

Branch coverage ensures that every branch (decision point) in the code is executed.

Example: Checkout Process

- Test Case ID: BC-001
- Test Case Description: Verify all branches in the checkout function are executed.
- Preconditions: Items are in the cart.
- Test Steps:
    1. Proceed with valid payment information.
    2. Proceed with invalid payment information.
- Expected Result: Both branches of the payment validation should be executed.

6.3.3 Path Coverage

Path coverage ensures that all possible paths in the code are executed at least once.

Example: Order Fulfillment

- Test Case ID: PC-001
- Test Case Description: Verify all paths in the order fulfillment function are executed.

- Preconditions: Order is placed.
- Test Steps:
    1. Process order with in-stock items.
    2. Process order with out-of-stock items.
- Expected Result: All code paths should be executed, including handling both in-stock and out-of-stock scenarios.

Chapter 7

Tools and Technologies

This chapter outlines the tools and technologies used in the development of the @doorz multi-vendor e-commerce platform, covering the programming languages and the operating environment.

7.1 Programming Languages

The @doorz platform leverages several programming languages to build a robust, scalable, and efficient application. Each language is chosen for its strengths and suitability to specific tasks within the project.

JavaScript

Purpose: JavaScript is the primary language used for both frontend and backend development. Its versatility allows for the creation of dynamic and interactive user interfaces as well as powerful server-side logic.

Usage:

- Frontend: React.js framework for building user interfaces.
- Backend: Node.js runtime environment for server-side scripting.
- State Management: Redux library for managing application state.

HTML & CSS

Purpose: HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) are fundamental technologies for building and styling web pages.

Usage:

- HTML: Structures the content of web pages.
- CSS: Styles the appearance of web pages, ensuring a consistent and visually appealing design.

Mongoose (for MongoDB)

Purpose: Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It provides a straightforward, schema-based solution to model application data.

Usage:

- Database Interaction: Defines schemas and models for MongoDB collections, ensuring data consistency and simplifying CRUD operations.

7.2 Operating Environment

The operating environment encompasses the software and hardware systems required to develop, deploy, and run the @doorz platform. This section details the various components of the operating environment.

Development Environment

- **Operating System:** The development can be done on various operating systems including Windows, macOS, and Linux, thanks to the cross-platform capabilities of the tools used.
- **Code Editor/IDE:** Visual Studio Code (VS Code) is recommended for its powerful extensions, debugging capabilities, and integrated terminal.
- **Version Control:** Git is used for version control, with GitHub serving as the repository hosting service to manage source code and track changes.

Frontend Environment

- **React.js:** A JavaScript library for building user interfaces. React is chosen for its component-based architecture, which promotes reusability and efficient rendering.
- **Redux:** A state management library used in conjunction with React to manage application state in a predictable manner.
- **Node Package Manager (npm):** Used to manage project dependencies and scripts for building, testing, and deploying the application.

Backend Environment

- **Node.js:** A JavaScript runtime built on Chrome's V8 JavaScript engine, used for server-side scripting and to build scalable network applications.
- **Express.js**: A minimal and flexible Node.js web application framework that provides robust features for building web and mobile applications.
- **Socket.io:** A library for real-time web applications, enabling bi-directional communication between web clients and servers.
- **Mongoose:** An ODM library for MongoDB, used to define schemas and models for data interactions.

Database Environment

MongoDB: A NoSQL database program that uses JSON-like documents with schema. MongoDB is chosen for its flexibility, scalability, and ease of integration with Node.js applications.

- **MongoDB Atlas:** A fully-managed cloud database service for MongoDB, providing automated backups, scaling, and security features.

Communication and Collaboration Tools

Google Meet: Used for team communication and collaboration.

Appendix A: User Documentation

User documentation provides comprehensive instructions on how to use the @doorz platform. It is designed to help administrators, vendors, and customers understand the functionalities and features available to them.

Administrator Guide

Login and Dashboard Access

- Login: Navigate to the admin login page, enter your credentials, and click on the login button.
- Dashboard: Once logged in, you will be directed to the admin dashboard where you can view earnings, events, orders, and user details.

Managing Users and Vendors

- View Users: Access the "All Users" section to see a list of all registered users.
- View Vendors: Access the "All Sellers" section to see a list of all registered vendors.
- Approve/Disapprove Vendors: Review vendor applications and approve or disapprove them based on compliance with platform policies.

Managing Products and Events

- View Products: Access the "All Products" section to view products listed by vendors.
- Manage Events: Access the "All Events" section to view and manage events created by vendors.

Handling Withdraw Requests

- View Withdraw Requests: Access the "Withdraw Requests" section to see all pending requests from vendors.
- Approve/Disapprove Requests: Review each request and approve or disapprove based on the platform's policies.

Vendor Guide

Account Setup and Login

- Signup: Navigate to the vendor signup page, fill in the necessary details, and submit the form.
- Login: Navigate to the vendor login page, enter your credentials, and click on the login button.

Managing Storefront

- Dashboard Access: Once logged in, you will be directed to the vendor dashboard where you can manage your storefront.
- Add Products: Use the "Add Products" section to list new products. Provide details such as name, description, price, stock, and images.
- Manage Products: Use the "Manage Products" section to update or remove existing product listings.

Handling Orders

- View Orders: Access the "Orders" section to view all orders placed by customers.
- Update Order Status: Update the status of orders as they move through processing, shipping, and delivery stages.

Creating and Managing Events

- Create Events: Use the "Create Events" section to create promotional events for your products.
- Manage Events: Use the "Manage Events" section to update or remove existing events.

Withdraw Funds

- Request Withdraw: Navigate to the "Withdraw" section to request the withdrawal of your earnings.
- View Withdraw Status: Check the status of your withdrawal requests in the same section.

Customer Guide

Account Creation and Login

- Signup: Navigate to the customer signup page, fill in the necessary details, and submit the form.
- Login: Navigate to the customer login page, enter your credentials, and click on the login button.

Browsing and Searching Products

- Homepage: Browse featured products, best deals, and categories on the homepage.
- Search: Use the search bar to find specific products by entering keywords.

Managing Cart and Wishlist

- Add to Cart: Click the "Add to Cart" button on product pages to add items to your cart.
- View Cart: Access the "Cart" page to view all items added to your cart.
- Add to Wishlist: Click the "Add to Wishlist" button to save products for later purchase.
- View Wishlist: Access the "Wishlist" page to view all saved items.

Checkout Process

- Initiate Checkout: Click the "Checkout" button in your cart.
- Step 1 - Details: Enter your personal and shipping details.
- Step 2 - Payment: Select a payment method (Credit Card, PayPal, COD).
- Step 3 - Confirmation: Review your order and complete the purchase.

Order Tracking and History

- Track Orders: Access the "Profile" page and navigate to the "Order History" section to track the status of your orders.
- View Order Details: Click on individual orders to view detailed information.

Contact Support

- Contact Form: Navigate to the "Contact Us" page and fill out the form to send queries to the support team.

Appendix B: Source Code

The source code for the @doorz platform is organized into several directories and files, each serving a specific purpose in the development of the application. Below is an overview of the source code structure.

Source Code:

https://github.com/affreelancer/atdoorz