```
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();
}
```

}

```
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();
    }
}
```

```
package project;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class AddItemsToInvoicePage extends Application {

    @Override
    public void start(Stage primaryStage) {
        // UI Components
        Label itemIdLabel = new Label("Item ID:");
        TextField itemIdField = new TextField();

        Label invoiceIdLabel = new Label("Invoice ID:");
        TextField invoiceIdField = new TextField();

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

        Label unitPriceLabel = new Label("Unit Price:");
        TextField unitPriceField = new TextField();

        Button submitButton = new Button("Submit");
        submitButton.setOnAction(e -> {
            // Retrieve input values
            int itemId = Integer.parseInt(itemIdField.getText());
            int invoiceId = Integer.parseInt(invoiceIdField.getText());
            int quantity = Integer.parseInt(quantityField.getText());
            double unitPrice = Double.parseDouble(unitPriceField.getText());
            double totalAmount = unitPrice * quantity;

            // Create DB connection and execute the necessary queries
            dbconnect db = new dbconnect();
            try (Connection con = db.connect()) {
                // 1. Check if enough quantity is available in the items table
                if (checkItemAvailability(con, itemId, quantity)) {
                    // 2. Add item to the invoice table
                    db.addItemsToInvoice(con, itemId, invoiceId, quantity, unitPrice, totalAmount);

                    // 3. Reduce the item quantity in the items table
                    reduceItemQuantity(con, itemId, quantity);

                    // Close the stage after successful operation
                    primaryStage.close();
                } else {
                    // Show an alert or message if not enough items are available
```

```
                System.out.println("Not enough items available.");
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    });

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

    grid.add(itemIdLabel, 0, 0);
    grid.add(itemIdField, 1, 0);
    grid.add(invoiceIdLabel, 0, 1);
    grid.add(invoiceIdField, 1, 1);
    grid.add(quantityLabel, 0, 2);
    grid.add(quantityField, 1, 2);
    grid.add(unitPriceLabel, 0, 3);
    grid.add(unitPriceField, 1, 3);
    grid.add(submitButton, 1, 4);

    Scene scene = new Scene(grid, 400, 250);
    primaryStage.setScene(scene);
    primaryStage.setTitle("Add Items to Invoice");
    primaryStage.show();
}

/**
 * Checks if the requested quantity is available in the items table.
 * @param con The database connection.
 * @param itemId The ID of the item.
 * @param quantity The quantity to check.
 * @return true if the quantity is available, false otherwise.
 */
private boolean checkItemAvailability(Connection con, int itemId, int quantity) throws SQLException {
    String query = "SELECT quantity FROM items WHERE item_id = ?";
    try (PreparedStatement stmt = con.prepareStatement(query)) {
        stmt.setInt(1, itemId);
        try (ResultSet rs = stmt.executeQuery()) {
            if (rs.next()) {
                int availableQuantity = rs.getInt("quantity");
                return availableQuantity >= quantity;
            }
        }
    }
    return false;
}

/**
 * Reduces the quantity of the item in the items table after it has been added to the invoice.
 * @param con The database connection.
 * @param itemId The ID of the item.
 * @param quantity The quantity to reduce.
 */
private void reduceItemQuantity(Connection con, int itemId, int quantity) throws SQLException {
```

```
        String query = "UPDATE items SET quantity = quantity - ? WHERE item_id = ?";
        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, quantity);
            stmt.setInt(2, itemId);
            stmt.executeUpdate();
        }
    }
}
```

```
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();
    }
}
```

```
package project;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.Date;

public class AddSupplyInvoicePage extends Application {

    @SuppressWarnings("unused")
    @Override
    public void start(Stage primaryStage) {
        // UI Components
        Label itemIdLabel = new Label("Item ID:");
        TextField itemIdField = new TextField();

        Label supplierIdLabel = new Label("Supplier ID:");
        TextField supplierIdField = new TextField();

        Label supplyPriceLabel = new Label("Supply Price:");
        TextField supplyPriceField = new TextField();

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

        Label billDateLabel = new Label("Bill Date:");
        DatePicker billDatePicker = new DatePicker();

        Button submitButton = new Button("Submit");
        submitButton.setOnAction(e -> {
            int itemId = Integer.parseInt(itemIdField.getText());
            int supplierId = Integer.parseInt(supplierIdField.getText());
            double supplyPrice = Double.parseDouble(supplyPriceField.getText());
            int quantity = Integer.parseInt(quantityField.getText());
            Date billDate = Date.valueOf(billDatePicker.getValue());

            dbconnect db = new dbconnect();
            Connection con = db.connect();
            db.addSupplyInvoice(con, itemId, supplierId, supplyPrice, quantity, billDate);
            primaryStage.close();
        });

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

        grid.add(itemIdLabel, 0, 0);
```

```
        grid.add(itemIdField, 1, 0);
        grid.add(supplierIdLabel, 0, 1);
        grid.add(supplierIdField, 1, 1);
        grid.add(supplyPriceLabel, 0, 2);
        grid.add(supplyPriceField, 1, 2);
        grid.add(quantityLabel, 0, 3);
        grid.add(quantityField, 1, 3);
        grid.add(billDateLabel, 0, 4);
        grid.add(billDatePicker, 1, 4);
        grid.add(submitButton, 1, 5);

        Scene scene = new Scene(grid, 400, 300);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Add Supply Invoice");
        primaryStage.show();
    }
}
```

```
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").toExternalForm(), 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);
    }
}
```

```
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;
    }
}
```

```
package project;


//import java.sql.Statement;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
//import java.sql.ResultSet;
import java.sql.SQLException;

public class dbconnect {

   public Connection connect(){
      final String username = "postgres";
      final String password = "priya";
      final String URL = "jdbc:postgresql://localhost:5432/Inventory System";
      Connection conn = null;
      try{

         conn = DriverManager.getConnection(URL, username, password);
         System.out.println("Connected to the PostgreSQL database successfully!");

      }
      catch(Exception e){
         System.out.println(e);
      }
      return conn;
   }
   public void addNewItem(Connection conn, String item, String desc, int quantity, int reorderLevel, float unitPrice, Date
date) {
      try {

         String query = "INSERT INTO items (item_name, description, quantity, reorder_level, unit_price, total_amount,
purchase_date) VALUES (?, ?, ?, ?, ?, ?, ?)";

         PreparedStatement stmt = conn.prepareStatement(query);

         stmt.setString(1, item);                  // item_name
         stmt.setString(2, desc);                  // description
         stmt.setInt(3, quantity);                 // quantity_in_stock
         stmt.setInt(4, reorderLevel);               // reorder_level
         stmt.setFloat(5, unitPrice);               // unit_price
         stmt.setFloat(6, unitPrice * quantity);     // total_amount (calculated as unitPrice * quantity)
         stmt.setDate(7, date);                    // purchase_date


         stmt.executeUpdate();
         System.out.println("Row inserted successfully.");


         stmt.close();
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
   public void addStockQuantity(Connection conn, int itemId, int additionalQuantity) {
```

```
        try {

            String query = "UPDATE items SET quantity = quantity + ? WHERE item_id = ?";
            PreparedStatement stmt = conn.prepareStatement(query);
            stmt.setInt(1, additionalQuantity); // additional quantity to add
            stmt.setInt(2, itemId);             // item_id of the item to update
            int rowsUpdated = stmt.executeUpdate();

            if (rowsUpdated > 0) {
                System.out.println("Stock quantity updated successfully.");
            } else {
                System.out.println("No item found with the given item_id.");
            }

            // Close the statement
            stmt.close();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
    public void addSupplier(Connection conn, String supplierName, String contactNo, String email, String address) {
        String sql = "INSERT INTO supplier (supplier_name, contact_no, email, address) VALUES (?, ?, ?, ?)";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, supplierName);
            pstmt.setString(2, contactNo);  // Contact number stored as a string (e.g., to allow special characters)
            pstmt.setString(3, email);
            pstmt.setString(4, address);

            int rowsAffected = pstmt.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Supplier added successfully!");
            } else {
                System.out.println("Failed to add supplier.");
            }
        } catch (SQLException e) {
            System.out.println("Error adding supplier: " + e.getMessage());
        }
    }
    public void addCustomer(Connection conn, String customerName, String contactNo, String email, String address, int
loyaltyPoints,Date lastPurchaseDate) {
        String sql = "INSERT INTO customer (customer_name, contact_no, email, address, loyalty_points, last_purchase_date)
"
                + "VALUES (?, ?, ?, ?, ?, ?)";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, customerName);
            pstmt.setString(2, contactNo);
            pstmt.setString(3, email);
            pstmt.setString(4, address);
            pstmt.setInt(5, loyaltyPoints);
            pstmt.setDate(6, lastPurchaseDate);

            int rowsInserted = pstmt.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("Customer added successfully.");
            } else {
```

```
            System.out.println("Failed to add customer.");
        }
    } catch (SQLException e) {
        System.out.println("Error adding customer: " + e.getMessage());
    }
}
public void addSupplyInvoice(Connection con, int itemId, int supplierId, double supplyPrice, int quantity, Date billDate)
{
    String sql = "INSERT INTO supplyInvoice (item_id, supplier_id, supply_price, quantity, bill_amount, bill_date) " +
                 "VALUES (?, ?, ?, ?, ?, ?)";

    try (PreparedStatement stmt = con.prepareStatement(sql)) {
        // Set the values for the prepared statement
        stmt.setInt(1, itemId);
        stmt.setInt(2, supplierId);
        stmt.setDouble(3, supplyPrice);
        stmt.setInt(4, quantity);
        stmt.setDouble(5,quantity*supplyPrice);
        stmt.setDate(6, billDate); // Use the automatically set bill date

        // Execute the insert query
        int rowsInserted = stmt.executeUpdate();
        if (rowsInserted > 0) {
            System.out.println("Supply invoice added successfully!");
        } else {
            System.out.println("Failed to add supply invoice.");
        }
    } catch (SQLException e) {
        System.out.println("Error while adding supply invoice: " + e.getMessage());
    }
}
public void addItemsToInvoice(Connection con, int itemId, int invoiceId, int quantity, double unitPrice, double
totalAmount) {
    String sql = "INSERT INTO items_invoice (item_id, invoice_id, quantity, unit_price, total_amount) " +
                 "VALUES (?, ?, ?, ?, ?)";

    try (PreparedStatement stmt = con.prepareStatement(sql)) {
        // Set the values for the prepared statement
        stmt.setInt(1, itemId);
        stmt.setInt(2, invoiceId);
        stmt.setInt(3, quantity);
        stmt.setDouble(4, unitPrice);
        stmt.setDouble(5, totalAmount);

        // Execute the insert query
        int rowsInserted = stmt.executeUpdate();
        if (rowsInserted > 0) {
            System.out.println("Item added to invoice successfully!");
        } else {
            System.out.println("Failed to add item to invoice.");
        }
    } catch (SQLException e) {
        System.out.println("Error while adding item to invoice: " + e.getMessage());
    }
}
}
```

```
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;
    }
}
```

```
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;
    }
}
```

```
package project;

import javafx.geometry.Pos;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.layout.*;
import javafx.stage.Stage;

public class LoginPage {

    // Method to create and return the login page layout
    @SuppressWarnings("unused")
    public VBox createLoginPage() {
        // Create the UI components

        // Title label
        Label titleLabel = new Label("Inventory Management System");
        titleLabel.setStyle("-fx-font-size: 36px; -fx-font-weight: bold; -fx-text-fill: #ffffff;");

        // Username field
        Label usernameLabel = new Label("Username:");
        TextField usernameField = new TextField();
        usernameField.setPromptText("Enter your username");
        usernameField.setStyle("-fx-font-size: 16px; -fx-padding: 10px;");

        // Password field
        Label passwordLabel = new Label("Password:");
        PasswordField passwordField = new PasswordField();
        passwordField.setPromptText("Enter your password");
        passwordField.setStyle("-fx-font-size: 16px; -fx-padding: 10px;");

        // Login button
        Button loginButton = new Button("Login");
        loginButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 15px;");

        // Clear button
        Button clearButton = new Button("Clear");
        clearButton.setStyle("-fx-background-color: #FF5733; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 15px;");

        // Message label (to show error/success messages)
        Label messageLabel = new Label();
        messageLabel.setStyle("-fx-font-size: 14px; -fx-text-fill: red;");

        // Action for login button
        loginButton.setOnAction(event -> {
            String username = usernameField.getText();
            String password = passwordField.getText();

            // Validate username and password (hardcoded validation for simplicity)
            if (username.isEmpty() || password.isEmpty()) {
                messageLabel.setText("Please enter both username and password.");
            } else if (username.equals("admin") && password.equals("admin123")) {
                messageLabel.setText("Login successful!");
                messageLabel.setStyle("-fx-font-size: 14px; -fx-text-fill: green;");

                // Here you would typically transition to the main inventory dashboard
                // For now, we just close the login window as an example
```

```
            Stage stage = (Stage) loginButton.getScene().getWindow();
            stage.close(); // Close the login window upon successful login

            // You can now open the main inventory page or other functionality
            // e.g., openDashboard();
        } else {
            messageLabel.setText("Invalid username or password.");
        }
    });

    // Action for clear button
    clearButton.setOnAction(event -> {
        usernameField.clear();
        passwordField.clear();
        messageLabel.setText("");
    });

    // Create the layout
    VBox layout = new VBox(20); // VBox layout with 20px spacing
    layout.setAlignment(Pos.CENTER);
    layout.setStyle("-fx-background-color: rgba(0, 0, 0, 0.7); -fx-padding: 40px; -fx-background-radius: 10px;");

    // Add UI elements to the layout
    layout.getChildren().addAll(
        titleLabel,
        usernameLabel,
        usernameField,
        passwordLabel,
        passwordField,
        loginButton,
        clearButton,
        messageLabel
    );

    Image image = new Image(getClass().getResource("/images/login.jpg").toExternalForm());
    BackgroundImage backgroundImage = new BackgroundImage(
        image, BackgroundRepeat.NO_REPEAT, BackgroundRepeat.NO_REPEAT,
        BackgroundPosition.CENTER, BackgroundSize.DEFAULT
    );
    layout.setBackground(new Background(backgroundImage));

    return layout;  // Return the layout
  }
}
```

```
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();
    }
}
```

```
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;
```

```
    }
}
```