**SETU Carlow**

**Purchase Manager**

**Project**

**Name: Ross Blundell**

**Date: 13/04/2023**

**Tutor: Jason Barron**

**Table of Contents**

***Section*  *Page***

**Description 3**

**Requirements 4**

**DB Table Screenshots 5**

**ER Diagram 8**

**Interesting source code snippets 9**

**Tests 17**

**Source Code 18**

**Description**

My project is an inventory and product purchase management system for a small retail business. The purpose of the system is to help the business keep track of its product inventory, process customer purchases, and generate invoices.

The main features of the system include a graphical user interface (GUI) for easy navigation, a product table displaying information about each product (name, description, price, and stock), the ability to add new products, edit existing products, and delete products, the ability to process customer purchases, generate invoices, and track sales, and a login system to restrict access to authorized personnel only.

The system also offers customer account creation and login to give customers access to a catalogue of products on offer which they can then purchase directly from the catalogue in the system.

Overall, the project aims to streamline the business's inventory management process, reduce errors and inefficiencies, and improve the customer shopping experience.

**Requirements**

A user-friendly graphical user interface (GUI) that allows users to interact with the system.

A database that stores information about products, customers, invoices, and sales.

The ability to add, update, and delete products and customers from the system.

The ability to generate invoices for purchases made by customers.

The ability to view and search for products and customers in the system.

The ability to generate reports on sales and inventory levels.

Robust error handling and input validation to prevent data entry errors and system crashes.

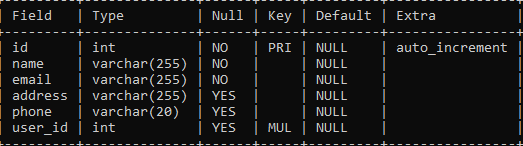
Efficient data processing and storage to handle a potentially large amount of data.

Good security practices to protect sensitive data and prevent unauthorized access.

User authentication and authorization to ensure that only authorized users can access the system and perform certain actions.

**DB Table Screenshots**

Customer Table Structure:

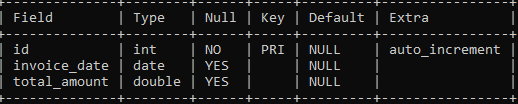


Customer Table Data:

Text

Description automatically generated with low confidence

Invoice Table Structure:



Invoice Table Data:

Table

Description automatically generated with medium confidence

Product Table Structure:

Table

Description automatically generated with medium confidence

Product Table Data:

Text

Description automatically generated

User Table Structure:

A screenshot of a computer

Description automatically generated with medium confidence

User Table Data:

A screenshot of a computer

Description automatically generated with low confidence

**ER Diagram**

Diagram

Description automatically generated

**Interesting Source Code Snippets**

private void addCustomer() {

        // Create a form panel to add customer details

        JTextField nameField = new JTextField(15);

        JTextField emailField = new JTextField(15);

        JTextField addressField = new JTextField(15);

        JTextField phoneField = new JTextField(15);

        JPanel customerFormPanel = new JPanel(new GridLayout(0, 2));

        customerFormPanel.add(new JLabel("Name:"));

        customerFormPanel.add(nameField);

        customerFormPanel.add(new JLabel("Email:"));

        customerFormPanel.add(emailField);

        customerFormPanel.add(new JLabel("Address:"));

        customerFormPanel.add(addressField);

        customerFormPanel.add(new JLabel("Phone:"));

        customerFormPanel.add(phoneField);

        // Flag to check if the input is valid

        boolean validInput = false;

        // Loop until the user enters valid input

        while (!validInput) {

            int result = JOptionPane.showConfirmDialog(null, customerFormPanel, "Add Customer", JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

            // If the user clicks OK, validate the input

            if (result == JOptionPane.OK\_OPTION) {

                String name = nameField.getText();

                String email = emailField.getText();

                String address = addressField.getText();

                String phone = phoneField.getText();

                // Check if the input is not valid and display an error message

                if (!isValidEmail(email)) {

                    JOptionPane.showMessageDialog(null, "Please enter a valid email address.", "Error", JOptionPane.ERROR\_MESSAGE);

                    continue;

                }

                try {

                    Integer.parseInt(phone);

                } catch (NumberFormatException e) {

                    JOptionPane.showMessageDialog(null, "Please enter a valid phone number (only integers).", "Error", JOptionPane.ERROR\_MESSAGE);

                    continue;

                }

                // If validation passes, insert the new customer into the database

                try {

                    String query = "INSERT INTO customer (name, email, address, phone) VALUES (?, ?, ?, ?)";

                    PreparedStatement preparedStatement = connection.prepareStatement(query);

                    preparedStatement.setString(1, name);

                    preparedStatement.setString(2, email);

                    preparedStatement.setString(3, address);

                    preparedStatement.setString(4, phone);

                    preparedStatement.executeUpdate();

                    // Refresh the customer table

                    populateTables();

                    // Set validInput to true to exit the loop

                    validInput = true;

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            } else {

                // If the user clicks "Cancel," exit the loop

                break;

            }

        }

    }

// Method to add product to database

    private void addProduct() {

        JTextField nameField = new JTextField(15);

        JTextField descriptionField = new JTextField(15);

        JTextField priceField = new JTextField(15);

        JTextField stockField = new JTextField(15);

        JPanel productFormPanel = new JPanel(new GridLayout(0, 2));

        productFormPanel.add(new JLabel("Name:"));

        productFormPanel.add(nameField);

        productFormPanel.add(new JLabel("Description:"));

        productFormPanel.add(descriptionField);

        productFormPanel.add(new JLabel("Price:"));

        productFormPanel.add(priceField);

        productFormPanel.add(new JLabel("Stock:"));

        productFormPanel.add(stockField);

        int result = JOptionPane.showConfirmDialog(null, productFormPanel, "Add Product", JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

        if (result == JOptionPane.OK\_OPTION) {

            String name = nameField.getText();

            String description = descriptionField.getText();

            String price = priceField.getText();

            String stock = stockField.getText();

            // Validate price input

            try {

                double priceValue = Double.parseDouble(price);

                if (priceValue <= 0) {

                    throw new NumberFormatException();

                }

            } catch (NumberFormatException e) {

                JOptionPane.showMessageDialog(null, "Price must be a number and be greater than 0.00", "Invalid price", JOptionPane.ERROR\_MESSAGE);

                return;

            }

            // Validate stock input

            int stockValue;

            try {

                stockValue = Integer.parseInt(stock);

                if (stockValue < 0) {

                    throw new NumberFormatException();

                }

            } catch (NumberFormatException e) {

                JOptionPane.showMessageDialog(null, "Stock must be a positive number", "Invalid stock", JOptionPane.ERROR\_MESSAGE);

                return;

            }

            // If validation passes, insert the new product into the database

            try {

                String query = "INSERT INTO product (name, description, price, stock) VALUES (?, ?, ?, ?)";

                PreparedStatement preparedStatement = connection.prepareStatement(query);

                preparedStatement.setString(1, name);

                preparedStatement.setString(2, description);

                preparedStatement.setDouble(3, Double.parseDouble(price));

                preparedStatement.setInt(4, stockValue); // Set the stock value

                preparedStatement.executeUpdate();

                // Refresh the product table

                populateTables();

            } catch (SQLException e) {

                e.printStackTrace();

            }

        }

    }

// handleLogin Method

    private void handleLogin() {

        String enteredUsername = usernameField.getText().trim();

        String enteredPassword = new String(passwordField.getPassword());

        boolean validUser = validateUser(enteredUsername, enteredPassword);

        if (validUser) {

            // Get the userRole

            try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/purchases", "root", "P0rtf0l10s;23");

                 PreparedStatement preparedStatement = connection.prepareStatement("SELECT role FROM user WHERE username = ?")) {

                preparedStatement.setString(1, enteredUsername);

                ResultSet resultSet = preparedStatement.executeQuery();

                if (resultSet.next()) {

                    userRole = resultSet.getString("role");

                }

            } catch (SQLException e) {

                e.printStackTrace();

            }

            // Call the loginSuccessListener with userRole if it's not null

            if (loginSuccessListener != null) {

                loginSuccessListener.accept(userRole);

            }

            // Close the login form

            dispose();

        } else {

            JOptionPane.showMessageDialog(this, "Username or Password incorrect.", "Login Failed", JOptionPane.ERROR\_MESSAGE);

        }

    }

    // Validateuser Method

    private boolean validateUser(String enteredUsername, String enteredPassword) {

        String query = "SELECT \* FROM user WHERE username = ? AND password = ?";

        try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/purchases", "root", "P0rtf0l10s;23");

             PreparedStatement preparedStatement = connection.prepareStatement(query)) {

            preparedStatement.setString(1, enteredUsername);

            preparedStatement.setString(2, enteredPassword);

            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {

                return true;

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return false;

    }

// Method to handle purchase

    private void handlePurchase() {

        int selectedRow = productTable.getSelectedRow();

        if (selectedRow == -1) {

            JOptionPane.showMessageDialog(this, "Please select a product to purchase.", "Error", JOptionPane.ERROR\_MESSAGE);

            return;

        }

        int productId = Integer.parseInt(productTable.getValueAt(selectedRow, 0).toString());

        String productName = productTable.getValueAt(selectedRow, 1).toString();

        double productPrice = Double.parseDouble(productTable.getValueAt(selectedRow, 3).toString());

        int confirmPurchase = JOptionPane.showConfirmDialog(this, "Are you sure you want to purchase " + productName + "?", "Confirm Purchase", JOptionPane.YES\_NO\_OPTION);

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

            try {

                // Insert the purchase into the invoice table

                String sql = "INSERT INTO invoice (invoice\_date, total\_amount) VALUES (?, ?);";

                PreparedStatement preparedStatement = connection.prepareStatement(sql);

                preparedStatement.setDate(1, new java.sql.Date(System.currentTimeMillis()));

                preparedStatement.setDouble(2, productPrice);

                preparedStatement.executeUpdate();

                // Update the stock value of the purchased product

                sql = "UPDATE product SET stock = stock - 1 WHERE id = ?";

                preparedStatement = connection.prepareStatement(sql);

                preparedStatement.setInt(1, productId);

                preparedStatement.executeUpdate();

                JOptionPane.showMessageDialog(this, "Product purchased successfully!", "Success", JOptionPane.INFORMATION\_MESSAGE);

                // Refresh the product table to reflect the updated stock value

                refreshProductTable();

            } catch (SQLException ex) {

                ex.printStackTrace();

                JOptionPane.showMessageDialog(this, "Error purchasing the product.", "Error", JOptionPane.ERROR\_MESSAGE);

            }

        }

    }

**Test**

During the development of the program, various types of testing were conducted to ensure the functionality and reliability of the system. Unit testing was performed to check the correctness of individual components and methods of the program. This included testing the database connection, input validation, and updating of database records.

Integration testing was also carried out to evaluate the interactions between the different modules of the program. This involved testing the communication between the user interface and the database management system. In addition, system testing was conducted to assess the performance, functionality, and usability of the entire system as a whole. This involved testing various scenarios, such as adding, editing, and deleting products, purchasing items, and generating invoices.

Furthermore, manual testing was done to ensure that the program met the specified requirements and was user-friendly. This involved verifying that the program worked as intended, identifying and fixing errors, and making any necessary adjustments. Overall, the testing process was critical to the success of the program, as it helped to ensure that it was reliable, efficient, and effective in meeting the needs of the users.

**Source Code**

Full Source Code can be found in my github repository below

<https://github.com/RossBlundell1305/OOSD/tree/main/Project>