

WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF SOFTWARE ENGINEERING

PROJECT TITLE: ONLINE MARKETPLACE PLATFORM

NAME	ID NO
EBISA GIRMA	NSR/0474/12
ELSABET AWRARIS	NSR/0515/12
ISAYAS MELKAMU	NSR/0791/12

ADVISOR: MR. JOSEPH WONDWOSEN

May 10, 2024

Wolkite University, Wolkite, Ethiopia

WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF SOFTWARE ENGINEERING

PROJECT TITLE: ONLINE MARKETPLACE PLATFORM

SUBMITED TO DEPARTMENT OF SOFTWARE ENGINEERING IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHLER OF SCIENCE IN SOFTWARE ENGINEERING

NAME	ID NO
EBISA GIRMA	NSR/0474/12
ELSABET AWRARIS	NSR/0515/12
ISAYAS MELKAMU	NSR/0791/12

ADVISOR: MR. JOSEPH WONDWOSEN

May 10, 2024

Wolkite University, Wolkite, Ethiopia

DECLARATION

This is to declare that this project work which is done under the supervision of Mr. Joseph Wondewosen, and having the title Online Marketplace Platform is the sole contribution of:

1. Ebisa Girma
2. Elsabet Awraris
3. Isayas Melkamu

No portion of the project work has been illicitly duplicated, constituting plagiarism. All referenced sections have been employed to support the ideas and have been appropriately cited. We acknowledge responsibility and accountability for any consequences should a breach of this declaration be substantiated.

Date: _____

Group Members:

Full Name	Signature
Ebisa Girma	_____
Elsabet Awraris	_____
Isayas Melkmau	_____

APPROVAL FORM

This confirms the receipt of the project SRS document titled "Online Marketplace Platform" from Ebisa Girma, Elsabet Awraris, and Isayas Melkamu by the Department of Software Engineering at Wolkite University's College of Computing and Informatics. Both the advisor and the Department of Software Engineering have endorsed this senior group project.

Advisor and Head Department Approval Form

Advisor Name	Signature	Date
_____	_____	_____
Department Head Name	Signature	Date
_____	_____	_____
Examiner Name	Signature	Date
_____	_____	_____
_____	_____	_____
_____	_____	_____

ACKNOWLEDGEMENT

First and foremost, we express our sincere gratitude to GOD for His constant presence in every aspect of our lives. His boundless love and unwavering mercy have been steadfast companions, guiding us through both the highs and lows, even in moments when we feel undeserving of such blessings.

A big thank goes to Wolkite University's College of Computing and Informatics for providing us with this opportunity to embark on our final project. This chance has significantly enhanced our learning experience, and we are truly grateful for it.

We also want to acknowledge and appreciate Mr. Joseph Wondwosen, our advisor, for his insightful intellectual guidance on our task responsibilities. His valuable advice and consistent support have been indispensable throughout our journey.

In closing, we extend our deep appreciation to our families for their unwavering support. Their encouragement has played a pivotal role in helping us successfully fulfil the mission set forth by Wolkite University.

TABLE OF CONTENTS

DECLARATION I
APPROVAL FORM II
ACKNOWLEDGEMENT III
LIST OF TABLES VII
LIST OF FIGURES VIII
LIST OF ABBREVIATION IX
ABSTRACT X
CHAPTER ONE 1
1. Introduction 1
1.1. Statement of Problem 1
1.2. Objectives of the Project 2
1.2.1. General Objective 2
1.2.2. Specific Objective 2
1.3. Feasibility Analysis 2
1.3.1. Operational Feasibility 3
1.3.2. Technical Feasibility 3
1.3.3. Economic Feasibility 3
1.4. Scope and Limitation of the project 3
1.4.1. Scope of the Project 3
1.4.2. Limitation of the Project 3
1.5. Significance of the project 4
1.6. Beneficiary of the project 4
1.7. Methodology of the Project 5
1.7.1. Data Collection Methodology 5
1.7.2. System Analysis and Design 5
1.7.4. System development Model 6
1.7.5. System Testing Methodology 6
CHAPTER TWO 8
2. DESCRIPTION OF THE EXISTING SYSTEM 8
2.1. Introduction of Existing System 8
2.2. Users of Existing System 8
2.3. Major Functions of the Existing System 9
2.4. Forms and Other Documents of the Existing Systems 9

2.5. Drawbacks of the Existing System 11
2.6. Business Rules of the Existing System 11
CHAPTER THREE 13
3. Proposed System 13
3.1. Functional Requirements 13
3.2. Non-Functional Requirement 15
3.2.1. Security 15
3.2.2. Performance 15
3.2.3. Error handling and Validation 15
3.2.4. Quality Issues 16
3.2.5. Backup and Recovery 16
3.2.6. Physical Environment 16
3.2.7. Resource Issues 16
3.2.8. User Interface and Human Factors 16
3.2.9. Hardware Consideration 17
3.2.10. Documentation 17
CHAPTER FOUR 18
4. System Analysis 18
4.1. System Model 18
4.1.1. Use Case Model 18
4.2. Object Model 60
4.2.1. Class Diagram 60
4.2.2. Data Dictionary 62
4.3. Dynamic Model 64
4.3.1. Sequence Diagram 65
4.3.2. Activity Diagram 68
4.3.3. State Chart Diagram 72
CHAPTER FIVE 74
5. Sytem Design 74
5.1. Design Goal 74
5.2. Proposed System Architecture 76
5.2.1. Subsystem Decomposition and Description 77
5.2.2. Hardware/Software Mapping 79
5.2.3. Detailed Class Diagram 80
5.2.4. Persistent Data Management 81

5.2.5. Access Control and Security 82
5.3. Packages 83
5.4. Algorithm Design 84
5.5. User Interface Design 84
CHAPTER SIX 86
6. Implementation and Testing 86
6.1. Implementation of the Database 86
6.2. Configuration of the Application Server 91
6.3. Configuration of Application Security 92
6.4. Implementation of User Interface 93
6.5. Testing 93
6.5.1. Test Case 93
6.5.2. Testing Tools and Environment 94
6.5.3. Unit Testing 95
6.5.4. System Testing 96
6.5.5. Integration Testing 98
CHAPTER SEVEN 100
7. Conclusion And Recommendation 100
7.1. Conclusion 100
7.2. Recommendation 100
8. REFERENCES 101

LIST OF TABLES

Table 4- 1 Use Case Description for Login *21*
Table 4- 2 Use Case Description of Registration *21*
Table 4- 3 Use Case Description for Manage Account *22*
Table 4- 4 Use Case Description for Manage Product Catagory *23*
Table 4- 5 Use Case Description for Approve Seller *24*
Table 4- 6 Use Case Description for View Order *25*
Table 4- 7 Use Case Description for Generate Report *26*
Table 4- 8 Use Case Description for View Product *27*
Table 4- 9 Use Case Description for Storefront Creation *29*
Table 4- 10 Use Case Description for Manage Product *30*
Table 4- 11 Use Case Description for Product Promotion *31*
Table 4- 12 Use Case Description for Chat *33*
Table 4- 13 Use Case Description for Feedback *34*
Table 4- 14 Use Case Description for Order Product *35*
Table 4- 15 Use Case Description for Payment *36*
Table 4- 16 Use Case Description fro Rating Products *37*
Table 4- 17 Use Case Description for View assigned delivery *38*
Table 4- 18 Use Case Description for Confirm delivery *39*
Table 4- 19 Use Case Description for Logout *40*
Table 4- 20 Data Dictionary for User *62*
Table 4- 21 Data Dictionary for Product Category *62*
Table 4- 22 Data Dictionary for Transaction *63*
Table 4- 23 Data Dictionary for Product *63*
Table 4- 24 Data Dictionary for Order *63*
Table 4- 25 Data Dictionary for Review *64*
Table 4- 26 Data Dictionary for Conversation *64*
Table 4- 27 Data Dictionary for Feedback *64*

LIST OF FIGURES

Figure 2- 1 Customer order on Shopfy *9*
Figure 2- 2 Product List on Alibaba *10*
Figure 2- 3 Shipping information Amazon *10*
Figure 2- 4 Shopfy Dashboard *11*
Figure 4- 1 Use Case Diagram *20*
Figure 4- 2 Class Diagram *61*
Figure 4- 3 Sequence Diagram for Login *65*
Figure 4- 4 Sequence Diagram for Order Product *66*
Figure 4- 5 Sequence Diagram for Storefront Creation *67*
Figure 4- 6 Sequence Diagram for Update User *67*
Figure 4- 7 Activity Diagram for Login *68*
Figure 4- 8 Activity Diagram for Order Product *69*
Figure 4- 9 Activity Diagram for Storefront Creation *70*
Figure 4- 10 Activity Diagram for User Update *71*
Figure 4- 11 State Chart Diagram for Login *72*
Figure 4- 12 State Chart Diagram for Order Product *72*
Figure 4- 13 State Chart Diagram for View Product *73*
Figure 5- 1 Proposed System Architecture *77*
Figure 5- 2 Component Diagram *79*
Figure 5- 3 Deployment Diagram *80*
Figure 5- 4 Detail Class Diagram *81*
Figure 5- 5 Persistent Data Diagram *82*
Figure 5- 6 Package Diagram *83*
Figure 5- 7 Login Page User Interface *85*
Figure 5- 8 Dashboard User Interface *85*

LIST OF ABBREVIATION

IDE	Integrated Development Environments
BR	Business Rule
UML	Unified Modelling Language
API	Application Programming Interface
UI	User Interface
UC	Use Case
ID	Identification
PDF	Portable Document Format
CSV	Comma-Separated Values
RAM	Random Access Memory
RBAC	Role Based Access Control
JWT	JSON Web Tokens
JS	JavaScript
E-commerce	Electronic Commerce
Enum	Enumeration
MongoDB	Mongo Database

ABSTRACT

This abstract outlines the core features and benefits of an Online Marketplace Platform. In the dynamic landscape of e-commerce, an online marketplace serves as a digital hub connecting buyers and sellers, providing a virtual space for transactions. The platform is designed with user-friendly interfaces, facilitating seamless navigation and enhancing user experience.

Key components of the Online Marketplace include robust product listings, secure payment gateways, and efficient order processing. The platform fosters a competitive environment, empowering sellers to reach a broader audience and buyers to access a diverse range of products. Integration of advanced search and trending enhances discoverability, ensuring users find products tailored to their preferences.

Security measures, such as encrypted transactions and user data protection, are paramount to instill trust among users. Additionally, customer support features contribute to a positive user experience by addressing concerns promptly.

The success of the Online Marketplace Platform hinges on continuous innovation, adapting to market trends, and fostering a vibrant community. This abstract encapsulates the essence of an online marketplace, emphasizing its pivotal role in shaping the future of digital commerce.

CHAPTER ONE

1. INTRODUCTION

The essence of economic growth is rooted in commerce, and a well-functioning commerce system is the very foundation of a nation's advancement. In today's digital era, E-commerce technology such as Shopify, Amazon and others presents incredible opportunities to simplify and optimize procedures in online commerce. The E-commerce is highly significant due to its direct relevance to the global economy. The current problem in e-commerce is that business is shifting from only company to individual, without including individual sellers. The motivation behind this project selection arises from of creating a neutral space where buyers and sellers can come together and negotiate the terms of their deal without third party.

"Online Marketplace Platform" is a web-based software application that acts as a virtual market hub where multiple sellers can set up their individual storefronts to sell products to a wide customer. It is a digital environment that connects multiple independent sellers with potential buyers, facilitating the buying and selling of various goods. These platforms provide the space where sellers can list their products, set prices, and make deals with buyers, while buyers can browse, compare, and purchase items from different sellers.

1.1. Statement of Problem

Individual sellers and customers face several problems associated with existing system such as e-commerce and traditional market. This system has several drawbacks, including:

- ✓ **Limited market reach for sellers**: Sellers might have a hard time finding customers who are far away from them, so they may only be able to sell to people who are close by.
- ✓ **Business-to-Individual sales**: The e-commerce system primarily supports business-to-individual way of marketing which allows only the e-commerce system owners. However, it lacks a platform for direct individual-to-individual marketing which would facilitate direct buyers and individual seller communication.
- ✓ **Difficulty in Product Promotion**: Sellers encounter challenges when attempting to effectively promote their products to a broader customer, mainly because they rely on a traditional market system, and e-commerce companies do not permit individual sellers to operate independently.

- ✓ **Limited Regional Availability**: The current system may not provide access to customers or sellers in all regions, limiting its coverage or may the e-commerce company not in the customer region.
- ✓ **Delivery Costs and Wait Times:** The problem with the current system is that it makes deliveries expensive, which can stop people and company from buying or selling things. Also, it takes a long time for things to be delivered, which can make people not want to use the system.
- ✓ **Lack of Competitive Pricing**: The existing system may not encourage competitive pricing, which can result in higher costs for customers.
- ✓ **The system is not scalable:** The existing traditional market system is not scalable, which means that it cannot be easily expanded to accommodate more stores or more products. This can limit the growth of the business.
- ✓ **The system is not secure:** The existing traditional market system is not secure, which means that data is at risk of being accessed by unauthorized users. This can lead to data breaches or other security incidents.

1.2. Objectives of the Project

1.2.1. General Objective

The main objective of this project is to develop an Online Marketplace Platform. We aim to develop this system to serve as the foundation for our entrepreneurial journey.

1.2.2. Specific Objective

The specific objectives of the project are the listed below.

- ✓ To analyze the limitations of the existing system and specify the new system.
- ✓ To implement the proposed and specified system in efficient way
- ✓ To enable sellers to register on the platform and create their storefronts, allowing them to showcase their products effectively.
- ✓ To implement a comprehensive system for managing product listings, enabling sellers to upload product details, images, and inventory information.
- ✓ To incorporate ratings and reviews, advanced search and filtering, user-to-user chat functionality, and reporting and analytics by enhancing the overall functionality and user experience of the system.

1.3. Feasibility Analysis

1.3.1. Operational Feasibility

The Online Marketplace Platform we are going to develop is designed to encompass all necessary operations to achieve its objectives. Its feature is user-friendly interface that seamlessly integrates with the environment and ensures the smooth execution of all user tasks without cumbersome procedures. Hence, the project is operationally feasible.

1.3.2. Technical Feasibility

The proposed system is designed for easy maintenance and repair, reducing the need for highly specialized experts or extensive technical support. This is achievable through the system's utilization of adaptable technologies and by offering training and guidance to the system owners, who already possess some basic technology knowledge, enabling them to easily use the system. Therefore, the project is technically feasible.

1.3.3. Economic Feasibility

The economic feasibility of the proposed system is determined by comparing the costs of system development with the benefits it provides. In our case, the new system offers numerous advantages to its users. It is cost-effective due to the utilization of low-cost software and hardware. Furthermore, it leads to significant cost savings by eliminating the need for traditional marketing efforts and materials, reducing expenses associated with market visits, and more.

1.4. Scope and Limitation of the project

1.4.1. Scope of the Project

The scope of the project is designing and developing Online Marketplace Platform. It includes everything regarding the efficient functioning of the online marketing such as:

- ✓ Seller Registration and Storefront Creation
- ✓ Product Management
- ✓ Ratings and Reviews
- ✓ Search and Filters
- ✓ Reports and Analytics
- ✓ User-to-User Chat

1.4.2. Limitation of the Project

The scope of Online Marketplace Platform excludes the following subsystems because of time constraint:

- ✓ User-to-User Chat tracking for trust issue

1.5. Significance of the project

This project has many significances. The following are some of the significances of the project.

- ✓ Enhanced Market Access: It provides sellers with a broader customer reach, enabling access to local, or national which might not have been feasible through traditional market channels.
- ✓ Easy accessibility and convenience: provide a wide selection of goods in one place and allows customers to shop whenever and from anywhere.
- ✓ Diverse Product Offerings: Having a large range of goods available in one location gives customers a lot of options across different categories.
- ✓ Economic Growth: Facilitates trade and commerce, creating job opportunities and business expansion.

1.6. Beneficiary of the project

This project has many beneficiaries to users of the system. The following are some of the beneficiaries of the project.

- ✓ **Sellers:** Seller's benefit from this online marketplace platforms by gaining access to a wider customer. This platform allows sellers to extend their market reach, which is often difficult to achieve through traditional means. By showcasing their products on this platform, sellers can establish a stronger online presence and connect with a diverse range of potential customers.
- ✓ **Buyers:** Buyers, whether individuals or businesses, enjoy several advantages on online marketplace platform. They have access to a wide variety of products, making it easier to find exactly what they need. This platform often encourages competitive pricing, with sellers competing to offer the best deals, ultimately saving buyers money.
- ✓ **Delivery Personnel:** They earn income through delivery fees and may receive additional tips from satisfied customers. The online marketplace platform ensures a consistent flow of delivery requests, providing delivery personnel with steady work opportunities.
- ✓ **System Owner:** System owner typically earn a commission as part of their revenue model. This commission is a percentage of each transaction's value that occurs on the online marketplace platform they operate. It serves as a primary source of income for

the system owners, enabling them to sustain and improve platform operations, invest in growth, and provide ongoing services to the platform's users.

1.7. Methodology of the Project

1.7.1. Data Collection Methodology

We used the following methods for collecting data.

- ✓ **Document analysis:** During the document analysis, we reviewed previous materials that contribute to the project's development, with a specific focus on identifying papers that can bring additional features to the project.
- ✓ **Website Feedback Collection**: In addition to document analysis, we gathered valuable insights by collecting feedback from various websites related to e-commerce and traditional markets. This method complemented the document analysis and provide a more comprehensive understanding of user preferences and market dynamics.

1.7.2. System Analysis and Design

- ✓ **High-Level Design:** We have chosen to use programs like Microsoft Visio and Drawio in order to generate a high-level system design. With the aid of these tools, comprehensive system architectural diagrams and flowcharts can be created more easily, which helps to clearly illustrate system components and their interconnections. By providing a clear depiction of the system's structure, this visual clarity promotes operational feasibility and facilitates more effective communication among stakeholders.
- ✓ **Low-Level Design:** We use Visual Studio Code for low-level system design. Sophisticated code editing features offered by these integrated development environments (IDEs) make the task of careful implementation planning and code documentation easier. By guaranteeing accurate code execution and simple problem identification, their potent debugging tools make systems more operationally feasible and more reliable.

1.7.3. Developing Tool and Methodology

In alignment with the current technology landscape, we have selected the most suitable programming languages and databases based on the following criteria:

- ✓ Ease of learning and efficient server-side performance.
- ✓ Power in creating dynamic and interactive web pages.

- ✓ Wide compatibility with various databases.
- ✓ Open-source nature.
- ✓ Cross-platform capabilities.
- ✓ Rich availability of frameworks.

Front End:

- ✓ Next JS (built on React)

Back End:

- ✓ Node JS with Express JS framework
- ✓ MongoDB

1.7.4. System development Model

The project developed following Agile methodologies, specifically the Scrum framework. The adoption of Agile practices is driven by the following considerations:

- ✓ it's a good fit for developing our project within our implementation timeframe.
- ✓ Agile's flexibility is helpful because our project requirement is always changing, and it lets us develop in smaller steps.
- ✓ Focus on continuous improvement
- ✓ Enhanced team collaboration,
- ✓ Keeping track of progress helped us be more efficient and make sure our project succeeds in implementation timeframe.

1.7.5. System Testing Methodology

The development team performed comprehensive testing encompassing both functional and non-functional aspects prior to the deployment of the Online Marketplace Platform. Functional testing verified the fulfilment of all requirements, while non-functional testing concentrated on security, reliability, and system robustness. The objective is to release flawless software that aligns with user expectations, functions seamlessly, and upholds elevated standards of usability and security.

Unit Testing:

Unit testing involves testing individual components or functions of the code in isolation to ensure they work as intended. For our project, we used testing frameworks Jest, which is

commonly used for JavaScript and React applications. Jest provides a simple and effective way to write and run unit tests for React.js components and functions.

Integration Testing:

Integration testing focuses on testing the interactions between different components or modules to ensure they work together correctly. In our project, we used Jest to perform integration tests. Jest supports testing asynchronous code, making it suitable for testing the integration of different parts of your application.

System Testing:

System testing involves testing the entire system as a whole to ensure that all components and modules work together seamlessly. In our project, we used end-to-end testing framework Cypress to simulate user interactions and test the complete application flow.

CHAPTER TWO

2. DESCRIPTION OF THE EXISTING SYSTEM

2.1. Introduction of Existing System

Currently, there are two types of Existing systems, such as E-commerce and traditional marketing, which represent distinct approaches to the marketplace. The following describes the existing systems of both E-commerce and traditional markets and explains how they work step by step.

E-commerce System and traditional markets:

E-commerce is a system that allows users to purchase and sell products online. In the current E-commerce landscape, sellers are those who have their own stores, like Amazon, Alibaba, etc. However, buyers can be anyone who purchases products online. This E-commerce platform serves as a connection between buyers and only those sellers who have their own stores and sell from their stores. In this system, buyers first register, and then they have the role of searching for any product they want and buying it online. The owner of the E-commerce platform has the responsibility to manage the orders of the customers (buyers) and deliver their products.

On the other hand, a traditional market system is a traditional method of buying and selling goods that connects sellers and buyers physically. In this system, sellers set up stalls in designated areas of the market, often paying rent to market organizers for the space. Customers (buyers) visit these markets to interact with sellers, inspect items, and negotiate prices. Payments are typically made in cash, and the exchange of goods occurs directly between the buyer and seller.

2.2. Users of Existing System

In the existing system, there are three primary types of users, each playing a crucial role in the operation of the system.

Buyers in E-commerce are the customers who browse the platform, search for products, and make purchasing decisions. They add items to their shopping cart, provide accurate shipping information, complete the payment process, and leave reviews and ratings for products based on their experiences. On the other hand, buyers in a traditional market system are the customers who visit the markets to interact with sellers, inspect items, and negotiate prices.

Sellers, in E-commerce are typically companies, are responsible for listing and selling products on the platform. They create and maintain product listings with accurate descriptions, manage their inventory, handle order processing, ship products, and provide excellent customer service to buyers. In a traditional market system, sellers set up their stalls or storefronts in designated areas of the market. They physically display and sell their products to buyers.

Delivery personnel, in E-commerce are particularly those responsible for transporting goods from sellers to buyers, play a crucial role in logistics and distribution. They ensure that products reach their intended recipients in a timely and secure manner.

2.3. Major Functions of the Existing System

Current E-commerce systems offer a variety of functionalities, including user registration and authentication, product listings, search and discovery features, shopping cart and checkout process, order management, feedback and ratings system, data security and privacy measures, shipping and delivery management, inventory management, and reporting and analytics tools.

2.4. Forms and Other Documents of the Existing Systems

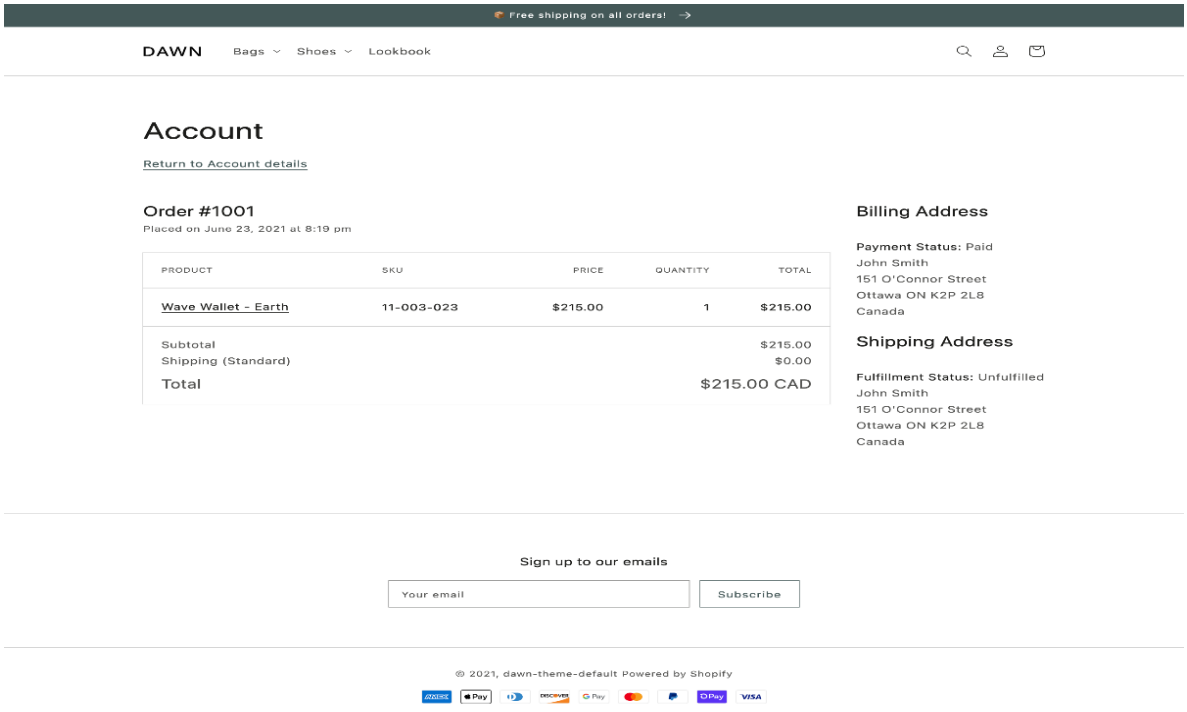


Figure 2- 1 Customer order on Shopfy

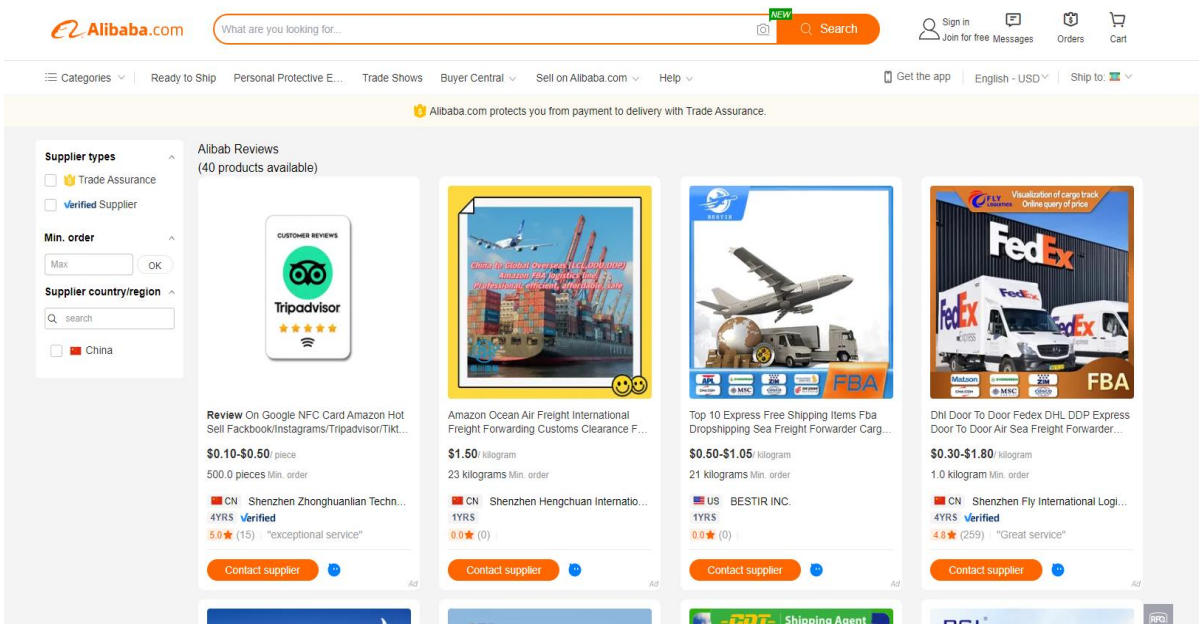


Figure 2- 2 Product List on Alibaba

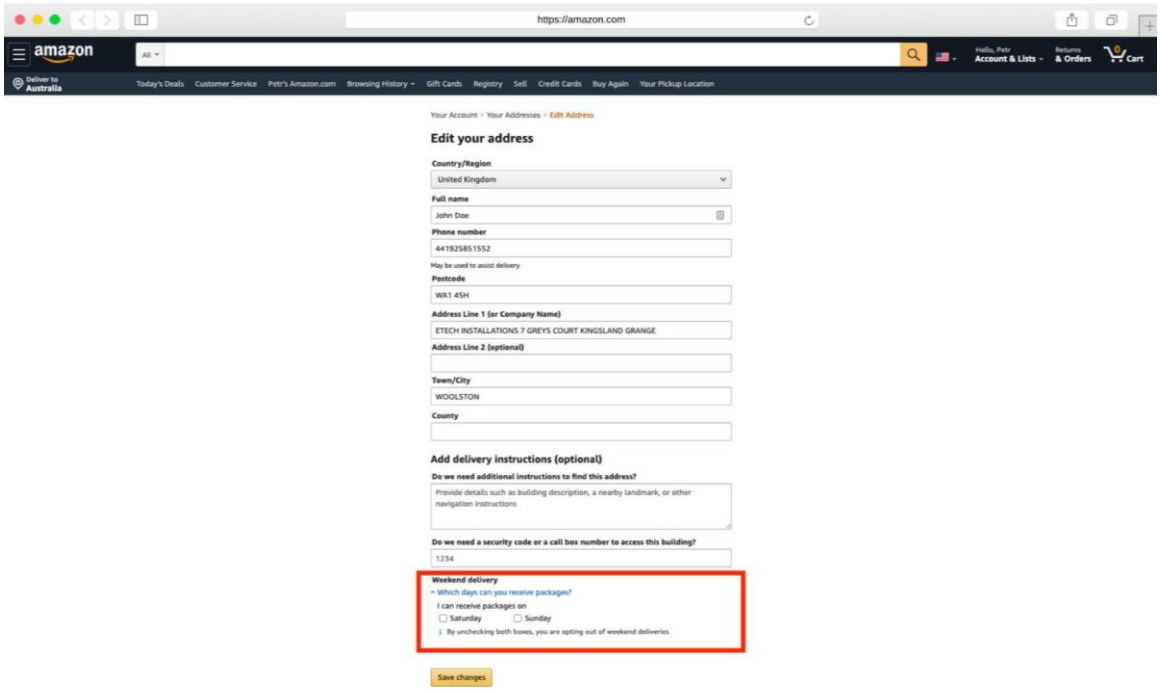


Figure 2- 3 Shipping information Amazon

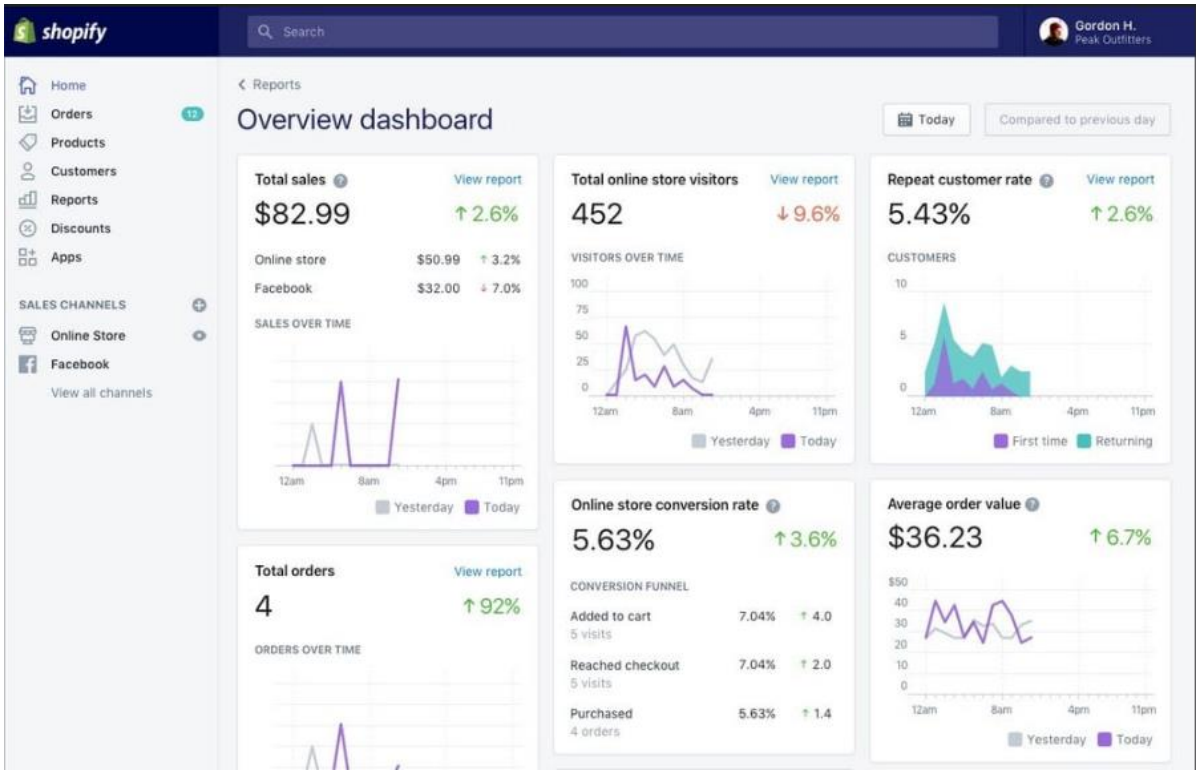


Figure 2- 4 Shopfy Dashboard

2.5. Drawbacks of the Existing System

The existing e-commerce and traditional market systems have several drawbacks for individual sellers and customers. These include limited market reach, the absence of an individual-to-individual marketing platform, difficulties in product promotion, limited regional availability, high delivery costs and wait times, and a lack of competitive pricing. The traditional market system specifically suffers from weak data security and low scalability. These issues affect the efficiency, convenience, and profitability of both online and offline commerce.

2.6. Business Rules of the Existing System

E-Commerce Business Rules:

BR1: Customers must create an account and provide accurate personal information before making a purchase.

BR2: Each product must have a detailed description, including price, specifications, and availability.

BR3: Customers are required to provide a valid payment method during the checkout process.

BR4: Discounts or promotional offers are applicable based on specific conditions, such as a minimum purchase amount or a promotional code.

BR5: The e-commerce platform must provide a secure and encrypted payment gateway to protect customer financial information.

BR6: Customers can return a product within 30 days of purchase, but it must be in its original condition and accompanied by the receipt.

BR7: The e-commerce platform must ensure the privacy and security of customer data, following to relevant data protection regulations.

BR8: Products out of stock should be clearly labelled as such, and customers should be informed when the product becomes available again.

BR9: Customer reviews and ratings should be moderated to ensure authenticity and appropriateness.

BR10: The e-commerce platform provides customer support through various channels, such as live chat, email, and phone, to address inquiries and issues promptly.

Traditional Market Business Rules:

BR1: Sellers prominently display prices for all products, and any discounts or promotions should be clearly communicated.

BR2: All transactions are conducted using the local currency, and sellers should provide accurate exchange.

BR3: Market operating hours or days are clearly communicated to both sellers and customers.

BR4: Market management should enforce rules regarding the allocation of marketplaces or spaces to sellers and ensure fair competition.

BR5: Sellers are required to obtain the necessary permits and licenses to operate in the market.

BR6: The market should have a mechanism for resolving disputes between sellers or between sellers and customers.

BR7: Sellers should not engage in deceptive practices or misrepresentation of their products.

CHAPTER THREE

3. PROPOSED SYSTEM

The "Online Marketplace Platform" is designed to address the problems faced by both sellers and buyers in traditional markets, as well as individual sellers in e-commerce. This system is used to address various issues such as data loss, ensuring easy and persistent product information storage, facilitating seamless communication between sellers and buyers within our platform, and resolving security concerns. In the process of designing and evaluating our system, we employ object-oriented software techniques, including the Unified Modelling Language (UML), which encompasses state charts, sequence diagrams, activity diagrams, deployment diagrams, component diagrams, class diagrams, and use case diagrams. When it comes to implementation, we utilize various programming languages such as Node.js for the backend, Next.js built on React.js for the frontend, and MongoDB for database storage. The intended system is highly user-friendly and efficiently manages data and tasks, outperforming existing systems.

The main Purpose of the proposed system is: -

- ✓ Facilitate improved access to the market.
- ✓ Provide users with easy access and convenience by enhancing product management.
- ✓ Enable a wide range of product options.
- ✓ The system has a contribution to economic growth.
- ✓ The system implemented with effective data storage practices to prevent data loss.
- ✓ Avoiding improper resource consumption
- ✓ Efficiency of the system increases in terms of time and performance.
- ✓ Simplify and streamline transaction processes for both buyers and sellers.
- ✓ Allow for scalability to accommodate growing numbers of users and products.
- ✓ Build trust and credibility between buyers and sellers in the online marketplace.
- ✓ Provide a platform for user feedback and product reviews to enhance the buying experience.

3.1. Functional Requirements

- ✓ **Buyer Registration:** Create an account on the Marketplace Platform by providing necessary information to establish identity.
- ✓ **Storefront Creation:** After registration buyer fills their detail identity to become seller and can set up their individual storefronts, including custom branding and store information by display your products, manage your orders, interact with customers.
- ✓ **Product management:** Sellers can post, update and delete a wide variety of products with detailed descriptions, images, prices, and categories
- ✓ **Inventory Management:** The system provides tools for sellers to manage their inventory, track product levels, and receive notifications for low product.
- ✓ **Ratings and Reviews:** The system provides customers with reviews and ratings for seller's products.
- ✓ **Search and Filters:** A robust search engine and filtering options to help customers easily find and compare products from different sellers.
- ✓ **Reports and Analytics:** Generate reports for sellers to track product sales, customer behaviour, and inventory turnover
- ✓ **User-to-User Chat:** Enable users to chat with each other within the platform. This can facilitate communication between buyers and sellers, allowing them to discuss product details, negotiate prices, and process transactions.
- ✓ **Payment Method:** Buyers can make payments for products using the system, and sellers can withdraw their balances following the delivery confirmation.
- ✓ **Product Trending:** Implement trending system that suggests products to customers based on their purchase behaviour. This can encourage cross-selling and upselling.
- ✓ **Order Tracking:** Provide a feature for customers to track the status and location of their orders in real-time. This enhances transparency and improves the overall customer experience.
- ✓ **Promotions and Discounts:** Allow sellers to create promotional campaigns and offer discounts on their products. This feature can attract more customers and drive sales.
- ✓ **Wishlist and Favourites:** Enable users to create wish lists or mark products as favourites for future reference. This helps in retaining user engagement and provides insights into popular products.
- ✓ **Delivery Person Registration:** The admin can register delivery personnel to manage order deliveries efficiently and deliver all order from the seller to customer.
- ✓ **Delivery:** Integrate delivery methods to provide flexibility standard shipping enhancing the overall delivery experience for customers.

3.2. Non-Functional Requirement

Non-functional requirements are in place to enhance the system's quality and performance. The proposed system encompasses the following non-functional requirements.

3.2.1. Security

For any information system, it is essential to ensure security across its three vital aspects: confidentiality, integrity, and availability. To achieve this security, our system development approach incorporates an authentication mechanism. This mechanism involves providing users with a unique username, password and role, thereby restricting access to the system solely to authenticated users. Moreover, our choice of Next.js is excellent for developing authentication and authorization systems due to its robust features. These include server-side rendering, static site generation, API routes, middleware's, and built-in support for popular authentication providers. These features enhance the security of the system and helps prevent potential attacks.

3.2.2. Performance

Response Time: In regular circumstances, the system made every effort to swiftly present results when a user initiates an inquiry. Next.js reduces response time by rendering pages on the server, optimizing assets, streaming UI, and using a faster bundler.

Processing Time: As the system developed using efficient programming language JavaScript and a well-structured database which use embedding and referencing way of data storing, it efficiently processed user activity requests and provide swift responses under standard operating conditions.

Concurrent Processing: The system designed to accommodate multiple users simultaneously, ensuring efficient and uninterrupted operation. Next.js can support concurrent users through its use of serverless functions and its routing system. MongoDB can support concurrent users through its use of sharding and replication.

3.2.3. Error handling and Validation

The proposed system incorporates validation rules, with the server side powered by Node.js and the client side using the Next.js built on React.js JavaScript library for validation and error management. This approach aims to minimize input errors. In cases of errors, the user interface generate informative error messages that are concise, meaningful, and clear, providing users with a clear understanding of the issue and guidance on how to resolve it. Furthermore, the system allows users to confirm actions like creation, deletion, or modification before they take

place. In response to erroneous inputs, the system prompts users to re-enter data in the correct format. Thus, our system maintains robust error handling and input validation through JavaScript error-handling mechanisms and datatype checks for incoming inputs.

3.2.4. Quality Issues

Availability: The system consistently accessible 24/7, whether operating within a local area network or a wide area network.

User Operability: The system developed with user-centric features, ensuring that even users with basic computer knowledge can easily operate its functions.

Reliability: The proposed system is designed to minimize the likelihood of crashes during its runtime, particularly since multiple users may access it simultaneously. This is achieved through effective user input validation and the prevention of incorrect storage of records.

3.2.5. Backup and Recovery

Maintaining a backup copy of data in a separate location ensures that critical information remains accessible even if the primary storage system fails or malfunctions. By replicating data elsewhere, system owners can safeguard their valuable assets and minimize downtime in the event of data loss or corruption. Regular backups provide a safety net, allowing for swift data restoration and business continuity.

3.2.6. Physical Environment

The system deployed and installed on a server computer. However, for enhanced features and improved resilience, we suggest deploying the system on a cloud platform that is not vulnerable to disasters.

3.2.7. Resource Issues

The system demands significant processing power and memory resources on both the client and server sides. This ensures that the system operates efficiently and effectively. Furthermore, these resource requirements guarantee optimal provisioning and maintain consistency in system performance.

3.2.8. User Interface and Human Factors

The proposed system under development feature an intuitive graphical user interface (GUI), ensuring that users can easily interact with it. Since users are accustomed to using computers, they can communicate with the system with ease.

3.2.9. Hardware Consideration

To proposed system performs optimally, it be compatible and highly responsive across commonly used mobile and desktop web browsers. The system is crafted to function smoothly, guaranteeing a seamless user experience with no malfunctions. This design accommodates the requirements of desktop applications, enabling customers to conveniently utilize their mobile web browsers for accessing and processing marketplace information.

3.2.10. Documentation

As part of our development process, we are creating a user manual documentation that ensures users encounter no difficulties when using the system. This user guide serves as a comprehensive resource, offering users a clear and detailed understanding of the system's operations and functions. With complete documentation, any potential issues or failures can be addressed and resolved with ease.

CHAPTER FOUR

4. SYSTEM ANALYSIS

4.1. System Model

This chapter encompasses several key tasks, including the creation of a use case model, documentation of the course of events for each use case, development of sequence and activity diagrams. Throughout this process, various object-oriented diagrams are employed, such as the system model use case diagram and object model class diagram, along with data directories. Additionally, dynamic models like state charts, sequence diagrams, and activity diagrams are utilized to effectively depict the system.

4.1.1. Use Case Model

4.1.1.1. Use Case Diagram

Use Case diagrams serve to depict the actors who initiate specific use cases and also highlight instances where an actor receives information from a use case. Essentially, these diagrams provide a visual representation of the system's requirements. Actors, defined as individuals, organizations, or external systems, play roles in various interactions with the system. In the context of our project, the actors and the associated use cases participating in the system are described.

1. **Admin:**
    - ✓ Manage Product Category
    - ✓ Manage Account
    - ✓ Approve seller
    - ✓ View orders
    - ✓ Generating reports
2. **Seller:**
    - ✓ Manage Products.
    - ✓ Chat
    - ✓ View Orders.
    - ✓ Product Promotion
    - ✓ Generating report
    - ✓ Provide Feedback
    - ✓ Receive payment
3. **Buyer:**

- ✓ View Product
- ✓ Registration
- ✓ Chat
- ✓ Make payment
- ✓ Make Orders
- ✓ Rating product
- ✓ Confirm Delivery
- ✓ Storefront Creation

4. **Delivery Personnel**
    - ✓ View Product
    - ✓ Chat
    - ✓ View Assigned Delivery
    - ✓ Process Payments
5. **Payment Processor:**
    - ✓ Process Payments



Figure 4- 1 Use Case Diagram

4.1.1.2. Use Case Diagram Description

The following tables show use cases description based on the system business or context of our system.

Table 4- 1 Use Case Description for Login

Use case id	UC-1	
Use case Name	Login	
Actor	All actors	
Description	Allows the users login into the system with their privilege page	
Precondition	The user must have user name and password	
Post condition	User get access to the system according to their predefined system privilege and finally logout or turn off the page	
Basic course of action	User action	System response
	Step 1. User activates the system. Step 3. User fills user name and password and click login button. Step 5. User authenticated and gets access to the system. Step 6. Use case end.	Step 2. The system response by displaying the login interfaces and prompts the user for the user's name and password. Step 4. System verifies user name and Password and display privilege page.
Alternative course of action	The system displays error message when the user enters incorrect password and user name and the system returns to step 3.	

Table 4- 2 Use Case Description of Registration

Use case id	UC-2
Use case Name	Registration
Actor	Buyer, Seller

Description	Provide a seamless and efficient registration process for both buyers and sellers.	
Precondition	The online marketplace platform is accessible and operational. The user has a device with internet connectivity.	
Post condition	The customer has a user account on the online marketplace platform. The customer is logged into the platform.	
Basic course of action	User action	System response
	Step 1. Users navigates to the platform's registration page. Step 3. Users fills out the form with accurate information and chooses a secure password. Step 5. Use case end.	Step 2. The system displays a clear and concise registration form with required fields. Step 4. Upon successful registration then directs them to their account dashboard or homepage.
Alternative course of action	If there are errors in the registration form, the system provides clear error messages and prompts users to correct the information than returns to step 3. In case of email or phone number duplication, the system alerts users and guides them to use unique credentials.	

Table 4- 3 Use Case Description for Manage Account

Use case id	UC-3
Use case Name	Manage Account
Actor	Admin
Description	Efficiently manage and oversee user accounts (buyers, sellers and delivery personnel) on the online

	marketplace platform, ensuring a smooth and trustworthy experience for all users.	
Precondition	The admin has logged in to their account on the online marketplace platform.	
Post condition	The admin can manage all user accounts on the platform. The admin can view and edit user profiles. The admin can activate or deactivate user accounts.	
Basic course of action	User action	System response
	Step 1. Admin logs into the admin portal using secure credentials. Step 3. Admin navigates to the "User Management" section within the admin dashboard. Step 5. Admin views a list of registered user accounts with essential details. Step 7. Admin modifies user account information and activate/deactivate user account. Step 9. Uue case end.	Step 2. System verifies credentials and grants access to the admin dashboard. Step 4. System loads the user management interface, displaying a list of registered users. Step 6. The system displays the list of user accounts, including usernames, emails, and account statuses. Step 8. the system apply user account modifications.
Alternative course of action	If there are errors during account information updates, the system provides clear error messages and prompts the admin to correct the information.	

Table 4- 4 Use Case Description for Manage Product Catagory

Use case id	UC-4

Use case Name	Manage Product Category	
Actor	Admin	
Description	Efficiently organize, maintain, and optimize product categories within the online marketplace platform,	
Precondition	Authentication as an admin user.	
Post condition	The product category management updates are reflected in the online marketplace platform.	
Basic course of action	User action	System response
	Step 1Admin logs into the admin portal and navigates to the "Category Management". Step 3. Admin adds a new product category, edits existing category details, and deactivates a product category. Step 5. End use case.	Step 2. The system verifies the credentials and loads the category management interface, displaying a list of existing product categories. Step 4. The system adds the new category, updates the category details and removes the category from the system.
Alternative course of action	If the admin attempts to add a new category with a name that already exists, the system prompts the admin to choose a unique name. If the admin attempts to deactivate a category, the system may prompt the admin to deactivate products exist under that category as well.	

Table 4- 5 Use Case Description for Approve Seller

Use case id	UC-5
Use case Name	Approve Seller
Actor	Admin

Description	Review and approve new seller applications to maintain a high-quality marketplace experience for buyers and ensure platform regulations are met.	
Precondition	Platform has a pre-defined seller registration process for collecting information and documents. Admin has access to a dedicated dashboard for reviewing applications. Platform has established clear criteria for seller approval.	
Post condition	Approved seller can start listing products on the platform.	
Basic course of action	User action	System response
	Step 1. Seller registers and completes application form, providing information like business details, legal documents, and product categories. Step 3. accesses the dashboard and opens the seller application with the assigned ID. Step 5. Use case end.	Step 2. Platform confirms application submission and assigns a unique ID for tracking. Step 4. Platform displays application details, including submitted information and documents.
Alternative course of action	The system displays error message when Application may contain missing or incomplete information and the system returns to step 1.	

Table 4- 6 Use Case Description for View Order

Use case id	UC-6
Use case Name	View Order
Actor	Seller
Description	Efficiently monitor, manage, and fulfill orders received on the platform.

Precondition	Seller has an active account on the platform with access to their seller dashboard. Platform has a dedicated section for managing orders within the seller dashboard.	
Post condition	The seller has efficiently viewed and managed incoming orders through the online marketplace platform. Order statuses and details are updated in real-time to maintain accuracy.	
Basic course of action	User action	System response
	Step 1. Seller opens the platform and logs in to their account. Step 3. Seller navigates to the "Orders" section within the dashboard. Step 5. Seller selects an order from the list to view its details. Step 7. Use case end.	Step 2. The system displays the seller dashboard with various options and menus. Step 4. The system displays a list of all received orders, categorized by status (e.g., "Processing," "Completed"). Step 6. The system displays a dedicated page with comprehensive information about the selected order.
Alternative course of action	If a customer cancels an order, the system updates the order status accordingly, and the seller is notified.	

Table 4- 7 Use Case Description for Generate Report

Use case id	UC-7
Use case Name	Generate Report

Actor	Admin, Seller	
Description	Generate comprehensive and customizable reports to provide actionable insights for platform growth and user experience improvement.	
Precondition	Users have appropriate access levels to generate and view reports based on their roles.	
Post condition	The admin and seller have successfully generated and reviewed the desired reports. The insights obtained from the reports support decision-making and business optimization.	
Basic course of action	User action	System response
	Step 1. Both the admin and seller log into their respective accounts on the online marketplace platform. Step 3. Within their dashboard, both the admin and seller locate and click on the "Reports" section. Step 5. The user (admin or seller) customizes the report by specifying parameters. Step 6. Use case end.	Step 2. The system redirects them to their personalized dashboard. Step 4. The system presents a variety of report types for the admin and seller. Step 6. The system processes the selected parameters and generates a detailed report based on the specified criteria.
Alternative course of action	In case of technical issues during report generation, the system provides an error message and guides the user to try again.	

Table 4- 8 Use Case Description for View Product

Use case id	UC-8
Use case Name	View Product
Actor	Admin, Seller, Buyer

Description	Provide each user type with a relevant and informative product view experience, tailored to their specific needs and roles on the platform.	
Precondition	The users must be logged into the online marketplace platform. Active product listings should be available on the platform.	
Post condition	The users have access to detailed information about the viewed product. The system updates the user's history with the viewed product. If added to the cart or Wishlist, the system reflects the changes accordingly.	
Basic course of action	User action	System response
	Step 1. The users navigate through product categories or uses the search function. Step 3. The users click on a specific product of interest. Step 5. The users review detailed product information. Step 7. The users decides to add the product to the shopping cart for immediate purchase or adds it to the Wishlist. Step 8. Use case end.	Step 2. The system presents a list of relevant product listings. Step 4. The system loads the product details page. Step 6. The system displays comprehensive details, including images, name, description, price, seller information, customer reviews, and specifications.
Alternative course of action	If a product is temporarily unavailable or out of stock, the system indicates this status to buyers and sellers.	

Table 4- 9 Use Case Description for Storefront Creation

Use case id	UC-9	
Use case Name	Storefront Creation	
Actor	Buyer	
Description	Create and customize their own online storefront within the platform, showcasing their products effectively and attracting potential buyers.	
Precondition	The buyer must be registered and logged into the online marketplace platform. The buyer account should be in good standing with no outstanding violations. The platform must support the creation of digital storefronts.	
Post condition	The buyer has successfully set up a storefront with personalized branding. The digital storefront is visible to Buyers within the online marketplace. The system updates the buyer's account with the newly created storefront details.	
Basic course of action	User action	System response
	Step 1. The buyer logs into their account and selects the "Create Storefront" option. Step 3. The Buyer provides essential information such as store name, and a brief description. Step 5. The buyer selects products to showcase on the storefront. Step 7. The buyer establishes store policies, including shipping, returns,	Step 2. The system guides the buyer through the initial setup process. Step 4. The system validates the information and updates the storefront preview. Step 6. The system integrates selected products into the storefront layout.

	and customer service details. Step 9. The buyer confirms the storefront setup. Step 11. Use case end.	Step 8. The system stores and displays the configured policies on the storefront. Step 10. The system publishes the storefront, making it visible to Buyers on the online marketplace.
Alternative course of action	If a buyer attempts to create a storefront with a name that already exists, the system provides a notification and prompts the seller to choose a unique name then returns to step 3.	

Table 4- 10 Use Case Description for Manage Product

Use case id	UC-10	
Use case Name	Manage Product	
Actor	Seller	
Description	Efficiently manage and optimize product information, inventory, pricing,	
Precondition	The Seller must be registered and logged into the online marketplace platform. The Seller should have permission to manage products associated with their account. The platform must support the addition and modification of product listings.	
Post condition	The Seller's product listings are updated with the latest information. Inventory levels are adjusted based on product additions or modifications. The changes made by the Seller are reflected on the online marketplace platform.	
Basic course of action	User action	System response

	Step 1. The Seller logs into their account. Step 3. The Seller selects the option to add a new product or selects an existing product to edit. Step 5. The Seller confirms the changes. Step 7. Use case end.	Step 2. The system provides options for managing products, including adding new products or editing existing ones. Step 4. The system prompts the Seller to input details such as product name, description, price, images, and inventory quantity or loads the current product details for the Seller to modify. Step 6. The system saves the updated product information and inventory levels.
Alternative course of action	If there are errors in the product information entered by the seller, the system provides clear error messages and prompts the seller to correct the information.	

Table 4- 11 Use Case Description for Product Promotion

Use case id	UC-11
Use case Name	Product Promotion
Actor	Seller
Description	Utilize various promotional tools and strategies to attract potential buyers.
Precondition	The Seller must be registered and logged into the online marketplace platform. The Seller account should be in good standing with no outstanding violations.

	The platform must support the creation and execution of promotional activities.	
Post condition	The promotional activity is visible to Buyers on the online marketplace. The system tracks and applies discounts or special offers during the promotional period. The Seller's promotional campaign data is updated based on user interactions and sales.	
Basic course of action	User action	System response
	Step 1. The Seller logs into their account. Step 3. The Seller selects the option to create a new promotion on existing product. Step 5. The Seller specifies the promotion type (e.g., percentage discount, flat amount discount) and sets the duration of the promotion. Step 7. The Seller confirms and activates the promotion.	step 2. The system provides options for managing promotions, including creating new promotions or editing existing ones. Step 4. The system prompts the Seller to define the promotion details, including discount type, duration, and eligible products. Step 6. The system validates the promotion details and prepares to apply the promotion during the specified timeframe. Step 8. The system makes the promotion live, applying the specified discounts to eligible products during the defined duration.

Alternative course of action	If there are errors in setting promotion parameters, the system provides clear error messages and prompts the seller to correct the information. If the promotion period expires, the system automatically ends the promotion and updates product prices to their original values.

Table 4- 12 Use Case Description for Chat

Use case id	UC-12	
Use case Name	Chat	
Actor	Users	
Description	Facilitate real-time communication between buyers and sellers as well as delivery personnel through a convenient and reliable chat platform within the online marketplace.	
Precondition	Users must be registered and logged into the online marketplace platform. The Seller should have an active product listing. The platform must support a messaging system for real-time communication.	
Post condition	The conversation history is stored securely for both the Buyer and Seller. Any transactions or agreements made during the chat are reflected in the respective order details.	
Basic course of action	User action	System response
	Step 1. The Buyer navigates to the product listing and clicks on a "Contact Seller" option.	Step 2. The system opens a chat interface, connecting the Buyer with the Seller.

	Step 3. The Buyer and Seller exchange messages to discuss product details, availability, shipping information, or any other relevant queries. Step 5.	Step 4. The system ensures real-time delivery of messages between the two parties.
Alternative course of action	Buyer encounters technical issues with the chat functionality and needs platform support.	

Table 4- 13 Use Case Description for Feedback

Use case id	UC-13	
Use case Name	Feedback	
Actor	Buyer, Seller	
Description	Gather valuable user feedback to enhance the platform experience for both buyers and sellers, foster trust and transparency, and drive continuous improvement.	
Precondition	Platform has a dedicated feedback system integrated with relevant functionalities like orders, listings, and user profiles. Users have clear instructions and incentives to provide feedback.	
Post condition	The buyer has successfully provided feedback on the transaction. The seller has received and, if applicable, responded to the feedback.	
Basic course of action	User action	System response
	Step 1. Users after completing a purchase or interacting with a seller,	Step 2. The system displays a feedback form with rating options

	buyer navigates to the order page or seller profile. Step 3. Buyer rates their experience and provides optional comments on areas like product quality, seller communication, shipping speed, and overall platform satisfaction. Step 5. Sellers reviews the feedback, including ratings and comments.	and open-ended text fields for comments Step 4. The system confirms submission and thanks the buyer for their feedback and notifies the seller of new feedback received on their orders or profile. Step 6. System provides tools for seller to respond to buyer feedback publicly or privately.
Alternative course of action	Buyer requests follow-up or clarification from seller based on feedback.	

Table 4- 14 Use Case Description for Order Product

Use case id	UC-14	
Use case Name	Order product	
Actor	Buyer	
Description	Allows customer to order products	
Precondition	Customer must login into the system	
Post condition	The system allow to order product/s according to customer want	
Basic course of action	User action	System response
	Step 1. The customer select item and click add to cart button. Step 3. Customer click checkout button.	Step 2. The system display cart. Step 4. The system display order form.

	Step 5. The customers fill the required information and submit.	Step 6. The system displays successful message. Step 7. The use case end.
Alternative course of action	If the customer inserts invalid data and missing the field the system displays an error message and the system returns back to step 5.	

Table 4- 15 Use Case Description for Payment

Use case id	UC-15	
Use case Name	Payment	
Actor	Buyer, Payment Processor	
Description	Enable secure and convenient online payments for buyers and sellers, creating a seamless and trust-worthy marketplace experience.	
Precondition	Buyers have created accounts on the platform and linked their preferred payment methods. Sellers have set up their store profiles and chosen their preferred payout methods.	
Post condition	The buyer has successfully completed the payment for the purchased products. The payment personnel have processed the transaction, and the order status is updated accordingly.	
Basic course of action	User action	System response
	Step 1. Buyer selects desired products, adds them to cart, and proceeds to checkout. Step 3. Buyer chooses their preferred payment method	Step 2. System displays available payment options. Step 4. Redirects buyer to the secure payment gateway interface for

	and enters necessary details. Step 5. Payment processor verifies buyer's payment information with their bank or financial institution. Step 7. Sellers receives notification of payment received and can track the payout status. Step 8. Use case end.	authentication and authorization. Step 6. Transfers the received payment to the seller's chosen payout method
Alternative course of action	Payment failure due to insufficient funds, incorrect information, or technical issues and return to step 3. Buyer initiates refund or return request for a purchased item.	

Table 4- 16 Use Case Description fro Rating Products

Use case id	UC-16	
Use case name	Rating Products	
Actor	Buyer	
Description	Facilitate trust and informed purchase decisions for buyers by allowing them to share their experiences with products through ratings and reviews.	
Precondition	Buyers have completed purchases on the platform and their associated orders are eligible for review. Platform has a dedicated product rating and review system integrated with order pages and product listings.	
Post condition	The buyer has successfully provided a rating and review for the purchased product. The product's overall rating is updated on the platform.	
Basic course of action	User action	System response

	Step 1. Buyers after receiving and using a purchased product, buyer navigates to the order page or product listing. Step 3. Rates the product using the star system and writes a detailed review, highlighting positive or negative aspects of their experience. Step 5. Use case end.	Step 2. The system displays a star rating system and open-ended text field for writing a review. Step 4. The confirms submission and thanks the buyer for their feedback.
Alternative course of action	Buyers filter product searches based on average star ratings and keywords from reviews.	

Table 4- 17 Use Case Description for View assigned delivery

Use case id	UC-17
Use case name	View assigned delivery
Actor	Delivery personnel
Description	Efficiently view and manage assigned deliveries within the online marketplace platform.
Precondition	Delivery person has an active account with the platform. Platform has a dedicated web interface for delivery personnel. Location services are enabled on the delivery person's device.
Post condition	The delivery personnel have successfully viewed, managed, and updated the status of their assigned deliveries.

	Delivery status information is updated in real-time on the online marketplace platform.	
Basic course of action	User action	System response
	Step 1. Delivery person launches the platform app or accesses the relevant web interface. Step 3. Use case end.	Step 2. The system displays a dedicated "Assigned Deliveries" section, listing all currently assigned orders for delivery.
Alternative course of action	Delivery personnel encounters unexpected complications during a delivery and needs platform support for assistance. Changes occur in delivery schedules or customer availability requiring route adjustments in real-time.	

Table 4- 18 Use Case Description for Confirm delivery

Use case id	UC-18
Use case name	Confirm delivery
Actor	Buyer
Description	Efficiently and accurately confirm deliveries on the online marketplace platform.
Precondition	Buyer has access order details within the platform app or web interface. Order is at the designated delivery date. Order details are readily available to the buyer with payment detail.

Post condition	The buyer have successfully confirmed the delivered order. The online marketplace platform updates the order status to reflect the successful delivery.	
Basic course of action	User action	System response
	Step 1. The buyer logs into their account on the online marketplace platform. Step 3. Within the order detail, the buyer can view a list of his order. Step 5. The buyer updates the order status to "Delivered" confirmation status. Step 7. Use case end.	Step 2. The system redirects the buyer to their personalized detail. Step 4. The system presents details such as order numbers, delivery date, and the current status of each shipping order. Step 6. The system records the confirmation and updates the order status.
Alternative course of action	If a customer raises concerns about the delivery after confirmation, the system may provide a mechanism for the delivery personnel to address and resolve the issue.	

Table 4- 19 Use Case Description for Logout

Use case id	UC-19
Use case name	Logout
Actor	All Actors
Description	Allow the users to exit from the system.

Precondition	The user must log into the system	
Post condition	The user exit from the system.	
Basic course of action	User action	System response
	Step 1. Users click logout link	Step 2. The system exits from the current page. Step 3. The use case end

4.1.1.3. Use Case Scenario

A scenario serves as a practical illustration of how an individual or organization interacts with a system, delineating the various activities, steps, and events that unfold during the interaction. It provides a comprehensive explanation of the system's functionalities and details the engagement between an actor and the user interface. Consequently, we outlined each use case for the proposed system in the following sections.

1. **Actor: Admin, use case: Manage Product Category**

Goal: Manage the organization and information of product categories on the platform.

Preconditions:

- ✓ The admin user has logged into the system with appropriate permissions.
- ✓ The product category management interface is accessible.

Main Flow:

1. Browse Categories: The admin user accesses the "Product Categories" page.
2. Select Category: The admin user selects a specific category from the list or uses search or filter options to locate the desired category.
3. View Category Details: The current category's name, description, category (if applicable), associated products, and other relevant attributes are displayed.
4. Manage Category: Depending on the desired action, the admin user can:
    - ✓ Edit: Modify the category's name, description, parent category, or other attributes.
    - ✓ Add: Create a new category by providing its name, description, parent category (if applicable), and other relevant attributes.

- ✓ Deactivate: Hide the category if it is no longer needed. Confirmation may be required for this action.

5. Save Changes: The admin user submits the changes made to the category.
6. Confirmation: The system provides visual or textual confirmation of successful updates or additions. In case of deletions, appropriate warnings or confirmations may be required.

Alternative Flows:

- ✓ If the edited category name already exists, the system should generate an error message and prevent saving the changes.
- ✓ If the admin attempts to delete a category with associated products, the system should warn the user and offer options to reassign the products or cancel the deletion.
- ✓ If an unexpected error occurs, the system should display a clear error message and log the incident for further investigation.

Postconditions:

- ✓ The edited or added category information is reflected in the system and becomes accessible to relevant users.
- ✓ The deleted category is removed from the system along with any associated metadata.
- ✓ The admin user can continue managing other product categories or perform other tasks within the system.

2. Actor: Admin, use case: Manage user Account

Goal: Manage user accounts on the platform, including information and access privileges.

Preconditions:

- ✓ The admin user has logged into the system with appropriate permissions.
- ✓ The user account management interface is accessible.

Main Flow:

1. Browse Users**:** The admin user opens the "Users" page.
2. Search or Filter Users: The admin user can search or filter users based on various criteria like username, email, role, status, etc.
3. Select User: The admin user selects a specific user account from the list.

4. View User Details: The current user's profile information, including username, email, role, registration date, last login, and other relevant attributes, is displayed.
5. Manage User Account: Depending on the desired action, the admin user can:
    - ✓ Edit**:** Modify the user's name, email, password (optionally require user confirmation for password changes), profile picture, or other modifiable attributes.
    - ✓ Change Role**:** Assign a different role to the user that defines their access privileges within the system.
    - ✓ Active/deactivate Account: Temporarily active or activate a user's account access.
6. Save Changes**:** The admin user submits the changes made to the user account.
7. Confirmation: The system provides visual or textual confirmation of successful updates, actions, or deletions.

Alternative Flows:

- ✓ If the edited username or email already exists, the system should generate an error message and prevent saving the changes.
- ✓ If the admin attempts to delete a user account with associated data or ongoing activities, the system should warn the user and offer options to handle such dependencies before deletion.
- ✓ If an unexpected error occurs, the system should display a clear error message and log the incident for further investigation.

Postconditions:

- ✓ The edited user account information is reflected in the system and becomes accessible to the user (except password, if changed).
- ✓ The deleted user account is removed from the system along with any associated data (confirm deletion policy with user base).
- ✓ The admin user can continue managing other user accounts or perform other tasks within the system.

3. Actor: Admin, use case: Approve seller

Goal: Review and approve new seller accounts to ensure platform standards and quality.

Preconditions:

- ✓ The admin user has logged into the system with appropriate permissions.
- ✓ The "Sellers" page or section for pending review is accessible.

Main Flow:

1. Review Pending Sellers: The admin user opens the "Pending Sellers" list or relevant section.
2. Filter or Search: The admin can filter or search for specific sellers based on various criteria like application date, category, name, etc.
3. Select Seller: The admin chooses a specific seller account for review.
4. View Seller Profile: The seller's profile information, including:
    - ✓ Application details and registration date
    - ✓ Business information and product categories
    - ✓ Identity verification documents and credentials
    - ✓ Seller's profile picture and description
    - ✓ User reviews or past performance (if available)
5. Review and Verify: The admin thoroughly review seller's profile, focusing on:
    - ✓ Completeness and accuracy of information
    - ✓ Legitimacy of business and products
    - ✓ Compliance with platform policies and terms
    - ✓ Potential fraud or risk indicators
    - ✓ Verification of identity documents against official sources (if applicable)
6. Decision**:** Based on the review, the admin can:
    - ✓ Approve: Grant the seller account access to the platform, notifying the seller via email or system notification.
    - ✓ Request More Information: If information is missing or unclear, the admin can request additional details from the seller before making a decision.
    - ✓ Reject: If the seller does not meet platform standards or raises concerns, the admin can reject the application, providing clear feedback and rejection reason to the seller.
7. Save Decision: The admin records the decision (approve, request information, reject) within the system.

Alternative Flows:

- ✓ If the verification process encounters technical issues, the admin should document the issue and follow up with the seller for alternative verification methods.
- ✓ If the admin requires further clarification or assistance with the review, they can consult with other team members or platform policies for guidance.
- ✓ In case of an appeal from a rejected seller, the admin should establish a procedure for reviewing the decision and potentially reconsidering based on new information or clarification.

Postconditions:

- ✓ Approved sellers gain access to the platform and can begin listing their products or services.
- ✓ Rejected sellers receive clear notification and explanation for the decision, with potential steps for addressing concerns and reapplying.
- ✓ Requested information is gathered from the seller to facilitate further review and decision-making.
- ✓ The admin user can continue reviewing other pending seller accounts or perform other tasks within the system.

4. **Actor: Admin, use case: Manage orders**

Goal: Monitor and manage customer orders on the platform, ensuring smooth fulfilment and customer satisfaction.

Preconditions:

- ✓ The admin user has logged into the system with appropriate permissions.
- ✓ The "Manage orders" section or page is accessible.

Main Flow:

1. Browse Orders: The admin opens the order list.
2. Filter or Search: The admin can filter or search for specific orders based on various criteria like order date, customer name, order status, product category, etc.
3. Select Order: The admin chooses a specific order for review and management.
4. View Order Details: The detailed order information is displayed, including:
    - ✓ Customer information and contact details
    - ✓ Ordered products and their quantities
    - ✓ Shipping address and delivery information

- ✓ Payment method and order amount
- ✓ Current order status and history of updates
- ✓ Internal notes or communications related to the order

5. Manage Order: Depending on the situation, the admin can:
    - ✓ Update Order Status: Change the order status to reflect progress, such as "processing," "completed" etc.
6. Save Changes: The admin submits the made changes to the order details or status.
7. Confirmation: The system provides visual or textual confirmation of successful updates or actions.

Alternative Flows:

- If modifying certain order status, the system should prevent unauthorized changes and guide the admin towards acceptable actions.
- If an unexpected error occurs during order management, the system should display a clear error message and log the incident for further investigation.

Postconditions:

- ✓ The updated order details and status are reflected in the system and may trigger automated notifications or actions based on the changes.
- ✓ The customer receives updates or notifications regarding their order, depending on the admin's actions.
- ✓ The admin user can continue managing other orders or perform other tasks within the system.

5. **Actor:** Admin, seller, use case: Generating reports

Goal: Access and generate reports to gain insights into platform activity, user behaviour, and performance metrics.

Preconditions:

- ✓ The admins or sellers have logged into the system with appropriate permissions.
- ✓ The "Generating reports" section or tool is accessible.

Main Flow:

1. Access Reports: The admins or sellers open the report selection interface.
2. Choose Report: The admins or sellers select the specific report they want to generate.

3. Specify Parameters: Depending on the report's complexity, the admins or sellers may be able to define various parameters:
    - ✓ Output Format: Select the desired format for the generated report (e.g., PDF, Excel, CSV, interactive dashboard).
4. Generate Report: The admins or sellers submits the chosen parameters and initiates the report generation process.
5. Progress and Notification: The system displays the report generation progress and informs the admins or sellers when it's completed.
6. Download or View Report: The admins or sellers can either download the report file (depending on the chosen format) or directly view the report within the system if it offers interactive dashboards or visualizations.

Alternative Flows:

- ✓ If the chosen report parameters are invalid or incompatible, the system should clearly highlight the errors and guide the admin towards accurate selection.
- ✓ If the report generation process encounters unexpected issues or takes longer than expected, the system should notify the admin and provide an estimated completion timeframe.
- ✓ For complex reports with extensive data sets, the system might offer options for scheduling report generation and automated delivery via email or internal notifications.

Postconditions:

- ✓ The generated report is available for download or viewing, depending on the admin's preference.
- ✓ The report contains comprehensive data insights, charts, or visualizations based on the chosen parameters and timeframe.
- ✓ The admin can analyse the report to gain valuable insights into user behaviour, platform performance, and areas for improvement.

6. **Actor: User (buyer or seller), use case: Registration**

Goal: Register on the platform to access and purchase products or services.

Preconditions:

- ✓ The customer has access to the platform and can navigate to the registration page.
- ✓ They have a valid email address.

Main Flow:

1. Access Registration**:** The customer finds and opens the "Create Account" or "Sign Up" page or section.
2. Review Requirements**:** The platform may display basic signup conditions, privacy policy, or age limitations before the registration form.
3. Complete Registration Form**:** The customer fills out the required fields, which may include:
    - ✓ Personal Information**:** Name, email address, password
    - ✓ Optional Information**:** Phone number, billing address, shipping address
4. Verify Email Address**:** After submitting the form, the system sends a verification email with a unique link.
5. Click Verification Link**:** The customer opens their email and clicks on the link to confirm their email address.
6. Account Activation**:** Upon successful verification, the system activates the customer account and may notify them via email or system notification.

Alternative Flows:

- ✓ If the email address is invalid or already in use, the system should display an error message and prompt the customer to correct the information.
- ✓ If the verification email is not received within a reasonable timeframe, the customer should be able to request a resend or choose alternative verification methods.
- ✓ In case of technical issues during registration, the platform should provide clear error messages and offer alternative ways to contact support for assistance.

Postconditions:

- ✓ The customer's account is activated and they can immediately start browsing products, adding items to their cart, and completing purchases.
- ✓ The platform may offer them a welcome message, tutorial resources, or personalized recommendations based on their preferences.

7. **Actor: Seller (Initially Registered as Buyer), use case: Storefront Creation**

Goal: Create a storefront and begin selling products on the platform.

Preconditions:

- ✓ The user has already registered as a buyer on the platform.
- ✓ They have a valid account and are logged in.

Main Flow:

1. Navigate to Storefront Creation: The seller finds and opens the "Create Storefront" or "Become a Seller" page or section within their buyer account.
2. Review Requirements**:** The platform may display information about seller requirements, fees, and terms of service before prompting the user to proceed.
3. Convert Buyer Account to Seller: The seller confirms their intention to convert their existing buyer account to a seller account and may need to accept additional terms or agreements specific to sellers.
4. Complete Storefront Information: The seller fills out details specific to their storefront, such as:
    - ✓ Storefront Name: A unique and recognizable name for their online store.
    - ✓ Product Categories: Select categories relevant to the products they plan to sell.
    - ✓ Business Description**:** Tell buyers about their brand, values, and what makes their products unique.
    - ✓ Optional Information**:** Add logos, banners, contact details, and other branding elements.
5. Submit Storefront Information: The seller submits the completed storefront information for review and approval.
6. Storefront Review and Approval: The platform reviews the storefront details and may require additional information or modifications before granting approval. The seller may be notified via email or system notification about the status of their application.
7. Start Selling: Upon successful approval, the seller's storefront becomes active, and they can begin listing and managing their products within their seller account.

Alternative Flows:

- ✓ If the seller's existing buyer account lacks necessary information or doesn't meet seller requirements, the platform should guide them towards completing their buyer profile or provide alternate pathways to register as a seller.

- ✓ If the storefront information submitted for review is incomplete or inconsistent with platform policies, the system should provide clear feedback and instructions for corrections before approval.
- ✓ In case of technical issues during storefront creation, the platform should provide error messages and alternative ways to contact support for assistance.

Postconditions:

- ✓ The seller has an active storefront and can access seller-specific features like product listing, order management, and seller analytics.
- ✓ The platform may offer welcome resources, tutorial guides, or recommendations to help the seller start selling effectively.

8. **Actor: Seller, use case: Manage product**

Goal: Create and manage their product listings on the platform to showcase and sell their products.

Preconditions:

- ✓ The seller has logged in to their active seller account.
- ✓ They have access to the "Manage Products" section or page within their account.

Main Flow:

1. Create New Product:
    - ✓ Product Details**:** Fill out information like product name, description, category, brand (if applicable), and key features.
2. Save and Publish: Submit or post the completed product information
3. Modify Existing Product:
    - ✓ Edit any product details, including name, description, price, images, or inventory levels.
    - ✓ Add or remove variations or change promotional settings.
    - ✓ Deactivate a product when it's no longer available for sale.

Alternative Flows:

- ✓ If the seller enters invalid information like negative prices or incorrect stock levels, the system should display clear error messages and guide them towards corrections.

- ✓ In case of technical issues during product creation or management, the platform should offer error messages and alternative ways to contact support for assistance.

Postconditions:

- ✓ The newly created product appears on the platform potentially visible in relevant search results and browsing categories.
- ✓ Existing products reflect the updated information or changes made by the seller.
- ✓ The seller can continue adding new products, editing existing ones, and monitoring their product performance within their account.

9. **Actor: User (Buyer or Seller): use case: Chat**

Goal: Communicate directly with another user on the platform through a live chat interface.

Preconditions:

- ✓ The user has logged in to their active account.
- ✓ They have access to the "Chat" section or feature within their account.
- ✓ They have a list of available buyers or sellers to choose from for initiating a chat.

Main Flow:

1. Navigate to Chat: The user opens the "Chat" section or icon within their account.
2. Browse Contacts: The platform displays a list of users they can initiate a chat with, potentially filtered by buyer/seller role, product categories, recent interactions, or other relevant criteria.
3. Select Buyer: The user chooses the specific buyer they want to chat with by clicking on their name or profile picture.
4. Open Chat Window: A dedicated chat window opens for the selected buyer, displaying any previous conversation history if available.
5. Send Message: The user types their message into the chat box and sends it to the buyer.
6. Receive Message: The buyer receives the message notification and can view it within the chat window when they become online or access the chat feature.

Alternative Flows:

- ✓ In case of technical issues during chat initiation or messaging, the platform should display error messages and provide alternative ways to contact support for assistance.

Postconditions:

- ✓ The sent message appears in the chat window for both the user and the buyer (when they become online).
- ✓ Both users can continue sending and receiving messages in real-time or at their convenience, maintaining an ongoing chat conversation.
- ✓ The chat history may be stored for future reference or review within the user's account settings.

10. Actor: Seller, use case: View order

Goal: View order's details to understand its status, customer information, and other relevant information without modifying anything.

Preconditions:

- ✓ The seller has logged in to their active seller account.
- ✓ They have access to the "View Orders" section within their account dashboard.

Main Flow:

1. Order Overview: The seller views a list of all placed orders, potentially filtered by date, status, product category, or buyer name.
2. Select Order: The seller chooses a specific order to view by clicking on its details or order number.
3. View Order Details: The complete order information is displayed, including:
    - ✓ Buyer information and contact details
    - ✓ Ordered products and their quantities
    - ✓ Price, shipping address, and payment information
    - ✓ Current order status

Alternative Flows:

- ✓ In case of technical issues during order viewing, the platform should display error messages and log the incident for further investigation.

Postconditions:

- ✓ The seller gains a thorough understanding of the specific order details without modifying anything.

- ✓ The seller can continue browsing other orders or perform other tasks within their account.

11. Actor: Seller, use case: Product promotion

Goal: Create a promotional offering for their product(s) to attract buyers and increase sales.

Preconditions:

- ✓ The seller has logged in to their active seller account.
- ✓ They have access to the "Product Promotion" section within their account dashboard.
- ✓ They have active products listed on the platform.

Main Flow:

1. **Create Promotion:** The seller chooses to create a new product promotion.
2. **Set Promotion Details:**
    - ✓ Name: Define a clear and enticing name for the promotion.
    - ✓ Description: Briefly explain the promotion, its benefits, and any applicable terms and conditions.
    - ✓ Discount: Specify the discount percentage, fixed amount, or any other type of discount offered.
    - ✓ Target Products: Select the specific product(s) or product categories the promotion applies to.
    - ✓ Duration: Set the start and end date or timeframe for the promotion's validity.
    - ✓ Optional Settings: Configure additional parameters like minimum purchase amount, promo code usage, or banner image for the promotion.
3. Preview and Submit: The seller review created promotion details and previews how it might appear on the platform.
4. Send for Approval: Once satisfied, the seller submits the promotion for admin review and approval.
5. Admin Review: The platform notifies the designated admin(s) about the submitted promotion. They can access the details, preview its presentation, and:
    - ✓ Approve: If the promotion complies with platform policies and is deemed appropriate, the admin approves it for publication.
    - ✓ Reject: If the promotion has issues or doesn't meet platform guidelines, the admin provides feedback and suggests revisions for the seller to address.

Postconditions:

- ✓ If approved, the promotion becomes active on the platform at the specified start date.
- ✓ The discounted product(s) appear on dedicated promotion pages, search results, and potentially other platform visibility channels.
- ✓ Buyers viewing the product information or promotion pages can see the applicable discount and promotion details.

Alternative Flows:

- ✓ If the seller encounters technical issues while creating or submitting the promotion, the platform should display error messages and offer alternative ways to contact support for assistance.
- ✓ If the admin rejects the promotion, they should provide clear feedback and instructions for the seller to make necessary edits and resubmit the promotion.

12. Actor: User (seller or buyer), use case: Feedback

Goal: Share their thoughts and suggestions for improvement regarding the platform's features, functionalities, or overall experience.

Preconditions:

- ✓ The user has logged in to their active account.
- ✓ They have access to the "Provide Feedback" section or feature within their account dashboard.

Main Flow:

1. Access Feedback Form: The user opens the dedicated feedback form or section within their account.
2. Write Feedback: The user writes their detailed feedback message, including:
    - ✓ Description of the issue or suggestion: Clearly explain their experience, concerns, or ideas for improvement.
    - ✓ Impact or context: Describe how the issue affects them or why their suggestion would be beneficial.
    - ✓ Optional details: Add relevant screenshots, data points, or specific examples to clarify their feedback.

3. Submit Feedback: The user submits their completed feedback message after reviewing it for clarity and accuracy.

Alternative Flows:

- ✓ If the feedback form encounters technical issues, the platform should display error messages and offer alternative ways to submit feedback, such as email or support chat.

13. Actor: Buyer, use case: View Product

Goal: Learn about a specific product they're interested and making a purchase decision.

Preconditions:

- The buyer has navigated to the product page, either through search, browsing categories, clicking on a recommendation, or a link from another product.
- The product page is available and accessible to the buyer.

Main Flow:

1. Product Overview: The buyer sees the product's name, main image, and current price prominently displayed.
2. Details Exploration: The buyer can:
    - ✓ Read the product description: Gain deeper understanding of the product's features, benefits, and intended use.
    - ✓ Explore detailed attributes: View available sizes, colours, materials, specifications, technical details, warranty information, shipping options, and any other relevant features.
3. Decision-Making Options: Depending on the platform and their interests, the buyer can:
    - ✓ Add the product to their cart or Wishlist for later purchase or consideration.
    - ✓ Ask the seller a question about the product to clarify any doubts.
    - ✓ Compare the product with similar offerings on the platform based on price, or other features,

Alternative Flows:

- ✓ If the product page is unavailable or contains errors, the platform should display informative messages and guide the buyer back to functional sections.

- ✓ The platform might offer personalized recommendations for similar products based on the buyer's browsing history or current product interest.

Postconditions:

- ✓ The buyer gains a comprehensive understanding of the product and its features, reviews, and potential limitations.
- ✓ Based on their exploration, the buyer can make an informed decision about purchasing the product, adding it to their Wishlist for later, or searching for alternatives.
- ✓ The platform captures the buyer's interaction data to personalize future recommendations and improve the overall product viewing experience.

14. Actor: Buyer, use case: Make payment

Goal: Securely complete the payment for their chosen product(s) and finalize their order.

Preconditions:

- ✓ The buyer has added the desired product(s) to their cart and proceeded to the checkout page.
- ✓ The checkout page is accessible and displays the order summary with product details, quantities, and prices.

Main Flow:

1. Choose Shipping Address: The buyer selects or verifies their preferred shipping address from saved addresses or enters a new one.
2. Select Payment Method: The buyer chooses their preferred payment method if there from available options.
3. Enter Payment Details: Depending on the chosen method, the buyer enters their payment information securely, such as card details, account credentials, or cash on delivery instructions.
4. Review and Confirm Order: The buyer reviews their order summary, including product details, shipping address, payment method, and total amount. They can adjust quantities or modify chosen options if needed.
5. Secure Payment Processing: The buyer confirms their payment. The platform securely processes the payment with appropriate security protocols and data encryption.
6. Confirmation and Post-Payment Actions:

- ✓ Order Confirmation: The buyer receives an immediate order confirmation email with details like order number, estimated delivery date, and tracking information (if available).
- ✓ Account Access: If a registered user, the buyer can access their order history and details within their account dashboard.

Alternative Flows:

- ✓ If the chosen payment method fails or encounters technical issues, the platform should display error messages, offer alternative payment options, or guide the buyer to contact support for assistance.
- ✓ If the buyer needs to edit their order details or address after confirming payment, the platform should offer options to adjust the order within specific limitations or contact customer support for assistance.

Postconditions:

- ✓ The buyer's payment is successfully processed, and their order is confirmed with estimated delivery and necessary details.
- ✓ The buyer receives immediate confirmation messages and has access to relevant order information for future reference.
- ✓ The platform records the order details and initiates fulfilment processes to deliver the purchased product(s) to the buyer.

15. Actor: Buyer, use case: Rating Product

Goal: Share their experience with a purchased product by leaving a rating and writing a detailed review.

Preconditions:

- ✓ The buyer has accessed the specific product page they want to review, either from their order history, product search, or a dedicated review section.
- ✓ The buyer meets the eligibility criteria for leaving a review on the platform, such as having a verified purchase or minimum account activity.

Main Flow:

1. Access Review Form: The buyer clicks on a "Write a Review" button or link prominently displayed on the product page.

2. Rate the Product: The buyer selects their overall rating using the provided rating system (e.g., 4 out of 5 stars).
3. Write a Review: The buyer writes a detailed review in a dedicated editor, potentially including:
    - ✓ Their experience with the product's features and performance.
    - ✓ Pros and cons they encountered.
    - ✓ Value for money or comparison with similar products.
4. Submit Review: The buyer reviews their rating and review before submitting it for publication.
5. Moderation and Publication: Depending on the platform's policies:
    - ✓ The review is automatically published after meeting basic criteria.
6. Post-Review Actions:
    - ✓ The buyer can see their submitted review on the product page alongside other user reviews.
    - ✓ They might receive notifications for replies or reactions to their review from other users or the seller.

Alternative Flows:

- ✓ If the buyer encounters technical issues with the review form or submission process, the platform should display clear error messages and offer alternative ways to contact support for assistance.
- ✓ The platform might provide guidelines or pre-defined prompts to encourage informative and helpful reviews.
- ✓ Buyers might be able to edit or update their submitted reviews within a specific timeframe before publication or after moderation feedback.

Postconditions:

- ✓ The buyer's rating and review are published on the product page, contributing to valuable user feedback and informing future purchase decisions.
- ✓ The buyer receives appropriate notifications and has access to further interaction options related to their review.
- ✓ The platform gathers valuable user data on product perceptions and utilizes it for product improvement, seller evaluation, and overall platform optimization.

16. **Actor:** Delivery Personnel, use case: view assigned delivery

Goal: Efficiently access and manage information about their assigned deliveries for the current day or scheduled time frame.

Preconditions:

- ✓ The delivery personnel have logged in to their active account
- ✓ The app has access to location services and internet connectivity.
- ✓ The app displays the current date and time.

Main Flow:

1. View Assigned Deliveries: The app automatically displays a list of deliveries assigned to the personnel for the current day or chosen schedule (e.g., next 2 hours, afternoon shift).
2. Delivery Details: Each delivery in the list shows relevant information, including:
    - ✓ Customer name and address (including map view and navigation options).
    - ✓ Order details (items, quantity, estimated weight/size).
    - ✓ Delivery time window or specific scheduled time.
    - ✓ Contact information for the customer or seller if needed.
3. Mark Delivery Progress: The personnel can update the status of each delivery as they progress:
    - ✓ "Picked up" when physically collected from the seller.
    - ✓ "On the way" when traveling towards the customer's location.
    - ✓ "Delivered" when successfully delivered to the customer.
    - ✓ "Attempted delivery" if unsuccessful, with options to record the reason (e.g., customer unavailable, wrong address).

Alternative Flows:

- ✓ If GPS location services are unavailable, the personnel can manually enter their location or request assistance from the platform.
- ✓ If a delivery cannot be completed due to unforeseen circumstances, the personnel can report the issue and follow the platform's established protocols for rescheduling or resolution.

Postconditions:

- ✓ The delivery personnel efficiently access and manages their assigned deliveries.

- ✓ Real-time updates on delivery status provide transparency and improve overall delivery efficiency.
- ✓ The platform gathers data on delivery progress and personnel performance for optimization and feedback purposes.

4.2. Object Model

An object model represents a logical interface for software or systems, constructed using object-oriented techniques. It facilitates the development of an architectural model for software or systems before the actual programming phase. In the envisioned system, we employ both class diagrams and data dictionaries as essential components.

4.2.1. Class Diagram

In software engineering, a class diagram is a type of static structural diagram that illustrates the attributes, actions (or methods), and relationships of the classes within a system, providing a blueprint for its structural definition.

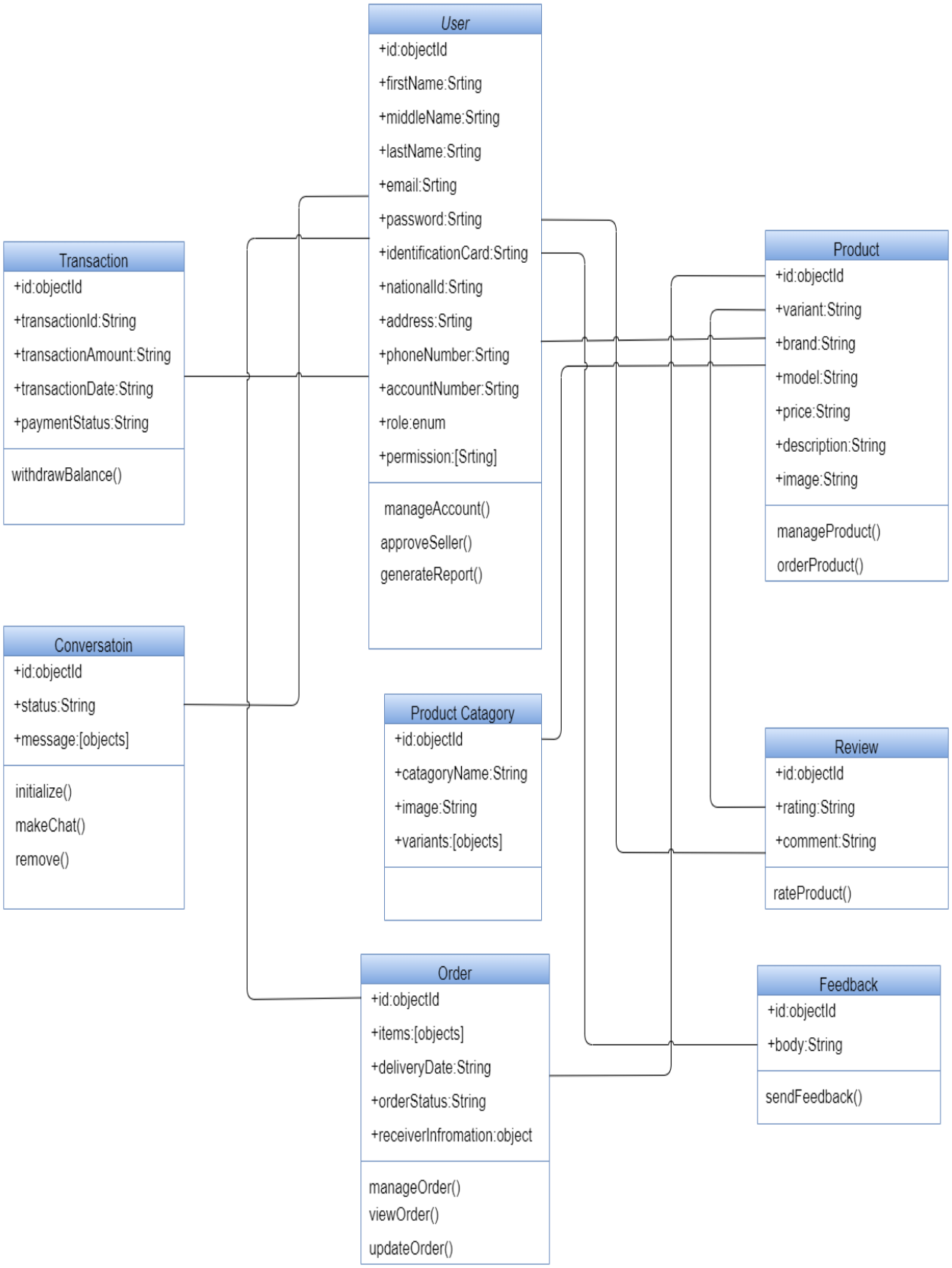


Figure 4- 2 Class Diagram

4.2.2. Data Dictionary

Table 4- 20 Data Dictionary for User

Field Name	Data Type	Description
First Name	String	First name of an individual
Middle Name	String	middle name of an individual
Last Name	String	Surname of an individual
Id	Object id	Unique identifier for the record or entity
Email	String	Email address associated with the individual
Address	String	Physical address of the individual
Password	String	Password associated with the individual's account
Identification card	String	Identification card information
National Id	String	National identification number associated with the individual
Phone Number	String	Contact phone number of the individual
Account Number	String	Account number associated with the individual
Role	Enum	Position of the individual within a system
Permission	String	Permissions granted to the individual within a system

Table 4- 21 Data Dictionary for Product Category

Field Name	Data Type	Description
Id	Object id	A unique identifier for the product
Category Name	String	Name of the product category to which the product belongs
Image	String	Image associated with the product
variants	[Object]	The type of the product

Table 4- 22 Data Dictionary for Transaction

Field Name	Data Type	Description
Id	Object id	A unique identifier for the transaction record
Account Holder	String	Account Holder full name
Transaction Amount	String	The monetary value or amount involved in the transaction
Bank Info	String	Information bank which chosen for transaction
Transaction Status	String	The status of the payment associated with the transaction
Remark	String	User comment on committed transaction

Table 4- 23 Data Dictionary for Product

Field Name	Data Type	Description
Id	Object id	A unique identifier for the product record
variants	String	A product type selected from category
Brand	String	The brand of the product
Model	String	Model name or number associated with the product
Price	String	The price of the product
Description	String	A brief text providing additional information or details about the product
Image	String	The image associated with the product

Table 4- 24 Data Dictionary for Order

Field Name	Data Type	Description
Id	Object id	A unique identifier for the order record

Receiver Information	Objects	Receiver information in detail
Delivery Date	String	The expected delivery date
Order Status	String	The status of the order
Items	[Objects]	The list of products in order

Table 4- 25 Data Dictionary for Review

Field Name	Data Type	Description
Id	Object id	A unique identifier for review record
Rating	String	The numerical or qualitative rating given by a user as feedback
Comment	String	Additional text provided by the user as comments or feedback

Table 4- 26 Data Dictionary for Conversation

Field Name	Data Type	Description
Id	Object id	A unique identifier for the record
Message	[objects]	The message associated with the record
Status	String	The status associated with the record

Table 4- 27 Data Dictionary for Feedback

Field Name	Data Type	Description
Id	Object Id	A unique identifier for the record
body	String	Feedback provided by the user

4.3. Dynamic Model

In this section, we delineated the dynamic characteristics of the system using sequence diagrams, activity diagrams, and state chart diagrams.

4.3.1. Sequence Diagram

In this section, we visually depict the sequential logic, capturing the temporal ordering of messages, with a focus on the business process perspective. The sequence diagram used to illustrate how the business model is executed. It showcases the interactions among objects over time, representing the objects and schemas involved in a given scenario and the sequence of messages exchanged between them to fulfill the scenario's functionality. The sequence diagrams for each use case in the proposed system are outlined below.

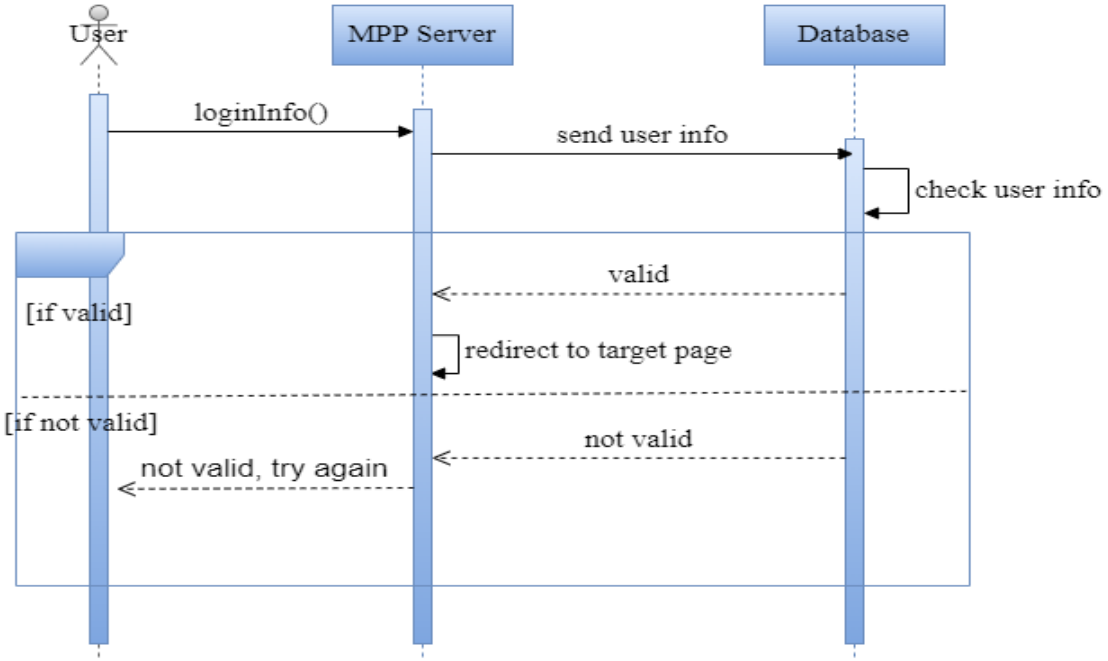


Figure 4- 3 Sequence Diagram for Login

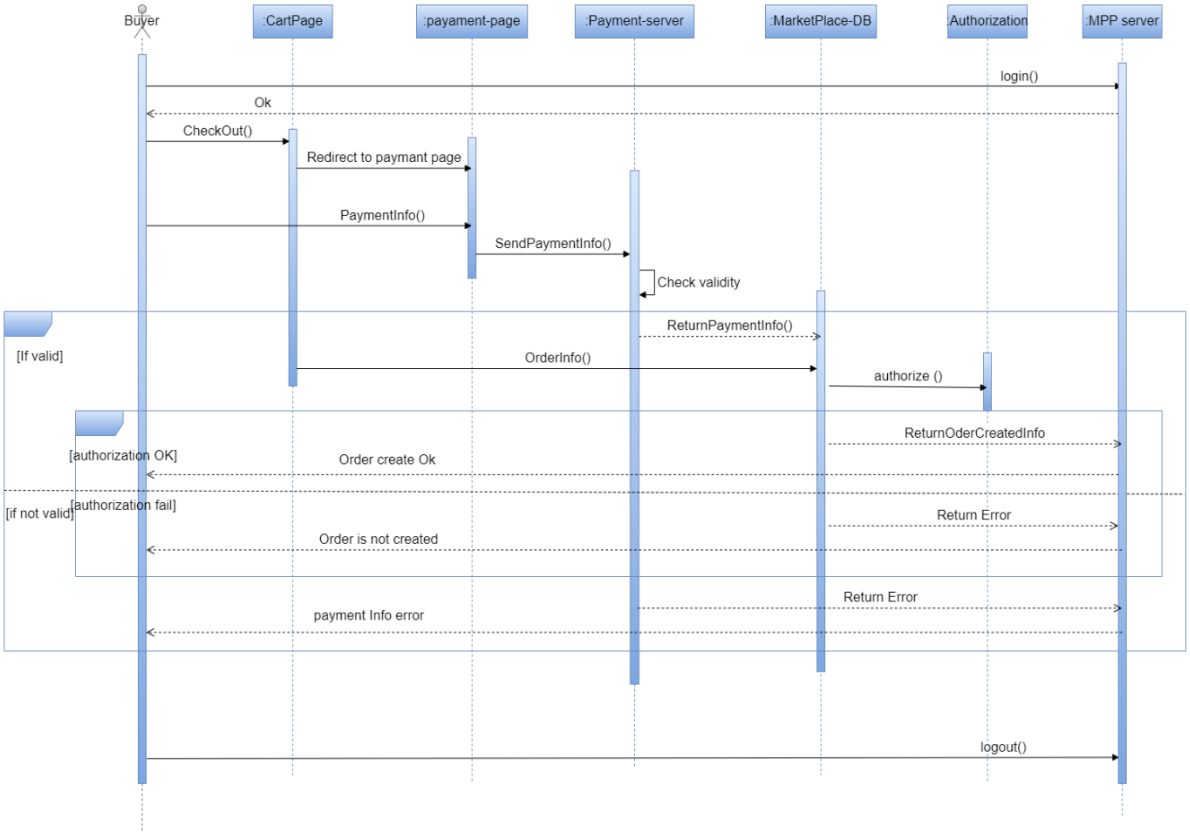


Figure 4- 4 Sequence Diagram for Order Product

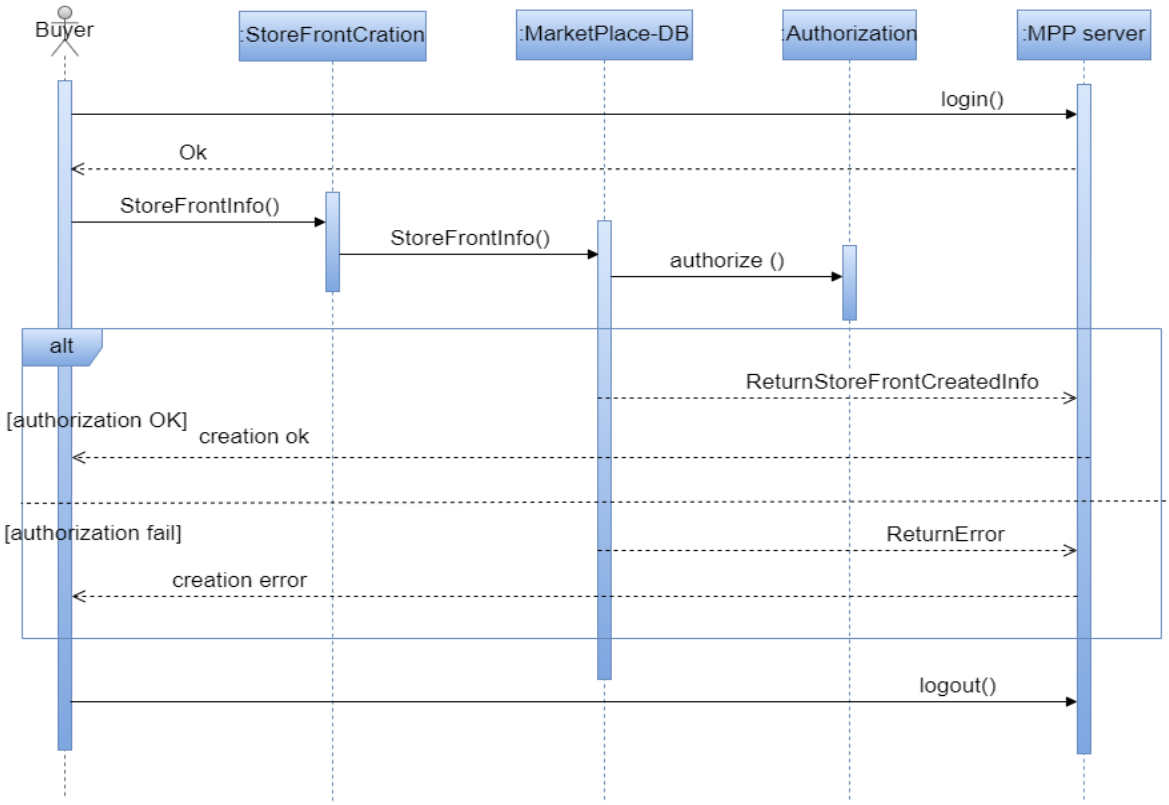


Figure 4- 5 Sequence Diagram for Storefront Creation

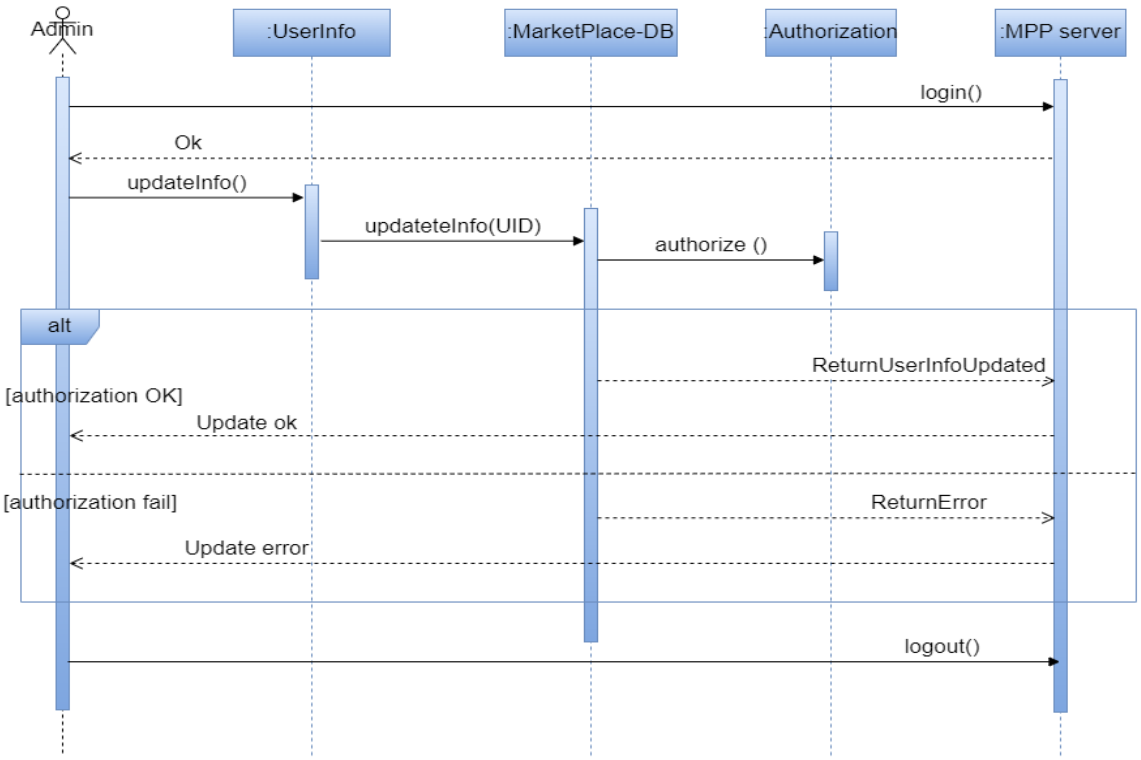


Figure 4- 6 Sequence Diagram for Update User

4.3.2. Activity Diagram

Below figure shows in the system, the control flow is drawn from one operation to another. This

flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc. The activity diagram shows: -

- ✓ Draw activity of the system
- ✓ Describe the sequence from one activity to another
- ✓ Describe the parallel, branched and concurrent flow of the system

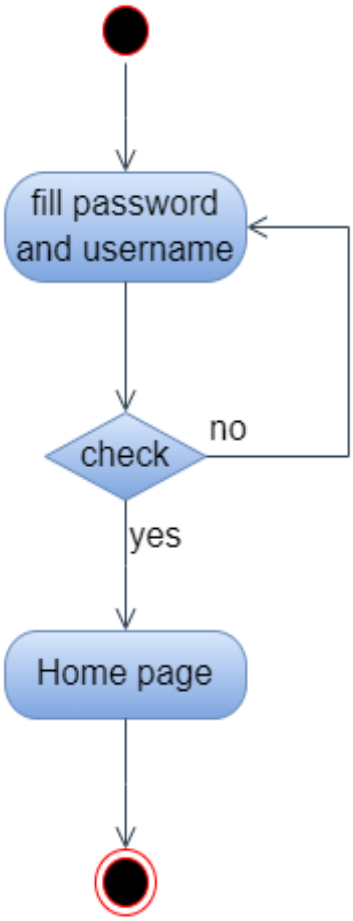


Figure 4- 7 Activity Diagram for Login

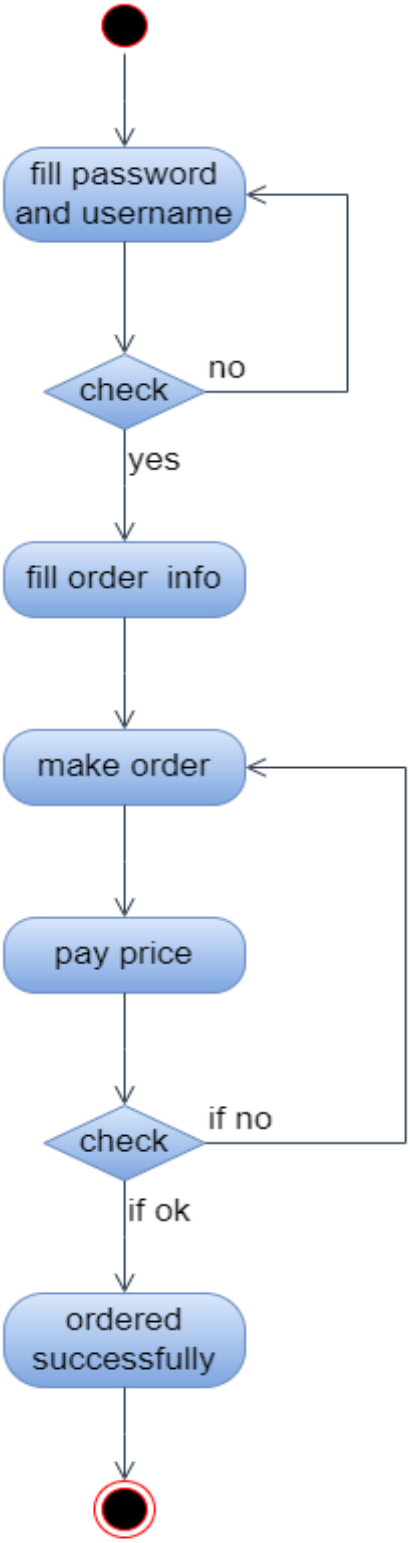


Figure 4- 8 Activity Diagram for Order Product

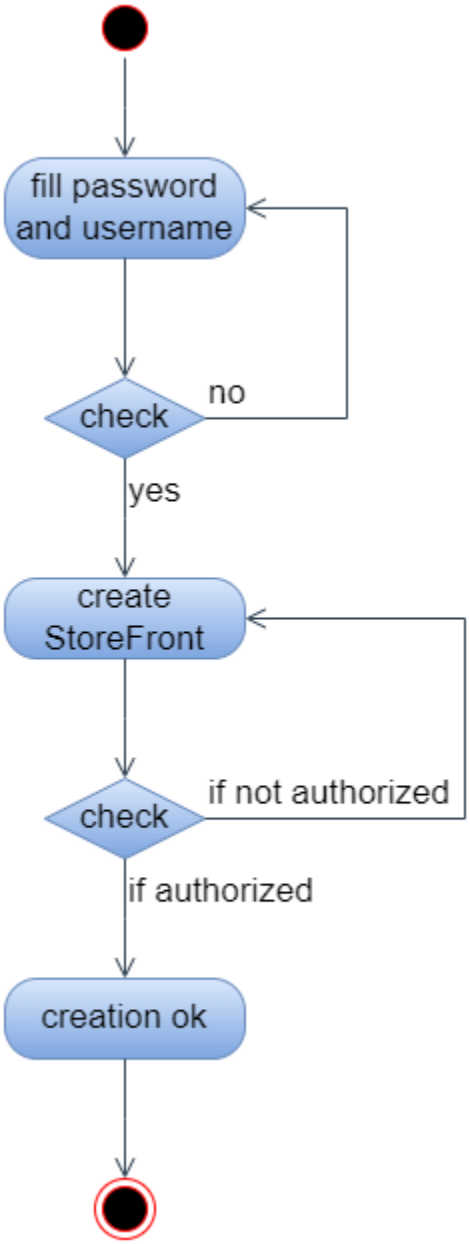


Figure 4- 9 Activity Diagram for Storefront Creation

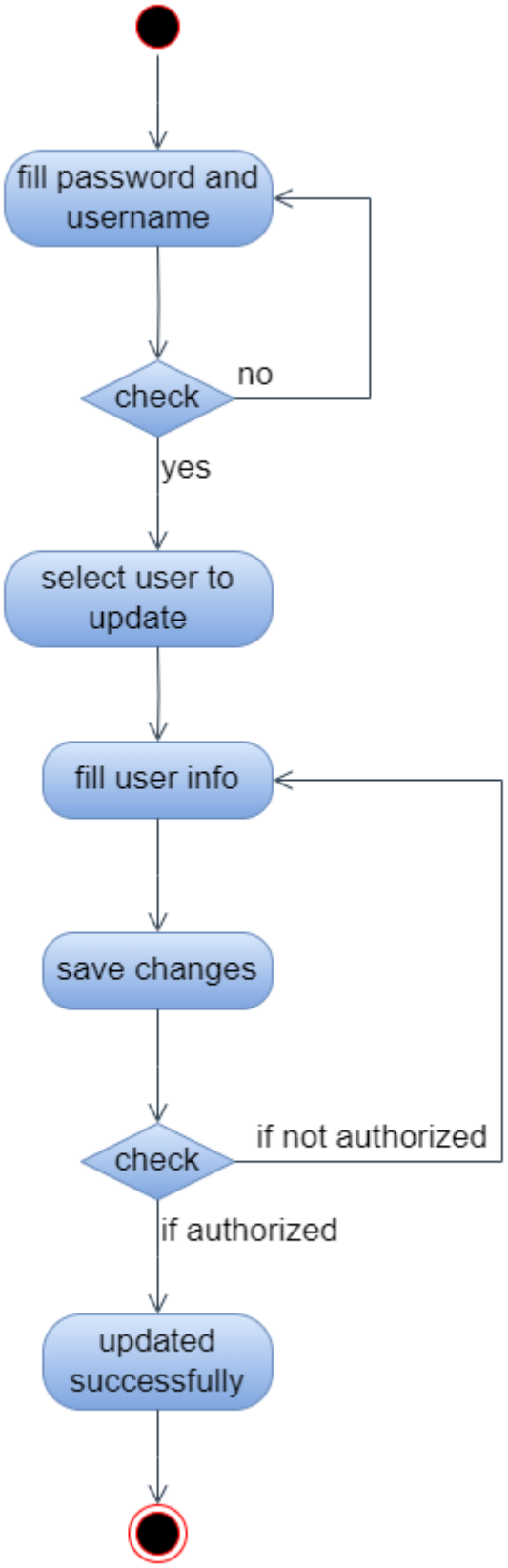


Figure 4- 10 Activity Diagram for User Update

4.3.3. State Chart Diagram

In the system dynamic model used to model the dynamic behavior of a class in response to time and changing external stimuli. As shown below

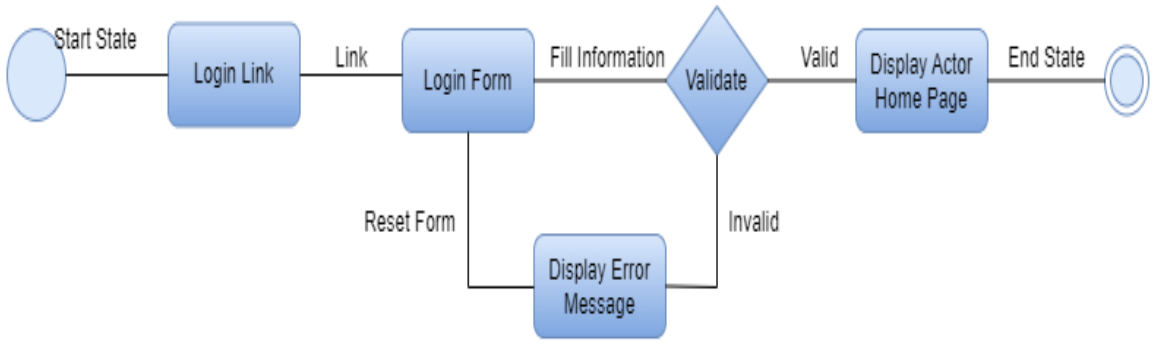


Figure 4- 11 State Chart Diagram for Login

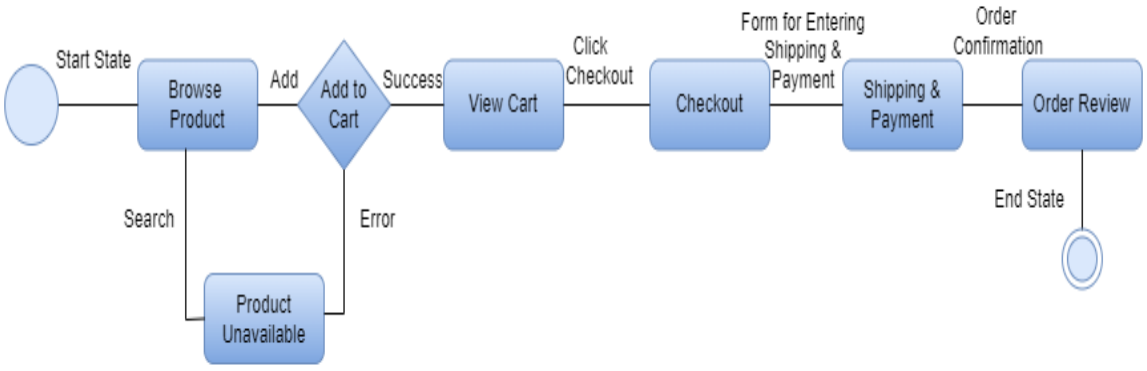


Figure 4- 12 State Chart Diagram for Order Product

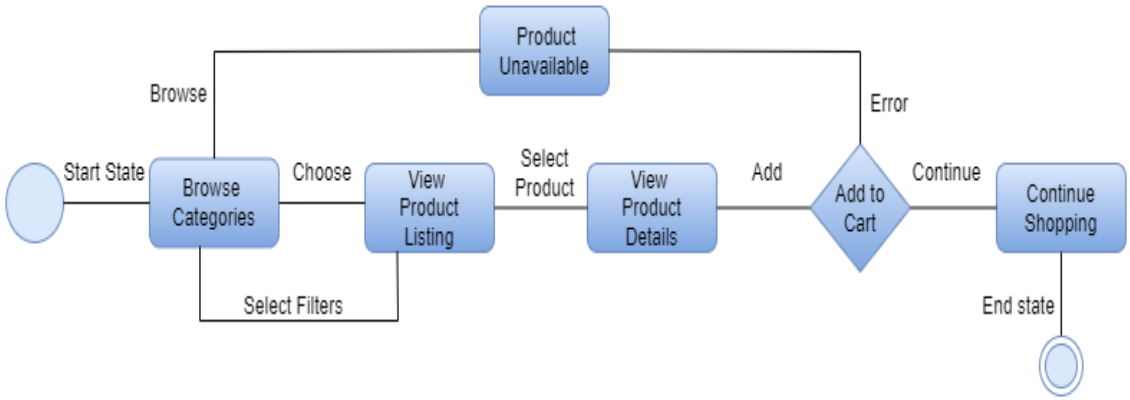


Figure 4- 13 State Chart Diagram for View Product

CHAPTER FIVE
5. SYTEM DESIGN

The successful implementation of a high-quality system depends on the nature of the designer's approach. The efficacy of the system design plays a crucial role in facilitating subsequent modifications once the system is operational. A well-designed system should exhibit adaptability, allowing for ease in making adjustments. The guidance for design objectives is provided by nonfunctional requirements, steering the development process.

The primary aim of system design is to manage complexity by breaking down the system into smaller, more manageable components. Various objectives contribute to this predominant goal. Design objectives serve as guidelines for developers, highlighting aspects that require careful consideration. In essence, system design tries to filter out irrelevant information and incorporate related details or functionalities.

To achieve this, several diagrams are employed in the design process. These include hardware-software mapping, current and proposed software architecture, persistent data management, and access control and security. These diagrams serve as essential tools in shaping and refining the system design, ensuring that it aligns with the specified objectives and requirements.

5.1. Design Goal

Design goals are statements crafted by a team to express the level of experience they intend a product to attain. The selection among various design options is influenced by these design goals. Take into account the objective of achieving an efficient design.

User Interface and Human Factors

The design incorporated a user-friendly graphical user interface (GUI) to facilitate straightforward interactions with the system. The system's interface featured distinct buttons that provide clear guidance on the necessary actions, guiding users through the process of entering their information. Additionally, the interface incorporated icons, frames, and a thoughtful use of colours to enhance visual appeal. The information flow within the interface is intuitive, clearly illustrating the navigation path for users, ensuring a seamless and engaging user experience.

Security Issues

To ensure the security of our information system and meet the crucial requirements of confidentiality, integrity, and availability, we are implementing authentication and authorization mechanisms by using Next Auth. These mechanisms involve providing users with a username and password, granting access solely to those individuals who have been accordingly authenticated and authorized. In terms of securing passwords within the database, we employ the bcrypt encryption technique. This method is instrumental in encrypting user passwords, providing a robust defence against potential attackers.

Performance Consideration

The system fulfilled the following outlined performance requirements.

Response Time: In regular circumstances, the system made every effort to swiftly present results when a user initiates an inquiry. Next.js reduces response time by rendering pages on the server, optimizing assets, streaming UI, and using a faster bundler.

Throughput: The system's performance optimized for throughput by focusing on concurrency control methods during implementation. Additionally, MongoDB, known for its proficiency in managing locks and ensuring consistency, utilized.

Memory: By adopting a straightforward analysis approach and making efficient use of memory, our system is designed to operate optimally within available RAM.

Error handling and Validation

The system incorporated validation rules, with the server side powered by Node.js and the client side using the Next.js built on React.js JavaScript library for validation and error management. This approach aims to minimize input errors. In cases of errors, the user interface generated informative error messages that are concise, meaningful, and clear, providing users with a clear understanding of the issue and guidance on how to resolve it. Furthermore, the system allows users to confirm actions like creation, deletion, or modification before they take place. In response to erroneous inputs, the system prompts users to re-enter data in the correct format. Thus, our system maintains robust error handling and input validation through JavaScript error-handling mechanisms and datatype checks for incoming inputs.

Quality Considerations

Availability: The probability that the system remains operational and capable of delivering relevant services whenever users require them.

Reliability: The system is designed to minimize crashes during its runtime, especially considering concurrent usage by multiple users. Achieving this involves implementing validation parameters for user input to prevent incorrect storage of records.

User Operability: The system ensures straightforward navigation, ensuring functionality for users with basic computer skills and it has user-centric design.

Backup and Recovery:

In the event of a system failure, having data stored in a separate location for backup simplifies the process of recovering lost information. This involves relocating the data to an alternative location. In cases where data is corrupted or lost, a swift restoration of the lost data can be achieved through the database's recovery capabilities. Data backup stands out as a crucial process within this system, mitigating the risks associated with data loss or corruption. To address these concerns, a database backup is essential, serving as a pre-existing copy of the database. In MongoDB, the mongodump tool is employed for data backup, while the mongorestore tool is utilized for the recovery of backed-up data.

5.2. Proposed System Architecture

The proposed system adopts a 3-tier architecture model, where the presentation layer serves as the client layer and represents the topmost interface of the application. This layer is the user's point of interaction with the system, facilitating information input. Its primary functionality lies in communication with the application layer, transmitting user inputs, including keyboard actions and mouse clicks, to the application layer.

The application layer, also known as the business logic layer, serves as an intermediary between the presentation layer and the data layer. It interacts with the data layer, retrieving information from the database and forwarding it to the presentation layer. This layer plays a vital role in managing the flow of information between the user interface and the database.

The data layer is responsible for storing the data entered by the user. Typically, clients access the system using a browser through the internet. In this scenario, when a user inputs data and takes specific actions, the application server processes the client's request, interacting with the database server to retrieve or store the necessary information.

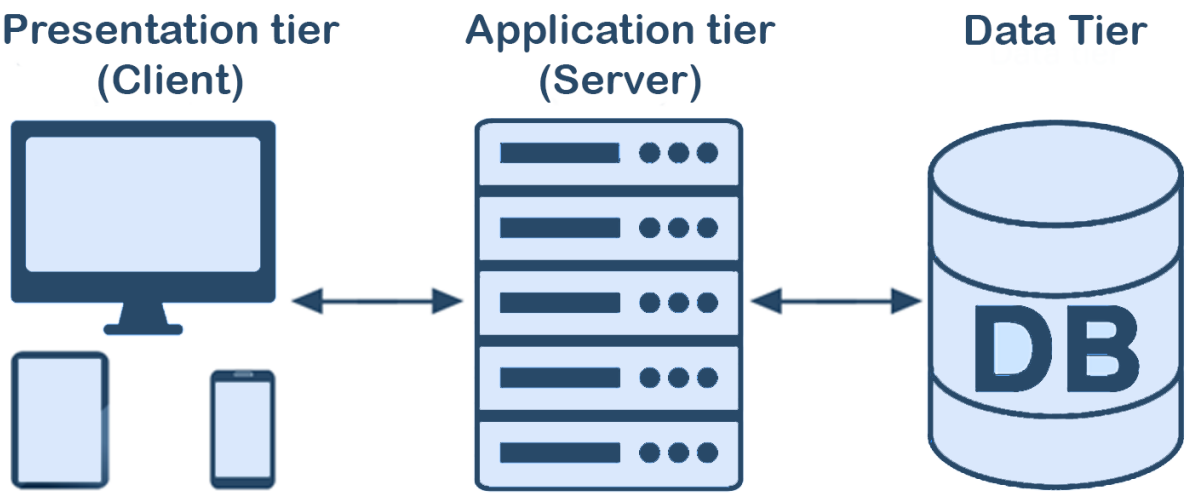


Figure 5- 1 Proposed System Architecture

5.2.1. Subsystem Decomposition and Description

Subsystem decomposition involves the systematic breakdown of the analytical model of the proposed system into manageable components. The objective of this decomposition process is to partition the system's schema into cohesive and significant components, thereby minimizing the complexity of the design model. In most cases, components within a system can be categorized as either computational units or data storage units. Each component is assigned a name, typically chosen to mirror its function or role within the system.

During the system's utilization, its numerous components are anticipated to interact cohesively to deliver the expected service. This interaction is precisely depicted by a component diagram. The diagram serves as a specification outlining the relationships and collaborations among the various components of the system.

Product and User Management: subsystem encompasses activities with "manage" prefixes, including Register (create), Update, Activate, and Deactivate. These activities fall under this subsystem and involve functions related to user registration, product management, and account administration.

Responsibilities:

- ✓ Manage Product Category
- ✓ Manage Products
- ✓ View Product
- ✓ Manage Account

- ✓ Registration

Order and Communication: subsystem handles tasks related to order management and communication functionalities.

Responsibilities:

- ✓ Manage Orders
- ✓ Make Orders
- ✓ Manage Order
- ✓ Chat
- ✓ Provide Feedback

Promotion and Reporting Component:

Responsibilities:

- ✓ Promotion
- ✓ Generating Report

Payment Processing Component:

Responsibilities:

- ✓ Make Payment
- ✓ Receive Payment
- ✓ Process Payments

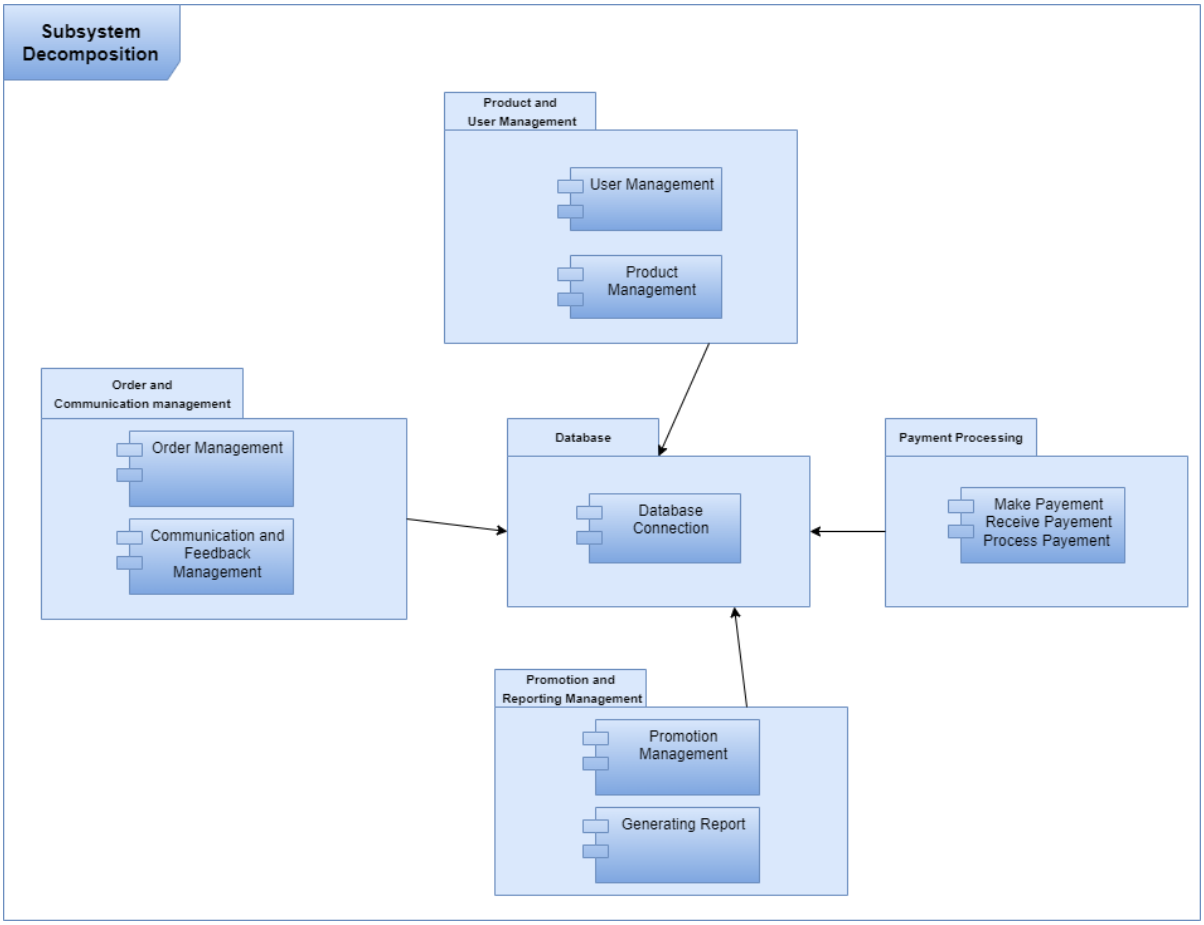


Figure 5- 2 Component Diagram

5.2.2. Hardware/Software Mapping

A deployment diagram illustrates the static representation of the runtime arrangement of processing nodes and the components that operate on those nodes. This mapping is utilized to showcase the hardware configuration when deploying the proposed system, facilitating the connection of distinct machines designated for configuration purposes. The diagram visually conveys how hardware and software collaborate within the deployment context.

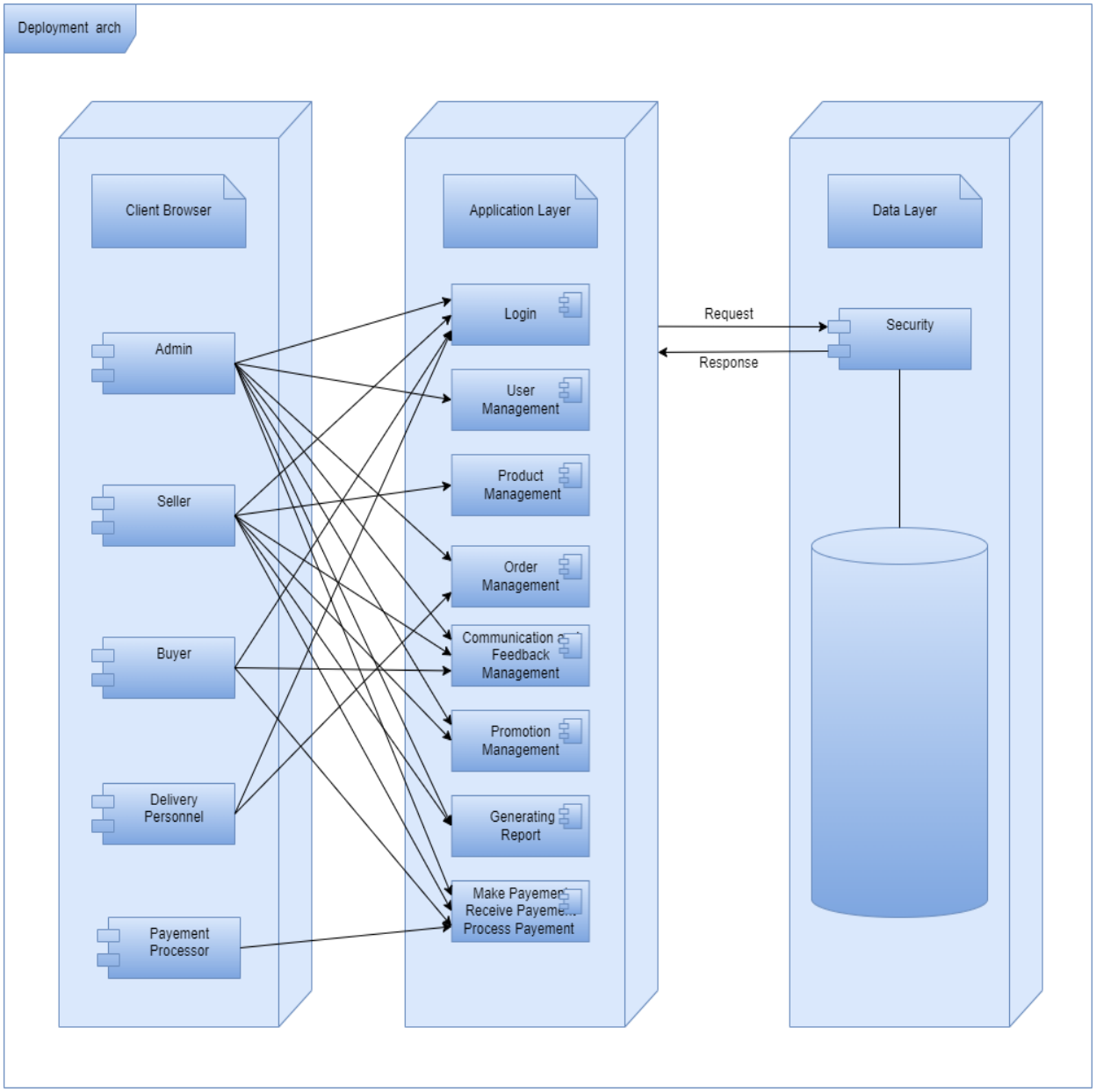


Figure 5- 3 Deployment Diagram

5.2.3. Detailed Class Diagram

In this section, we present a detailed class diagram of our system, showcasing classes, attributes, methods, attribute data types, and the visibility (public, private, and protected) of both attributes and methods. Additionally, we illustrate concepts such as inheritance, association, aggregation, composition, and system dependencies. The table below outlines the specifics of the class diagram for our system.

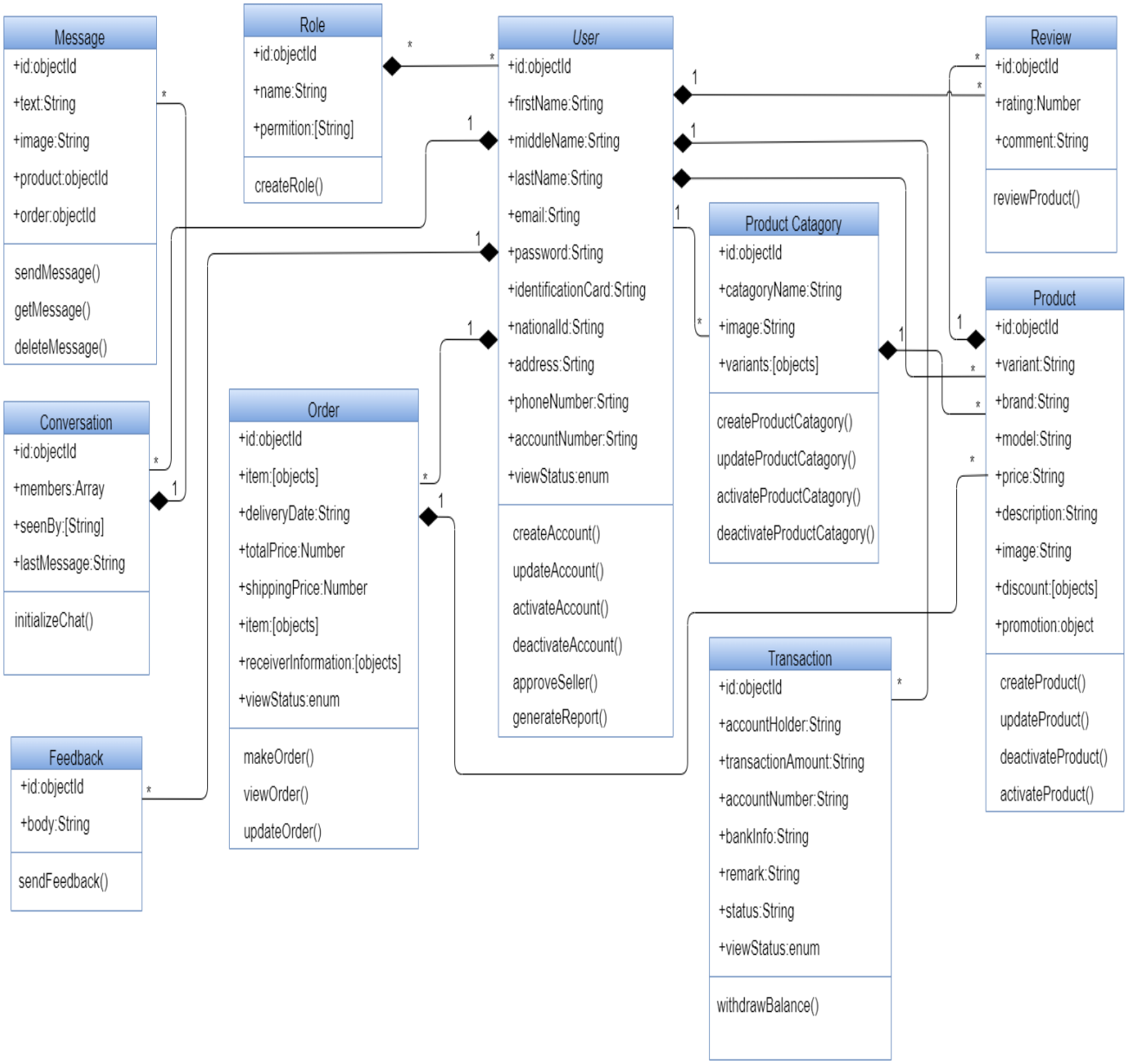


Figure 5- 4 Detail Class Diagram

5.2.4. Persistent Data Management

The persistent data management component of the proposed system outlines the infrastructure required to store and manage its enduring data.

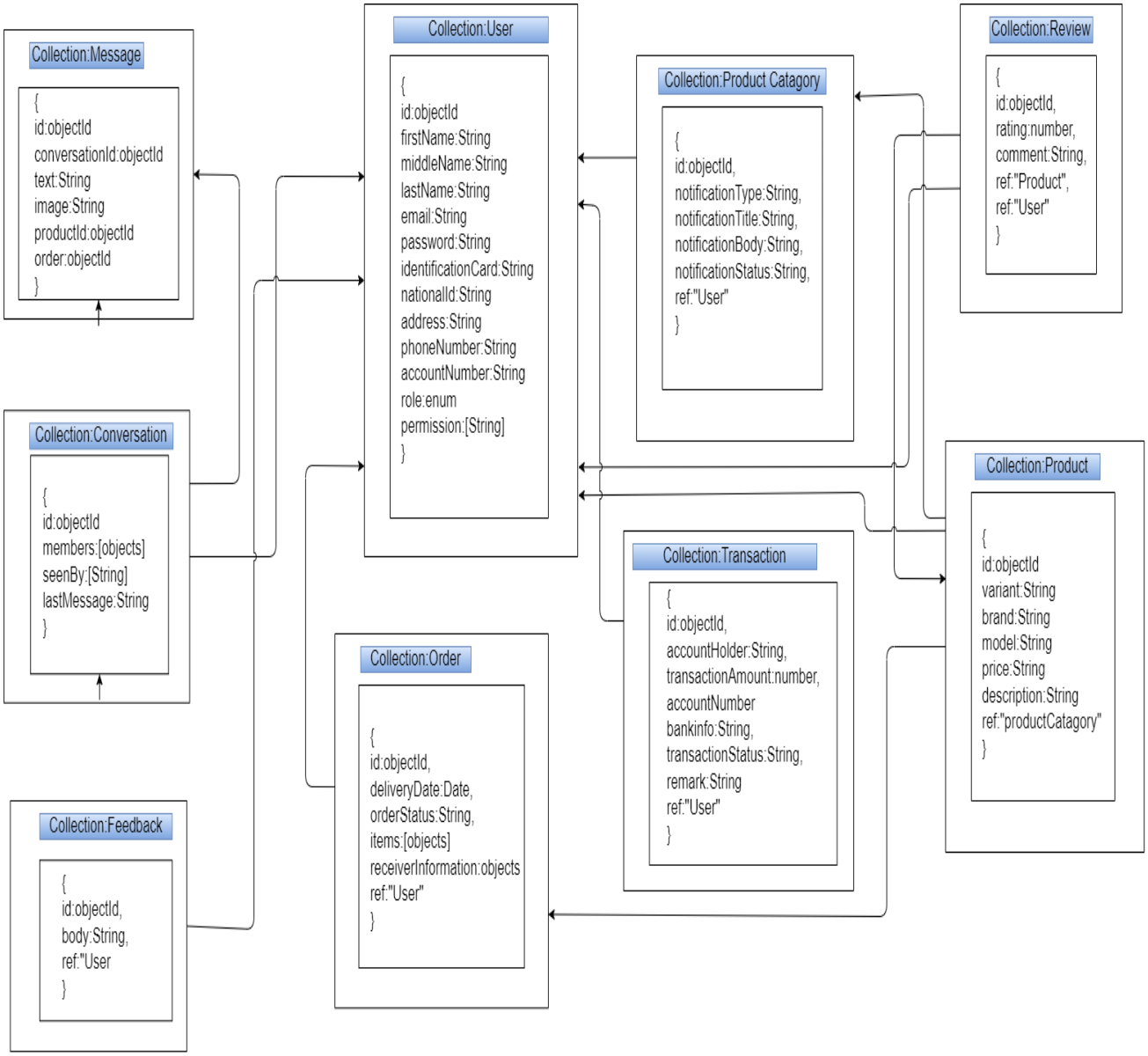


Figure 5- 5 Persistent Data Diagram

5.2.5. Access Control and Security

In the context of the proposed system built using Next.js, role-based access control (RBAC), and JSON Web Tokens (JWT) for authorization, the following principles are implemented to ensure access control and security:

Confidentiality: The system employs role-based access control, ensuring that only authorized individuals with specific roles can access designated information and data. Through the use of JSON Web Tokens, authentication is enforced via username and password, guaranteeing that private data remains confidential. This approach upholds personal privacy by restricting access to sensitive information to only those with the appropriate authorization.,

Integrity: Role-based access control mechanisms are in place to govern and limit the ability to modify or change data within the system. Only users with the necessary roles and permissions are granted the capability to alter data, preserving the integrity of the information stored in the system. This helps prevent unauthorized modifications and ensures that data remains reliable and accurate.

Availability: The system prioritizes availability by ensuring that it is accessible at all times to authorized users.

5.3. Packages

This section breaks down subsystems into packages as well as expansion on code files organization. It gives a complete picture of each package and lists its dependencies with other packages as well as stating how it should be used. Using UML package diagrams, the document seeks to diagrammatically represent the structure of packages.

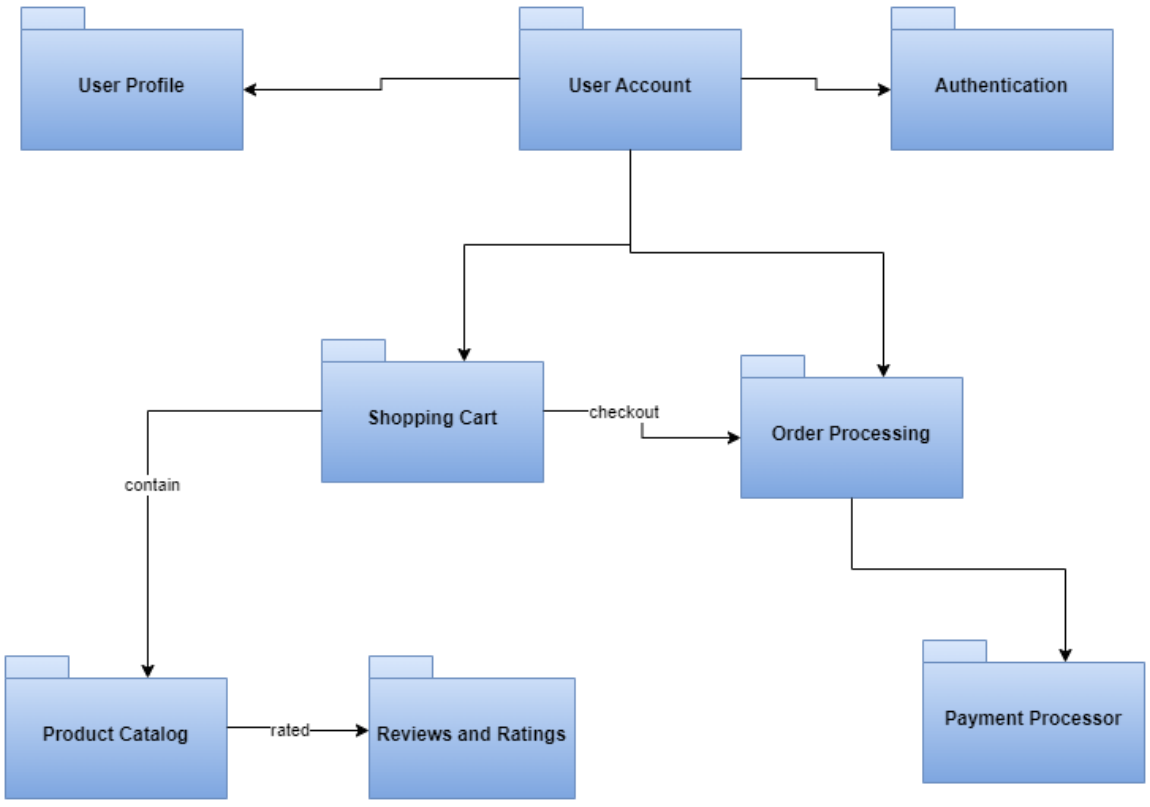


Figure 5- 6 Package Diagram

5.4. Algorithm Design

Algorithms are created to display the system's program flow. It displays the logical sequence of steps for each function. The coding portion of the implementation relies on this design component. The following is a list of some of the algorithms.

Login:

- ✓ System display login form on Login page.
- ✓ User enters username and passwords.
- ✓ User click on login in button

If user is authorized and authenticated the system display role based user account:

Else username and password are not correct, then the system displays the wrong username or password message and stays in login page.

```
start
    Login Page
    Enter username and password
    if(get.username===entered.username && get.password.compare(entered.password))
        Display Home Page
    else
        Stay on Login Page
        Display log in failed message
end
```

Approve Seller:

- ✓ System display admin account page.
- ✓ Select seller and click approve seller button

If all information of seller filled fully and no information left the system approve seller and display seller is approved successfully message and notification

Else if seller information is not filled fully system display information not filled fully message

```
start
    Admin account Seller approval Page
    Select seller and click approve button
    If(sellerInfromationFilled==100%)
        Seller approved and Display successfully approved message
    Else
        Display approval failed message
end
```

5.5. User Interface Design

User Interface Design, often referred to as UI design, involves the visual arrangement of elements that users interact with in a website or technological product. The objective is to create

designs that are not only visually appealing but also functional, keeping the user's needs in mind. The primary goal is to simplify and enhance the user's interaction, making it as efficient as possible in achieving their goals. In the proposed system, we have crafted a user interface that aims to elevate the overall user experience.

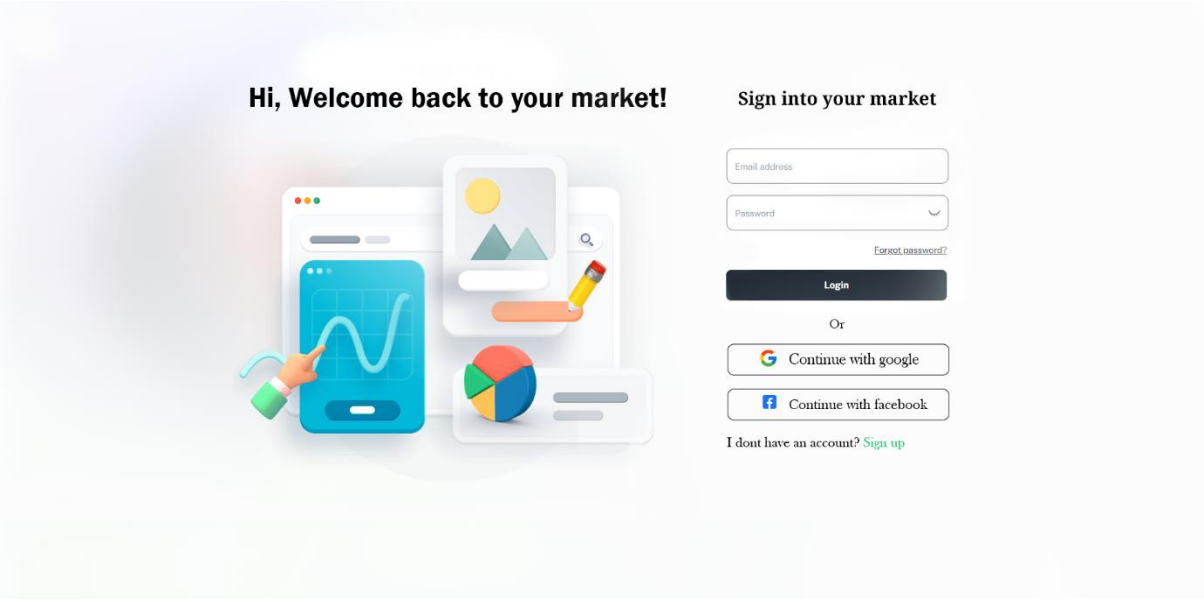


Figure 5- 7 Login Page User Interface

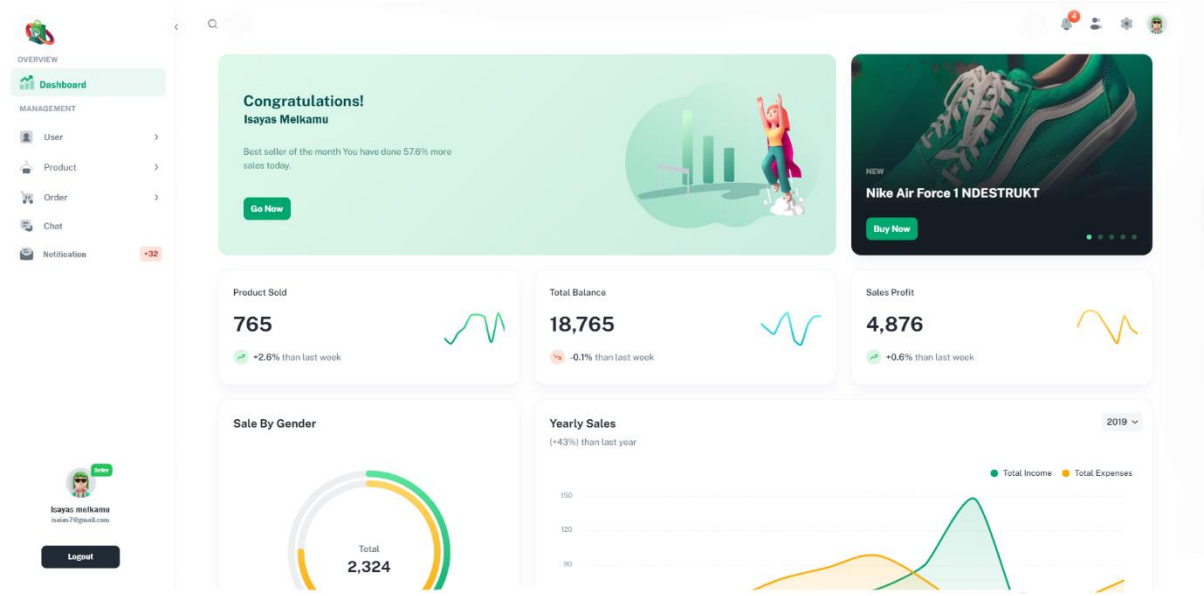


Figure 5- 8 Dashboard User Interface

CHAPTER SIX

6. IMPLEMENTATION AND TESTING

6.1. Implementation of the Database

We have chosen MongoDB as the database management system (DBMS) for our marketplace platform project due to its flexibility, scalability, and proficiency in handling unstructured data efficiently. In this report, we outline the implementation process of the MongoDB database according to the provided guidelines.

Tables as Persistent Models

MongoDB diverges from traditional relational databases by utilizing collections instead of tables for storing documents. Each collection can be likened to a table, with documents within collections serving as individual records. In our implementation with Mongoose, we have established models that correspond to MongoDB collections, as depicted below.

The following are some schema from our database model:

User schema

```
import { statusData } from "@/utils/permission";
import mongoose from "mongoose";

const { Schema } = mongoose;

const userSchema = new Schema(
  {
    profileImage: {
      type: String,
    },

    firstName: {
      type: String,
      required: true,
    },
    middleName: {
      type: String,
      required: true,
    },
    lastName: {
      type: String,
    },

    identificationCard: {
      type: String,
```

```
    },
    nationalId: {
      type: String,
    },
    address: {
      type: String,
    },
    location: {
      type: [Number],
      required: true,
    },
    phoneNumber: {
      type: String,
    },
    bankInfo: {
      type: String,
    },
    accountNumber: {
      type: String,
    },
    email: {
      type: String,
      unique: true,
      required: true,
    },
    password: {
      type: String,
    },
    role: [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Role",
        required: true,
      },
    ],
    isSeller: {
      type: Boolean,
      default: false,
    },
    status: {
      type: String,
      default: statusData.Active,
    },
    balance: {
      type: Number,
      default: 0,
    },
  },
```

```
  { timestamps: true }
);

export default mongoose.models.User || mongoose.model("User", userSchema);
```

Role schema

```
import mongoose from "mongoose";

const { Schema } = mongoose;

const roleSchema = new Schema(
  {
    role: {
      type: String,
      unique: true,
      required: true,
    },
    permission: [String],
  },
  { timestamps: true }
);

export default mongoose.models.Role || mongoose.model("Role", roleSchema);
```

Review schema

```
import mongoose from "mongoose";

const { Schema } = mongoose;

const reviewSchema = new Schema(
  {
    productId: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Product",
    },
    userId: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "user",
      required: true,
    },
    comment: {
      type: String,
      required: true,
    },
    comment: {
      type: String,
      required: true,
    },
```

```
  },
  { timestamps: true }
);

export default mongoose.models.Review || mongoose.model("Review",
reviewSchema);
```

ProductCategory schema

```
import { statusData } from "@/utils/permission";
import mongoose from "mongoose";

const { Schema } = mongoose;

const productCategorySchema = new Schema(
  {
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    categoryName: {
      type: String,
      required: true,
      unique: true,
    },
    categoryImage: {
      type: String,
      required: true,
    },
    status: {
      type: String,
      default: statusData.Active,
    },
    variants: [
      {
        name: {
          type: String,
          required: true,
          unique: true,
        },

        image: {
          type: String,
          required: true,
        },
        status: {
          type: String,
```

```
          default: statusData.Active,
        },
        brands: [
          {
            name: {
              type: String,
              required: true,
            },

            image: {
              type: String,
              required: true,
            },
            status: {
              type: String,
              default: statusData.Active,
            },
          },
        ],
      },
    ],
  },
  { timestamps: true }
);

export default mongoose.models.ProductCategory ||
  mongoose.model("ProductCategory", productCategorySchema);
```

Configuring a Schedule of Database Backup

MongoDB provide built in configuring schedule for database backup and we can accomplish it through mongodump a command-line tool that creates a binary export of the contents of a MongoDB instance. In our case, since we utilize the cloud database MongoDB Atlas, we only configure the backup frequency and retention policy, as database backups are managed by the provider. Backup frequency is how frequently backups should occur (e.g., every 6 hours, daily). Retention policy is how long backups should be retained (e.g., retain backups for 7 days, 30 days, or longer).

Configuring Database-Level Security

Our project prioritized database security to protect sensitive information and ensure authorized access. We utilized Next.js for application development and NextAuth for authentication and authorization. Next.js middleware aids in implementing database-level security effectively. NextAuth provides seamless authentication and authorization. Next.js middleware enforces

access control at the server-side, validating user authentication tokens and authorizing access based on predefined rules.

6.2. Configuration of the Application Server

Justification for Next.js:

In Next.js 14, API route handlers serve as an application server for handling HTTP requests. These handlers allow you to define server-side logic directly within your Next.js project, making it easy to create serverless APIs without the need for an external application server like Express.js. Here's how Next.js 14 API route handlers work as an application server.

Starting and Shutting Down the Application Server:

Server can start and shut down the Next.js application server using commands in the terminal. using npm start to start the server and Ctrl + C to shut it down.

Configuring Access from the Same Machine:

By default, when you start the Next.js development server using npm start, it binds to localhost allowing access from the same machine.

Configuring Access from a Remote Machine:

To allow access from a remote machine, server can start with the host parameter set to 0.0.0.0. For example, npm run dev -- --hostname 0.0.0.0 --port 8000.

Separation of Application Server and Database:

Application Server and Database Separation: For the purpose of segregating our application server from the database, we opt for cloud-based database solutions like MongoDB Atlas. This approach ensures that our application remains independent from the underlying database infrastructure, offering enhanced flexibility and scalability.

Configuration of our application to connect to the database involves utilizing environment variables for connection strings. Our MongoDB connection string, for instance, is set as follows:

```
MONGO=mongodb+srv://marketplace:marketplace@cluster0.qp0lqio.mongodb.net/marketplace?retryWrites=true&w=majority
```

To establish the connection between our database and application server, we employ the following controller:

```
import mongoose from "mongoose";

const connect = async () => {
  try {
    await mongoose.connect(process.env.MONGO);
  } catch (error) {
    console.log("connect to mongodb failed", error);
  }
};

export default connect;
```

In our system, we utilize the ***connect function*** at various points to establish the connection between our Application Server and the Database. This function is employed anywhere within our system where database connectivity is required, ensuring seamless communication between the server and the database.

Configuring the Server to Work on Different Port Numbers:

By default, the Next.js development server runs on port 3000. You can configure it to use a different port by specifying the PORT environment variable.

For example, ” npm run dev -- --port 8000” will start the server on port 8000.

6.3. Configuration of Application Security

Considering the security considerations specified in the system's non-functional requirements, configuring application security is paramount to guarantee the integrity, confidentiality, and availability of the marketplace platform web application. Here's how we tackle these responsibilities:

Input Validation: We check the information users enter to make sure it's safe and correct. We do this both on the client and in our server using special tools like Zod and React Hook Form. This helps us set clear rules for the data and make sure it's safe before we use it.

Encryption/Decryption: To safeguard sensitive user credentials, we've implemented the bcrypt hashing algorithm for password encryption. This ensures strong security measures to protect user information from unauthorized access.

Role Definition and Access Privileges: Clear role definitions are established for different user types within our system, such as Admin, buyer, seller, and personnel deliver. Each role is

assigned specific access privileges, ensuring that users can only perform actions and access resources relevant to their roles.

Session Management: We used the NextAuth built-in library for session management. Its built-in features ensure secure maintenance of user sessions by automatically handling token generation, expiration, and refreshment. The benefits of NextAuth include simplified authentication setup, support for various authentication providers, and enhanced security through its robust session management capabilities.

6.4. Implementation of User Interface

In crafting the user interface, our central aim revolves around embracing a user-centered design methodology. This approach ensures that users have full autonomy and control over their interactions within the system. To accomplish this objective, we've integrated several features, including options for both dark and light modes, enabling users to tailor their interface to suit personal preferences and environmental conditions. Moreover, we've streamlined user experience by minimizing cognitive load through intuitive navigation paths, clear labels, and contextual prompts, thereby facilitating seamless interaction and reducing mental strain. Consistency is upheld across the entire user interface through standardized design elements, layout structures, and interaction patterns, fostering a sense of familiarity and ease of use. These concerted efforts culminate in a user-centric interface that prioritizes user empowerment, cognitive simplicity, and uniformity, ultimately enriching the overall user experience.

6.5. Testing

Testing is a crucial aspect of software development that plays a pivotal role in verifying the correctness, reliability, and functionality of a system. In the context of the Online Marketplace Platform thorough testing is essential to ensure that the system meets the requirements and expectations of its users.

6.5.1. Test Case

Test Case 1

Test Case Name: Verify User Registration

Description: Test the user registration functionality to ensure users can successfully register with valid credentials.

Inputs: User details (name, email, password)

Expected Output: User account created successfully

Steps to Reproduce:

Navigate to the registration page.

Enter valid user details in the registration form.

Click the "Register" button.

Verify that the user account is created successfully and a confirmation message is displayed.

Test Case 2

Test Case Name: Verify Product Search Functionality

Description: Test the product search functionality to ensure users can find products based on search queries.

Inputs: Search query (product name, category, etc.)

Expected Output: Relevant products displayed in search results

Steps to Reproduce:

Navigate to the search bar and enter a valid search query.

Click "Search" and verify relevant product listings appear.

Check essential product details like name, price, and category.

Click a product to confirm it matches the search criteria.

Test with different queries for validation and handle errors appropriately.

Verify pagination or infinite scroll works for multiple pages of search results.

6.5.2. Testing Tools and Environment

Testing Tools

- Jest (for unit testing)
- Cypress (for system testing)
- Postman (for integration testing)

Testing Environment

- Windows 11 operating system
- Chrome browser

6.5.3. Unit Testing

Unit testing within the Online Marketplace Platform involves testing individual components and modules using the Jest framework. This approach strengthens our system's reliability by identifying and resolving issues early, ensuring smooth functionality, and enhancing overall robustness. Below are some unit tests for various components

RegistrationForm unit testing:

```
import { render, fireEvent, waitFor } from "@testing-library/react";
import RegistrationForm from "./RegistrationForm";

describe("RegistrationForm component", () => {
  it("submits the form with valid data", async () => {
    const { getByLabelText, getByText } = render(<RegistrationForm />);

    // Fill out the form
    fireEvent.change(getByLabelText("First Name"), {
      target: { value: "John" },
    });
    fireEvent.change(getByLabelText("Last Name"), { target: { value: "Doe" } });
    fireEvent.change(getByLabelText("Email"), {
      target: { value: "example@gmail.com" },
    });
    fireEvent.change(getByLabelText("Password"), {
      target: { value: "password" },
    });
    fireEvent.change(getByLabelText("Confirm Password"), {
      target: { value: "password" },
    });

    // Submit the form
    fireEvent.click(getByText("Submit"));

    // Wait for redirection or success message (assuming redirection upon success)
    await waitFor(() => {
      expect(window.location.pathname).toEqual("/login");
    });
  });
});
```

Outputs of the above unit testing code

```
RegistrationForm component
  ✓ submits the form with valid data

1 passing (10ms)
```

LoginForm unit testing:

```
import { render, fireEvent, waitFor } from "@testing-library/react";
import LoginForm from "./LoginForm";

jest.mock("next-auth/react", () => ({
  useSession: jest.fn().mockReturnValue({ status: "authenticated" }),
}));

describe("LoginForm component", () => {
  it("submits the form with valid data", async () => {
    const { getByLabelText, getByText } = render(<LoginForm />);

    // Fill out the form
    fireEvent.change(getByLabelText("Email"), {
      target: { value: "example@gmail.com" },
    });
    fireEvent.change(getByLabelText("Password"), {
      target: { value: "password" },
    });

    // Submit the form
    fireEvent.click(getByText("Login"));

    // Wait for redirection or success message (assuming redirection upon success)
    await waitFor(() => {
      // Check if the callbackUrl is set correctly
      expect(localStorage.getItem("prevpath")).toBe("/");
    });
  });
});
```

Outputs of the above unit testing code

```
LoginForm component
  ✓ submits the form with valid data

1 passing (10ms)
```

6.5.4. System Testing

System testing within the Online Market Place Platform entails assessing the functionality of the entire system to confirm it aligns with specified requirements. Cypress was utilized to automate UI tests, replicating real user interactions across various browsers. Below are sample system tests for our system.

```
describe("Marketplace Platform", () => {
  beforeEach(() => {
```

```
  // Visit the homepage before each test
  cy.visit("/");
 });

 it("should allow users to register", () => {
  // Navigate to the registration page
  cy.contains("Register").click();

  // Fill out the registration form
  cy.get('input[name="firstName"]').type("John");
  cy.get('input[name="lastName"]').type("Doe");
  cy.get('input[name="email"]').type("johndoe@example.com");
  cy.get('input[name="password"]').type("password123");
  cy.get('input[name="confirmPassword"]').type("password123");

  // Submit the registration form
  cy.contains("Register").click();

  // Assert that the user is redirected to the dashboard after registration
  cy.url().should("include", "/dashboard");
 });

 it("should allow registered users to log in", () => {
  // Navigate to the login page
  cy.contains("Login").click();

  // Fill out the login form
  cy.get('input[name="email"]').type("johndoe@example.com");
  cy.get('input[name="password"]').type("password123");

  // Submit the login form
  cy.contains("Login").click();

  // Assert that the user is redirected to the dashboard after login
  cy.url().should("include", "/dashboard");
 });

 it("should allow users to browse products", () => {
  // Navigate to the products page
  cy.contains("Products").click();

  // Assert that the products page is loaded
  cy.url().should("include", "/products");

  // Assert that there are products displayed on the page
  cy.get(".product").should("have.length.gt", 0);
 });
```

```
  it("should allow users to add items to the cart", () => {
    // Navigate to the products page
    cy.contains("Products").click();

    // Click on the first product
    cy.get(".product").first().click();

    // Assert that the product details page is loaded
    cy.url().should("include", "/products");

    // Click on the "Add to Cart" button
    cy.contains("Add to Cart").click();

    // Assert that the item is added to the cart
    cy.contains("Cart").click();
    cy.contains("1 item(s) in cart");
  });
});
```

Outputs of the above system testing code

```
Marketplace Platform
  ✓ should allow users to register
  ✓ should allow registered users to log in
  ✓ should allow users to browse products
  ✓ should allow users to add items to the cart

4 passing (10s)
```

6.5.5. Integration Testing

In the Online Market Place Platform, integration testing focuses on evaluating the interaction among modules to guarantee smooth collaboration. Tools like Postman are utilized to test API endpoints, validating data flow and communication integrity across the system's API endpoints. Below are some Integration tests for various our system's API endpoints.

User Registration API endpoint Integration testing:

This integration test aims to verify the functionality of the user registration endpoint by sending a POST request to the /api/user route with valid user registration data. The test ensures that the user is successfully created in the system and that the appropriate success message is returned.

1. Send a POST request to http://localhost:3000/api/user with the following request body:

```
{
  "firstName": "John",
  "middleName": "Doe",
  "email": "John@gmail.com",
  "password": "12345",
  "confirm_password": "12345",
  "role": "Buyer"
}
```

2. Receive the response from the server.

Expected Outcome:

The response status code should be 200 OK.

The response body should contain the message "User has been created successfully".

Actual Outcome:

The response status code was 200 OK.

The response body contained the message "User has been created successfully".

CHAPTER SEVEN

7. CONCLUSION AND RECOMMENDATION

7.1. Conclusion

The Online Marketplace Platform project seeks to transform e-commerce by tackling key challenges in current systems. Through direct communication between sellers and buyers, wider market access, and competitive pricing, the platform enhances convenience for users. The project's feasibility is supported by operational, technical, and economic analyses, ensuring its viability and sustainability. Using a robust methodology involving data collection, system analysis, and modern tools like Next JS and Node JS with MongoDB, the project aims to build a user-friendly online marketplace. Future opportunities for growth highlight its importance in fostering economic expansion and streamlining online commerce.

In simple terms, Online Marketplace platform isn't just a project we've finished it's a testament to our hard work and our commitment to making online shopping better for everyone. And as we keep going, we're excited to keep making our platform even more helpful and user-friendly to meet the ever-changing needs of online shoppers.

7.2. Recommendation

To further enhance the effectiveness of the Online Marketplace Platform, the following recommendations are suggested:

- ✓ Implement a feedback collection process to gather insights from users regularly, enabling continuous improvement and the addition of new features to meet evolving needs.
- ✓ Provide comprehensive training and ongoing support to both sellers and buyers to ensure they can effectively utilize all platform features and maximize benefits.
- ✓ Explore integrating artificial intelligence (AI) algorithms to offer predictive insights and personalized recommendations based on user behaviour and transaction history.
- ✓ Ensure the platform is accessible for people with disabilities, promoting inclusivity and providing an equitable user experience for all users.
- ✓ Optimize Mobile Accessibility Optimize the platform for mobile devices to accommodate the growing number of users accessing the marketplace through smartphones and tablets.

8. REFERENCES

Software Engineering 10th Edition (April 2015) by Ian Sommerville (Author) Pearson Education.

Draw.io. (2024). Flowchart Maker & Online Diagram Software. App.diagrams.net. https://app.diagrams.net/

Visual Paradigm. (2019). What is Class Diagram? Visual-Paradigm.com. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

GitHub. (2024). GitHub: Where the world builds software. https://github.com/