Style Guidelines for Final Year Project ReportsCustomer Management System

Semester Project Report

Session 2021-2025

A 4th Year Student

A project submitted in the final fulfillment of the

International Islamic University Degree

of

BSc. (Hons.)BS in Software Engineering (BSSE)



Department of Computer Science

International Islamic University H-10, Islamabad

Semester Spring

**Related details:**

|  |  |
| --- | --- |
| Course Name | Advanced Computer Programming |
| Instructor Name | Zohair Ahmad |
| Student Name |  |
| ST-Registration No |  |
| Submission Date | 02 June 2024 |
| Semester | 6th |

[Customer Management System 1](#_Toc168011756)

[Semester Project Report 1](#_Toc168011757)

[Session 2021-2025 1](#_Toc168011758)

[1 Introduction 4](#_Toc168011759)

[1.1 Project Overview: 4](#_Toc168011760)

[1.2 Technologies Used 4](#_Toc168011761)

[2 GUI Design 4](#_Toc168011762)

[2.1 Overview: 4](#_Toc168011763)

[2.2 Screenshots: 4](#_Toc168011764)

[2.3 Navigation Flow: 4](#_Toc168011765)

[3 Database Design 4](#_Toc168011766)

[3.1 Schema Diagram: 4](#_Toc168011767)

[3.2 Table Descriptions: 4](#_Toc168011768)

[3.3 Sample Data: 4](#_Toc168011769)

[4 Implementation 4](#_Toc168011770)

[4.1 Code Snippets: 4](#_Toc168011771)

[4.2 Explanation: 4](#_Toc168011772)

[5 Conclusion 5](#_Toc168011773)

[5.1 Summary: 5](#_Toc168011774)

[6 Appendices 5](#_Toc168011775)

[6.1 User Manual: 5](#_Toc168011776)

[6.2 Source Code: 5](#_Toc168011777)

[6.3 Live Project Demo: 5](#_Toc168011778)

[7 Final Year Project (FYP) Report Outline 6](#_Toc168011779)

# Introduction

## Project Overview:

**Purpose**

The primary purpose of this application is to facilitate the efficient management of customer information for businesses. The application aims to streamline the process of recording, storing, and retrieving customer details by providing a user-friendly interface. This helps companies to maintain accurate records, enhance customer service, and make informed decisions based on collected data.

**Main Features**

**User-Friendly Data Entry:**

* + **Forms for Customer Details:** Simple and intuitive forms to input customer name, amount paid, address, and company name.
  + **Validation:** Real-time validation to ensure data accuracy and completeness.

**Data Management:**

* + **CRUD Operations:** Create, read, update, and delete customer records with ease.

**Data Display:**

* + **Table View:** Displays a list of customers with essential details, and supports sorting and filtering.

**User Experience Enhancements:**

* + **Dialog Boxes:** User-friendly dialogs for confirmations, alerts, and error messages.
  + **Keyboard Shortcuts:** Accessibility features for quick navigation and data entry.

**Security:**

* + **User Authentication:** Optional feature to secure access to the application.

**Target Users**

* **Small to Medium-Sized Businesses (SMBs)**
* **Sales and Marketing Teams**
* **Administrative Staff**
* **Business Analysts**
* **Entrepreneurs and Sole Proprietors**

## Technologies Used

* JAVA
* JSwing
* AWT
* JDBC
* MySQL

# GUI Design

## Overview:

**Layout**

**MainWindow:**

* + **Title Bar:** Displays the page name, e.g., "Signup Form"
  + **Labels and Text Fields:** For entering User information such as **UserName**, **Email, Password, Confirm Password.**
  + **Button:** Includes button **Submit** typically placed at the bottom of the form for easy access.

**Login Window:**

* **Labels and Text Fields:** For entering User information such as **Email and Password.**
* **Button:** Includes button **Login** typically placed at the bottom of the form for easy access.

**Data Entry Form:**

* + **Labels and Text Fields:** For entering customer details such as **Name**, **Payment**, **Address**, and **Company Name**.
  + **Buttons:** Includes buttons such as **Add**, **Update**, **Delete**, and **Show Table**, typically placed at the bottom of the form for easy access.

**Details/Information Frame:**

* + **Table View:** Displays a table of customer records with columns for **cst\_Name**, **cst\_Payment**, **cst\_Address**, and **cst\_Company**.
  + **Action Buttons:** Actions such as **Load data**, and **Logout** may be included here for convenience.

**Design Patterns**

**Model-View-Controller (MVC):**

* + **Model:** Represents the customer data and the logic for storing and retrieving this data.
  + **View:** The graphical interface that displays the data (forms, tables, etc.) and receives user input.
  + **Controller:** Handles the user input, interacts with the model, and updates the view accordingly.

**Observer Pattern:**

* + Ensures that changes in the data model automatically update the views, keeping the interface synchronized with the underlying data.

**Singleton Pattern:**

* + Used for components that should have a single instance across the application, such as the database connection manager or configuration settings.

**User Experience Considerations**

**Simplicity and Clarity:**

* + The interface is designed to be straightforward, with clear labels and an intuitive layout to reduce the learning curve for users. Forms are logically laid out, grouping related fields.

**Responsiveness:**

* + The application responds quickly to user inputs, providing immediate feedback through the status bar and dialog boxes. This includes loading screens or progress indicators for long operations.

**Error Handling and Validation:**

* + Input fields include validation checks to ensure that the data entered is correct (e.g., numeric checks for the **Amount Paid** field). User-friendly error messages guide the user in correcting any mistakes.

**Consistency:**

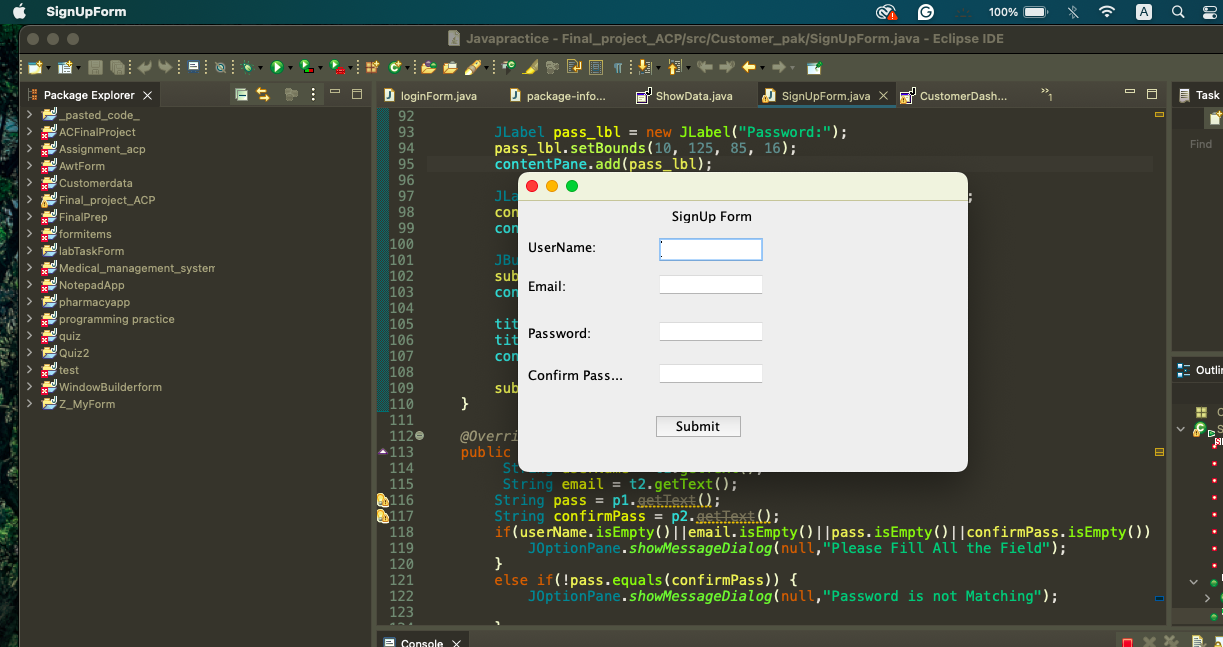
* + Consistent design elements (such as button placement, color schemes, and fonts) help users quickly learn how to navigate the application. Standardized icons and common UI elements enhance familiarity.

**Accessibility:**

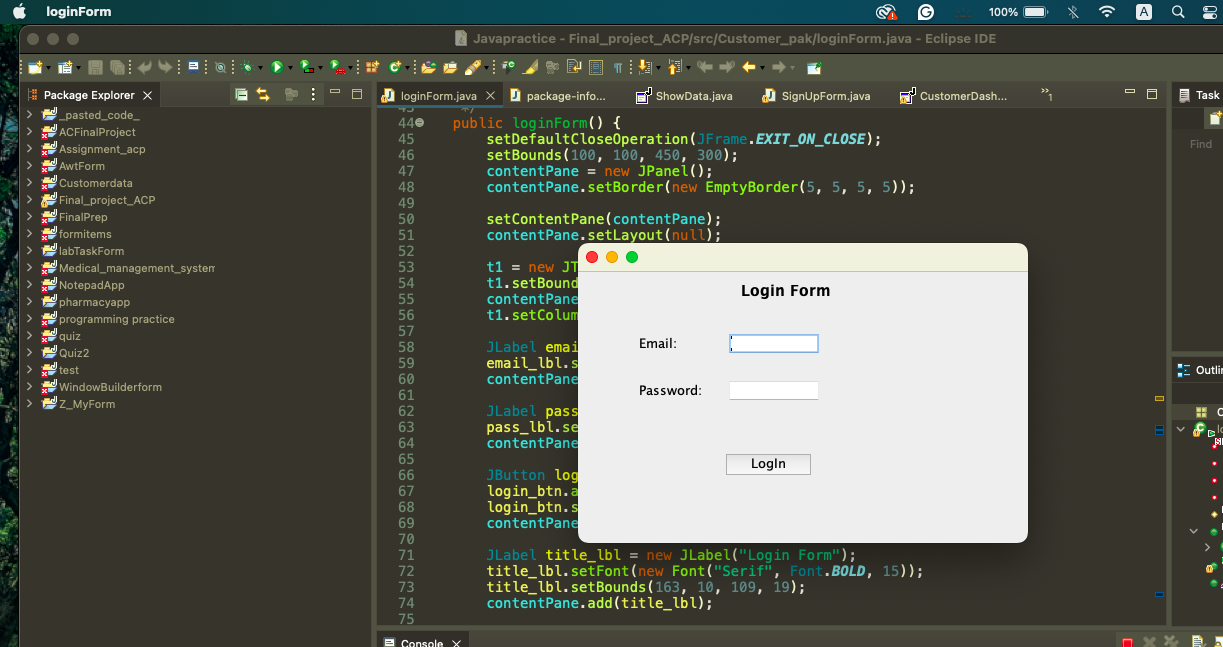
* + The interface includes keyboard shortcuts for all major functions, allowing users to operate the application without a mouse. Text fields and buttons are sized appropriately for users with varying levels of dexterity.

## Screenshots:

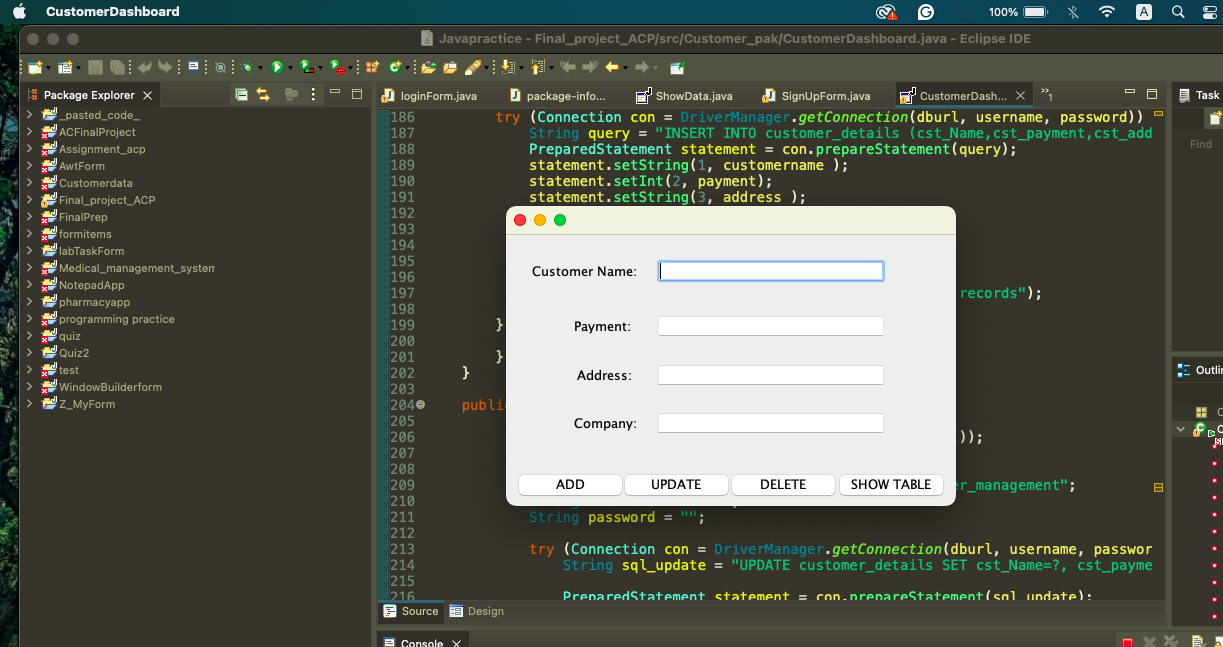
**Signup Window**



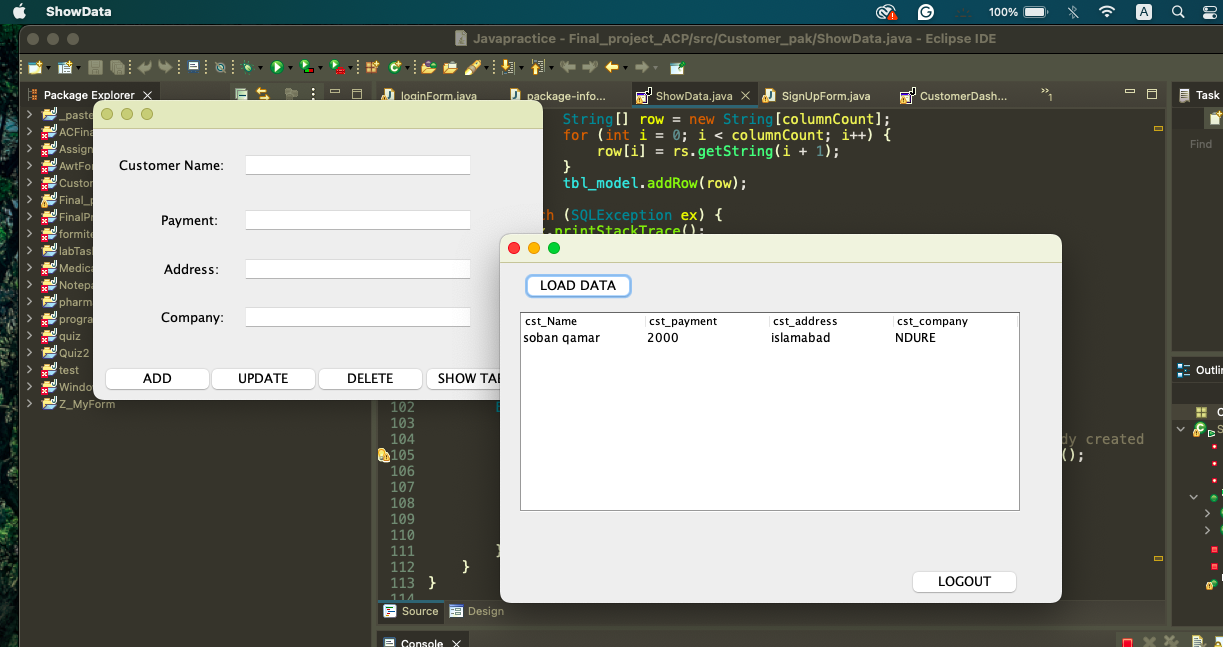
**Login window**



**CustomerDashboard**



**Show Table**



## Navigation Flow:

**Main Window**

**Launch Screen:**

* + **Initial View:** Upon launching the application, the user is presented with the main window that includes signup form.

**Login window**

* + Includes **Email**, **password** and **Login button**.
  + **User Interaction:** Clicking **login** opens a form for customer data entry.

**Data Entry Form**

**Form Interaction:**

* + **Input Fields:** Users fill **Name**, **Payment amount**, **Address**, and **Company Name**.
  + **Buttons:** **Add**, **Update**, **Delete**, and **ShowTable**.
    - **Save:** Validates and add customer data.
    - **Update:** Saves changes to the selected customer record.
    - **Delete:** Removes the current customer record.
    - **ShowTable:** Show the records of customers in table.

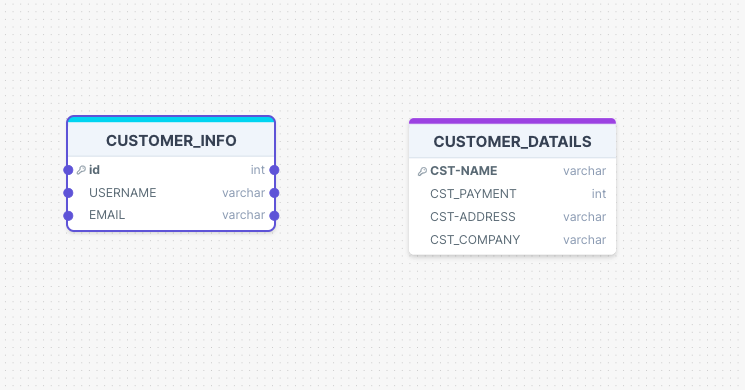
**Details/Information Table**

**Table View:**

* + **Customer Records:** Displays all customer records with sortable columns.
  + **Buttons:** Includes **Loaddata** and **Logout** to exit the application or add new records.

# Database Design

## Schema Diagram:



## Table Descriptions:

**Customer\_info Table:**

* + **ID** (Primary Key): A unique identifier for each customer.
  + **Name**: The name of the customer.
  + **Email**: The email address of the customer.

**Customer\_details Table:**

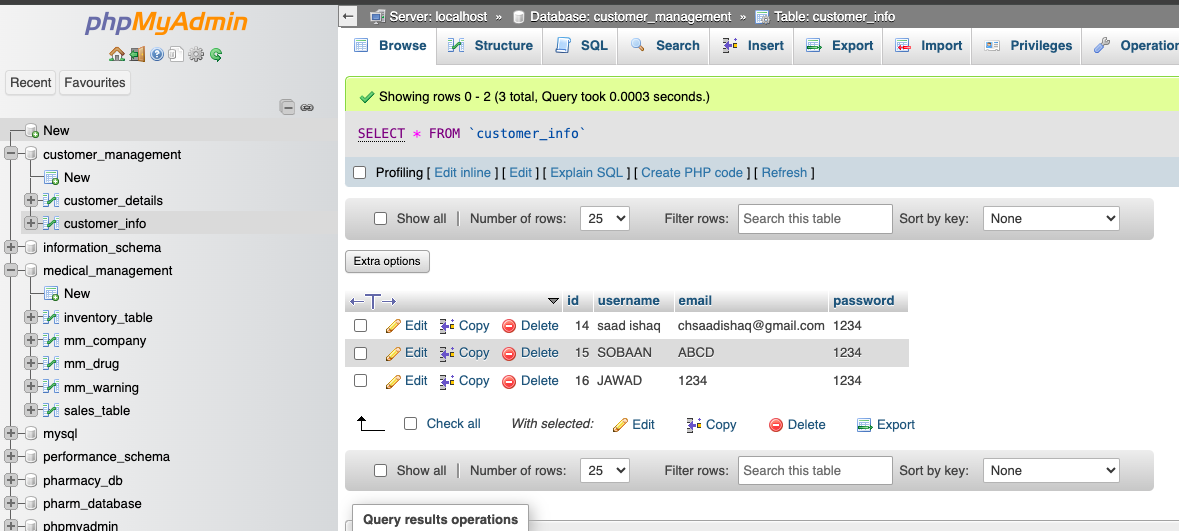
* + **cst\_id** (Foreign Key): A reference to the **ID** in the **Customer\_info** table, establishing a relationship between the tables.
  + **cst\_name**: The name of the customer (redundant if already present in **Customer\_info**, but kept for quick reference).
  + **cst\_payment**: The amount the customer has paid.
  + **cst\_Address**: The address of the customer.
  + **Company**: The name of the company from which the customer purchased things.

**Relationship**

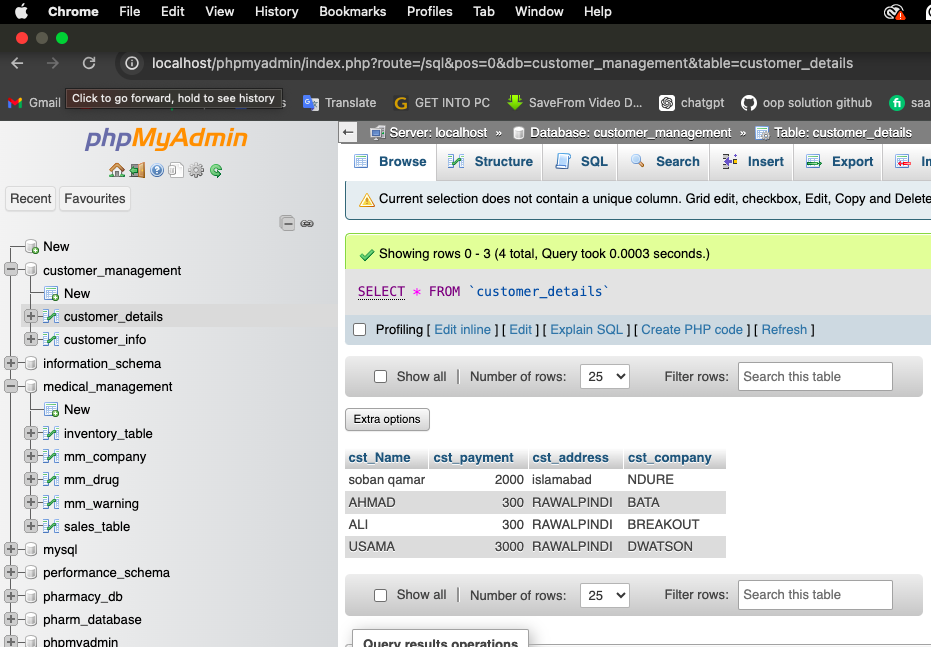
* There is a one-to-many relationship between **Customer\_info** and **Customer\_details**. Each customer in the **Customer\_info** table can have multiple entries in the **Customer\_details** table, which represents different purchases or payment records associated with that customer.

## Sample Data:

**Customer\_info Table:**

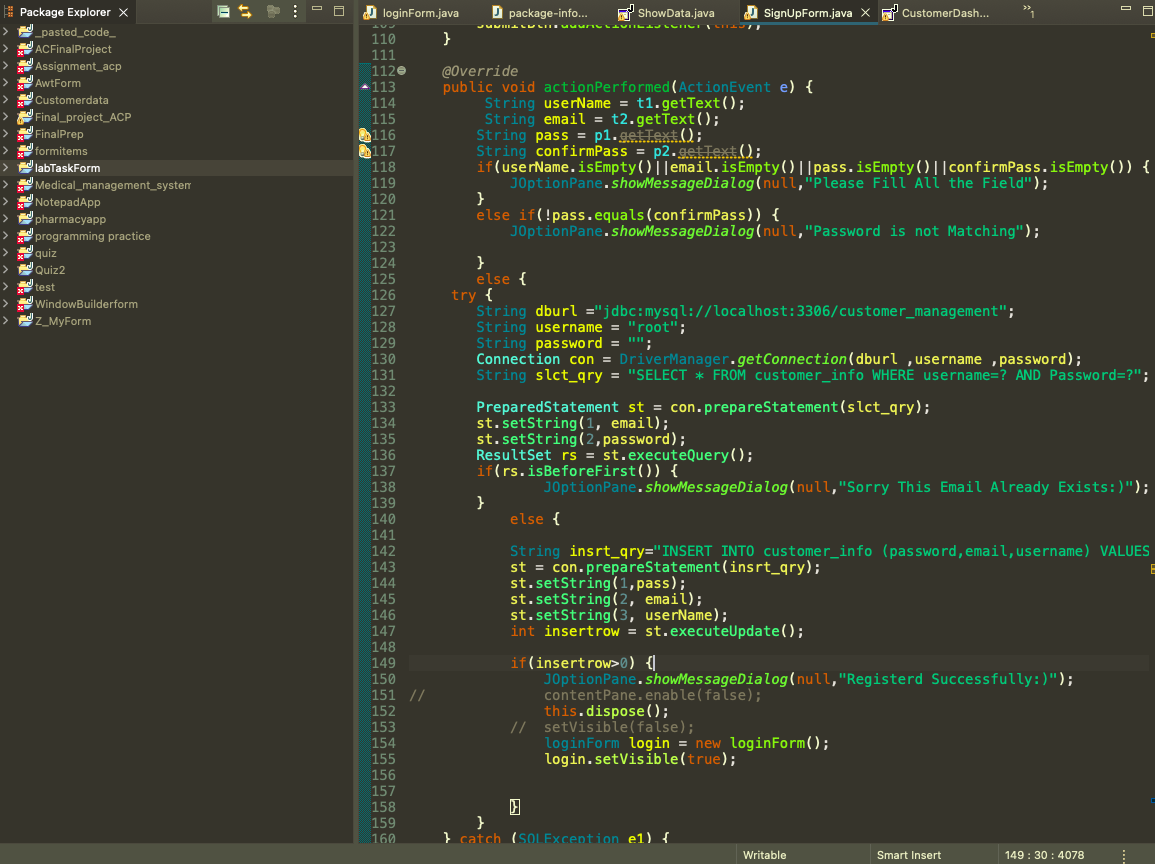


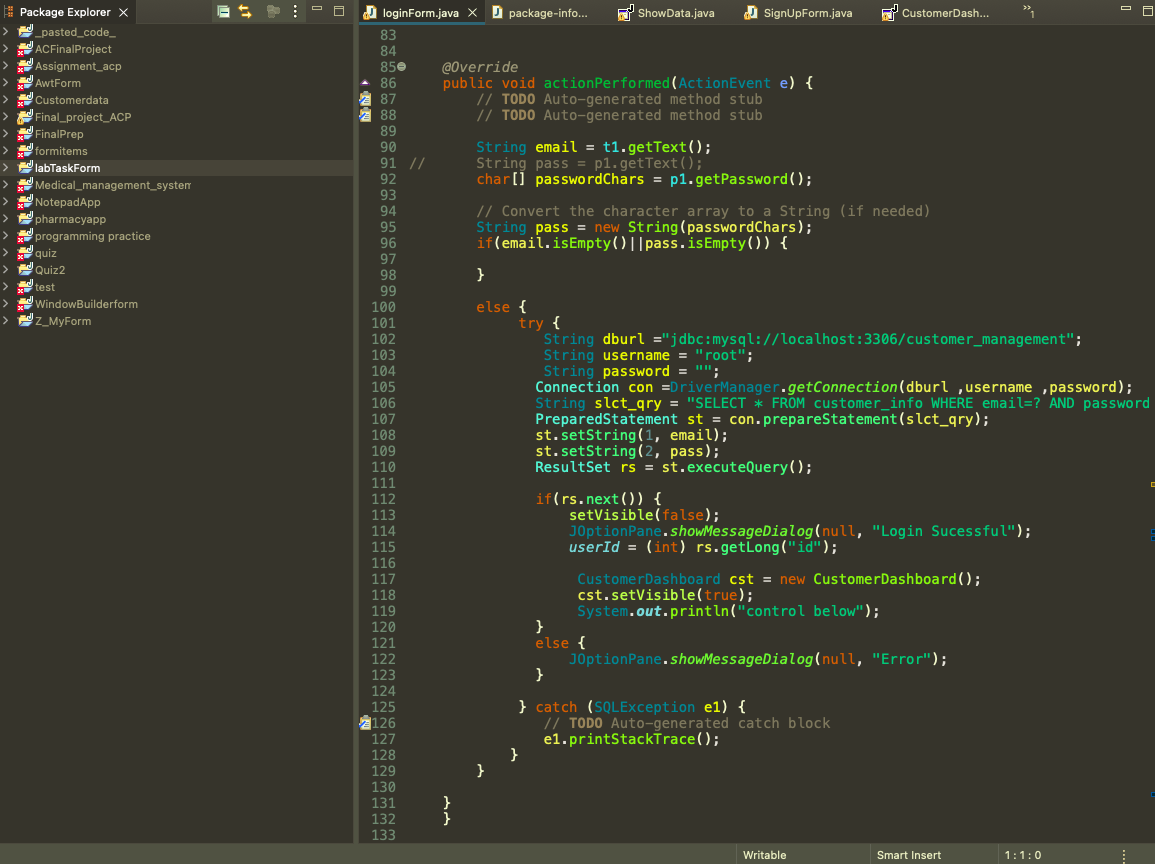
**Customer\_details Table:**

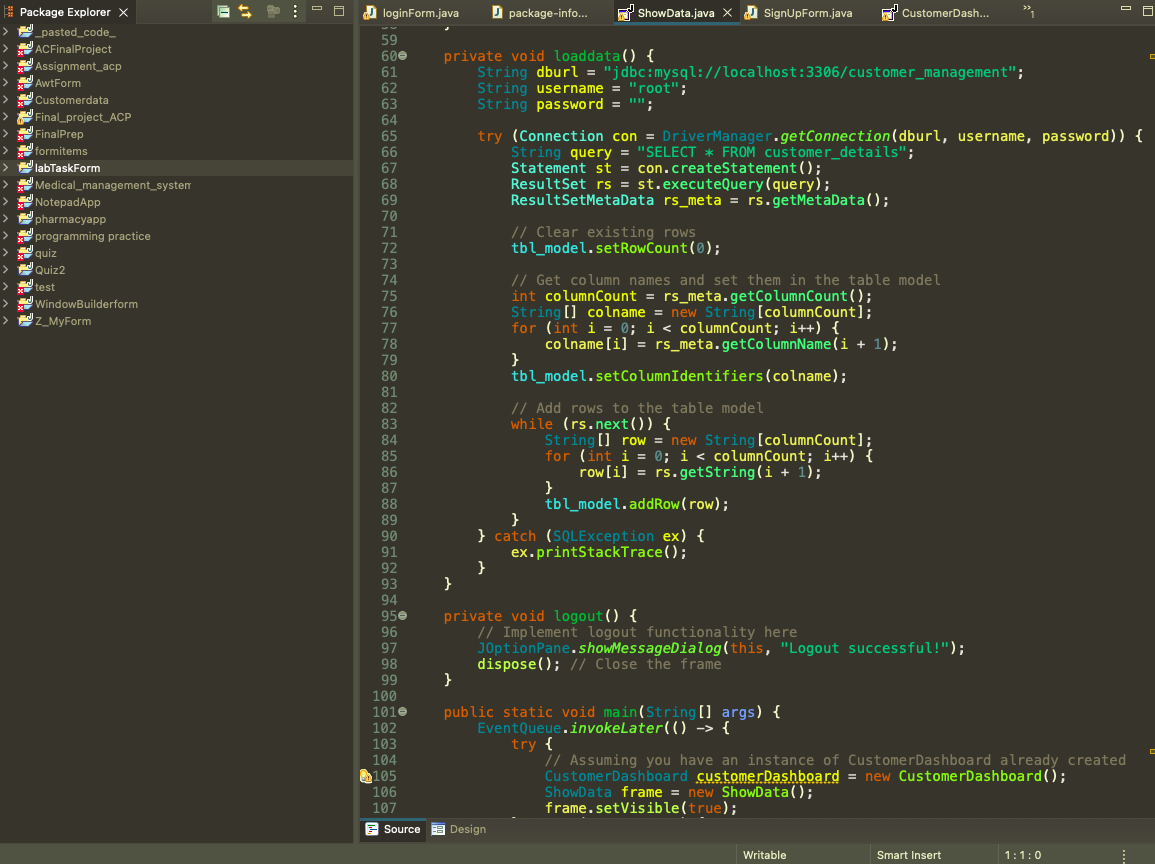


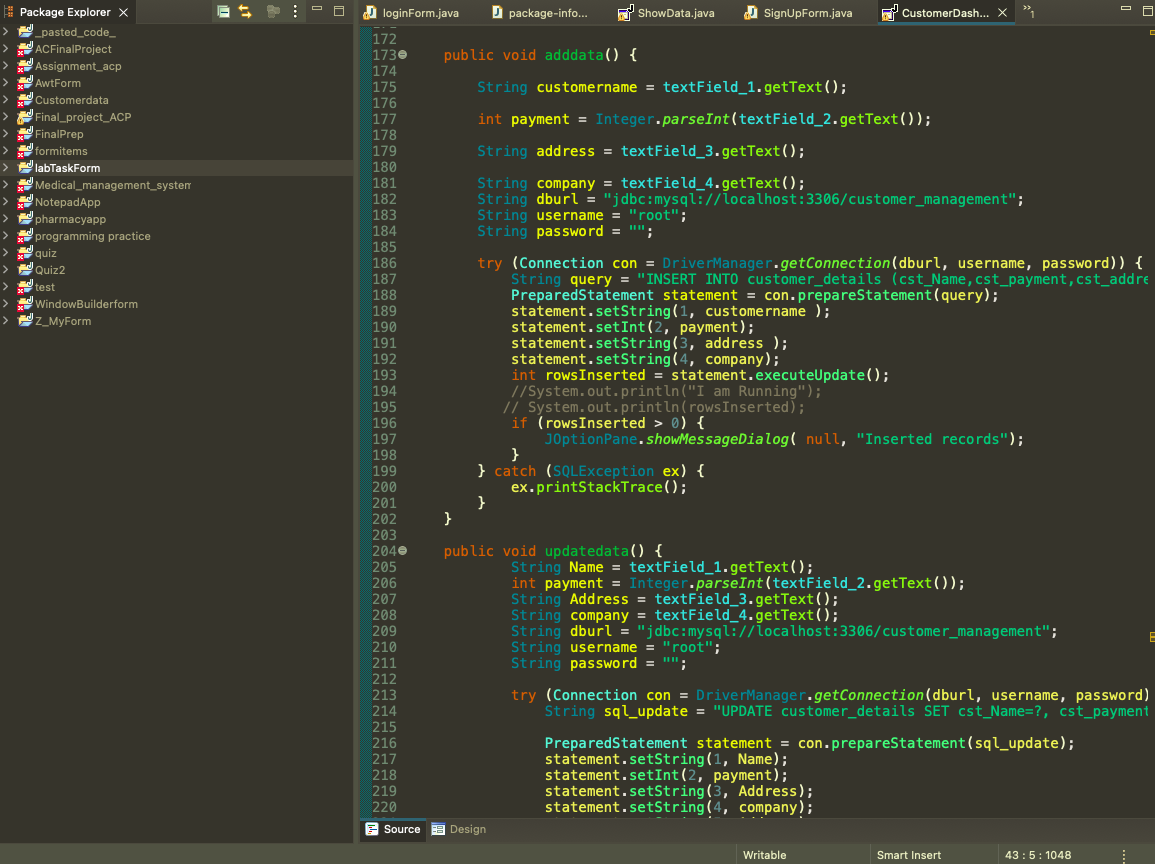
# Implementation

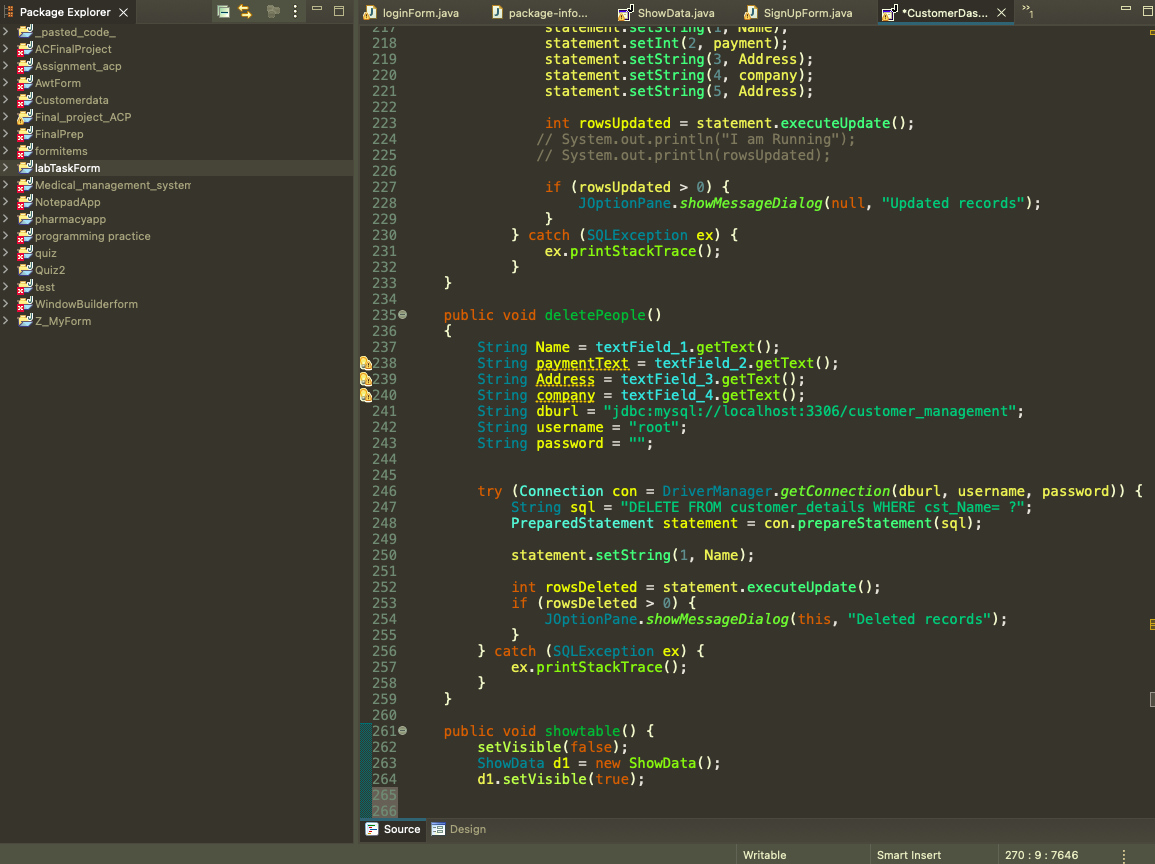
## Code Snippets:











## Explanation:

**Overview of the Application Structure**

The application consists of the following main components:

* **GUI Components:** Built using Swing and AWT, including frames, labels, text fields, buttons, and tables.
* **Data Model:** Represents the customer data and handles CRUD operations.
* **Controller:** Manages the interactions between the GUI and the data model, handling user actions and updating the view.
* **Database Connection:** Manages interactions with the database, executing SQL queries to retrieve and store data.

**Components and Their Interactions**

**GUI Components**

* **Main Frame:**
* The main window of the application is created using **JFrame**.
* Contains a **JButton** for options (**submit, Login**).
* **Panels:**
* **Data Entry Form:** Another **JPanel** with **JLabel** and **JTextField** for customer details (**UserName**, **payment**, **Address**, **Company**).
* **Buttons:**
* **JButton** components for actions like **Add**, **Update**, **Delete**, **ShowTable loadata, and Logout.**

**Data Model**

* **Customer Class:**
* Represents a customer with fields for **ID**, **Name**, **Email**, **Amount Paid**, **Address**, and **Company**.
* Includes getters and setters for each field.
* **CustomerDAO Class:**
* Handles database operations (CRUD) for customer data.
* Contains methods to add, update, delete, and fetch customer records from the database.
* **Controller Class:**
* Manages interactions between the GUI and the data model.
* Handles user actions such as button clicks and form submissions.
* Updates the view based on changes in the model.

**How They Work Together**

* **Initialization:**
  + The main application class (**CustomerManagementApp**) sets up the GUI components and initializes the **CustomerController**.
* **User Interaction:**
  + Users interact with the GUI by filling out forms, clicking buttons, and navigating through menus.
  + The **CustomerController** listens for these interactions and performs the corresponding actions.
* **Data Handling:**
  + The **CustomerController** interacts with the **CustomerDAO** to perform CRUD operations on the database.
  + For example, when the user clicks **Save**, the controller collects data from the form, creates a **Customer** object, and calls **addCustomer** on the **CustomerDAO**.
* **Updating the View:**
  + After performing a data operation, the controller refreshes the table view to reflect the latest data.
* **Database Interaction:**
  + The **CustomerDAO** class manages the database connection and executes SQL queries to manipulate customer data.
  + It ensures that data is correctly stored and retrieved from the database.

# Conclusion

## Summary:

**What I Learned**

**Java Swing and AWT:**

* + Gained hands-on experience with creating and managing GUIs using Java Swing and AWT components.
  + Learned how to layout different panels, buttons, forms, and tables to create a user interface.

**MVC Design Pattern:**

* + Understood the importance of separating the application's data, user interface, and control logic.
  + Implemented a clear division between the Model (data handling), View (UI), and Controller (user interactions).

**Database Integration:**

* + Learned to interact with a MySQL database from Java using JDBC.
  + Designed and implemented a normalized database schema, ensuring data integrity and efficient querying.

**User Experience (UX) Considerations:**

* + Focused on creating a responsive user interface.
* **Future Improvements**

**Enhanced Validation and Error Handling:**

* Implement more robust input validation to prevent incorrect data entry.
* Improve error handling to provide more detailed and user-friendly error messages.

**Advanced Search and Filter Options:**

* + Implement advanced search functionalities with multiple criteria.
  + Add filters to the data table for better data management and retrieval.

**User Authentication and Permissions:**

* + Introduce user authentication to secure access to the application.
  + Implement role-based permissions to restrict certain actions based on user roles.

**UI Enhancements:**

* + - Improve the UI design with modern components and layouts.
    - Make the application responsive to different screen sizes and resolutions.

**Performance Optimization:**

* + Optimize database queries and improve data handling to enhance application performance.
  + Use caching strategies for frequently accessed data to reduce database load.

# Appendices

## User Manual:

## <https://drive.google.com/drive/folders/1-MB64q_3LhnrA1Wicepgkuxp2pidOZ01?usp=sharing>

## Source Code:

<https://drive.google.com/drive/folders/1GGCecaUi7F2FXoMfB4_DtN6ij5ySEKXs?usp=drive_link>

## Live Project Demo:

<https://drive.google.com/file/d/1k_pEYBBEjCcK7Iiv2xrPshVp1WVAfWHF/view?usp=drive_link>