**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”,** Belagavi-590018, Karnataka



**Advanced Java(BCS613D)**

**Mini Project Report**

**Submitted by**

**USN NAME**

**1BI22CS164 Srivallabha KM**

**1BI22CS133 Rohit K H**

**1BI22CS152 Shashi Madari**

Under the Guidance of

**Chaithra G V**

Assistant Professor

Department of CSE, BIT

Bengaluru-560004

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**BANGALORE INSTITUTE OF TECHNOLOGY**

K.R. Road, V.V. Pura, Bengaluru-560 004

**2024-25**

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”,** Belagavi-590018, Karnataka

**BANGALORE INSTITUTE OF TECHNOLOGY**

Bengaluru-560 004

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Certificate**

This is to certify that the Mini Project work on Advanced Java (BCS613D) carried out by

**USN NAME**

**1BI22CS164 Srivallabha KM**

**1BI22CS133 Rohit K H**

**1BI22CS152 Shashi Madari**

Bonafide students of VI semester B.E for the practical fulfillment of the requirements for the Bachelor’s Degree in Computer Science & Engineering of the Visvesvaraya Technological University during the academic year 2024-25. The Mini Project report has been approved as it satisfies the academic requirements in respect the Assessment

**Faculty Name Head of Department**

# TABLE OF CONTENTS

1. Introduction

2. Objective

3. Mini Project 1

3.1.Problem Statement 1

3.2 Implementation 1

4. Mini Project 2

4.1.Problem Statement 2

4.2 Implementation 2

5 Output Screenshots

6. Conclusion

7. References

**Chapter 1**

**Introduction**

This report includes two mini projects based on Advanced Java concepts. Both projects help in understanding how Java can be used to build real applications — one as a desktop app and the other as a web app.

**Mini Project 1: Java Swing Application with Database Connectivity**

This project is a desktop application built using Java Swing. It connects to a database using JDBC. The user can select a table, choose specific columns, and select a collection type (like List or Set) to view the data. The project also uses important Java concepts like Generics, Iterators, Comparators, and Enhanced For-Loops to handle and display data efficiently.

**Mini Project 2: JSP and Servlet-Based Web Application**

This project is a simple web application built using JSP and Servlets. It has two main parts: a login system and a calculator. The login page takes a username and password, which is checked using a Servlet. The calculator page takes two numbers and an operation (like add or divide), and the Servlet returns the result. The project also includes servlet configuration using web.xml.

These projects help in learning how to create user-friendly desktop and web applications using Advanced Java.

**Chapter 2**

**Objective**

1. To gain practical knowledge of Java GUI programming using Swing.
2. To understand JDBC connectivity and database interaction.
3. To implement collections and generics in Java.
4. To develop web applications using JSP and Servlets.

**Chapter 3**

**Mini Project 1**

**3.1 Problem Statement**

To develop a Java Swing application with MySQL/JDBC integration to manage medical inventory. Users can select tables (Medicines/Suppliers), apply collections (ArrayList/TreeSet) with Generics, and use Iterators/Comparators for sorting. Features include CRUD operations, low-stock/expiry alerts, and search functionality, all displayed in a user-friendly GUI with enhanced for-loops for clean data processing. Built with XAMPP for database backend.

**3.2 Implementation**

**Medical Store Inventory System - Implementation Details**

**1. Java Swing GUI**

• The main application window is created using JFrame.  
• Buttons like Load, Insert, Update, Delete, and Search are added using JButton.  
• Data is displayed using a JTextArea inside a JScrollPane.  
• JComboBox is used to let users select:  
 - The table to operate on: Medicines or Suppliers  
 - The Java collection to use: ArrayList or TreeSet  
• Event handling is done using ActionListeners for each button.

**2. JDBC Database Connectivity**

• Database used: MySQL (via XAMPP).  
• Connection is established using:  
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MedicalStore", "root", "");  
• JDBC operations (CRUD) are performed using PreparedStatement and ResultSet.  
• Driver loaded via:  
Class.forName("com.mysql.cj.jdbc.Driver");

**3. Data Retrieval and Display**

• loadData() runs a SELECT \* FROM table query.  
• Fetched data from ResultSet is mapped into Java objects (Medicine or Supplier).  
• Objects are added to the chosen collection (List, Set) and displayed in the text area.

**4. CRUD Operations**

• Insert: User inputs data via JOptionPane.showInputDialog(). Data is inserted using INSERT INTO with PreparedStatement.  
• Update: Updates based on primary key using UPDATE SQL command.  
• Delete: Uses DELETE FROM table WHERE id = ?.  
• Search: Iterates over collection using Iterator or enhanced for-loop to find matches.

**5. Collections & Generics**

• Uses:  
List<Medicine> medList = new ArrayList<>();  
Set<Supplier> supSet = new TreeSet<>(new SupplierNameComparator());  
• Generics ensure type safety for all collections.  
• Comparators used to sort data in TreeSet by name, price, etc.

**6. Sorting Mechanism**

• Sorting is automatic in TreeSet using a custom Comparator.  
• For ArrayList, uses Collections.sort() or stream().sorted() for alternative sorting options.

**7. Search Feature**

• Searches done over the selected collection using:  
for (Medicine m : medList) {  
 if (m.getName().equalsIgnoreCase(searchInput)) { ... }  
}

**8. Alert System**

• Low stock:  
if (medicine.getQuantity() <= medicine.getThreshold()) {  
 // Alert  
}  
• Expiry check:  
if (medicine.getExpiryDate().isBefore(LocalDate.now().plusDays(30))) {  
 // Expiry warning  
}

**9. Input Validation & Error Handling**

• All inputs are validated (e.g., quantity must be a number).  
• Try-catch blocks are used to handle SQLException and input errors.  
• Resources like ResultSet, Statement, and Connection are closed in finally blocks.

**10. Code Structure**

• Each module (CRUD, alert, DB connection, GUI) is separated into classes or methods.  
• Follows SOLID principles for maintainability.  
• GUI remains responsive using SwingWorker where needed.

**Database Setup Script**

CREATE DATABASE IF NOT EXISTS MedicalStore;

USE MedicalStore;

-- Create Suppliers table

CREATE TABLE IF NOT EXISTS Suppliers (

supplier\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

contact VARCHAR(20),

email VARCHAR(100),

address VARCHAR(200)

);

-- Create Medicines table

CREATE TABLE IF NOT EXISTS Medicines (

medicine\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

batch\_no VARCHAR(50) NOT NULL,

price DECIMAL(10,2) NOT NULL,

quantity INT NOT NULL,

threshold INT NOT NULL,

expiry\_date DATE NOT NULL,

supplier\_id INT,

FOREIGN KEY (supplier\_id) REFERENCES Suppliers(supplier\_id)

);

-- Insert sample supplier data

INSERT INTO Suppliers VALUES

(1, 'Pharma Distributors', '9876543210', 'pharma@example.com', '123 Pharma St, Mumbai'),

(2, 'MediCorp', '8765432109', 'medicorp@example.com', '456 Health Ave, Delhi'),

(3, 'Global Medicines', '7654321098', 'global@example.com', '789 World Rd, Bangalore');

-- Insert sample medicine data

INSERT INTO Medicines VALUES

(1, 'Paracetamol', 'BATCH001', 5.50, 50, 10, '2024-12-31', 1),

(2, 'Ibuprofen', 'BATCH002', 7.25, 15, 5, '2025-06-30', 1),

(3, 'Amoxicillin', 'BATCH003', 12.75, 8, 5, '2024-09-15', 2),

(4, 'Cetirizine', 'BATCH004', 4.20, 30, 10, '2025-03-31', NULL),

(5, 'Omeprazole', 'BATCH005', 8.90, 25, 10, '2024-11-30', 3),

(6, 'Aspirin', 'BATCH006', 3.50, 40, 15, '2025-01-31', NULL),

(7, 'Diazepam', 'BATCH007', 15.25, 12, 5, '2024-08-15', 2),

(8, 'Atorvastatin', 'BATCH008', 9.75, 18, 5, '2025-05-31', 3);

**Java Implementation: JDBC and Swing Setup**

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.FlowLayout;

import java.awt.Font;

import java.awt.Insets;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

import java.util.Map;

import java.util.TreeSet;

import java.util.stream.Collectors;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.SwingUtilities;

import javax.swing.border.EmptyBorder;

public class MedicalStoreApp extends JFrame {

JComboBox<String> tableSelector, collectionSelector;

JTextArea outputArea;

JButton loadButton, insertButton, updateButton, deleteButton, searchButton, alertButton;

public MedicalStoreApp() {

setTitle("🏥 Medical Store Inventory System");

setSize(1000, 700);

setLocationRelativeTo(null);

setLayout(new BorderLayout(10, 10));

getContentPane().setBackground(new Color(245, 255, 250));

// Top Panel

JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 10));

topPanel.setBackground(new Color(70, 130, 180));

JLabel tableLabel = new JLabel("Table:");

tableLabel.setForeground(Color.WHITE);

tableLabel.setFont(new Font("Arial", Font.BOLD, 14));

JLabel collectionLabel = new JLabel("Collection:");

collectionLabel.setForeground(Color.WHITE);

collectionLabel.setFont(new Font("Arial", Font.BOLD, 14));

tableSelector = new JComboBox<>(new String[]{"Medicines", "Suppliers"});

collectionSelector = new JComboBox<>(new String[]{"ArrayList", "TreeSet"});

loadButton = new JButton("Load Data");

loadButton.setFont(new Font("SansSerif", Font.BOLD, 14));

loadButton.setBackground(new Color(255, 215, 0));

insertButton = new JButton("Insert");

insertButton.setFont(new Font("SansSerif", Font.BOLD, 14));

insertButton.setBackground(new Color(60, 179, 113));

insertButton.setForeground(Color.WHITE);

updateButton = new JButton("Update");

updateButton.setFont(new Font("SansSerif", Font.BOLD, 14));

updateButton.setBackground(new Color(255, 140, 0));

updateButton.setForeground(Color.WHITE);

deleteButton = new JButton("Delete");

deleteButton.setFont(new Font("SansSerif", Font.BOLD, 14));

deleteButton.setBackground(new Color(220, 20, 60));

deleteButton.setForeground(Color.WHITE);

searchButton = new JButton("Search");

searchButton.setFont(new Font("SansSerif", Font.BOLD, 14));

searchButton.setBackground(new Color(138, 43, 226));

searchButton.setForeground(Color.WHITE);

alertButton = new JButton("Low Stock");

alertButton.setFont(new Font("SansSerif", Font.BOLD, 14));

alertButton.setBackground(new Color(255, 69, 0));

alertButton.setForeground(Color.WHITE);

topPanel.add(tableLabel);

topPanel.add(tableSelector);

topPanel.add(collectionLabel);

topPanel.add(collectionSelector);

topPanel.add(loadButton);

topPanel.add(insertButton);

topPanel.add(updateButton);

topPanel.add(deleteButton);

topPanel.add(searchButton);

topPanel.add(alertButton);

// Output Area

outputArea = new JTextArea();

outputArea.setEditable(false);

outputArea.setFont(new Font("Monospaced", Font.PLAIN, 13));

outputArea.setMargin(new Insets(10, 10, 10, 10));

JScrollPane scrollPane = new JScrollPane(outputArea);

scrollPane.setBorder(new EmptyBorder(10, 10, 10, 10));

add(topPanel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

loadButton.addActionListener(e -> loadData());

insertButton.addActionListener(e -> insertRecord());

updateButton.addActionListener(e -> updateRecord());

deleteButton.addActionListener(e -> deleteRecord());

searchButton.addActionListener(e -> searchRecord());

alertButton.addActionListener(e -> showLowStockAlert());

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setVisible(true);

}

private Connection getConnection() throws SQLException {

return DriverManager.getConnection("jdbc:mysql://localhost:3306/MedicalStore", "root", "");

}

private void loadData() {

String table = (String) tableSelector.getSelectedItem();

String collection = (String) collectionSelector.getSelectedItem();

outputArea.setText("");

try (Connection conn = getConnection()) {

if ("Medicines".equals(table)) {

List<Medicine> medicineList = new ArrayList<>();

try (Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM Medicines")) {

while (rs.next()) {

medicineList.add(new Medicine(

rs.getInt("medicine\_id"),

rs.getString("name"),

rs.getString("batch\_no"),

rs.getFloat("price"),

rs.getInt("quantity"),

rs.getInt("threshold"),

rs.getString("expiry\_date"),

rs.getInt("supplier\_id")

));

}

}

outputArea.append(String.format("%-4s | %-20s | %-10s | %-7s | %-8s | %-8s | %-12s | %-10s\n",

"ID", "Name", "Batch No", "Price", "Quantity", "Threshold", "Expiry Date", "Supplier ID"));

outputArea.append("--------------------------------------------------------------------------------------------------------\n");

if ("TreeSet".equals(collection)) {

TreeSet<Medicine> sortedSet = new TreeSet<>(Comparator.comparing(Medicine::getName));

sortedSet.addAll(medicineList);

for (Medicine med : sortedSet) {

outputArea.append(med + "\n");

}

} else {

for (Medicine med : medicineList) {

outputArea.append(med + "\n");

}

}

} else if ("Suppliers".equals(table)) {

List<Supplier> supplierList = new ArrayList<>();

try (Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM Suppliers")) {

while (rs.next()) {

supplierList.add(new Supplier(

rs.getInt("supplier\_id"),

rs.getString("name"),

rs.getString("contact"),

rs.getString("email"),

rs.getString("address")

));

}

}

outputArea.append(String.format("%-4s | %-20s | %-15s | %-25s | %-30s\n",

"ID", "Name", "Contact", "Email", "Address"));

outputArea.append("--------------------------------------------------------------------------------------------------------\n");

for (Supplier s : supplierList) {

outputArea.append(s + "\n");

}

}

} catch (Exception ex) {

ex.printStackTrace();

outputArea.setText("❌ Error loading data: " + ex.getMessage());

}

}

private void insertRecord() {

String table = (String) tableSelector.getSelectedItem();

if ("Medicines".equals(table)) {

try {

String idStr = JOptionPane.showInputDialog(this, "Enter Medicine ID (int):");

if (idStr == null) return;

int id = Integer.parseInt(idStr.trim());

String name = JOptionPane.showInputDialog(this, "Enter Medicine Name:");

if (name == null) return;

String batchNo = JOptionPane.showInputDialog(this, "Enter Batch Number:");

if (batchNo == null) return;

String priceStr = JOptionPane.showInputDialog(this, "Enter Price:");

if (priceStr == null) return;

float price = Float.parseFloat(priceStr.trim());

String quantityStr = JOptionPane.showInputDialog(this, "Enter Quantity:");

if (quantityStr == null) return;

int quantity = Integer.parseInt(quantityStr.trim());

String thresholdStr = JOptionPane.showInputDialog(this, "Enter Threshold (low stock alert):");

if (thresholdStr == null) return;

int threshold = Integer.parseInt(thresholdStr.trim());

String expiryDate = JOptionPane.showInputDialog(this, "Enter Expiry Date (YYYY-MM-DD):");

if (expiryDate == null) return;

// Validate date format

try {

LocalDate.parse(expiryDate);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Invalid date format. Please use YYYY-MM-DD");

return;

}

String supplierIdStr = JOptionPane.showInputDialog(this, "Enter Supplier ID (0 if none):");

if (supplierIdStr == null) return;

int supplierId = Integer.parseInt(supplierIdStr.trim());

try (Connection conn = getConnection();

PreparedStatement ps = conn.prepareStatement(

"INSERT INTO Medicines(medicine\_id, name, batch\_no, price, quantity, threshold, expiry\_date, supplier\_id) " +

"VALUES (?, ?, ?, ?, ?, ?, ?, ?)")) {

ps.setInt(1, id);

ps.setString(2, name);

ps.setString(3, batchNo);

ps.setFloat(4, price);

ps.setInt(5, quantity);

ps.setInt(6, threshold);

ps.setString(7, expiryDate);

ps.setInt(8, supplierId);

ps.executeUpdate();

}

JOptionPane.showMessageDialog(this, "Medicine added successfully.");

loadData();

} catch (NumberFormatException nfe) {

JOptionPane.showMessageDialog(this, "Invalid number format. Try again.");

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(this, "SQL Error: " + sqle.getMessage());

}

} else if ("Suppliers".equals(table)) {

try {

String idStr = JOptionPane.showInputDialog(this, "Enter Supplier ID (int):");

if (idStr == null) return;

int id = Integer.parseInt(idStr.trim());

String name = JOptionPane.showInputDialog(this, "Enter Supplier Name:");

if (name == null) return;

String contact = JOptionPane.showInputDialog(this, "Enter Contact Number:");

if (contact == null) return;

String email = JOptionPane.showInputDialog(this, "Enter Email:");

if (email == null) return;

String address = JOptionPane.showInputDialog(this, "Enter Address:");

if (address == null) return;

try (Connection conn = getConnection();

PreparedStatement ps = conn.prepareStatement(

"INSERT INTO Suppliers(supplier\_id, name, contact, email, address) " +

"VALUES (?, ?, ?, ?, ?)")) {

ps.setInt(1, id);

ps.setString(2, name);

ps.setString(3, contact);

ps.setString(4, email);

ps.setString(5, address);

ps.executeUpdate();

}

JOptionPane.showMessageDialog(this, "Supplier added successfully.");

loadData();

} catch (NumberFormatException nfe) {

JOptionPane.showMessageDialog(this, "Invalid number format. Try again.");

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(this, "SQL Error: " + sqle.getMessage());

}

}

}

private void updateRecord() {

String table = (String) tableSelector.getSelectedItem();

try (Connection conn = getConnection()) {

if ("Medicines".equals(table)) {

String idStr = JOptionPane.showInputDialog(this, "Enter the Medicine ID to update:");

if (idStr == null) return;

int id = Integer.parseInt(idStr.trim());

// Fetch current record

Medicine current = null;

try (PreparedStatement ps = conn.prepareStatement("SELECT \* FROM Medicines WHERE medicine\_id = ?")) {

ps.setInt(1, id);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

current = new Medicine(

rs.getInt("medicine\_id"),

rs.getString("name"),

rs.getString("batch\_no"),

rs.getFloat("price"),

rs.getInt("quantity"),

rs.getInt("threshold"),

rs.getString("expiry\_date"),

rs.getInt("supplier\_id")

);

} else {

JOptionPane.showMessageDialog(this, "No Medicine found with that ID.");

return;

}

}

}

// Show current values and ask for new values (can leave blank to keep current)

String name = JOptionPane.showInputDialog(this,

"Enter new Medicine Name (leave blank to keep '" + current.getName() + "'):");

if (name == null) return;

if (name.trim().isEmpty()) name = current.getName();

String batchNo = JOptionPane.showInputDialog(this,

"Enter new Batch No (leave blank to keep '" + current.getBatchNo() + "'):");

if (batchNo == null) return;

if (batchNo.trim().isEmpty()) batchNo = current.getBatchNo();

String priceStr = JOptionPane.showInputDialog(this,

"Enter new Price (leave blank to keep '" + current.getPrice() + "'):");

if (priceStr == null) return;

float price = priceStr.trim().isEmpty() ? current.getPrice() : Float.parseFloat(priceStr.trim());

String quantityStr = JOptionPane.showInputDialog(this,

"Enter new Quantity (leave blank to keep '" + current.getQuantity() + "'):");

if (quantityStr == null) return;

int quantity = quantityStr.trim().isEmpty() ? current.getQuantity() : Integer.parseInt(quantityStr.trim());

String thresholdStr = JOptionPane.showInputDialog(this,

"Enter new Threshold (leave blank to keep '" + current.getThreshold() + "'):");

if (thresholdStr == null) return;

int threshold = thresholdStr.trim().isEmpty() ? current.getThreshold() : Integer.parseInt(thresholdStr.trim());

String expiryDate = JOptionPane.showInputDialog(this,

"Enter new Expiry Date (YYYY-MM-DD) (leave blank to keep '" + current.getExpiryDate() + "'):");

if (expiryDate == null) return;

if (expiryDate.trim().isEmpty()) expiryDate = current.getExpiryDate();

else {

try {

LocalDate.parse(expiryDate);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Invalid date format. Please use YYYY-MM-DD");

return;

}

}

String supplierIdStr = JOptionPane.showInputDialog(this,

"Enter new Supplier ID (leave blank to keep '" + current.getSupplierId() + "'):");

if (supplierIdStr == null) return;

int supplierId = supplierIdStr.trim().isEmpty() ? current.getSupplierId() : Integer.parseInt(supplierIdStr.trim());

try (PreparedStatement ps = conn.prepareStatement(

"UPDATE Medicines SET name=?, batch\_no=?, price=?, quantity=?, threshold=?, expiry\_date=?, supplier\_id=? " +

"WHERE medicine\_id=?")) {

ps.setString(1, name);

ps.setString(2, batchNo);

ps.setFloat(3, price);

ps.setInt(4, quantity);

ps.setInt(5, threshold);

ps.setString(6, expiryDate);

ps.setInt(7, supplierId);

ps.setInt(8, id);

ps.executeUpdate();

}

JOptionPane.showMessageDialog(this, "Medicine updated successfully.");

loadData();

} else if ("Suppliers".equals(table)) {

String idStr = JOptionPane.showInputDialog(this, "Enter the Supplier ID to update:");

if (idStr == null) return;

int id = Integer.parseInt(idStr.trim());

// Fetch current record

Supplier current = null;

try (PreparedStatement ps = conn.prepareStatement("SELECT \* FROM Suppliers WHERE supplier\_id = ?")) {

ps.setInt(1, id);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

current = new Supplier(

rs.getInt("supplier\_id"),

rs.getString("name"),

rs.getString("contact"),

rs.getString("email"),

rs.getString("address")

);

} else {

JOptionPane.showMessageDialog(this, "No Supplier found with that ID.");

return;

}

}

}

String name = JOptionPane.showInputDialog(this,

"Enter new Supplier Name (leave blank to keep '" + current.getName() + "'):");

if (name == null) return;

if (name.trim().isEmpty()) name = current.getName();

String contact = JOptionPane.showInputDialog(this,

"Enter new Contact (leave blank to keep '" + current.getContact() + "'):");

if (contact == null) return;

if (contact.trim().isEmpty()) contact = current.getContact();

String email = JOptionPane.showInputDialog(this,

"Enter new Email (leave blank to keep '" + current.getEmail() + "'):");

if (email == null) return;

if (email.trim().isEmpty()) email = current.getEmail();

String address = JOptionPane.showInputDialog(this,

"Enter new Address (leave blank to keep '" + current.getAddress() + "'):");

if (address == null) return;

if (address.trim().isEmpty()) address = current.getAddress();

try (PreparedStatement ps = conn.prepareStatement(

"UPDATE Suppliers SET name=?, contact=?, email=?, address=? WHERE supplier\_id=?")) {

ps.setString(1, name);

ps.setString(2, contact);

ps.setString(3, email);

ps.setString(4, address);

ps.setInt(5, id);

ps.executeUpdate();

}

JOptionPane.showMessageDialog(this, "Supplier updated successfully.");

loadData();

}

} catch (NumberFormatException nfe) {

JOptionPane.showMessageDialog(this, "Invalid number format. Try again.");

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(this, "SQL Error: " + sqle.getMessage());

}

}

private void deleteRecord() {

String table = (String) tableSelector.getSelectedItem();

try (Connection conn = getConnection()) {

if ("Medicines".equals(table)) {

String idStr = JOptionPane.showInputDialog(this, "Enter the Medicine ID to delete:");

if (idStr == null) return;

int id = Integer.parseInt(idStr.trim());

int confirm = JOptionPane.showConfirmDialog(this,

"Are you sure you want to delete Medicine with ID " + id + "?",

"Confirm Delete",

JOptionPane.YES\_NO\_OPTION);

if (confirm == JOptionPane.YES\_OPTION) {

try (PreparedStatement ps = conn.prepareStatement("DELETE FROM Medicines WHERE medicine\_id = ?")) {

ps.setInt(1, id);

int affected = ps.executeUpdate();

if (affected == 0) {

JOptionPane.showMessageDialog(this, "No Medicine found with that ID.");

} else {

JOptionPane.showMessageDialog(this, "Medicine deleted.");

}

}

loadData();

}

} else if ("Suppliers".equals(table)) {

String idStr = JOptionPane.showInputDialog(this, "Enter the Supplier ID to delete:");

if (idStr == null) return;

int id = Integer.parseInt(idStr.trim());

int confirm = JOptionPane.showConfirmDialog(this,

"Are you sure you want to delete Supplier with ID " + id + "?",

"Confirm Delete",

JOptionPane.YES\_NO\_OPTION);

if (confirm == JOptionPane.YES\_OPTION) {

// First check if any medicines reference this supplier

try (PreparedStatement checkPs = conn.prepareStatement(

"SELECT COUNT(\*) FROM Medicines WHERE supplier\_id = ?")) {

checkPs.setInt(1, id);

try (ResultSet rs = checkPs.executeQuery()) {

if (rs.next() && rs.getInt(1) > 0) {

JOptionPane.showMessageDialog(this,

"Cannot delete supplier - there are medicines associated with this supplier.");

return;

}

}

}

try (PreparedStatement ps = conn.prepareStatement("DELETE FROM Suppliers WHERE supplier\_id = ?")) {

ps.setInt(1, id);

int affected = ps.executeUpdate();

if (affected == 0) {

JOptionPane.showMessageDialog(this, "No Supplier found with that ID.");

} else {

JOptionPane.showMessageDialog(this, "Supplier deleted.");

}

}

loadData();

}

}

} catch (NumberFormatException nfe) {

JOptionPane.showMessageDialog(this, "Invalid number format. Try again.");

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(this, "SQL Error: " + sqle.getMessage());

}

}

private void searchRecord() {

String table = (String) tableSelector.getSelectedItem();

outputArea.setText("");

if ("Medicines".equals(table)) {

String[] options = {"By Name", "By Batch No"};

int choice = JOptionPane.showOptionDialog(this,

"Search Medicine By:",

"Search Option",

JOptionPane.DEFAULT\_OPTION,

JOptionPane.QUESTION\_MESSAGE,

null,

options,

options[0]);

if (choice == -1) return; // User cancelled

String searchTerm = JOptionPane.showInputDialog(this, "Enter search term:");

if (searchTerm == null || searchTerm.trim().isEmpty()) return;

try (Connection conn = getConnection()) {

List<Medicine> results = new ArrayList<>();

if (choice == 0) { // By Name

try (PreparedStatement ps = conn.prepareStatement(

"SELECT \* FROM Medicines WHERE name LIKE ?")) {

ps.setString(1, "%" + searchTerm + "%");

try (ResultSet rs = ps.executeQuery()) {

while (rs.next()) {

results.add(new Medicine(

rs.getInt("medicine\_id"),

rs.getString("name"),

rs.getString("batch\_no"),

rs.getFloat("price"),

rs.getInt("quantity"),

rs.getInt("threshold"),

rs.getString("expiry\_date"),

rs.getInt("supplier\_id")

));

}

}

}

} else { // By Batch No

try (PreparedStatement ps = conn.prepareStatement(

"SELECT \* FROM Medicines WHERE batch\_no LIKE ?")) {

ps.setString(1, "%" + searchTerm + "%");

try (ResultSet rs = ps.executeQuery()) {

while (rs.next()) {

results.add(new Medicine(

rs.getInt("medicine\_id"),

rs.getString("name"),

rs.getString("batch\_no"),

rs.getFloat("price"),

rs.getInt("quantity"),

rs.getInt("threshold"),

rs.getString("expiry\_date"),

rs.getInt("supplier\_id")

));

}

}

}

}

outputArea.append(String.format("%-4s | %-20s | %-10s | %-7s | %-8s | %-8s | %-12s | %-10s\n",

"ID", "Name", "Batch No", "Price", "Quantity", "Threshold", "Expiry Date", "Supplier ID"));

outputArea.append("--------------------------------------------------------------------------------------------------------\n");

for (Medicine med : results) {

outputArea.append(med + "\n");

}

if (results.isEmpty()) {

outputArea.append("No matching medicines found.");

}

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(this, "SQL Error: " + sqle.getMessage());

}

} else if ("Suppliers".equals(table)) {

String searchTerm = JOptionPane.showInputDialog(this, "Enter supplier name to search:");

if (searchTerm == null || searchTerm.trim().isEmpty()) return;

try (Connection conn = getConnection()) {

List<Supplier> results = new ArrayList<>();

try (PreparedStatement ps = conn.prepareStatement(

"SELECT \* FROM Suppliers WHERE name LIKE ?")) {

ps.setString(1, "%" + searchTerm + "%");

try (ResultSet rs = ps.executeQuery()) {

while (rs.next()) {

results.add(new Supplier(

rs.getInt("supplier\_id"),

rs.getString("name"),

rs.getString("contact"),

rs.getString("email"),

rs.getString("address")

));

}

}

}

outputArea.append(String.format("%-4s | %-20s | %-15s | %-25s | %-30s\n",

"ID", "Name", "Contact", "Email", "Address"));

outputArea.append("--------------------------------------------------------------------------------------------------------\n");

for (Supplier s : results) {

outputArea.append(s + "\n");

}

if (results.isEmpty()) {

outputArea.append("No matching suppliers found.");

}

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(this, "SQL Error: " + sqle.getMessage());

}

}

}

private void showLowStockAlert() {

outputArea.setText("");

try (Connection conn = getConnection()) {

List<Medicine> lowStockMedicines = new ArrayList<>();

try (Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(

"SELECT \* FROM Medicines WHERE quantity <= threshold")) {

while (rs.next()) {

lowStockMedicines.add(new Medicine(

rs.getInt("medicine\_id"),

rs.getString("name"),

rs.getString("batch\_no"),

rs.getFloat("price"),

rs.getInt("quantity"),

rs.getInt("threshold"),

rs.getString("expiry\_date"),

rs.getInt("supplier\_id")

));

}

}

outputArea.append("⚠️ LOW STOCK ALERT ⚠️\n");

outputArea.append("The following medicines are below or at their threshold levels:\n\n");

outputArea.append(String.format("%-4s | %-20s | %-10s | %-8s | %-8s\n",

"ID", "Name", "Batch No", "Quantity", "Threshold"));

outputArea.append("------------------------------------------------------------\n");

if (lowStockMedicines.isEmpty()) {

outputArea.append("No medicines are currently low in stock.\n");

} else {

for (Medicine med : lowStockMedicines) {

outputArea.append(String.format("%-4d | %-20s | %-10s | %-8d | %-8d\n",

med.getMedicineId(), med.getName(), med.getBatchNo(),

med.getQuantity(), med.getThreshold()));

}

}

// Check for expired medicines

outputArea.append("\n\n⚠️ EXPIRY ALERT ⚠️\n");

outputArea.append("The following medicines are expired or will expire soon:\n\n");

LocalDate today = LocalDate.now();

LocalDate soon = today.plusDays(30); // Next 30 days

List<Medicine> expiredMedicines = new ArrayList<>();

List<Medicine> expiringSoonMedicines = new ArrayList<>();

try (Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM Medicines")) {

while (rs.next()) {

Medicine med = new Medicine(

rs.getInt("medicine\_id"),

rs.getString("name"),

rs.getString("batch\_no"),

rs.getFloat("price"),

rs.getInt("quantity"),

rs.getInt("threshold"),

rs.getString("expiry\_date"),

rs.getInt("supplier\_id")

);

LocalDate expiryDate = LocalDate.parse(med.getExpiryDate());

if (expiryDate.isBefore(today)) {

expiredMedicines.add(med);

} else if (expiryDate.isBefore(soon) {

expiringSoonMedicines.add(med);

}

}

}

if (!expiredMedicines.isEmpty()) {

outputArea.append("EXPIRED MEDICINES:\n");

outputArea.append(String.format("%-4s | %-20s | %-10s | %-12s\n",

"ID", "Name", "Batch No", "Expiry Date"));

outputArea.append("------------------------------------------------\n");

for (Medicine med : expiredMedicines) {

outputArea.append(String.format("%-4d | %-20s | %-10s | %-12s\n",

med.getMedicineId(), med.getName(),

med.getBatchNo(), med.getExpiryDate()));

}

}

if (!expiringSoonMedicines.isEmpty()) {

outputArea.append("\nMEDICINES EXPIRING SOON (within 30 days):\n");

outputArea.append(String.format("%-4s | %-20s | %-10s | %-12s\n",

"ID", "Name", "Batch No", "Expiry Date"));

outputArea.append("------------------------------------------------\n");

for (Medicine med : expiringSoonMedicines) {

outputArea.append(String.format("%-4d | %-20s | %-10s | %-12s\n",

med.getMedicineId(), med.getName(),

med.getBatchNo(), med.getExpiryDate()));

}

}

if (expiredMedicines.isEmpty() && expiringSoonMedicines.isEmpty()) {

outputArea.append("No expired or soon-to-expire medicines found.\n");

}

} catch (Exception ex) {

ex.printStackTrace();

outputArea.setText("❌ Error checking stock alerts: " + ex.getMessage());

}

}

// Medicine class

static class Medicine {

private int medicine\_id;

private String name;

private String batch\_no;

private float price;

private int quantity;

private int threshold;

private String expiry\_date;

private int supplier\_id;

public Medicine(int medicine\_id, String name, String batch\_no, float price,

int quantity, int threshold, String expiry\_date, int supplier\_id) {

this.medicine\_id = medicine\_id;

this.name = name;

this.batch\_no = batch\_no;

this.price = price;

this.quantity = quantity;

this.threshold = threshold;

this.expiry\_date = expiry\_date;

this.supplier\_id = supplier\_id;

}

public int getMedicineId() { return medicine\_id; }

public String getName() { return name; }

public String getBatchNo() { return batch\_no; }

public float getPrice() { return price; }

public int getQuantity() { return quantity; }

public int getThreshold() { return threshold; }

public String getExpiryDate() { return expiry\_date; }

public int getSupplierId() { return supplier\_id; }

@Override

public String toString() {

return String.format("%-4d | %-20s | %-10s | $%-6.2f | %-8d | %-8d | %-12s | %-10d",

medicine\_id, name, batch\_no, price, quantity, threshold, expiry\_date, supplier\_id);

}

}

// Supplier class

static class Supplier {

private int supplier\_id;

private String name;

private String contact;

private String email;

private String address;

public Supplier(int supplier\_id, String name, String contact, String email, String address) {

this.supplier\_id = supplier\_id;

this.name = name;

this.contact = contact;

this.email = email;

this.address = address;

}

public int getSupplierId() { return supplier\_id; }

public String getName() { return name; }

public String getContact() { return contact; }

public String getEmail() { return email; }

public String getAddress() { return address; }

@Override

public String toString() {

return String.format("%-4d | %-20s | %-15s | %-25s | %-30s",

supplier\_id, name, contact, email, address);

}

}

public static void main(String[] args) {

// Create database tables if they don't exist

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/", "root", "");

Statement stmt = conn.createStatement()) {

// Create database if not exists

stmt.execute("CREATE DATABASE IF NOT EXISTS MedicalStore");

stmt.execute("USE MedicalStore");

// Create Medicines table

stmt.execute("CREATE TABLE IF NOT EXISTS Medicines (" +

"medicine\_id INT PRIMARY KEY, " +

"name VARCHAR(100) NOT NULL, " +

"batch\_no VARCHAR(50) NOT NULL, " +

"price DECIMAL(10,2) NOT NULL, " +

"quantity INT NOT NULL, " +

"threshold INT NOT NULL, " +

"expiry\_date DATE NOT NULL, " +

"supplier\_id INT)");

// Create Suppliers table

stmt.execute("CREATE TABLE IF NOT EXISTS Suppliers (" +

"supplier\_id INT PRIMARY KEY, " +

"name VARCHAR(100) NOT NULL, " +

"contact VARCHAR(20), " +

"email VARCHAR(100), " +

"address VARCHAR(200))");

// Add some sample data if tables are empty

ResultSet rs = stmt.executeQuery("SELECT COUNT(\*) FROM Medicines");

rs.next();

if (rs.getInt(1) == 0) {

stmt.execute("INSERT INTO Medicines VALUES " +

"(1, 'Paracetamol', 'BATCH001', 5.50, 50, 10, '2024-12-31', 1), " +

"(2, 'Ibuprofen', 'BATCH002', 7.25, 15, 5, '2025-06-30', 1), " +

"(3, 'Amoxicillin', 'BATCH003', 12.75, 8, 5, '2024-09-15', 2), " +

"(4, 'Cetirizine', 'BATCH004', 4.20, 30, 10, '2025-03-31', null)");

}

rs = stmt.executeQuery("SELECT COUNT(\*) FROM Suppliers");

rs.next();

if (rs.getInt(1) == 0) {

stmt.execute("INSERT INTO Suppliers VALUES " +

"(1, 'Pharma Distributors', '9876543210', 'pharma@example.com', '123 Pharma St, Mumbai'), " +

"(2, 'MediCorp', '8765432109', 'medicorp@example.com', '456 Health Ave, Delhi')");

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "Error initializing database: " + e.getMessage(),

"Database Error", JOptionPane.ERROR\_MESSAGE);

}

SwingUtilities.invokeLater(MedicalStoreApp::new);

}

}

**Chapter 4**

**Mini Project 2**

**4.1 Problem Statement**

To build a web application using JSP and Servlets that enables user authentication through a login system and performs basic arithmetic operations via a calculator interface. The application should validate user credentials, process arithmetic inputs, and display results dynamically, demonstrating core concepts of web development with JSP, Servlets, and servlet configuration.

4.2 Implementation   
  
📦 Maven-WebApp

├── 📄 pom.xml

└── 📂 src

└── 📂 main

├── 📂 java

│ └── 📂 com

│ └── 📂 example

│ ├── 📄 AuthServlet.java

│ └── 📄 CalculatorServlet.java

├── 📂 resources

└── 📂 webapp

├── 📂 WEB-INF

│ └── 📄 web.xml

├── 📄 calculator.jsp

├── 📄 error.jsp

├── 📄 index.jsp

Maven WebApp Project - Complete Code and File Explanation

1. **pom.xml**

This is the Maven configuration file that manages project dependencies, plugins, and build lifecycle.

<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.example</groupId>  
 <artifactId>Maven-WebApp</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>war</packaging>  
  
 <dependencies>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.1</version>  
 <scope>provided</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

1. **AuthServlet.java**

Handles user authentication. If credentials are valid, redirects to calculator.jsp; otherwise, to error.jsp.

package com.example;

import javax.servlet.\*;  
import javax.servlet.http.\*;  
import java.io.IOException;  
  
public class AuthServlet extends HttpServlet {  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 String username = request.getParameter("username");  
 String password = request.getParameter("password");  
  
 if ("admin".equals(username) && "password".equals(password)) {  
 RequestDispatcher dispatcher = request.getRequestDispatcher("calculator.jsp");  
 dispatcher.forward(request, response);  
 } else {  
 RequestDispatcher dispatcher = request.getRequestDispatcher("error.jsp");  
 dispatcher.forward(request, response);  
 }  
 }  
}

1. **CalculatorServlet.java**

Performs arithmetic calculations (add, subtract, multiply, divide) and returns results to calculator.jsp.

package com.example;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import java.io.IOException;  
  
public class CalculatorServlet extends HttpServlet {  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 try {  
 int num1 = Integer.parseInt(request.getParameter("num1"));  
 int num2 = Integer.parseInt(request.getParameter("num2"));  
 String operation = request.getParameter("operation");  
 int result = 0;  
  
 switch (operation) {  
 case "add":  
 result = num1 + num2;  
 break;  
 case "subtract":  
 result = num1 - num2;  
 break;  
 case "multiply":  
 result = num1 \* num2;  
 break;  
 case "divide":  
 result = num1 / num2;  
 break;  
 }  
  
 request.setAttribute("result", result);  
 RequestDispatcher dispatcher = request.getRequestDispatcher("calculator.jsp");  
 dispatcher.forward(request, response);  
 } catch (Exception e) {  
 RequestDispatcher dispatcher = request.getRequestDispatcher("error.jsp");  
 dispatcher.forward(request, response);  
 }  
 }  
}

1. **web.xml**

The deployment descriptor that maps servlets to URL patterns.

<web-app xmlns="http://java.sun.com/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee   
 http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  
 version="3.0">  
  
 <servlet>  
 <servlet-name>AuthServlet</servlet-name>  
 <servlet-class>com.example.AuthServlet</servlet-class>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>AuthServlet</servlet-name>  
 <url-pattern>/login</url-pattern>  
 </servlet-mapping>  
  
 <servlet>  
 <servlet-name>CalculatorServlet</servlet-name>  
 <servlet-class>com.example.CalculatorServlet</servlet-class>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>CalculatorServlet</servlet-name>  
 <url-pattern>/calculate</url-pattern>  
 </servlet-mapping>  
  
</web-app>

1. **index.jsp**

Login page with form fields for username and password. Submits to AuthServlet.

<html>  
<head><title>Login</title></head>  
<body>  
 <h2>Login</h2>  
 <form action="login" method="post">  
 Username: <input type="text" name="username"><br>  
 Password: <input type="password" name="password"><br>  
 <input type="submit" value="Login">  
 </form>  
</body>  
</html>

1. **calculator.jsp**

Calculator interface to input two numbers and select an operation. Shows the result.

<html>  
<head><title>Calculator</title></head>  
<body>  
 <h2>Calculator</h2>  
 <form action="calculate" method="post">  
 Number 1: <input type="text" name="num1"><br>  
 Number 2: <input type="text" name="num2"><br>  
 Operation:   
 <select name="operation">  
 <option value="add">Add</option>  
 <option value="subtract">Subtract</option>  
 <option value="multiply">Multiply</option>  
 <option value="divide">Divide</option>  
 </select><br>  
 <input type="submit" value="Calculate">  
 </form>  
  
 <%  
 if (request.getAttribute("result") != null) {  
 out.println("<h3>Result: " + request.getAttribute("result") + "</h3>");  
 }  
 %>  
</body>  
</html>

1. **error.jsp**

Displays error message when authentication fails or invalid input is given.

<html>  
<head><title>Error</title></head>  
<body>  
 <h2>Error</h2>  
 <p>Invalid input or credentials. Please try again.</p>  
</body>  
</html>

**Chapter 5**

**Output Screenshots**

**Project 1 Screenshots**

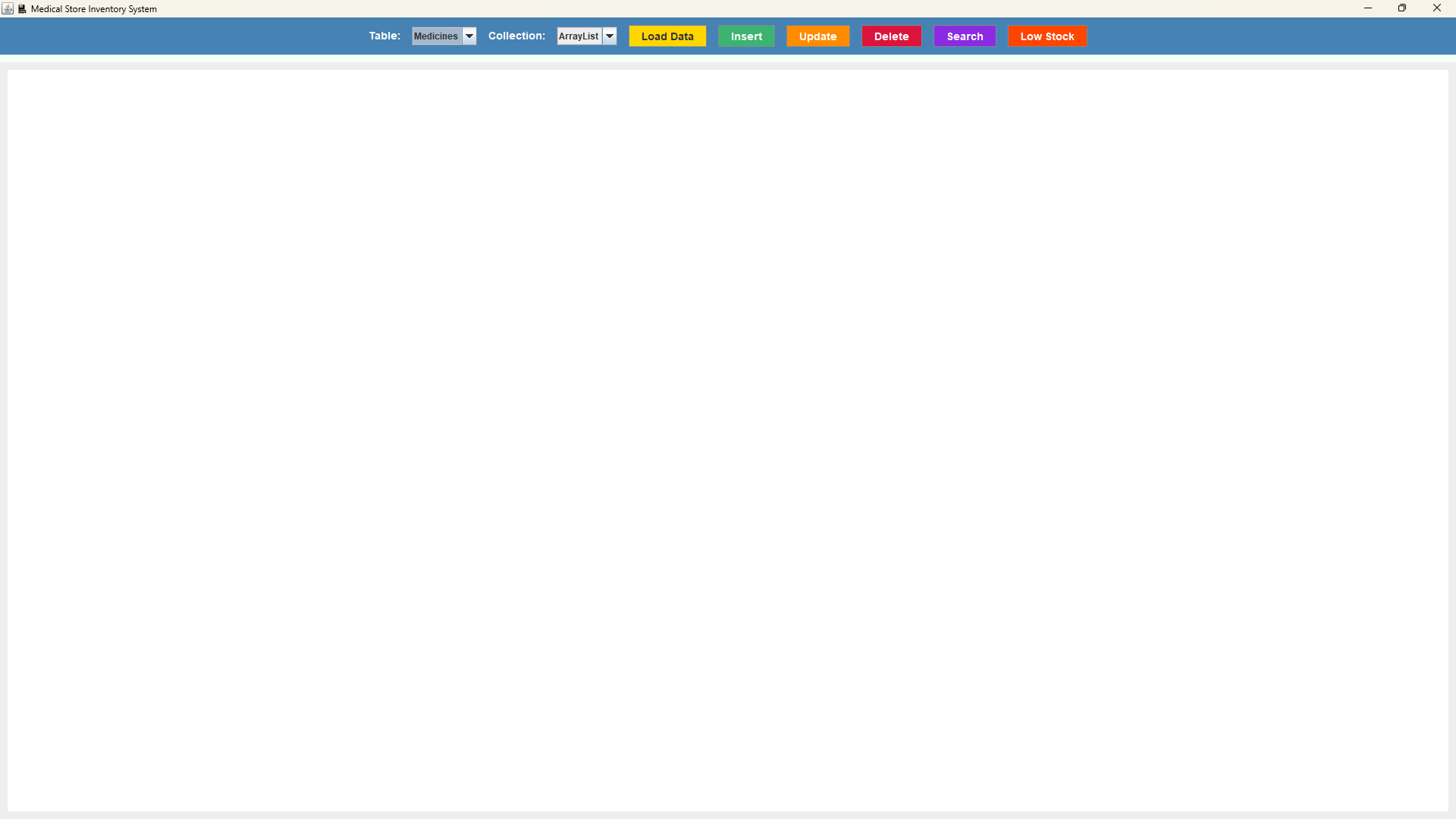


Fig.1 Home page

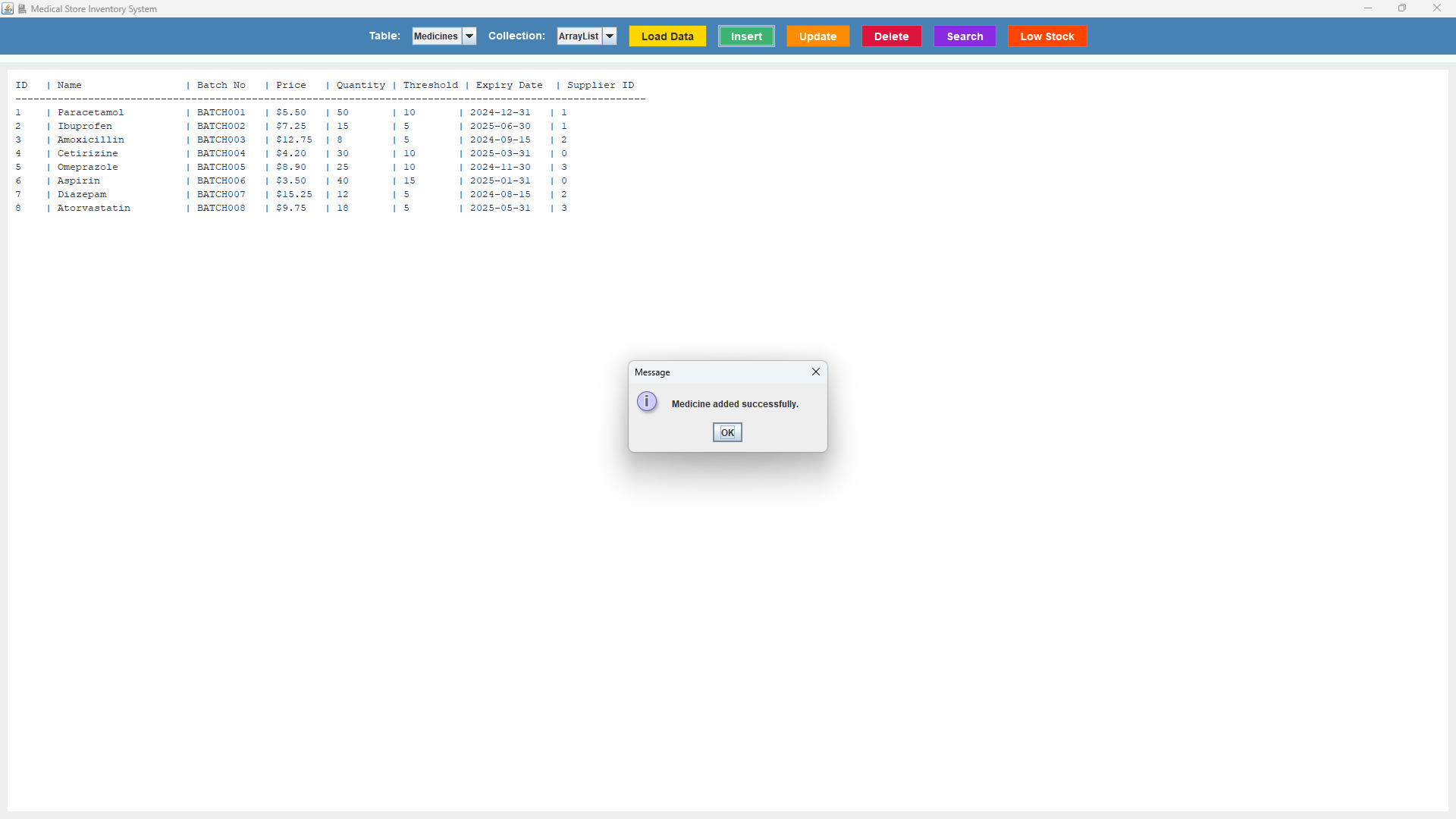


Fig.2 Inserting Data

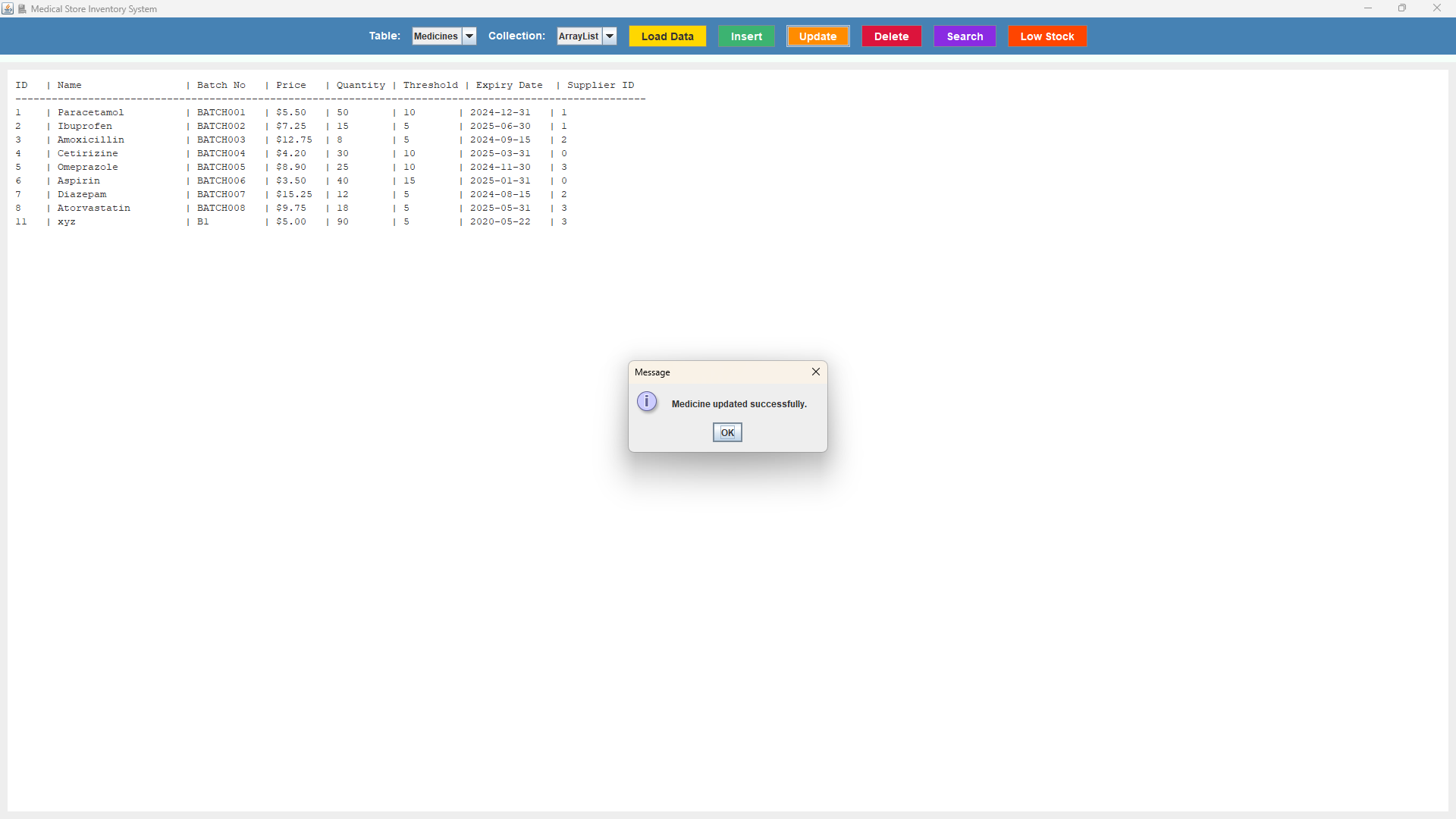


Fig.3 Updating Data

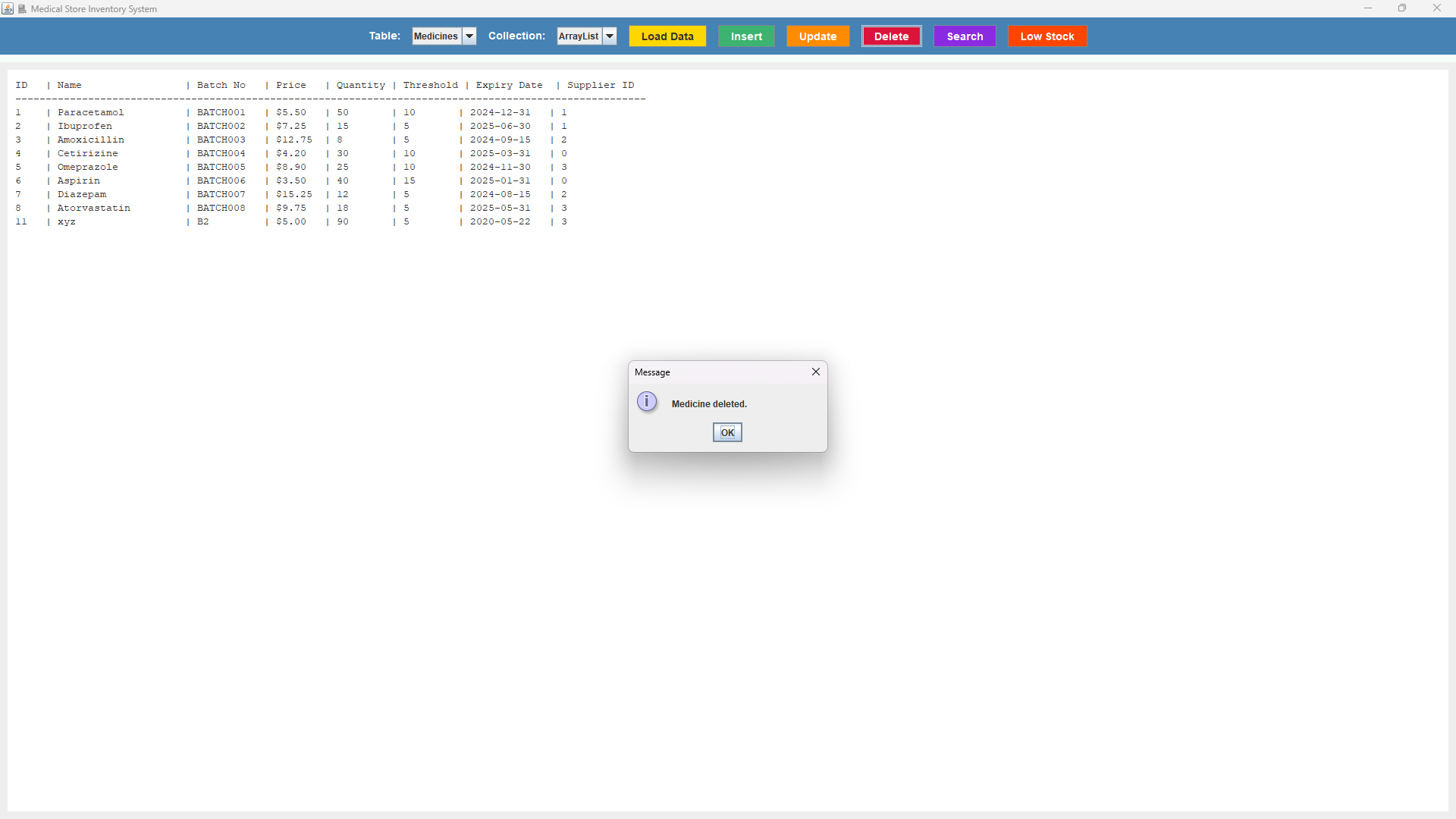


Fig.4 Deleting Data

Project 2 Screenshots

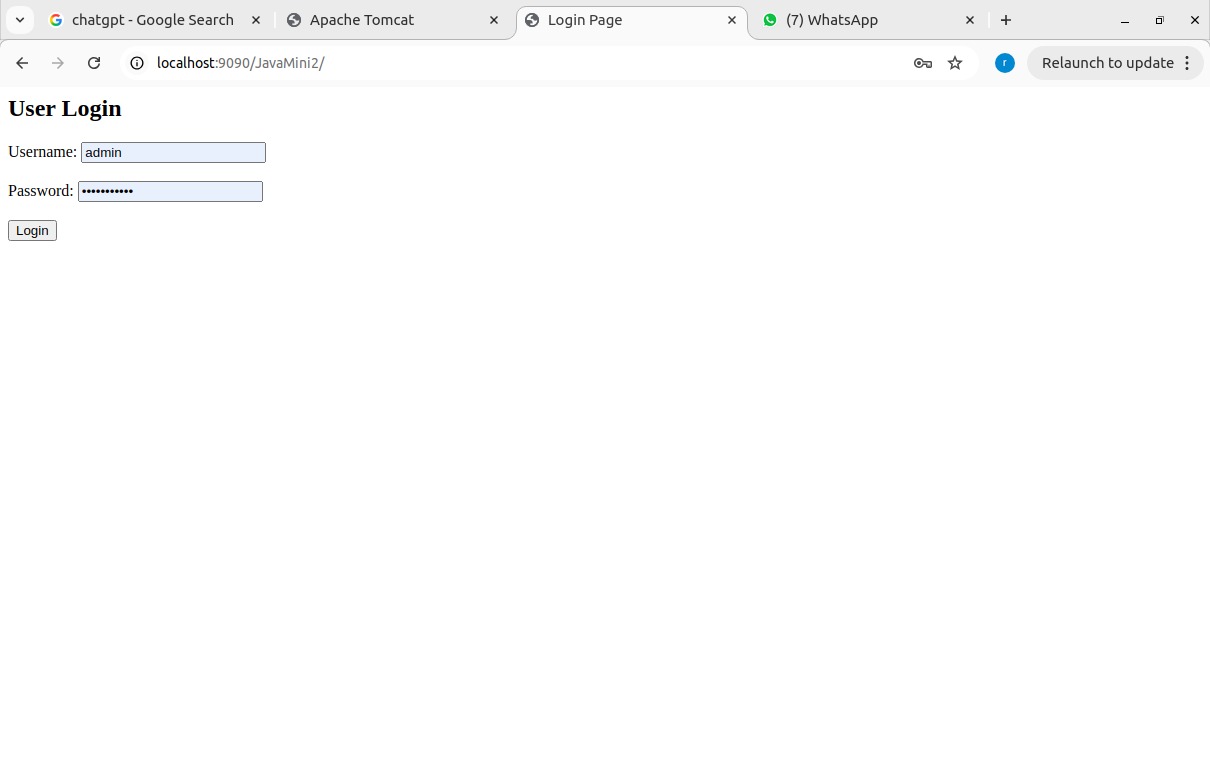


Fig.4 Login Page

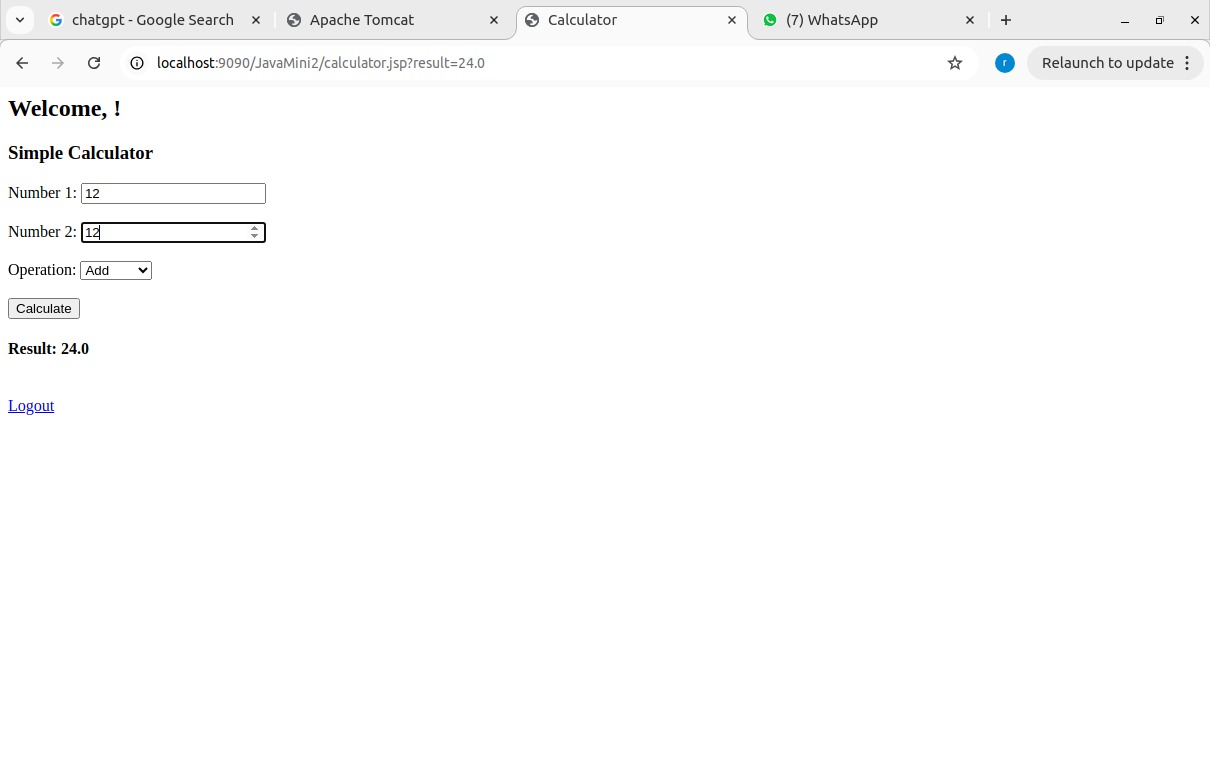


Fig.5 Simple Calculator

**Chapter 6**

**Conclusion**

This document presents two distinct Java-based applications that demonstrate core programming and software engineering concepts using both desktop and web technologies.

The **Medical Store Inventory System** is a Java Swing-based desktop application integrated with MySQL using JDBC. It offers complete CRUD functionality to manage medicines and suppliers while utilizing Java collections like ArrayList and TreeSet for in-memory data handling. Features like low stock and expiry alerts make the system practical and efficient for small medical businesses. This project helped reinforce concepts of GUI development, event handling, data management, and modular code structure.

The **JSP & Servlet-Based Web Calculator** demonstrates the use of Java EE technologies to build a functional web application. The system includes login authentication and a basic calculator interface using JSPs and Servlets. It follows a standard Maven directory structure and uses a deployment descriptor (web.xml) for configuration. The project introduced web development fundamentals such as session handling, request processing, and error redirection in a server-client model. It serves as a strong foundation for building more complex enterprise web applications.

Together, both projects illustrate the breadth of Java as a language—from desktop application development to dynamic web solutions—while reinforcing good software design practices

**Chapter 7**

**References**

[1] Oracle, “Java™ Platform, Standard Edition Documentation,” [Online]. Available: <https://docs.oracle.com/javase/8/docs/>.  
[2] Oracle, “MySQL 8.0 Reference Manual,” [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en/>.  
[3] Oracle, “Java Swing Tutorial,” [Online]. Available: <https://docs.oracle.com/javase/tutorial/uiswing/>.  
[4] GeeksforGeeks, “JDBC in Java,” [Online]. Available: https://www.geeksforgeeks.org/jdbc-in-java/.  
[5] Oracle, “Java EE 7 Tutorial,” [Online]. Available: <https://docs.oracle.com/javaee/7/tutorial/>.  
[6] Eclipse Foundation, “Jakarta Servlet Specification,” [Online]. Available: https://jakarta.ee/specifications/servlet/.  
[7] w3schools, “JSP Tutorial,” [Online]. Available: https://www.w3schools.com/jsp/.  
[8] Apache Maven, “Maven Documentation,” [Online]. Available: <https://maven.apache.org/guides/index.html>.