

# **MINI PROJECT REPORT**

## **Group Creating System**

PROJECT BY:

DARSH MARU (32)

JAYRAJ KAMLIYA (26)

MANAS MORE (35)

# 1. INTRODUCTION

**Our Project** refers to a system designed to efficiently store, organize, and manage information about members within various groups. This system can be applied in numerous contexts, such as organizing teams in a company, participants in an event, or members of clubs and organizations.

The purpose of this project is to create a structured database that allows for the secure storage and retrieval of member information, ensuring that it can be accessed and managed easily. The system typically includes functionality for adding, removing, updating, and viewing member data, which includes details like names, contact information, roles, and group affiliations.

## 1.1 Importance of Group Members Storage:

- **Data Organization:** It ensures that information about members is organized, making it easy to track and manage groups of varying sizes.
- **Efficiency:** Automating the process of storing and retrieving member details reduces manual effort and the risk of errors.
- **Scalability:** As groups grow, it becomes essential to have a system that can handle larger amounts of data without losing efficiency.
- **Data Security:** Storing member information in a secure system protects sensitive data and ensures privacy.

## 1.2 Scope of the Project:

This project focuses on developing a system that can:

- Store basic member information such as name, email, phone number, and role within the group.
- Allow for the creation and management of different groups.
- Enable the addition, modification, and removal of members from groups.
- Provide easy access to stored information, with options to filter or search members within groups.
- Offer a user-friendly interface for interaction with the system.

### **1.3 Objectives of the Project:**

- To develop a robust database for storing group member information.
- To implement a user interface that simplifies data input, modification, and retrieval.
- To ensure data integrity and security during storage and access.
- To make the system scalable for managing large groups without performance issues.
- To provide an intuitive and efficient solution for organizations or individuals needing to manage group-related information.

## **2. PROBLEM DEFINITION**

The primary problem being solved with the **PROJECT** system is the inefficiency and challenges in manually managing group member information, especially as groups grow larger. When member data is stored in disorganized ways—such as in spreadsheets, notebooks, or fragmented files—finding, updating, and organizing that information becomes tedious and error-prone. This can lead to issues such as:

- **Data Inconsistency:** Manually maintaining records can result in inconsistencies, such as duplicate entries or outdated information.
- **Inefficient Access:** Searching for specific member details across unstructured formats can take excessive time and effort, especially in large groups.
- **Limited Scalability:** As the group expands, it becomes increasingly difficult to maintain and retrieve information manually, resulting in inefficient operations.
- **Lack of Data Security:** Inadequate storage systems increase the risk of unauthorized access or data breaches, which can compromise sensitive member information.
- **Difficulty in Updating Information:** Manually updating group information (such as when members join, leave, or change roles) can lead to errors or missed updates.

### 3. MODULES

The **Group Members Storage** system can be broken down into several key modules, each responsible for handling specific functions related to the management and organization of group member information. The following sections provide an overview of each module:

#### 3.1 User Management Module:

This module handles the administration of system users, particularly useful in multi-user environments such as organizations. It ensures that only authorized personnel can access and manipulate the group member data.

##### Key Features:

- **User Authentication:** Secure login functionality to ensure only registered users can access the system.
- **Role-Based Access Control (RBAC):** Assign different access levels based on user roles (e.g., Admins can modify or delete member data, while regular users can only view).
- **Profile Management:** Allow users to manage their own account details, such as passwords or personal information.

##### Benefits:

- Ensures data security and prevents unauthorized access to sensitive group information.
- Enables collaboration within teams by assigning appropriate user permissions.

#### 3.2 Group Creation and Management Module:

This module allows users to create, modify, and manage groups within the system. It is particularly useful when the system needs to handle multiple groups or teams within an organization.

##### Key Features:

- **Create New Group:** Users can create new groups, assigning a group name, description, and possibly other attributes such as group type or category.
- **Edit/Delete Groups:** Authorized users can modify group details or delete groups no longer in use.
- **Group Categorization:** Users may categorize groups based on criteria like team type, department, or project focus for easier navigation.

**Benefits:**

- Provides a clear organizational structure, making it easy to manage different groups.
- Enhances the system's ability to handle multiple independent groups simultaneously.

### 3.3 Member Storage Module:

The core of the system, this module is responsible for storing and organizing information about individual group members. It allows for efficient management of member data, ensuring it is stored securely and can be accessed easily.

**Key Features:**

- **Add Members:** Users can input new member information, including details like name, email, phone number, role, and any other relevant fields.
- **Edit Member Information:** Users can update existing member details to reflect any changes (e.g., role changes, contact updates).
- **Delete Members:** Allows the removal of members from the system when they are no longer part of a group.
- **Member Search/Filter:** Users can search or filter group members based on specific criteria such as name, role, or group affiliation.

**Benefits:**

- Simplifies the process of adding and updating group members.
- Ensures all member data is stored in a structured and secure manner, which enhances data retrieval.

### 3.4 Database Management Module

This module is responsible for maintaining the structure of the underlying database where all group and member data is stored. It ensures data integrity and supports smooth interaction between the system and the stored data.

**Key Features:**

- **Database Structure Management:** Manages the creation and update of database tables for storing group and member data (e.g., relational tables linking group IDs to member IDs).
- **Data Validation:** Ensures that inputted member and group data is valid and conforms to predefined standards (e.g., email format, mandatory fields).

- **Backup and Recovery:** Provides functionality for regularly backing up data and restoring it in case of system failure.

**Benefits:**

- Ensures the integrity and reliability of data stored in the system.
- Reduces the risk of data loss through regular backup operations.

## **6. Results and Display Module**

This module handles the presentation of the stored data to the users, ensuring that the system interface allows for easy navigation and interaction.

**Key Features:**

- **Display Group List:** Users can view a list of all available groups in the system and select one to view its members.
- **Display Member List:** Once a group is selected, this feature displays all members of that group in an organized format, along with their key details.
- **Sorting and Filtering Options:** Users can sort members by name, role, or other criteria, and filter the list to focus on specific subsets of members.

**Benefits:**

- Provides a user-friendly interface for interacting with the stored data.
- Ensures users can quickly find and manage the information they need without difficulty.

These modules, working together, create a comprehensive **Group Members Storage** system that efficiently manages the storage, organization, and security of group-related data. Each module is designed to address a specific aspect of the system, ensuring smooth operation and scalability as the system evolves or expands.

## 4. DATABASE

**Login and Registration Database (logininfo):**

The screenshot shows a Java IDE with a MySQL connection established. The connection string is `jdbc:mysql://localhost:3306/test?zeroDateTimeBehavior=CONVERT_TZ`. A SQL query is entered in the editor:

```
1 SELECT * FROM logininfo LIMIT 100;
2
```

The query results are displayed in a table below the editor. The table has five columns: #, Fname, Lname, Email, and Password. The first two rows of data are visible:

#	Fname	Lname	Email	Password
1	Ayush	Mohite	Ayush	Ayush
2	Manas	More	Manas	1234

The IDE interface includes a toolbar at the top with various icons for file operations, a status bar at the bottom showing 'Output' and 'SQL 3 x', and a right-hand pane for additional information.



**Group Database (groupinfo):**

The screenshot shows a MySQL IDE interface. At the top, the connection string is 'jdbc:mysql://localhost:3306/test?zeroDateTimeBehavior=CONVERT\_TZ'. The SQL editor contains the query: `SELECT ** FROM groupinfo LIMIT 100;¶`. Below the editor, a toolbar shows 'Max. rows: 100' and 'Fetched Rows: 6'. The results pane displays a table with 6 rows and 7 columns: #, groupid, noofmem, date, class, subject, and projecttopic. The data is as follows:

#	groupid	noofmem	date	class	subject	projecttopic
1	1	2	2024-10-20	SE9	OOP	asd
2	2	2	2024-10-20	SE9	OOP	asd
3	3	2	2024-10-20	SE9	OOP	asd
4	4	1	2024-10-20	asda	xca	asd
5	5	2	2024-10-20	SE09	OS	ghghf
6	6	2	2024-10-19	SE09	Java-OOP	Project

At the bottom, there are tabs for 'Output' and 'SQL 3 x'.

### Student Database (studentinfo):

The screenshot shows a Java Swing application window titled "Connection: jdbc:mysql://localhost:3306/test?zeroDateTimeBehavior=CONVERT\_T..." with a standard Windows toolbar. The main text area contains the SQL query: `SELECT * FROM studentinfo LIMIT 100; 1`. Below the text area, a button labeled "SELECT \* FROM studentinfo..." is visible. At the bottom, a status bar displays "Max. rows: 100" and "Fetched Rows: 7". To the right of the status bar, a table displays the query results with 7 rows and 8 columns: #, smno, groupid, sname, smail, pm, batch, rollno, and gender. The table data is as follows:

#	smno	groupid	sname	smail	pm	batch	rollno	gender
1	1	3	Ayush	Ayush	1651651	B	34	M
2	2	3	Manas	Manas	6561141	B	35	M
3	3	4	asda	asd	34235345	B	35	
4	4	5	Ayush	Ayush@	23435	B	34	M
5	5	5	Manas	Manas@	4567	B	35	M
6	6	6	Manas	Manas	444797	B	35	M
7	7	6	Jayraj	Jayraj	454587	B	26	M

At the bottom of the window, there are two tabs: "Output" and "SQL 3". The "Output" tab is currently active, showing an empty text area. The "SQL 3" tab is also visible. The bottom right corner of the window features standard Windows navigation buttons.

## **5. IMPLEMENTATION**

### **GroupMain.java:**

```
package p1;

import javax.swing.*;

public class GroupMain {
    public static void main(String[] args) {
        // Create and display the LoginApplet
        LoginApplet loginApplet = new LoginApplet();
        loginApplet.setVisible(true); // Show the login page
    }
}
```

### **LoginPage.java:**

```
package p1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;

public class LoginApplet extends JFrame implements ActionListener {
    JTextField emailField;
    JPasswordField passwordField;
    JButton loginButton, registerButton;
    JLabel emailLabel, passwordLabel, loginLabel, createAccountLabel;
    public static HashMap<String, String[]> logininfo = new HashMap<>();

    public LoginApplet() {
        // Set up the JFrame
        setTitle("Login Page");
        setLayout(null); // Absolute positioning
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setBackground(new Color(255, 204, 102));

        // Title - "Login"
        loginLabel = new JLabel("Login", JLabel.CENTER);
        loginLabel.setFont(new Font("SansSerif", Font.BOLD, 24));
        loginLabel.setBounds(110, 20, 150, 40);
        add(loginLabel);
    }
}
```

```

// Email Label and TextField
emailLabel = new JLabel("Email id :");
emailLabel.setFont(new Font("SansSerif", Font.PLAIN, 14));
emailLabel.setBounds(90, 80, 80, 25);
add(emailLabel);

emailField = new JTextField(20);
emailField.setBounds(180, 80, 150, 25);
add(emailField);

// Password Label and PasswordField
passwordLabel = new JLabel("Password :");
passwordLabel.setFont(new Font("SansSerif", Font.PLAIN, 14));
passwordLabel.setBounds(90, 120, 80, 25);
add(passwordLabel);

passwordField = new JPasswordField(20);
passwordField.setBounds(180, 120, 150, 25);
add(passwordField);

// Login Button
loginButton = new JButton("Login");
loginButton.setBounds(140, 160, 100, 30);
loginButton.setFont(new Font("SansSerif", Font.PLAIN, 14));
loginButton.addActionListener(this);
add(loginButton);

// "Don't have an account? Create one" label
createAccountLabel = new JLabel("Don't have an account? Create one", JLabel.CENTER);
createAccountLabel.setFont(new Font("SansSerif", Font.PLAIN, 12));
createAccountLabel.setBounds(90, 200, 200, 20);
add(createAccountLabel);

// Register Button
registerButton = new JButton("Register");
registerButton.setBounds(140, 230, 100, 30);
registerButton.setFont(new Font("SansSerif", Font.PLAIN, 14));
registerButton.addActionListener(this);
add(registerButton);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == loginButton) {
        String email = emailField.getText();
        String password = String.valueOf(passwordField.getPassword());

        if (validateLogin(email, password)) {
            JOptionPane.showMessageDialog(this, "Login Successful!");
            this.setVisible(false);
            new HomeSwing(email).setVisible(true); // Open Home Page after login
        } else {
            JOptionPane.showMessageDialog(this, "Invalid credentials! Please try again.", "Error",
JOptionPane.ERROR_MESSAGE);

```

```

    }
} else if (e.getSource() == registerButton) {
    this.setVisible(false);
    new RegistrationPage().setVisible(true); // Redirect to Registration Page
}
}

private boolean validateLogin(String email, String password) {
    String DB_URL = "jdbc:mysql://localhost:3306/test?"; // Replace with your DB name
    String USER = "root"; // Replace with your MySQL username
    String PASS = ""; // Replace with your MySQL password

    boolean isValid = false;

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS)) {
        String query = "SELECT Password FROM logininfo WHERE Email = ?";
        PreparedStatement preparedStatement = conn.prepareStatement(query);
        preparedStatement.setString(1, email);
        ResultSet resultSet = preparedStatement.executeQuery();

        if (resultSet.next()) {
            String storedPassword = resultSet.getString("Password");
            isValid = storedPassword.equals(password);
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }

    return isValid;
}

public static void main(String[] args) {
    new LoginApplet().setVisible(true);
}
}

```

## RegistrationPage.java

```

i package p1;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.HashMap;

public class RegistrationPage extends JFrame implements ActionListener {
    JTextField firstNameField, lastNameField, emailField;
    JPasswordField newPasswordField, confirmPasswordField;
    JButton registerButton, backToLoginButton;
    JLabel titleLabel, firstNameLabel, lastNameLabel, emailLabel, passwordLabel, confirmPasswordLabel;
}

```

```

public static HashMap<String, String[]> logininfo = new HashMap<>();

// Database connection details
// Class.forName("com.mysql.cj.jdbc.Driver");
// Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?" + "user=root");
static final String DB_URL = "jdbc:mysql://localhost:3306/test?"; // Replace with your database name
static final String USER = "root"; // Replace with your MySQL username
static final String PASS = ""; // Replace with your MySQL password

public RegistrationPage() {

    // Set up the JFrame
    setTitle("Registration Page");
    setLayout(null);
    setSize(400, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    getContentPane().setBackground(new Color(255, 204, 102));

    // Title - "Registration"
    titleLabel = new JLabel("Registration", JLabel.CENTER);
    titleLabel.setFont(new Font("SansSerif", Font.BOLD, 24));
    titleLabel.setBounds(100, 20, 180, 40);
    add(titleLabel);

    // First Name and Last Name Labels and TextFields
    firstNameLabel = new JLabel("First Name:");
    firstNameLabel.setFont(new Font("SansSerif", Font.PLAIN, 14));
    firstNameLabel.setBounds(60, 80, 100, 25);
    add(firstNameLabel);

    firstNameField = new JTextField(20);
    firstNameField.setBounds(185, 80, 150, 25);
    add(firstNameField);

    lastNameLabel = new JLabel("Last Name:");
    lastNameLabel.setFont(new Font("SansSerif", Font.PLAIN, 14));
    lastNameLabel.setBounds(60, 110, 100, 25);
    add(lastNameLabel);

    lastNameField = new JTextField(20);
    lastNameField.setBounds(185, 110, 150, 25);
    add(lastNameField);

    // Email Label and TextField
    emailLabel = new JLabel("Email id:");
    emailLabel.setFont(new Font("SansSerif", Font.PLAIN, 14));
    emailLabel.setBounds(60, 150, 100, 25);
    add(emailLabel);

    emailField = new JTextField(20);
    emailField.setBounds(185, 150, 150, 25);
    add(emailField);

    // New Password and Confirm Password Labels and PasswordFields
    passwordLabel = new JLabel("New Password:");
    passwordLabel.setFont(new Font("SansSerif", Font.PLAIN, 14));

```

```

passwordLabel.setBounds(60, 190, 100, 25);
add(passwordLabel);

newPasswordField = new JPasswordField(20);
newPasswordField.setBounds(185, 190, 150, 25);
add(newPasswordField);

confirmPasswordLabel = new JLabel("Confirm Password:");
confirmPasswordLabel.setFont(new Font("SansSerif", Font.PLAIN, 14));
confirmPasswordLabel.setBounds(60, 220, 150, 25);
add(confirmPasswordLabel);

confirmPasswordField = new JPasswordField(20);
confirmPasswordField.setBounds(185, 220, 150, 25);
add(confirmPasswordField);

// Register Button
registerButton = new JButton("Register");
registerButton.setBounds(140, 260, 100, 30);
registerButton.setFont(new Font("SansSerif", Font.PLAIN, 14));
registerButton.addActionListener(this);
add(registerButton);

// Back to Login Button
backToLoginButton = new JButton("Back to Login");
backToLoginButton.setBounds(115, 300, 150, 30);
backToLoginButton.setFont(new Font("SansSerif", Font.PLAIN, 14));
backToLoginButton.addActionListener(this);
add(backToLoginButton);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == registerButton) {
        // Validate password confirmation
        String newPassword = String.valueOf(newPasswordField.getPassword());
        String confirmPassword = String.valueOf(confirmPasswordField.getPassword());

        if (!newPassword.equals(confirmPassword)) {
            JOptionPane.showMessageDialog(this, "Passwords do not match! Please try again.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Save to the MySQL database
        String firstName = firstNameField.getText();
        String lastName = lastNameField.getText();
        String email = emailField.getText();

        if (firstName.isEmpty() || lastName.isEmpty() || email.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please fill all fields!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS)) {
            String insertSQL = "INSERT INTO logininfo (Fname, Lname, Email, Password) VALUES (?, ?, ?, ?)";

```

```

        PreparedStatement preparedStatement = conn.prepareStatement(insertSQL);
        preparedStatement.setString(1, firstName);
        preparedStatement.setString(2, lastName);
        preparedStatement.setString(3, email);
        preparedStatement.setString(4, newPassword);
        preparedStatement.executeUpdate();

        JOptionPane.showMessageDialog(this, "Registration Successful!");
        this.setVisible(false); // Hide registration form
        new LoginApplet().setVisible(true); // Show login form
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error during registration: " + ex.getMessage(), "Database
Error", JOptionPane.ERROR_MESSAGE);
    }
    } else if (e.getSource() == backToLoginButton) {
        this.setVisible(false);
        new LoginApplet().setVisible(true);
    }
}

public static void main(String[] args) {
    new RegistrationPage().setVisible(true);
}
}

```

## HomeSwing.java:

```

package p1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class HomeSwing extends JFrame implements ActionListener {
    JLabel welcomeLabel, emailLabel;
    JButton createGroupButton, viewGroupButton, logoutButton;
    String userEmail;
    static final String DB_URL = "jdbc:mysql://localhost:3306/test?"; // Replace with your database name
    static final String USER = "root"; // Replace with your MySQL username
    static final String PASS = ""; // Replace with your MySQL password

    public HomeSwing(String email) {
        this.userEmail = email;

        // Set up the JFrame
        setTitle("Home Page");
        setLayout(null);
    }
}

```

```

setSize(400, 300);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
getContentPane().setBackground(new Color(255, 204, 102));

// Welcome Label
if (RegistrationPage.logininfo != null && RegistrationPage.logininfo.containsKey(email)) {
    String firstName = RegistrationPage.logininfo.get(email)[0];
    welcomeLabel = new JLabel("Welcome, " + firstName + "!", JLabel.CENTER);
} else {
    welcomeLabel = new JLabel("Welcome!", JLabel.CENTER);
}

welcomeLabel.setFont(new Font("SansSerif", Font.BOLD, 24));
welcomeLabel.setBounds(90, 30, 200, 40);
add(welcomeLabel);

// Email Label
emailLabel = new JLabel("Email: " + email, JLabel.CENTER);
emailLabel.setFont(new Font("SansSerif", Font.PLAIN, 14));
emailLabel.setBounds(100, 80, 200, 20);
add(emailLabel);

// Create Group Button
createGroupButton = new JButton("Create Group");
createGroupButton.setBounds(120, 120, 150, 30);
createGroupButton.setFont(new Font("SansSerif", Font.PLAIN, 14));
createGroupButton.addActionListener(this);
add(createGroupButton);

// View Group Button
viewGroupButton = new JButton("View Groups");
viewGroupButton.setBounds(120, 160, 150, 30);
viewGroupButton.setFont(new Font("SansSerif", Font.PLAIN, 14));
viewGroupButton.addActionListener(this);
add(viewGroupButton);

// Logout Button
logoutButton = new JButton("Logout");
logoutButton.setBounds(120, 200, 150, 30);
logoutButton.setFont(new Font("SansSerif", Font.PLAIN, 14));
logoutButton.addActionListener(this);
add(logoutButton);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == createGroupButton) {
        this.setVisible(false);
        new ProjectForm(userEmail).setVisible(true); // Open project form to create a group
    } else if (e.getSource() == viewGroupButton) {
        this.setVisible(false);
        HashMap<String, ArrayList<String[]>> groupData = getGroupData();
        if (groupData.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No groups available.", "Information",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            new ViewGroupSwing(userEmail, groupData).setVisible(true); // Open view group form
        }
    }
}

```



```

    } else if (e.getSource() == logoutButton) {
        this.setVisible(false);
        new LoginApplet().setVisible(true); // Go back to login page
    }
}

private HashMap<String, ArrayList<String[]>> getGroupData() {
    HashMap<String, ArrayList<String[]>> groupData = new HashMap<>();

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS)) {
        // Retrieve all group information
        String query = "SELECT g.groupid, g.projecttopic, s.sname, s.smail FROM groupinfo g JOIN studentinfo s
ON g.groupid = s.groupid ORDER BY g.groupid, s.sno";
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery();

        int lastGroupId = -1;
        ArrayList<String[]> members = null;

        while (rs.next()) {
            int groupId = rs.getInt("groupid");
            String projectTopic = rs.getString("projecttopic");
            String studentName = rs.getString("sname");
            String studentEmail = rs.getString("smail");

            // If it's a new group, create a new list for its members
            if (groupId != lastGroupId) {
                if (lastGroupId != -1) {
                    groupData.put("Group " + lastGroupId + " (" + projectTopic + ")", members);
                }
                members = new ArrayList<>();
                lastGroupId = groupId;
            }
            members.add(new String[]{studentName, studentEmail});
        }

        // Add the last group
        if (members != null) {
            groupData.put("Group " + lastGroupId, members);
        }

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Error retrieving group data: " + ex.getMessage(), "Database
Error", JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    }

    return groupData;
}

public static void main(String[] args) {
    new HomeSwing("example@example.com").setVisible(true); // Testing purposes
}
}

```

## ProjectForm.java:

```

package p1;

import javax.swing.*.*;
import java.awt.*.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.*;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;

public class ProjectForm extends JFrame implements ActionListener {

    private String userEmail; // Field to store the email
    // Define form components
    JTextField dateField;
    JTextField subjectField, classField, projectTopicField;
    JComboBox<String> numberOfMembersCombo;
    JTable memberTable;
    JButton submitButton, backButton, clearButton; // Removed viewDataButton

    // Define a table model to hold the data
    DefaultTableModel tableModel;

    // Database connection details
    static final String DB_URL = "jdbc:mysql://localhost:3306/test"; // Change DB URL as per your
    config
    static final String USER = "root"; // Change it if your MySQL username is different
    static final String PASS = ""; // Add your MySQL password here

    public ProjectForm(String userEmail) {
        this.userEmail = userEmail;
        // Set up JFrame
        setTitle("Project Form");
        setSize(600, 450);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(null);

        // Number of Members Dropdown
        JLabel membersLabel = new JLabel("No. of Members:");
        membersLabel.setBounds(20, 20, 120, 25);
        add(membersLabel);

        String[] membersOptions = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
        numberOfMembersCombo = new JComboBox<>(membersOptions);
        numberOfMembersCombo.setBounds(150, 20, 50, 25);
        numberOfMembersCombo.addActionListener(this); // Add ActionListener
        add(numberOfMembersCombo);

        // Date Field

```

```

JLabel dateLabel = new JLabel("Date: (dd-mm-yyyy)");
dateLabel.setBounds(250, 20, 150, 25);
add(dateLabel);

dateField = new JTextField();
dateField.setBounds(400, 20, 150, 25);
add(dateField);

// Subject Field
JLabel subjectLabel = new JLabel("Subject:");
subjectLabel.setBounds(20, 60, 100, 25);
add(subjectLabel);

subjectField = new JTextField();
subjectField.setBounds(130, 60, 150, 25);
add(subjectField);

// Class Field
JLabel classLabel = new JLabel("Class:");
classLabel.setBounds(20, 100, 100, 25);
add(classLabel);

classField = new JTextField();
classField.setBounds(130, 100, 150, 25);
add(classField);

// Project Topic Field
JLabel projectTopicLabel = new JLabel("Project Topic:");
projectTopicLabel.setBounds(20, 140, 100, 25);
add(projectTopicLabel);

projectTopicField = new JTextField();
projectTopicField.setBounds(130, 140, 300, 25);
add(projectTopicField);

// Red Note: Leader Details First!!!
JLabel noteLabel = new JLabel("Leader Details First!!!");
noteLabel.setBounds(130, 200, 200, 25);
noteLabel.setForeground(Color.RED); // Set text color to red
add(noteLabel);

// Member Table
String[] columnNames = {"Sr.", "Name", "Email", "14-Digit No.", "Batch", "Roll no",
"Gender(M/F)"};
tableModel = new DefaultTableModel(columnNames, 0); // Initialize with 0 rows
memberTable = new JTable(tableModel);
JScrollPane scrollPane = new JScrollPane(memberTable);
scrollPane.setBounds(20, 230, 550, 100);
add(scrollPane);

// Submit Button

```

```

submitButton = new JButton("Submit");
submitButton.setBounds(350, 340, 100, 30);
submitButton.addActionListener(this);
add(submitButton);

// Back Button (to return to Login/Registration Page)
backButton = new JButton("Back");
backButton.setBounds(460, 340, 100, 30);
backButton.addActionListener(this);
add(backButton);

// Clear Button
clearButton = new JButton("Clear");
clearButton.setBounds(240, 340, 100, 30);
clearButton.addActionListener(this);
add(clearButton);

// Update the table based on the default selected number of members
updateMemberTable();
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == numberOfMembersCombo) {
        // Update the member table when the number of members changes
        updateMemberTable();
    } else if (e.getSource() == submitButton) {
        // Perform form submission action
        submitForm();
    } else if (e.getSource() == backButton) {
        // Navigate back to the login page
        this.setVisible(false);
        new HomeSwing(userEmail).setVisible(true); // Return to login page
    } else if (e.getSource() == clearButton) {
        // Clear all fields and table data
        clearForm();
    }
}

private void updateMemberTable() {
    // Get the selected number of members
    int selectedIndex = numberOfMembersCombo.getSelectedIndex();
    int numberOfMembers = selectedIndex + 1; // ComboBox index starts from 0

    // Clear the existing rows
    tableModel.setRowCount(0);

    // Add rows based on the selected number of members
    for (int i = 0; i < numberOfMembers; i++) {
        tableModel.addRow(new Object[]{i + 1, "", "", "", "", "", ""}); // Empty fields for user input
    }
}

```

```

}

private void submitForm() {
// Get form data
String subject = subjectField.getText();
String className = classField.getText();
String projectTopic = projectTopicField.getText();
String dateInput = dateField.getText();

// Validate input
if (subject.isEmpty() || className.isEmpty() || projectTopic.isEmpty() || dateInput.isEmpty()) {
    JOptionPane.showMessageDialog(this, "Please fill all required fields.", "Error",
JOptionPane.ERROR_MESSAGE);
    return;
}

// Validate date format
SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
dateFormat.setLenient(false); // Prevents lenient parsing

Date date1;
try {
    date1 = dateFormat.parse(dateInput); // Parse the date
} catch (ParseException e) {
    JOptionPane.showMessageDialog(this, "Please enter the date in the format DD-MM-YYYY.",
    "Date Format Error", JOptionPane.ERROR_MESSAGE);
    return; // Exit if the date format is incorrect
}

// Insert data into the database
try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS)) {
    // Insert group details into groupinfo table
    String insertGroupSQL = "INSERT INTO groupinfo (noofmem, date, class, subject, projecttopic)
VALUES (?, ?, ?, ?, ?)";
    PreparedStatement groupStatement = conn.prepareStatement(insertGroupSQL,
Statement.RETURN_GENERATED_KEYS);

    groupStatement.setInt(1, numberOfMembersCombo.getSelectedIndex() + 1); // Number of
members
    groupStatement.setDate(2, new java.sql.Date(date1.getTime())); // Date
    groupStatement.setString(3, className); // Class
    groupStatement.setString(4, subject); // Subject
    groupStatement.setString(5, projectTopic); // Project topic

    // Execute group insertion and retrieve the generated groupid
    int affectedRows = groupStatement.executeUpdate();
    if (affectedRows == 0) {
        throw new SQLException("Inserting group failed, no rows affected.");
    }

    try (ResultSet generatedKeys = groupStatement.getGeneratedKeys()) {

```

```

        if (generatedKeys.next()) {
            int groupId = generatedKeys.getInt(1); // Get the generated groupId

            // Insert each student's details into studentinfo table
            String insertStudentSQL = "INSERT INTO studentinfo (groupId, sname, smail, prn, batch,
rollno, gender) VALUES (?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement studentStatement = conn.prepareStatement(insertStudentSQL);

            for (int i = 0; i < tableModel.getRowCount(); i++) {
                studentStatement.setInt(1, groupId); // Use the retrieved groupId
                studentStatement.setString(2, (String) tableModel.getValueAt(i, 1)); // Name
                studentStatement.setString(3, (String) tableModel.getValueAt(i, 2)); // Email
                studentStatement.setString(4, (String) tableModel.getValueAt(i, 3)); // PRN (14-digit
number)
                studentStatement.setString(5, (String) tableModel.getValueAt(i, 4)); // Batch
                studentStatement.setInt(6, Integer.parseInt((String) tableModel.getValueAt(i, 5))); // Roll
No
                studentStatement.setString(7, (String) tableModel.getValueAt(i, 6)); // Gender

                // Execute the student insertion query
                studentStatement.executeUpdate();
            }

            JOptionPane.showMessageDialog(this, "Project details and student information submitted
successfully!");
        } else {
            throw new SQLException("Inserting group failed, no ID obtained.");
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error while inserting data into the database.\n" +
ex.getMessage(), "Database Error", JOptionPane.ERROR_MESSAGE);
    }
}

// Method to clear all the fields and table
private void clearForm() {
    // Clear the text fields
    subjectField.setText("");
    classField.setText("");
    projectTopicField.setText("");
    dateField.setText(""); // Clear the email field

    // Clear the table data
    tableModel.setRowCount(0);

    // Reset the member table based on the number of members selected
    updateMemberTable();
}

```

```

    public static void main(String[] args) {
        new ProjectForm("example@example.com").setVisible(true);
    }
}

```

## ViewGroupSwing.java:

```

package p1;

import javax.swing.*.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*.*;
import java.util.ArrayList;
import java.util.HashMap;

public class ViewGroupSwing extends JFrame {

    private JTable groupTable;
    private DefaultTableModel tableModel;
    private String userEmail;
    private static final String DB_URL = "jdbc:mysql://localhost:3306/test"; // Update with your DB
    name
    private static final String USER = "root"; // Your MySQL username
    private static final String PASS = ""; // Your MySQL password
    private HashMap<String, ArrayList<String[]>> groupInfoDB; // Ensure this is correctly declared

    public ViewGroupSwing(String userEmail, HashMap<String, ArrayList<String[]>> groupInfoDB) {
        this.userEmail = userEmail;

        setTitle("View Groups");
        setSize(800, 400);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null); // Center the window

        // Create table model with column names
        String[] columnNames = {"Group ID", "Project Topic", "Members", "Actions"};
        tableModel = new DefaultTableModel(columnNames, 0);
        groupTable = new JTable(tableModel);
        groupTable.setRowHeight(30);
        groupTable.setFont(new Font("SansSerif", Font.PLAIN, 14));

        // Populate the table with group data
        populateTable();

        // Add table to scroll pane
        JScrollPane scrollPane = new JScrollPane(groupTable);

```

```

add(scrollPane, BorderLayout.CENTER);

// Add a back button
JButton backButton = new JButton("Back");
backButton.addActionListener(e -> {
    this.setVisible(false); // Hide current window
    new HomeSwing(userEmail).setVisible(true); // Open HomeSwing
});
add(backButton, BorderLayout.SOUTH);
}

private void populateTable() {
    HashMap<Integer, String> groupMap = new HashMap<>(); // Map to hold group IDs and topics
    String query = "SELECT g.groupid, g.projecttopic FROM groupinfo g JOIN studentinfo s ON
g.groupid = s.groupid WHERE s.smail = ?";

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
        PreparedStatement stmt = conn.prepareStatement(query)) {

        stmt.setString(1, userEmail);
        ResultSet rs = stmt.executeQuery();

        while (rs.next()) {
            int groupId = rs.getInt("groupid");
            String projectTopic = rs.getString("projecttopic");
            groupMap.put(groupId, projectTopic);

            // Now retrieve members for each group
            addGroupMembersToTable(groupId, projectTopic);
        }

    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private void addGroupMembersToTable(int groupId, String projectTopic) {
    String query = "SELECT s.sname, s.smail FROM studentinfo s WHERE s.groupid = ?";

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
        PreparedStatement stmt = conn.prepareStatement(query)) {

        stmt.setInt(1, groupId);
        ResultSet rs = stmt.executeQuery();
        StringBuilder memberList = new StringBuilder();

        while (rs.next()) {
            String studentName = rs.getString("sname");
            String studentEmail = rs.getString("smail");
            memberList.append(studentName).append(" (").append(studentEmail).append("), ");
        }
    }
}

```



```

        if (memberList.length() > 0) {
            memberList.setLength(memberList.length() - 2); // Remove the last comma and space
        }

        // Add row to the table with group ID, project topic, and members
        Object[] rowData = {groupId, projectTopic, memberList.toString(),
createActionPanel(groupId)};
        tableModel.addRow(rowData);
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private JPanel createActionPanel(int groupId) {
    JPanel panel = new JPanel();
    panel.setLayout(new FlowLayout(FlowLayout.LEFT));

    JButton editButton = new JButton("Edit");
    editButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Open edit dialog or new frame for editing group details
            new EditGroupSwing(groupId, userEmail).setVisible(true);
        }
    });

    JButton deleteButton = new JButton("Delete");
    deleteButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            deleteGroup(groupId);
        }
    });

    panel.add(editButton);
    panel.add(deleteButton);
    return panel;
}

private void deleteGroup(int groupId) {
    int confirmation = JOptionPane.showConfirmDialog(this,
        "Are you sure you want to delete this group?",
        "Confirm Deletion",
        JOptionPane.YES_NO_OPTION);

    if (confirmation == JOptionPane.YES_OPTION) {
        String deleteQuery = "DELETE FROM groupinfo WHERE groupid = ?";
        try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
            PreparedStatement stmt = conn.prepareStatement(deleteQuery)) {

```

```

        stmt.setInt(1, groupId);
        stmt.executeUpdate();
        // Refresh the table
        tableModel.setRowCount(0); // Clear the current table data
        populateTable(); // Repopulate the table
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
}

public static void main(String[] args) {
    // For testing, create a sample HashMap for group information
    HashMap<String, ArrayList<String[]>> sampleGroupData = new HashMap<>();
    ArrayList<String[]> members = new ArrayList<>();
    members.add(new String[]{"Alice", "alice@example.com"});
    members.add(new String[]{"Bob", "bob@example.com"});
    sampleGroupData.put("Group 1 (Project A)", members);

    SwingUtilities.invokeLater(() -> new ViewGroupSwing("testuser@example.com",
sampleGroupData).setVisible(true));
}
}

```

## EditGroupSwing.java:

```

package p1;

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class EditGroupSwing extends JFrame {

    private JTextField projectTopicField;
    private JTextField memberEmailsField; // To add or remove members
    private int groupId;
    private String userEmail;
    private static final String DB_URL = "jdbc:mysql://localhost:3306/test"; // Update with your DB
name
    private static final String USER = "root"; // Your MySQL username
    private static final String PASS = ""; // Your MySQL password

    public EditGroupSwing(int groupId, String userEmail) {
        this.groupId = groupId;
        this.userEmail = userEmail;
    }
}

```

```

setTitle("Edit Group");
setSize(400, 300);
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
setLocationRelativeTo(null); // Center the window

// Create UI components
JLabel projectTopicLabel = new JLabel("Project Topic:");
projectTopicField = new JTextField(20);

JLabel memberEmailsLabel = new JLabel("Member Emails (comma separated):");
memberEmailsField = new JTextField(20);

JButton saveButton = new JButton("Save Changes");
saveButton.addActionListener(new SaveButtonListener());

// Layout the components
setLayout(new GridLayout(5, 2));
add(projectTopicLabel);
add(projectTopicField);
add(memberEmailsLabel);
add(memberEmailsField);
add(saveButton);
}

private class SaveButtonListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        String projectTopic = projectTopicField.getText();
        String memberEmails = memberEmailsField.getText();

        // Update the project topic in the database
        updateGroupDetails(projectTopic, memberEmails);
    }
}

private void updateGroupDetails(String projectTopic, String memberEmails) {
    String updateQuery = "UPDATE groupinfo SET projecttopic = ? WHERE groupid = ?";

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
        PreparedStatement stmt = conn.prepareStatement(updateQuery)) {

        // Update the group project topic
        stmt.setString(1, projectTopic);
        stmt.setInt(2, groupId);
        stmt.executeUpdate();

        // Optionally, handle member updates (this can be more complex, depending on your
        requirements)
        updateGroupMembers(memberEmails);

        JOptionPane.showMessageDialog(this, "Group details updated successfully.");
    }
}

```

```

        this.dispose(); // Close the edit window

    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error updating group details: " + ex.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

private void updateGroupMembers(String memberEmails) {
    // Here you can implement logic to update group members based on the input
    // You could split the emails and add/remove members as needed

    // For simplicity, let's assume you are just adding members
    String[] emails = memberEmails.split(",");
    String insertQuery = "INSERT INTO studentinfo (smail, groupid) VALUES (?, ?)";

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
        PreparedStatement stmt = conn.prepareStatement(insertQuery)) {

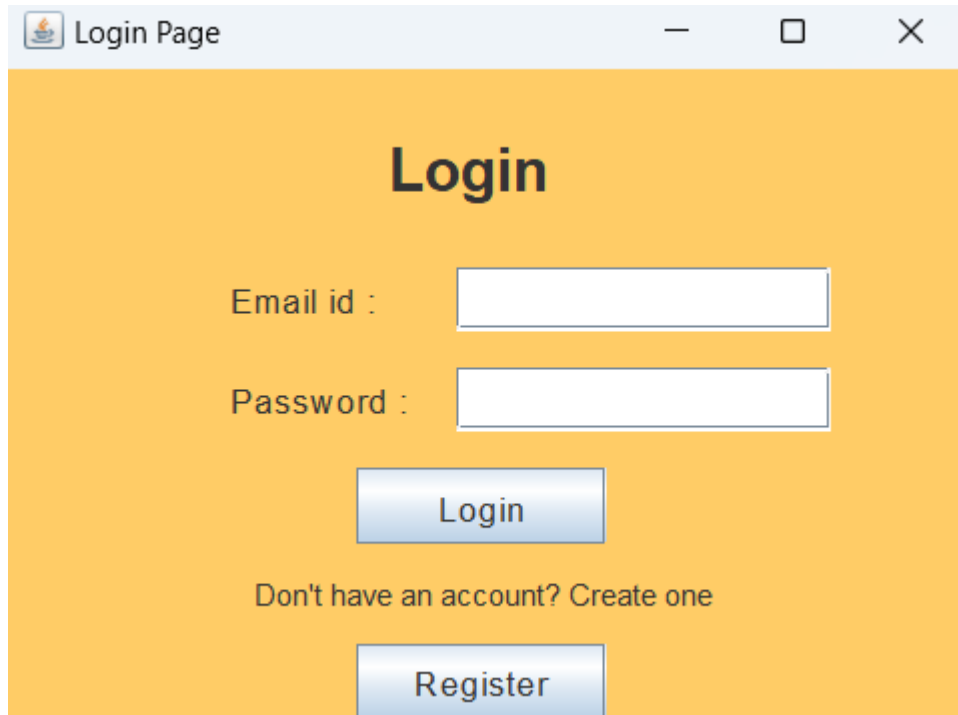
        for (String email : emails) {
            stmt.setString(1, email.trim());
            stmt.setInt(2, groupId);
            stmt.executeUpdate();
        }

    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error updating members: " + ex.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

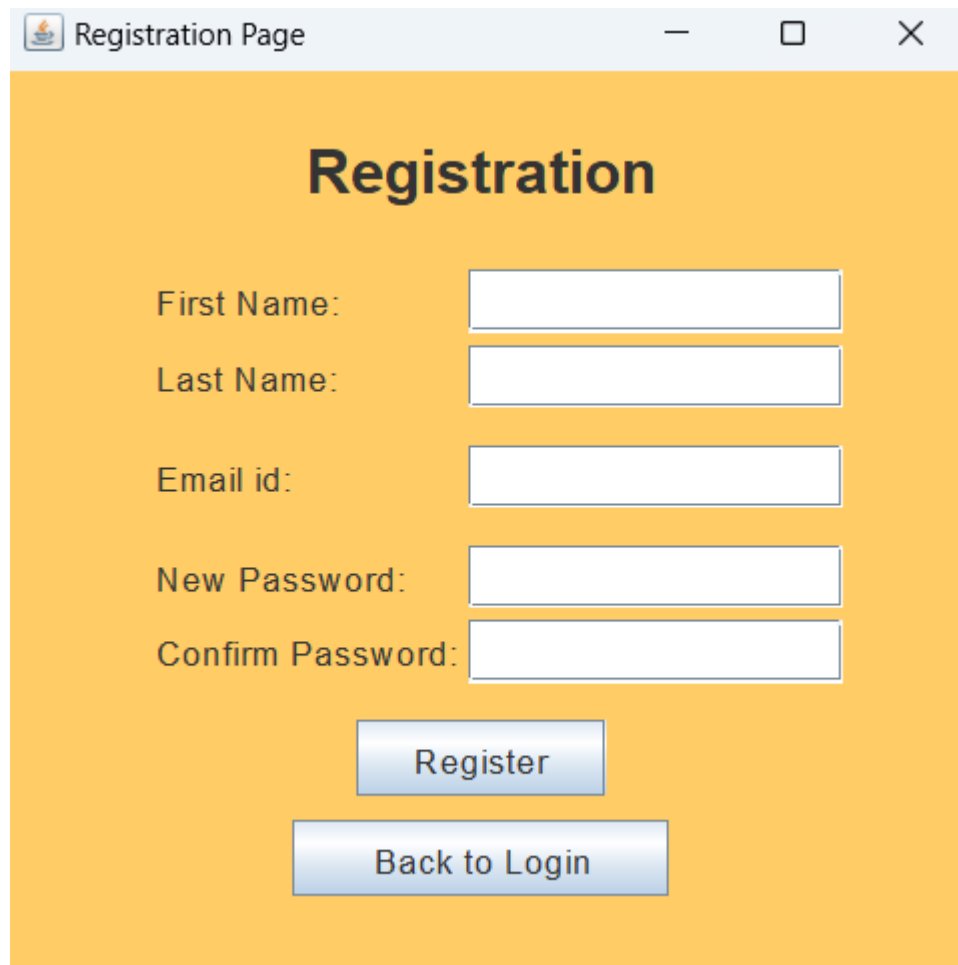
public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new EditGroupSwing(1,
        "example@example.com").setVisible(true)); // Testing purposes
}
}

```

## 6. RESULTS



The screenshot shows a web browser window titled "Login Page". The page has a solid orange background. At the top center, the word "Login" is written in a large, bold, black font. Below this, there are two input fields: "Email id :" followed by a white rectangular box, and "Password :" followed by another white rectangular box. Under the password field is a blue button with the text "Login" in white. Below the button is the text "Don't have an account? Create one" in a smaller black font. At the bottom center is another blue button with the text "Register" in white.



The screenshot shows a web browser window titled "Registration Page". The page has a solid orange background. At the top center, the word "Registration" is written in a large, bold, black font. Below this, there are five input fields arranged vertically: "First Name:" followed by a white rectangular box, "Last Name:" followed by a white rectangular box, "Email id:" followed by a white rectangular box, "New Password:" followed by a white rectangular box, and "Confirm Password:" followed by a white rectangular box. Below the "Confirm Password" field is a blue button with the text "Register" in white. At the bottom center is another blue button with the text "Back to Login" in white.

Project Form

No. of Members:

1

Date: (dd-mm-yyyy)

Subject:

Class:

Project Topic:

Leader Details First!!!

Sr.	Name	Email	14-Digit No.	Batch	Roll no	Gender(M/F)
1						

Clear

Submit

Back

Login Page

## Login

Email id :

Password :

Login

Don't have an account? Create one

Register

Registration Page

## Registration

First Name:

Last Name:

Email id:

New Password:

Confirm Password:

Register

Back to Login

Registration Page

# Registration

First Name:

Database Error

**X** Error during registration: Duplicate entry 'Manas' for key 'Email'

OK

Confirm Password:

Register

Back to Login

Home Page

# Welcome!

Email: Ayush

Create Group

View Groups

Logout



Project Form

No. of Members:  Date: (dd-mm-yyyy)

Subject:

Class:

Project Topic:

**Leader Details First!!!**

Sr.	Name	Email	14-Digit No.	Batch	Roll no	Gender(M/F)
1	Manas	Manas	444797	B	35	M
2	Jayraj	Jayraj	454587	B	26	M

Project Form


No. of Members:  Date: (dd-mm-yyyy)

Subject:

Class:

Project Topic:

**Message**

 **Project details and student information submitted successfully!**

Sr.	Name	Email	14-Digit No.	Batch	Roll no	Gender(M/F)
1	Manas	Manas	444797	B	35	M
2	Jayraj	Jayraj	454587	B	26	M

## **7. CONCLUSION**

The Group Members Storage project has successfully achieved its primary goal of creating a streamlined, efficient, and secure system for managing and organizing group member data. By achieving the objectives, the system not only enhances data organization but also reduces the manual effort required to manage large amounts of member data, improving overall productivity for organizations or individuals managing multiple groups.

### **Future Improvements or Potential Extensions**

Though the project is functional and fulfills its primary goals, there are several potential improvements and extensions that could enhance its functionality and adaptability in the future:

#### **1. Advanced Search and Filter Features:**

- Future versions of the system could include more sophisticated search and filtering options, allowing users to search based on multiple criteria (e.g., group role, joining date, location).
- Adding an advanced search function with fuzzy matching and dynamic filters could make the system even more user-friendly for large datasets.

#### **2. Mobile App Integration:**

- A mobile version or app could be developed for users who need access to group and member information on the go, enhancing accessibility and ease of use in various environments (e.g., events, team meetings).

#### **3. Notifications and Alerts:**

- An alert system could be integrated to notify administrators or group managers of key events (e.g., when new members are added, when members leave, or when membership details are updated).
- Members themselves could receive notifications about group updates or reminders for important activities.

#### **4. Role-Based Task Assignment and Management:**

- The system could be extended to include task assignment features where group leaders or administrators can assign tasks to specific members or roles within the group, and track progress or completion status.
- A task-tracking dashboard could provide both members and admins with a visual overview of ongoing projects.

## 5. Integration with Other Systems:

- The system could be integrated with external tools or platforms, such as project management systems, communication apps (like Microsoft Teams), or event planning tools, to enhance collaboration and coordination.
- APIs could be developed to allow other systems to interact with the group member storage system, enabling seamless data sharing across platforms.

## 6. Member Self-Service Portal:

- A member portal could be created where individual group members can log in to update their own information, reducing the administrative burden on group leaders or managers.
- This portal could include personalized dashboards that provide group-related notifications and updates.

## 7. Enhanced Security Measures:

- Future improvements could introduce more advanced security features such as two-factor authentication (2FA) for users, encrypted communication channels (e.g., for mobile apps), and enhanced data privacy controls to comply with stricter regulations.
- Audit trails and activity logs could be expanded to give administrators more visibility into changes made within the system, further strengthening security.

These enhancements would make the **Group Members Storage** system even more versatile, scalable, and adaptable to various industries and user needs, ultimately improving its overall functionality and value to users.

## **8. REFERENCE**

<https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html>

JTable: <https://docs.oracle.com/javase/tutorial/uiswing/components/table.html>

JButton: <https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html>

JFrame: <https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html>