

International Islamic University, Islamabad

User and Note Management Application

PROJECT

JAVA

MAY 30, 2024

Osama Malik | 4319-BSSE-F21-A

Advanced Computer Programming

Mr. Zohair Ahmad

Contents

Chapter 01 03

[Chapter 02 06](#_Toc168174361)

Chapter 03 13

[Chapter 04 16](#_Toc168174361)

Chapter 05 23

**CHAPTER 01**

**INTRODUCTION**

* 1. **Project Overview**

The project, titled "User and Note Management Application," is a desktop application developed to manage users and their associated notes. The application provides an intuitive interface for users to create accounts, log in, and manage their personal notes efficiently. The primary purpose of this application is to simplify the process of note-taking and organization, ensuring users can easily create, view, update, and delete their notes.

* 1. **Purpose:**

The main objective of the User and Note Management Application is to offer a simple, user-friendly platform for managing notes. Users can create an account, log in to the application, and maintain a personal collection of notes. This tool is especially beneficial for individuals who need to organize their thoughts, tasks, and important information in one accessible place.

* 1. **Main Features:**
  + **User Management:** Users can create new accounts with a unique username and password.
  + **Note Management:** Logged-in users can create new notes, view existing notes, edit note content, and delete unwanted notes.
  + **User Interface:** A straightforward and easy-to-navigate graphical user interface (GUI) designed using Java Swing.
  + **Database Integration:** Secure storage of user data and notes using a MySQL database to ensure data persistence and reliability. We achieve this using DAO design pattern.
  1. **Target Users:**

The application is designed for a limited range of users who need a reliable and straightforward note-taking solution. This includes students, professionals, writers, and anyone who requires a digital platform to organize their notes effectively.

* 1. **Technologies Used**

The development of the User and Note Management Application involved several key technologies and tools:

* + **Java:** The core programming language used for the application's logic and functionality.
  + **Java Swing:** A part of Java Foundation Classes (JFC), used for building the graphical user interface (GUI) of the application. Swing provides a set of 'lightweight' (all-Java language) components that work the same on all platforms.
  + **Abstract Window Toolkit (AWT):** Used alongside Swing to create GUI components and handle events.
  + **Java Database Connectivity (JDBC):** An API used to connect and execute queries with the MySQL database.
  + **MySQL:** The relational database management system used to store user information and notes securely.
  + **Integrated Development Environment (IDE):** Used Eclipse for writing and debugging the code.

**CHAPTER 02**

**GUI DESIGN**

**2.1. Design of the Graphical User Interface (GUI)**

The User and Note Management Application features a user-friendly graphical user interface (GUI) designed with Java Swing and Abstract Window Toolkit (AWT). The design focuses on simplicity, ease of use, and an intuitive layout to enhance the user experience.

**2.2. Layout Design:**

* + **Main Menu:** The application opens with a main menu that provides two primary options: "Add User" and "Open Notepad." This menu is designed using a GridBagLayout for flexible and responsive positioning of buttons.
  + **User Management Screen:** This screen allows users to create new accounts. It features text fields for entering a username and password, arranged in a simple grid layout for clarity and ease of use.
  + **Notepad Screen:** The notepad screen is divided into two main panels:
  + **Notes List Panel:** A vertical panel on the left side displaying the list of notes. This panel uses a BorderLayout and includes a JList component to list notes and buttons for adding, viewing, and deleting notes. Background of this is Blue.
  + **Note Content Panel:** A larger panel on the right side for displaying and editing the content of the selected note. This area includes a JTextArea for note content, allowing users to write and edit their notes freely.

**2.3. Design Patterns:**

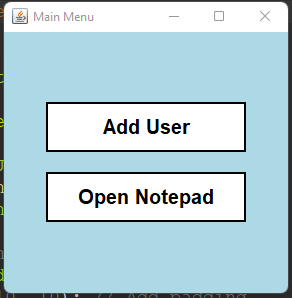
Model-View-Controller (MVC): The application adheres to the MVC design pattern to separate concerns:

* + **Model:** Represents the data layer, including User and Note classes and their corresponding DAO classes for database operations.
  + **View:** Handles the user interface components built using Swing and AWT.
  + **Controller:** Manages the logic and user input, acting as an intermediary between the model and view.

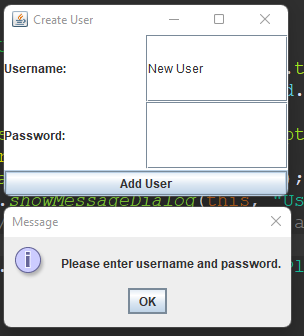
**2.4. User Experience (UX):**

* + **Consistency:** The application maintains a consistent look and feel across different screens, using similar color schemes, fonts, and button styles.
  + **Responsiveness:** The layout adapts to various screen sizes, ensuring a smooth experience on different devices.
  + **Feedback:** Interactive elements like buttons provide visual feedback on user actions, enhancing the usability of the application.
  + **Simplicity:** The interface avoids clutter, focusing on essential functionalities to make the application easy to navigate and use.
  + **Screenshots:**

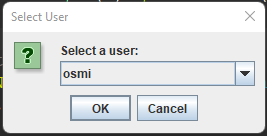
Main Menu

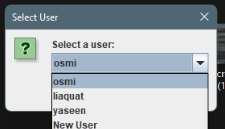
****

Login Screen

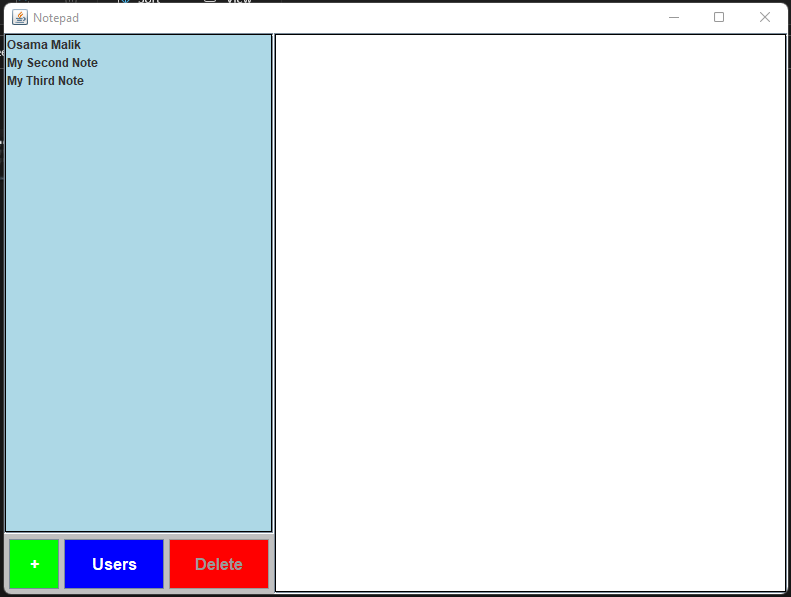


User Selection Screen

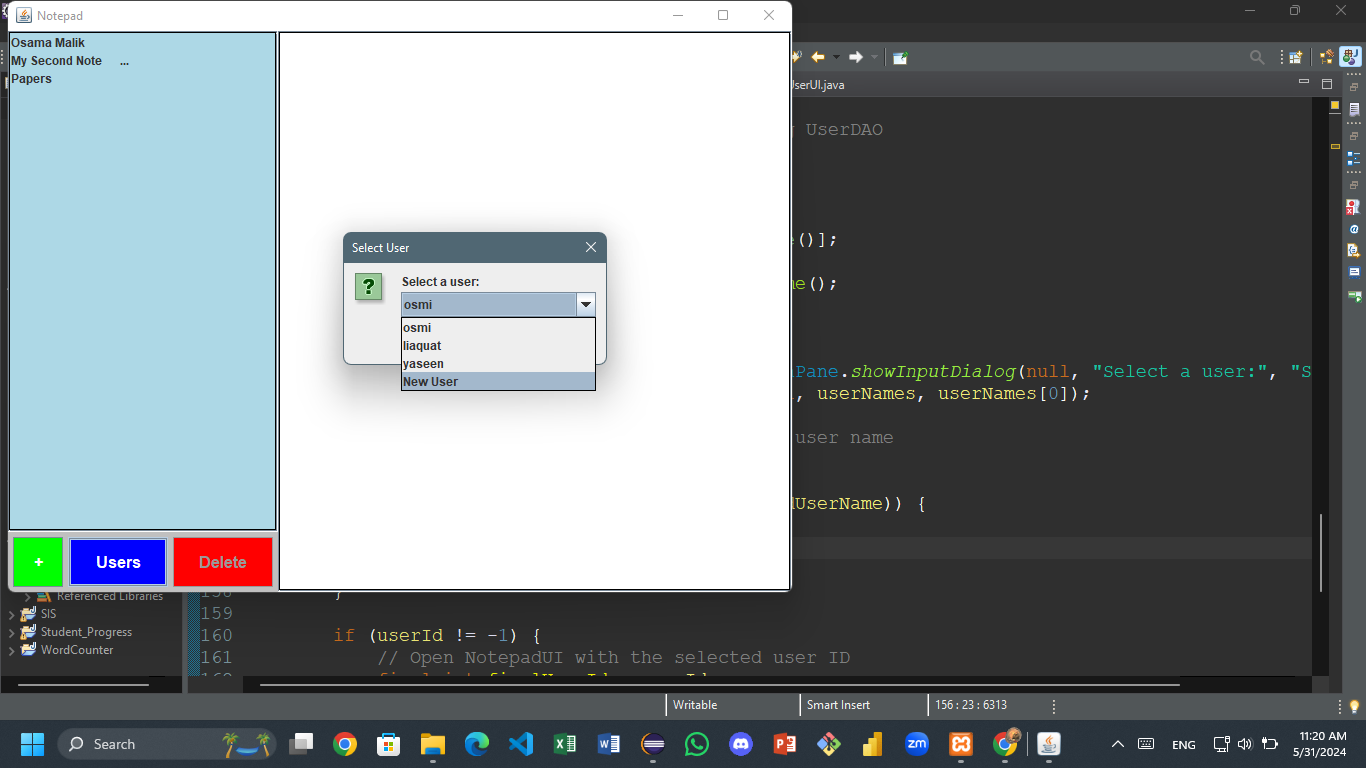


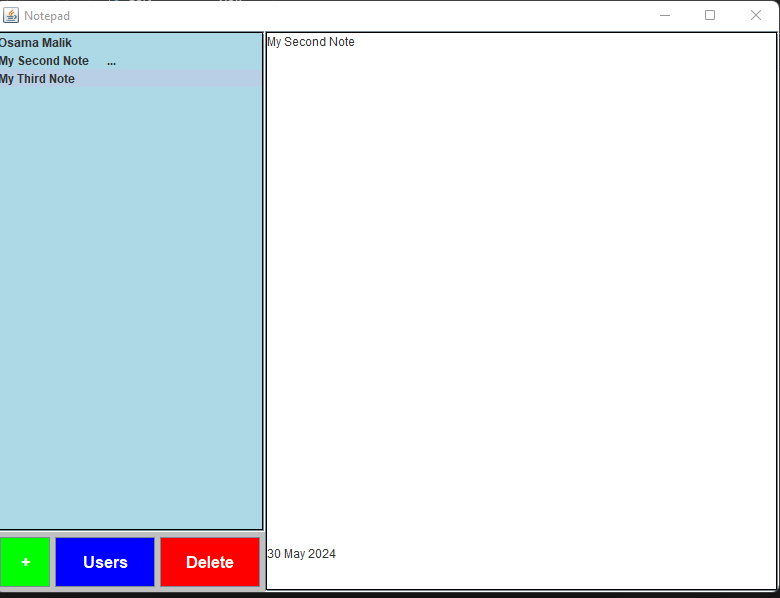


Notepad Screen



Notepad Screen after clicking User Button





**2.5. Navigation Flow**

The navigation flow of the User and Note Management Application is designed to be intuitive and straightforward, guiding users through a series of logical steps:

* + **Main Menu:** Upon launching the application, users are presented with the main menu screen.
  + **Add User Button:** Clicking this button navigates users to the User Management screen.
  + **Open Notepad Button:** Clicking this button opens a dialog to select a user and then navigates to the Notepad screen for the chosen user.
  + **User Management Screen:**

Users can create a new account by entering a username and password and clicking the "Add User" button.

Upon successful account creation, users receive a confirmation message, and the screen closes, returning them to the main menu.

* + **Notepad Screen:**

The notepad screen displays after selecting a user from the dialog that appears when "Open Notepad" is clicked in the main menu.

* + **Notes List Panel:**

Add Note Button ("+"): Clicking this button adds a new note to the list and selects it for editing.

* + **View Users Button:** Opens a dialog to select a different user.
  + **Delete Note Button:** Deletes the selected note from the list.
  + **Note Content Panel:**

Users can type and edit the content of the selected note in the JTextArea.

Changes are automatically saved as users interact with the note.

* + **Interaction Flow:**

**Creating a User:** Users can easily create an account from the main menu and are guided back to the main menu upon completion.

* + **Managing Notes:** Users can navigate between notes, add new notes, delete existing ones, and edit note content within the notepad screen without needing to switch back to the main menu.
  + **User Selection:** The option to switch users allows for flexible management of notes across different accounts, making it suitable for multiple users on a single system.

**CHAPTER 03**

**DATABASE DESIGN**

#### 3.1. Schema Diagram

#### 

|  |
| --- |
| Users |
| id (INT, PK, AI)username (VARCHAR)password (VARCHAR) |

|  |
| --- |
| Notes |
| id (INT, PK, AI)user\_id (INT, FK)content (TEXT) |

#### 

#### MULTIPLICITY: One Users have many Notes.

#### 3.2. Table Descriptions

1. **users**

* **Purpose:** Stores information about the users of the application.
* **Columns:**
* id (INT, PRIMARY KEY, AUTO\_INCREMENT): Unique identifier for each user.
* username (VARCHAR(50), NOT NULL): Username of the user.
* password (VARCHAR(50), NOT NULL): Password of the user, stored in plain text (note: this should be hashed in a real application for security).

1. **notes**

* **Purpose:** Stores the notes created by users.
* **Columns:**
* id (INT, PRIMARY KEY, AUTO\_INCREMENT): Unique identifier for each note.
* user\_id (INT, FOREIGN KEY, NOT NULL): Identifier of the user who created the note, references users(id).
* content (TEXT, NOT NULL): The content of the note.

#### 3.3. Sample Data

Here is some example data that might be stored in the tables:

Users

|  |  |  |
| --- | --- | --- |
| **id** | **username** | **password** |
| 1 | osmi | 123 |
| 2 | liaquat | 1234 |
| 3 | yaseen | 12345 |
| 4 | New User | 2222 |

Notes

|  |  |  |
| --- | --- | --- |
| **id** | **user\_id** | **content** |
| 1 | 1 | Osama Malik |
| 2 | 1 | My Second Note |
| 3 | 3 | Yaseen Notes notes Auto update funtionality |

**CHAPTER 04**

**IMPLEMENTATION**

**4.1. Code Snippets**

**Database Connection**

**DBConnection.java**

public class DBConnection {

private static final String URL = "jdbc:mysql://localhost:3306/notepad";

private static final String USER = "root";

private static final String PASSWORD = "";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

}

**Explanation:**

* **Purpose:** Establishes a connection to the MySQL database.
* **Logic:**

The getConnection() method uses DriverManager.getConnection() to connect to the database with the specified URL, username, and password.

The method returns a Connection object, which is used in other parts of the application to execute SQL queries.

**User Management (CRUD Operations)**

**UserDAO.java**

public class UserDAO {

public void createUser(String username, String password) {

String query = "INSERT INTO users (username, password) VALUES (?, ?)";

try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, username);

stmt.setString(2, password);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public User getUserById(int userId) {

User user = null;

String query = "SELECT \* FROM users WHERE id = ?";

try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, userId);

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

if (rs.next()) {

user = new User();

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return user;

}

public List<User> getAllUsers() {

List<User> users = new ArrayList<>();

String query = "SELECT \* FROM users";

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

while (rs.next()) {

User user = new User();

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

users.add(user);

}

} catch (SQLException e) {

e.printStackTrace();

}

return users;

}

}

**Explanation:**

* **Purpose:** Provides CRUD operations for user management.
* **Logic:**

**createUser:** Inserts a new user into the users table using a prepared statement to prevent SQL injection.

**getUserById:** Retrieves a user by their ID. Executes a SELECT query and maps the result set to a User object.

**getAllUsers:** Retrieves all users from the users table and returns a list of User objects.

**Note Management (CRUD Operations)**

**NoteDAO.java**

public class NoteDAO {

public List<Note> getNotesByUserId(int userId) {

List<Note> notes = new ArrayList<>();

String query = "SELECT \* FROM notes WHERE user\_id = ?";

try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, userId);

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

while (rs.next()) {

Note note = new Note();

note.setId(rs.getInt("id"));

note.setUserId(rs.getInt("user\_id"));

note.setContent(rs.getString("content"));

notes.add(note);

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return notes;

}

public void createNoteForUser(int userId, String content) {

String query = "INSERT INTO notes (user\_id, content) VALUES (?, ?)";

try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, userId);

stmt.setString(2, content);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void deleteNoteById(int noteId) {

String query = "DELETE FROM notes WHERE id = ?";

try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, noteId);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void updateNoteContent(int noteId, String content) {

String query = "UPDATE notes SET content = ? WHERE id = ?";

try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, content);

stmt.setInt(2, noteId);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

**Explanation:**

* **Purpose:** Provides CRUD operations for note management.
* **Logic:**

**getNotesByUserId:** Retrieves all notes for a specific user, mapping the result set to a list of Note objects.

**createNoteForUser:** Inserts a new note for a user using a prepared statement.

**deleteNoteById:** Deletes a note by its ID.

**updateNoteContent:** Updates the content of a note by its ID.

**User Authentication**

**UserUI.java**

public class UserUI extends JFrame {

private JTextField usernameField;

private JPasswordField passwordField;

public UserUI() {

initializeUI();

}

private void initializeUI() {

setTitle("Create User");

setSize(300, 200);

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

setLayout(new BorderLayout());

getContentPane().setBackground(Color.WHITE);

JPanel inputPanel = new JPanel(new GridLayout(2, 2));

inputPanel.setBackground(Color.WHITE);

JLabel usernameLabel = new JLabel("Username:");

usernameField = new JTextField();

JLabel passwordLabel = new JLabel("Password:");

passwordField = new JPasswordField();

inputPanel.add(usernameLabel);

inputPanel.add(usernameField);

inputPanel.add(passwordLabel);

inputPanel.add(passwordField);

JButton createUserButton = new JButton("Add User");

createUserButton.addActionListener(e -> createUser());

add(inputPanel, BorderLayout.CENTER);

add(createUserButton, BorderLayout.SOUTH);

}

private void createUser() {

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

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

if (!username.isEmpty() && !password.isEmpty()) {

UserDAO userDAO = new UserDAO();

userDAO.createUser(username, password);

JOptionPane.showMessageDialog(this, "User created successfully!");

dispose();

} else {

JOptionPane.showMessageDialog(this, "Please enter username and password.");

}

}

}

**Explanation:**

* **Purpose:** Provides the UI and logic for user registration.
* **Logic:**

**initializeUI:** Sets up the UI components, including text fields for username and password, and a button to create a user.

**createUser:** Handles the action of creating a user. It retrieves the entered username and password, checks if they are not empty, and calls UserDAO.createUser to insert the new user into the database. It shows a confirmation dialog upon success or an error message if the input fields are empty.

**4.2. Explanation of Logic and Integration**

The User and Note Management Application is built using a combination of Java Swing for the UI, JDBC for database interactions, and MySQL for data storage. The application follows the Model-View-Controller (MVC) design pattern to ensure separation of concerns and maintainability.

* **Model (Data Layer):** The User and Note classes represent the data entities, while the UserDAO and NoteDAO classes handle database operations for these entities. The DAOs use JDBC to perform CRUD operations, ensuring secure and efficient data access.
* **View (UI Layer):** The GUI is built using Java Swing components, with classes like Main, UserUI, and NotepadUI defining the layout and behavior of different screens. Swing's layout managers (e.g., GridBagLayout, BorderLayout, GridLayout) are used to arrange UI components in a user-friendly manner.
* **Controller (Logic Layer):** The controller logic is embedded in the action listeners of the Swing components. For instance, in UserUI, the createUser() method acts as the controller, processing user input and interacting with the UserDAO to create a new user in the database.

**Interaction Flow:**

* **Main Menu:** Users start at the main menu, where they can choose to create a new user or manage notes.
* **User Creation:** In the UserUI, users enter their details and create an account, which is saved in the MySQL database via UserDAO.createUser().
* **Notepad:** In the NotepadUI, users can view, create, delete, and edit notes. The notes are loaded from the database using NoteDAO.getNotesByUserId(), and changes are saved back to the database with methods like createNoteForUser(), deleteNoteById(), and updateNoteContent().

**CHAPTER 05**

**Conclusion**

**5.1. Summary**

The User and Note Management Application is a robust, user-friendly desktop application designed to facilitate efficient management of users and their notes. Through the development of this project, I have gained valuable experience in software development, from designing user interfaces to implementing database operations and applying design patterns.

**Project Overview:**

* **Purpose:** To provide a simple and effective tool for users to manage their notes, with functionalities for user account creation and personalized note management.
* **Technologies Used:** Java, JSwing, AWT, JDBC, and MySQL.
* **Features:** User creation, note creation, viewing, updating, and deletion, all integrated into a cohesive user interface.

**Key Learnings:**

* **Java Programming:** Enhanced proficiency in Java, particularly in creating GUI applications with Swing and AWT.
* **Database Management:** Gained practical experience with JDBC and MySQL for database connectivity and CRUD operations.
* **Design Patterns:** Applied the Model-View-Controller (MVC) design pattern to separate concerns and improve the maintainability of the codebase.
* **User Interface Design:** Learned how to design intuitive and responsive user interfaces that enhance user experience.
* **Error Handling:** Implemented error handling to manage database connectivity issues and user input validation, improving the application's robustness.

**Future Improvements:**

* **User Authentication:** Implementing a login system to secure user accounts and prevent unauthorized access.
* **Data Encryption:** Enhancing security by encrypting user passwords and sensitive data stored in the database.
* **Search Functionality:** Adding a search feature to allow users to quickly find specific notes.
* **Rich Text Editing:** Upgrading the note-taking area to support rich text formatting, allowing users to style their notes with bold, italics, and other formatting options.
* **Mobile Version:** Developing a mobile version of the application to provide users with the convenience of accessing their notes on the go.
* **Cloud Sync:** Implementing cloud synchronization features to allow users to access their notes from multiple devices seamlessly.

In conclusion, the User and Note Management Application serves as a comprehensive tool for managing personal notes and user accounts. The project has provided me with a solid foundation in desktop application development, database management, and UI design, and has identified several areas for future enhancement to further improve functionality and user experience.

**SOURCE CODE and other Materials:**

[**https://github.com/Osmi-Bytes/Java-Project.git**](https://github.com/Osmi-Bytes/Java-Project.git)

**Runnable Jar file and SQL File:**

[**https://drive.google.com/drive/folders/1oZLkXYJ1rA39rAJc-eOn95rr49u9Fyvv?usp=drive\_link**](https://drive.google.com/drive/folders/1oZLkXYJ1rA39rAJc-eOn95rr49u9Fyvv?usp=drive_link)