Modular Software Development

In-20-CS2833

Vikkramanayaka A.G.P.S.	200683X
Heshan K.B.S	200215R
Salwathura P.C.	200553B
Rahal S. A. V.	200493N
Arachchi A.C.H	200044P

Table of Contents

1.0. Introduction

1.1 Purpose

The purpose of the system is to give users a platform to publicly advertise the sale of their assets, such as real estate, homes, equipment, electronics, vehicles, fashions and other things, online. At the same time, it makes it simple for buyers to find and buy these assets. By offering a user-friendly interface, a cash-on-delivery payment option, and a rating system to assess both buyers and sellers, the system aims to make the buying and selling process easier.

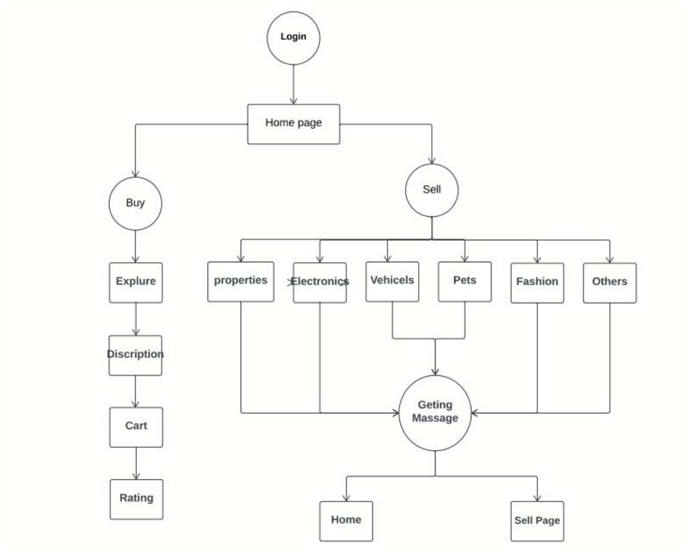
1.2 System Overview

The system functions as a central marketplace where buyers and sellers can both showcase their available listings. Sellers will be able to create thorough listings for their assets, complete with descriptions, pictures, and prices. On the other hand, customers can look through various categories or conduct a specific item search to find what they want.

The system accepts cash on delivery as a payment option to ensure a safe and reliable transaction process. This implies that customers can make purchases without having to use online payment methods and can instead pay in cash when the item is delivered. The system also features a rating system that enables users to comment on and rate their interactions with buyers and sellers. With the aid of this rating system, users can transact with confidence and make wise decisions.

2.0. Overall Description

2.1. System Environment



3.0. System Requirements Specifications

3.1. Functional Requirements

1) Registration of Users:
    - Users should have the ability to set up accounts with distinctive usernames and passwords.
    - The system must securely validate and store user registration data.
    - To activate their accounts, users might need to validate their email addresses.

2) Authenticating users:
    - Users should be able to access the system with the credentials they registered with.
    - The system should verify user credentials and give authorized users access.
    - Passwords need to be transmitted and stored securely.

3) Asset Browse and Search:
    - Buyers' ought to be able to conduct asset searches using a variety of filters, including location, category, price range, and keywords.
    - In order to hone search results, the system should offer advanced search options and filters.
    - Buyers' ought to be able to easily browse through the listings.

4) Asset Listing:
    - Sellers must be able to create listings for their assets that include all necessary information, including the title, description, images, price, location, and category.
    - The listing information should be verified by the system to ensure its accuracy and completeness.
    - Sellers ought to be able to modify, update, and remove their listings.

5) Cash on Delivery Payment:
    - Cash on delivery should be an option for customers to choose during the checkout process.
    - Both the buyer and the seller should receive a confirmation of the purchase from the system.
    - When a purchase is confirmed, sellers should be informed and given the details of the transaction.

6) Rating and Feedback:
    - Buyers and sellers should be able to rate and comment on their experiences through the system.
    - Users should have the option to post reviews and ratings for particular transactions.
    - Based on their performance, the system should determine and show overall ratings for sellers.

7) User Management:
    - Users should have access to view and modify their account data, including contact and profile information.
    - A dashboard or control panel should be available to sellers so they can manage their listings and keep tabs on their sales.
    - Customers ought to be able to look at their past purchases and follow the progress of active transactions.

8) Notifications:
    - Users should receive notifications from the system when significant events like new listings, buyer inquiries, transaction updates, and feedback are happening.
    - Depending on the user's preferences, notifications can be delivered via email or in-app notifications.

9) Administration of the system:
    - To manage user accounts, listings, and system operations generally, the system should have administrative features.
    - Administrators should be able to keep an eye on and control user behavior, including flagging offensive content or settling arguments.

3.2. Non-Functional Requirements

1) Performance:
    - To guarantee a seamless and responsive user experience, the system should have quick response times.
    - The platform should be able to support numerous concurrent users without noticeably degrading its performance.
    - Even with many listings, the search function ought to deliver prompt and precise results.

2) Security:
    - Sensitive information should be protected using encryption protocols during user authentication and data transmission.
    - In order to prevent unauthorized access, the system should put in place safeguards like strict password guidelines and account lockouts after numerous failed login attempts.
    - User data should be securely stored and guarded against unauthorized access, including personal data and transaction specifics.

3) Compatibility:
    - In order to provide users with a seamless experience across various platforms, the system should be compatible with a variety of hardware and web browsers.
    - The ability of users to access and interact with the system efficiently on mobile devices should be taken into account.

4) Localization:
    - Users should be able to choose their preferred language for the user interface, and the system should support multiple languages.
    - To accommodate users from various regions, localization of date and currency formats should be taken into consideration.

5) Privacy:
    - Data protection and privacy laws should be followed by the system to guarantee that user information is handled legally.
    - Users should be able to manage the visibility of their personal information and their privacy settings.

6) Reliability:
    - The platform should be accessible for users to access and carry out their desired actions, and the system should be extremely reliable, minimizing downtime.
    - Data loss or system failures should be prevented by putting precautions in place like data backups and disaster recovery plans.

7) Usability:
    - The user interface needs to be simple to use, appealing to the eye, and intuitive.
    - Users should be able to easily create listings, conduct asset searches, and complete transactions with the help of the system's clear instructions and guidance.
    - To ensure that people with disabilities can use the platform, accessibility issues should be taken into account.

8) Scalability:
    - The system ought to be built to support future expansion and increased user activity.
    - Without sacrificing performance, it should be scalable to accommodate an increase in the number of listings, users, and transactions.

3.3. User Roles

Sellers and buyers are the two main types of users, each with a specific role. Each type of user in the system has particular rights and obligations. The system's identified user roles are listed below.

1) Sellers:
    - Detailed information about the assets, such as descriptions, pictures, prices, and contact details, can be provided by the sellers.
    - Whenever necessary, they can modify or remove their listings.
    - In order to monitor their sales, control their inventory, and view transaction history, sellers may also have access to a dashboard or control panel.
    - Sellers are people or businesses who want to use the platform to sell their assets.
    - The assets they want to sell must have listings created and maintained by them.

2) Buyers:
    - Buyers are people or businesses looking to buy the assets listed on the platform.
    - They can look through the listings that are currently available, perform a search for particular assets, and view comprehensive asset details.
    - By choosing an asset and starting the checkout process, buyers can start the buying process.
    - They can choose to pay for their purchases with cash on delivery.
    - After a transaction is complete, buyers can rate and comment on their interactions with sellers.
    - Additionally, they might have access to a dashboard or account page where they can view their past purchases, keep tabs on current transactions, and manage their personal data.

A user may hold both the seller and buyer roles, enabling them to conduct business on the platform as both a seller and a buyer. All participants will have a simple and secure asset selling experience thanks to the user roles' assistance in defining the permissions and actions that are accessible to various users.

3.4. High-Level System Architecture

The asset selling platform's system architecture is made up of a number of essential parts that cooperate to deliver the required functionality and guarantee a positive user experience. Here is a high-level breakdown of the system's parts and how they work together:

1) User Interface (UI):
   The user interface component presents system features and enables user interaction, including registration, login, browsing, creating, and purchasing. It communicates with other components for data retrieval and input capture.

2) Data Layer:
   The data layer stores, manages, and retrieves system data, typically using a DBMS, and is accessed by the application layer for user requests.

3) Application Layer:
   The application layer manages business logic, user requests, and interacts with the data layer for data fetch and updating.

4) External Services and APIs:
   The system can integrate external services and APIs to improve functionality and provide additional features, such as email providers, geolocation, and payment gateways, enabling seamless communication and data exchange.

4.0. High-Level Design with Use Cases

4.1. System Overview

The asset selling system is a web-based platform that makes it easier to buy and sell different types of assets. The system is designed using a client-server architecture, where clients are users who access the platform using a web browser or a mobile application, and servers are in charge of processing user requests, handling data, and overseeing system functionality.

There are some key components:

1) Client-Side:
   The client-side component is the user interface (UI), which users interact with through web pages or mobile apps. It provides features like registration, login, asset browsing, listing creation, and transaction management, capturing user input and forwarding requests to the server for processing.

2) Server-Side:

The web server handles client-side requests and communication, while the application layer manages business logic and processing functionality. It handles user authentication, asset listing, search, transaction processing, and rating management. The database stores and manages the system's data, including user information, asset listings, transactions, and ratings, providing persistent storage and efficient retrieval and manipulation.

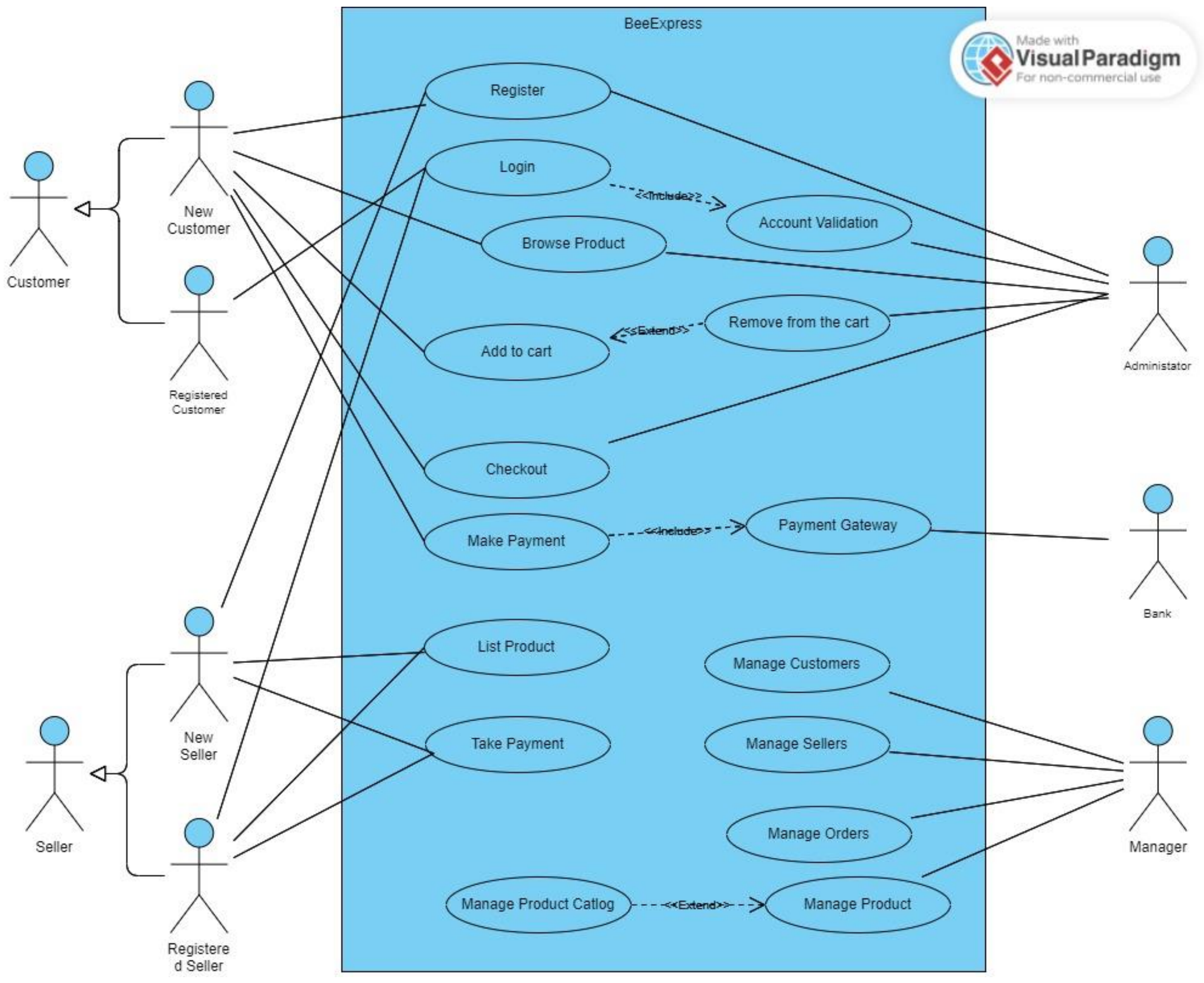
3) External Services:

The system can integrate with email service providers for verification emails, geolocation services for asset location, and a payment gateway for cash on delivery payments processing and verification.

Interactions and Flow:

- On the client-side, users interact with the user interface, using it to access different features and enter data.
- Requests from users are sent to the web server, which forwards them to the appropriate application layer components.
- When necessary, the application layer retrieves or updates data from the database to handle user requests.
- In order to complete certain functionalities, the application layer may communicate with external services like email service providers or payment gateways.
- Users see the updated data on the client side after the server-side components produce and send back responses.
- Through the user interface, users can carry on interacting with the system by making new requests or taking additional actions.

4.2. Use Case Diagram



4.3. Detailed Use Case Description

- Use Case: Register

  Description: The customer or Seller registers a new account on the BeeExpress website.

- Use Case: Login

  Description: The customer or Seller logs into their account on the website.

- Use Case: Browse Products

  Description: The customer can search and view products available on the BeeExpress.

- Use Case: Add to Cart

  Description: The customer adds a product to their shopping cart.

- Use Case: Remove from Cart

  Description: The customer removes a product from their shopping cart.

- Use Case: Checkout

  Description: The customer proceeds to the checkout process to finalize their purchase.

- Use Case: Make Payment

  Description: The customer provides payment details, and the payment gateway processes the payment. Bank transfer the funds.

.

- Use Case: Manage Product

  Description: The manager can add, edit, or delete products from the website's catalog.

- Use Case: Manage Orders

  Description: The manager can view, update, or cancel orders placed by customers.

- Use Case: Manage Customers

  Description: The manager can manage customer accounts, including updating information and handling customer inquiries.

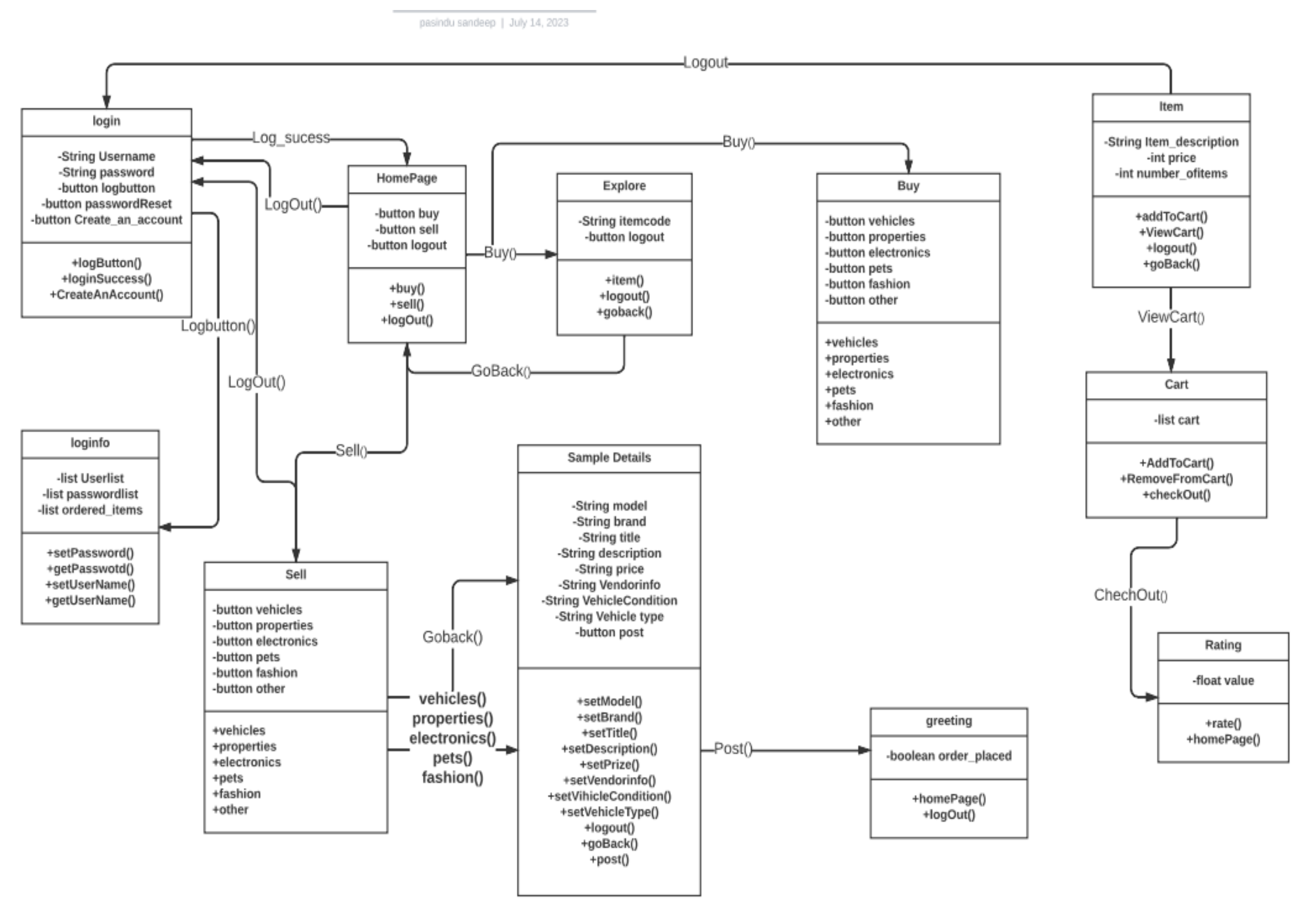
- Use Case: Manage Sellers

  Description: The manager can manage seller accounts

- Use Case: Take Payments

  Description: The seller receives payment for the products sold, and the payment gateway processes the transaction.

4.4. Class Diagram



5.0. Key User Interface

5.1. User Interface Overview:

The asset selling system's user interface (UI) is essential to giving users a seamless and understandable experience. To ensure usability, visual appeal, and effective interaction, the UI design should abide by specific principles and guidelines. Here is a summary of the general design principles and recommendations for the asset selling system's user interfaces:

1) Simplicity and Clarity:
   UI design should be clean, uncluttered, and use concise language for labels, instructions, and error messages. Provide intuitive navigation and content organization for user convenience.

2) Responsive Design:
   Create a responsive UI that adapts to various screen sizes and devices, ensuring appropriate sizing, spacing, and scalability for optimal usability.

3) Visual Hierarchy and Readability:
   Create a clear visual hierarchy, prioritize important elements, use appropriate fonts, colors, and formatting for readability and legibility.

4) Consistency:
   Consistency in visual elements, terminology, and UI elements is crucial for a smooth user experience. Use consistent labels and placement to avoid confusion and ensure a seamless user experience across various screens and components.

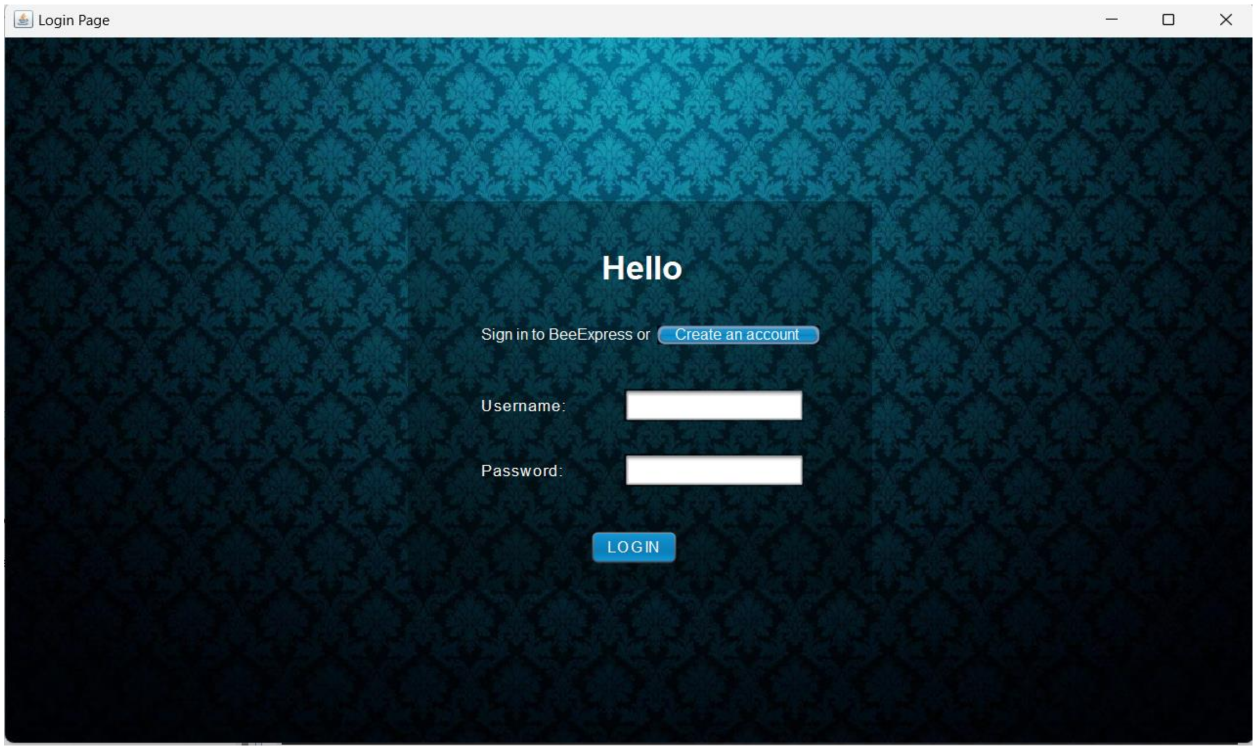
5) User-Friendly:
   Design UI for user-friendliness, ease of use, and familiar patterns to reduce learning curves and provide clear, contextual cues for actions and processes.

6) Feedback and Error Handling:
   Offer users immediate feedback on actions, including button clicks and form submissions, and display error messages and validation to help resolve issues.

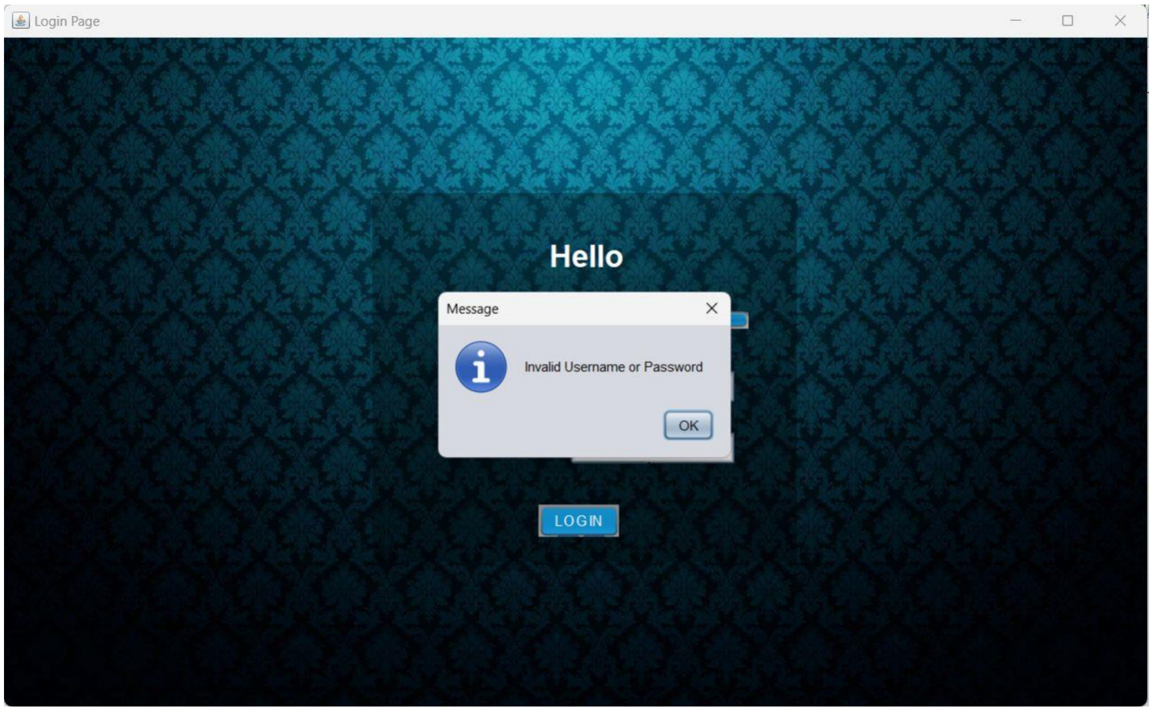
7) Accessibility:
   Adhere to accessibility guidelines for user-friendly interfaces, including color contrast, alternative text, and keyboard navigation

5.2. Mockups and Wireframes:

The User, after clicking on sign in or register button, this window will appear.



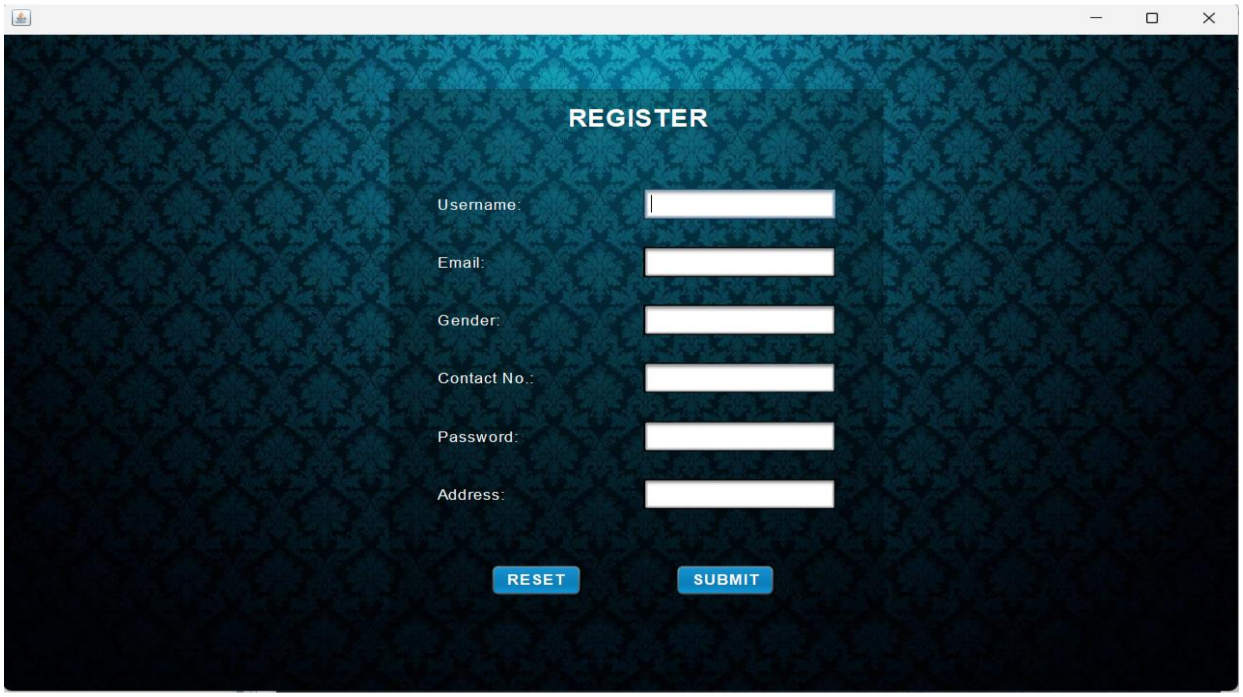
If user has an account, he/she can log in to his/her account. Otherwise, he/she have to create an account through the Create an account button.



Invalid Username or Password

If the username or password entered by the user is incorrect this dialogue will appear.

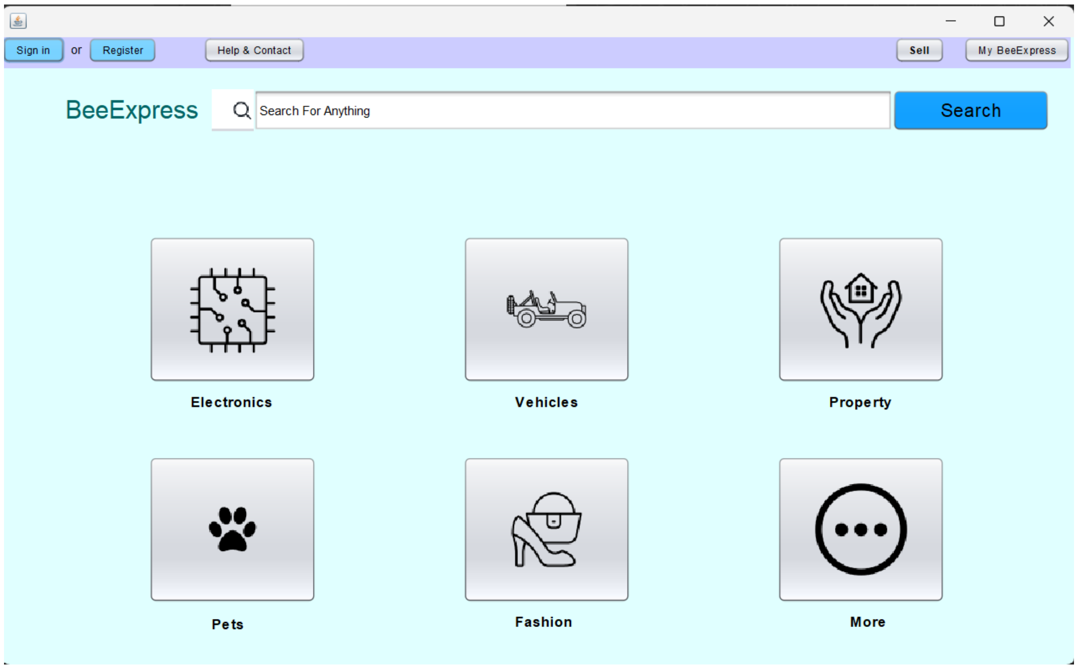
After clicking the create an account button, this page will appear.



After that, he/she can create an account by giving those details and clicking the submit button.

After clicking submit button, the Hello page will appear again. And can be logged with username and password.

After logged in, this home page will appear.

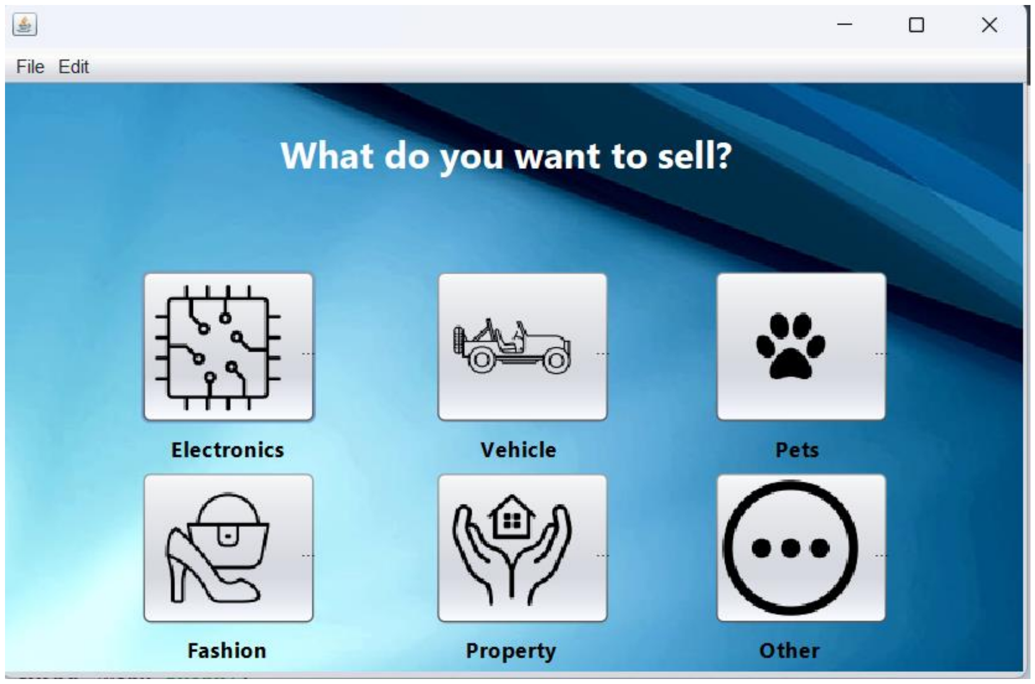


Our online marketplace's home page offers a user-friendly and simple interface that meets the demands of consumers and merchants alike. For users wishing to conduct various transactions, the page is intended to provide a quick and easy experience.

Users are presented with clear options to choose between purchasing and selling as soon as they land on the main page. Users can save time and effort by making this initial selection, which enables them to quickly browse to the part that corresponds with their intentions. The alternatives are clearly visible, providing accessibility.

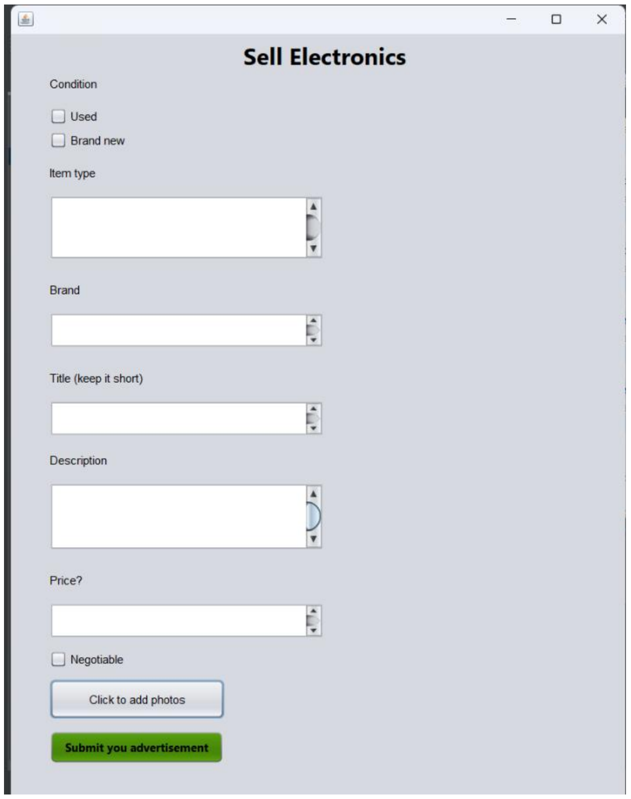
The home page offers a thorough platform for product browsing and selection for individuals who are interested in purchasing. Users may rapidly find information by using the search bar, which is prominently displayed at the top of the page.

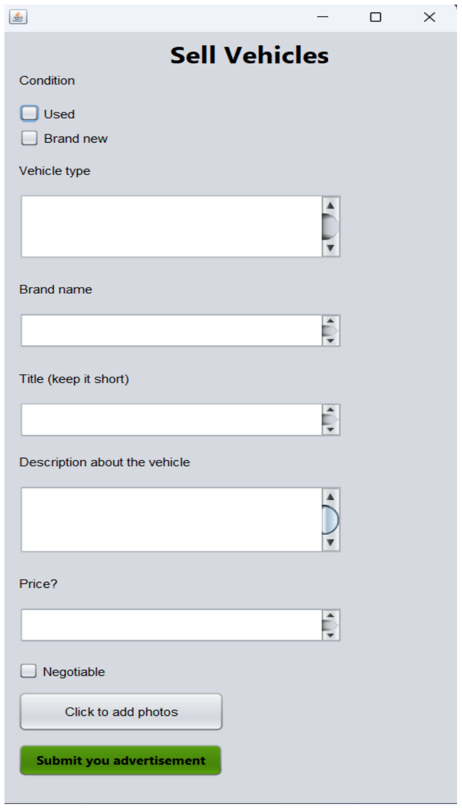
After that if the user wants to sell something, This page will appear



On this page, sellers have the option to choose the category for the item they want to sell.

Upon selecting the "Electronics" category, a form will dynamically appear, allowing the seller to provide specific details about the electronics products they are selling. After clicking submit they can publish their advertisement.





5.3. User Interaction

Through the provided user interfaces, users will engage in various interactions with the asset selling system, performing tasks like navigation, data input, and feedback. The following describes how users will engage with the system.

1) Feedback and Rating:
   Users can provide feedback and ratings for transactions and overall experiences within the system. Both buyers and sellers can leave comments on transaction quality. The system collects and displays user feedback, building trust and reputation among users.

2) Data Input:
   Users input data and information through forms and fields on the user interface. Sellers input assets like title, description, price, location, and images, while buyers use search criteria and delivery address details. Sellers create listings, while buyers search for relevant assets.

4) Navigation:
   The system's user interface offers clear, intuitive navigation options for users to access various functionalities, including registration, login, asset browsing, listing creation, and transaction management.

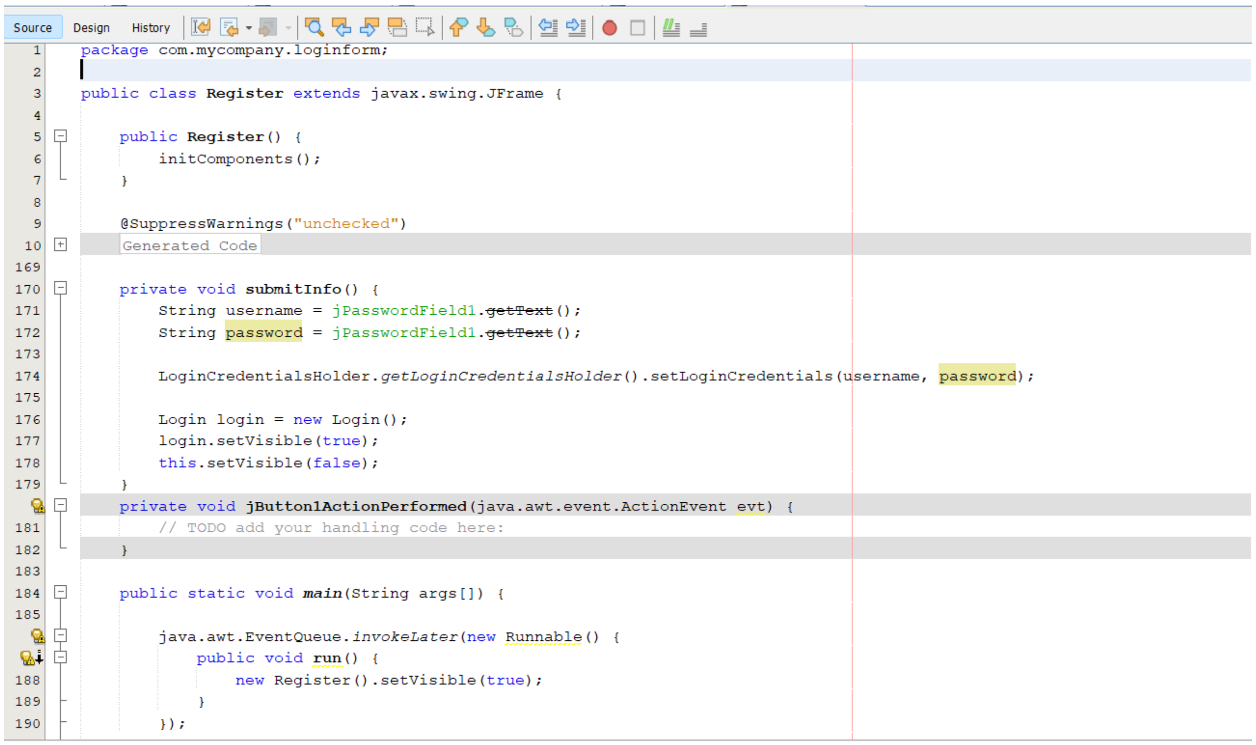
5) Notifications:
   The system sends users notifications for important activities and updates, including new messages, buyer inquiries, purchase confirmations, and feedback. These notifications can be delivered through in-app, email, or dedicated centers, and can be accessed directly in users' inboxes.

6) Search and Filtering:
   Buyers can search for assets using keywords, location, category, and price range. The system offers search and filtering options, allowing users to refine results and find specific requirements. Users can interact with filters, select options, and enter queries. Results are displayed user-friendly, allowing browsing and accessing detailed information.

Users will be able to use these interactions to navigate the system, enter relevant information for various activities, give feedback, and receive notifications. Users will be able to interact with the system easily and effectively thanks to the user interfaces, which are created to facilitate a seamless and intuitive user experience.

Appendix

Login.java



```
package com.mycompany.loginform;

public class Register extends javax.swing.JFrame {

    public Register() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void submitInfo() {
        String username = jPasswordField1.~~getText~~();
        String password = jPasswordField1.~~getText~~();

        LoginCredentialsHolder.getLoginCredentialsHolder().setLoginCredentials(username, password);

        Login login = new Login();
        login.setVisible(true);
        this.setVisible(false);
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Register().setVisible(true);
            }
        });
```

Regiter.java

```
package com.mycompany.loginform;


import javax.swing.*;
import java.util.Map;

public class Login extends javax.swing.JFrame {

    private static Map<String, String> map = null;

    public Login() {
        initComponents();
        map = LoginCredentialsHolder.getLoginCredentialsHolder().getLoginCredentials();
    }
    private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
        Register reg = new Register();
        reg.setVisible(true);
        this.setVisible(false);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Login().setVisible(true);
            }
        });
    }
```

<u>LoginCredentialHolder.java</u>

```
package com.mycompany.loginform;



import java.util.HashMap;
import java.util.Map;

/**
 * Singleton In-Memory LoginCredentialsHolder
 */
public class LoginCredentialsHolder {

    private static LoginCredentialsHolder loginCredentialsHolder = null;

    private Map<String, String> loginCredentials = new HashMap<>();

    public LoginCredentialsHolder () {
        loginCredentials.put("Dulanga", "Dula");
        loginCredentials.put("amarabandu", "Rupasinghe");
    }

    public static LoginCredentialsHolder getLoginCredentialsHolder() {
        return loginCredentialsHolder == null ? loginCredentialsHolder = new LoginCredentialsHolder() : loginCredentialsHolder;
    }

    public Map<String, String> getLoginCredentials () {
        return loginCredentials;
    }

    public void setLoginCredentials (String username, String password) {
        loginCredentials.put(username, password);
    }


}
```

** End **