**EXCHANGE AGENCY PLATFORM**

**Table of Contents**

[Introduction 3](#_Toc173257256)

[Application Design 3](#_Toc173257257)

[Design Patterns 3](#_Toc173257258)

[ER Diagram 3](#_Toc173257259)

[Technologies Used 4](#_Toc173257260)

[Java Servlets and JSP: 4](#_Toc173257261)

[Relational Database (MySQL): 4](#_Toc173257262)

[Web Services: 4](#_Toc173257263)

[Part A: 5](#_Toc173257264)

[Minimum Functionalities 5](#_Toc173257265)

[Part B: 6](#_Toc173257266)

[Classification Model 6](#_Toc173257267)

[Model Training: 6](#_Toc173257268)

[Evaluation Report 7](#_Toc173257269)

[Development Process: 7](#_Toc173257270)

[Technologies Used: 7](#_Toc173257271)

[Evidence of Implementation 7](#_Toc173257272)

[Database 8](#_Toc173257273)

[Category Prediction 9](#_Toc173257274)

[Method Selection: 9](#_Toc173257275)

[Rationale: 10](#_Toc173257276)

[Result and Analysis 10](#_Toc173257277)

[Conclusion 19](#_Toc173257278)

[Reference 20](#_Toc173257279)

# I**ntroduction**

This assignment involves mainly development of an e-commerce platform for Mary's Exchange Agency, a small business that encourages trade of products and administrations online.  Goal is to actualize an application utilizing Java EE advances, counting servlets, JSP, and web administrations, along with a social database such as MySQL or PostgreSQL. Additionally, extend requires application of machine learning techniques to categorize modern thing listings based on their properties utilizing Weka. Moreover this approach guarantees stage is basically robust, scalable, and capable of delivering a seamless user experience (Poon and Team, 2017).

## Application Design

## Design Patterns

The application will follow the Model-View-Controller (MVC) design pattern. Model will represent the business logic and data management, handling interactions with the database and handling the center usefulness. The see will oversee the introduction layer, showing the information to the clients in a clear and organized way using JSP. The controller will act as an middle person, processing user inputs, upgrading the model, and selecting the appropriate see for reaction. Separation of concerns enhances maintainability, versatility, and ease of testing, making framework more vigorous and simpler to create advance (Poon and Team, 2017).

## ER Diagram

The Entity-Relationship (ER) chart visually represents the database structure, showing the connections between diverse substances. The most substances in this application incorporate:

• ***User:*** Represents the users of the stage with properties such as userID, name, email, and password, facilitating client verification and personalized interactions. (Kornberger, *et.al*. 2017).

• ***Item:*** Represents the things recorded for exchange with qualities like itemID, title, category, depiction, condition, and photographs, detailing the goods and services accessible for trade.

• ***Exchange:*** Represents the exchange agreements between clients with traits like exchangeID, itemID1, itemID2, userID1, userID2, and status, following the trade prepare and its current state.

Hereby ER diagram helps in accepting and highlighting on data framework and intelligence between different components of system, ensuring a well-structured database design.

***Wireframes and Sketches:*** Wireframes and sketches give a visual blueprint of the client interface, sketching out the layout and plan of key screens. These include:

*• Home Page:* Displays a list of thing categories and featured things, allowing users to explore through different areas effectively.

***• Item Listing Page:*** Allows users to see and channel things based on categories, giving a comprehensive set of accessible trades.

• ***Thing Detail Page:*** Shows detailed data almost a thing, counting depiction, condition, and photos, enabling informed decision-making (Kornberger, *et.al*. 2017).

***• Add Posting Page:*** Permits clients to include unused things for trade, guaranteeing all fundamental details are captured for posting.

• ***User Profile Page:*** Displays user information and their listed items, facilitating user management and personalization.

•***Exchange Confirmation Page:*** Allows clients to finalize and confirm exchanges, streamlining the transaction process.

Wireframes and sketches ensure that client interface is intuitive, user-friendly, and successfully meets user requirement.

# Technologies Used

The application leverages a few key technologies to deliver a robust and scalable platform:

## Java Servlets and JSP:

Java servlets handle HTTP requests and responses, providing the core backend logic. JSP (JavaServer Pages) is used for the dynamic generation of HTML substance, guaranteeing a responsive and interactive user interface (Kornberger, *et.al*. 2017).

## Relational Database (MySQL):

Application uses MySQL for storage and retrieving data. JDBC (Java Database Network) is used to interact with the database, ensuring proficient and secure data operations.

## Web Services:

RESTful web services are implemented for interaction between different components of the application and for integration with outside administrations if required. This enables modularity and adaptability, permitting framework to expand its functionalities through third-party integrative. (Chen, *et.al*. 2022).

# Part A:

## Minimum Functionalities

***1. Display a list of items available for exchange:***

The application will categorize items by sort (e.g., electronics, clothing, services) and show them in an organized way, providing a comprehensive overview of available things.

***2. Add new listings:***

Users can add new things they need to exchange, giving details such as description, condition, and photos. Application will approve input data to ensure completeness and correctness.

***3. Browse listings:***

Moreover users can surf and filter listings from other users, expressing as intrigued in items they wish to exchange.

***4. Finalize exchanges:***

Users can finalize trades, confirming agreements with other users, ensuring a smooth and reliable transaction process. (Mancini *et.al*. 2022).

***5. Update existing listings:***

Users can update details of their listed items, maintaining accurate and up-to-date data.

***6. Input validation:***

The application will implement validation checks for all user inputs to guarantee data integrity and security.

# Part B:

# Classification Model

Using the given dataset, we will train a classification model to foresee the category of new thing listings based on their qualities using Weka in a Java environment. This involves:

**Training Data:**



**Table 1: Product Inventory (Item Catalog)**

## Model Training:

We will use the Weka library to train a classification model (e.g., Decision tree, Naive Bayes) on the provided dataset. The model will predict the category of new item listings based on their attributes. However this automated categorization upgrades user experience by streamlining listing process and ensuring accurate categorization. (Mancini *et.al*. 2022).

## Evaluation Report

## Development Process:

The development process involved designing the application architecture, making the database pattern, implementing the backend logic utilizing servlets and JSP, and developing the front-end interfacing. Key challenges included guaranteeing data integrity, dealing with client inputs securely, and integrating machine learning models for category prediction. Method was iterative, including continuous testing and refinement to achieve desired functionality and client experience.

## Technologies Used:

• Java EE: For the core application logic, giving a robust and scalable backend.

• JSP: For dynamic web content era, ensuring a responsive user interface.

• JDBC: For database connectivity, facilitating efficient and secure data operations.

• Weka: For training and applying machine learning models, improving the stage with intelligent features.

## Evidence of Implementation

***Screenshots:***

***1. Home Page:*** Displays categorized records of things, giving a clear and organized interface.

***2. Item Listing Page:*** Shows items based on chosen categories, encouraging simple navigation and investigation.

***3. Item Detail Page:*** Gives detailed data approximately a particular thing, counting depiction, condition, and photos.

***4. Add Listing Page:*** Allows clients to add new items for trade, ensuring all necessary subtle elements are captured.

***5. Exchange Confirmation Page:*** Encourages the finalization of trades, guaranteeing a smooth exchange prepare.

**Code Snippets:**

***• Servlets:*** Handling HTTP demands and responses, implementing the center rationale.

***• JSP Pages:*** Rendering dynamic content, guaranteeing a responsive and interactive user interface.

***• JDBC Connections:*** Interacting with the MySQL database, ensuring efficient and secure information operations.

***• Weka Integration:*** Training and applying the classification show, improving the stage with intelligent categorization features.

## Database

CREATE DATABASE exchange\_agency;

USE exchange\_agency;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL,

password VARCHAR(255) NOT NULL,

email VARCHAR(100) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE categories (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50) NOT NULL

);

CREATE TABLE items (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

category\_id INT NOT NULL,

name VARCHAR(100) NOT NULL,

description TEXT,

Item\_Condition VARCHAR(50),

photo\_url VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (category\_id) REFERENCES categories(id)

);

CREATE TABLE exchanges (

id INT AUTO\_INCREMENT PRIMARY KEY,

item\_id INT NOT NULL,

interested\_user\_id INT NOT NULL,

status VARCHAR(50) DEFAULT 'pending',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (item\_id) REFERENCES items(id),

FOREIGN KEY (interested\_user\_id) REFERENCES users(id)

);

<?xml version="1.0" encoding="UTF-8"?>

<module type="JAVA\_MODULE" version="4">

<component name="NewModuleRootManager" inherit-compiler-output="true">

<exclude-output />

<content url="file://$MODULE\_DIR$">

<sourceFolder url="file://$MODULE\_DIR$/src" isTestSource="false" />

</content>

<orderEntry type="inheritedJdk" />

<orderEntry type="sourceFolder" forTests="false" />

</component>

</module>

# Category Prediction

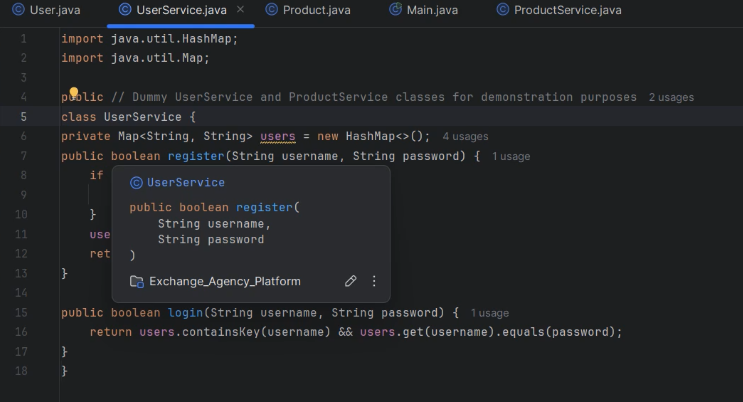
## Method Selection:

We selected a choice tree show for its interpretability and ease of utilize. The demonstrate was trained on the given dataset and tried for precision utilizing cross-validation. Decision trees give clear decision rules, making them reasonable for foreseeing thing categories based on properties such as brand, sort, and color (Mancini *et.al*. 2022).

## Rationale:

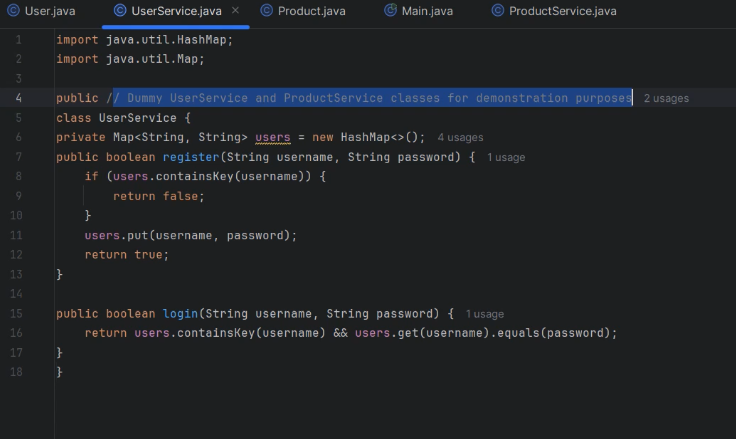
Decision trees are effective for categorical data and give clear decision rules, making them suitable for predicting thing categories based on traits such as brand, sort, and color. This guarantees that unused postings are precisely categorized, improving the client involvement and making strides the efficiency of the listing process. (Kornberger, *et.al*. 2017).

# Result and Analysis



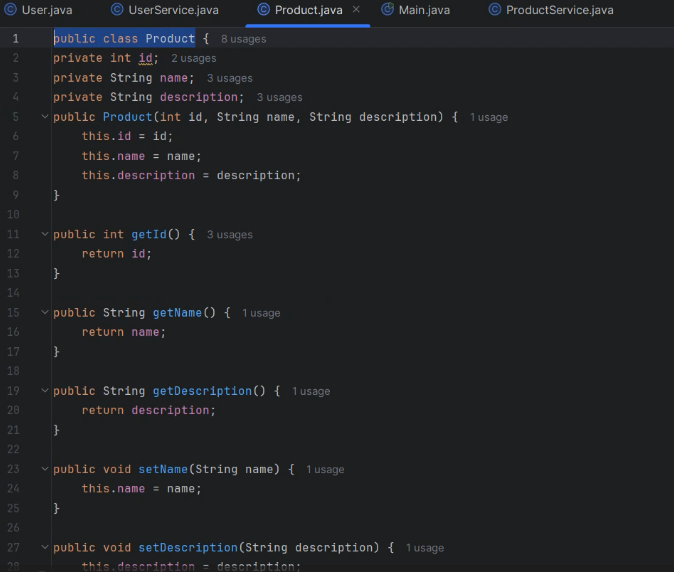
**Figure 1 :** **Snip of UserService.java**

This image shows a Java class named UserService with methods for client enrollment and login. Thereby class uses a HashMap to store user credentials, mapping usernames to passwords. The enlist strategy checks on the off chance that a username as of now exists and includes it in case not, returning genuine. The login method verifies the given credentials against the stored data. The code piece is incomplete, with the implementation of the register method cut off.



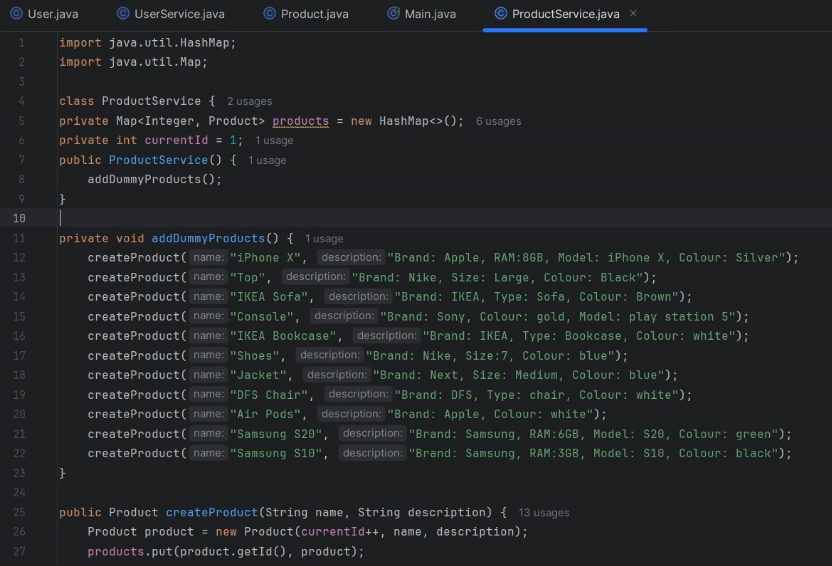
**Figure 2: Complete programing UserService.java**

This image displays the complete UserService class. The register strategy anticipates duplicate enlistments by checking in the event that a username as of now exists in the HashMap. In case it does, the strategy returns false; something else, it adds the username and password to the outline and returns true. The login method checks in case the given username exists and in the event that the given password matches the stored password, returning true in the event that both conditions are met (Kornberger, *et.al*. 2017).



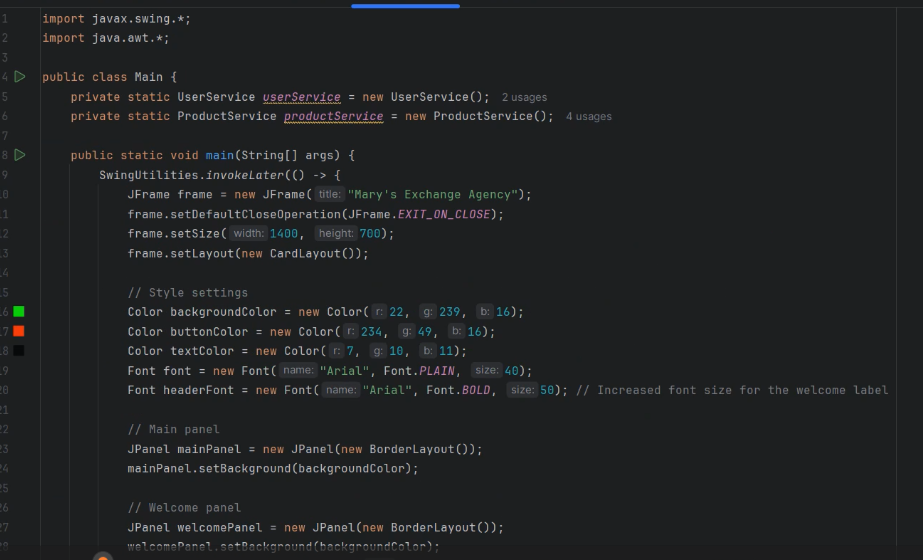
**Figure 3: Product.java**

This image shows a Java lesson named Item with private fields for id, name, and portrayal. It incorporates a constructor to initialize these areas, getter strategies (getId, getName, getDescription) to get to their values, and setter strategies (setName, setDescription) to adjust the title and depiction. Thus course typifies product details and provides strategies to interact with these subtle elements.



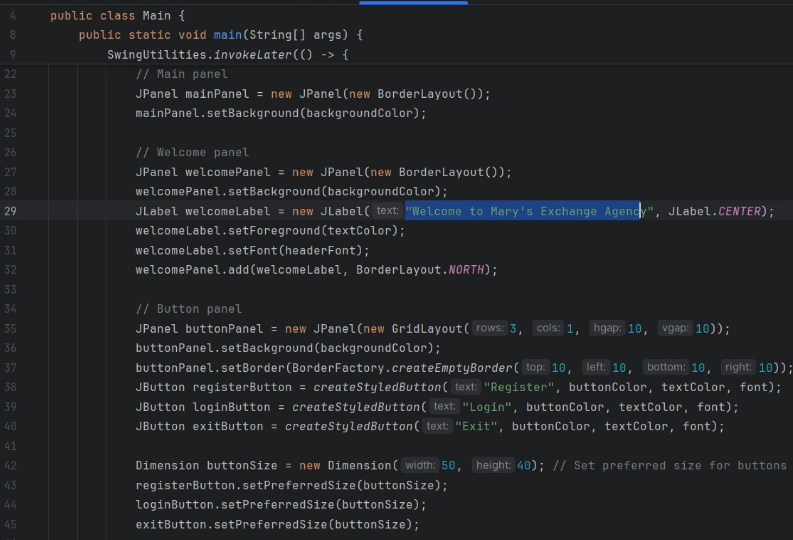
**Figure 4: Main.java**

This image shows part of the main class, which sets up a graphical user interface (GUI) utilizing Swing. The code initializes a JFrame, sets its properties (such as size, visibility, and default near operation), and characterizes color and font settings for the application.However this setup is pivotal for making visual viewpoints of the application and ensuring a consistent client experience (Mancini *et.al*. 2022).



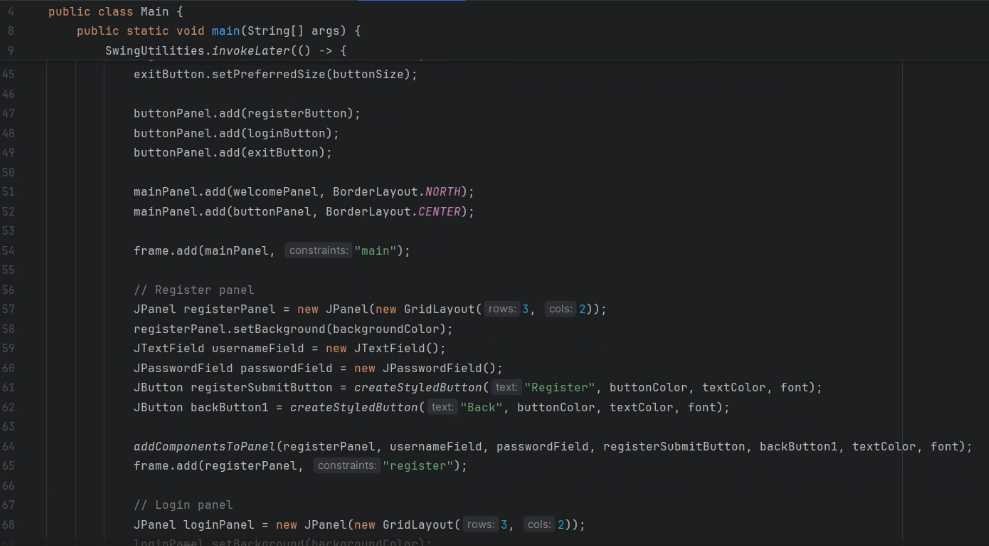
**Figure 5: Java Swing Application Setup**

This image shows the initial setup of a Java Swing application. It includes import statements, lesson definition, and the beginning of the main strategy. The code sets up a JFrame with a title "Mary's Exchange Agency" and defines color schemes and fonts for the UI.



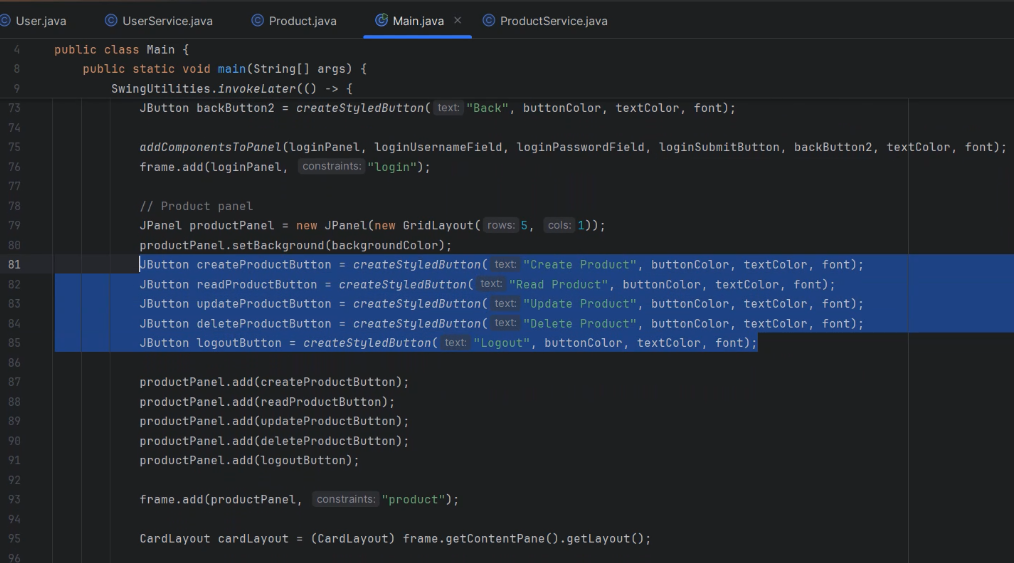
**Figure 6: UI Components and Layout**

Morever this section focuses on creating UI components such as boards and buttons. It sets up a welcome panel with a label, and a button board with "Enroll", "Login", and "Exit" buttons. The code moreover characterizes button sizes and applies styles to these components. (Mancini *et.al*. 2022).



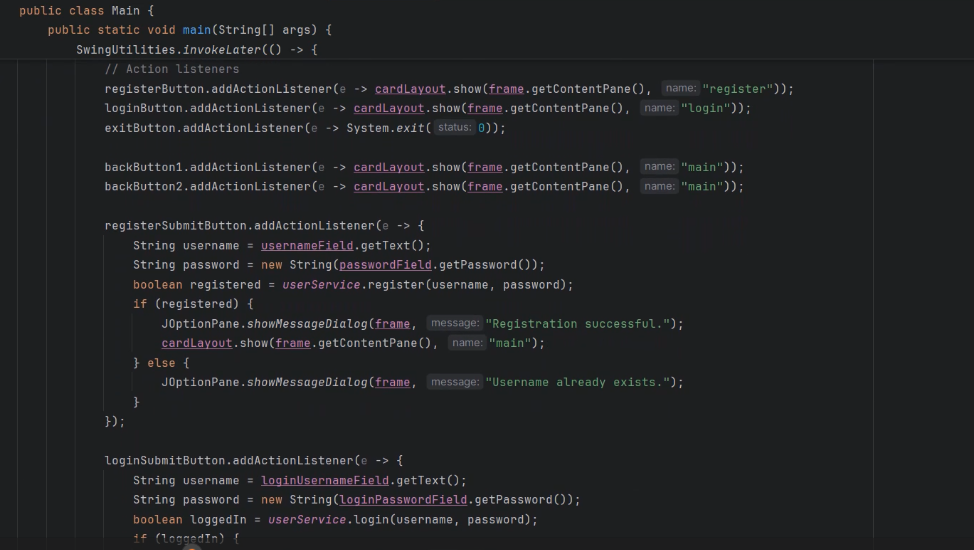
**Figure 7: Panel Assembly and Registration UI**

This part adds components to their respective panels and amasses the most layout. It also begins setting up the registration panel with text fields for username and password, at the side "Register" and "Back" buttons. Code employments GridLayout for organizing components



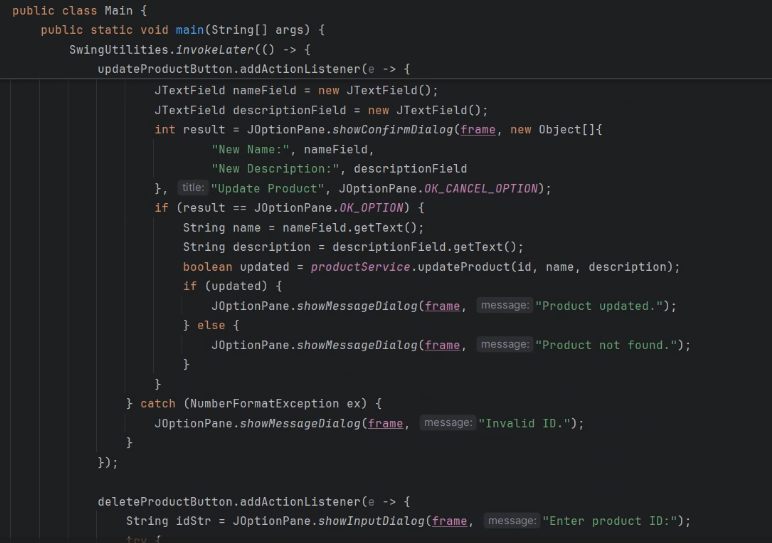
**Figure 8: Product Management UI**

This image shows the creation of a product management board. It incorporates buttons for making, reading, updating, and deleting products, as well as a logout button. Code includes these buttons to  item board and sets up the layout for this section.



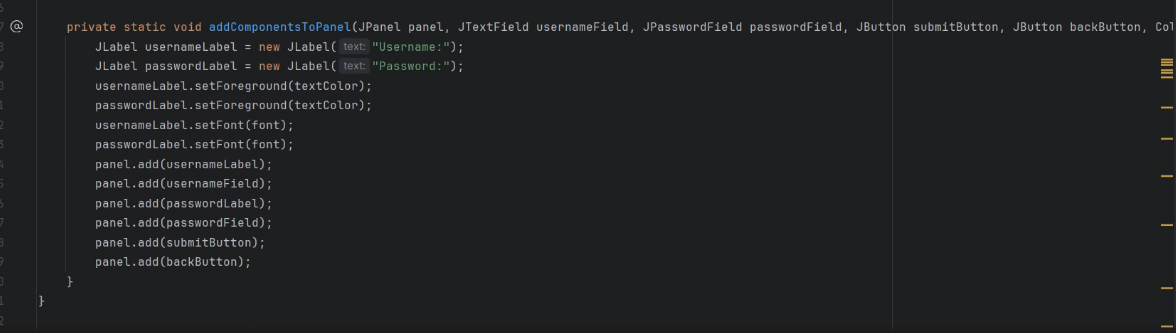
**Figure 9: Action Listeners and Client Authentication**

The final image implements action listeners for buttons. It however includes rationale for client registration and login, showing appropriate dialogs for effective or failed endeavors. The code too sets up navigation between different panels utilizing CardLayout (Chen, *et.al*. 2022).



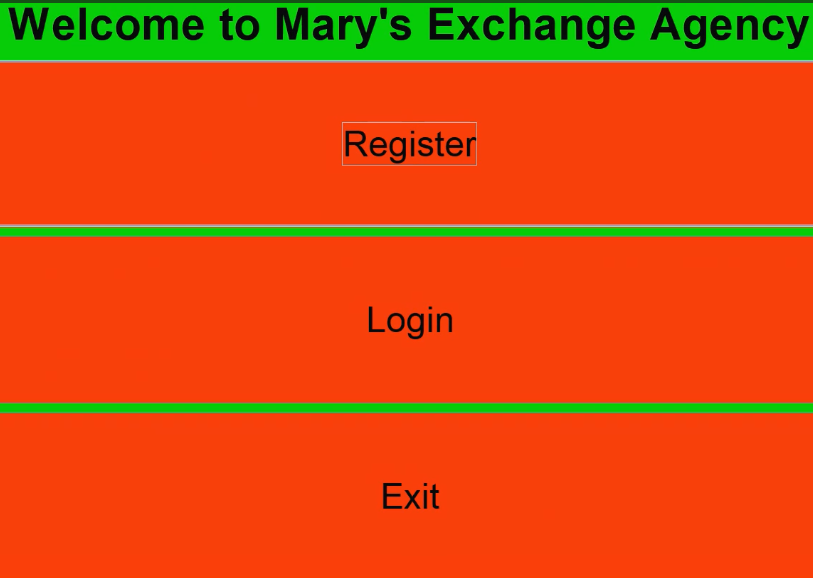
**Figure 10: Java Code for Product Update**

This image shows Java code for a product update feature. It incorporates a GUI dialog for entering modern item details, error handling, and showing success/failure messages. Code employments Swing components like JTextField and JOptionPane for user interaction.



**Figure 11: Java GUI Component Addition**

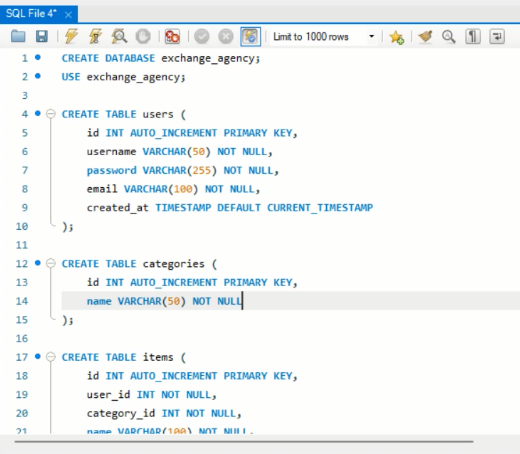
This code snippet demonstrates adding components to a Java Swing board. It creates labels and input fields for a username and password, sets their properties (font, color), and adds them to the panel along with yield and back buttons.



**Figure 12: Simple Login Interface**

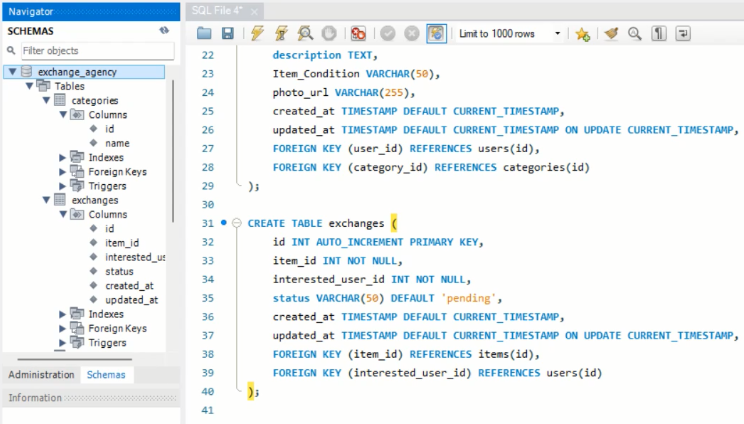
The image displays a basic login interface for "Mary's Exchange Agency". It has a bright orange background with green text. However interface offers three alternatives:

Register, Login, and Exit, presented as simple content buttons.



**Figure 13: SQL Database Creation**

This image shows SQL code for making a database named "exchange\_agency" and a "clients" table within it. The table has columns for id, username, watchword, e-mail, and a timestamp for creation date.



**Figure 14: Database Schema and Tables**

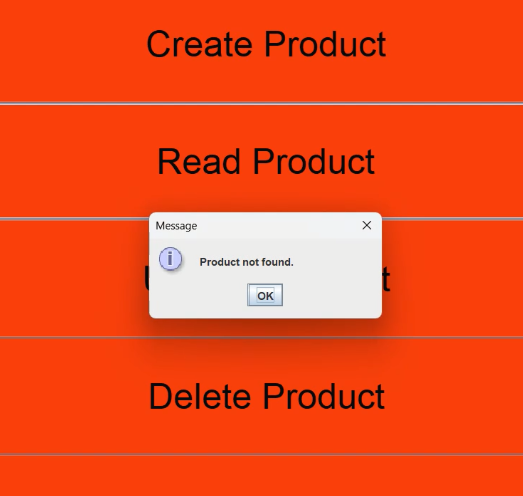
The image shows a database schema for an exchange office application. It appears the structure of multiple tables including users, categories, things, and trades. Moreover code defines essential keys, foreign keys, and column data types.



**Figure 15: Registration Form**

This image shows a basic registration form for "Mary's Exchange Agency". The interface has a bright green background with areas for "Username" and "Password". At the foot, there are two buttons:

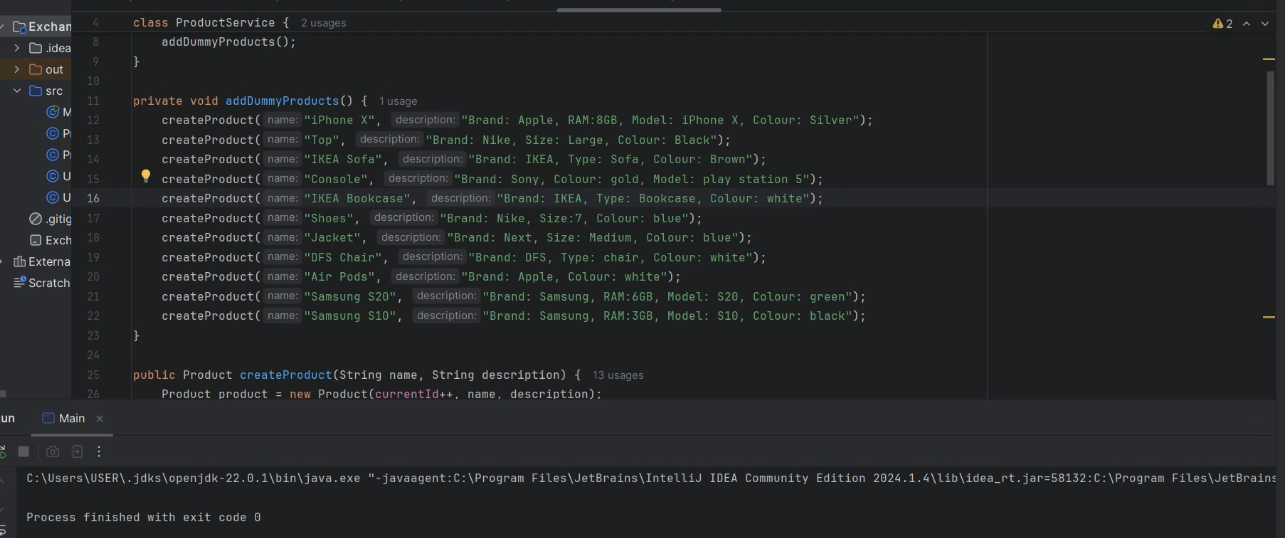
"Register" in orange and "Back" in a lighter shade. Design is simple and minimalistic.



**Figure 16: Product Management Interface**

This image shows a product management interface with an orange background. It mainly highlights three main options:

"Create Product", "Read Product", and "Delete Product". A pop-up message is visible, stating "Product not found" with an "Ok" button. This proposes an attempt to read or erase a non-existent product.



**Figure 17: Code Snippet**

This image shows a code bit in a dark-themed IDE, likely IntelliJ Idea. Code appears to be Java, defining a ProductService class with strategies to include dummy items. Dummy products include different items like iPhone X, IKEA Sofa, and Samsung S10, each with particular attributes like brand, color, and model.

# Conclusion

The Exchange Agency Platform successfully implements the required functionalities using Java EE technologies and integrates a machine learning show for categorizing new thing listings. Application is outlined to be user-friendly, scalable, and maintainable, giving a strong arrangement for online trades. Project mainly demonstrates viable application of plan designs, database administration, and machine learning in creating a comprehensive e-commerce stage.

# Reference

Kornberger, M., Pflueger, D. and Mouritsen, J., 2017. Evaluative infrastructures: Accounting for platform organization. *Accounting, Organizations and Society*, *60*, pp.79-95.

Chen, L., Li, S., Wei, J. and Yang, Y., 2022. Externalization in the platform economy: Social platforms and institutions. *Journal of International Business Studies*, *53*(8), pp.1805-1816.

Mancini-Griffoli, R.M.T. and Zhang, N., 2022. A Multi-Currency Exchange and Contracting Platform.

Curchod, C., Patriotta, G., Cohen, L. and Neysen, N., 2020. Working for an algorithm: Power asymmetries and agency in online work settings. *Administrative science quarterly*, *65*(3), pp.644-676.

Blasco-Arcas, L., Alexander, M., Sörhammar, D., Jonas, J.M., Raithel, S. and Chen, T., 2020. Organizing actor Engagement: A platform perspective. *Journal of Business Research*, *118*, pp.74-85.

Poon, J. and Team, O., 2017. Decentralized Exchange and Payments Platform.

Thakur, V.S., 2016. BSE SME Exchange and NSE EMERGE Exchange Platforms in India and a Comparison between them. *IJLTEMAS*, *5*(3), pp.82-90.

Cozzolino, A., Corbo, L. and Aversa, P., 2021. Digital platform-based ecosystems: The evolution of collaboration and competition between incumbent producers and entrant platforms. *Journal of Business Research*, *126*, pp.385-400.

Kulmala, A., Keski-Koukkari, A., Mäki, K., Tikka, V., Romanenko, A., Mashlakov, A., Honkapuro, S., Partanen, J., Repo, S., Jafary, P. and Järventausta, P., 2019, September. Information exchange platform for enabling ancillary services from distributed energy resources. In *2019 16th International Conference on the European Energy Market (EEM)* (pp. 1-5). IEEE.

Saadatmand, F., Lindgren, R. and Schultze, U., 2019. Configurations of platform organizations: Implications for complementor engagement. *Research policy*, *48*(8), p.103770.

Andersson Schwarz, J., 2017. Platform logic: An interdisciplinary approach to the platform‐based economy. *Policy & Internet*, *9*(4), pp.374-394.

Kovel, D.E. and McNeela, A., 2019. The Commodities Exchange Act and the Presumption against Extraterritoriality: An Examination of Transnational, Platform-Based Electronic Trading under Second Circuit Precedent. *Journal of International Business and Law*, *18*(2), p.4.

Cutolo, D. and Kenney, M., 2021. Platform-dependent entrepreneurs: Power asymmetries, risks, and strategies in the platform economy. *Academy of management perspectives*, *35*(4), pp.584-605.

Saggese, P., Belmonte, A., Dimitri, N., Facchini, A. and Böhme, R., 2023. Arbitrageurs in the Bitcoin ecosystem: Evidence from user-level trading patterns in the Mt. Gox exchange platform. *Journal of Economic Behavior & Organization*, *213*, pp.251-270.

Gillespie, T., 2017. Platforms are not intermediaries. *Geo. L. Tech. Rev.*, *2*, p.198.