Inventory Management System

Vraj Patel
16014023060
Department of Electronics Engineering
KJ Somaiya School of Engineering
vraj.pp@somaiya.edu

Sahana Iyer
16014023046
Department of Electronics Engineering
KJ Somaiya School of Engineering
sahanag.iyer@somaiya.edu

Under the Guidance of
Prof. Umang Patel
Prof. Pragya Gupta
Department of Electronics Engineering
KJ Somaiya School of Engineering

***Abstract*—This project presents an Inventory Management System tailored for small businesses. The system allows basic efficient tracking and management of inventory data, including fields such as product ID, name, price per item, quantity, total price, and customer information. By providing an organized structure and monitored updates, this system improves the management of stock levels, aids in data-driven decisions, and enhances the store's operational efficiency.**

***Index Terms*—Inventory Management, Data Organization, Small businesses, Database, Consistent data entry**

I. Introduction

Inventory management is crucial for small businesses to operate efficiently and meet customer demand. Without a robust system, managing inventory can become challenging, leading to issues like stockouts or overstocking, which directly impact profitability. This project aims to develop an Inventory Management System that supports small businesses by providing real-time updates, efficient data handling, and an intuitive interface. By organizing and tracking essential inventory data such as product ID, name, price, quantity, and customer details, the system empowers small businesses to make data-driven decisions, optimize stock levels, and enhance overall operational efficiency.

II. System Overview

The Inventory Management System is built as a standalone application using Java Swing, GUI and mySQL database. The system allows users to add, update, delete, and view inventory data efficiently. The architecture includes a simple GUI for user interaction, a backend that manages data processing, and a database to store inventory records.

III. System Architecture

The system architecture comprises the following layers and components, designed to support comprehensive inventory management:

- **User Interface (UI):** Provides an intuitive interface using Java Swing for easy interaction. The UI allows users to manage inventory, suppliers, and pending orders efficiently.
- **Backend Logic:** Handles core functionality, including:
  - **Inventory Manager:** Manages CRUD (Create, Read, Update, Delete) operations for inventory items.
  - **Supplier Manager:** Tracks supplier details, enabling businesses to manage, view contact information, and ensure a steady supply chain.
  - **Order Manager:** Handles pending orders, allowing users to track incoming stock and manage outstanding orders, ensuring inventory levels are accurately maintained.
- **Database Layer:** A structured database that stores data for inventory, suppliers, and pending orders. This layer ensures data consistency and facilitates real-time tracking.

IV. Design and Implementation

The system is designed to be modular, ensuring easy maintenance and future upgrades. The core modules include:

- **Inventory Manager:** Handles CRUD (Create, Read, Update, Delete) operations for inventory data.
- **Data Persistence:** Stores inventory data in a local database to maintain consistency and accuracy.

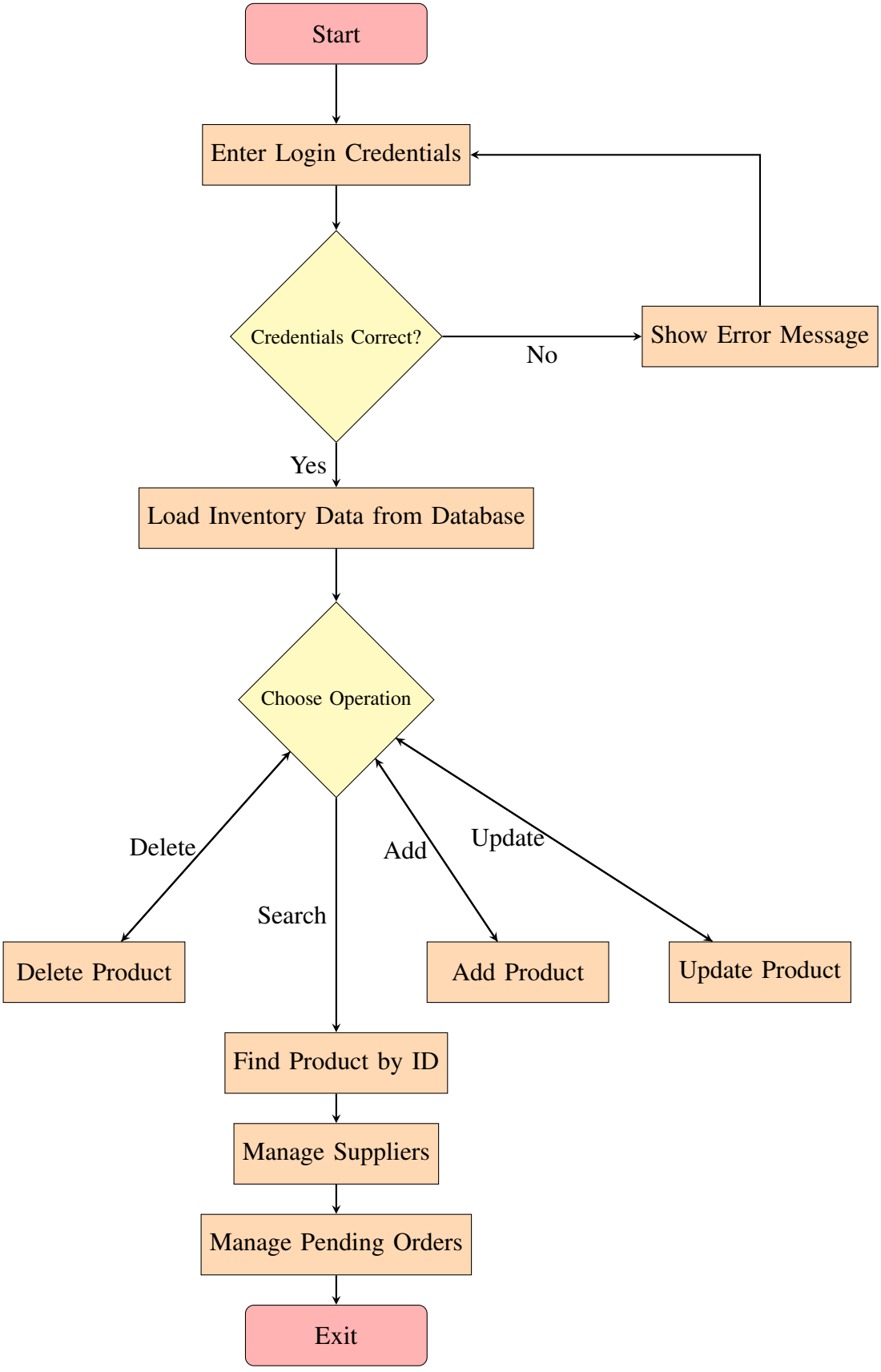
V. Key Features

- **Data Management:** Allows efficient entry and management of inventory data.
- **Real-Time Tracking:** Provides up-to-date information on stock levels.
- **User-Friendly Interface:** Facilitates ease of use and quick access to inventory records.

VI. System Workflow

The following flowchart illustrates the workflow of the Inventory Management System, including the login process, inventory data management, and additional functionalities like managing suppliers and pending orders.

A. Algorithm Flowchart:



VII. Challenges

Implementing efficient data handling mechanisms and ensuring system stability were some of the challenges faced during development. Managing data consistency across various components was critical to ensure data integrity.

Test Case 1: User Login (Admin)

Test Objective: Verify that the admin can log in with valid credentials.

Test Steps:

1) Launch the application.
2) Enter the admin username and password.
3) Click the "Login" button.

Expected Result: The admin should be redirected to the inventory dashboard.

Test Data:

- Username: `admin@union.com`
- Password: `password123`

Test Case 2: User Login (Invalid Credentials)

Test Objective: Verify that the system rejects login with invalid credentials.

Test Steps:

1) Launch the application.
2) Enter an invalid admin username and/or password.
3) Click the "Login" button.

Expected Result: The system should display an error message (e.g., "Invalid username or password").

Test Data:

- Username: `wrongUser`
- Password: `wrongPass`

Test Case 3: Add New Item to Inventory

Test Objective: Verify that the system allows adding a new item to the inventory.

Test Steps:

1) Log in as an admin.
2) Enter details required.
    - ID: `5`
    - Name: `Product F`
    - Quantity: `30`
    - Price: `299`
3) click the 'Add Product' button.

Expected Result: The new item should be successfully added to the inventory.

Test Data:

- ID: `5`
- Name: `Product F`
- Quantity: `30`
- Price: `299`

Test Case 4: Update Item Details

Test Objective: Verify that the system allows updating the details of an existing item.

Test Steps:

1) Log in as an admin.
2) Search for an existing item (e.g., `Laptop`).
3) Update the quantity or price.
4) Click the "Save Changes" button.

Expected Result: The system should update the item details and display a confirmation message.

Test Data:

- ID: `6`
- Name: `Product G`
- Quantity: `700`
- Price: `500`

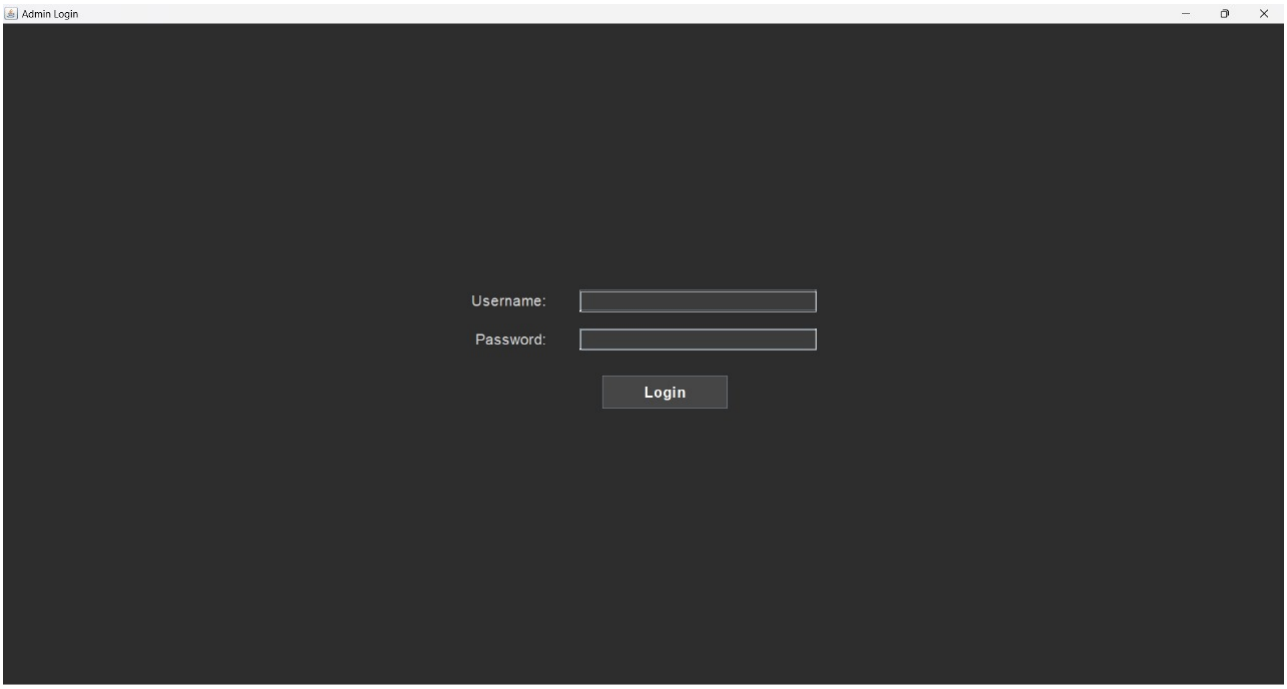


Fig. 1. TEST CASE 1

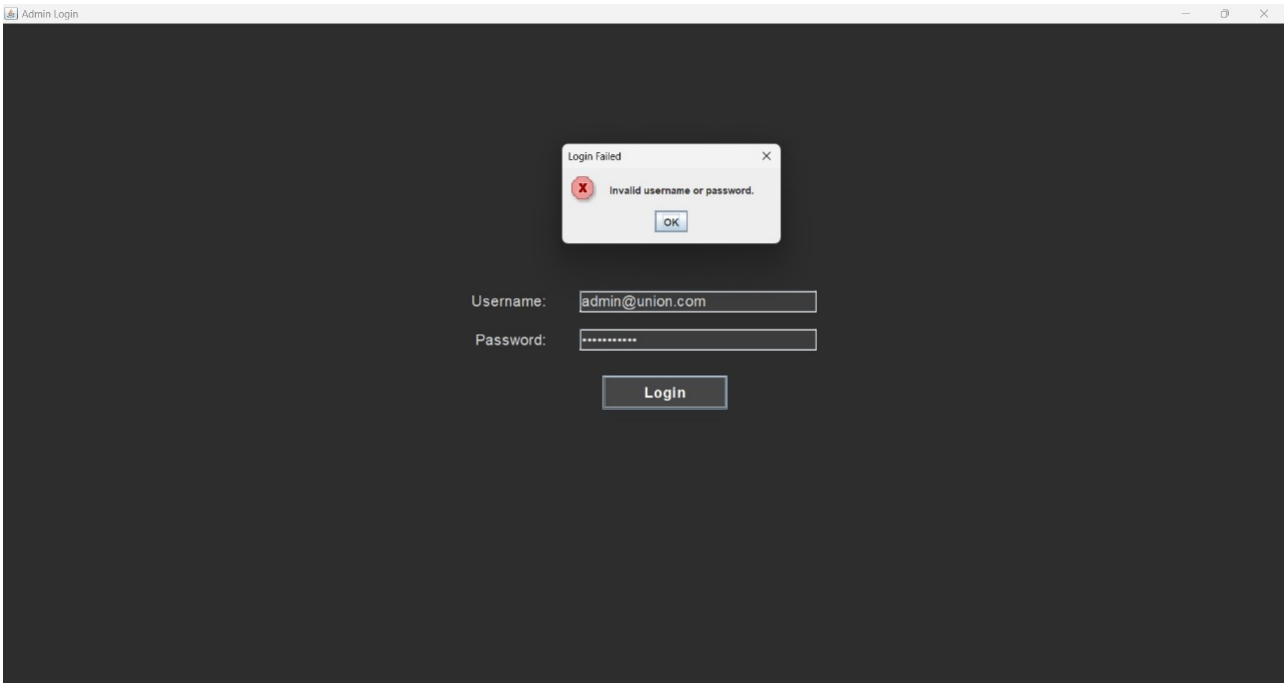


Fig. 2. TEST CASE 2

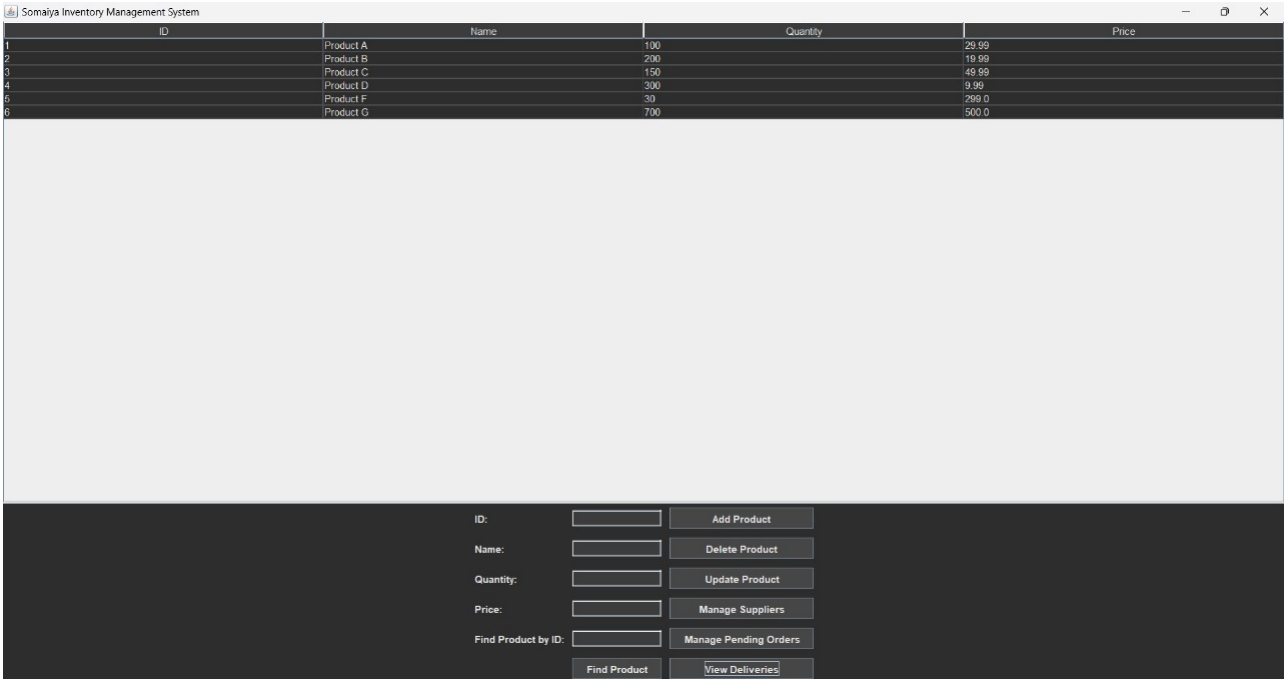


Fig. 3. TEST CASES 3,4

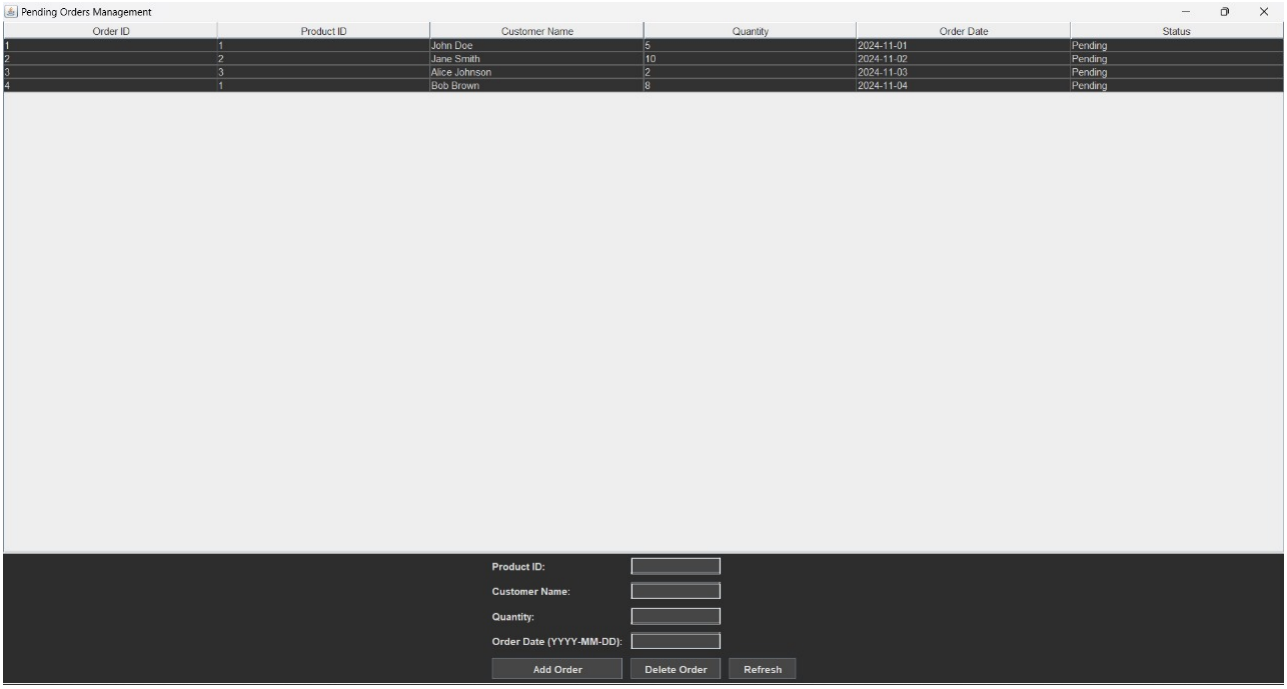


Fig. 4. PENDING ORDERS MANAGEMENT

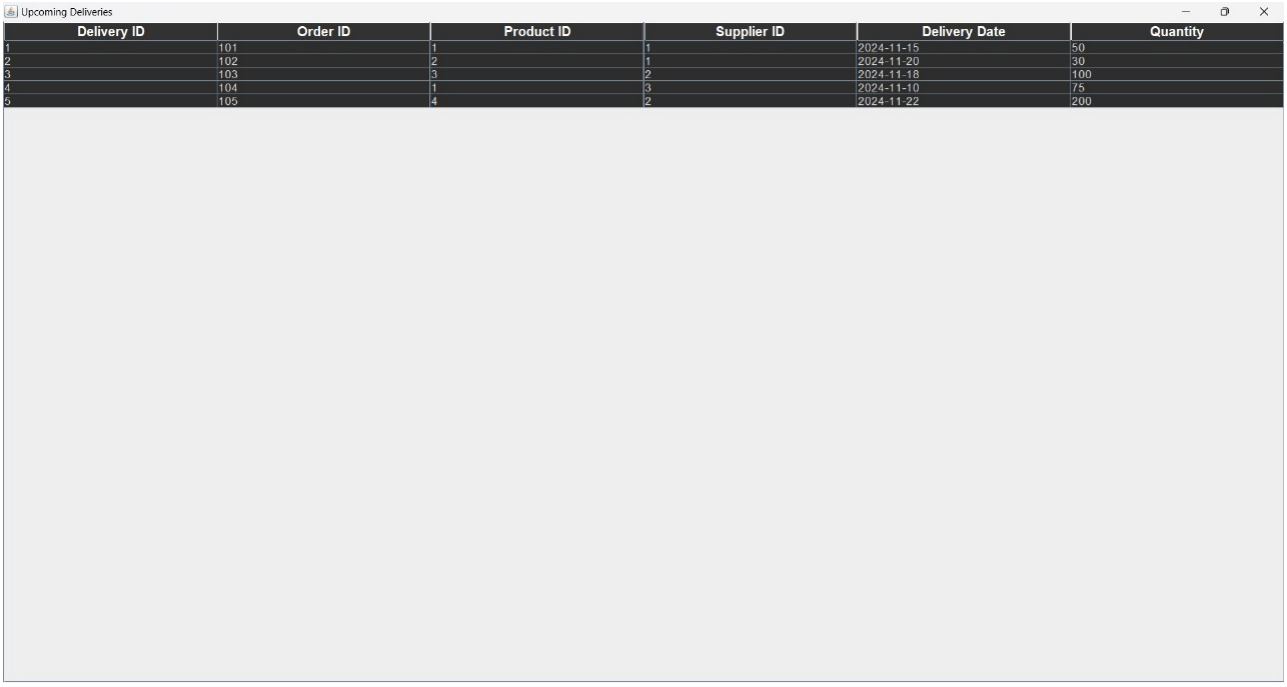


Fig. 5. UPCOMING DELIVERIES

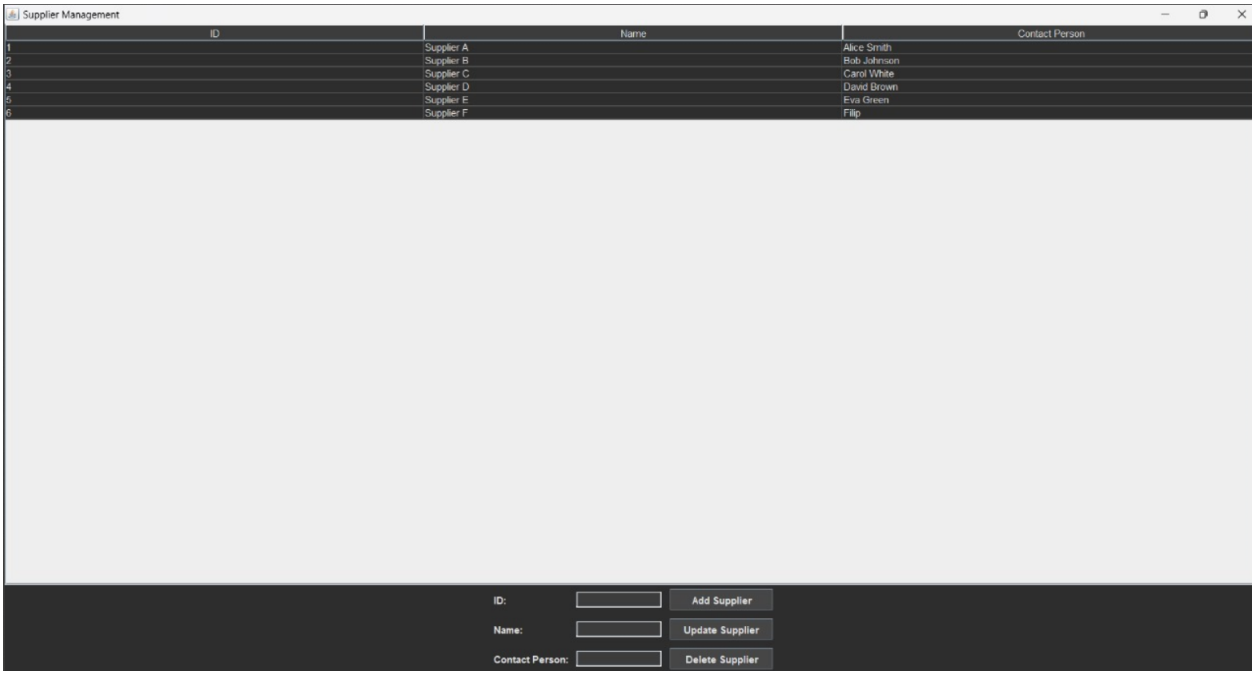


Fig. 6. SUPPLIERS MANAGEMENT

VIII. Object-Oriented Programming Concepts in Inventory Management System

The following are the key Object-Oriented Programming (OOP) concepts implemented in the provided code:

1. Object

An **Object** is an instance of a class. In the given code, instances of the `InventoryManagementSystem` and `AdminLogin` classes are created as objects. For example:

- `InventoryManagementSystem inventoryManagementSystem = new InventoryManagementSystem();`
- `AdminLogin login = new AdminLogin();`

Each object holds its own data and behavior as defined by the class.

2. Class

A **Class** is a blueprint for creating objects. It defines properties (attributes) and methods (functions) that the objects will have. In the provided code, the classes `InventoryManagementSystem` and `AdminLogin` define the structure for objects representing the inventory system and the login system, respectively.

3. Inheritance

Inheritance is the mechanism where one class can inherit the properties and behaviors of another. In the code, inheritance is not explicitly demonstrated by extending a parent class. However, since the `JFrame` class (from the Swing library) is extended by both `InventoryManagementSystem` and `AdminLogin`, they inherit the properties and methods of the `JFrame` class, such as creating and displaying a window.

4. Polymorphism

Polymorphism allows methods to do different things based on the object they are acting upon. In the given code, the method `ActionListener` is used for multiple buttons like `addButton`, `deleteButton`, `updateButton`, etc. Each button, when clicked, performs a different action (`addProduct`, `deleteProduct`, `updateProduct`), demonstrating method polymorphism.

5. Abstraction

Abstraction is the concept of hiding the complex implementation details and showing only the essential features. The code uses abstraction in the following way:

- The details of database operations (like connecting to the database and executing queries) are abstracted inside methods such as `loadProductsFromDatabase`, `addProduct`, `deleteProduct`, etc.
- The classes `InventoryManagementSystem` and `AdminLogin` abstract away the internal complexity of user interface creation, focusing on user interactions.

6. Encapsulation

Encapsulation is the bundling of data and methods that operate on that data within a single unit, i.e., a class. The code demonstrates encapsulation in the following ways:

- The attributes like `idField`, `nameField`, `quantityField`, and `priceField` are encapsulated in the `InventoryManagementSystem` class, and their values are accessed and modified through specific methods.
- The database credentials and connection details are encapsulated within the methods and not exposed outside the class.

IX. Testing and Results

The application was rigorously tested for:

- **Functionality:** Ensured smooth operation of CRUD functions.
- **Data Persistence:** Verified accurate data storage and retrieval.
- **Error Handling:** Tested application response to incorrect or missing data entries.

The test results indicate that the system meets its objectives, providing a robust solution for inventory management.

X. Conclusion

The Inventory Management System effectively meets the unique requirements of small businesses, particularly those in retail, by offering a streamlined approach to data management and inventory tracking. Its user-friendly interface and modular architecture allow small business owners to easily manage their stock, ensuring accurate records and timely order fulfillment.

Future enhancements could focus on incorporating features such as cloud-based data storage for increased accessibility and security, as well as integration with e-commerce platforms to facilitate online sales and expand market reach. These improvements would further empower small businesses to adapt to the evolving retail landscape and drive growth in an increasingly competitive environment.

XI. Plagiarism Check