Online Exchange Web Application

Software Requirements Specification

1.0

05-06-2024

Team Members  
  
Balanjani Kamasani  
Sushma Mundlamuri  
Tejaswini Pydipally  
Nikhitha Sree Atluri

Submitted in partial fulfillment

Of the requirements of

CSIS 44-691 Graduate Directed Project 1

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| <date> | <Version 1> | <Your Name> | <First Revision> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

**Table of Contents**

1. Introduction

1.1. Purpose

1.2. Scope

1.3. Definitions, Acronyms, and Abbreviations

1.4. References

1.5. Overview

2. General Description

2.1. Product Perspective

2.2. Product Functions

2.3. User Characteristics

2.4. General Constraints

2.5. Assumptions and Dependencies

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

3.1.2. Hardware Interfaces

3.1.3. Software Interfaces

3.1.4. Communications Interface

3.2. Functional Requirements

3.3. Use Cases

3.4. Class/Objects

3.5. Non-Functional Requirements

3.5.1. Performance

3.5.2. Reliability

3.5.3 Availability

3.5.4. Security

3.5.5. Portability

3.6. Inverse Requirements

3.7. Design Constraints

3.8. Logical Database Requirements

3.9. Other Requirements

3.10. Prototypes (for complete project)

3.11. Use Case Diagrams

4. Design

4.1. ER diagram

4.2. GUI

5. Analysis Models

5.1. Data Flow Diagram

5.2. Sequence Diagram

1. **Introduction**

**1.1 Purpose**

* To provide a secure and efficient environment for users to buy, sell, and exchange goods and services.
* To simplify the process of listing and finding items or services through advanced search and categorization features.
* To enhance user satisfaction through customer support and user-friendly interface designs.

**1.2 Scope**

* User and Transaction Management: The platform will allow users to register, and manage their profiles, and it will track and store details of their transactions. Users can buy, sell, or trade items such as books, electronics, and clothes
* Search Functionality: In an online exchange project, the search functionality is crucial for users to find specific products or services among the available listings.
* Auction Functionality: The platform will support an auction-style sales process where users can list items for sale, and potential buyers can place bids over a specified period. The item will be sold to the highest bidder
* Database Usage: We will implement a database to securely store and manage user data, including personal details, item listings, and transaction histories. This database will support essential functions like user authentication, item searches, and purchase history.
* Platform Accessibility: Initially, the platform will be web-based, designed to be accessible on both desktop and mobile browsers.
* Payment System Integration: The platform will integrate a secure payment processing system to handle transactions once an auction concludes. This system will facilitate the transfer of funds from the highest bidder to the seller, ensuring that all financial transactions are safe and traceable.

**1.3 Definitions, Acronyms, and Abbreviations**

* SQL (Structured Query Language): A domain-specific language used for managing and manipulating relational databases.
* UI (User Interface): The graphical layout of an application, including buttons, menus, and other interactive elements, through which users interact.
* HTML (Hypertext Markup Language): The standard markup language for creating web pages and web applications.
* CSS (Cascading Style Sheets): A style sheet language used to enhance the user Experience
* JS(JavaScript): JavaScript is a programming language commonly used to create interactive and dynamic content on websites. It is primarily executed in web browsers to enhance the functionality and interactivity of web pages
* User: Any registered individual or business entity that uses the platform to buy, sell, or exchange items.
* Listing: A posted item or service available for sale or exchange on the platform.
* Transaction: A formal agreement between a buyer and a seller for the exchange of goods, services, or currency.
* Payment Gateway: A service provider that facilitates online transactions by enabling the transfer of funds between buyers and sellers.

**1.4 References**

**Online Exchange Platforms:**

* eBay <https://www.ebay.com/>
* Amazon Marketplace <https://sellercentral.amazon.com/>
* Etsy <https://www.etsy.com/>

**Research Papers and Case Studies:**

* "The Impact of Online Exchange Platforms on Consumer Behavior" by John Doe, Journal of E-commerce Research.
* "Building Trust in Online Marketplaces: A Case Study of Airbnb" by Jane Smith, Proceedings of the ACM Conference on Computer-Supported Cooperative Work.

**1.5 Overview**

* Description of the main modules of the platform: User Registration, Product Listing, Search and Filter, Transaction Processing, and User Feedback.
* Sellers add their stuff for sale by uploading pictures and describing them.
* Buyers look for what they want by searching and using filters to narrow down the options.
* Transaction Processing: Buyers pick what they want, pay for it, and sellers send it. Both can leave feedback afterward.
* Outline of the user journey from registration to completing a transaction.

**2. General Description**

**2.1 Product Perspective:**  
  
An innovative addition to our online exchange web application that improves the user experience by displaying items in a more immersive and dynamic manner. This tool allows customers to view items from various perspectives, gaining a better knowledge of their features, design, and overall quality.  
  
**2.2 Product Functions:**

* **User Management:** This function allows users to manage their accounts, including registration, login, and profile management.
* **Product Listing:** This function enables users to list their products, including product information, images, and categorization.
* **Search and Filtering:** This function allows users to search for products and filter the results based on various criteria, such as category, price, and location.
* **Payment Gateway Integration**: This function enables users to make transactions securely and efficiently.
* **Rating and Review System:** This function allows users to rate and review products and sellers, building trust and improving the overall user experience.
* **Chat and Messaging:** This function enables users to communicate with each other, facilitating transactions and improving the user experience.
* **Product Perspective**: This function allows users to view products from different angles, providing a more immersive and interactive experience.
* **Push Notifications:** This function keeps users informed about new products, messages, and other important updates.
* **Product Recommendation:** This function suggests relevant products to users based on their search history and preferences.

**2.3 User Characteristics**

* **Tech-Savvy:** Users of the web application are likely to be comfortable with technology and the internet, using various devices to access the platform.
* **Bargain Hunters:** Many users are likely to be bargain hunters, looking for the best deals and prices on products.
* **Sellers:** Some users are likely to be sellers, looking to sell their used or new products to a wider audience.
* **Buyers:** Other users are likely to be buyers, looking for products to purchase at competitive prices.
* **Time-Conscious**: Users of the web application are likely to be time-conscious, looking for a quick and efficient way to buy and sell products.
* **Security-Conscious:** Users are likely to be concerned about security, looking for a platform that provides secure transactions and protects their personal information.
* **Mobile-Friendly:** Many users are likely to access the web application using mobile devices, requiring a mobile-friendly user interface.
* **Geographically Diverse:** Users of the web application are likely to be geographically diverse, requiring a platform that can handle transactions across different regions and countries.
* **Active Communicators:** Users may be active communicators, looking for a platform that enables easy communication between buyers and sellers.

**2.4 General Constraints**

* **Technological Compatibility:** The immersive viewing feature must be compatible with a wide range of devices and web browsers to ensure accessibility for all users.
* **Bandwidth Considerations:** The feature should be optimized to minimize bandwidth usage without compromising the quality of the viewing experience, particularly for users with slower internet connections.
* **Data Privacy and Security:** Measures must be implemented to ensure the privacy and security of user data, particularly if the immersive viewing feature involves storing or transmitting sensitive information.
* **Development Resources:** The development team may have constraints in terms of time, budget, and expertise when implementing this feature, which could impact its scope and timeline.

**2.5 Assumptions and Dependencies**

**Assumptions:**

* Users will value the ability to examine products in detail before making a purchase decision, indicating a demand for immersive viewing features.
* Integration of immersive viewing functionality will contribute positively to user engagement, potentially increasing user retention and conversion rates.

**Dependencies:**

* The successful implementation of the immersive viewing feature may rely on access to sufficient product data, including high-quality images and detailed specifications.
* Integration with third-party tools or APIs may be necessary to support advanced interactive features, such as 360-degree product views or augmented reality experiences.

**3. Specific Requirements**

**3.1. External Interface Requirements**

**3.1.1. User Interfaces**

* **User Registration/Login Interface**: A user-friendly interface allowing users to register for an account or log in with existing credentials.
* **Dashboard Interface:** A personalized dashboard for each user displaying their account information, transaction history, current orders, and other relevant data.
* **Item Posting Interface:** An intuitive interface enabling users to post items they want to exchange, including fields for item details, images, and exchange preferences.
* **Search and Browse Interface:** A search and browse functionality allowing users to discover items based on various criteria such as category, location, or keyword.
* **Messaging Interface:** A messaging system facilitating communication between users to negotiate terms, ask questions, and finalize exchanges.
* **Profile Management Interface:** An interface for users to manage their profile information, including contact details, preferences, and privacy settings.
* **Notification Interface:** A notification system to alert users about new messages, exchange requests, updates on items, and other important events.

**3.1.2. Hardware Interfaces**

* **Web Browser Interface:** The application should be accessible via standard web browsers on desktop and mobile devices.
* **Device Camera Interface:** If the application supports uploading images for listings, it should integrate with device cameras for capturing photos.

**3.1.3. Software Interfaces**

* **Analytics Interface:** Integration with analytics tools or services to gather insights into user behavior, traffic patterns, popular items/categories, and other metrics to improve the platform's performance and user experience.
* **Admin Interface:** An administrative interface for managing user accounts, resolving disputes, moderating listings, and enforcing community guidelines.
* **Content Management Interface**: A content management system (CMS) interface for managing static and dynamic content on the platform, including help articles, FAQs, and informational pages.
* **Integration Interface:** Interfaces for integrating with external systems such as inventory management software, shipping services, or third-party APIs for additional features and functionalities.
* **Security Interface:** Integration with security tools and services for implementing measures such as authentication, authorization and monitoring to ensure the platform's security and protect user data.
* **Localization Interface:** Integration with localization services or tools to support multiple languages and regions, allowing users to access the platform in their preferred language and currency.
* **Backup and Recovery Interface:** Interfaces for implementing backup and recovery procedures to safeguard user data and ensure business continuity in the event of system failures or data loss incidents.
* **Version Control Interface:** Integration with version control systems for managing codebase changes, tracking updates, and collaborating on software development efforts effectively.

**3.1.4. Communications Interface**

**Protocols:**

* **HTTP/HTTPS**: Use of HTTP/HTTPS protocols for web traffic and to deliver content from the server to the user's web browser.
* **WebSocket:** For real-time data exchange, can be used to continuously update the order book. As new orders are placed, modified, or canceled, these updates are sent in real-time to all clients viewing them, ensuring that all participants have up-to-the-minute data.

**Data Formats:**

* **Interchange Formats:** JSON, XML for data serialization. It can be used for the purpose of exporting data, such as trade orders or updating account information.
* **File Transfer:** FTP/SFTP protocols for secure file exchanges. Users may need to import or export data. The application can facilitate these operations via file transfer protocols, ensuring data integrity and security.

**3.2. Functional Requirements**

**Operations and Workflows:**

* **User Registration:** The registration process should include fields for essential information such as name, email address, password, and optional fields like profile picture or bio. Upon submission, the system should verify the email address and guide users through account setup.
* **Listing Creation:** Sellers should be guided through the process of creating listings, providing structured forms for entering details like title, description, category, price, location, and uploading images. The system should ensure that mandatory fields are filled and validate data formats.
* **Search and Browse Functionality:** The search interface should allow users to enter keywords, apply filters, or browse categories to find relevant listings. Results should be displayed in a user-friendly manner, with options for sorting and pagination.
* **Messaging Between Users:** Users should be able to initiate conversations with each other regarding listings. The messaging system should support real-time communication and provide notifications for new messages.
* **Transaction Management:** The platform should facilitate transactions between buyers and sellers, providing options for payment processing and order tracking. Users should be guided through the checkout process, with clear steps for completing the transaction.
* **Rating and Reviews:** After a transaction is completed, both buyers and sellers should be prompted to leave ratings and reviews for each other. The system should ensure that reviews are genuine and provide mechanisms for reporting abusive or fraudulent behavior.
* **Administrative Tasks:** Admins should have access to a dashboard where they can manage users, listings, reported content, and site settings. The dashboard should provide insights into platform activity and enable admins to take appropriate actions.

Bidding Functionality:

* **Bidding Process:** The platform should enable users to place bids on listings, indicating the amount they are willing to pay for an item. Bidding should be conducted in a structured manner, with clear instructions and guidelines provided to users. The bidding process should allow potential buyers to place bids over a specified period, with the item ultimately being sold to the highest bidder.
* **Bid Management:** Sellers should be able to view and manage incoming bids on their listings, including accepting, rejecting, or counter-offering bids. Users should receive notifications for bid updates and outcomes.
* **Bid History:** Users should have access to a bid history, where they can view their own bidding activity as well as the bidding history of listings they are interested in. Transparency in bidding activity enhances trust and facilitates fair transactions.
* **Automatic Bid Handling:** The system should support automatic bid handling mechanisms, such as setting maximum bid amounts or implementing bidding increments, to streamline the bidding process and prevent bid sniping.
* **Integration with Listing:** Bidding functionality should seamlessly integrate with the listing creation process, allowing sellers to specify whether their listings are open for bidding and setting starting bid prices or reserve prices if applicable.
* **Real-Time Updates:** Users participating in bidding should receive real-time updates on the status of their bids, including notifications for outbid scenarios or auction countdowns if applicable. Real-time updates enhance user engagement and facilitate timely decision-making.
* **Reporting and Dispute Resolution:** The platform should provide mechanisms for reporting suspicious or fraudulent bidding activities, as well as resolving disputes related to bidding, such as bid retractions or non-payment issues. Admins should have tools to investigate and take appropriate actions.

**Formats and Validity of Data:**

* **Input Data:** The system should enforce data validation rules to ensure that users enter information in the correct format (e.g., email addresses must contain "@" symbol). Error messages should be displayed for invalid input, guiding users to correct their entries.
* **Output Data:** Listings and other displayed data should be formatted in a visually appealing manner, with clear titles, descriptions, and images. The system should handle edge cases gracefully to prevent data rendering errors.

**User Interface Behavior:**

* **Responsive Design:** The user interface should adapt seamlessly to different screen sizes and devices, providing consistent functionality and aesthetics.
* **Intuitive Navigation:** Navigation menus, buttons, and links should be logically organized and labeled to help users find their way around the platform. Visual cues such as hover effects and animations can enhance usability.
* **Feedback Mechanisms:** The system should provide immediate feedback to users after they perform actions, indicating whether the action was successful or if there were any errors. Feedback messages should be clear and concise, guiding users on the next steps.

**Data Integrity and Security:**

* **Data Encryption:** Sensitive user data, such as passwords and payment information, should be encrypted using industry-standard encryption algorithms to prevent unauthorized access.
* **Backup and Redundancy:** Regular backups of the database should be performed to ensure data integrity and availability. Redundancy measures should be in place to mitigate the risk of data loss due to hardware failure or other incidents.
* **Access Control:** Role-based access control (RBAC) should be implemented to restrict access to sensitive data and functionalities based on user roles. Admins should have elevated privileges to manage data and settings.

**Regulatory Requirements:**

* **Data Privacy:** The platform should comply with data privacy regulations such as GDPR or CCPA, ensuring that user data is collected and processed lawfully and transparently. Users should be informed about how their data is used and have options to manage their privacy settings.
* **Consumer Protection:** Terms of service and privacy policies should be clearly communicated to users, outlining their rights and responsibilities. The platform should provide mechanisms for resolving disputes and addressing customer concerns.
* **Online Transactions:** The platform should implement secure payment processing mechanisms and comply with relevant regulations governing online transactions, such as PCI DSS compliance for handling credit card information.

**User Access/Authorization:**

* **Authentication Mechanisms:** The system should support various authentication methods, including username/password, social media login, and two-factor authentication (2FA). Strong password policies should be enforced to enhance security.
* **Role-Based Access Control (RBAC):** Different user roles (e.g., regular user, seller, admin) should have predefined sets of permissions, allowing them to access specific features and data based on their roles and responsibilities. Access controls should be granular and configurable by admins.

**Safety Requirements:**

* **Reporting and Blocking:** Users should have the ability to report inappropriate content or behavior, such as fraudulent listings or abusive users. The platform should provide mechanisms for blocking or flagging such content and taking appropriate action.
* **Secure Transactions:** Measures should be in place to protect users against fraud and ensure the security of transactions. This may include SSL encryption for data transmission, secure payment gateways, and fraud detection algorithms.
* **Compliance:** The platform should comply with safety standards and regulations applicable to online marketplaces, including regulations governing product safety and liability. Sellers should be required to adhere to safety guidelines when listing products for sale.

**3.3 Use Cases:**

* **View Products:** This allows users to browse and view the products available for purchase.
* **Purchase Products:** This is the main action that users can take on the platform, which is to purchase products.
* **Buyer:** This indicates that the platform is designed for buyers to purchase products.
* **Check Out:** This is the process of reviewing and confirming the details of a purchase before completing it.
* **Login:** This allows users to log in to their accounts on the platform.
* **Communication:** This suggests that there is a feature for users to communicate with each other or with the platform's support team.
* **Browse Products:** This is another way of saying "View Products".
* **Add to Cart:** This allows users to add products to their shopping cart before proceeding to checkout.
* **Seller:** This indicates that there are sellers on the platform who are offering products for sale.
* **Logout:** This allows users to log out of their accounts on the platform.
* **Feedback:** This suggests that there is a feature for users to provide feedback on their experience with the platform.

**3.4. Class/Objects**

* **User:**

Attributes: username, email, password, profile picture, location, account status

Methods: register(), login(), updateProfile(), deactivateAccount()

* **Item:**

Attributes: title, description, category, condition, images, owner (User object), location, exchange preferences

Methods: createListing(), editListing(), deleteListing()

* **Message:**

Attributes: sender (User object), receiver (User object), content, timestamp

Methods: sendMessage(), deleteMessage()

* **Notification:**

Attributes: recipient (User object), content, timestamp, read status

Methods: sendNotification(), markAsRead()

* **Admin:**

Attributes: username, email, password

Methods: login(), manageUsers(), manageListings(), resolveDisputes()

**3.5. Non-Functional Requirements**

**3.5.1. Performance:**

* **Response Time:** The system should respond to user interactions promptly, with page load times optimized to minimize waiting.
* **Scalability:** The application should be designed to handle increasing user loads without significant degradation in performance. This may involve implementing horizontal scaling techniques and optimizing database queries.
* **Throughput:** The system should be capable of handling a high volume of concurrent transactions and requests efficiently.
* **Resource Utilization:** The application should utilize system resources (CPU, memory, disk I/O) efficiently to ensure optimal performance.

**3.5.2. Reliability:**

* **Fault Tolerance:** The system should be resilient to failures, with mechanisms in place to recover gracefully from errors or crashes.
* **Error Handling**: Comprehensive error handling mechanisms should be implemented to detect, log, and handle errors encountered during system operation.
* **Data Integrity:** Measures should be in place to ensure the integrity of data stored in the system, with safeguards against data corruption or loss.
* **Redundancy:** Redundant components, such as backup servers and failover systems, should be deployed to ensure continuous operation in case of hardware or software failures.

**3.5.3. Availability:**

* **Uptime:** The system should strive for high availability, with minimal downtime for maintenance or upgrades. This may involve implementing rolling updates and deploying redundant infrastructure.
* **Monitoring:** Continuous monitoring should be in place to track system health and performance metrics, with automated alerts for potential issues or anomalies.
* **Disaster Recovery:** A robust disaster recovery plan should be established to mitigate the impact of catastrophic events, ensuring data integrity and minimal disruption to service.

**3.5.4. Security:**

* **Data Encryption:** Sensitive data should be encrypted both in transit and at rest to protect against unauthorized access.
* **Authentication and Authorization:** Strong authentication mechanisms should be implemented to verify the identity of users, with role-based access controls to enforce least privilege principles.
* **Data Privacy:** The system should comply with data privacy regulations, with measures in place to safeguard user privacy and protect personal information.
* **Vulnerability Management:** Regular security assessments and penetration testing should be conducted to identify and remediate potential vulnerabilities in the system.

**3.5.5. Portability:**

* **Platform Independence:** The application should be designed to run on various operating systems and hardware configurations, ensuring compatibility across different environments.
* **Containerization:** The use of containerization technologies (e.g., Docker) can enhance portability by encapsulating the application and its dependencies for easy deployment and migration.
* **Cloud Compatibility:** The system should be cloud-ready, with support for deployment on popular cloud platforms such as AWS, Azure, or Google Cloud. This enables flexibility in scaling and managing resources based on demand.

**3.6. Inverse Requirements**

**Performance Inverse Requirement:**

* The system should not have unreasonably long response times, exceeding X seconds for common user interactions.
* The system should not degrade in performance under heavy user loads, maintaining acceptable response times even during peak usage periods.

**Reliability Inverse Requirement:**

* The system should not experience frequent crashes or downtime, with a maximum allowable downtime of X hours per month.
* The system should not lose data integrity, ensuring that all user data remains accurate and consistent at all times.

**Availability Inverse Requirement:**

* The system should not have extended periods of unavailability, with a target uptime of X% over a specified time period.
* The system should not rely on single points of failure, ensuring redundancy and fault tolerance to minimize the risk of downtime.

**Security Inverse Requirement:**

* The system should not suffer from security vulnerabilities, with regular security assessments and timely patching of identified issues.
* The system should not compromise user privacy, adhering to data protection regulations and best practices for handling sensitive information.

**Portability Inverse Requirement:**

* The system should not be tied to specific hardware or software platforms, allowing for easy deployment across different environments.
* The system should not encounter compatibility issues when migrating between different hosting providers or operating systems.

**Usability Inverse Requirement:**

* The system should not present confusing or misleading user interfaces, ensuring clarity and consistency in design.
* The system should not require excessive user training or documentation to use effectively, prioritizing intuitive usability.

**Scalability Inverse Requirement:**

* The system should not encounter significant performance degradation as the user base grows, maintaining responsiveness and throughput.
* The system should not have prohibitive scaling costs, with scaling strategies designed to be cost-effective and efficient.

**Interoperability Inverse Requirement:**

* The system should not have compatibility issues with third-party integrations or APIs, ensuring smooth interoperability with external systems.
* The system should not restrict users from accessing their data or using the platform from different devices or platforms.

**3.7. Design Constraints**

**Technology Constraints:**

* **Programming Languages and Frameworks:** The project may be constrained to use specific programming languages (e.g., JavaScript, Python, Java) and frameworks (e.g., React, Django, Spring) based on the expertise of the development team or compatibility requirements.
* **Database Systems:** Constraints may exist regarding the choice of database systems (e.g., MySQL, PostgreSQL, MongoDB) based on factors like scalability, performance, and data modeling requirements.
* **Third-Party Integrations:** The project may need to integrate with external APIs or services, and there may be constraints on the availability or compatibility of these integrations.

**Resource Constraints:**

* **Budget:** There may be budget constraints that limit the resources available for development, including hardware, software licenses, and external services.
* **Time:** The project may be subject to tight deadlines, requiring efficient resource allocation and prioritization of features.
* **Human Resources:** Constraints may exist regarding the availability and expertise of team members, influencing the design and implementation approach.

**Regulatory and Compliance Constraints:**

* **Data Privacy Regulations:** The project may need to comply with data privacy regulations such as GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act), imposing constraints on data handling and security measures.
* **Payment Processing Regulations:** If the application involves financial transactions, there may be regulatory constraints related to payment processing and security standards (e.g., PCI DSS compliance).

**Performance Constraints:**

* **Scalability:** The application may need to accommodate a certain number of concurrent users or a specified growth rate, imposing constraints on scalability and performance optimization.
* **Response Time:** There may be requirements for minimum acceptable response times for user interactions, constraining design decisions related to system architecture and optimization.

**User Experience Constraints:**

* **Accessibility:** The application may need to comply with accessibility standards (e.g., WCAG) to ensure equal access for users with disabilities, imposing constraints on design and development practices.
* **Localization:** Constraints may exist regarding the support for multiple languages and cultural preferences, influencing the design of user interfaces and content management systems.

**Security Constraints:**

* **Data Encryption:** Constraints may require the encryption of sensitive data at rest and in transit to ensure compliance with security standards and protect user privacy.
* **Authentication and Authorization:** There may be constraints on the implementation of secure authentication mechanisms and access controls to prevent unauthorized access to sensitive information.

**Operational Constraints:**

* **Infrastructure Requirements:** Constraints may exist regarding the deployment environment, such as limitations on available hardware resources or hosting providers.
* **Maintenance and Support:** Constraints may require the design of the system to be easily maintainable and supportable, with considerations for future updates, bug fixes, and user support.

**3.8. Logical Database Requirements**

**User Management:**

* **User profiles:** Store information about users such as username, email, password (hashed), and contact details.
* **Authentication tokens:** If implementing token-based authentication, you'll need to store tokens securely.
* **User roles and permissions:** Define user roles (e.g., buyer, seller, admin) and their corresponding permissions.

**Product Catalog:**

* **Product information:** Store details about products such as name, description, price, category, and availability.
* **Images:** Store images associated with products.
* **Inventory management:** Track the quantity of each product available for sale.

**Transactions:**

* **Order history:** Store information about orders, including the products purchased, quantity, price, timestamps, and customer details.
* **Payment details:** If handling payments within the application, securely store payment information, such as payment method, transaction ID, and status.
* **Shipping details:** If applicable, store information related to shipping, including addresses, shipping methods, and tracking numbers.

**Reviews and Ratings:**

* Allow users to leave reviews and ratings for products.
* Store review content, ratings, timestamps, and associated product IDs.

**Search and Filtering:**

* Implement features for users to search and filter products based on various criteria, such as category, price range, and keywords.
* Ensure efficient indexing and querying to support fast and accurate search results.

**Security and Privacy:**

* Implement measures to secure sensitive data, such as encryption for passwords and payment information.
* Ensure compliance with data protection regulations, such as GDPR or CCPA, by providing mechanisms for data access control and user consent.

**Performance Optimization:**

* Normalize the database structure to minimize redundancy and improve data integrity.
* Use indexing and caching techniques to enhance query performance.
* Consider scalability requirements and design the database to handle potential increases in data volume and user traffic.

**Backup and Recovery:**

* Implement regular backups to prevent data loss in case of system failures or disasters.
* Define procedures for data recovery to restore the database to a consistent state in the event of corruption or data loss.

**Analytics and Reporting:**

* Incorporate features for generating reports and analyzing user behavior, sales trends, and other metrics.
* Store relevant data for analytics purposes, such as user activity logs and sales data.

**3.9. Other Requirements**

**Data Management Requirements**

* **Data Quality:** Requirements ensuring accuracy, completeness, and timeliness of the data used and generated by the platform.
* **Data Archiving:** Strategies and requirements for data retention, archiving processes, and data purging schedules.
* **Data Privacy:** Protocols for ensuring the privacy of user data, including encryption, anonymization techniques, and compliance with data protection laws.

**User Experience (UX) Requirements**

* **Navigation:** Requirements detailing the ease of navigating through the application interfaces.
* **Response Time:** User interface response times must be quick to ensure a seamless experience for high-frequency traders.
* **Feedback Mechanisms:** Real-time feedback on user actions, system errors, and confirmations.

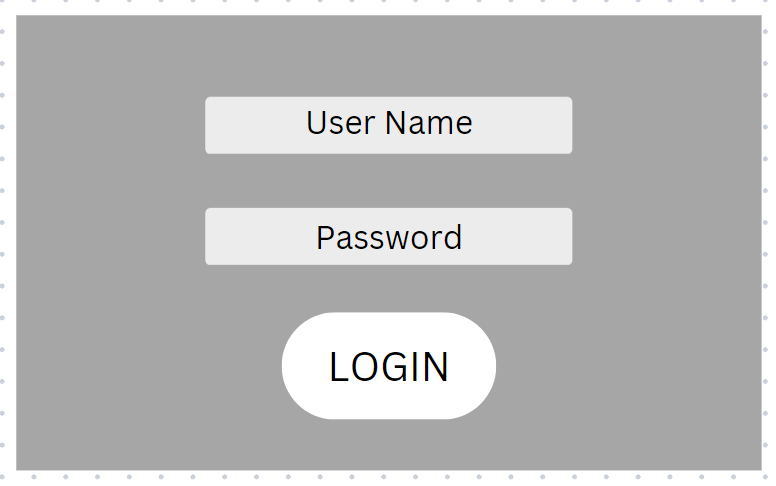
**Testing Requirements**

* **Unit Testing:** Guidelines for developing unit tests to cover individual components or functions.
* **System Testing:** Comprehensive testing to verify that the entire application meets the specified requirements.
* **Performance Testing:** Detailed requirements for testing application performance under various loads.
* **Security Testing:** Specific requirements for security testing, including penetration testing and vulnerability assessments.

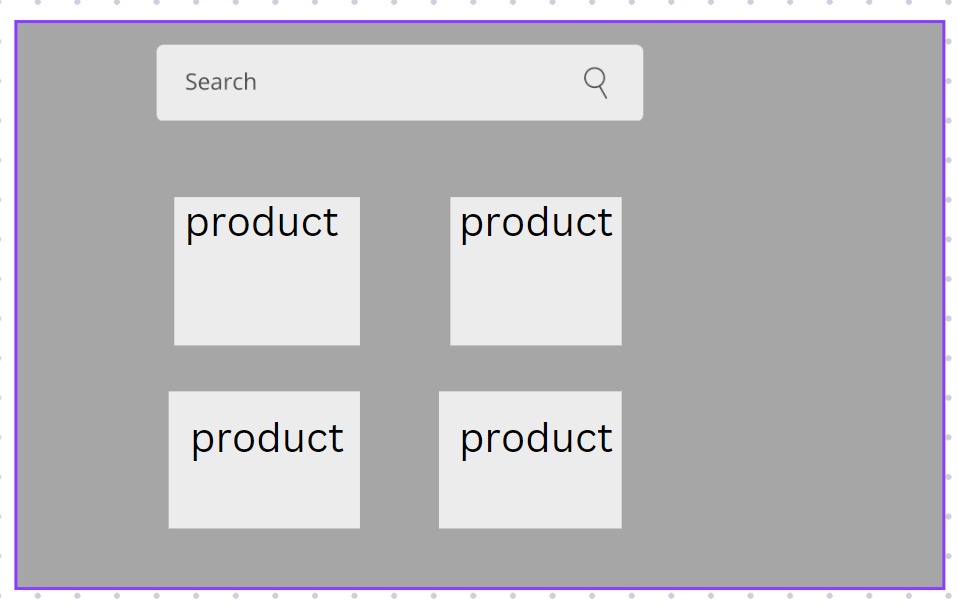
**Deployment Requirements**

* **Environment Setup:** Specifications for setting up development, testing, staging, and production environments.
* **Deployment Process:** Step-by-step process for deploying the application, including rollbacks and version control.
* **Continuous Integration/Continuous Deployment (CI/CD):** Requirements for CI/CD pipelines to automate the deployment process and ensure high-quality code.

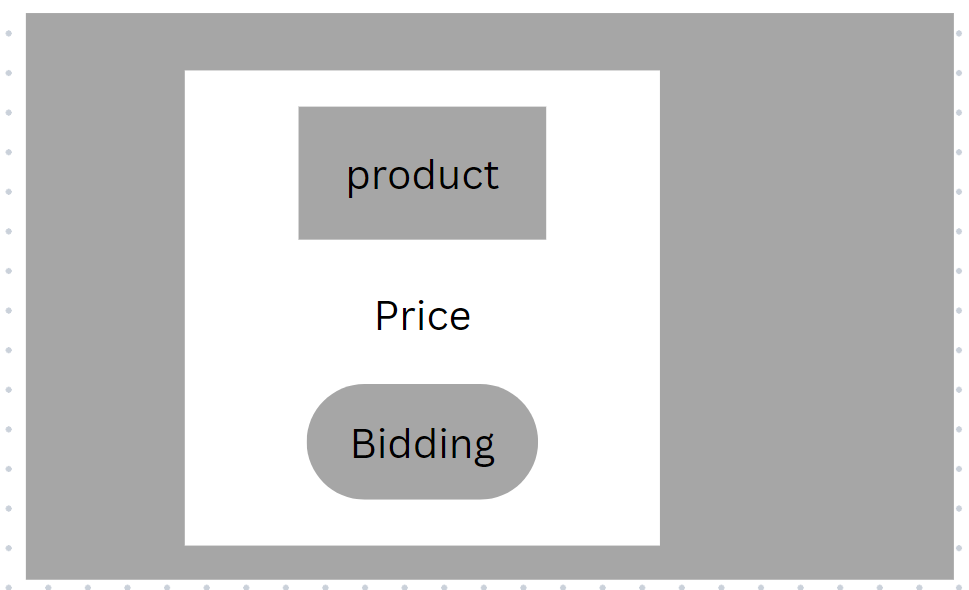
**3.10. Prototypes (for complete project)**

**User Registration Screen:**

**Listing Creation Screen:**

****

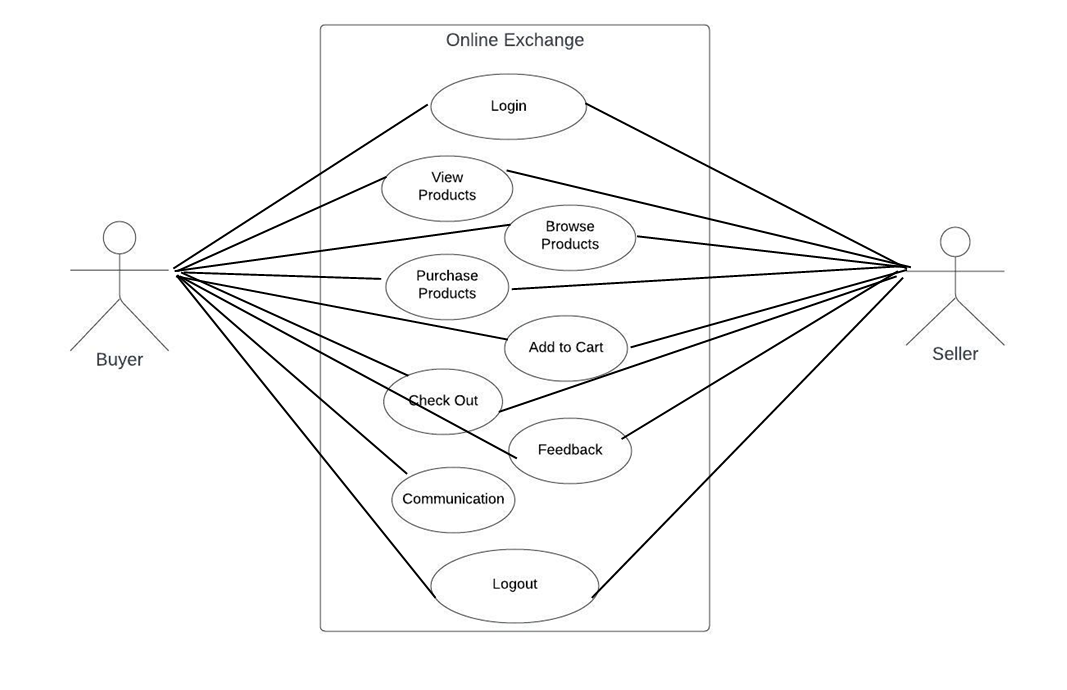
**Bidding Screen:**

****

**Rating and Reviews Screen:**

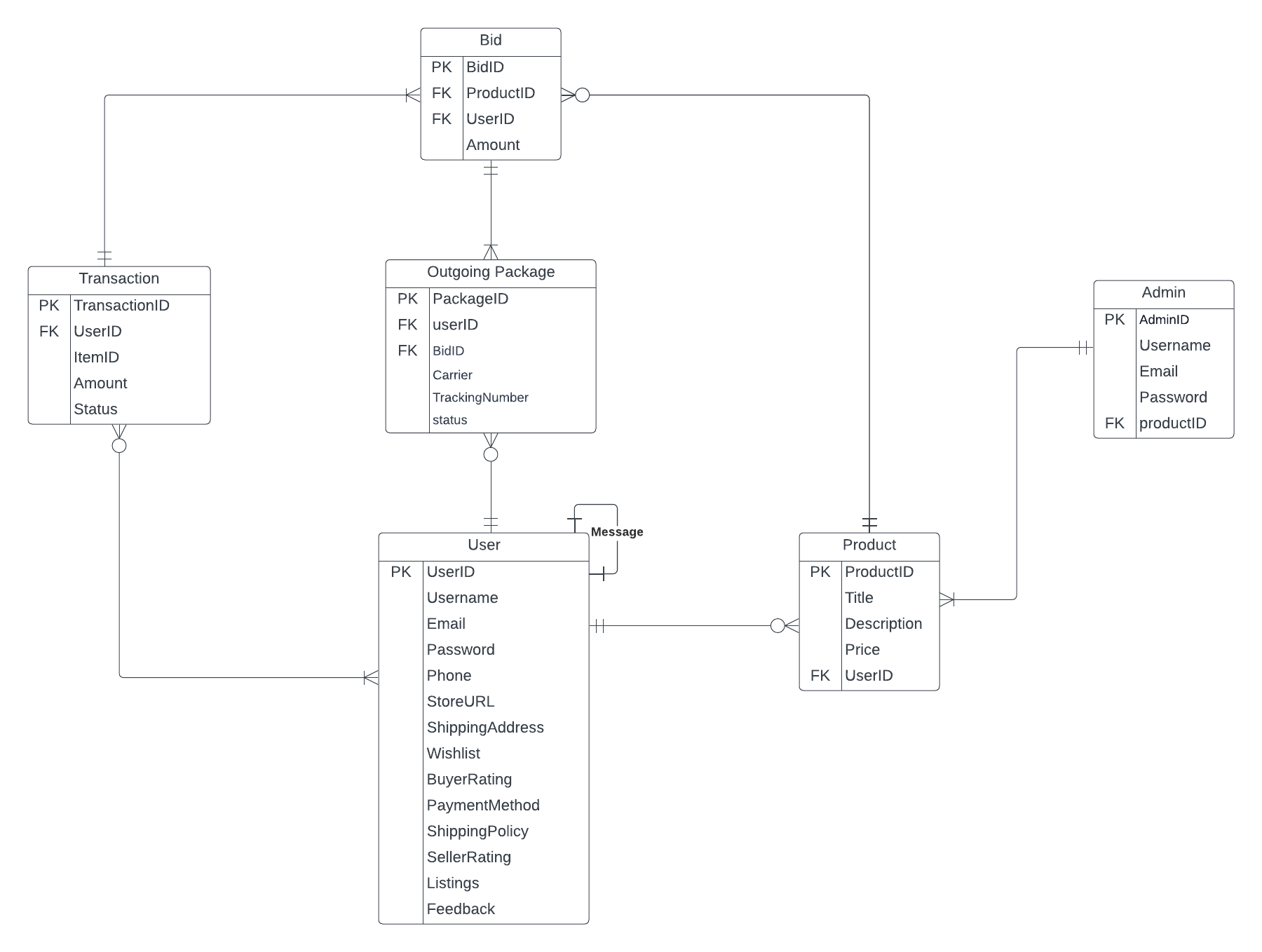
****

**3.11. Use Case Diagram:**

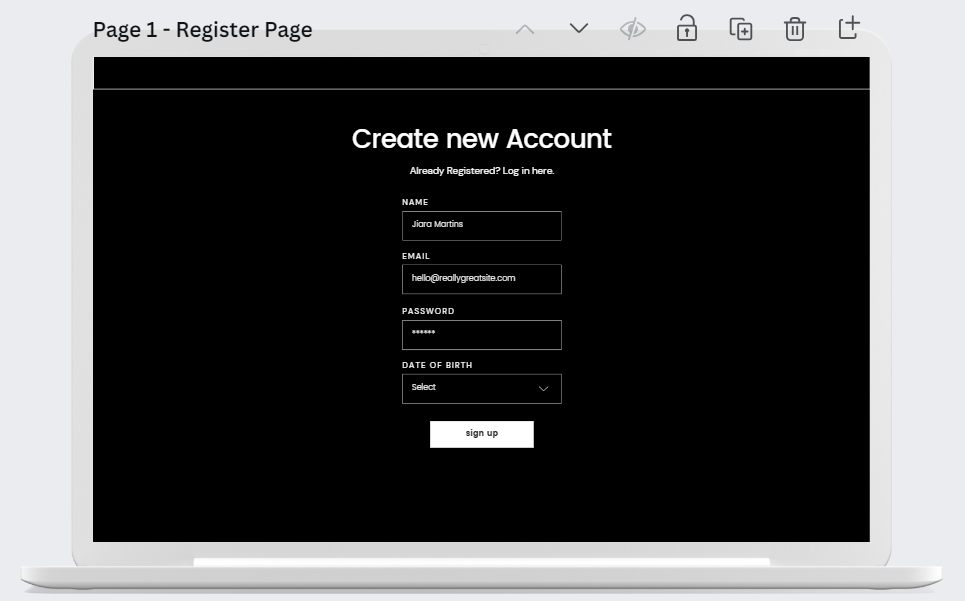
****

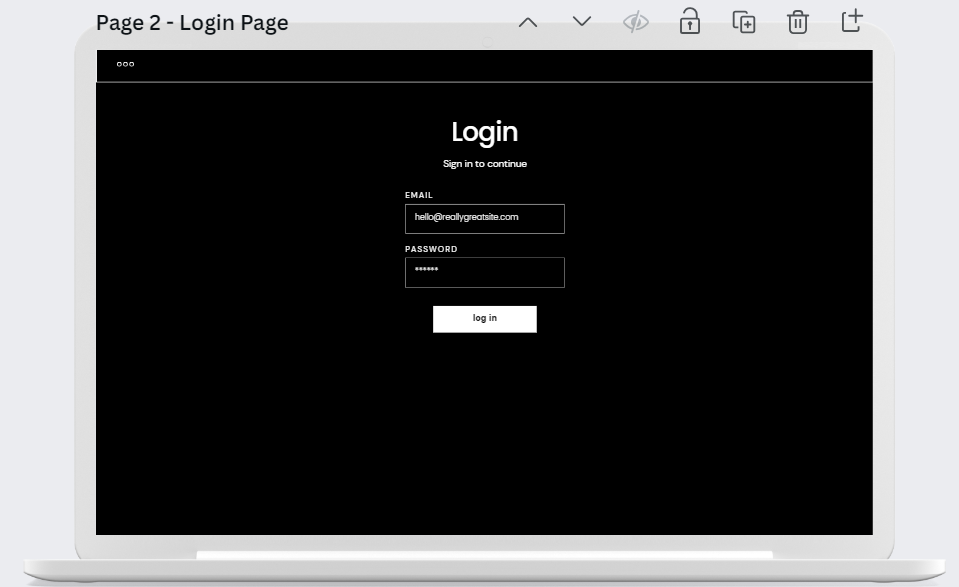
**4. Design**

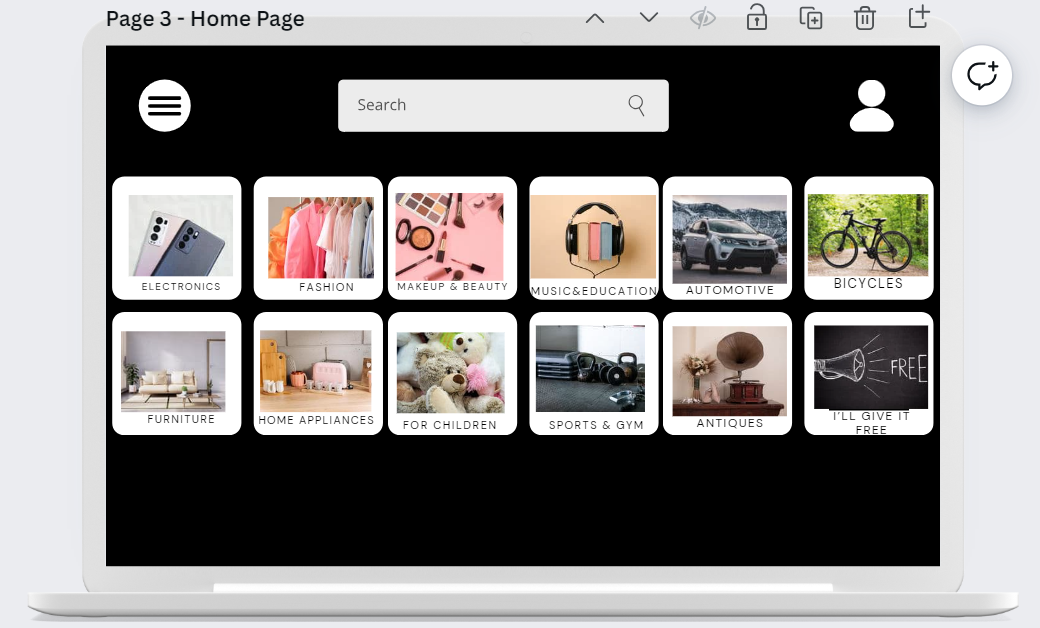
**4.1. ER diagram**

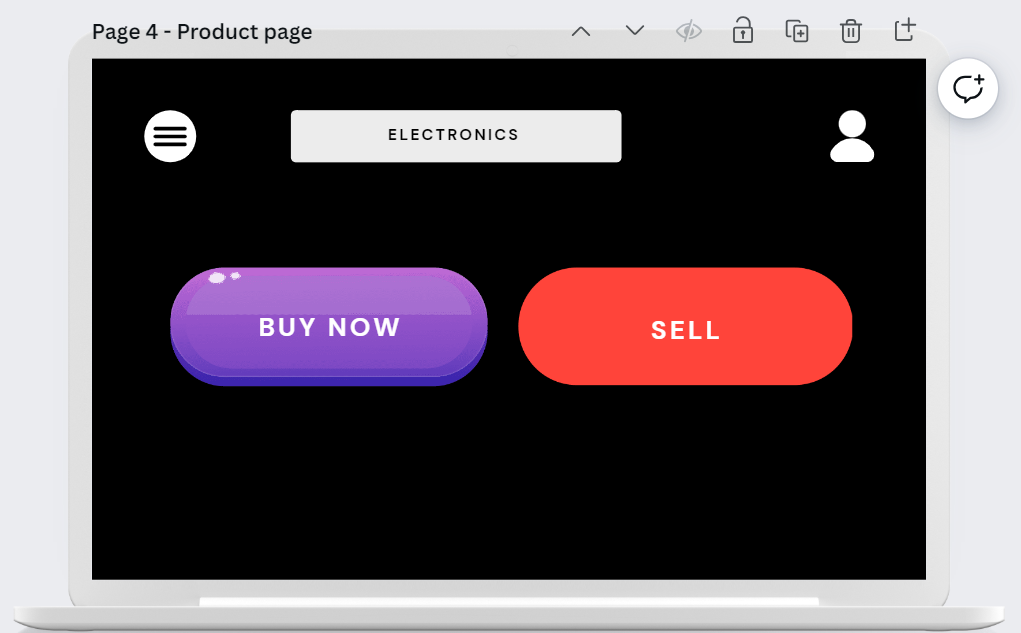


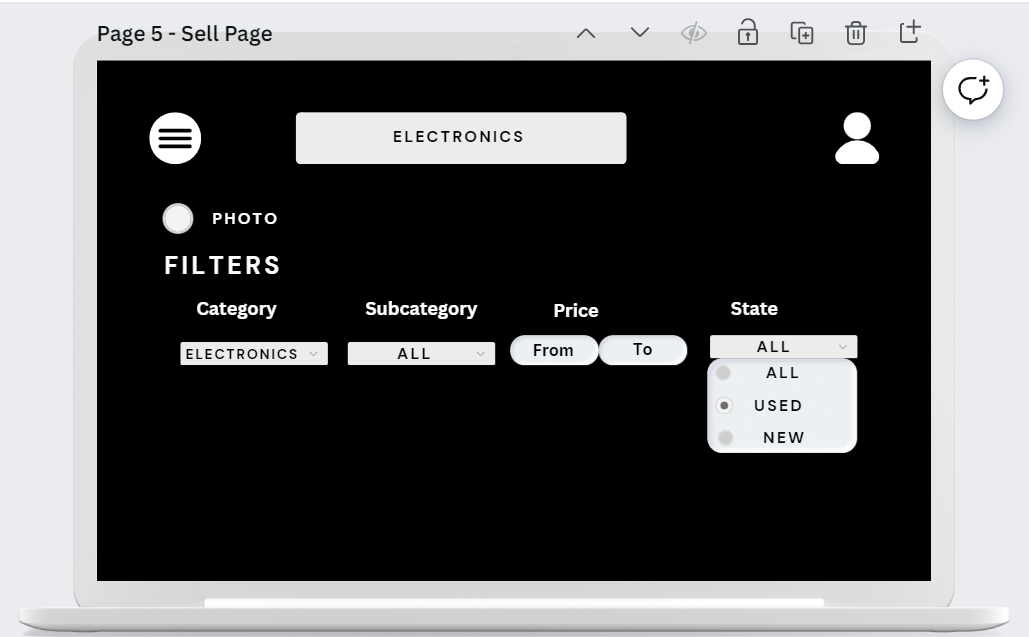
**4.2. GUI**

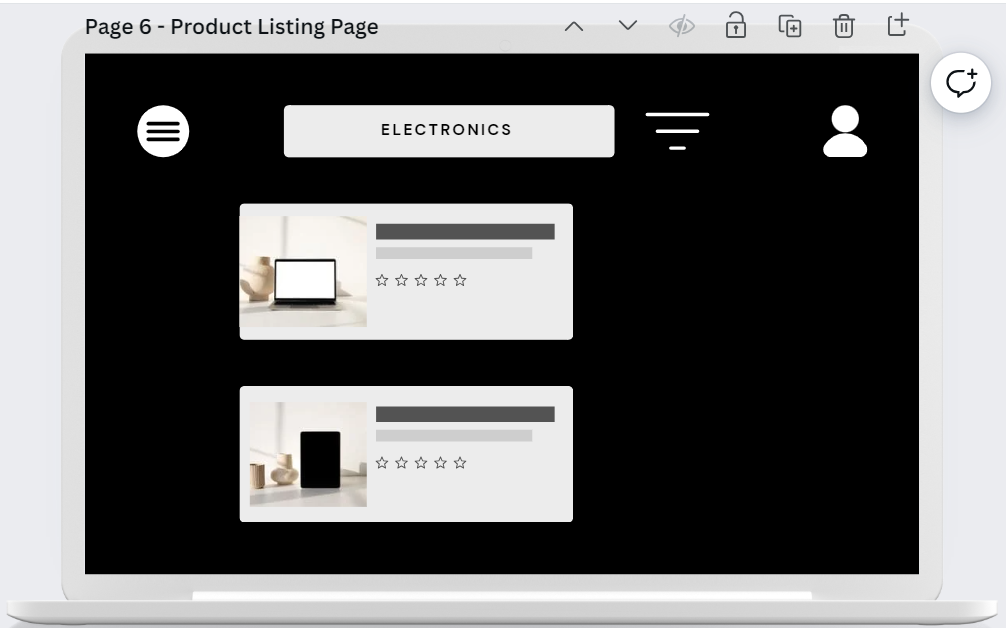
****

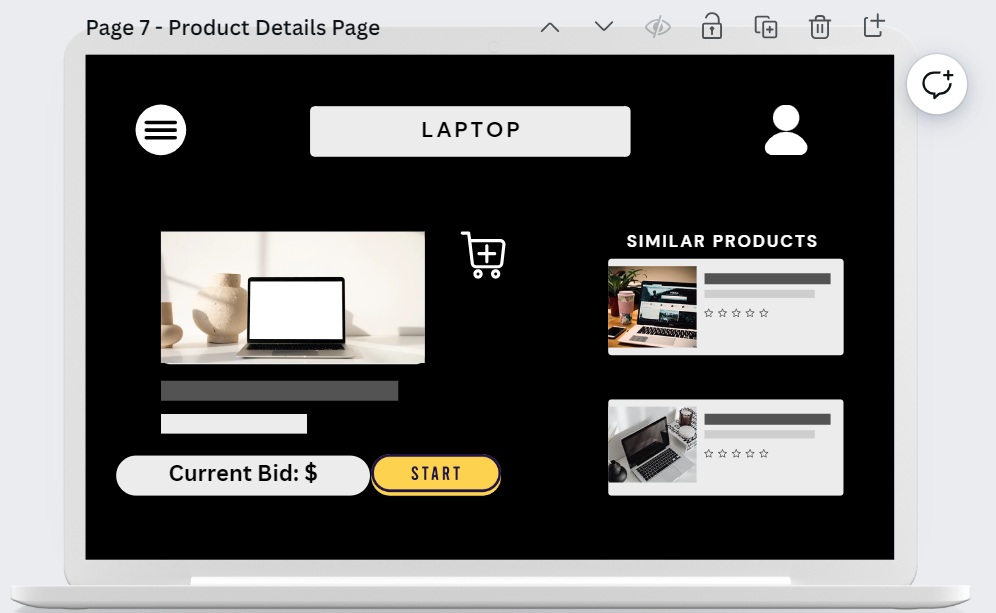
****

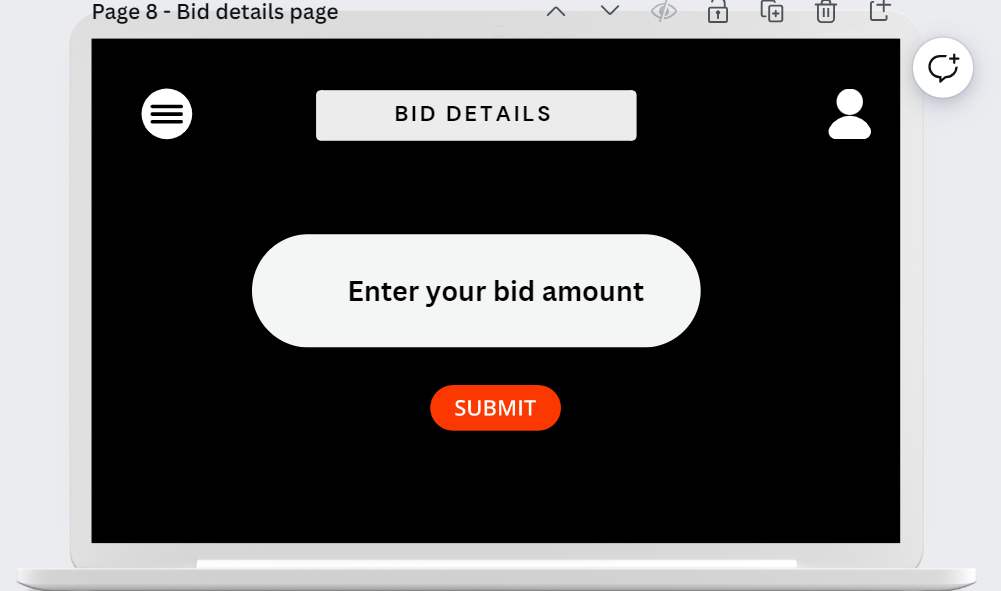
****

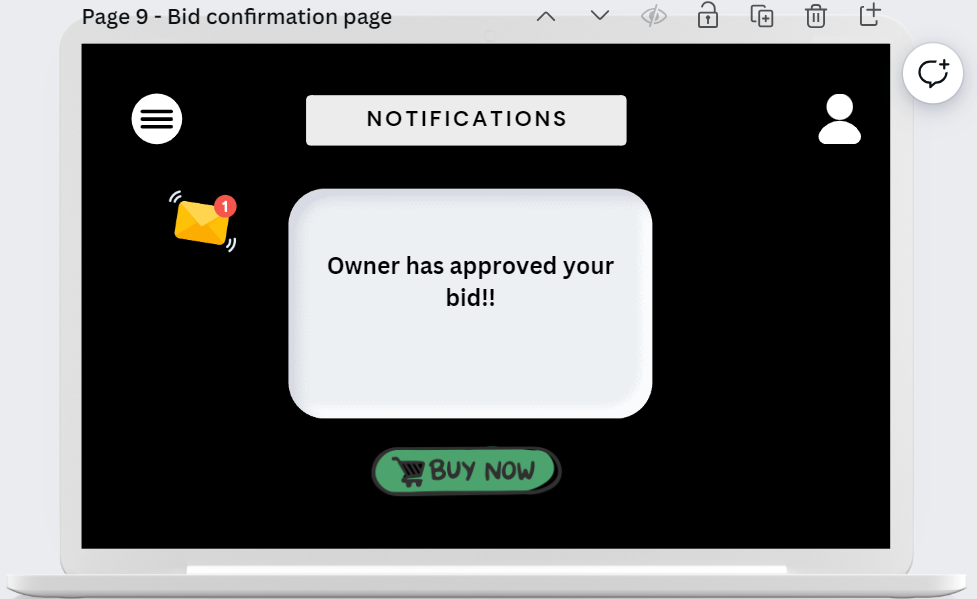
****

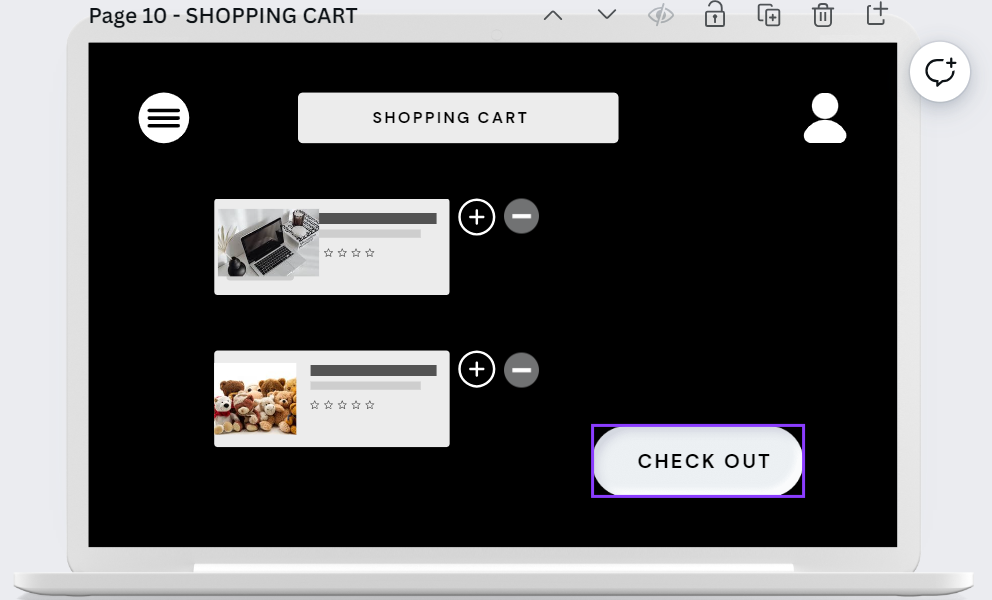
****

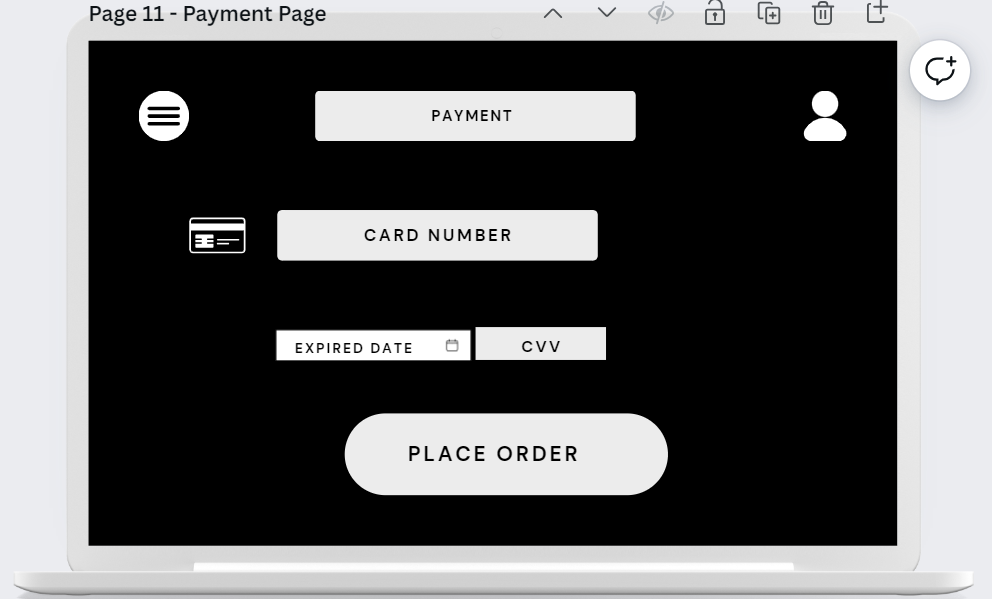
****

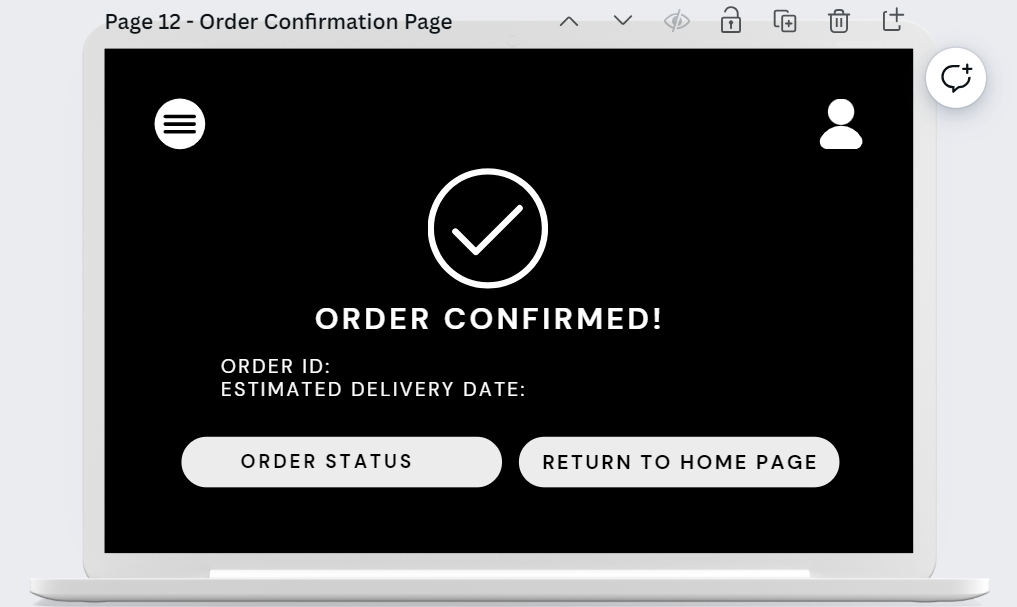
****

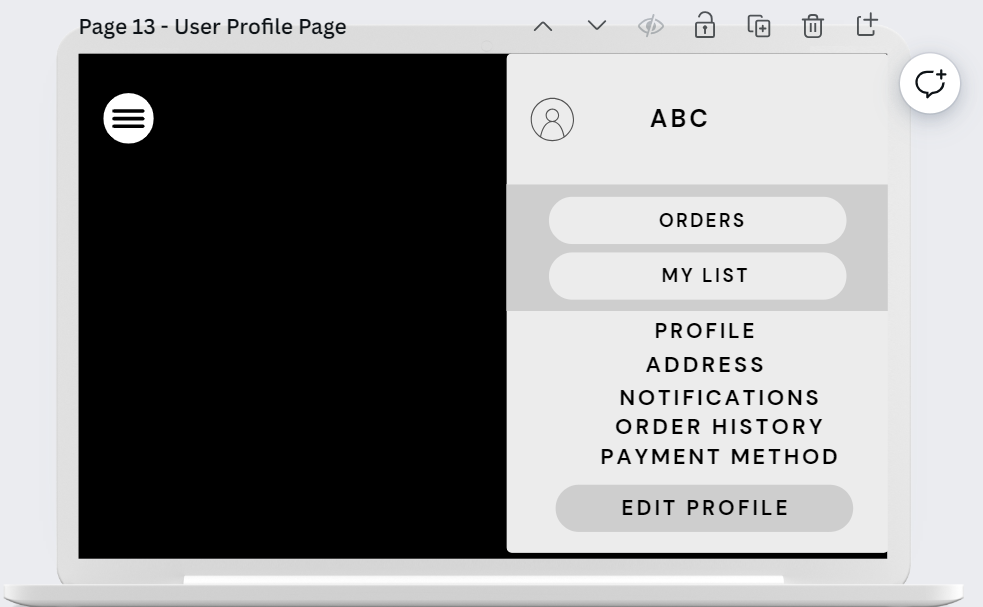
****

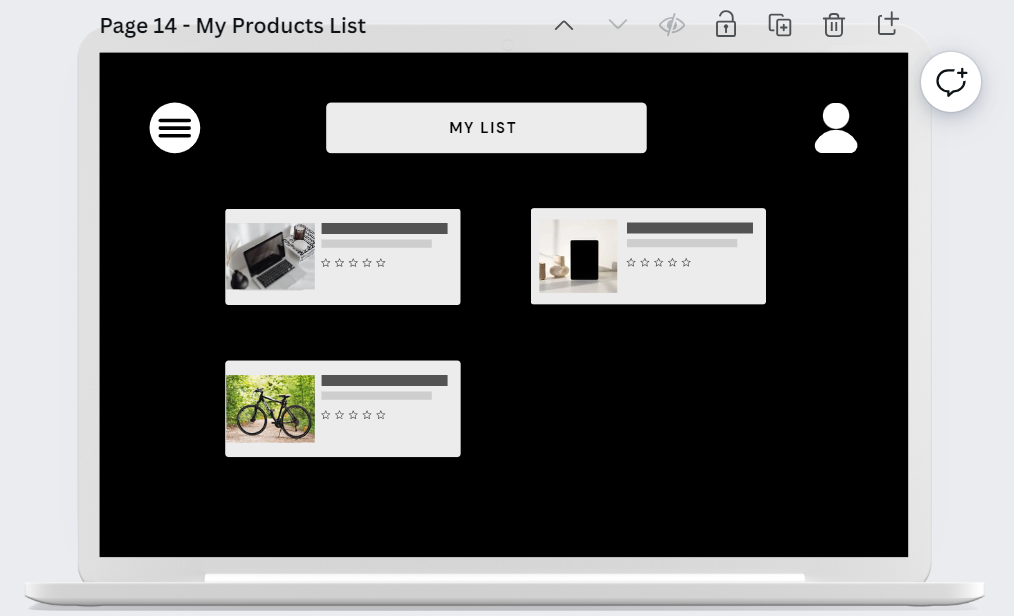
****

****

****

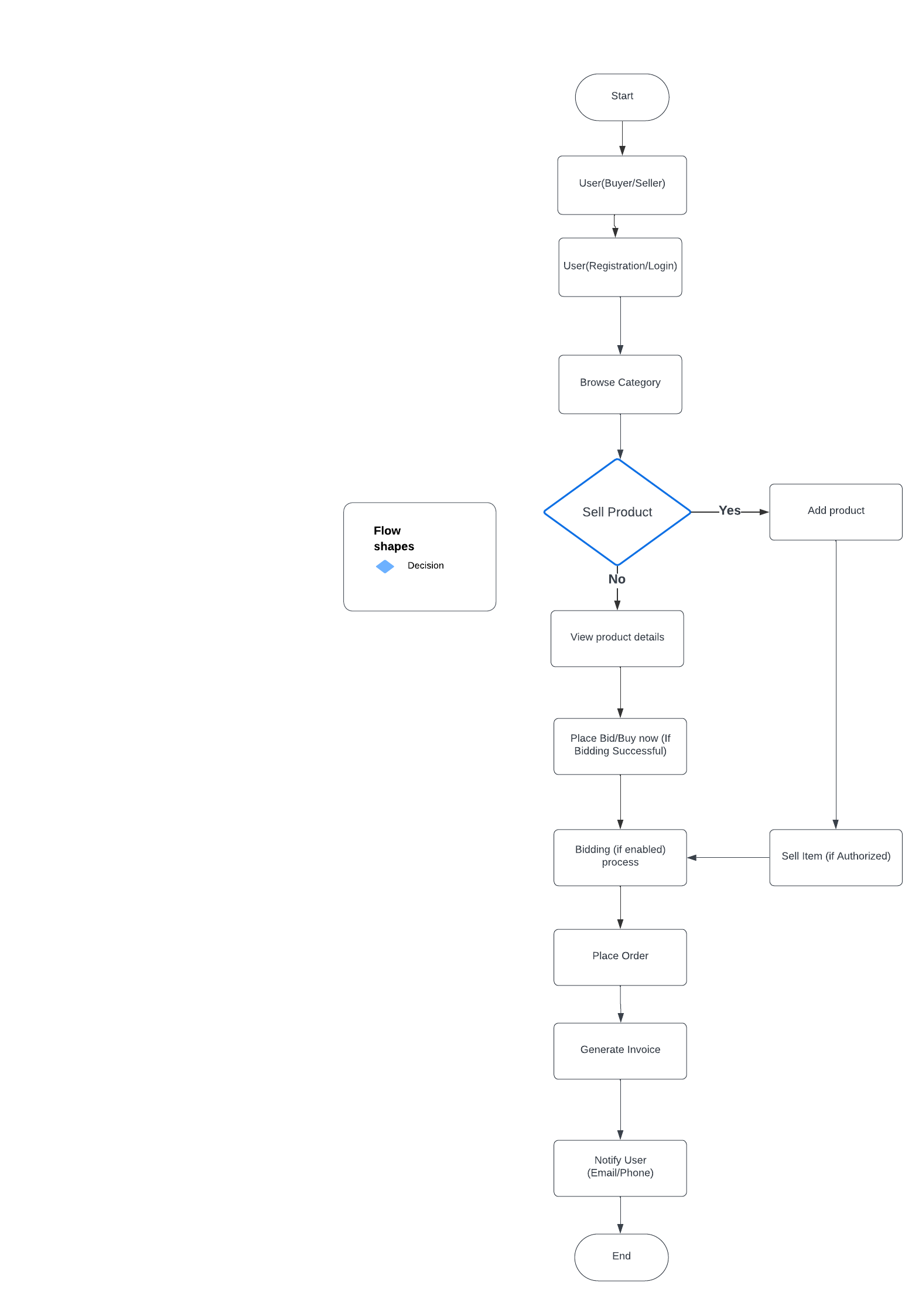
****

****

****

5. Analysis Models

5.1. Data Flow Diagram



5.2. Sequence Diagram

