Software Design Specification

for

Swappify

Version 1.0 approved

Prepared by

Rohail Rathore

Mustan Ali

SZABIST

19th December 2024

Table of Content

Table of Tables

Table of Figures

1. Introduction

1.1 Purpose of this document

The purpose of this document is to provide a detailed description of the system, including its functionality and underlying assumptions. It is designed to give the intended audience comprehensive insights into these aspects. Additionally, the document outlines the system's software architecture and explains the sequence or workflow for specific tasks. This information is intended to facilitate understanding and ease of use for both stakeholders and system developers.

1.2 Scope of the development project

This project is focused on creating a digital platform that offers a reliable and user-friendly environment for barter trading, where users can easily exchange goods without involving money. The project scope includes the development of several key features to enhance user experience and community trust.

- **Advanced Search:** Users can filter trade items by category to quickly find specific goods.
- **User Authentication:** Includes verification processes to ensure user security and trustworthiness.
- **Trade Proposal System:** Enables users to propose, negotiate, and finalize trades directly with others.
- **Wishlist:** Users can bookmark items of interest and share them with friends for future reference.
- **Review & Rating System:** Users can rate and review trading partners to build community trust.
- **Messaging Feature:** An integrated messaging system allows users to communicate & negotiate trade in real-time.

By focusing on these key areas, Swapify aims to provide a secure, efficient, and transparent barter trading experience that promotes sustainability and community engagement. The platform will concentrate on creating a user-friendly environment where trust and value are prioritized over monetary transactions.

1.3 Definitions, acronyms, and abbreviations

- **Barter Trading:** A system of exchange in which goods or services are traded directly for other goods or services without using money.
- **HTTP:** Hypertext Transfer Protocol, the foundation of data communication on the web.
- **HTTPS:** Hypertext Transfer Protocol Secure, an extension of HTTP that uses encryption for secure data transmission.
- **Data Encryption:** The process of converting data into a coded format to prevent unauthorized access.
- **User Authentication:** The process of verifying the identity of a user trying to access the system.

1.4 References

- React Documentation https://react.dev/learn
- Node.js v22.9.0 Documentation https://nodejs.org/docs/latest/api
- Express 5.x API Reference https://expressjs.com/en/5x/api.html
- MongoDB Documentation https://www.mongodb.com/docs

1.5 Overview of Document

- **Section 1.0**: An introduction to the document, explaining its purpose, providing definitions, acronyms, abbreviations, references, and a brief overview of each section within the document.
- **Section 2.0**: A comprehensive description of the system architecture, including general constraints, data design, program structure, and alternative models considered.
- **Section 3.0**: Detailed descriptions of each system component, covering their purpose, dependencies, interfaces, resources, processing logic, and data structures, to guide the implementation process.
- **Section 4.0**: A focused discussion on user interface design, detailing the interface design rules, GUI components, and a visual or descriptive representation of the user interface.

- **Section 5.0**: An exploration of reuse strategies, including leveraging existing modules, tools, and open-source resources, and discussing their role in the product's design and implementation.
- **Section 6.0**: Documentation of key design decisions and trade-offs, providing insight into the reasoning behind the chosen design and explaining ideas that were not implemented.
- **Section 7.0**: Pseudocode for components, offering clear and structured logic for system functionality to aid in development and testing.
- **Section 8.0**: Appendices containing supplementary materials, including class diagrams, sequence diagrams, and other visual aids that provide additional context and clarity for the system design.

2. System Architecture Description

2.1. Section Overview

This section outlines the components and subsystems of Swappify, including the system's data design. It covers the system requirements, hardware and software environments, limitations, an ERD diagram, the architectural model with its major components, and alternatives to the chosen architectural model.

2.2. General Constraints

The goal is to develop an application capable of handling large datasets while ensuring a seamless user experience for both sending and receiving responses. The application is designed to be compatible with all major web browsers, focusing on accessibility and responsiveness. Additionally a key constraint is the use of cloud deployment for the application. This approach is intended to optimize performance and ensure efficient handling of data without compromising the user experience.

2.3. Data Design

The system uses MongoDB as its NoSQL database, where data is stored in collections consisting of documents. Each collection contains related documents, with each document representing a specific data entity. These collections are designed to store various types of data and allow for flexible, scalable storage of unstructured information. External files in CSV and JSON formats are used for data import and export. For a detailed diagram of the database structure, refer to the **ERD** in **Section 8** of the appendices.

2.4. Program Structure

The system follows a client-server architecture design. The client-side is built using React JS, providing a dynamic interface that interacts with the server side via API requests. The server side is powered by Node JS and Express JS, handling business logic and database interactions with MongoDB. The system utilizes several key classes that represent the major components and entities in the application. Each class is responsible for handling specific functionality and interactions within the system. These classes work together to ensure proper data handling and application functionality. For a detailed visual representation, refer to the **Class-Diagram** in **Section 8** of the appendices.

2.5. Alternatives Considered

No alternative architectural models were considered. The client server architecture design was chosen for its scalability, maintainability, and efficient data management. This model ensures a clear separation between the user interface and server side logic, providing flexibility and long term performance.

3. Detailed Description of Components

3.1. Section Overview

This section outlines the foundation for implementing the system. In Section 3.2, we provide a comprehensive description of the components, including their details, data members, methods, and a structured template for each component.

3.2. Component and Detail

3.2.1. Login

Identification	Login
Type	Module
Purpose	Its purpose is to authenticate users by validating their email and password, ensuring that only authorized users can access their accounts.
Function	The Login module processes user inputs (email and password), validates the email format, and checks for empty fields. It authenticates users by comparing credentials with the database.
Subordinates	The module consists of the login form, authentication logic, and error handling, fulfilling requirements for displaying forms, validating credentials, and showing error messages.
Dependencies	It depends on the user database & email validation.
Interfaces	The module interacts with the user through the login form and communicates with the database for authentication. Error messages are displayed for invalid inputs.
Resources	The Login module requires server resources, a database for checking user credentials, and hashing services for password verification.
Processing	The module validates the email and password, compares them with database records, and either authenticates the user or shows an error message.
Data	The module processes user email and password for authentication, ensuring proper formatting and verification against stored credentials.

Table 1: Login Component Detail

3.2.2. SignUp

Identification	Signup
Type	Module
Purpose	It allows new users to register by submitting their name, email, password, and other details to create an account.
Function	The system validates the input, checks if the email is already registered, saves user data, and sends a verification email. Once the user clicks the verification link, the account is activated, and they are redirected to the login page.

Subordinates	The module includes the sign-up form, input validation, database storage, and email verification.
Dependencies	It depends on the user database, email validation & account activation.
Interfaces	The module interacts with the user via the sign-up form, with the database for storing information, and the package to send verification links.
Resources	It requires server resources, a database for checking user credentials, and hashing services for password verification.
Processing	The system validates inputs, checks for existing emails, stores data, sends a verification email, and activates the account after verification.
Data	The system processes the user's name, email, password, and mobile number, storing them in the database if the email is not already registered.

Table 2: SignUp Component Detail

3.2.3. Forgot Password

Identification	Forgot Password
Type	Module
Purpose	It enables users to securely reset their password by receiving a generated password via their registered email, ensuring account recovery through email verification and link validation.
Function	The system provides a form to enter the registered email, checks if the email exists, sends a new password if valid, and authenticates the user after the password reset.
Subordinates	The module includes email validation, reset link generation, password updating, and user authentication.
Dependencies	It depends on the user database for email validation, an email service to send reset links, and a hashing service for secure password storage.
Interfaces	The module interacts with users through the "Forgot Password" form, with the database for email verification, and the email service to deliver reset links.
Resources	It requires server resources to process requests, a database for validating registered emails, and an email service to send reset links.
Processing	The system verifies the user's email, generates a secure reset password link, sends it to the user's registered email, and updates the password in the database upon reset.
Data	The system processes the user's registered email and stores the new password securely after resetting, ensuring data integrity and account security.

Table 3:Forgot Password Component Detail

3.2.4. Profile Management

Identification	Profile Management

Type	Module
Purpose	The purpose is to enable users to modify their profile details, ensuring their information is up-to-date for effective communication and accurate platform interactions.
Function	The user navigates to their profile, edits fields like name, email, and phone number, and saves the changes. The system validates the updated information, saves it, and displays the new details. If invalid data is entered, the system shows error messages.
Subordinates	The module includes the profile page, edit functionality, and input validation.
Dependencies	It depends on the user account database to store and retrieve profile information.
Interfaces	The system interacts with the user via the profile page and the "Edit" functionality to update information.
Resources	The module requires server resources for updating and storing profile data.
Processing	The system allows users to update their profile, validates the input, and saves the changes to the database.
Data	The system processes and stores updated user information, such as name, email, and phone number.

Table 4: Profile Management Component Detail

3.2.5. List Item

Identification	List Item
Type	Module
Purpose	It enables users to list new items for trade by providing essential details, expanding item inventory and fostering trade opportunities.
Function	The system allows users to fill out a form with item details, validates the inputs, and saves the item to the database. It provides feedback for successful listing or errors for invalid inputs.
Subordinates	The module includes the item listing form, input validation, database storage, and feedback messages.
Dependencies	It relies on the database for storing item details, file validation for images, and user authentication for account access.
Interfaces	The module interacts with the user through the "Add New Item" form and connects with the database for storing validated item details.
Resources	It requires server resources for processing item details, a database to store listings, and validation tools for verifying input formats.
Processing	The system verifies the input data, validates mandatory fields and file formats, stores the new item in the database, and displays a confirmation or error message based on the outcome.
Data	The system processes and stores item details provided by the user, including item title, description, category, and image files.

Table 5: List Item Component Detail

3.2.6. Edit Item

Identification	Edit Item
Type	Module
Purpose	It allows users to modify the details of an existing item they have listed for trade, ensuring that item listings are accurate and up-to-date..
Function	The system displays the current item details in an editable form, validates the updated information, saves the changes to the database, and provides confirmation upon successful updates.
Subordinates	The module includes the item editing form, input validation checks, and database storage for updated item details.
Dependencies	It depends on the user account system for access, the item database for storing and retrieving item details, and validation services for checking mandatory fields and data accuracy.
Interfaces	The module interacts with users via the item editing form, the database for saving updates, and error handling for invalid inputs.
Resources	It requires server resources to process the edits, a database to store the updated item details, and validation services for ensuring the data is correct.
Processing	The system processes the updated item details, validates the input, saves the new information to the database, and displays a confirmation message after successful updates.
Data	The system processes the item's updated details such as title, description, price, and images, ensuring they meet the required formats and validations before saving them in the database.

Table 6: Edit Item Component Detail

3.2.7. Delete Item

Identification	Delete Item
Type	Module
Purpose	It allows users to remove an existing item they have listed for trade, ensuring the inventory is kept accurate and updated by enabling users to manage their listings effectively.
Function	The system displays a button to delete an item, removes the item from the listings upon user confirmation, updates the database, and provides a confirmation message.
Subordinates	The module includes the delete button, the action to remove the item from the database, and the confirmation message displayed to the user.
Dependencies	It depends on the user account system for access, the item database for deleting and retrieving item details, and confirmation services for ensuring the item is successfully removed.

Interfaces	The module interacts with users via the item deletion button, the database for removing the item, and error handling for system failures.
Resources	It requires server resources to process the item removal, a database to update listings, and validation services to ensure the correct item is deleted.
Processing	The system processes the request to delete an item, removes the item from the user's listings, updates the database, and displays a confirmation message to the user.
Data	The system processes the item's details and deletes the selected item from the database after confirming the action.

Table 7: Delete Item Component Detail

3.2.8. View Item Listing

Identification	View Item Listing
Type	Module
Purpose	The purpose is to enable users to view all items available for trade, helping them make informed trading decisions and enhancing their engagement with the platform.
Function	The user navigates to the item listings page, and the system displays all available items with basic details. The user can select an item to view its full details, including descriptions, images, and trade options. If no items are available, the system displays a message.
Subordinates	The module includes item listing display and detailed view functionality.
Dependencies	It depends on the item database to retrieve and display available items.
Interfaces	The system interacts with the user through the item listings page and detailed item view.
Resources	The module requires server resources to retrieve and display item data.
Processing	The system processes the user's request to view available items and their details.
Data	The system handles data related to item listings, including basic details and full descriptions.

Table 8: View Item Listing Component Detail

3.2.9. Search & Filter

Identification	Search & Filter
Type	Module
Purpose	Its purpose is to enhance the user experience by enabling users to easily find relevant items based on search terms or selected filters (e.g., category, condition).
Function	The user enters a keyword or selects filter options, and the system retrieves and displays matching items. If no items match, the system displays a message. Users can then select an item for detailed information.

Subordinates	The module includes the search bar, filter options, item listings display, and message display for no results.
Dependencies	It depends on the item database to retrieve and filter item listings.
Interfaces	The system interacts with the user through the search bar and filter options and displays filtered results on the item listings page.
Resources	The module requires server resources to process searches, filter data, and display results.
Processing	The system processes the user's search or filter inputs, retrieves matching items, and displays them. If no results are found, it shows a message.
Data	The system processes search keywords and filter criteria to query and display relevant item listings.

Table 9: Search & Filter Component Detail

3.2.10. Add to Wishlist

Identification	Add to Wishlist
Type	Module
Purpose	It allows users to save items of interest to their wishlist for future reference, enabling easier access to desired items without repeated searching.
Function	The system allows users to add items to their wishlist, displays a confirmation message, and shows the saved items when the user navigates to the wishlist section.
Subordinates	The module includes the "Add to Wishlist" button, the wishlist section in the user profile, and the process of adding and removing items from the wishlist.
Dependencies	It depends on the user account system for item tracking, the wishlist database for saving and retrieving items, and the confirmation service for notifying the user.
Interfaces	The module interacts with users via the "Add to Wishlist" button and the wishlist section, with the database for storing and retrieving wishlist items, and error handling for system failures.
Resources	It requires server resources for processing wishlist actions, a database for storing wishlist items, and the user interface for displaying the wishlist.
Processing	The system processes the user's request to add an item to the wishlist, updates the wishlist in the database, and displays the user's wishlist when accessed.
Data	The system processes the item details, storing them in the user's wishlist for future access and removing items as requested by the user.

Table 10: Add to Wishlist Component Detail

3.2.11. View Wishlist

Identification	View Wishlist
Type	Module

Purpose	It allows users to access and browse the items they have previously added to their wishlist, enabling easy review and interaction with saved items.
Function	The system retrieves and displays the saved wishlist items, provides details of each item, and allows users to view full item details or initiate trades. If the wishlist is empty, it displays a message informing the user.
Subordinates	The module includes the wishlist display, the retrieval of item details from the database, and the interaction with the items (viewing full details or initiating trade).
Dependencies	It depends on the user account system, the wishlist database for storing saved items, and the error handling service for managing retrieval issues.
Interfaces	The module interacts with users through the wishlist section in the profile, with the database for retrieving item data, and displays error messages when necessary.
Resources	It requires server resources for fetching and displaying wishlist items, a database for storing and retrieving saved items, and the user interface for browsing and managing the wishlist.
Processing	The system processes the user's request to view the wishlist, retrieves the saved items, displays the items with relevant details, and handles cases such as an empty wishlist.
Data	The system processes the user's wishlist data, including item names, images, and descriptions, and presents them for user interaction.

Table 11: View Wishlist Component Detail

3.2.12. Item Detail

Identification	Item Detail
Type	Module
Purpose	It allows users to view detailed information about a specific item, including its name, description, condition, category, images, and owner details, to help them make informed decisions about the item.
Function	The system fetches and displays comprehensive details about an item, including available actions such as initiating a trade or adding to a wishlist. If the item is unavailable, a message is displayed to notify the user.
Subordinates	The module includes item detail display, actions such as "Propose Trade" or "Add to Wishlist," and error handling for unavailable items or system failures.
Dependencies	It depends on the item database for retrieving item details, the user interface for presenting the details, and the action services for managing trade proposals and wishlist additions.
Interfaces	The module interacts with users through the item detail page, with the database for retrieving item data.
Resources	It requires server resources for fetching and displaying item details, a database for storing item information, and the interface for presenting data and interacting with users.
Processing	The system processes the request to display item details, retrieves data from the database, displays the item with its details, and handles actions such as trade proposals or adding items to the wishlist.

Data	The system processes the item's name, description, condition, category, images, and owner information to present them on the item detail page and allow user interaction.

Table 12: Item Detail Component Detail

3.2.13. Trade Proposal

Identification	Trade Proposal
Type	Module
Purpose	This feature enables users to propose exchanges of items, facilitating seamless and transparent trading by specifying desired items and offering their own for trade.
Function	The user selects an item they wish to trade for, proposes a trade by offering their own item, and submits the proposal. The system validates the inputs, saves the proposal, and notifies the other user.
Subordinates	The feature includes the proposal form and trade validation process.
Dependencies	The feature depends on the user's login status and having an item listed for trade.
Interfaces	The system interacts with the user via the trade proposal form and notifies the recipient of the proposal.
Resources	The system uses the database to store trade proposals.
Processing	The system processes trade proposals, validates inputs, saves data, and sends notifications to the recipient.
Data	The system handles trade proposal data, including item selection and notifications.

Table 13: Trade Proposal Component Detail

3.2.14. Accept Trade

Identification	Accept Trade
Type	Module
Purpose	It allows users to confirm and finalize a trade proposal by accepting the offered terms, ensuring successful transactions and item exchanges between users on the platform.
Function	The system displays the trade proposal details, and upon user confirmation, it updates the trade status in the database and notifies the other user of the acceptance, ensuring the exchange is finalized.
Subordinates	The module includes displaying trade proposal details, the accept/reject actions, updating trade status, and notifying the other user of the trade decision.
Dependencies	It depends on the trade database for tracking proposals, the user interface for displaying trade details, and notification services for informing users about the trade status.
Interfaces	The module interacts with users through the trade proposal page, with the database for managing trade status, and with notification systems for alerting the proposing user upon acceptance.

Resources	It requires server resources to process trade proposals, a database to store trade statuses, and notification systems to communicate the result of the trade acceptance.
Processing	The system retrieves and displays the trade details, updates the trade status upon user acceptance or rejection, and ensures both users are notified of the decision.
Data	The system processes trade proposal information, including item details, user identities, and trade status updates, to facilitate the exchange and notify both users about the acceptance or rejection.

Table 14: Accept Trade Component Detail

3.2.15. Reject Trade

Identification	Reject Trade
Type	Module
Purpose	It allows users to decline unwanted trade proposals, ensuring they maintain control over their trade interactions and only engage in desired exchanges.
Function	The system displays the trade proposal details, and upon rejection by the user, it updates the trade status in the database and notifies the proposing user of the rejection.
Subordinates	The module includes displaying trade proposal details, the reject action, updating trade status, and notifying the proposing user about the rejection.
Dependencies	It depends on the trade database for tracking proposals, the user interface for displaying trade details, and notification systems for informing users about the trade status.
Interfaces	The module interacts with users through the trade proposal page, with the database for managing trade statuses, and with notification systems to alert the proposing user about the rejection.
Resources	It requires server resources to process trade proposals, a database to store trade statuses, and notification systems to communicate the rejection outcome to the proposing user.
Processing	The system retrieves and displays the trade proposal details, updates the trade status upon user rejection, and ensures the proposing user is notified about the decision.
Data	The system processes trade proposal information, including item details, user identities, and trade status updates, to facilitate the rejection and notify the proposing user.

Table 15: Reject Trade Component Detail

3.2.16. Cancel Trade

Identification	Cancel Trade
Type	Module
Purpose	It allows users to cancel an ongoing trade proposal they initiated, ensuring they maintain control over active trades and can withdraw proposals before acceptance.

Function	The system displays the active trade, prompts the user for confirmation to cancel, updates the trade status in the database to "Cancelled," and notifies the other user of the cancellation.
Subordinates	The module includes retrieving the active trade details, the cancel action, updating the trade status, and notifying the other party about the cancellation.
Dependencies	It depends on the trade database to track active trade proposals, the user interface to present the cancel option, and the notification system to inform the other party about the cancellation.
Interfaces	The module interacts with the user through the trade proposal section, the database to update trade status, and the notification system to inform the other party about the cancellation.
Resources	It requires server resources to process the cancellation, a database to update the trade status, and notification systems to alert the other user of the cancellation.
Processing	The system retrieves the active trade, prompts for confirmation to cancel, updates the trade status, and ensures the other user is notified of the cancellation.
Data	The system processes trade proposal details, user identities, and cancellation status to update the trade status and notify the other party.

Table 16: Cancel Trade Component Detail

3.2.17. Advanced Trade Management

Identification	Advanced Trade Management
Type	Module
Purpose	The purpose of the Advanced Trade Management feature, specifically the Counter Offer, is to allow users to negotiate trade terms by selecting an item from the sender’s list while keeping their previously proposed item selected. It fosters negotiation and enhances the trade experience by allowing users to propose alternate terms.
Function	This feature enables users to create counter trade offers by selecting items from the sender's list, while keeping their own previously proposed item. It supports seamless trade negotiation and notifies the sender when a counter offer is sent.
Subordinates	The feature includes opening a trade proposal page, validating item selection, saving the counter proposal, sending notifications, and error handling for failed submissions.
Dependencies	This feature depends on active trade data (items and trade offers), a system to track user selections, and a notification system to inform the sender of the counter proposal.
Interfaces	The system interacts with the user through the trade interface, communicates with the database to save the counter offer, and uses the notification system to inform the sender.
Resources	The feature requires server resources to handle counter offer submissions, database resources for tracking trade changes, and a notification system for informing users of the counter offer.
Processing	The system fetches the trade proposal, allows the user to select an item from the sender’s list, validates the selection, saves the counter offer, and notifies the sender. If the user cancels, the system discards the changes.

Data	The system processes the trade details, user selections, and trade status to ensure that the counter offer is properly handled and that the sender is notified. It also manages error handling and displays confirmation or error messages as necessary.

Table 17: Advanced Trade Management Component Detail

3.2.18. Manage Active Trades

Identification	Manage Active Trades
Type	Module
Purpose	The Manage Active Trades feature enables users to view, accept, or reject ongoing trade proposals. This functionality ensures that users can actively engage in trading and take control of the status of their trades, making the platform more interactive and user-driven.
Function	Users can view all active trade proposals, manage their status by either accepting or rejecting the trade, and receive notifications upon updating trade statuses. This ensures seamless interaction between users and improves trading efficiency.
Subordinates	The feature includes the display of active trade proposals, the ability to accept or reject trades, notifications to both parties involved in the trade, and error handling for system failures or issues when processing requests.
Dependencies	This feature relies on an active trade database to fetch trade proposals, a user notification system, and a backend system that tracks and updates the trade status based on user actions.
Interfaces	The system interacts with the trade interface for displaying active trade proposals, the database for updating trade status, and the notification system for informing users of trade updates.
Resources	The feature requires server resources for fetching active trades, updating trade statuses, and sending notifications. Additionally, it requires database resources for storing trade information and status.
Processing	The system processes the user's action of accepting or rejecting a trade, updates the trade status in the database, and notifies the other user involved in the trade. If there are no active trades, the system displays an appropriate message.
Data	The system processes data regarding active trade proposals, their details, trade status, and user actions. The system also manages the notification data to inform both users about the trade status updates.

Table 18: Manage Active Trades Component Detail

3.2.19. Ban User Account

Identification	Ban User Account
Type	Module
Purpose	The Ban User Accounts feature enables administrators to manage user behavior by banning accounts that violate platform policies. This ensures a safe and secure

	environment by allowing administrators to take action against disruptive or rule-breaking users.
Function	Administrators can view all user accounts, select specific users, and ban them from the platform if necessary. Banned users will no longer be able to access or interact with the platform.
Subordinates	The feature includes user account listing, selection for banning, system updates to account status, and error handling for issues during the banning process. Additionally, it includes a notification system to inform the admin of successful actions or errors.
Dependencies	This feature depends on an admin interface for managing user accounts, a backend database to store and update user statuses, and proper permission settings to ensure that only authorized admins can ban accounts.
Interfaces	The system interfaces with the admin dashboard for account management, the database for retrieving and updating user account statuses, and the notification system for informing administrators of successful or failed actions.
Resources	The feature requires database resources to update user account statuses, server resources to process admin requests, and access to the admin portal for administrators to manage accounts.
Processing	The system processes the admin's request to ban a user by updating the account status in the database. If the action is successful, a confirmation message is shown; if not, an error message is displayed.
Data	The system handles user account data, including user profiles, account statuses, and permissions. It also manages data related to the admin's action, such as the user selected for banning and confirmation messages.

Table 19: Ban User Account Component Detail

3.2.20. Delete List Item

Identification	Delete List Item
Type	Module
Purpose	The Delete List Items feature enables administrators to remove items listed by users on the platform. This ensures the integrity of the marketplace by enforcing platform policies and guidelines, preventing the sale or display of prohibited or inappropriate items.
Function	Administrators can view a list of all items listed by users, select items to review or remove, and delete those items from the marketplace when necessary.
Subordinates	The feature includes the listing of user-submitted items, item selection for removal, deletion of items from the database, and error handling in case of issues. It also includes confirmation or error messages for administrators.
Dependencies	This feature depends on the admin portal for managing listings, a backend database for storing and removing item listings, and proper permissions to ensure only authorized admins can delete items.
Interfaces	The system interfaces with the admin dashboard for viewing listed items, the database for retrieving and deleting items, and a notification system to confirm actions or alert administrators of any errors.

Resources	The feature requires database resources to store, retrieve, and delete item listings, server resources for processing the deletion request, and access to the admin portal for authorized users to manage items.
Processing	When an admin deletes an item, the system removes the item from the listings and the database. A confirmation message is displayed to the admin to verify the successful removal. If an error occurs during the deletion process, an error message is displayed.
Data	The system handles data related to the items listed by users, including item descriptions, categories, and statuses. It also manages data related to the admin's actions, such as which items are selected for removal and logs of those actions.

Table 20: Delete List Item Component Detail

3.2.21. Rate & Review System

Identification	Rate & Review System
Type	Module
Purpose	The Rate and Review System enables users to provide feedback on their trading partners after completing a trade. This feature helps enhance trust within the platform's community by allowing users to share their experiences, both positive and negative, and assist future traders in making informed decisions.
Function	The system allows users to submit a rating (typically in the form of stars or numerical value) and a written review after completing a trade. These ratings and reviews are saved in the database and can be viewed on the rated user's profile.
Subordinates	The feature includes the display of the rating and review form, input validation for ratings and reviews, storing feedback in the database, and updating the user profile to reflect the new ratings/reviews.
Dependencies	The feature depends on the user's completion of a trade, access to the "Rate Trade" section, the backend database for storing ratings and reviews, and user profiles that will display the ratings/reviews.
Interfaces	The system interfaces with the user interface for submitting ratings and reviews, the backend database for saving and retrieving the reviews, and the user profile pages to display feedback for future reference.
Resources	The feature requires access to the database for storing ratings and reviews, as well as system resources to validate and save the feedback. It also requires an interface for users to submit their ratings.
Processing	When the user submits their rating and review, the system validates the input, ensures the rating is within a valid range (e.g., 1-5 stars), and saves the review to the database. A confirmation message is displayed to the user once the process is complete. If an error occurs, the system displays an error message.
Data	The system manages data related to the ratings and reviews submitted by users. This includes the trade partner being rated, the rating value, the review text, and the user submitting the feedback. It also manages any error messages if issues occur during the submission process.

Table 21: Rate & Review System Component Detail

3.2.22. Email Notification

Identification	Email Notification
Type	Module
Purpose	The feature ensures users are notified of significant events like trade proposals, acceptances, rejections, or cancellations to help them make informed decisions.
Function	The system sends automated email notifications to users based on relevant trade events and updates to keep them informed.
Subordinates	The feature depends on event triggers, email content generation, and email delivery to the correct recipients.
Dependencies	It relies on verified user email addresses, system event triggers, and an email service to send notifications.
Interfaces	It interacts with the database for user email details and with the email service to send notifications.
Resources	The feature requires access to a database for user email addresses and an email service for delivery.
Processing	When an event occurs, the system generates and sends an email notification to the user's registered email address.
Data	The system processes user email addresses and event details to ensure notifications are sent accurately and timely.

Table 22: Email Notification Component Detail

3.2.23. Trade History

Identification	Trade History
Type	Module
Purpose	It helps users review and analyze completed trades for record-keeping. This feature enhances transparency.
Function	The feature retrieves and shows a list of completed trades, providing details about each. Users can click to view trade-specific information.
Subordinates	The system relies on the user's trade data and account information to display history. It integrates with the database for trade records.
Dependencies	It requires a user account with completed trades and access to stored trade data. The system must be able to retrieve these records.
Interfaces	The feature interfaces with the user account system and trade database to present historical trade data.
Resources	The system uses the user's account and trade data to generate the history. It also requires database access to store trade records.

Processing	Upon request, the system fetches the user's trade data from the database and displays the details of completed trades.
Data	The system processes trade-related data, including item details, trade dates, and partner information, to generate accurate trade histories.

Table 23: Trade History Component Detail

3.2.24. Send Message

Identification	Send Message
Type	Module
Purpose	The purpose is to allow users to send messages to others, fostering communication and collaboration within the platform.
Function	The user selects a trade or another user to message, types a message, and clicks "Send." The system delivers the message in real-time to the recipient. If there is an error, the system displays an error message.
Subordinates	The module includes the messaging interface for composing and sending messages.
Dependencies	It depends on the user's account and active trades to initiate messages.
Interfaces	The system interacts with the user through the messaging interface, allowing users to compose and send messages.
Resources	The module requires server resources for real-time message delivery and storage.
Processing	The system processes the user's message and delivers it to the intended recipient.
Data	The system handles message data, including the content of the message and delivery status.

Table 24: Send Message Component Detail

3.2.25. Remove from Wishlist

Identification	Remove from Wishlist
Type	Module
Purpose	It allows users to declutter their wishlist, keeping only relevant items. This ensures users can track their desired items effectively.
Function	Users can navigate to their wishlist and select an item to remove. The system updates the wishlist after removing the selected item.
Subordinates	The system works with the user's wishlist data and profile. It integrates with the user interface to display and modify wishlist items.
Dependencies	It depends on the user's account and the wishlist data being available. The system must be able to modify the wishlist entries in the database.
Interfaces	The feature interfaces with the user's profile and wishlist storage. It allows the removal of items from the user's saved list.

Resources	The system accesses the user's profile and wishlist data. It also requires database access to update the wishlist content.
Processing	Upon removal, the system updates the wishlist by removing the selected item and ensures the change is reflected across the platform.
Data	The system processes data related to the wishlist, including item names, descriptions, and user preferences to update the list accordingly.

Table 25: Remove from Wishlist Component Detail

4. User Interface Design

4.1. Section Overview

This section outlines the design and structure of the user interface for the system. It covers the principles, standards, and conventions used to create an intuitive and user friendly experience. The section includes the key UI components, layout, navigation, and any external libraries or frameworks used for building the interface. Additionally, the design of various screens and user interactions will be discussed to ensure the system is visually appealing and easy to use.

4.2. Interface Design Rules

- **Consistency**: The design maintains a consistent layout, color scheme, and behavior across different pages and screens to provide a unified experience for the user.
- **Usability**: The interface is designed to be intuitive, with clear and easily recognizable controls, buttons, and labels that guide users through their tasks.
- **Responsiveness**: The design adapts seamlessly to different screen sizes and resolutions, ensuring a smooth experience across various devices, from desktops to mobile phones.
- **User Experience**: The interface is optimized for ease of navigation, minimizing user inputs while maximizing functionality and efficiency. This ensures a user-friendly environment, enhancing the overall experience.

4.3. GUI Components

The system utilizes several key libraries and dependencies to build the user interface, ensuring a dynamic and responsive experience. These components include:

- **React**: A JavaScript library for building user interfaces, enabling the creation of dynamic, responsive, and interactive web pages through reusable components.
- **Axios**: A promise-based HTTP client used for making API requests, allowing the application to fetch and send data to the back-end server.
- **React-Router-Dom**: A routing library used for handling client-side navigation, enabling smooth transitions between pages or views without full page reloads.
- **React-Icons**: A library providing a collection of customizable icons that enhance the visual appeal and usability of the user interface.

These libraries and components work together to deliver a robust, efficient, and user-friendly interface for the application.

4.4. Detailed Description

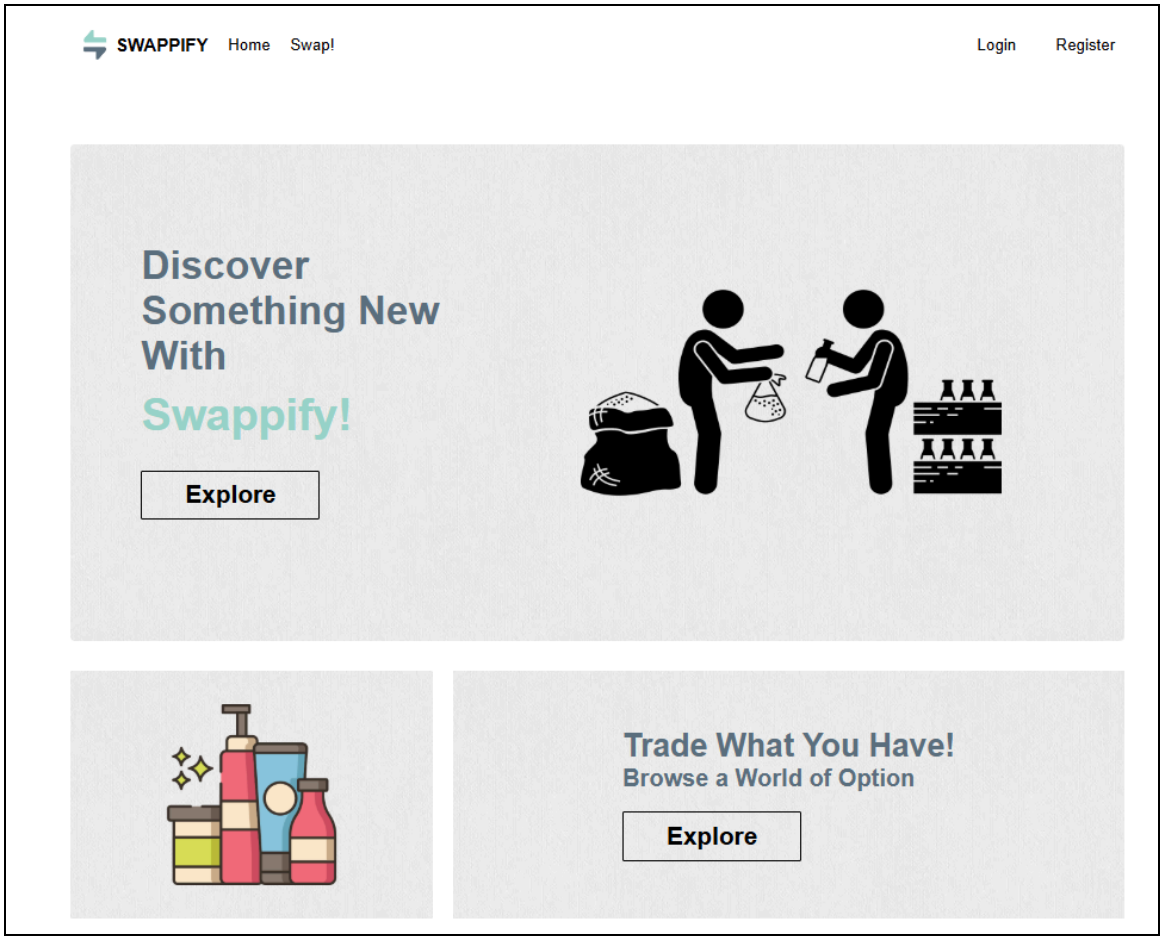

Figure 1. Home Page

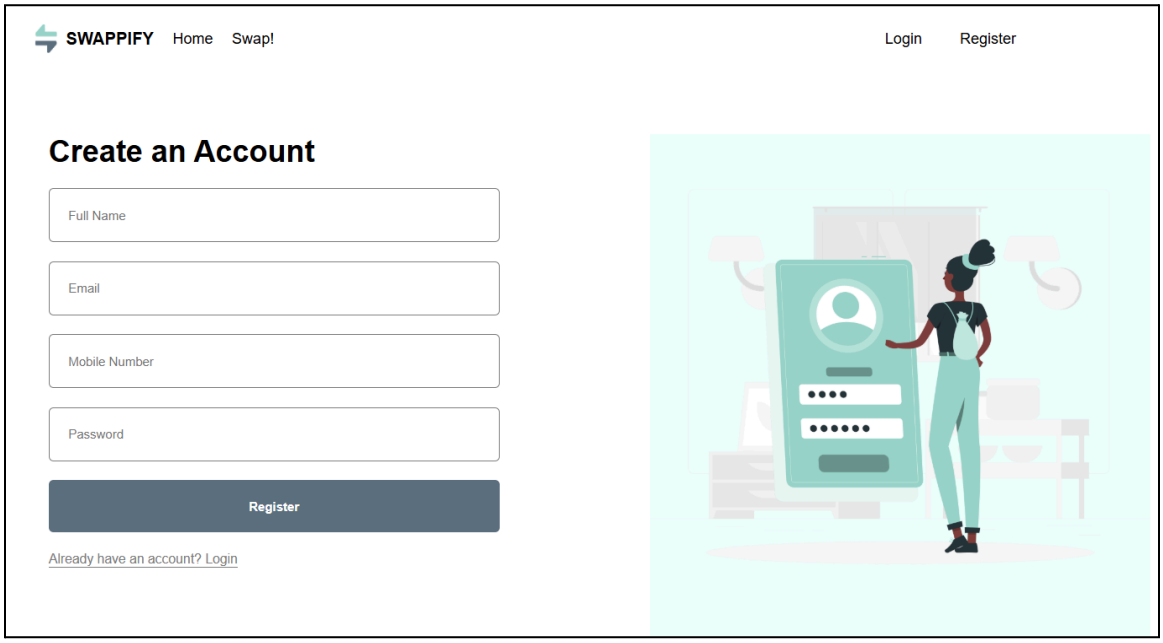

Figure 2. Sign Up Page

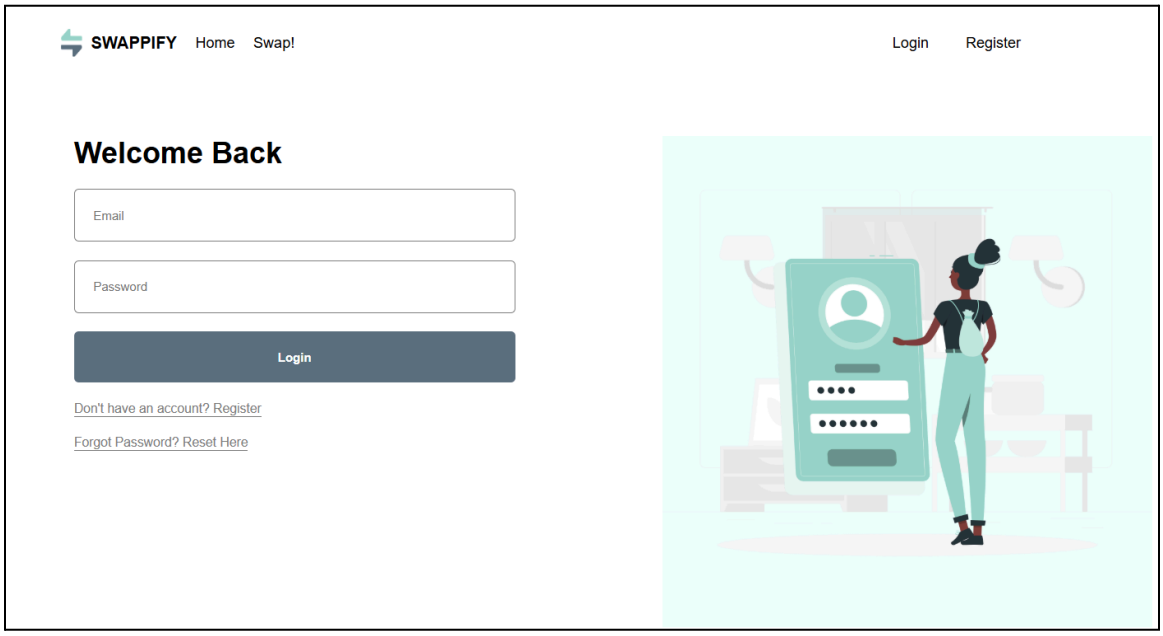

Figure 3. Login Page

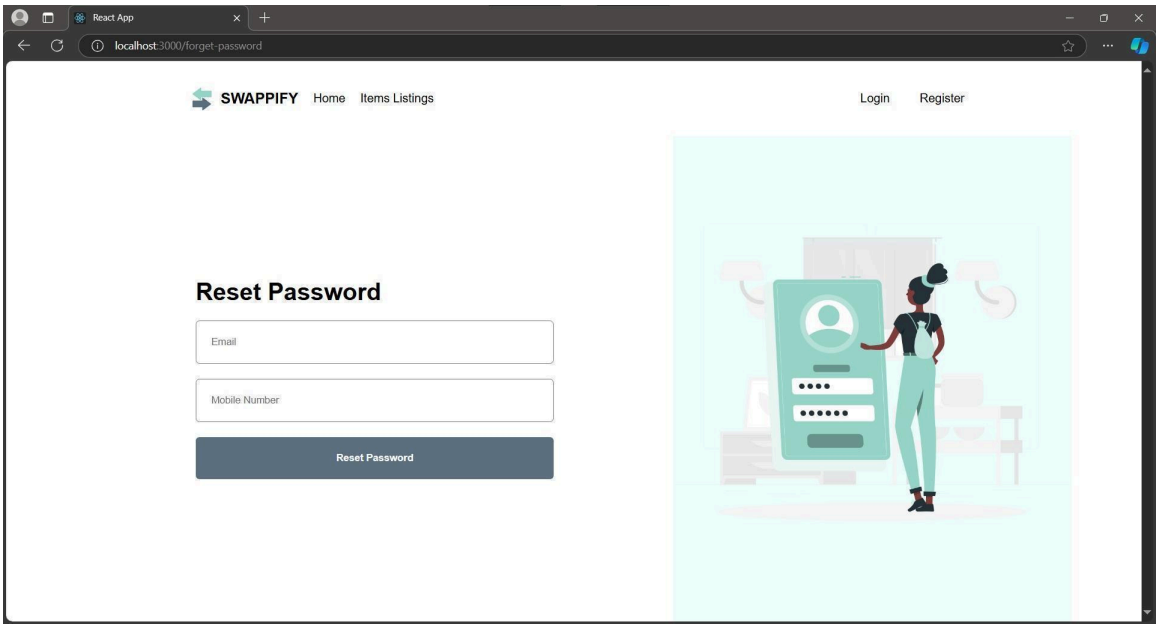

Figure 4. Forget Password Page

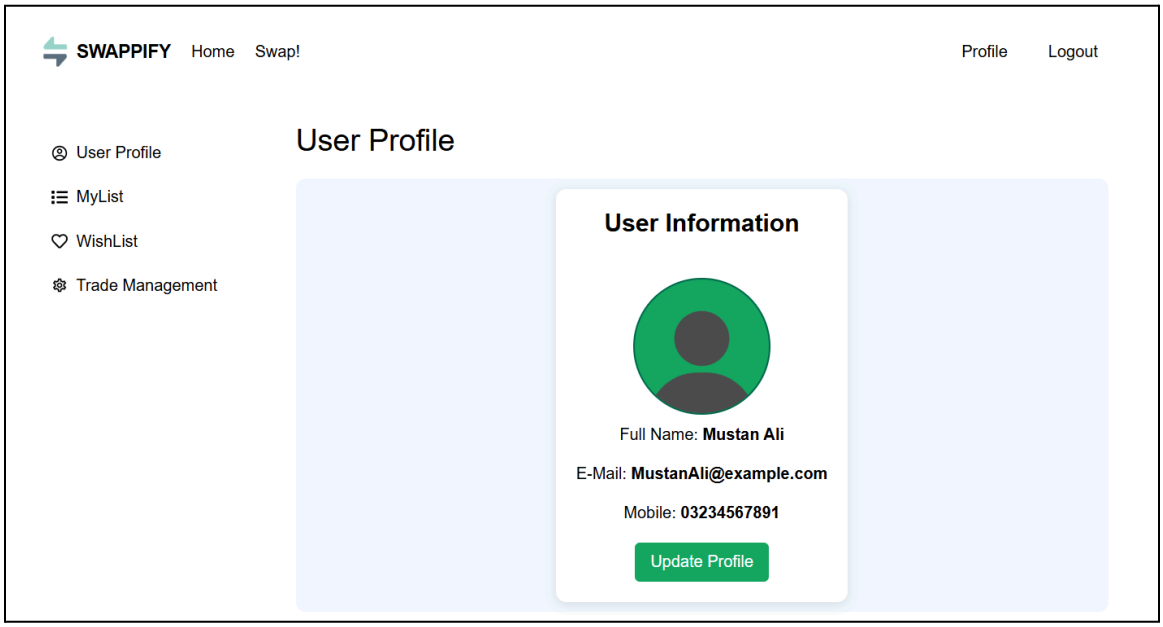

Figure 5. Profile Page

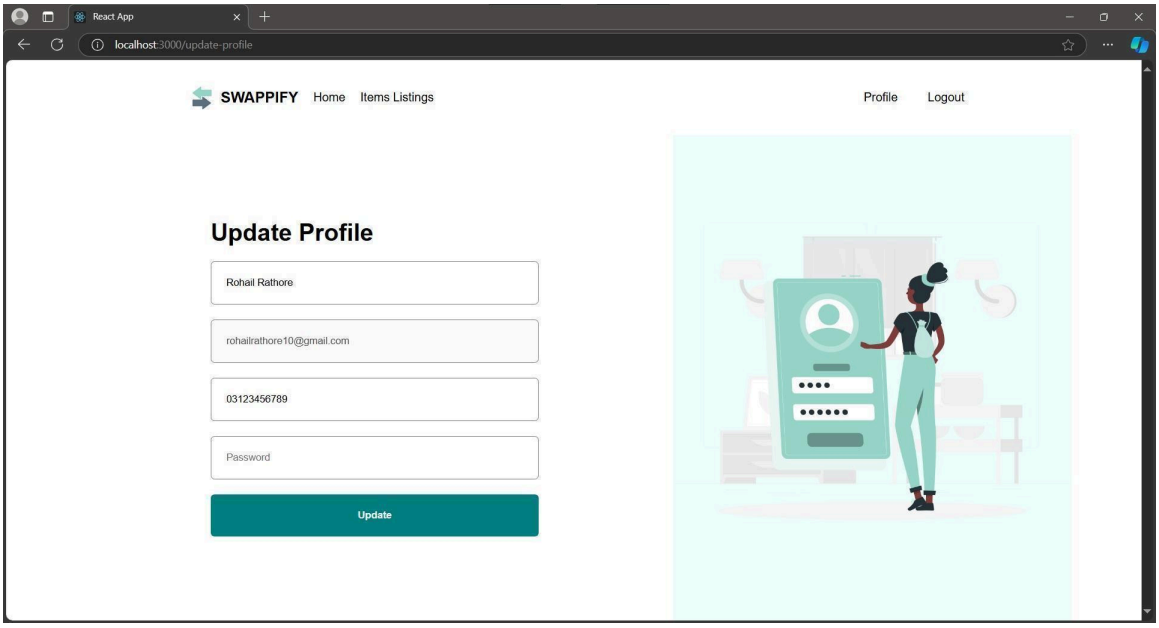

Figure 6. Update Profile Page

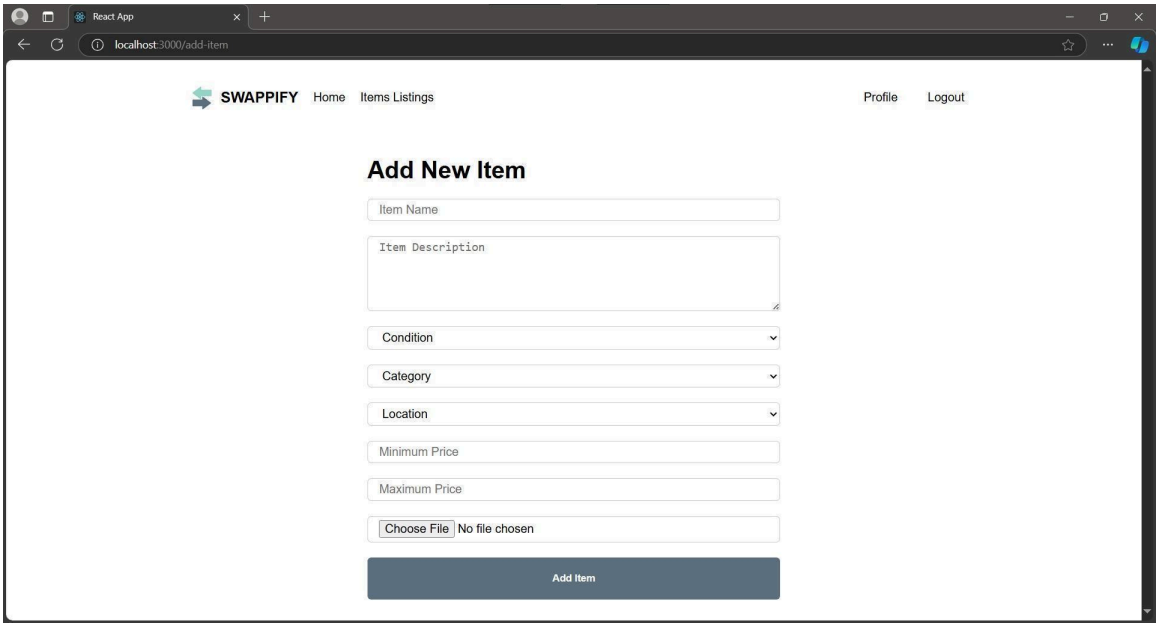

Figure 7. Add Item Page

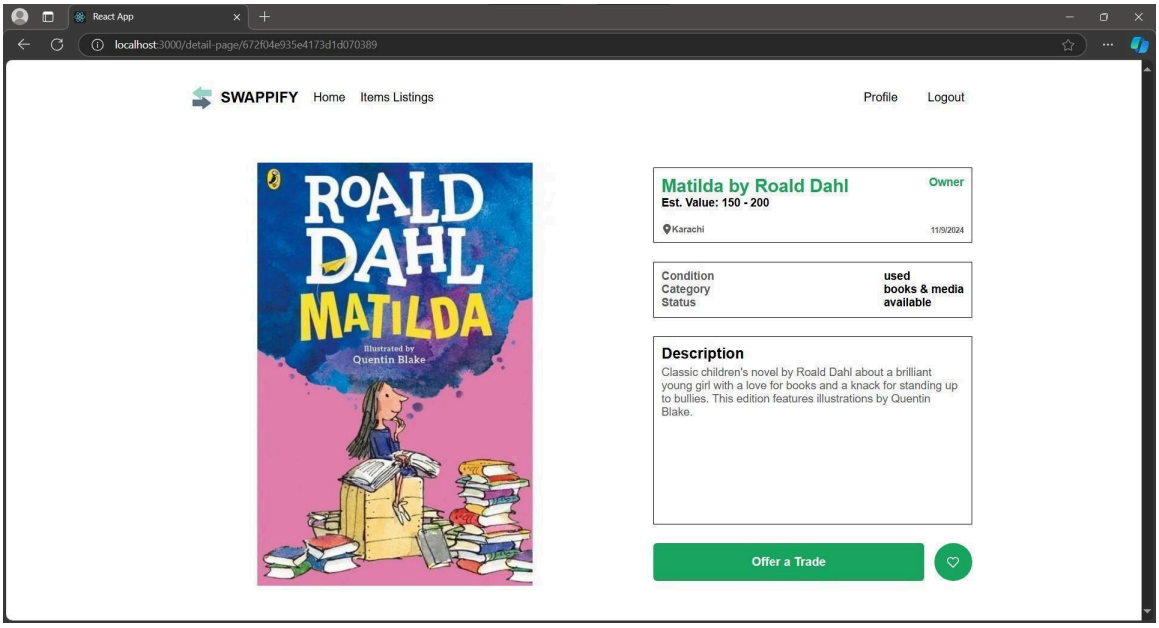

Figure 8. Item Detail Page

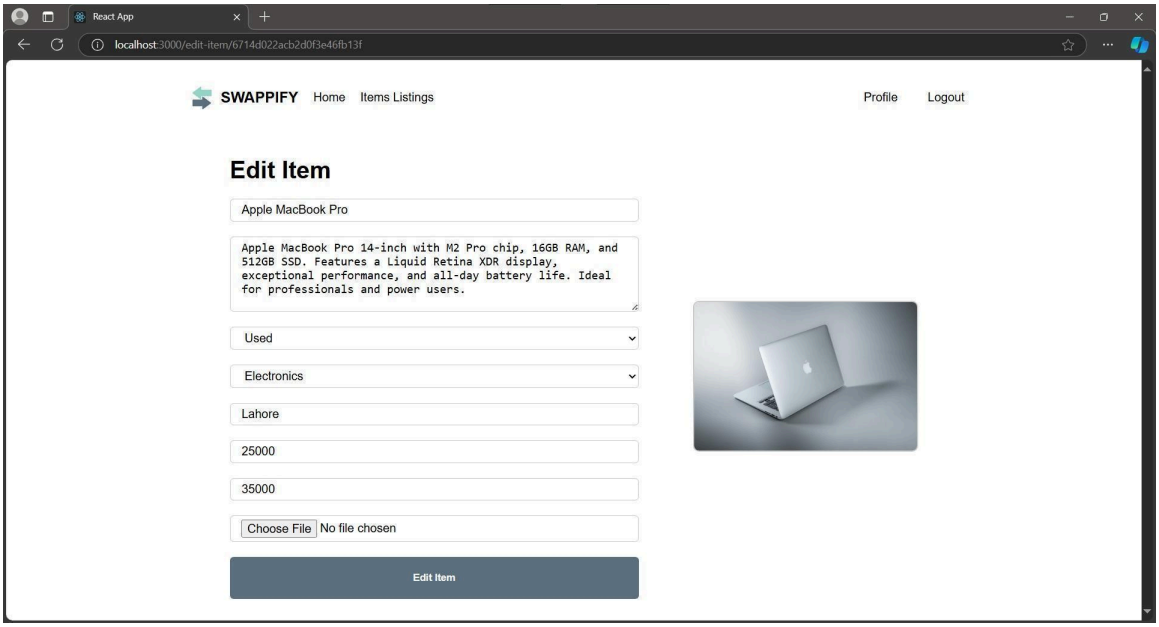

Figure 9. Edit Item Page

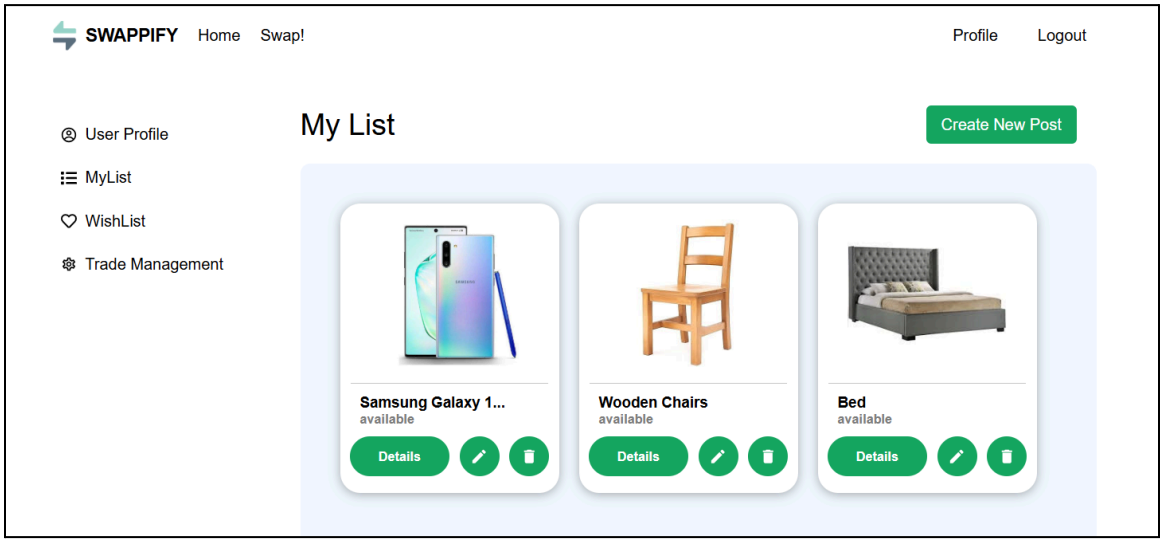

Figure 10. User Item Page

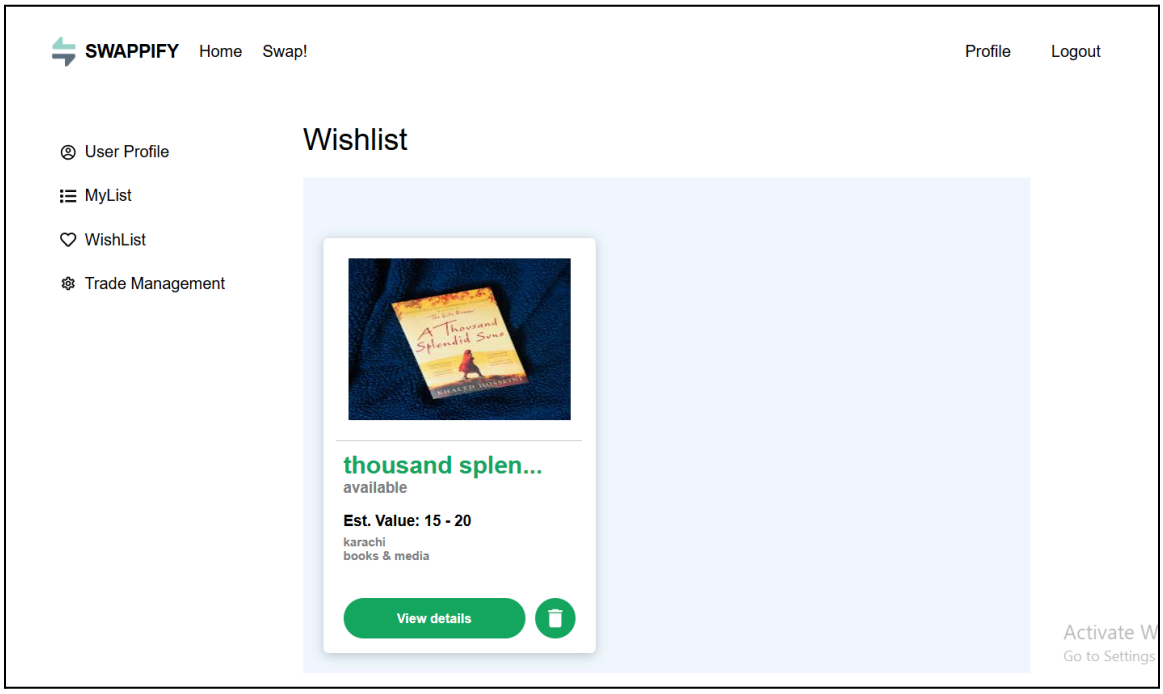

Figure 11. Wishlist Page

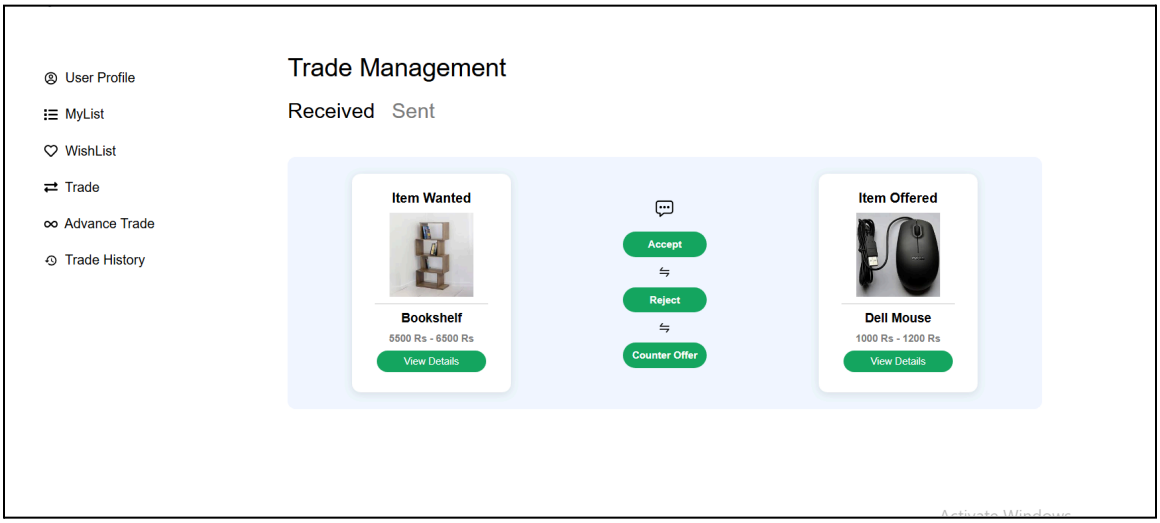

Figure 12. Trade Management: Received Request Page

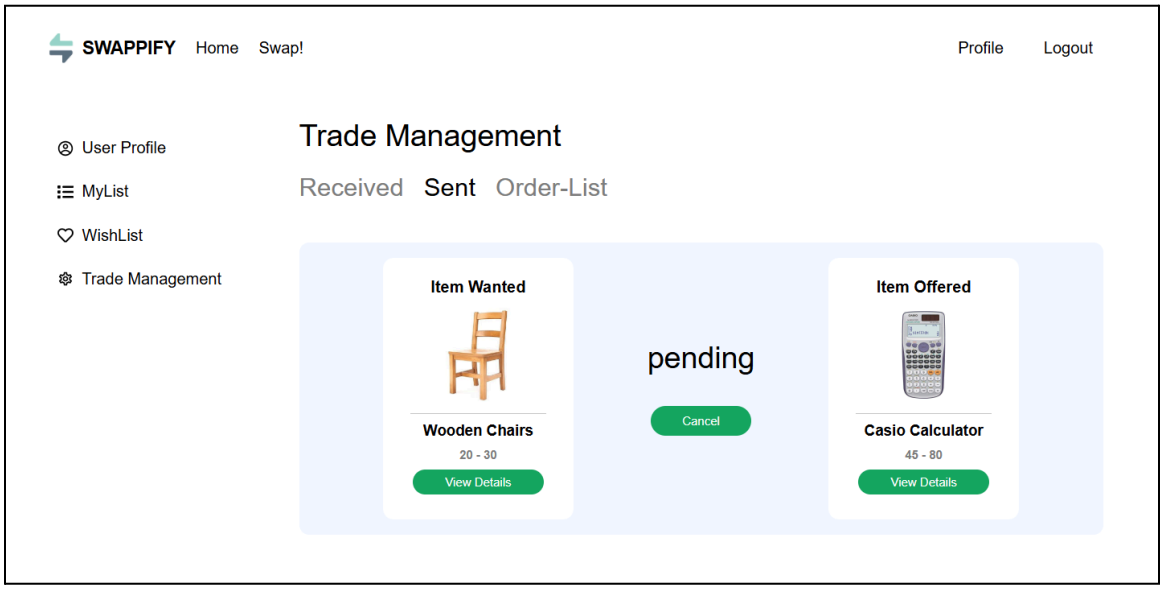

Figure 13. Trade Management: Sent Request Page

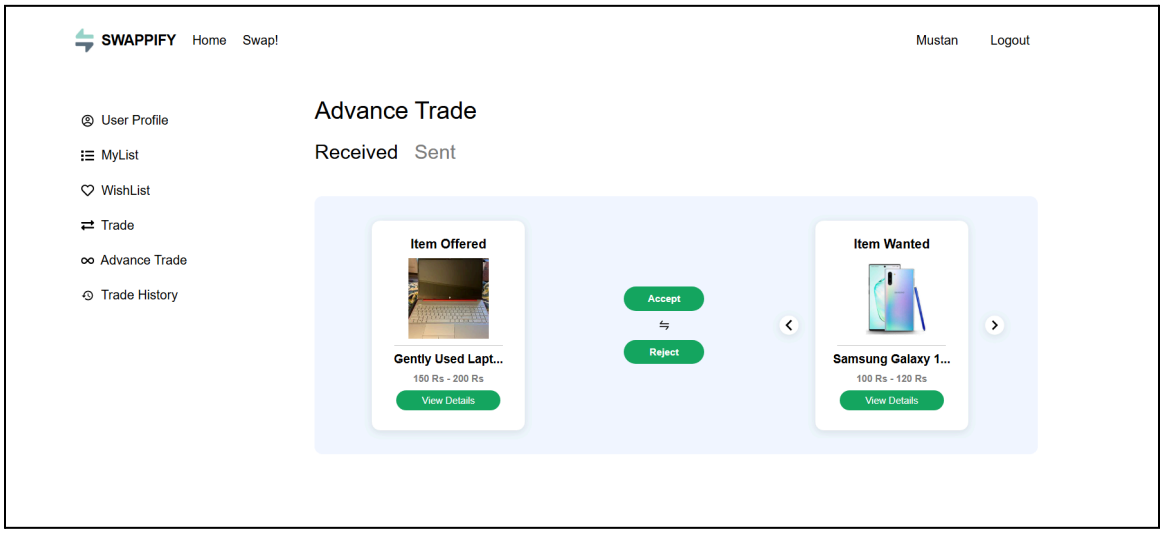

Figure 14. Advance Trade Management: Received Request Page

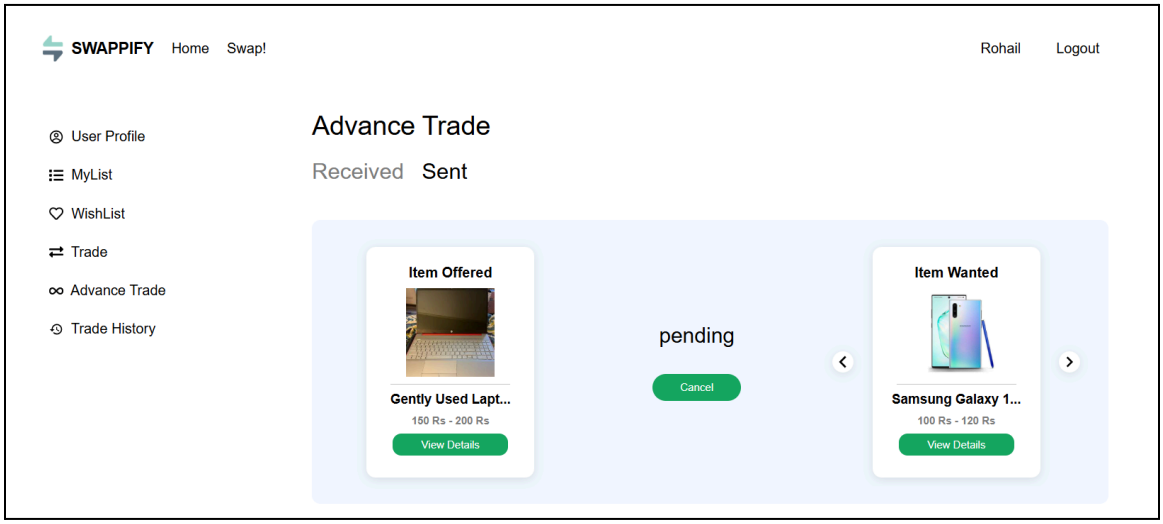

Figure 15. Advance Trade Management: Sent Request Page

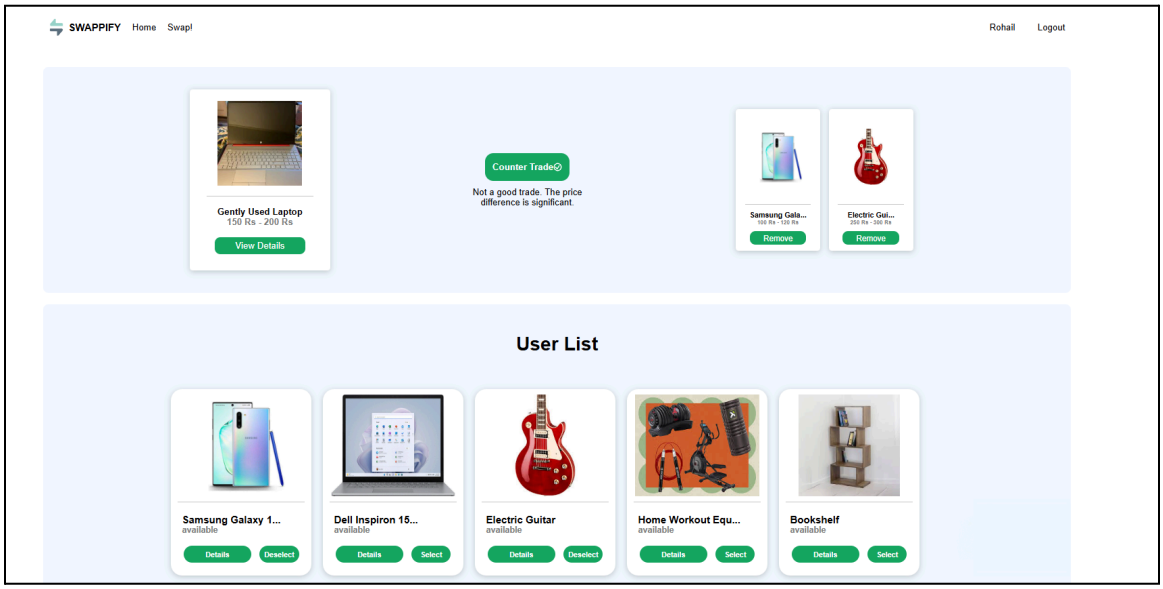

Figure 16. Advance Trade: Sent Offer

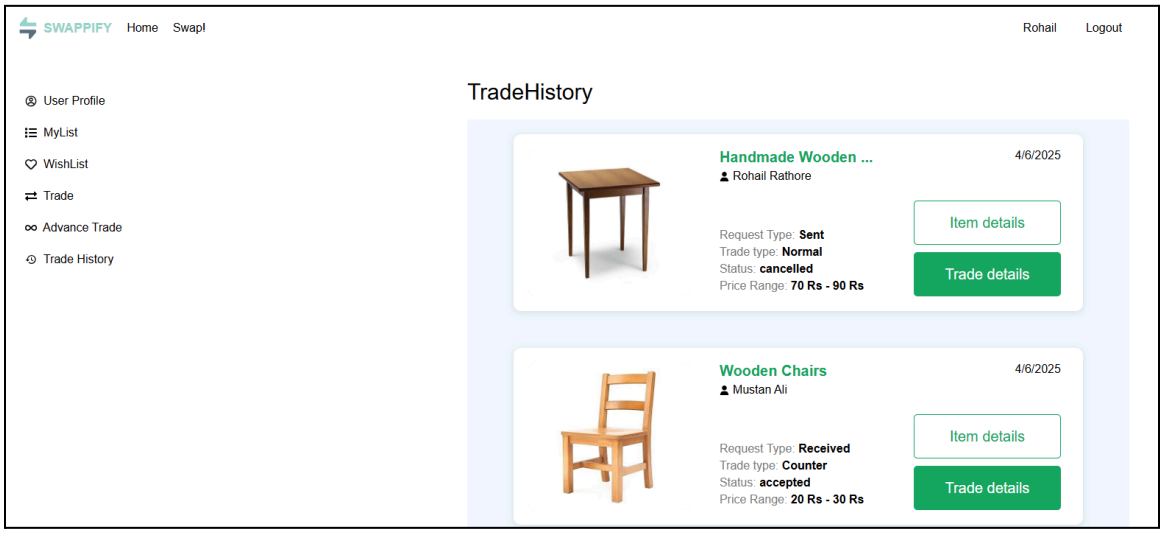

Figure 17. Trade History

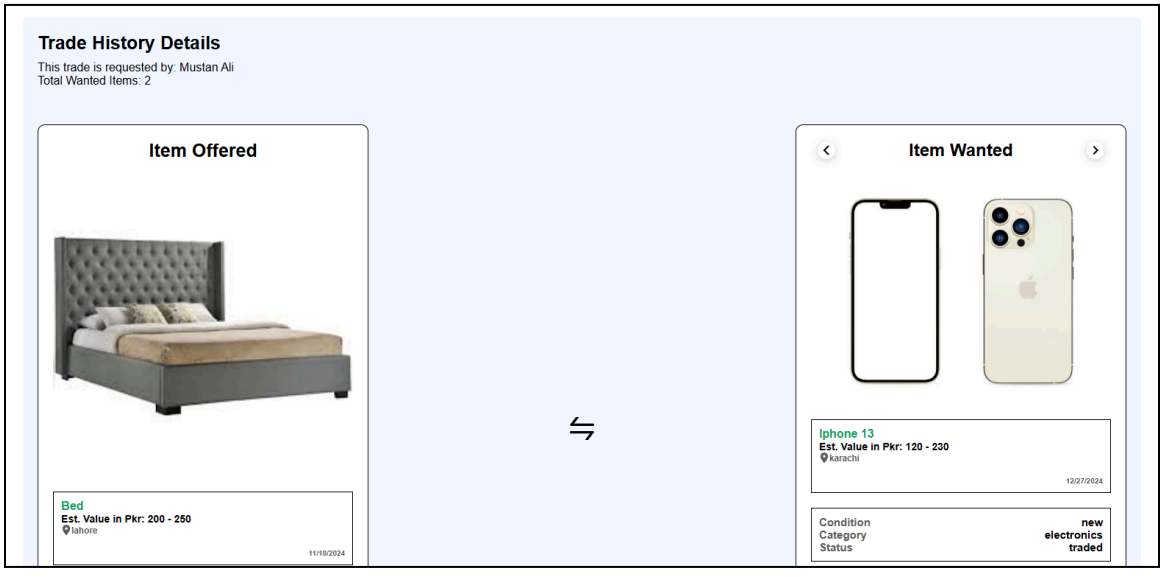

Figure 18. Trade Detail

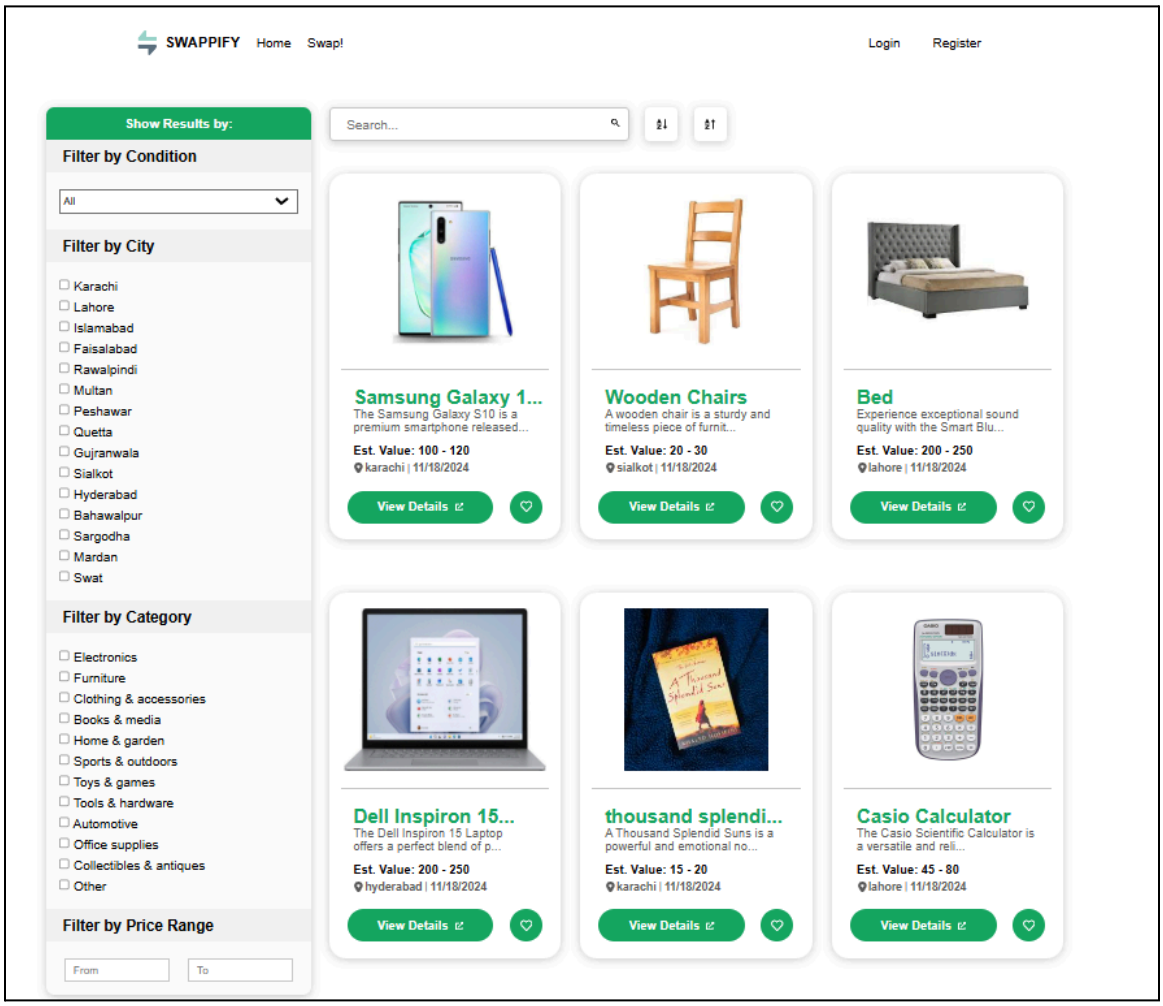

Figure 19. Listed Items Page

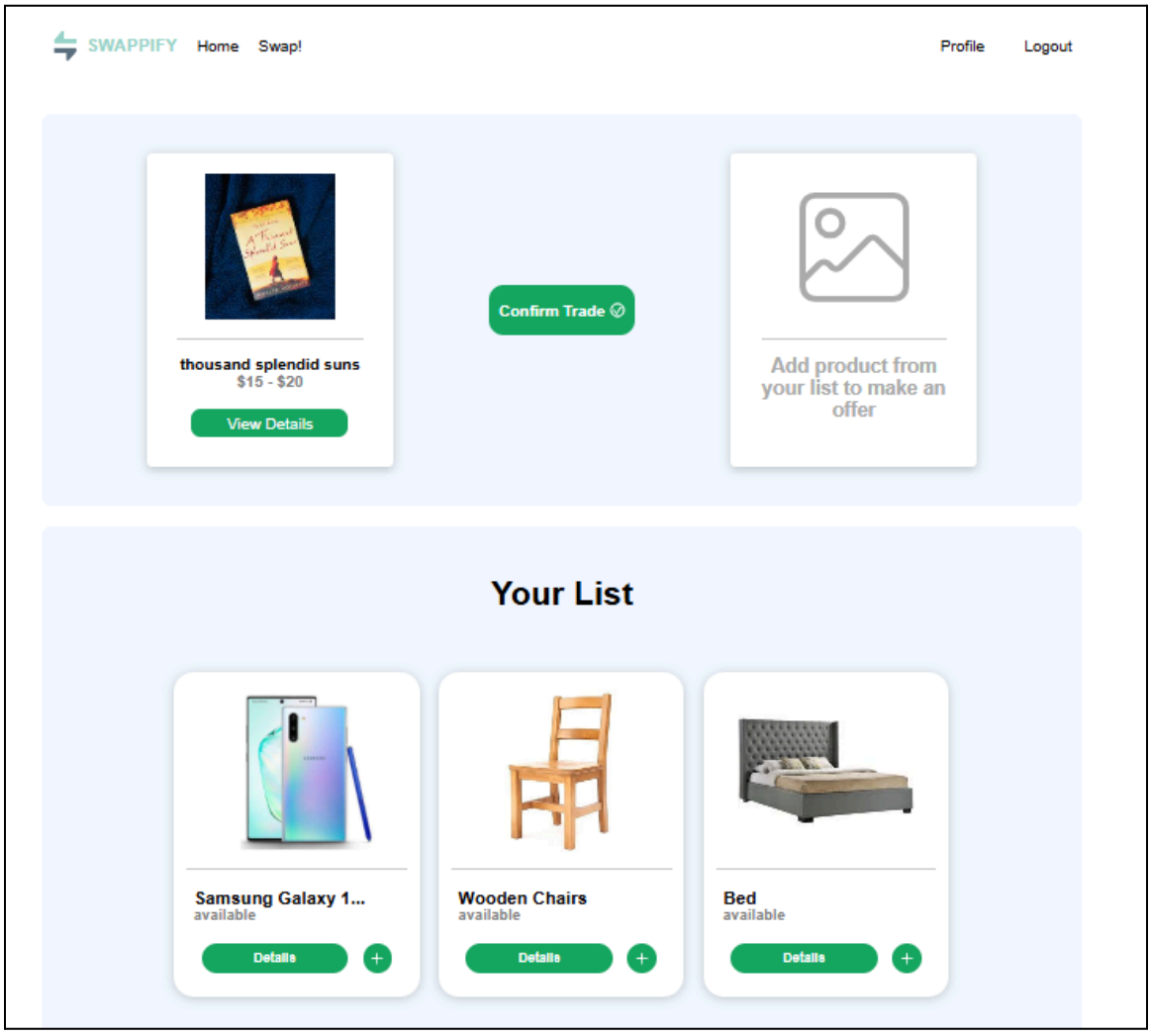

Figure 20. Trade Proposal Page

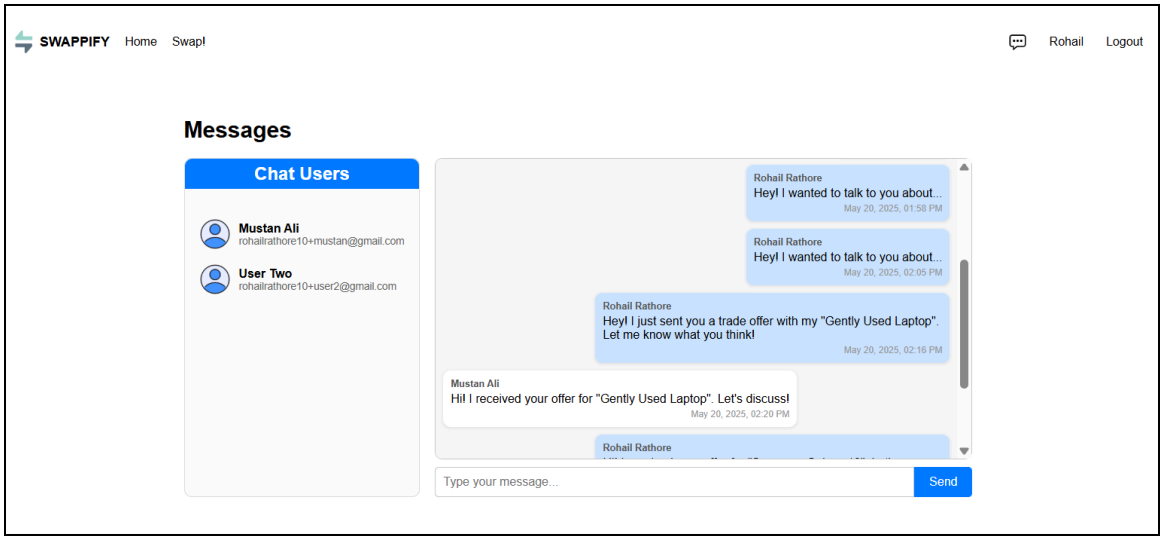

Figure 21. Message System

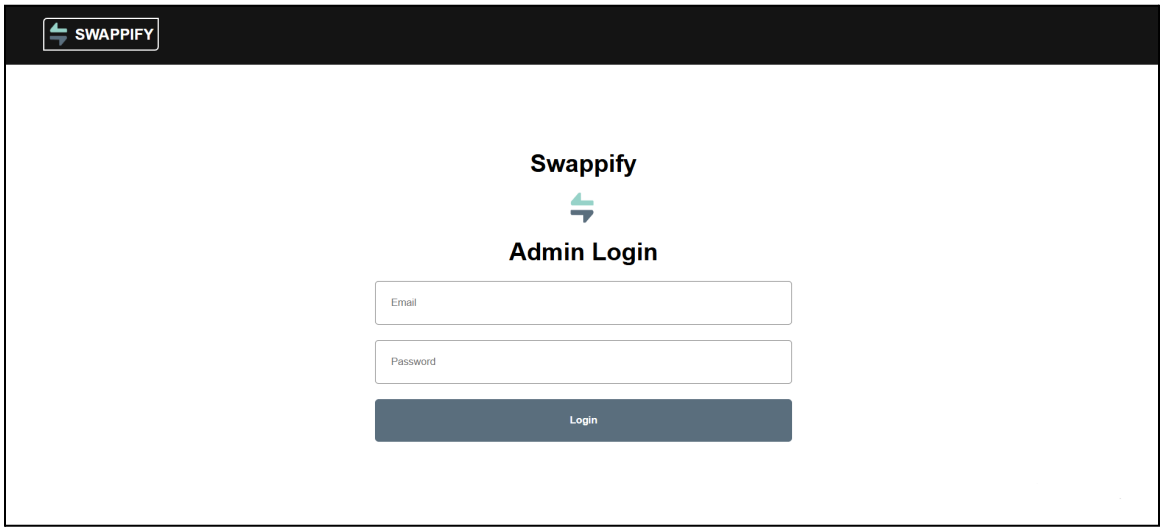

Figure 22. Admin Login

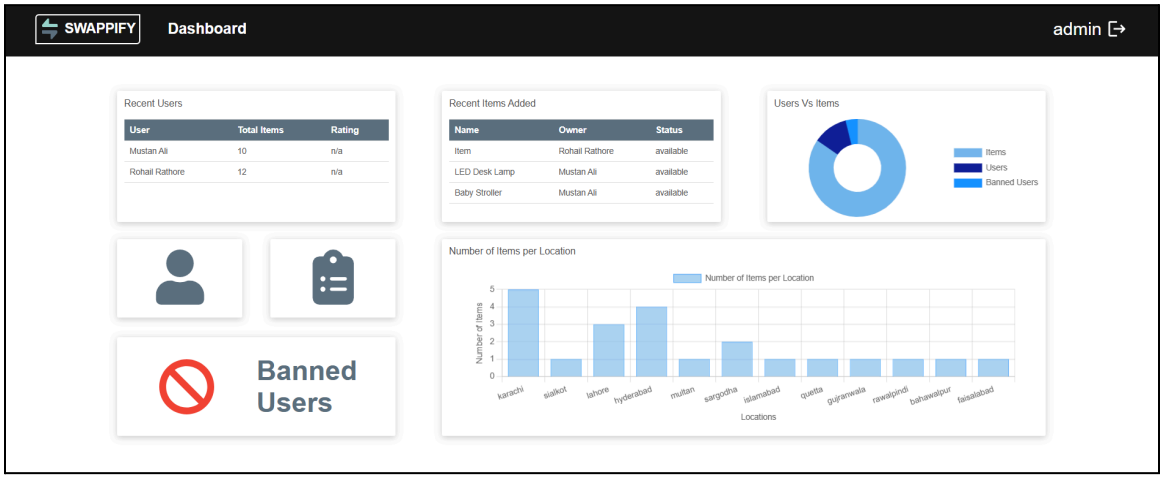

Figure 23. Admin Dashboard

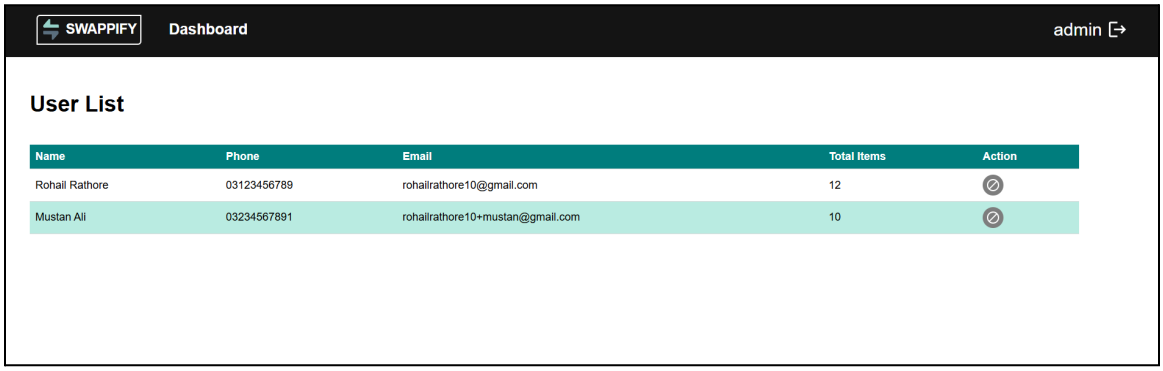

Figure 24. Admin Dashboard: Users List

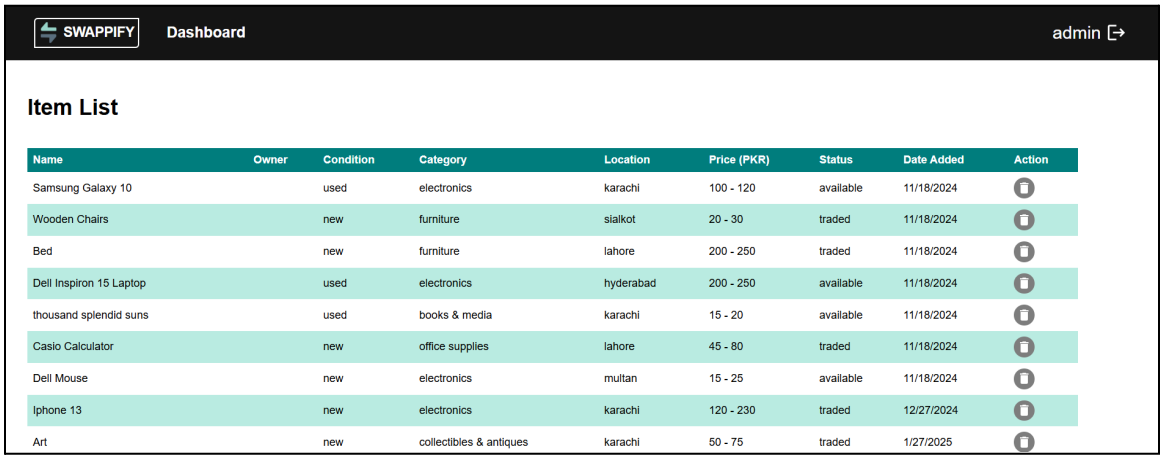

Figure 25. Admin Dashboard: Items List

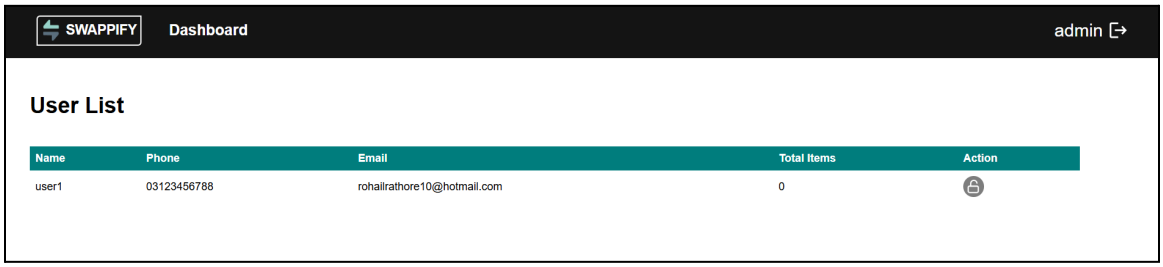

Figure 26. Admin Dashboard: Banned Users List

5.0 Reuse & Relationships to Other Products

This project leverages established frameworks and design patterns to ensure efficiency and consistency, using tools like ReactJS and Axios for their reliability and scalability, which allowed the focus to remain on core functionalities. Open-source libraries such as React-Router-Dom and Mongoose were integrated to reduce development time and maintain a reliable and efficient system. Although some external modules were considered, they were discarded due to compatibility issues, performance concerns, or their inability to meet the project's specific requirements, leading to the decision to develop custom solutions.

6.0 Design Decisions & Tradeoffs

In designing the system, we chose ReactJS for its component based architecture and fast rendering, despite considering alternatives like Vue.js and Angular. For the backend, Node.js with Express.js was selected over frameworks like Django for its lightweight nature and seamless integration with JavaScript. MongoDB was chosen over PostgreSQL for its flexibility in handling unstructured data, which suited our needs better. We decided to use WebSockets for real time communication, ensuring instant updates, which were crucial for enhancing the user experience. These decisions prioritize simplicity, scalability, and efficiency while meeting the project's immediate needs.

7.0 Pseudocode for components

Login:

1. Enter email and password
2. If email and password are correct then
3. Authenticate based on the role which is Trader/Admin.
4. Redirect to profile

SignUp:

1. Enter email, password, full name & mobile number.
2. If validation of details is successful then
3. Send verification email.

Forgot Password:

1. Enter registered email & mobile number.
2. If email & mobile exists & belong to same user then:
3. Send a new password to the email.

Profile Management:

1. Navigate to the profile page.
2. View current profile information.
3. Select "Edit" to update profile fields.
4. If updated fields are valid then:
5. Save the updated information in the database.
6. Display the updated profile information.
7. If inputs are invalid, display appropriate error messages.

List Item:

1. Navigate to the "Add New Item" section.
2. Fill out the item details in the provided form.
3. If all fields are filled & data formats are valid then:
4. Save the item in the database.
5. If inputs are invalid, display appropriate error messages.

Edit Item:

1. Navigate to my list & select an item to edit.
2. View the current details of the selected item in an editable form.
3. Update the desired fields and click the "Save" button.
4. If all fields are filled & data is valid then:
5. Save the updated item details in the database.
6. Display "Item successfully updated."
7. If inputs are invalid, display appropriate error messages.

Delete Item:

1. Navigate to my list select an item to delete.
2. If the item is selected, then:
3. Ask for confirmation, if confirmation is given then
4. Remove the item from the listings.
5. Update the database.
6. If a system error occurs, display an error message.

View Item Listings:

1. Navigate to the item listings page.
2. If items are available then:
3. Retrieve & display a list of all available items with essential details.
4. Allow the user to select an item to view more details
5. If no items are available, display "No items available."
6. If a system error occurs, display an error message.

Search & Filters:

1. Enter a keyword in the search bar or select filter options
2. If items match the search term or filters then:

3. Display a list of filtered items.
4. Allow the user to select an item to view more details
5. If no results are found, display "No items found."
6. If a system error occurs, display an error message.

Add to Wishlist:

1. Navigate to an item & click "Add to Wishlist.":
2. Add the item to the user's wishlist.
3. Display "Item added to wishlist."
4. If a system error occurs, display an error message.

View Wishlist:

1. Navigate to the wishlist section through the profile.
2. If the wishlist is not empty, then:
3. Retrieve & display the list of saved items with details such as item name, image & description.
4. If the user clicks on an item, display the full details & trade options for that item.
5. If the wishlist is empty, display: "Your wishlist is empty"
6. If a system error occurs display an error message

Item Detail:

1. Click on an item from the item listing page or wishlist.
2. Retrieve & display item details such as name, description, condition, category, images, and owner information..
3. Display available actions.
4. If the user selects an action, perform the action.
5. If a system error occurs display an error message

Trade Proposal:

1. Navigate to an item listing you want to trade for and click "Propose Trade."
2. Select your own item to offer in exchange.
3. Submit the trade proposal by clicking the "Submit" button.
4. If the proposal is submitted:
5. Save the proposal in the database.
6. Notify the other user about the trade proposal.
7. Display confirmation message: "Trade proposal sent."
8. If the user cancels the proposal before submission:
9. Abort the trade proposal, and no data is saved.
10. If a system error occurs, display an error message..

Accept Trade:

1. Navigate to trade proposals and select a pending trade to review.
2. The system displays the trade details, including the item offered.
3. Click the "Accept" button to confirm the trade.
4. The system finalizes the trade by updating the trade status in the database & notifies the other user of the acceptance.
5. If a system error occurs, display an error message.

Reject Trade:

1. Navigate to trade proposals and select a pending trade to review.
2. The system displays the trade details, including the item offered.
3. Click the "Reject" button to decline the trade.
4. The system updates the trade status in the database.
5. If a system error occurs, display an error message.

Cancel Trade:

6. Navigate to the "Sent Requests" section.
7. The system fetches the details of the selected trade & displays a "Cancel Trade" button.
8. Click the "Cancel Trade" button.
9. The system updates the trade status to "Cancelled" in the database.
10. If a system error occurs, display an error message.

Advanced Trade Management:

1. The user goes to the "Active Trades" section & clicks the "Counter Offer" button for a specific trade.
2. The system opens a trade proposal page with the recipient's item pre-selected.
3. Select an Item for Counter Offer:
4. The user selects an item from the sender's listed items to include in the counter proposal.
5. The system updates the trade proposal with the selected item from the sender.
6. The user clicks the "Send Counter Offer" button.
7. If no item is selected from the sender's list, the system displays an error message.
8. If an item is selected:
9. The system saves the counter proposal in the database.
10. The system notifies the sender about the counter proposal.
11. A confirmation message is displayed: "Counter offer sent successfully."
12. If a system error occurs, display an error message.

Manage Active Trades:

1. The user navigates to the "Active Trades" section.
2. The system displays a list of all active trade proposals, including trade details & status.
3. The user selects a trade to manage and either accepts or rejects it.
4. The system updates the trade status based on the user's action and notifies the other party.
5. If a system error occurs, display an error message.

Ban User Accounts:

1. The admin navigates to the "User Accounts" section.
2. The system displays a list of all user accounts with details such as account status & profile information.
3. The admin selects a user account to ban.
4. The system processes the admin's action, updates the user's account status in the database.
5. If a system error occurs, display an error message.

Delete List Items:

1. The admin navigates to the "Listed Items" section.
2. The system displays a list of all items listed by users, including item details such as title, description, category, and status.
3. The admin selects an item to view or remove.
4. The system processes the admin's action:
5. If the item is removed, it is deleted from the listings and the database.
6. If a system error occurs, display an error message.

Rating and Review:

1. The user navigates to the "Rate Trade" section after completing a trade.
2. The system displays a form allowing the user to provide a rating & write a review.
3. The user submits the rating and review by clicking the "Submit" button.
4. The system validates the input and saves the rating and review to the database.
5. If a system error occurs, display an error message.

Email Notification:

1. A relevant event occurs (e.g., new trade proposal, trade accepted, trade rejection).
2. The system generates an email with the details of the event.

3. The system sends the email to the user's registered email address.

Trade History:

1. The user navigates to the "Trade History" section.
2. The system retrieves & displays a list of all the user's completed trades, including details such as the traded items, trade partners, and trade dates.
3. The user selects a specific trade to view more details.
4. The system displays the full details of the selected trade, including items traded, counterparties, and the status of the trade.

Send Messages:

1. User selects a user to message
2. Display a messaging interface with a text input field.
3. User types a message and clicks "Send".
4. Message is saved in the database.
5. Recipient receives the message in real-time.

Remove from Wishlist:

1. The user navigates to their wishlist.
2. The system displays a list of all items currently in the user's wishlist.
3. The user selects an item to remove and clicks the "Remove" button.
4. The system removes the item from the wishlist and updates the database.

8.0 Appendices

Class Diagram	Describes the structure of a system
Object Diagram	Expresses possible object combinations of a specific Class Diagram
Statechart Diagram	Expresses possible states of a class (or a system)
Activity Diagram	Describes activities and actions taking place in a system
Sequence Diagram	Shows one or several sequences of messages sent among a set of objects
Collaboration Diagram	Describes a complete collaboration among a set of objects
Use-case Diagrams	Illustrates the relationships between use cases
Component Diagram	A special case of a Class Diagram used to describe components within a software system
Deployment Diagram	A special case of a Class Diagram used to describe hardware within the overall system architecture
System Block diagram	A diagram showing the major components of the system with its interconnections and external interfaces

Class Diagram

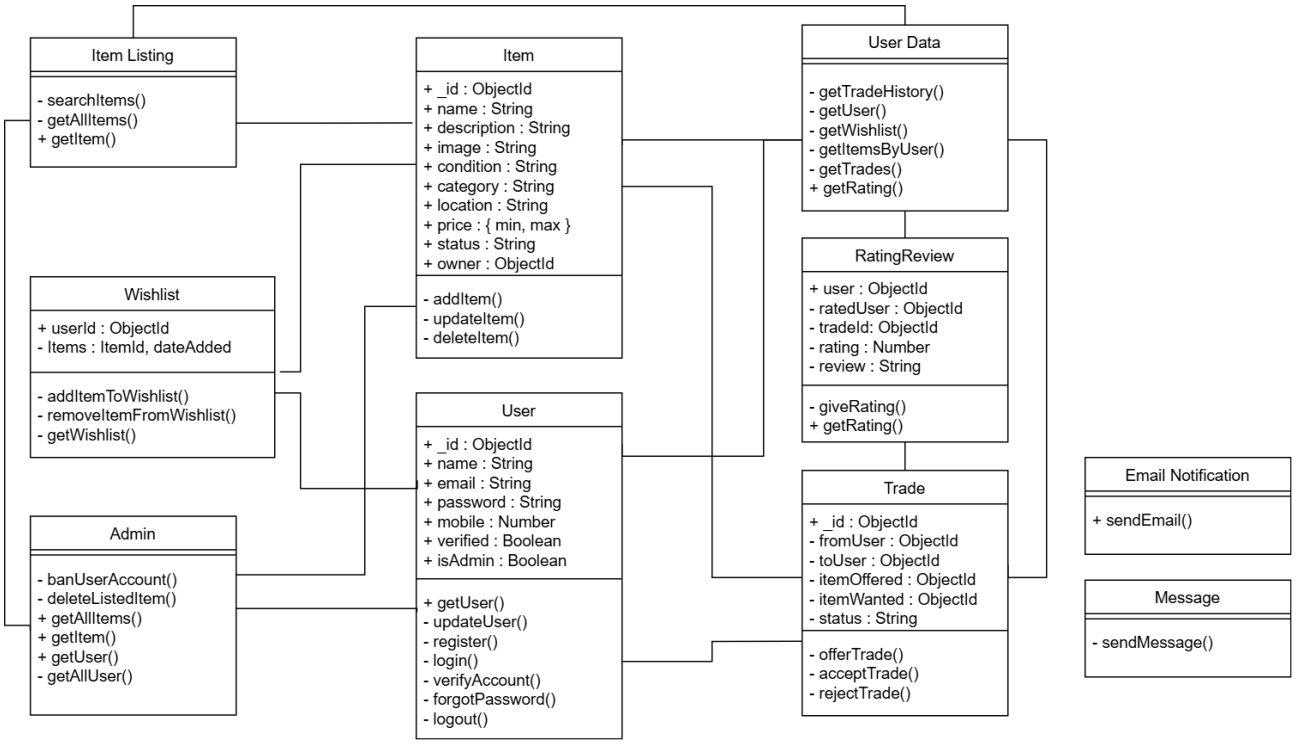

Figure 27: Class Diagram

State Chart Diagram

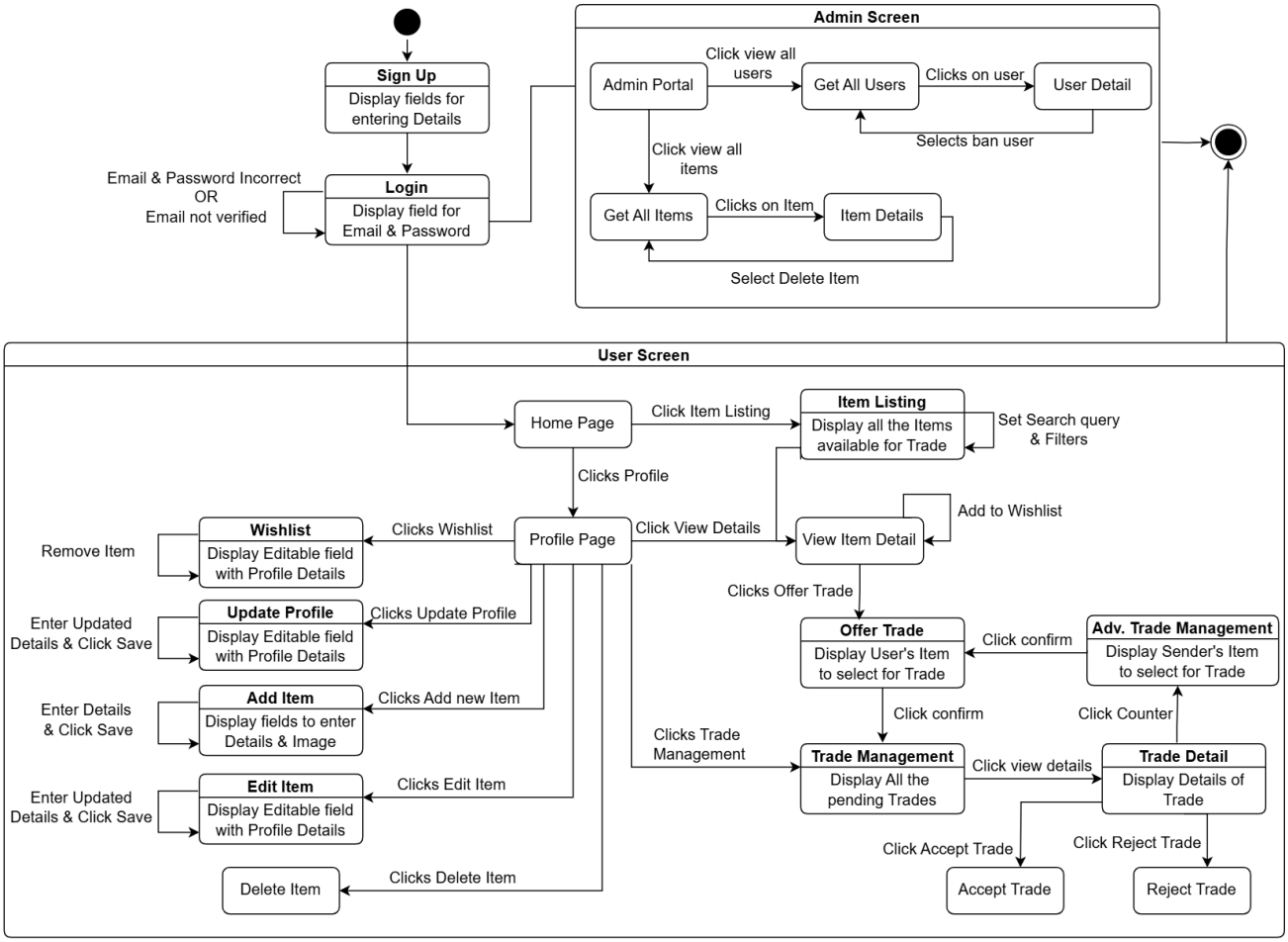

Figure 28: State Chart Diagram

Component Diagram

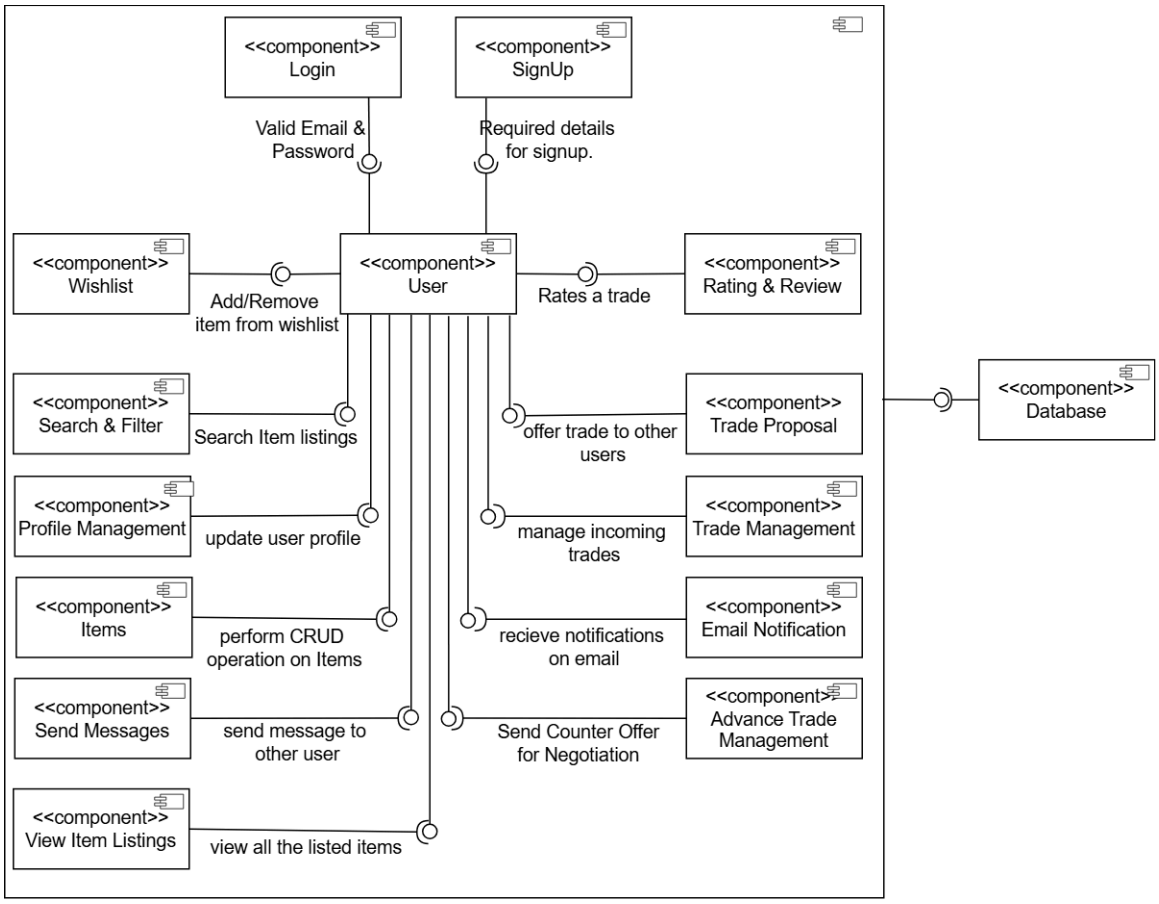

Figure 29: Component Diagram (User)

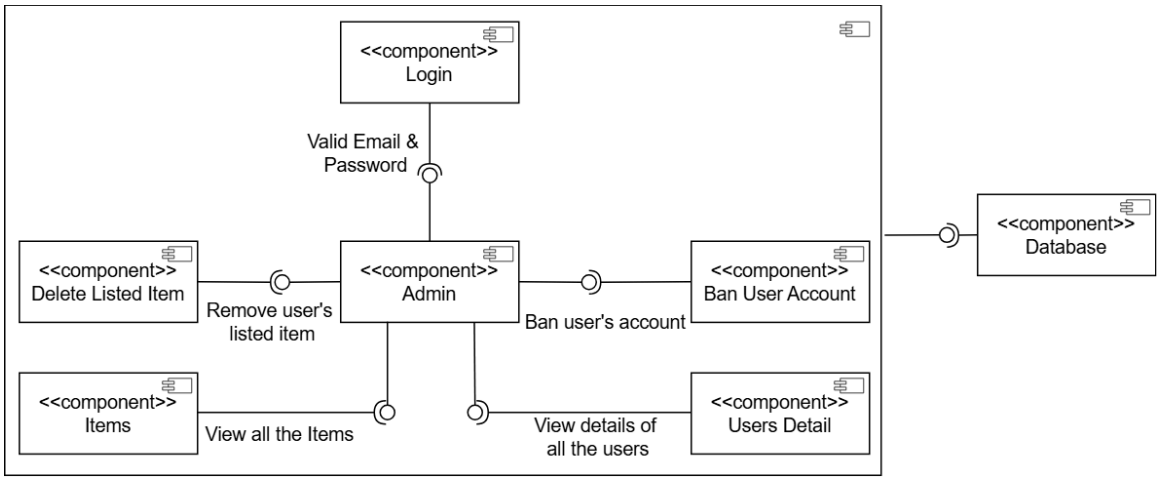

Figure 30: Component Diagram (Admin)

Use Case Diagram

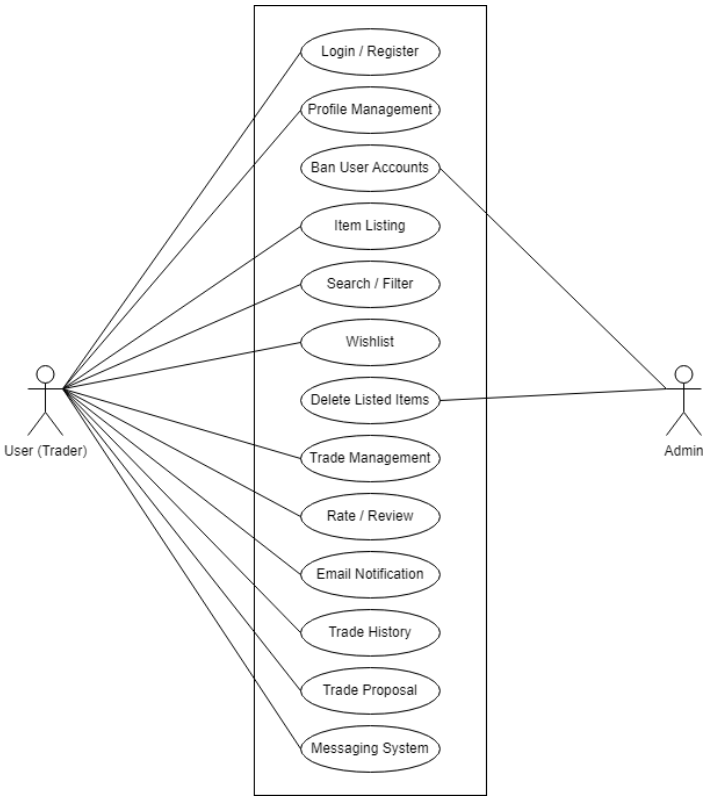

Figure 31: Use Case Diagram

Entity Relationship Diagram(ERD)

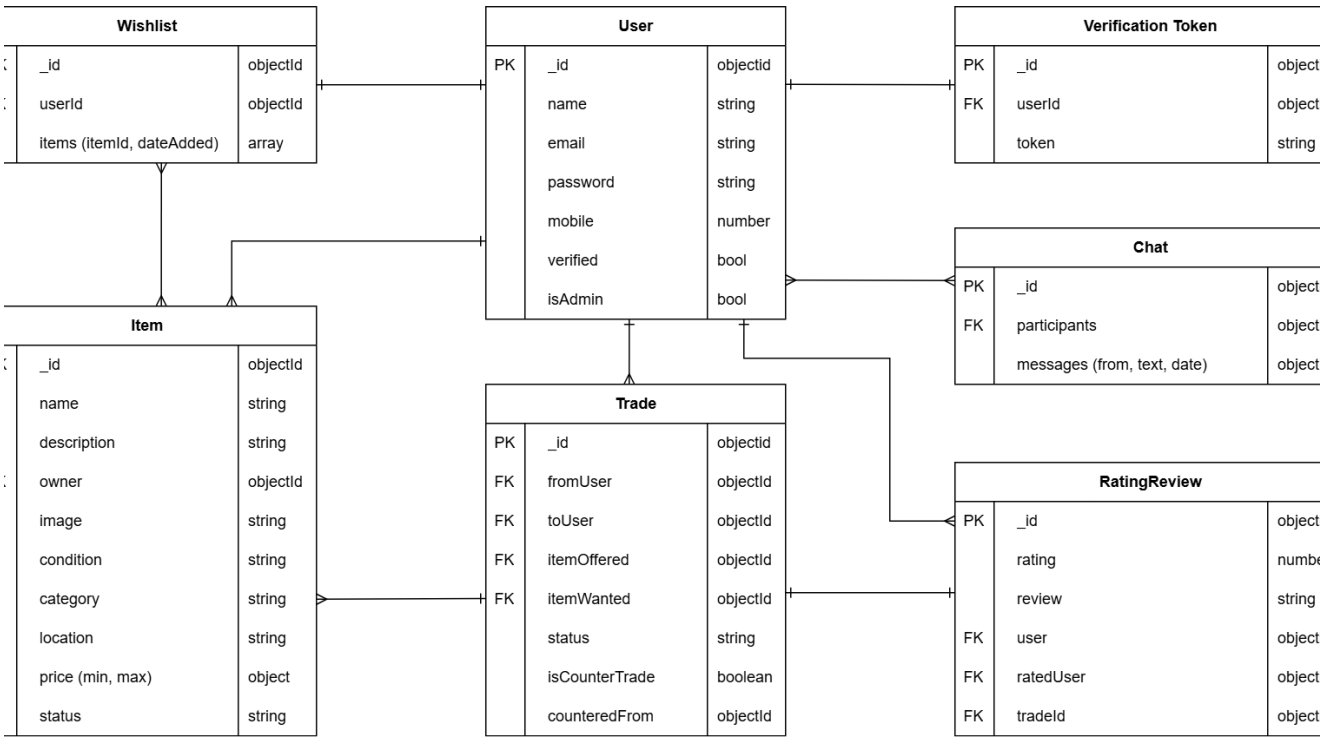

Figure 32: Entity Relationship Diagram

Activity Diagram

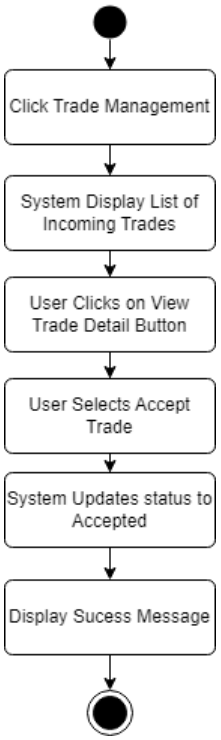

Figure 33: Activity Diagram (Accept Trade)

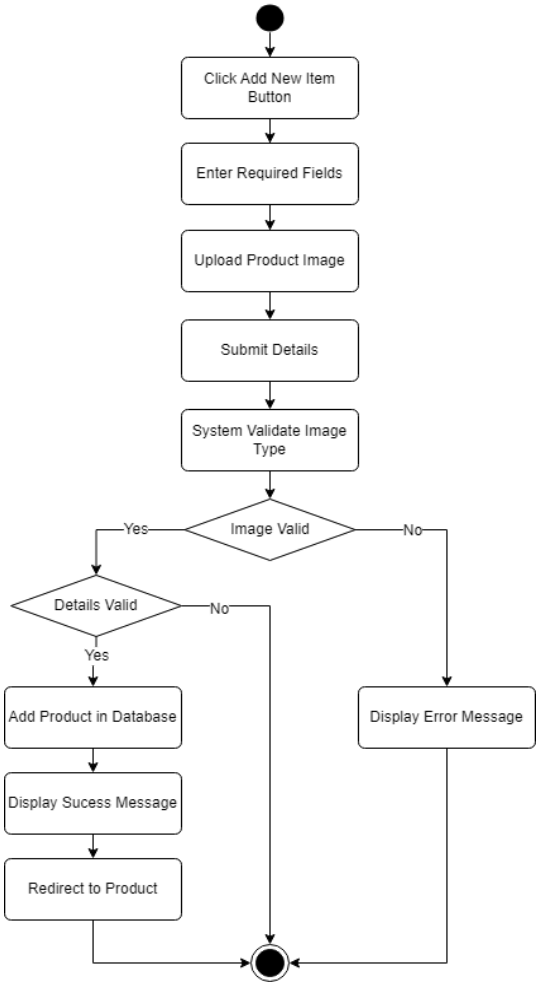

Figure 34: Activity Diagram (Add Item)

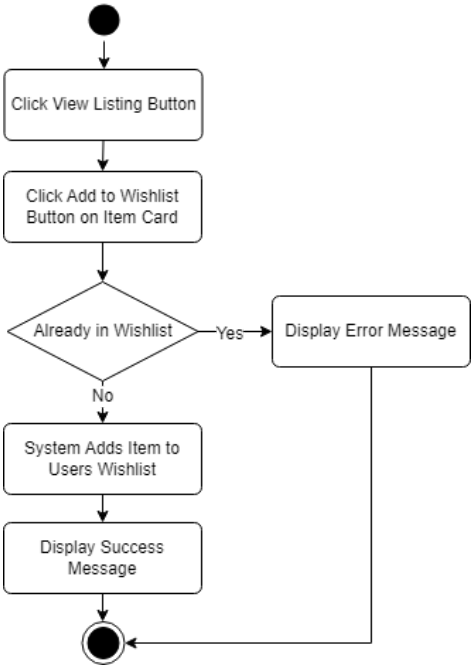

Figure 35: Activity Diagram (Add to Wishlist)

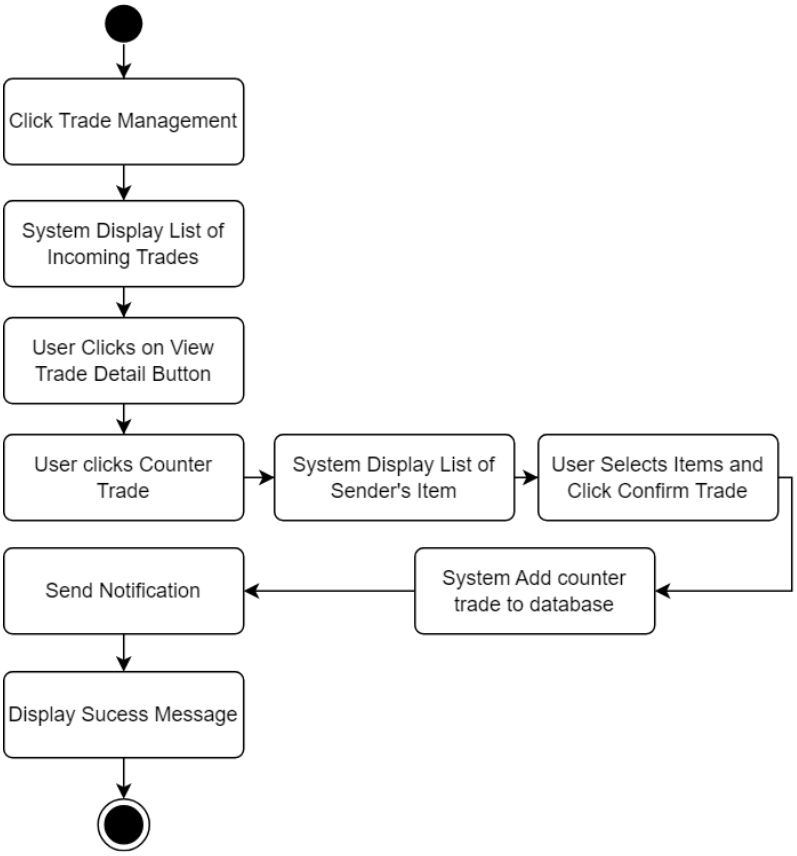

Figure 36: Activity Diagram (Advance Trade Management)

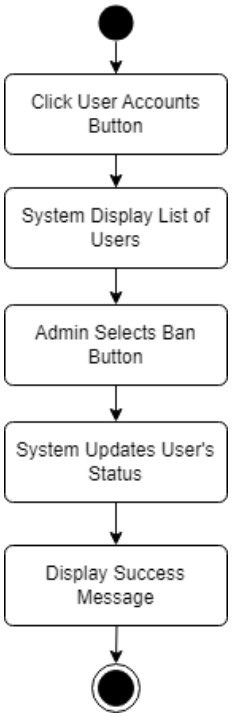

Figure 37: Activity Diagram (Ban User Account)

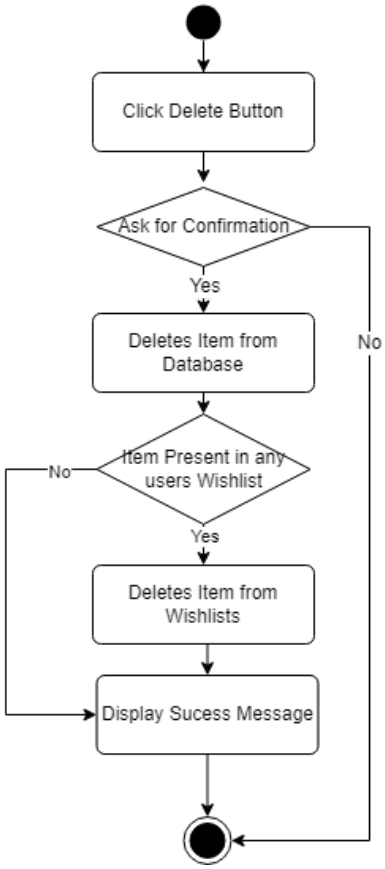

Figure 38: Activity Diagram (Delete Item)

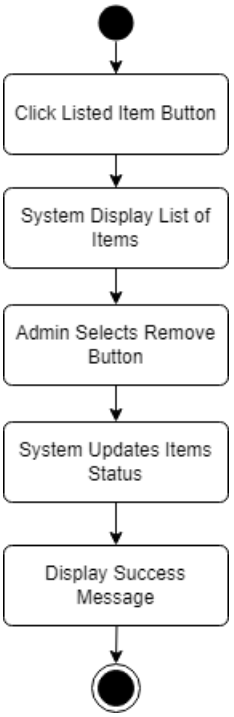

Figure 39: Activity Diagram (Delete List Item)

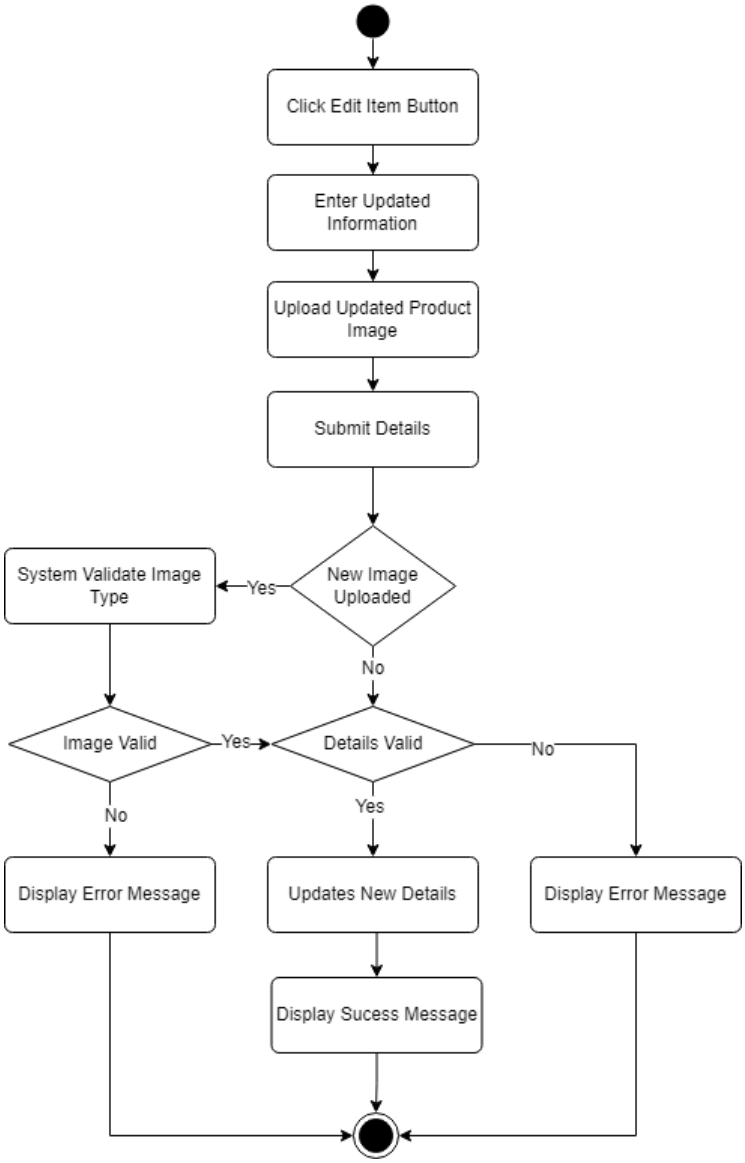

Figure 40: Activity Diagram (Edit Item)

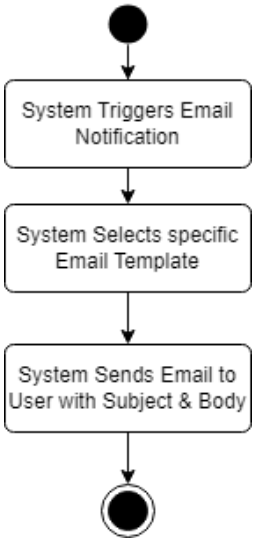

Figure 41: Activity Diagram (Email Notification)

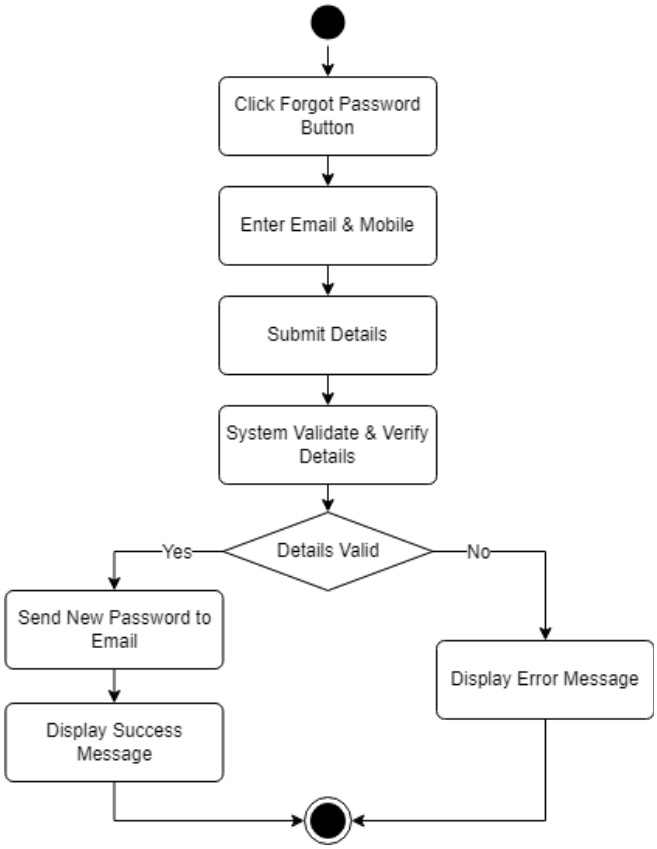

Figure 42: Activity Diagram (Forget Password)

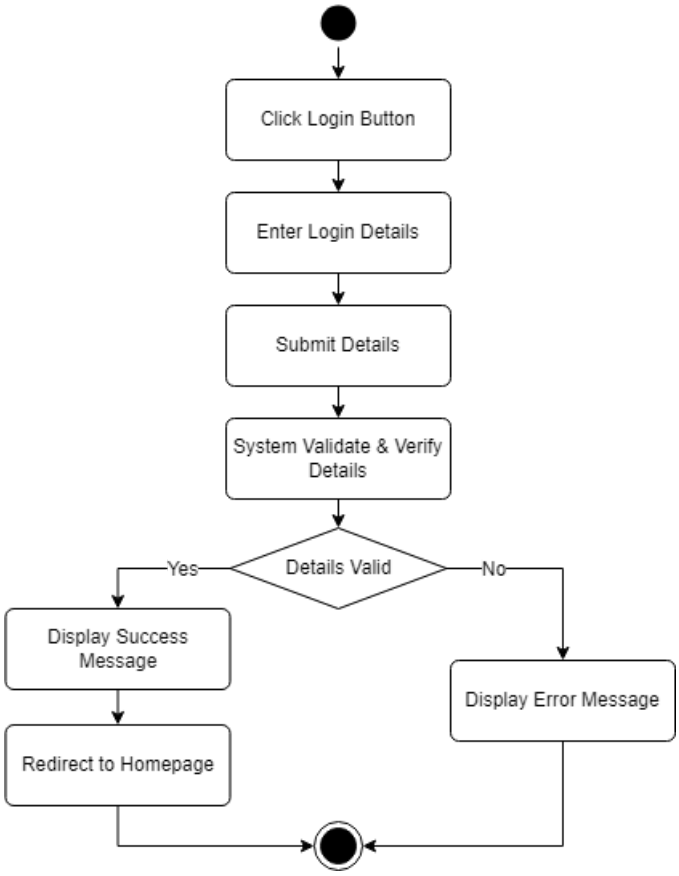

Figure 43: Activity Diagram (Login)

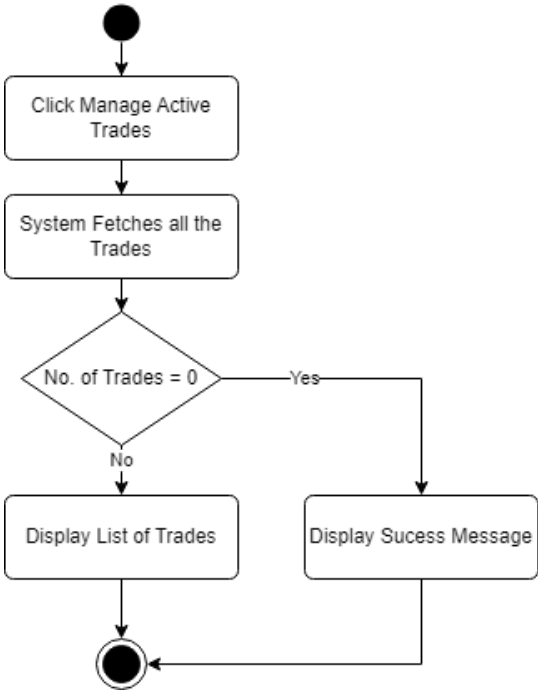

Figure 44: Activity Diagram (Manage Active Trade)

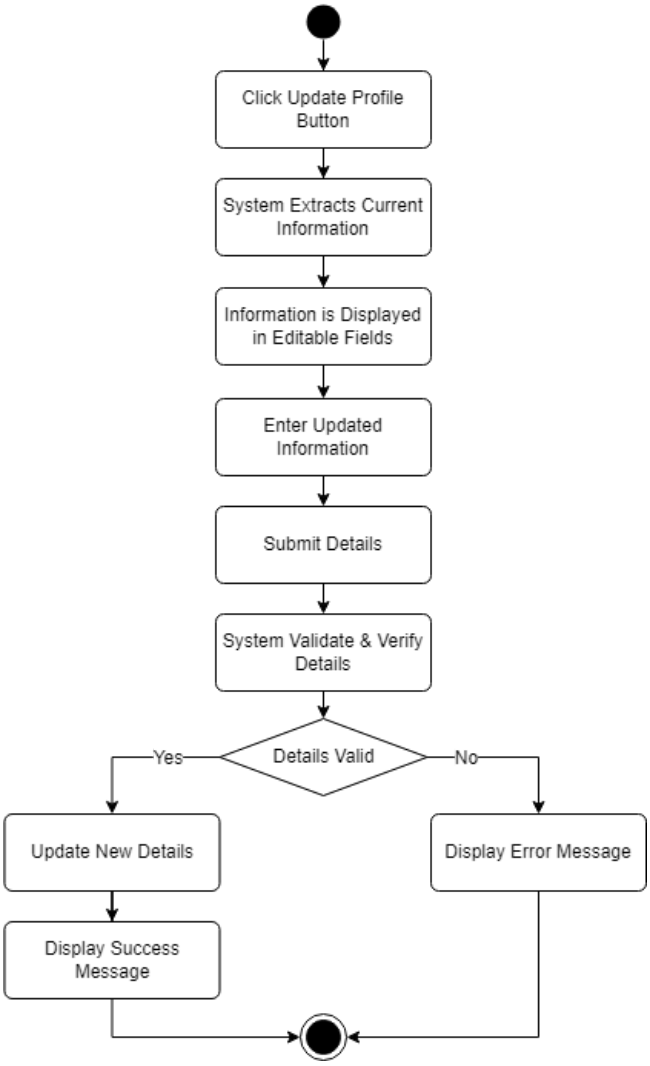

Figure 45: Activity Diagram (Profile Management)

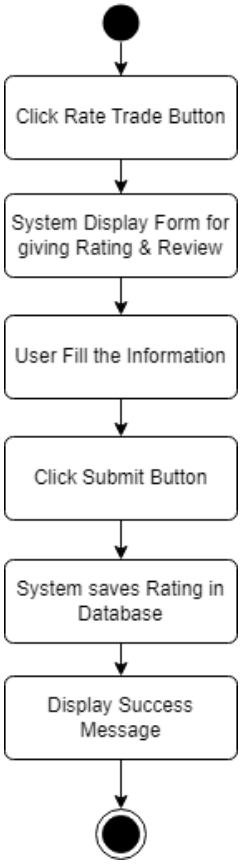

Figure 46: Activity Diagram (Rating & Review)

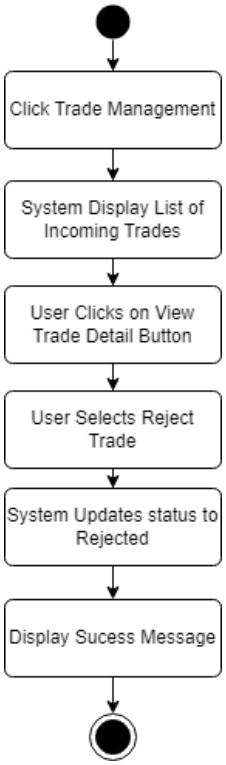

Figure 47: Activity Diagram (Reject Trade)

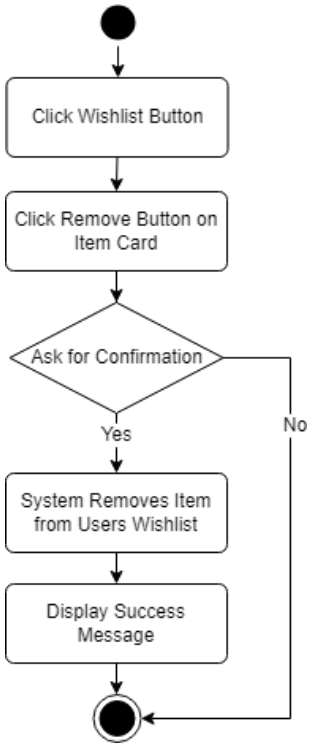

Figure 48: Activity Diagram (Remove from Wishlist)

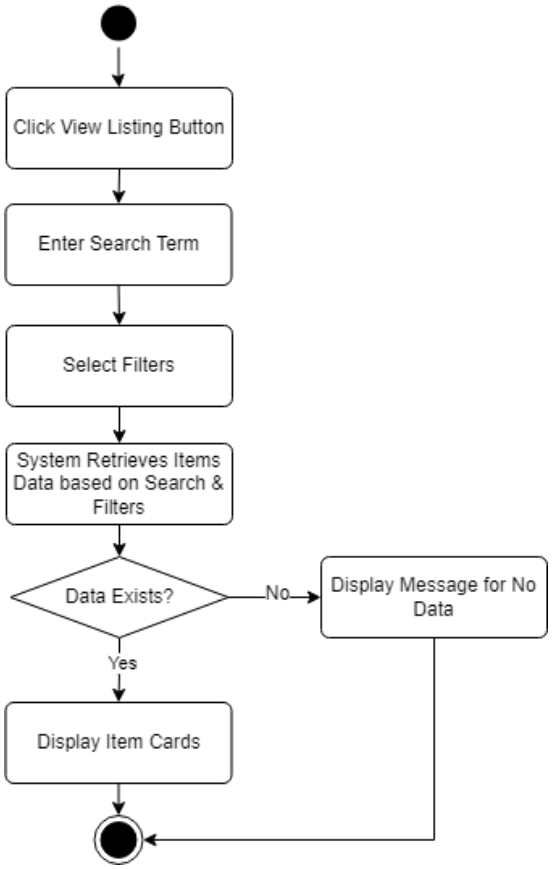

Figure 49: Activity Diagram (Search & Filter)

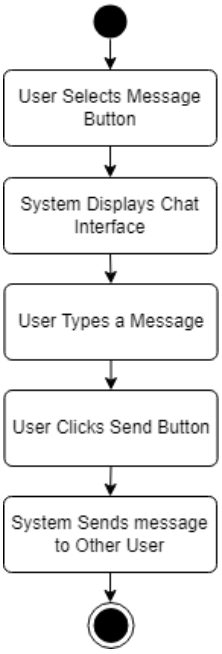

Figure 50: Activity Diagram (Send Message)

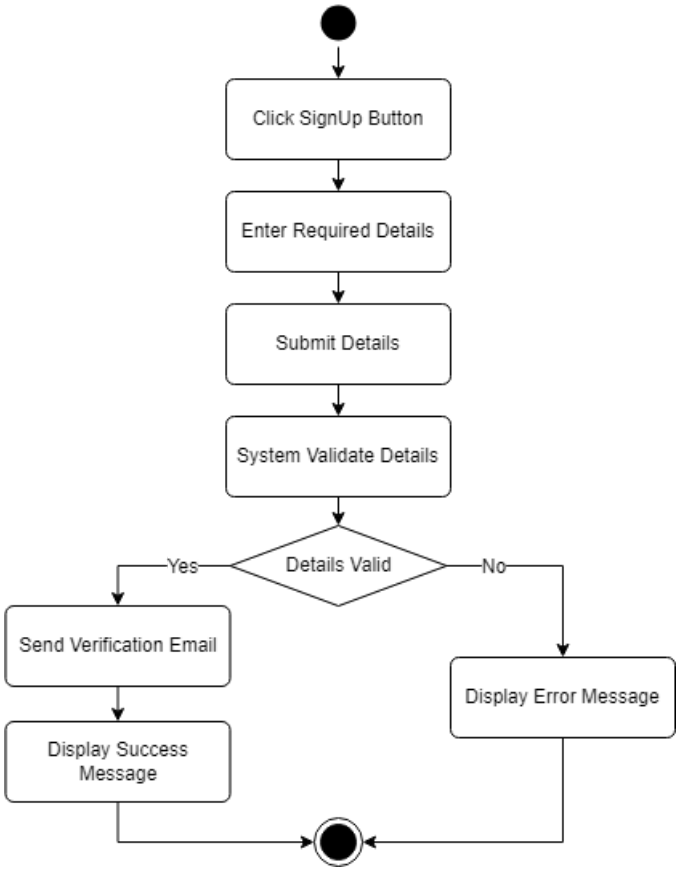

Figure 51: Activity Diagram (Sign Up)

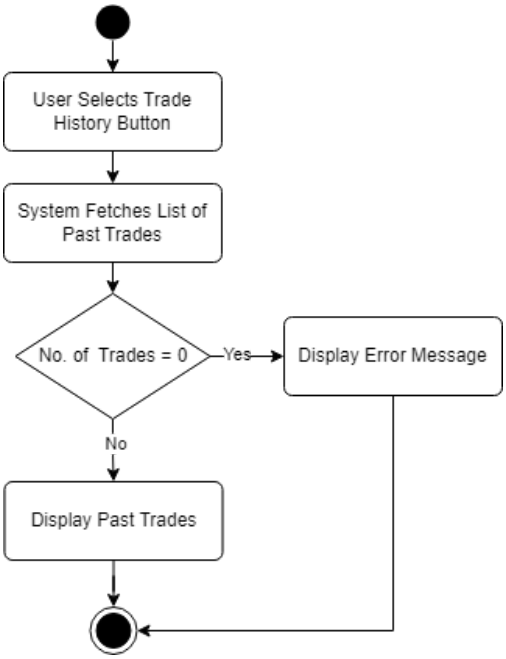

Figure 52: Activity Diagram (Trade History)

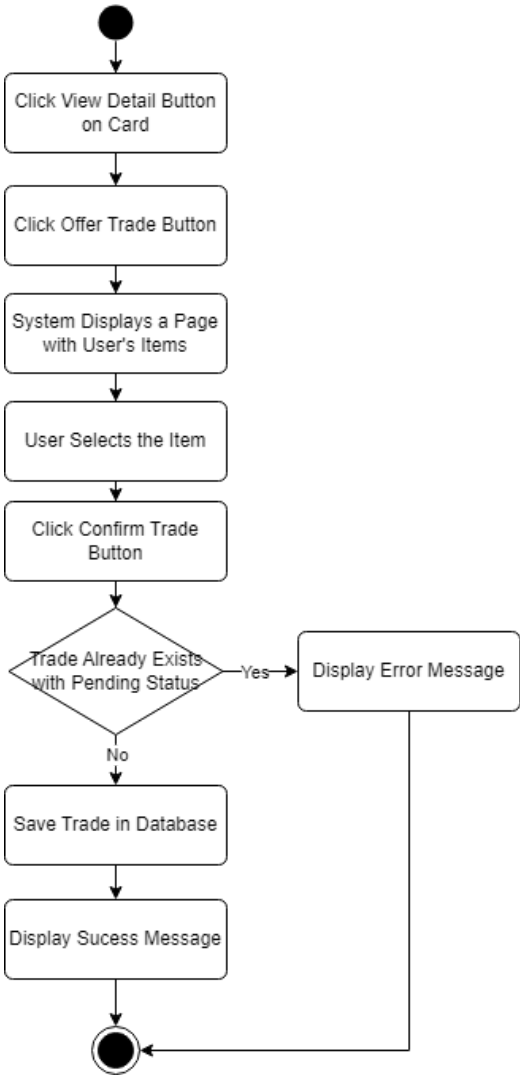

Figure 53: Activity Diagram (Trade Proposal)

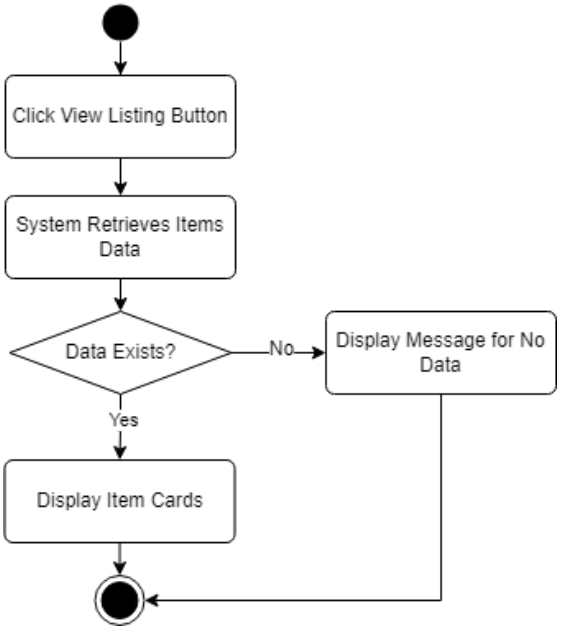

Figure 54: Activity Diagram (View Listing)

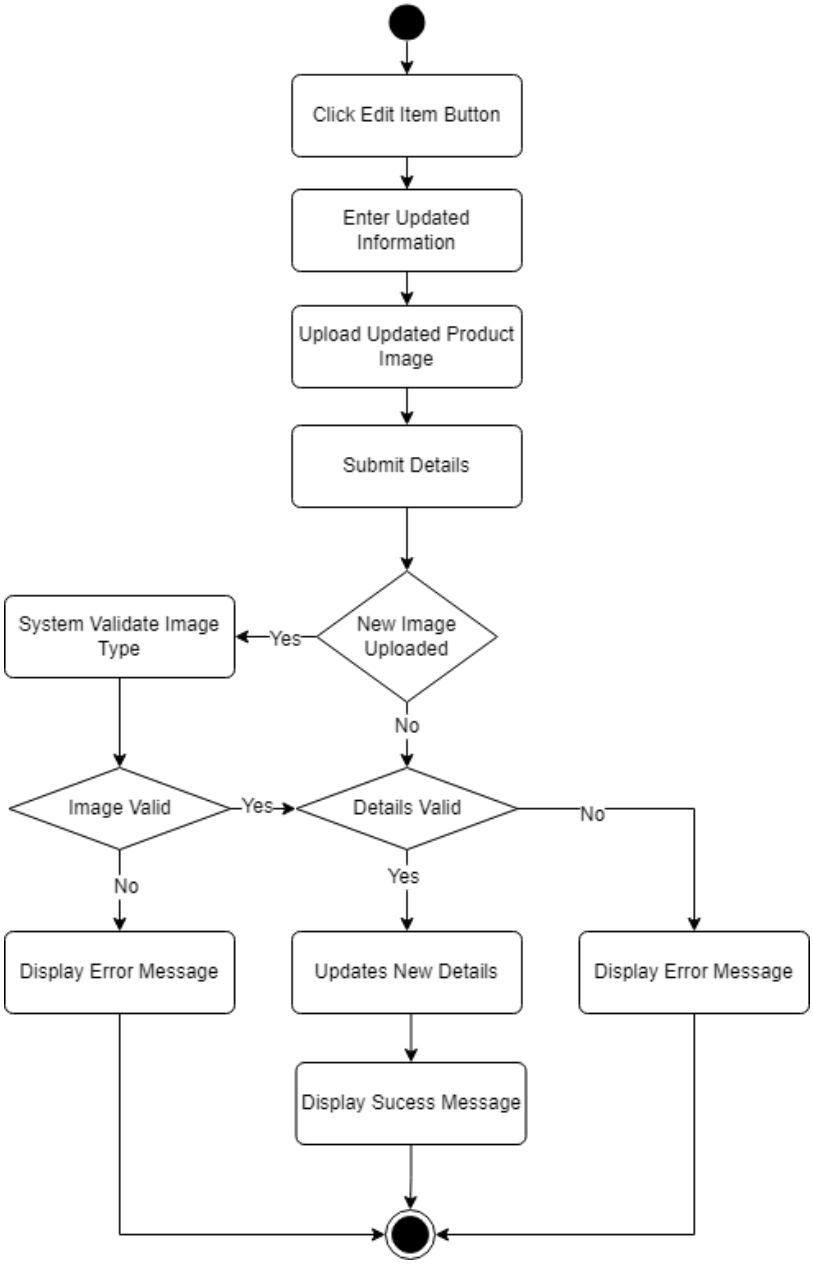

Figure 55: Activity Diagram (Edit Item)

Sequence Diagram

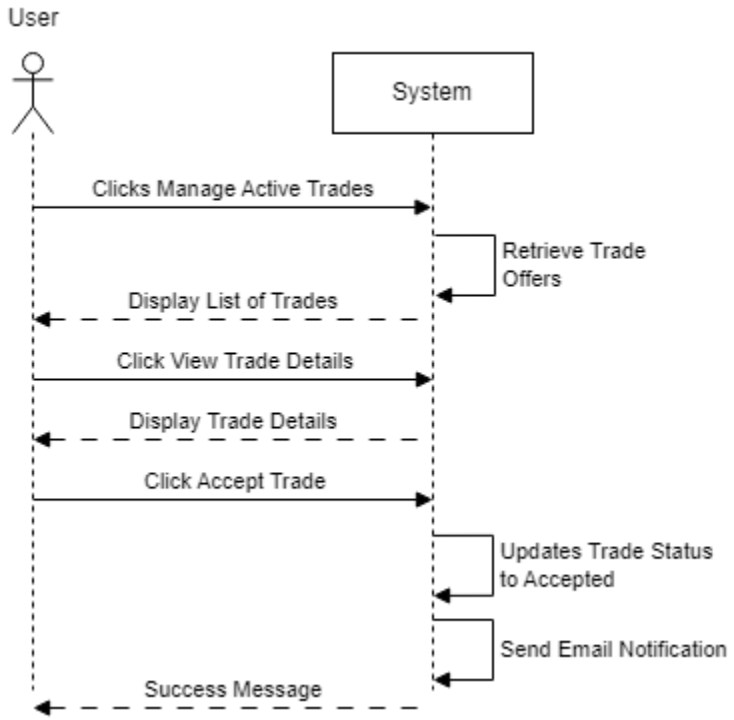

Figure 56: Sequence Diagram (Accept Trade)

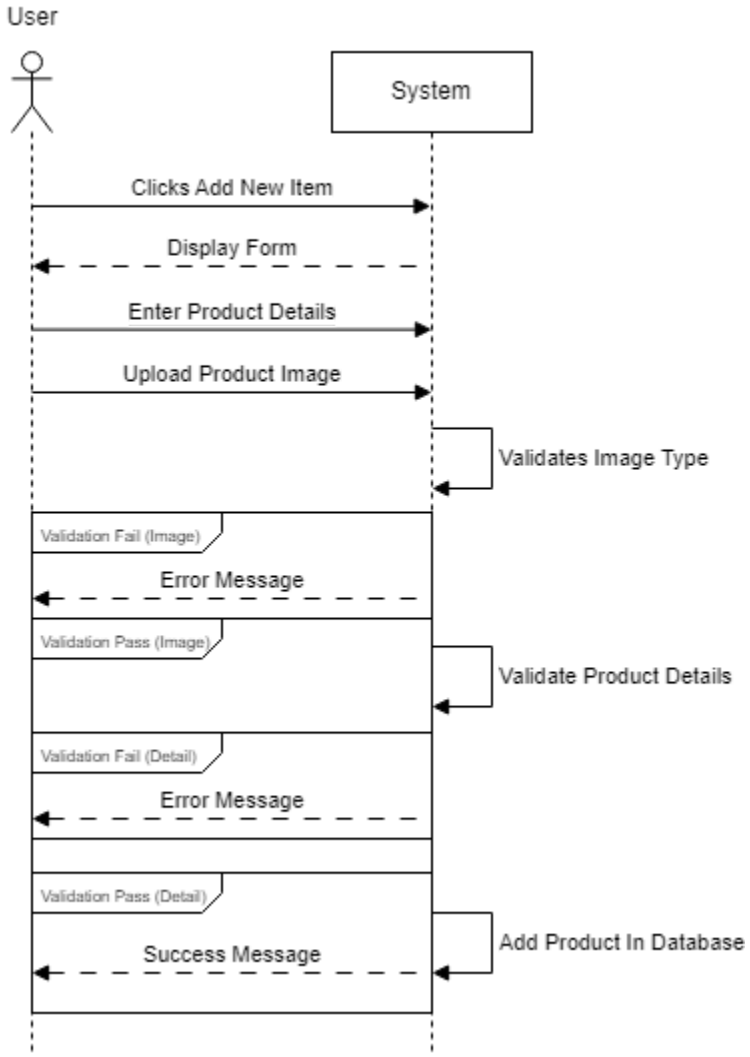

Figure 57: Sequence Diagram (Add Item)

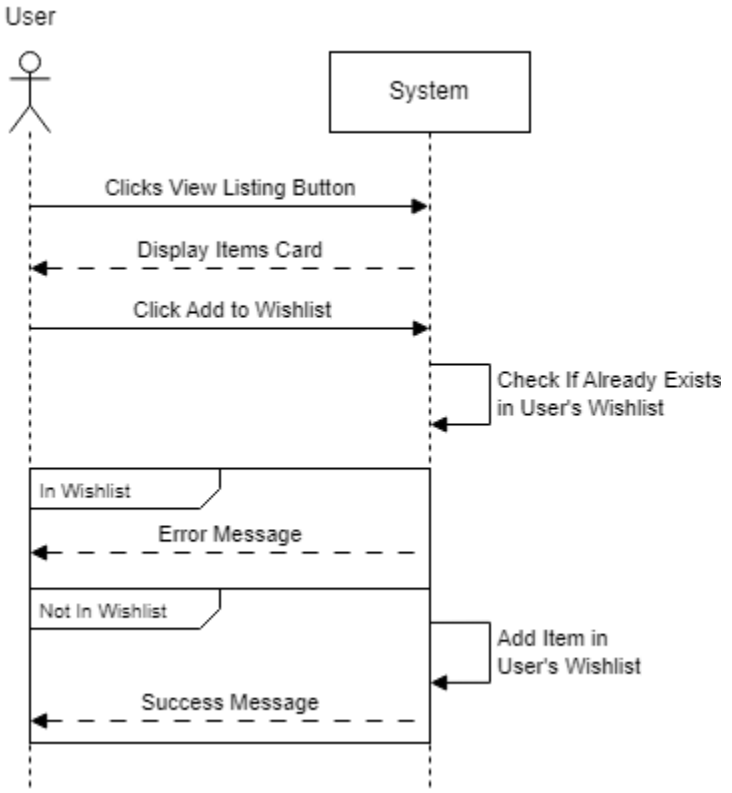

Figure 58: Sequence Diagram (Add to Wishlist)

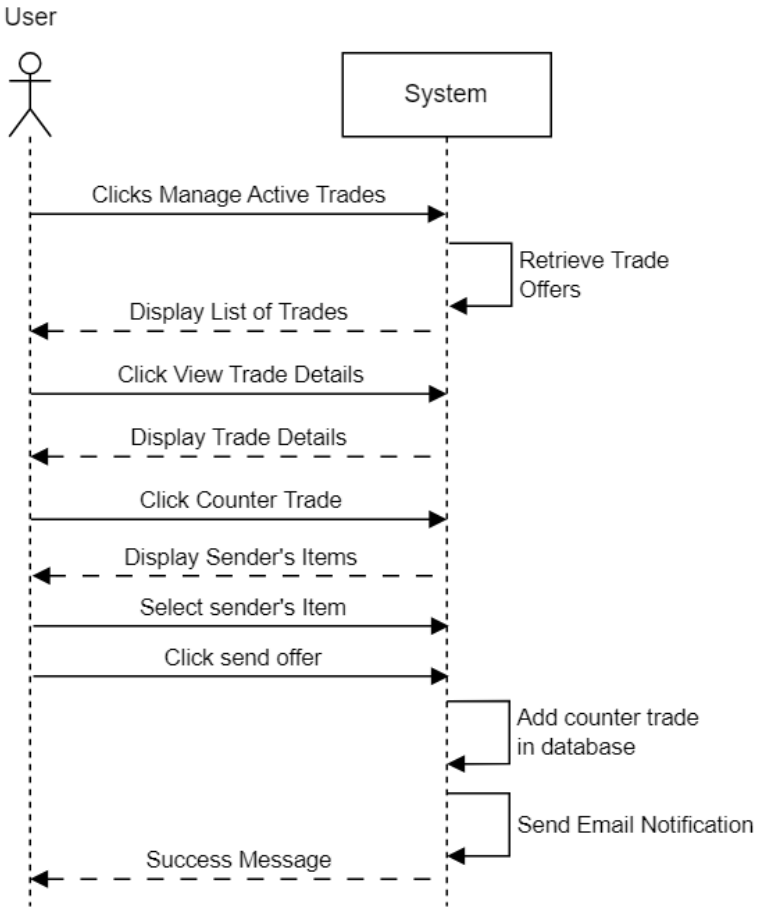

Figure 59: Sequence Diagram (Advance Trade Management)

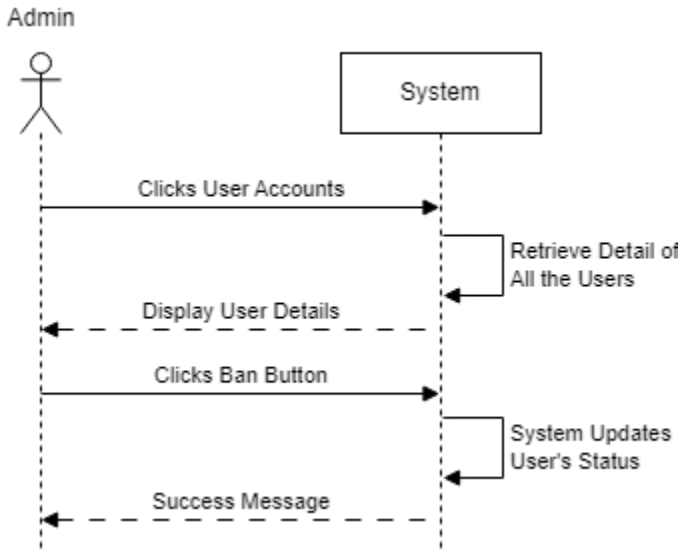

Figure 60: Sequence Diagram (Ban User Account)

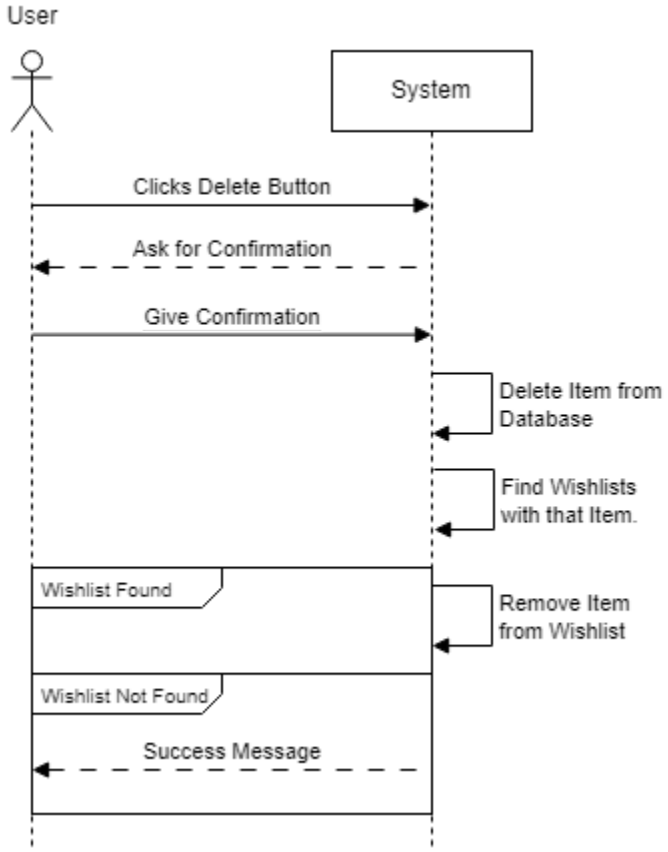

Figure 61: Sequence Diagram (Delete Item)

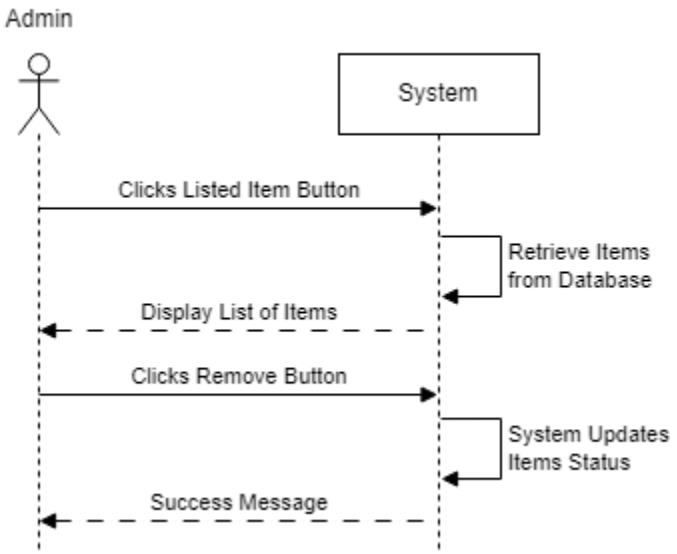

Figure 62: Sequence Diagram (Delete Listed Item)

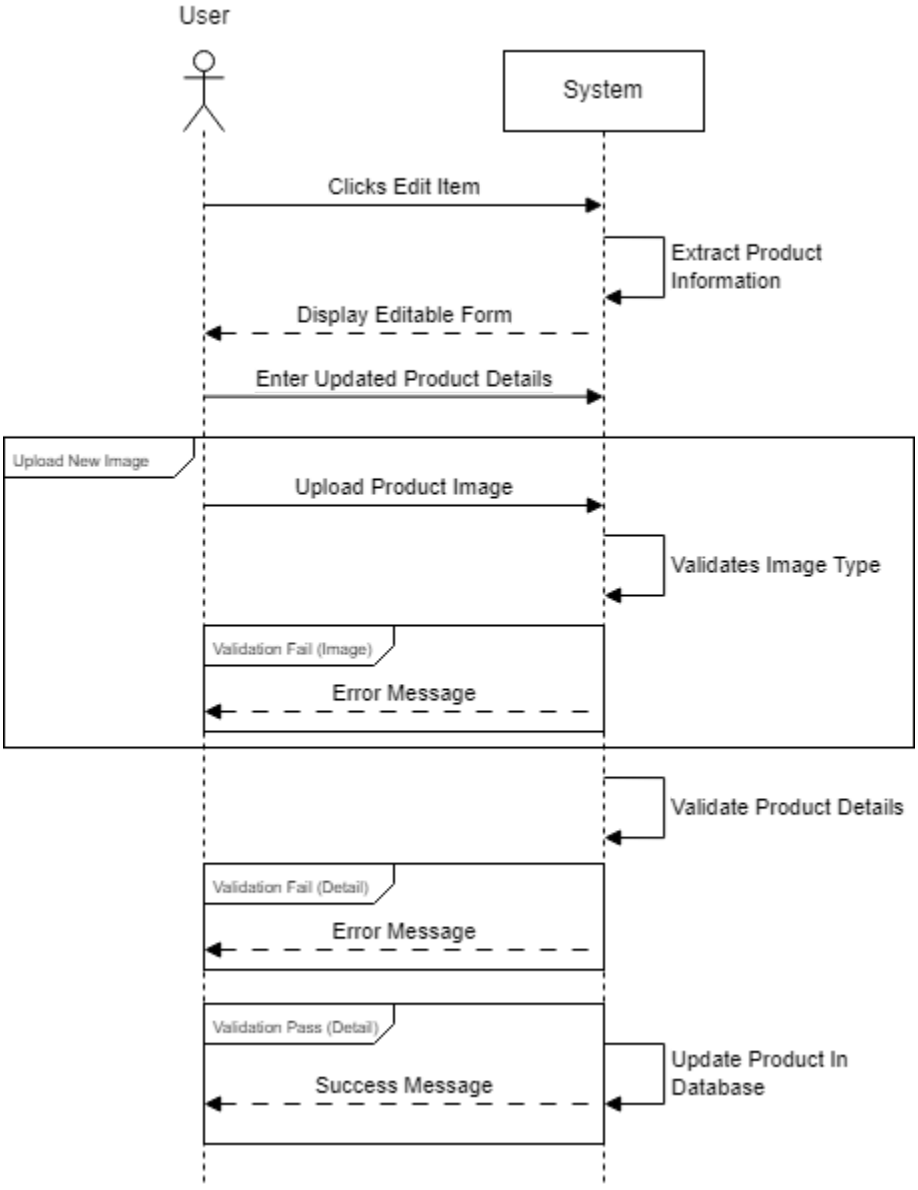

Figure 63: Sequence Diagram (Edit Item)

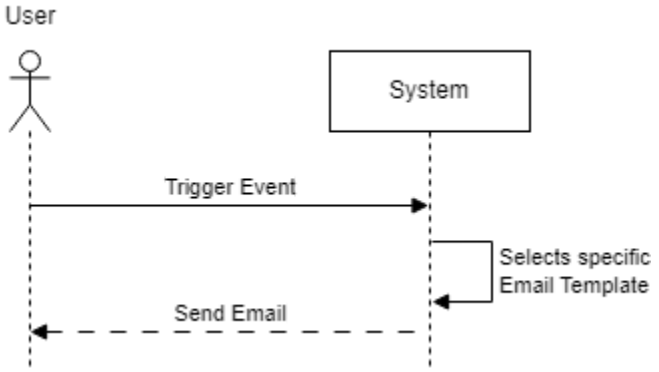

Figure 64: Sequence Diagram (Email Notification)

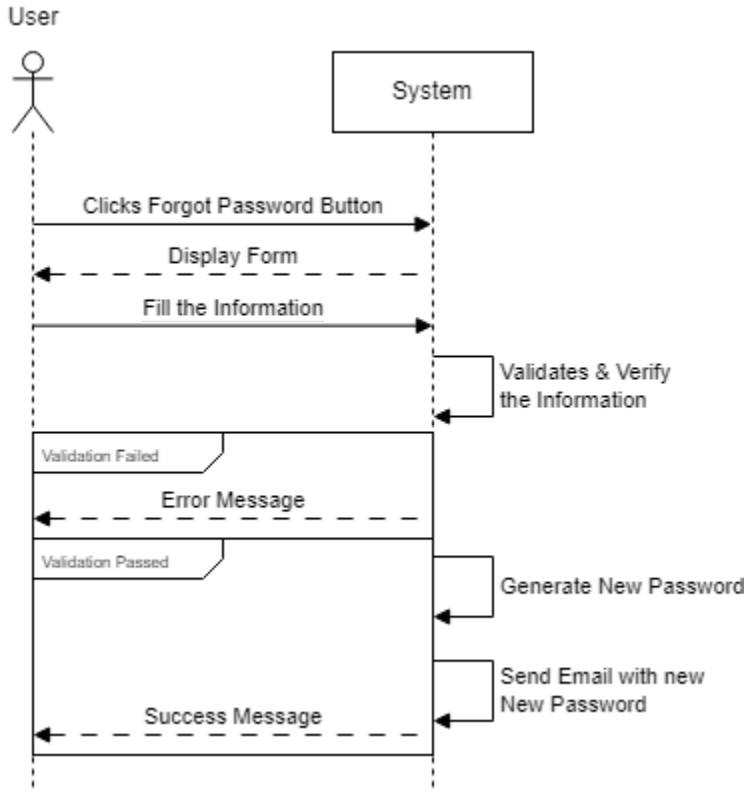

Figure 65: Sequence Diagram (Forget Password)

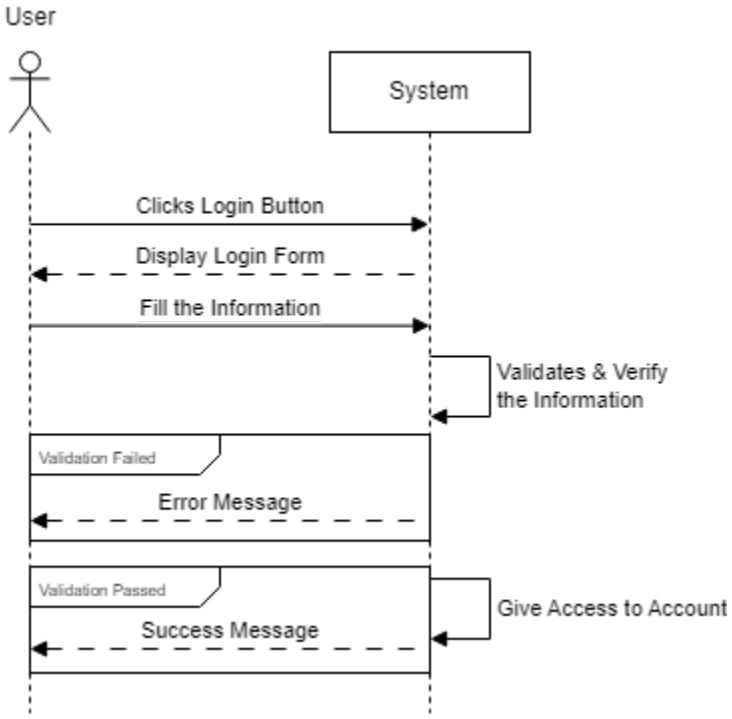

Figure 66: Sequence Diagram (Login)

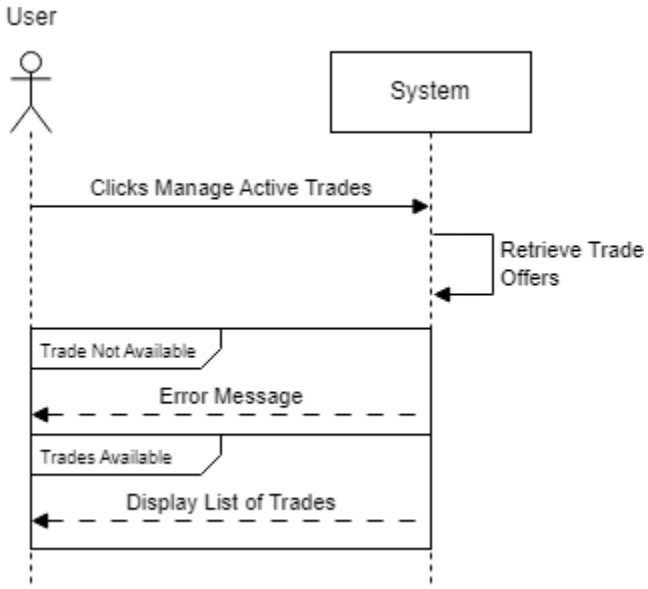

Figure 67: Sequence Diagram (Manage Active Trade)

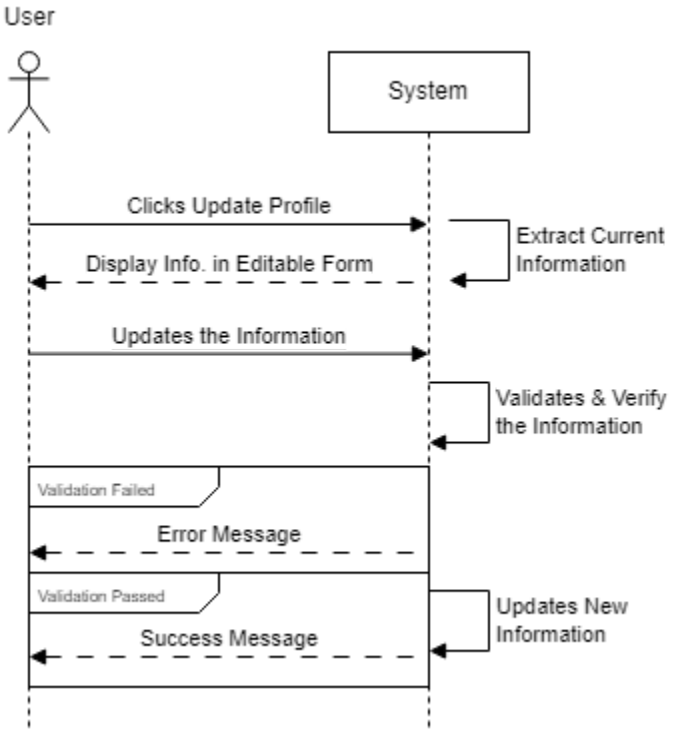

Figure 68: Sequence Diagram (Profile Management)

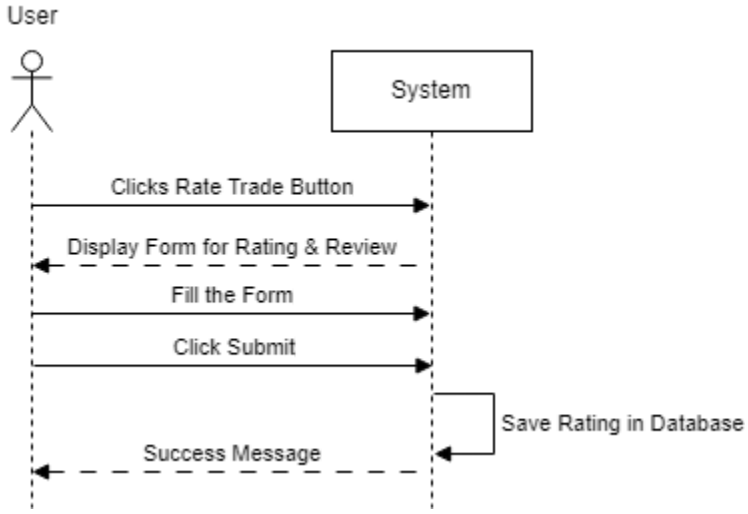

Figure 69: Sequence Diagram (Rating & Review)

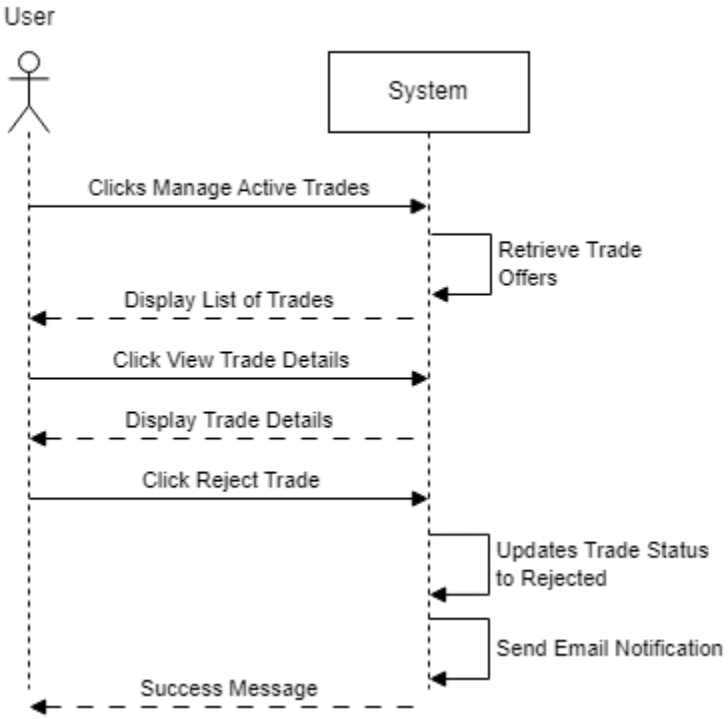

Figure 70: Sequence Diagram (Reject Trade)

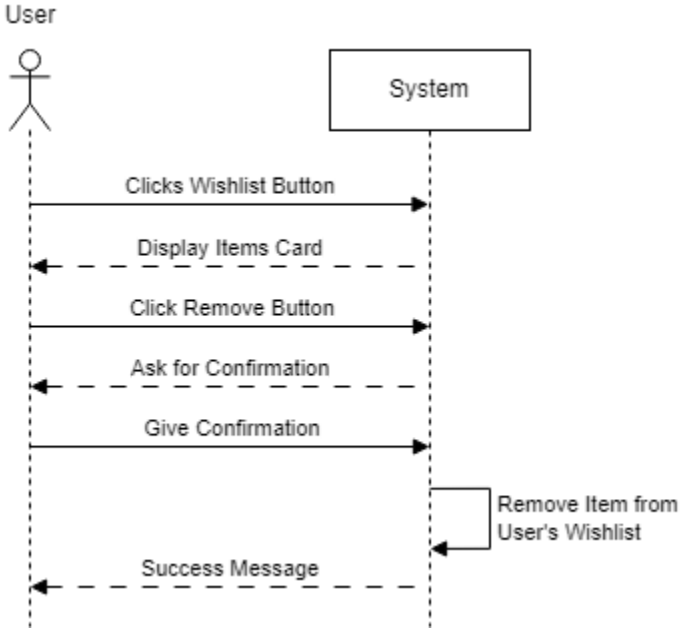

Figure 71: Sequence Diagram (Remove from Wishlist)

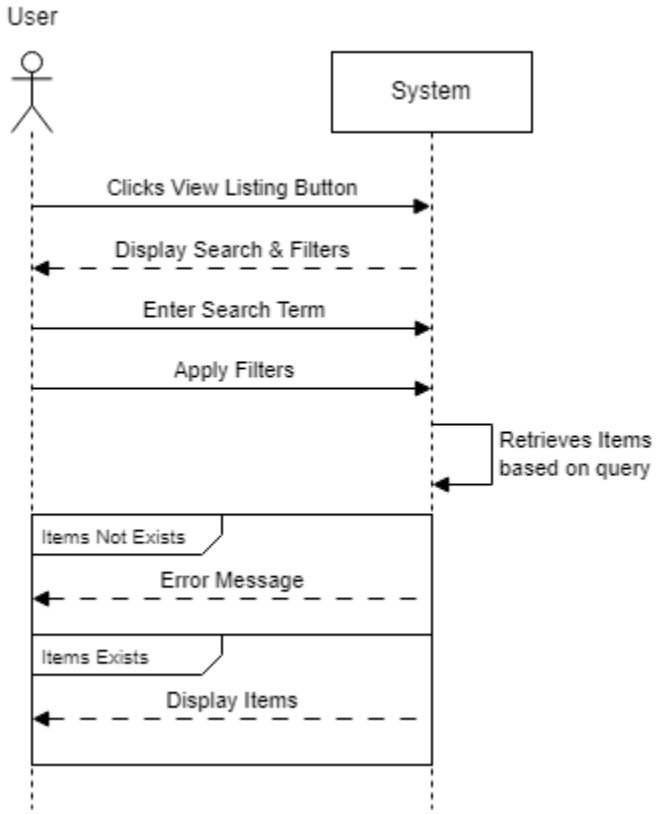

Figure 72: Sequence Diagram (Search & Filter)

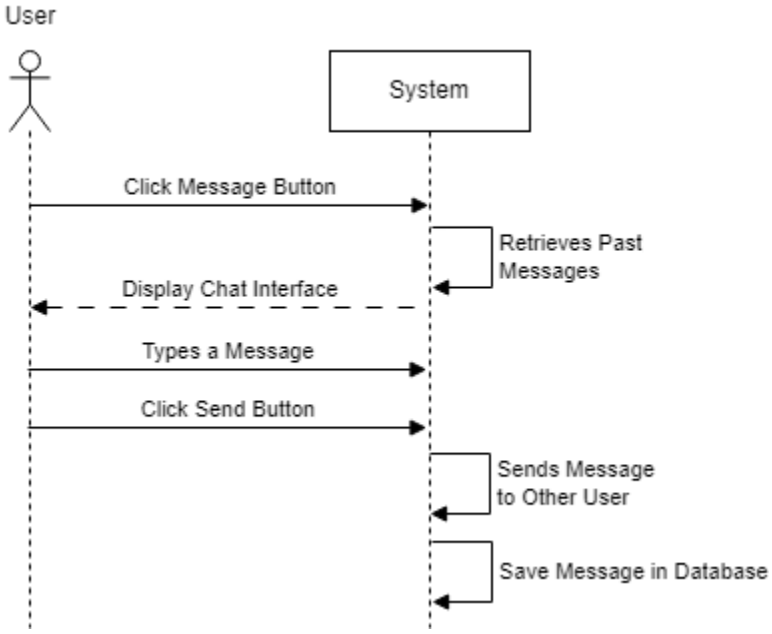

Figure 73: Sequence Diagram (Send Message)

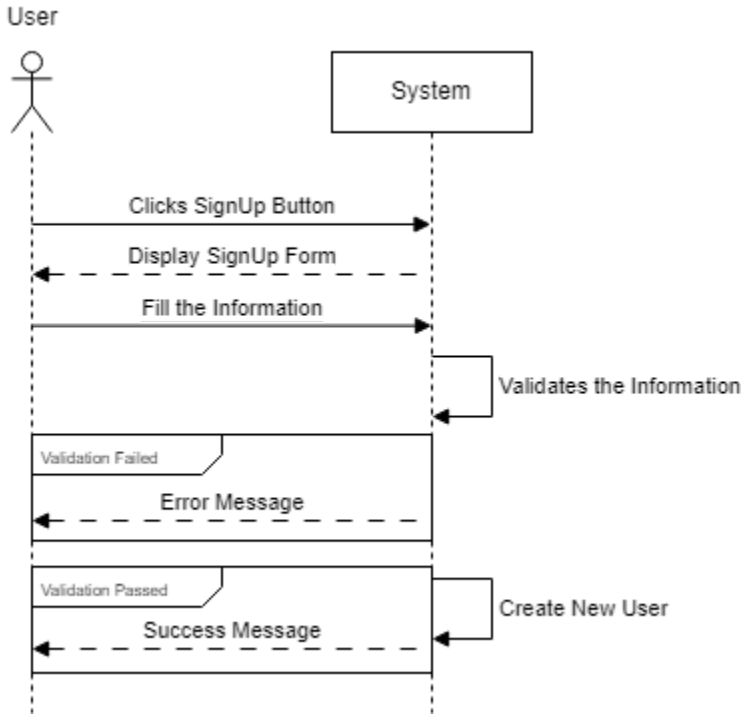

Figure 74: Sequence Diagram (Sign Up)

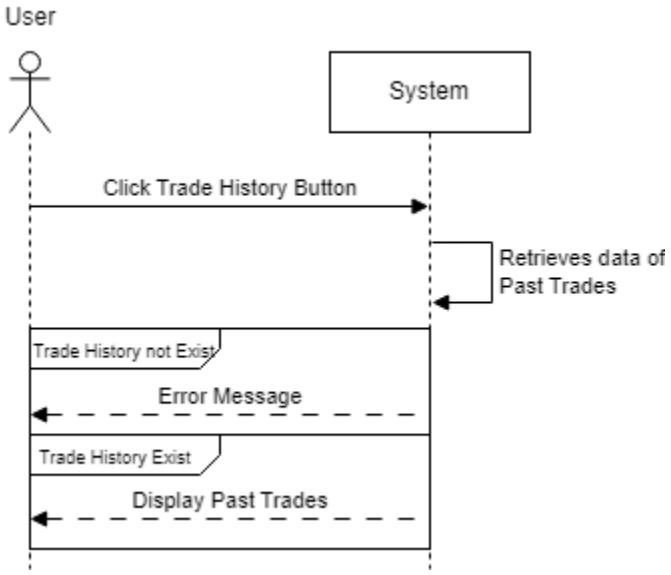

Figure 75: Sequence Diagram (Trade History)

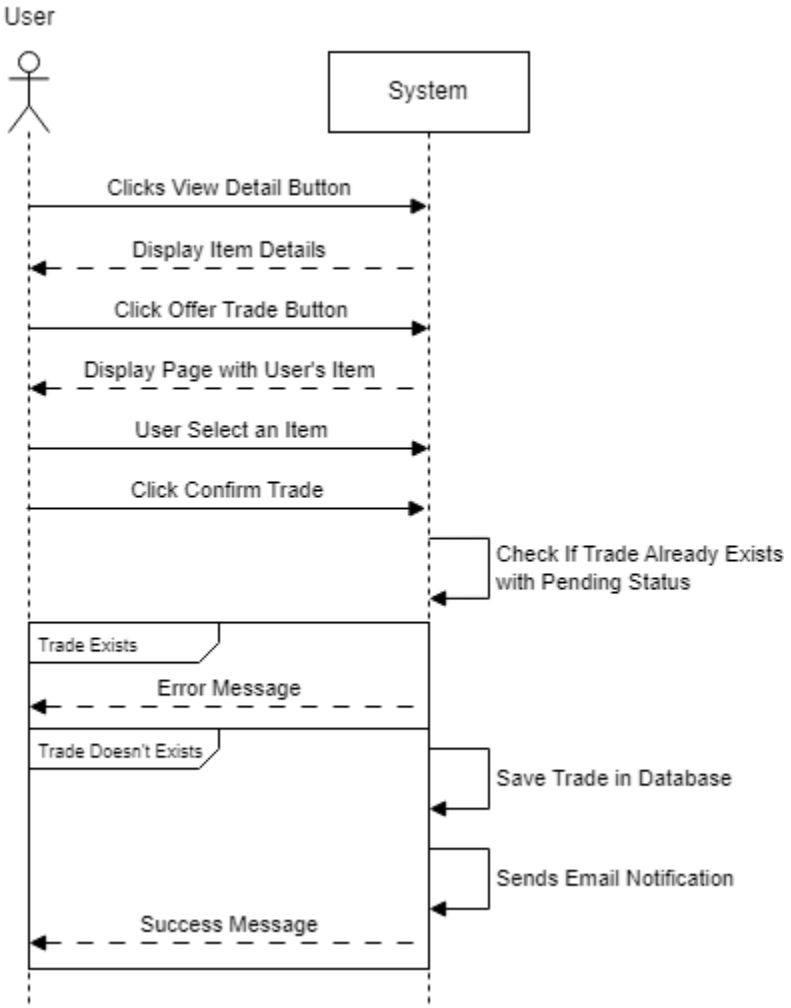

Figure 76: Sequence Diagram (Trade Proposal)

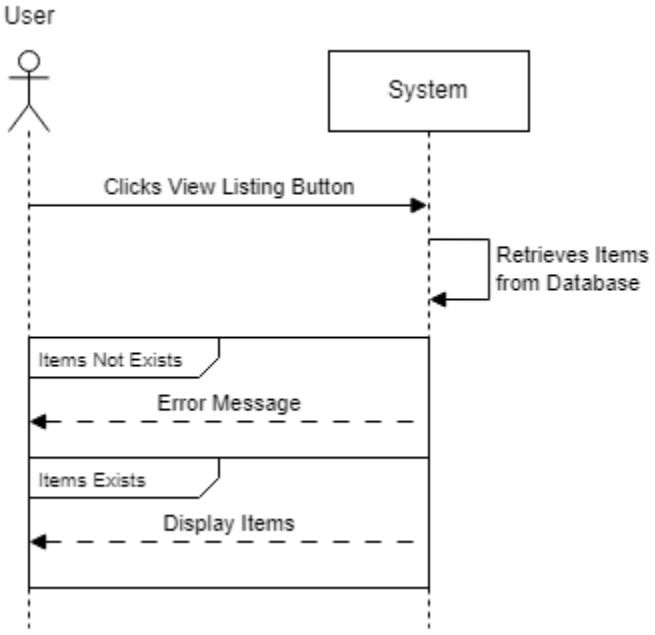

Figure 77: Sequence Diagram (View Listing)