RAMAIAH INSTITUTE OF TECHNOLOGY

MSR NAGAR, BENGALURU, 560054

Mini Project Report on

Inventory Management System

Submitted in partial fulfillment of the OTHER COMPONENT requirements of the course Advanced Java for the IV Semester of ***Bachelor of Engineering in Information Science and Engineering***

Submitted by

Nithish Reddy

(1MS23IS029)

Anushree

(1MS23IS019)

Under the Guidance of

Faculty Incharge

Evangeline D

Dept. of ISE

Department of Information Science and Engineering

Ramaiah Institute of Technology

2024 – 2025

INDEX

Description of the project

The Java Inventory Management System is a desktop application designed to help small businesses manage their stock efficiently. It provides a user-friendly graphical interface (GUI) built with Java Swing, allowing users to interact with the system easily. The project utilizes a database (likely MySQL, based on the earlier command) to persistently store information about products, users, and stock transactions.

Objective

This college project aims to design and implement a desktop-based inventory management system using Java, Swing for the GUI, and MySQL for data persistence. The system streamlines core inventory operations—product tracking, stock control, transaction auditing, and user management—while demonstrating key software engineering principles learned during the course.

Core Functionalities

1. **User Authentication**
   - Secure login with credential validation
   - Session management (`loggedInUser` tracking)
2. **Product Management**
   - CRUD Operations : Add, view, update, and delete products
   - Data Validation : Price (decimal), quantity (integer), and mandatory field checks
   - Real-time Updates : Table refresh after modifications
3. **Stock Control**
   - Stock In/Out Transactions : Dedicated interface for inventory adjustments
   - Quantity Validation : Prevents overstocking/understocking
   - Live Dashboard : Current stock levels displayed in a table
4. **Transaction History**
   - Audit Trail : Timestamped records of all stock movements
   - Filtering : View transactions by product
   - Performance Optimization : Product name caching for fast lookups
5. **Role-Based Navigation**
   - Main Dashboard : Intuitive access to features (Manage Products, Stock, Transactions)
   - Logout : Secure session termination

The project demonstrates fundamental concepts of Java GUI development, database interaction using JDBC, and a basic three-tier architecture using Data Access Objects (DAOs) to separate the UI, application logic, and data access layers. While the current implementation has a basic login, it highlights the need for more robust security measures like password hashing in real-world applications.

Technical Stack

Component	**Technology**
Frontend	Java Swing (GUI)
Backend Logic	Core Java (OOP, Event Handling)
Database	MySQL
Data Access	JDBC, DAO Pattern
Architecture	Layered (GUI → Models → DB)

Implementation (Psuedo-codes of prime methods)

1. User Login Authentication

```
Function validateUser(username, password):
    Connect to the database
    Prepare SQL to fetch user where username = ? and password = ?
    Execute query
    If result exists:
        Return user object with role
    Else:
        Return null
```

2. Add Product

```
Function addProduct(product):
    Connect to the database
    Prepare SQL INSERT statement with product data
    Execute update
    If successful:
        Return true
    Else:
        Return false
```

3. Edit Product

```
Function updateProduct(product):
    Connect to the database
    Prepare SQL UPDATE statement where id = product.id
    Execute update
    Return success status
```

4. Delete Product

```
Function deleteProduct(productId):
    Connect to the database
    Prepare SQL DELETE statement for given ID
    Execute update
    Return success status
```

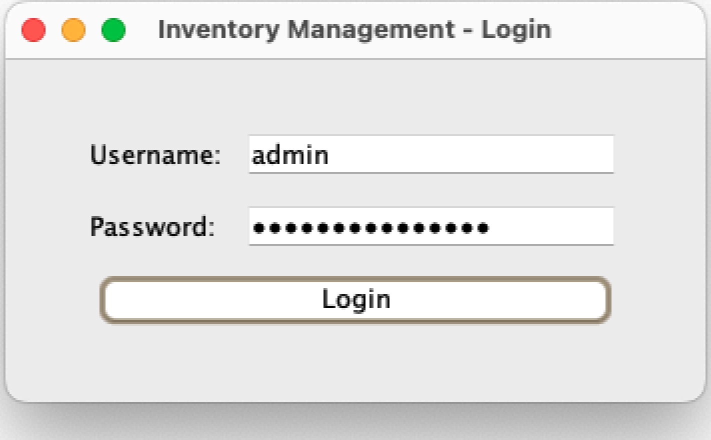
5. Record Stock-in / Stock-out

```
Function recordTransaction(productId, type, quantity):
    Begin transaction
    If type == 'IN':
        Increase product quantity by given amount
    Else if type == 'OUT':
        Decrease product quantity by given amount
    Insert transaction record in stock_transaction table
    Commit transaction
    Return success status
```

6. Get Current Inventory

```
Function getAllProducts():
    Connect to the database
    Prepare SELECT * FROM products
    Execute query
    For each record:
        Create Product object and add to list
    Return list of products
```

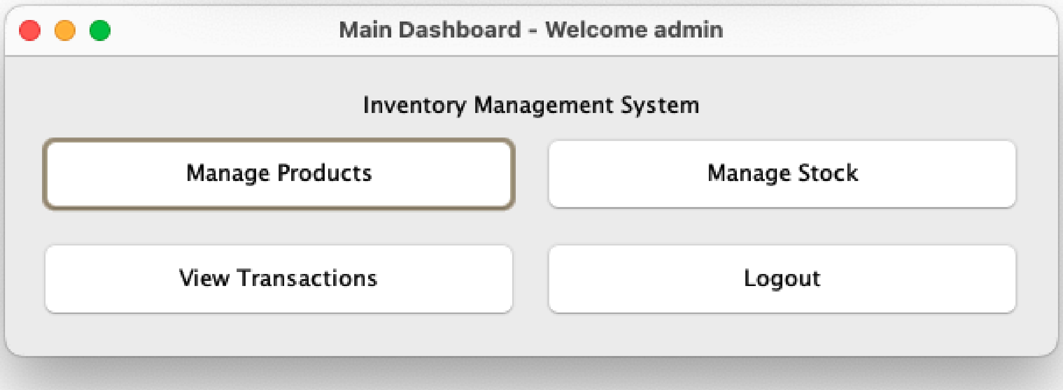
Screenshots



Inventory Management - Login

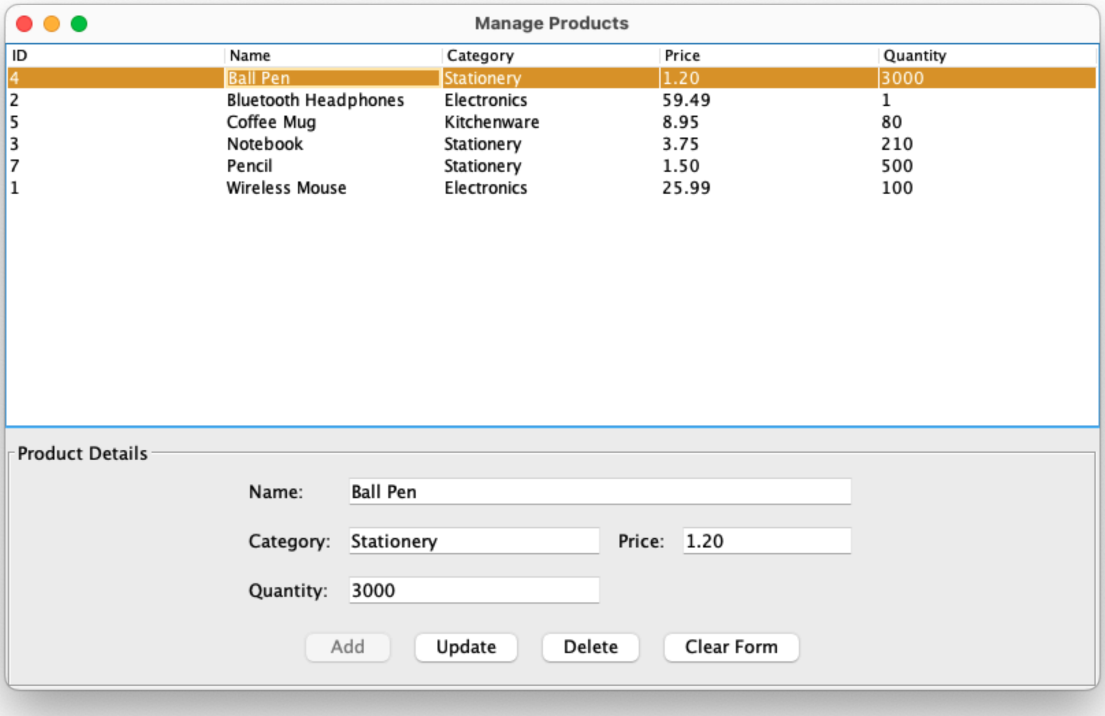
Username: admin

Password: ●●●●●●●●●●●●●●●



Main Dashboard - Welcome admin

Inventory Management System



Manage Products

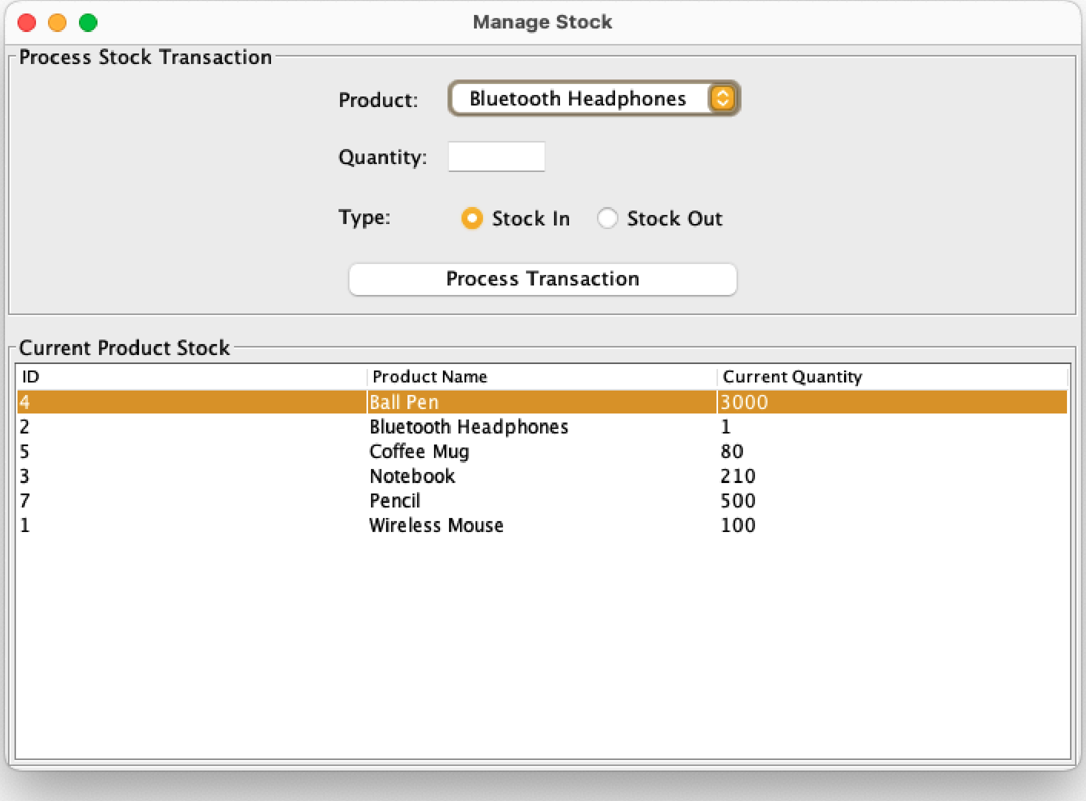
ID	Name	Category	Price	Quantity
4	Ball Pen	Stationery	1.20	3000
2	Bluetooth Headphones	Electronics	59.49	1
5	Coffee Mug	Kitchenware	8.95	80
3	Notebook	Stationery	3.75	210
7	Pencil	Stationery	1.50	500
1	Wireless Mouse	Electronics	25.99	100

Product Details

Name: Ball Pen

Category: Stationery Price: 1.20

Quantity: 3000



Process Stock Transaction

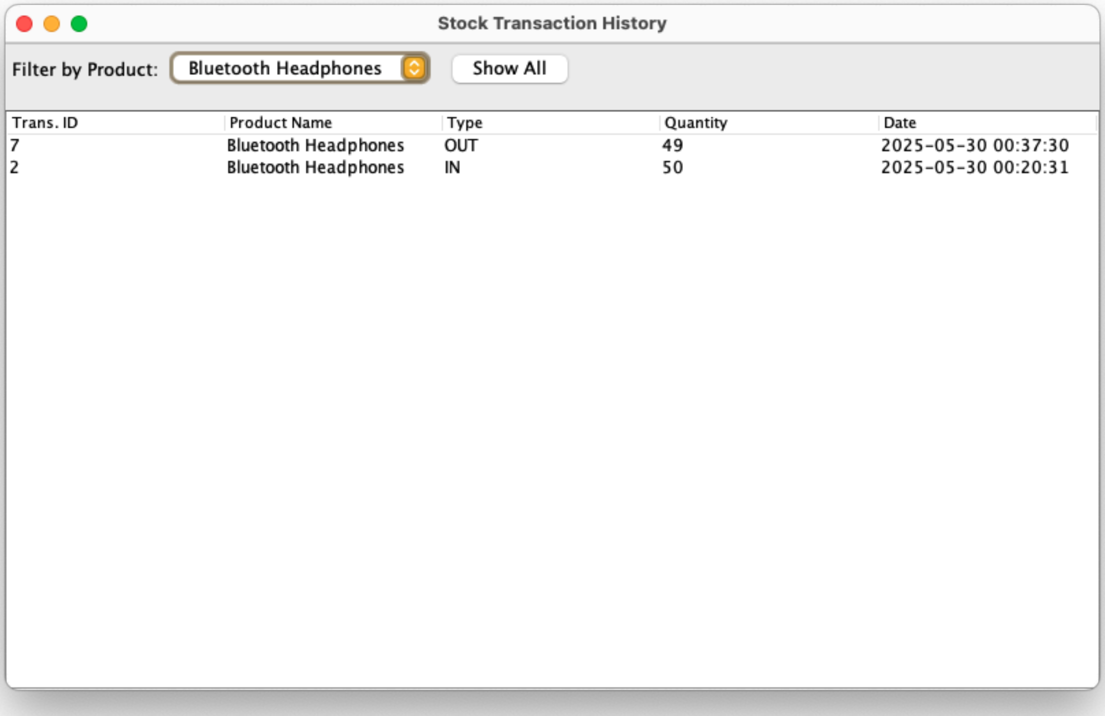
Product: Bluetooth Headphones

Quantity:

Type: ● Stock In ○ Stock Out

Current Product Stock

ID	Product Name	Current Quantity
4	Ball Pen	3000
2	Bluetooth Headphones	1
5	Coffee Mug	80
3	Notebook	210
7	Pencil	500
1	Wireless Mouse	100



Filter by Product: Bluetooth Headphones

Trans. ID	Product Name	Type	Quantity	Date
7	Bluetooth Headphones	OUT	49	2025-05-30 00:37:30
2	Bluetooth Headphones	IN	50	2025-05-30 00:20:31

Conclusion

This Java mini-project demonstrates a **complete, functional inventory management system** built with Swing for the GUI and MySQL for data persistence. Here's a concise summary of its key aspects:

Core Features Implemented

1. **User Authentication**
   - Secure login with role-based access control (static `loggedInUser` tracking)
   - Validation for empty/wrong credentials
2. **Product Management**
   - CRUD operations (Create, Read, Update, Delete products)
   - Input validation for prices/quantities
   - Tabular display with real-time updates
3. **Stock Control**
   - "Stock In/Out" transactions with quantity validation
   - Radio buttons for transaction type
   - Live stock-level monitoring
4. **Transaction History**
   - Filterable records by product
   - Automatic product-name caching for performance
   - Timestamped audit trail

Technical Strengths

- **Layered Architecture**:
  Clear separation between GUI (`gui`), data models (`models`), and database access (`db`).

- **DAO Pattern**:
  `ProductDAO`, `UserDAO`, and `StockTransactionDAO` encapsulate database operations.

- **Swing Best Practices**:
  - `GridBagLayout` for responsive UIs
  - `TableModel` for JTable data binding
  - Thread-safe `SwingUtilities.invokeLater()`
- **Data Validation**:
  Robust checks for negative quantities, invalid prices, and empty fields.

- **Error Handling**:
  User-friendly error dialogs for database failures/constraints.

Final Assessment

This project successfully delivers a **functional inventory system** covering core workflows: user auth, product lifecycle, stock tracking, and auditing. Its modular design facilitates maintenance, and the UI offers intuitive navigation. While security improvements are needed for production use, the foundation demonstrates strong Java/Swing proficiency and effective database integration.

Future Scope: Barcode scanning, supplier management, and sales modules could extend this system into a full POS solution.

References

1. Oracle Java Documentation[Link]
2. MySQL 8.0 Reference Manual [Link]
3. MySQL Connector/J Developer Guide[Link]
4. Java Swing Tutorial (Oracle) [Link]
5. GeeksforGeeks – JDBC in Java[Link]
6. Stack Overflow[Link]
7. GitHub – Sample Inventory Management Systems in Java[Link]