ALPHA STATIONARY SHOP INVENTORY MANAGEMNT SYSTEM

A MINI PROJECT

SUBMITTED BY

LOGAPRIYA S G - 230701164

MADHUMITHA B M - 230701168

In partial fulfilment for the award of the Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI- 602105

2024-25

BONAFIDE CERTIFIACTE

Certified that this project report "**Alpha Stationary Shop INVENTORY MANAGEMENT SYSTEM"** is a Bonafide work of "**B M MADHUMITHA (230701168) & S G LOGAPRIYA (230701164)"** who carried out the project work under the Supervision of **Mrs.B.S.Dharshini- Assistant Professor (SG)**

Submitted for the Practical Examination held on ___________

Signature of

Mrs.B.S.Dharshini

Assistant Professor (SG)

Computer Science and Engineering

Rajalakshmi Engineering College

Thandalam, Chennai-602 105

ABSTRACT

The Alpha Stationary Shop Inventory Management System is a robust solution aimed at optimizing and automating the shop's inventory management processes through an advanced and user-friendly interface. Developed using JavaFX for a modern, intuitive graphical user experience and supported by PL/SQL for efficient database operations, the system effectively addresses the challenges of traditional manual inventory handling.

With JavaFX as the primary frontend technology, the application delivers a responsive, interactive, and visually appealing user interface. It provides a seamless experience for both administrators and customers, allowing for intuitive navigation and efficient task execution. Key components of the interface include real-time data visualization for inventory tracking, dynamic tables for product listings, and interactive forms for user input and management actions.

The backend, powered by PL/SQL, ensures secure and efficient data handling, supporting core functionalities such as inventory tracking, supplier management, customer data storage, and automated invoice generation. The integration between JavaFX and the PL/SQL database facilitates real-time updates, ensuring accurate monitoring of stock levels and enabling timely replenishment, which reduces the risks of overstocking and stockouts.

This scalable and modular system design ensures adaptability to future business needs and technological advancements, making it a reliable and future-proof investment for the shop. Overall, the Alpha Stationary Shop Inventory Management System represents a significant shift from manual processes to an automated, data-driven, and user-focused approach, leveraging the strengths of JavaFX for frontend development and PL/SQL for robust backend management.

<u>**TABLE OF CONTENTS**</u>

INTRODUCTION

INTRODUCTION:

The project provides a comprehensive platform for handling various aspects of shop management, including inventory tracking, customer and supplier management, sales processing, and invoicing. The user-friendly interface, developed using JavaFX, delivers a responsive and visually appealing experience, simplifying complex tasks for both shop administrators and customers. By integrating PL/SQL, the system ensures efficient data handling, accurate record-keeping, and robust transaction processing.

IMPLEMENTTATION:

This project uses JAVA FX for building the frontend phase and PL SQL to handle the backend and databases.

WEBSITE FEATURE:

A responsive website is created using features of JAVA FX with an active home page.

The home page offers options like Add items, Add customers, Add suppliers, Add invoices and also the options to display the same details.

A responsive corresponding pages are linked with each options.

SYSTEM SPECIFICATION

1. Hardware Requirements

- Processor: Intel Core i5 (10th Gen or higher) / AMD Ryzen 5 or equivalent
- RAM: Minimum 8 GB (16 GB recommended for smoother development)
- Storage: 256 GB SSD (Solid State Drive) or higher.

2. Software Requirements

- Frontend: Java FX
- Backend: Pl Sql
- Operating System: Windows 10

SAMPLE CODE

Home Page:

```
package project;


import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TableCell;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.image.Image;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.stage.Stage;


import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
public class App extends Application {


    @Override
    public void start(Stage primaryStage) {
        // Creating buttons for the actions
        Button addItemButton = new Button("Add Item");
        Button addSupplierButton = new Button("Add Supplier");
        Button addCustomerButton = new Button("Add Customer");
        Button addInvoiceButton = new Button("Create Invoice");
        Button addSupplyInvoiceButton = new Button("Add Supply Invoice");
        Button addItemsToInvoiceButton = new Button("Add Items to Invoice");
        Button viewSuppliersButton = new Button("View Suppliers");
        Button viewItemsButton = new Button("View Items");
        Button viewCustomersButton = new Button("View Customers");
        Button viewInvoicesButton = new Button("View Invoices");


        // Style buttons to make the text fully visible
        String buttonStyle = "-fx-font-size: 14px; -fx-padding: 10px 20px; -fx-
background-color: #555555; -fx-text-fill: white; -fx-border-radius: 5px; -fx-
background-radius: 5px; -fx-pref-width: 200px;";
        addItemButton.setStyle(buttonStyle);
        addSupplierButton.setStyle(buttonStyle);
        addCustomerButton.setStyle(buttonStyle);
        addInvoiceButton.setStyle(buttonStyle);
        addSupplyInvoiceButton.setStyle(buttonStyle);
        addItemsToInvoiceButton.setStyle(buttonStyle);
        viewSuppliersButton.setStyle(buttonStyle);
        viewItemsButton.setStyle(buttonStyle);
        viewCustomersButton.setStyle(buttonStyle);
        viewInvoicesButton.setStyle(buttonStyle);

        // Set button actions
        addItemButton.setOnAction(e -> showAddItemPage());
        addSupplierButton.setOnAction(e -> showAddSupplierPage());
        addCustomerButton.setOnAction(e -> showAddCustomerPage());
        addInvoiceButton.setOnAction(e -> showAddInvoicePage());
        addSupplyInvoiceButton.setOnAction(e -> showAddSupplyInvoicePage());
        addItemsToInvoiceButton.setOnAction(e ->
    showAddItemsToInvoicePage());
        viewSuppliersButton.setOnAction(e -> displaySuppliers(primaryStage));
        viewItemsButton.setOnAction(e -> displayItems(primaryStage));
        viewCustomersButton.setOnAction(e -> displayCustomers(primaryStage));
        viewInvoicesButton.setOnAction(e -> displayInvoices(primaryStage));

        // Create VBox layout for buttons
        VBox leftColumn = new VBox(20);  // 20px spacing between buttons in the
    left column
        leftColumn.setAlignment(Pos.CENTER_LEFT);

        VBox rightColumn = new VBox(20);  // 20px spacing between buttons in
    the right column
        rightColumn.setAlignment(Pos.CENTER_LEFT);
        VBox centerColumn = new VBox(20);  // 20px spacing between buttons in
the center
        centerColumn.setAlignment(Pos.CENTER);

        // Add buttons to the left, right, and center columns
        leftColumn.getChildren().addAll(
            addItemButton, addSupplierButton, addCustomerButton,
addInvoiceButton,
            addSupplyInvoiceButton
        );

        rightColumn.getChildren().addAll(
            addItemsToInvoiceButton, viewSuppliersButton, viewItemsButton,
            viewCustomersButton, viewInvoicesButton
        );

        centerColumn.getChildren().addAll(
            // If you want to center specific buttons, you can add them here
        );

        // Create HBox layout to hold left, right, and center columns
        HBox hBox = new HBox(40); // 40px spacing between the columns
        hBox.setAlignment(Pos.CENTER);
        hBox.getChildren().addAll(leftColumn, centerColumn, rightColumn);

        // Create main layout with background image
        VBox mainLayout = new VBox();
        mainLayout.getChildren().add(hBox);


        // Set the background image
        BackgroundImage background = new BackgroundImage(
            new
Image(getClass().getResource("/project/Background/login.png").toExternalFor
m(), 800, 600, false, true),
            BackgroundRepeat.NO_REPEAT, BackgroundRepeat.NO_REPEAT,
            BackgroundPosition.CENTER, BackgroundSize.DEFAULT
        );
        mainLayout.setBackground(new Background(background));


        // Set up the scene and stage
        Scene scene = new Scene(mainLayout, 800, 600);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Inventory Management System");
        primaryStage.show();
    }


    // Navigation methods for different pages
    private void showAddItemPage() {
        AddItemPage itemPage = new AddItemPage();
        Stage itemStage = new Stage();
        itemPage.start(itemStage);
    }


    private void showAddSupplierPage() {
        AddSupplierPage supplierPage = new AddSupplierPage();
        Stage supplierStage = new Stage();
        supplierPage.start(supplierStage);
    }

    private void showAddCustomerPage() {
        AddCustomerPage customerPage = new AddCustomerPage();
        Stage customerStage = new Stage();
        customerPage.start(customerStage);
    }

    private void showAddInvoicePage() {
        AddInvoicePage invoicePage = new AddInvoicePage();
        Stage invoiceStage = new Stage();
        invoicePage.start(invoiceStage);
    }

    private void showAddSupplyInvoicePage() {
        AddSupplyInvoicePage supplyInvoicePage = new AddSupplyInvoicePage();
        Stage supplyInvoiceStage = new Stage();
        supplyInvoicePage.start(supplyInvoiceStage);
    }

    private void showAddItemsToInvoicePage() {
        AddItemsToInvoicePage itemsToInvoicePage = new
AddItemsToInvoicePage();
        Stage itemsToInvoiceStage = new Stage();
        itemsToInvoicePage.start(itemsToInvoiceStage);
    }

    // Go back to the home page
    private void goBackToHomePage(Stage primaryStage) {
        start(primaryStage); // This will navigate back to the home page by calling
the start method
    }

    // Methods to display items, suppliers, customers, and invoices
    @SuppressWarnings("unchecked")
    public void displayItems(Stage primaryStage) {
        String query = "SELECT * FROM items";
        ObservableList<Item> items = FXCollections.observableArrayList();
        dbconnect d = new dbconnect();

        try (Connection conn = d.connect();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {

            while (rs.next()) {
                items.add(new Item(
                        rs.getInt("item_id"),
                        rs.getString("item_name"),
                        rs.getString("description"),
                        rs.getInt("quantity"),
                        rs.getInt("reorder_level"),
                    rs.getBigDecimal("unit_price"),
                    rs.getBigDecimal("total_amount"),
                    rs.getDate("purchase_date")
                ));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Create a TableView
        TableView<Item> tableView = new TableView<>(items);
        TableColumn<Item, Integer> idColumn = new TableColumn<>("Item ID");
        TableColumn<Item, String> nameColumn = new TableColumn<>("Item
    Name");
        TableColumn<Item, String> descColumn = new
    TableColumn<>("Description");
        TableColumn<Item, Integer> quantityColumn = new
    TableColumn<>("Quantity");
        TableColumn<Item, Integer> reorderColumn = new
    TableColumn<>("Reorder Level");
        TableColumn<Item, String> priceColumn = new TableColumn<>("Unit
    Price");

        idColumn.setCellValueFactory(new PropertyValueFactory<>("itemId"));
        nameColumn.setCellValueFactory(new
    PropertyValueFactory<>("itemName"));
        descColumn.setCellValueFactory(new
    PropertyValueFactory<>("description"));
```

```
        quantityColumn.setCellValueFactory(new
PropertyValueFactory<>("quantity"));
        reorderColumn.setCellValueFactory(new
PropertyValueFactory<>("reorderLevel"));
        priceColumn.setCellValueFactory(new
PropertyValueFactory<>("unitPrice"));

        tableView.getColumns().addAll(idColumn, nameColumn, descColumn,
quantityColumn, reorderColumn, priceColumn);

        // Create "Back" button to return to the home page
        Button backButton = new Button("Back to Home Page");
        backButton.setStyle("-fx-font-size: 14px; -fx-padding: 10px 20px; -fx-
background-color: #555555; -fx-text-fill: white;");
        backButton.setOnAction(e -> goBackToHomePage(primaryStage));

        // Create a VBox to hold the TableView and Back button
        VBox layout = new VBox(20, tableView, backButton);
        Scene scene = new Scene(layout, 800, 600);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Items List");
        primaryStage.show();
    }

    public void displaySuppliers(Stage primaryStage) {
        String query = "SELECT * FROM supplier";
        ObservableList<Supplier> suppliers = FXCollections.observableArrayList();
        dbconnect d = new dbconnect();
    try (Connection conn = d.connect();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(query)) {

        while (rs.next()) {
            suppliers.add(new Supplier(
                    rs.getInt("supplier_id"),
                    rs.getString("supplier_name"),
                    rs.getString("contact_no"),
                    rs.getString("email"),
                    rs.getString("address")
            ));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    // Create TableView for Suppliers
    TableView<Supplier> tableView = new TableView<>(suppliers);
    TableColumn<Supplier, Integer> idColumn = new
TableColumn<>("Supplier ID");
    TableColumn<Supplier, String> nameColumn = new
TableColumn<>("Supplier Name");
    TableColumn<Supplier, String> contactColumn = new
TableColumn<>("Contact No");
```

```
        TableColumn<Supplier, String> emailColumn = new
TableColumn<>("Email");
        TableColumn<Supplier, String> addressColumn = new
TableColumn<>("Address");

        idColumn.setCellValueFactory(new PropertyValueFactory<>("supplierId"));
        nameColumn.setCellValueFactory(new
PropertyValueFactory<>("supplierName"));
        contactColumn.setCellValueFactory(new
PropertyValueFactory<>("contactNo"));
        emailColumn.setCellValueFactory(new PropertyValueFactory<>("email"));
        addressColumn.setCellValueFactory(new
PropertyValueFactory<>("address"));

        tableView.getColumns().addAll(idColumn, nameColumn, contactColumn,
emailColumn, addressColumn);

        // Create the "Back" button
        Button backButton = new Button("Back to Home Page");
        backButton.setStyle("-fx-font-size: 14px; -fx-padding: 10px 20px; -fx-
background-color: #555555; -fx-text-fill: white;");
        backButton.setOnAction(e -> goBackToHomePage(primaryStage));

        // Create the layout and set the scene
        VBox layout = new VBox(20, tableView, backButton);
        Scene scene = new Scene(layout, 800, 600);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Suppliers List");
        primaryStage.show();
    }

    public void displayCustomers(Stage primaryStage) {
        String query = "SELECT * FROM customer";  // Modify according to your
database schema
        ObservableList<Customer> customers =
FXCollections.observableArrayList();
        dbconnect d = new dbconnect();

        try (Connection conn = d.connect();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {

            while (rs.next()) {
                customers.add(new Customer(
                        rs.getInt("customer_id"),
                        rs.getString("customer_name"),
                        rs.getString("contact_no"),
                        rs.getString("email"),
                        rs.getString("address"), 0,
                        rs.getDate("last_purchase_date")
                ));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        // Create TableView for Customers
        TableView<Customer> tableView = new TableView<>(customers);
        TableColumn<Customer, Integer> idColumn = new
TableColumn<>("Customer ID");
        TableColumn<Customer, String> nameColumn = new
TableColumn<>("Customer Name");
        TableColumn<Customer, String> contactColumn = new
TableColumn<>("Contact No");
        TableColumn<Customer, String> emailColumn = new
TableColumn<>("Email");
        TableColumn<Customer, String> addressColumn = new
TableColumn<>("Address");
        TableColumn<Customer, Date> dateColumn = new TableColumn<>("Last
Purchase Date");

        idColumn.setCellValueFactory(new
PropertyValueFactory<>("customerId"));
        nameColumn.setCellValueFactory(new
PropertyValueFactory<>("customerName"));
        contactColumn.setCellValueFactory(new
PropertyValueFactory<>("contactNo"));
        emailColumn.setCellValueFactory(new PropertyValueFactory<>("email"));
        addressColumn.setCellValueFactory(new
PropertyValueFactory<>("address"));
        dateColumn.setCellValueFactory(new
PropertyValueFactory<>("lastPurchaseDate"));

        tableView.getColumns().addAll(idColumn, nameColumn, contactColumn,
emailColumn, addressColumn,dateColumn);
```

```
        // Create the "Back" button
        Button backButton = new Button("Back to Home Page");
        backButton.setStyle("-fx-font-size: 14px; -fx-padding: 10px 20px; -fx-
background-color: #555555; -fx-text-fill: white;");
        backButton.setOnAction(e -> goBackToHomePage(primaryStage));

        // Create the layout and set the scene
        VBox layout = new VBox(20, tableView, backButton);
        Scene scene = new Scene(layout, 800, 600);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Customers List");
        primaryStage.show();
    }

    public void displayInvoices(Stage primaryStage) {
        String query = "SELECT * FROM invoice";  // Modify according to your
database schema
        ObservableList<Invoice> invoices = FXCollections.observableArrayList();
        dbconnect d = new dbconnect();

        try (Connection conn = d.connect();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {

            while (rs.next()) {
                invoices.add(new Invoice(
                    rs.getInt("invoice_id"),
                    rs.getInt("customer_id"),
                    rs.getBigDecimal("bill_amount"),
                    rs.getDate("invoice_date")
                ));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Create TableView for Invoices
        TableView<Invoice> tableView = new TableView<>(invoices);
        TableColumn<Invoice, Integer> idColumn = new TableColumn<>("Invoice
    ID");
        TableColumn<Invoice, Integer> customerIdColumn = new
    TableColumn<>("Customer ID");
        TableColumn<Invoice, String> dateColumn = new TableColumn<>("Invoice
    Date");
        TableColumn<Invoice, String> totalAmountColumn = new
    TableColumn<>("Bill Amount");

        idColumn.setCellValueFactory(new PropertyValueFactory<>("invoiceId"));
        customerIdColumn.setCellValueFactory(new
    PropertyValueFactory<>("customerId"));
        dateColumn.setCellValueFactory(new
    PropertyValueFactory<>("invoiceDate"));
        totalAmountColumn.setCellValueFactory(new
    PropertyValueFactory<>("totalAmount"));
```

```
        tableView.getColumns().addAll(idColumn, customerIdColumn,
dateColumn, totalAmountColumn);

        // Create the "Back" button
        Button backButton = new Button("Back to Home Page");
        backButton.setStyle("-fx-font-size: 14px; -fx-padding: 10px 20px; -fx-
background-color: #555555; -fx-text-fill: white;");
        backButton.setOnAction(e -> goBackToHomePage(primaryStage));

        // Create the layout and set the scene
        VBox layout = new VBox(20, tableView, backButton);
        Scene scene = new Scene(layout, 800, 600);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Invoices List");
        primaryStage.show();
    }
 public static void main(String[] args) {
        launch(args);
    }
}
```

Add Items Page:

```
package project;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.Date;
import java.time.LocalDate;

public class AddItemPage extends Application {

    @SuppressWarnings("unused")
    @Override
    public void start(Stage primaryStage) {
        // Labels and TextFields
        Label itemNameLabel = new Label("Item Name:");
        TextField itemNameField = new TextField();

        Label descriptionLabel = new Label("Description:");
        TextField descriptionField = new TextField();

        Label quantityLabel = new Label("Quantity:");
        TextField quantityField = new TextField();

        Label reorderLevelLabel = new Label("Reorder Level:");
        TextField reorderLevelField = new TextField();
        Label unitPriceLabel = new Label("Unit Price:");
        TextField unitPriceField = new TextField();

        Label purchaseDateLabel = new Label("Purchase Date:");
        DatePicker purchaseDatePicker = new DatePicker(LocalDate.now());

        // Submit button
        Button submitButton = new Button("Add Item");

        submitButton.setOnAction(e -> {
            String itemName = itemNameField.getText();
            String description = descriptionField.getText();
            int quantity = Integer.parseInt(quantityField.getText());
            int reorderLevel = Integer.parseInt(reorderLevelField.getText());
            float unitPrice = Float.parseFloat(unitPriceField.getText());
            Date purchaseDate = Date.valueOf(purchaseDatePicker.getValue());

            // Create dbconnect object and insert the item
            dbconnect db = new dbconnect();
            Connection conn = db.connect();
            db.addNewItem(conn, itemName, description, quantity, reorderLevel,
    unitPrice, purchaseDate);
            primaryStage.close();
        });

        // Layout setup
        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);

        // Add components to grid
        grid.add(itemNameLabel, 0, 0);
        grid.add(itemNameField, 1, 0);
        grid.add(descriptionLabel, 0, 1);
        grid.add(descriptionField, 1, 1);
        grid.add(quantityLabel, 0, 2);
        grid.add(quantityField, 1, 2);
        grid.add(reorderLevelLabel, 0, 3);
        grid.add(reorderLevelField, 1, 3);
        grid.add(unitPriceLabel, 0, 4);
        grid.add(unitPriceField, 1, 4);
        grid.add(purchaseDateLabel, 0, 5);
        grid.add(purchaseDatePicker, 1, 5);
        grid.add(submitButton, 1, 6);

        // Scene setup
        Scene scene = new Scene(grid, 400, 350);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Add Item");
        primaryStage.show();
    }
```

```
}
```

<u>Add Customers Page:</u>

```
package project;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.Date;
import java.time.LocalDate;

public class AddCustomerPage extends Application {

    @SuppressWarnings("unused")
    @Override
    public void start(Stage primaryStage) {
        // Labels and TextFields
        Label customerNameLabel = new Label("Customer Name:");
        TextField customerNameField = new TextField();

        Label contactNoLabel = new Label("Contact No:");
        TextField contactNoField = new TextField();
        Label emailLabel = new Label("Email:");
        TextField emailField = new TextField();

        Label addressLabel = new Label("Address:");
        TextField addressField = new TextField();

        Label loyaltyPointsLabel = new Label("Loyalty Points:");
        TextField loyaltyPointsField = new TextField();

        Label lastPurchaseDateLabel = new Label("Last Purchase Date:");
        DatePicker lastPurchaseDatePicker = new DatePicker(LocalDate.now());

        // Submit button
        Button submitButton = new Button("Add Customer");

        submitButton.setOnAction(e -> {
            String customerName = customerNameField.getText();
            String contactNo = contactNoField.getText();
            String email = emailField.getText();
            String address = addressField.getText();
            int loyaltyPoints = Integer.parseInt(loyaltyPointsField.getText());
            Date lastPurchaseDate =
Date.valueOf(lastPurchaseDatePicker.getValue());

            // Create dbconnect object and insert the customer
            dbconnect db = new dbconnect();
            Connection conn = db.connect();
            db.addCustomer(conn, customerName, contactNo, email, address,
loyaltyPoints, lastPurchaseDate);
            primaryStage.close();
        });

        // Layout setup
        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);

        // Add components to grid
        grid.add(customerNameLabel, 0, 0);
        grid.add(customerNameField, 1, 0);
        grid.add(contactNoLabel, 0, 1);
        grid.add(contactNoField, 1, 1);
        grid.add(emailLabel, 0, 2);
        grid.add(emailField, 1, 2);
        grid.add(addressLabel, 0, 3);
        grid.add(addressField, 1, 3);
        grid.add(loyaltyPointsLabel, 0, 4);
        grid.add(loyaltyPointsField, 1, 4);
        grid.add(lastPurchaseDateLabel, 0, 5);
        grid.add(lastPurchaseDatePicker, 1, 5);
        grid.add(submitButton, 1, 6);
        // Scene setup
        Scene scene = new Scene(grid, 400, 350);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Add Customer");
        primaryStage.show();
    }
}
```

Add Invoice Page:

```
package project;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.time.LocalDate;

public class AddInvoicePage extends Application {

    @Override
```

```
public void start(Stage primaryStage) {
    // Labels and TextFields
    Label invoiceDateLabel = new Label("Invoice Date:");
    DatePicker invoiceDatePicker = new DatePicker(LocalDate.now());

    Label customerIdLabel = new Label("Customer ID:");
    TextField customerIdField = new TextField();

    Label totalAmountLabel = new Label("Total Amount:");
    TextField totalAmountField = new TextField();

    // Submit button
    Button submitButton = new Button("Create Invoice");

    submitButton.setOnAction(e -> {
        // Get form input values
        try {
            int customerId = Integer.parseInt(customerIdField.getText());
            float totalAmount = Float.parseFloat(totalAmountField.getText());
            Date invoiceDate = Date.valueOf(invoiceDatePicker.getValue());

            // Create dbconnect object and insert the invoice
            dbconnect db = new dbconnect();
            Connection conn = db.connect();

            // SQL query to insert the new invoice
            String sqlInsertInvoice = "INSERT INTO invoice (invoice_date,
customer_id, bill_amount) VALUES (?, ?, ?)";
            PreparedStatement stmtInsertInvoice =
conn.prepareStatement(sqlInsertInvoice);

            // Set parameters for the prepared statement
            stmtInsertInvoice.setDate(1, invoiceDate);  // Set invoice_date
            stmtInsertInvoice.setInt(2, customerId);    // Set customer_id
            stmtInsertInvoice.setFloat(3, totalAmount); // Set bill_amount

            // Execute the query to insert the invoice
            int rowsAffected = stmtInsertInvoice.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Invoice created successfully.");

                // Update the last_purchase_date in the customer table
                String sqlUpdateCustomer = "UPDATE customer SET
last_purchase_date = ? WHERE customer_id = ?";
                PreparedStatement stmtUpdateCustomer =
conn.prepareStatement(sqlUpdateCustomer);

                // Set parameters for the prepared statement
                stmtUpdateCustomer.setDate(1, invoiceDate);  // Set the invoice
date as the last purchase date
                stmtUpdateCustomer.setInt(2, customerId);    // Set customer_id

                // Execute the update query
                int updateRowsAffected = stmtUpdateCustomer.executeUpdate();
                if (updateRowsAffected > 0) {
                    System.out.println("Customer's last purchase date updated
successfully.");
                    primaryStage.close();
                } else {
                    System.out.println("Failed to update the customer's last purchase
date.");
                }

                // Close the window after success
                primaryStage.close();
            } else {
                System.out.println("Failed to create invoice.");
            }

            // Close the connection
            conn.close();
        } catch (SQLException sqlEx) {
            System.out.println("Database error: " + sqlEx.getMessage());
            sqlEx.printStackTrace();
        } catch (Exception ex) {
            System.out.println("Error: " + ex.getMessage());
            ex.printStackTrace();
        }
    });

    // Layout setup
```

```
        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);

        // Add components to grid
        grid.add(invoiceDateLabel, 0, 0);
        grid.add(invoiceDatePicker, 1, 0);

        grid.add(customerIdLabel, 0, 2);
        grid.add(customerIdField, 1, 2);

        grid.add(totalAmountLabel, 0, 5);
        grid.add(totalAmountField, 1, 5);
        grid.add(submitButton, 1, 6);

        // Scene setup
        Scene scene = new Scene(grid, 400, 350);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Create Invoice");
        primaryStage.show();
    }
}
```

Add Supplier Page:

```
package project;
```

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.Connection;

public class AddSupplierPage extends Application {

    @SuppressWarnings("unused")
    @Override
    public void start(Stage primaryStage) {
        // Labels and TextFields
        Label supplierNameLabel = new Label("Supplier Name:");
        TextField supplierNameField = new TextField();

        Label contactNoLabel = new Label("Contact No:");
        TextField contactNoField = new TextField();

        Label emailLabel = new Label("Email:");
        TextField emailField = new TextField();

        Label addressLabel = new Label("Address:");
```

```
TextField addressField = new TextField();

// Submit button
Button submitButton = new Button("Add Supplier");

submitButton.setOnAction(e -> {
   String supplierName = supplierNameField.getText();
   String contactNo = contactNoField.getText();
   String email = emailField.getText();
   String address = addressField.getText();

   // Create dbconnect object and insert the supplier
   dbconnect db = new dbconnect();
   Connection conn = db.connect();
   db.addSupplier(conn, supplierName, contactNo, email, address);
   primaryStage.close();
});

// Layout setup
GridPane grid = new GridPane();
grid.setPadding(new Insets(10, 10, 10, 10));
grid.setVgap(8);
grid.setHgap(10);

// Add components to grid
grid.add(supplierNameLabel, 0, 0);
```

```
        grid.add(supplierNameField, 1, 0);
        grid.add(contactNoLabel, 0, 1);
        grid.add(contactNoField, 1, 1);
        grid.add(emailLabel, 0, 2);
        grid.add(emailField, 1, 2);
        grid.add(addressLabel, 0, 3);
        grid.add(addressField, 1, 3);
        grid.add(submitButton, 1, 4);

        // Scene setup
        Scene scene = new Scene(grid, 400, 350);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Add Supplier");
        primaryStage.show();
    }
}
```

Display Items Page:

```
package project;

import java.math.BigDecimal;
import java.util.Date;

public class Item {
    private int itemId;
    private String itemName;
    private String description;
    private int quantity;
    private int reorderLevel;
    private BigDecimal unitPrice;
    private BigDecimal totalAmount;
    private Date purchaseDate;

    // Constructor
    public Item(int itemId, String itemName, String description, int quantity, int
reorderLevel,
                BigDecimal unitPrice, BigDecimal totalAmount, Date purchaseDate) {
        this.itemId = itemId;
        this.itemName = itemName;
        this.description = description;
        this.quantity = quantity;
        this.reorderLevel = reorderLevel;
        this.unitPrice = unitPrice;
        this.totalAmount = totalAmount;
        this.purchaseDate = purchaseDate;
    }

    // Getters and setters
    public int getItemId() {
        return itemId;
    }

    public void setItemId(int itemId) {
        this.itemId = itemId;
    }

    public String getItemName() {
        return itemName;
    }

    public void setItemName(String itemName) {
        this.itemName = itemName;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
```

```
public int getReorderLevel() {
    return reorderLevel;
}

public void setReorderLevel(int reorderLevel) {
    this.reorderLevel = reorderLevel;
}

public BigDecimal getUnitPrice() {
    return unitPrice;
}

public void setUnitPrice(BigDecimal unitPrice) {
    this.unitPrice = unitPrice;
}

public BigDecimal getTotalAmount() {
    return totalAmount;
}

public void setTotalAmount(BigDecimal totalAmount) {
    this.totalAmount = totalAmount;
}

public Date getPurchaseDate() {
    return purchaseDate;
    }

    public void setPurchaseDate(Date purchaseDate) {
        this.purchaseDate = purchaseDate;
    }
}
```

Display Customers Page:

```
package project;

import java.util.Date;

public class Customer {
    private int customerId;
    private String customerName;
    private String contactNo;
    private String email;
    private String address;
    private int loyaltyPoints;
    private Date lastPurchaseDate;

    // Constructor
    public Customer(int customerId, String customerName, String contactNo,
String email, String address,
                    int loyaltyPoints, Date lastPurchaseDate) {
        this.customerId = customerId;
        this.customerName = customerName;
    this.contactNo = contactNo;

    this.email = email;

    this.address = address;

    this.loyaltyPoints = loyaltyPoints;

    this.lastPurchaseDate = lastPurchaseDate;

}


// Getters and setters

public int getCustomerId() {

    return customerId;

}


public void setCustomerId(int customerId) {

    this.customerId = customerId;

}


public String getCustomerName() {

    return customerName;

}


public void setCustomerName(String customerName) {

    this.customerName = customerName;

}


public String getContactNo() {

    return contactNo;
```

```
}

public void setContactNo(String contactNo) {
   this.contactNo = contactNo;
}

public String getEmail() {
   return email;
}

public void setEmail(String email) {
   this.email = email;
}

public String getAddress() {
   return address;
}

public void setAddress(String address) {
   this.address = address;
}

public int getLoyaltyPoints() {
   return loyaltyPoints;
}
```

```
    public void setLoyaltyPoints(int loyaltyPoints) {
        this.loyaltyPoints = loyaltyPoints;
    }

    public Date getLastPurchaseDate() {
        return lastPurchaseDate;
    }

    public void setLastPurchaseDate(Date lastPurchaseDate) {
        this.lastPurchaseDate = lastPurchaseDate;
    }
}
```

<u>Display Invoice Page:</u>

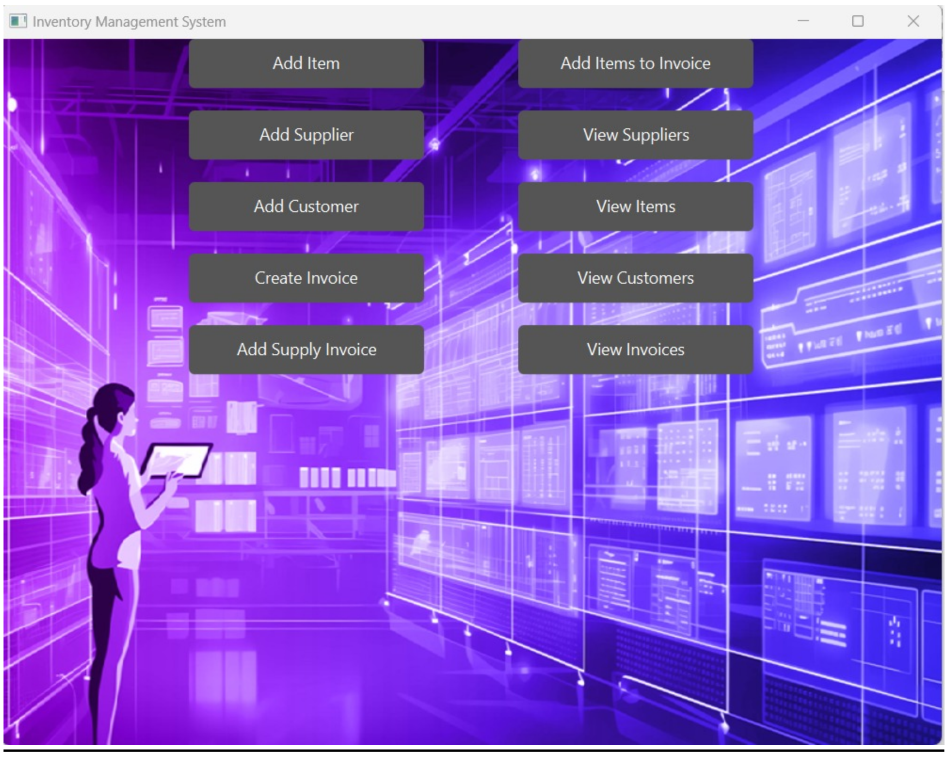
```
package project;

import java.math.BigDecimal;
import java.util.Date;

public class Invoice {
    private int invoiceId;
    private int customerId;
    private BigDecimal totalAmount;
    private Date invoiceDate;

    // Constructor
    public Invoice(int invoiceId, int customerId, BigDecimal totalAmount, Date
invoiceDate) {
        this.invoiceId = invoiceId;
        this.customerId = customerId;
        this.totalAmount = totalAmount;
        this.invoiceDate = invoiceDate;
    }

    // Getters and setters
    public int getInvoiceId() {
        return invoiceId;
    }

    public void setInvoiceId(int invoiceId) {
        this.invoiceId = invoiceId;
    }

    public int getCustomerId() {
        return customerId;
    }

    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

    public BigDecimal getTotalAmount() {
        return totalAmount;
    }

    public void setTotalAmount(BigDecimal totalAmount) {
        this.totalAmount = totalAmount;
    }

    public Date getInvoiceDate() {
        return invoiceDate;
    }

    public void setInvoiceDate(Date invoiceDate) {
        this.invoiceDate = invoiceDate;
    }
}
```

Display Supplier Page:

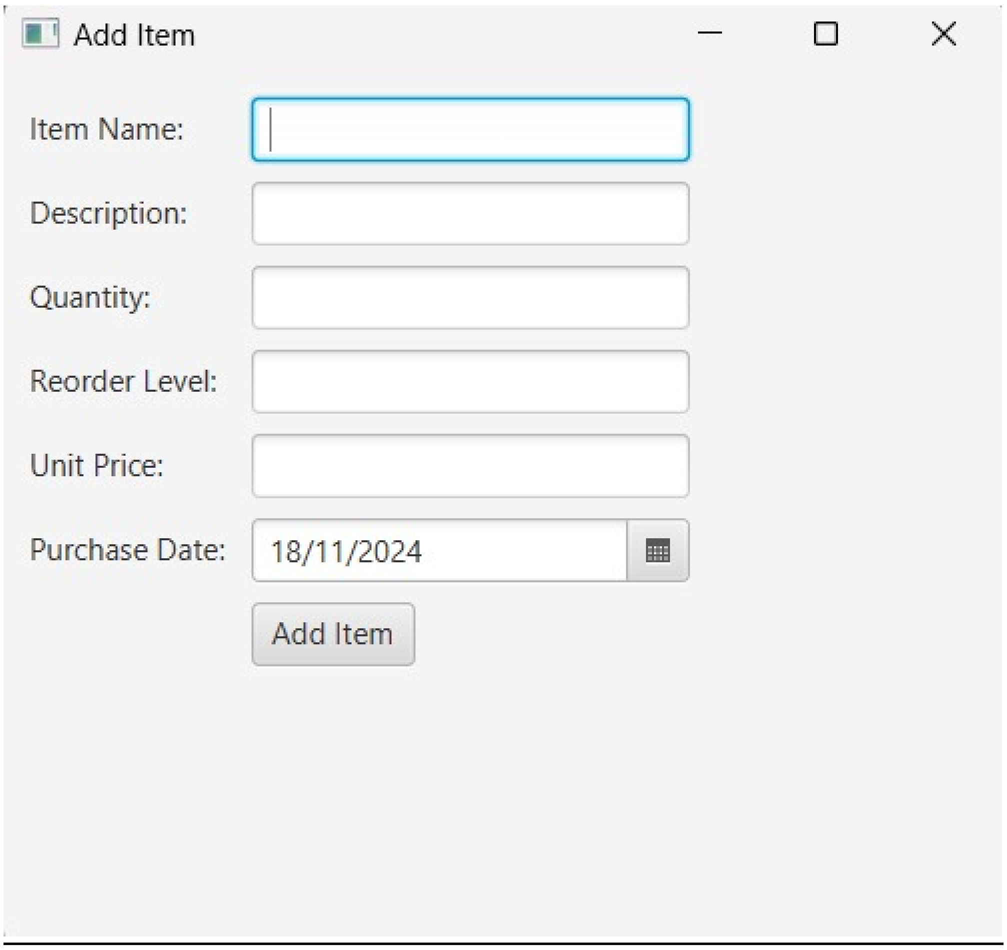
```
package project;

public class Supplier {
    private int supplierId;
    private String supplierName;
    private String contactNo;
    private String email;
    private String address;

    // Constructor
    public Supplier(int supplierId, String supplierName, String contactNo, String
email, String address) {
        this.supplierId = supplierId;
        this.supplierName = supplierName;
        this.contactNo = contactNo;
        this.email = email;
        this.address = address;
    }

    // Getters and setters
    public int getSupplierId() {
        return supplierId;
    }

    public void setSupplierId(int supplierId) {
        this.supplierId = supplierId;
    }

    public String getSupplierName() {
        return supplierName;
    }

    public void setSupplierName(String supplierName) {
        this.supplierName = supplierName;
    }

    public String getContactNo() {
        return contactNo;
    }

    public void setContactNo(String contactNo) {
        this.contactNo = contactNo;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
}
```

SNAPSHOTS

Home Page:



Add Items Page:



Item Name:

Description:

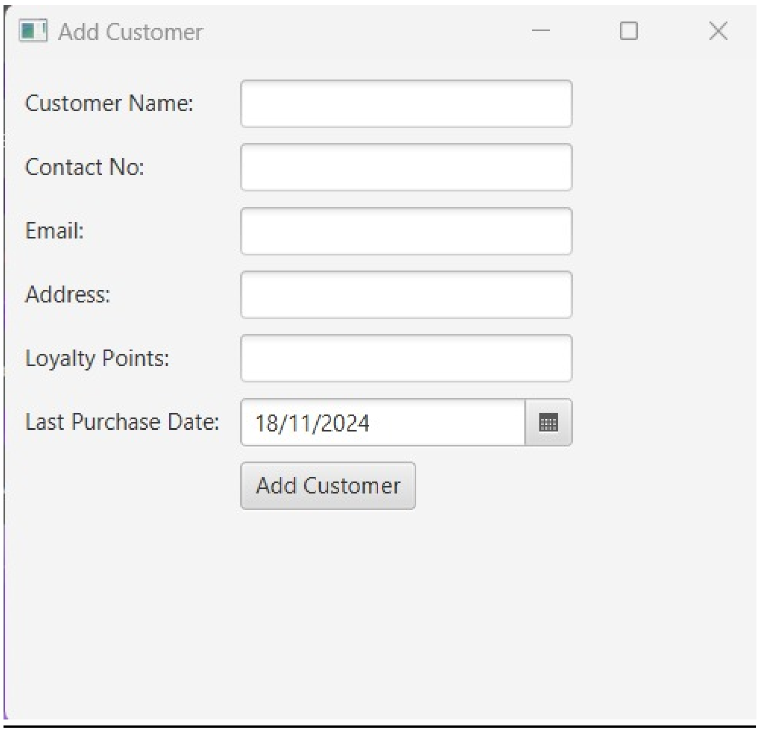
Quantity:

Reorder Level:

Unit Price:

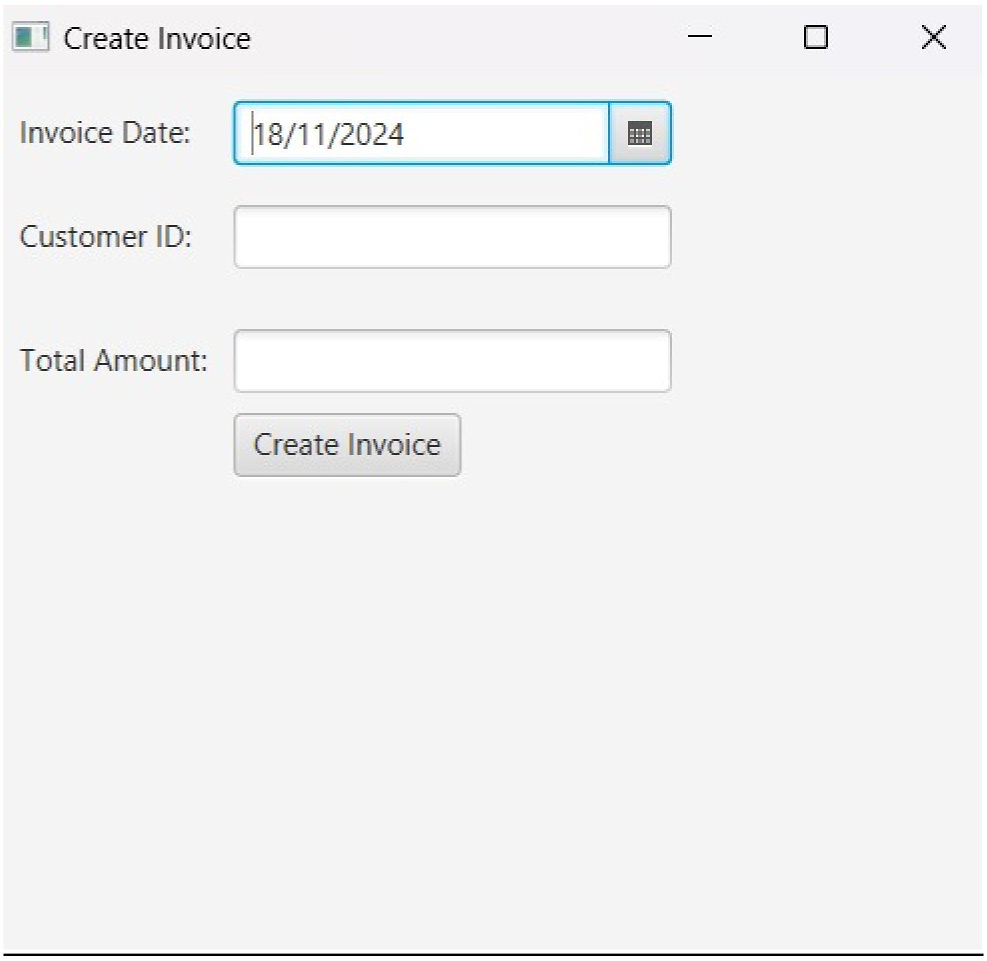
Purchase Date: 18/11/2024

Add Customer Page:


Add Customer

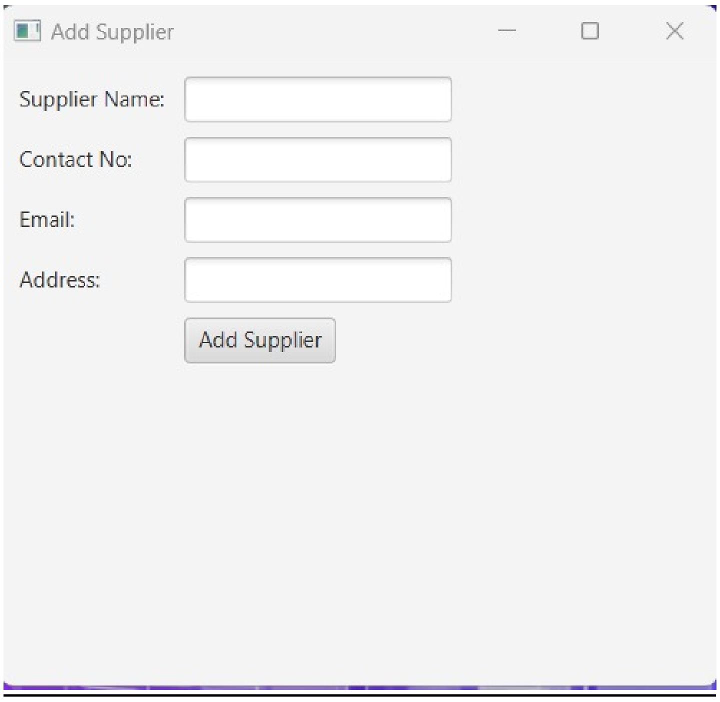
Customer Name:	
Contact No:	
Email:	
Address:	
Loyalty Points:	
Last Purchase Date:	18/11/2024

Add Invoice Page:


Create Invoice

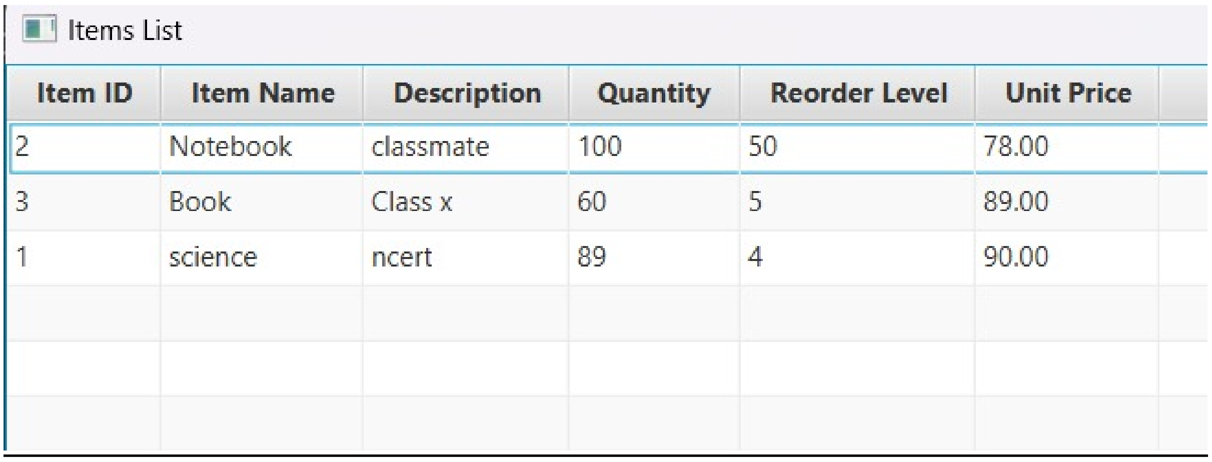
Invoice Date:	18/11/2024
Customer ID:	
Total Amount:	

Add Supplier Page:



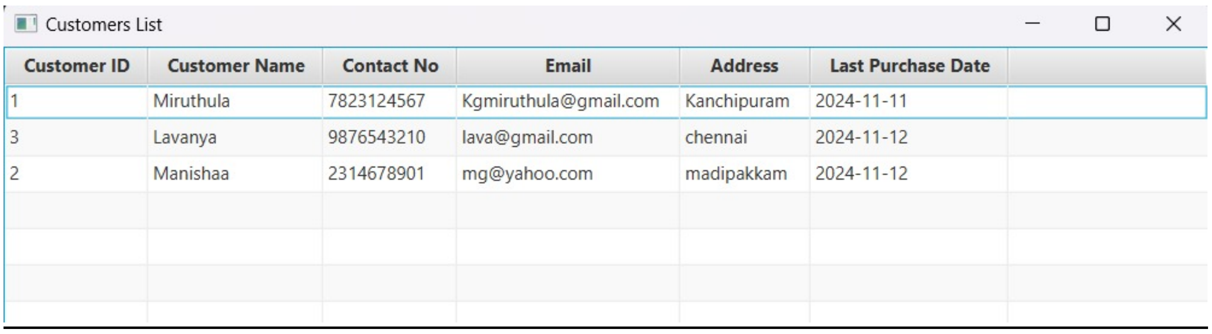
Supplier Name:	
Contact No:	
Email:	
Address:	

Display Items Page:



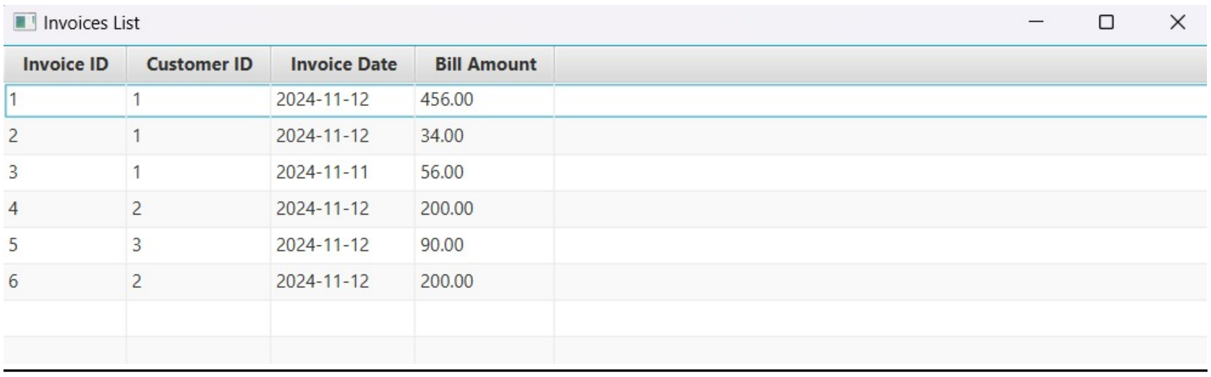
Item ID	Item Name	Description	Quantity	Reorder Level	Unit Price
2	Notebook	classmate	100	50	78.00
3	Book	Class x	60	5	89.00
1	science	ncert	89	4	90.00

Display Customer Page:



Customer ID	Customer Name	Contact No	Email	Address	Last Purchase Date
1	Miruthula	7823124567	Kgmiruthula@gmail.com	Kanchipuram	2024-11-11
3	Lavanya	9876543210	lava@gmail.com	chennai	2024-11-12
2	Manishaa	2314678901	mg@yahoo.com	madipakkam	2024-11-12

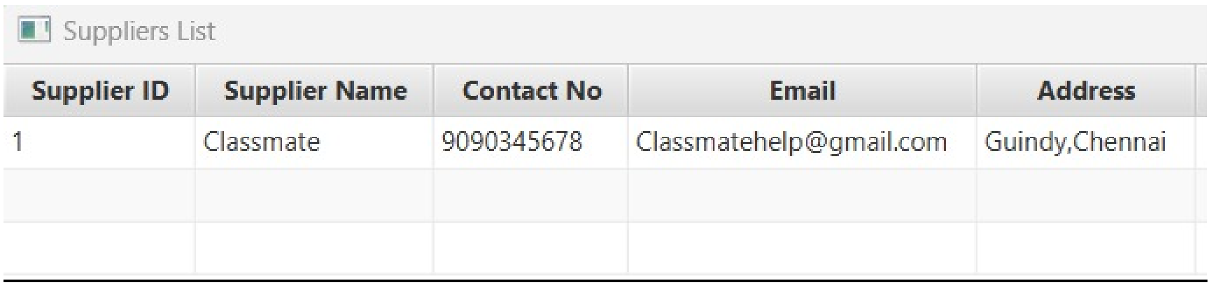
Display Invoice Page:



Invoices List

Invoice ID	Customer ID	Invoice Date	Bill Amount
1	1	2024-11-12	456.00
2	1	2024-11-12	34.00
3	1	2024-11-11	56.00
4	2	2024-11-12	200.00
5	3	2024-11-12	90.00
6	2	2024-11-12	200.00

Display Supplier Page:



Suppliers List

Supplier ID	Supplier Name	Contact No	Email	Address
1	Classmate	9090345678	Classmatehelp@gmail.com	Guindy,Chennai

CONCLUSION

The Alpha Stationary Shop Inventory Management System successfully addresses the challenges of manual inventory handling and retail operations by providing an efficient, automated solution tailored to the specific needs of a stationery shop. Through the use of JavaFX for an interactive, user-friendly graphical interface and PL/SQL for secure and robust database management, the project demonstrates the practical integration of frontend and backend technologies.