**CMS ID :** 053-23-0028

**Name :** Anand Saroop

**Program :** BS Software Engineering – II Section – A

**Course :** Object Oriented Programming (Theorey)

**Assignment :** Final Semester Project

**Project Name :** Gym Management System

**Institute :** Sukkur IBA University

**Instructor :** Dr Faheem Akhtar Rajput

**Project :** Gym Management System

**Description :**  In this project, I have used the Gym Present in EDC Block of Sukkur IBA University. This gym management system firstly asks the username and password as per the if else logic.

**User Name :** root **Password :** root

If the correct password is entered, the login window will dispose and a new Frame (**Main Page** ) will be opened that will give the choice to open a table from the tables:

1. **Members :** Members Enrolled
2. **MemberShips :** MemberShips Available
3. **Trainers :** Trainer Information

Click Either button, The new Frame will be opened with respect to the table connected but it will not dispose the Main Page.

Each Frame consists of the Table and data related the database table. Buttons to add, edit, delete and search data are available.

In each Frame, Header is set with IBA and logo and respective heading.

Every Frame consists of the components and their respective labels in accordance with their repective Tables.

All the necessary things like GUI, Event-handling and Databases are included.

**Database Description:**

1. **Database used :** MySQL.
2. **Database Type :** Relational Database
3. **Database Name :** GymManagement
4. **Number of Tables :** 3
5. **Table Names :** Trainer, member , membershipplan
6. **Trainer fields** : TrainerID , TrainerName, TrainerAge, TrainerShift, TrainerSalary
7. **Member fields** : MemID, MemName, MemAge,MemberShipID,FeesStatus,TrainerID,Shift
8. **Membershipplan fields** : MemberShipID, MemberShipName, Cardio, Bodybuilding, PersonalTrainer, MemberShipFees
9. **Relations :** Each member is assigned a trainer with the help of trainer ID as the foreign key. This means that there is a number of members connected that workout under a trainer. Each Trainer is connected to a membership through membershipID as the foreign key. This means that Each membership is subscriber by a certain number of members. You can not delete record from membership and trainer table if they are connected with any member.

**MySQL Code :**

**Note :** I had studied Relational Database Management system in class 11th . So due to that I am Familiar to many of the RDBMS concepts. However, I had forgetten a little bit syntax as I do not have proper command over SQL. Due to that reason I have took some help from ChatGPT in assisting my database SQL code. Especially in making relational tables. However I still have created tables, their fields and Keys myself.

**Code :**

**For Database :**  CREATE DATABASE GymManagement

CREATE TABLE Trainer (

TrainerID INT PRIMARY KEY,

TrainerName VARCHAR(30),

TrainerAge INT,

TrainerShift ENUM('Morning', 'Afternoon', 'Evening', 'Night'),

TrainerSalary DOUBLE

);

CREATE TABLE MemberShipPlan (

MemberShipID INT PRIMARY KEY,

MemberShipName VARCHAR(50),

Cardio BOOLEAN,

Bodybuilding BOOLEAN,

PersonalTrainer BOOLEAN,

MemberShipFees DOUBLE

);

CREATE TABLE Member (

MemID INT PRIMARY KEY,

MemName VARCHAR(50),

MemAge INT,

MembershipID INT,

FeesStatus BOOLEAN,

TrainerID INT,

Shift ENUM('Morning', 'Afternoon', 'Evening', 'Night'),

FOREIGN KEY (TrainerID) REFERENCES Trainer(TrainerID),

FOREIGN KEY (MembershipID) REFERENCES MemberShipPlan(MemberShipID)

);

Trainer's Shift

DELIMITER $$

CREATE TRIGGER SetMemberShift

BEFORE INSERT ON Member

FOR EACH ROW

BEGIN

DECLARE trainer\_shift ENUM('Morning', 'Afternoon', 'Evening', 'Night');

SELECT TrainerShift INTO trainer\_shift FROM Trainer WHERE TrainerID = NEW.TrainerID;

SET NEW.Shift = trainer\_shift;

END $$

DELIMITER ;

**Classes and interfaces :**

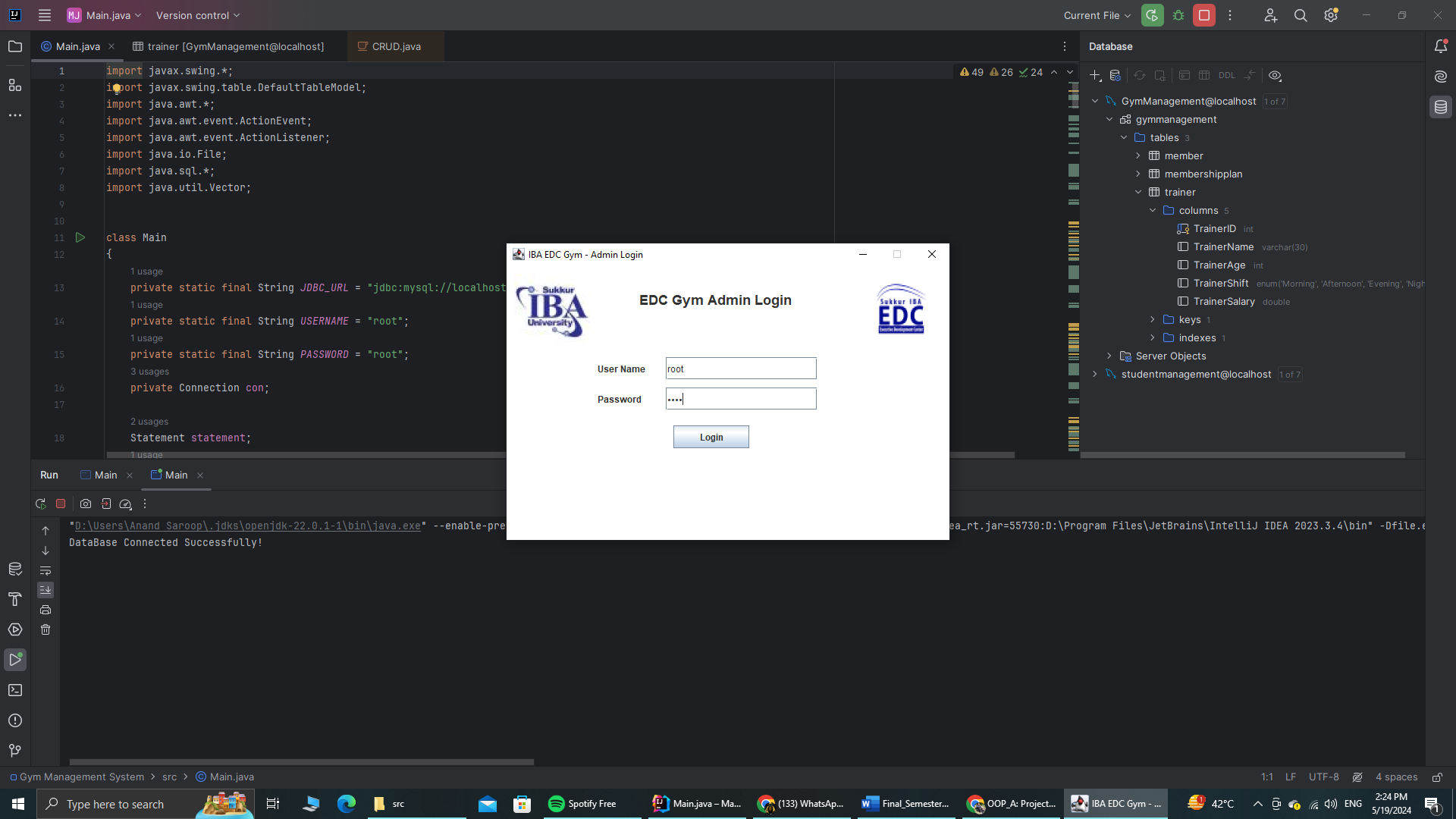
* Main class
* Login class
* Trainer class
* Member Class
* MemberShip Class
* JFrame as inheritance
* ActionListener as interface

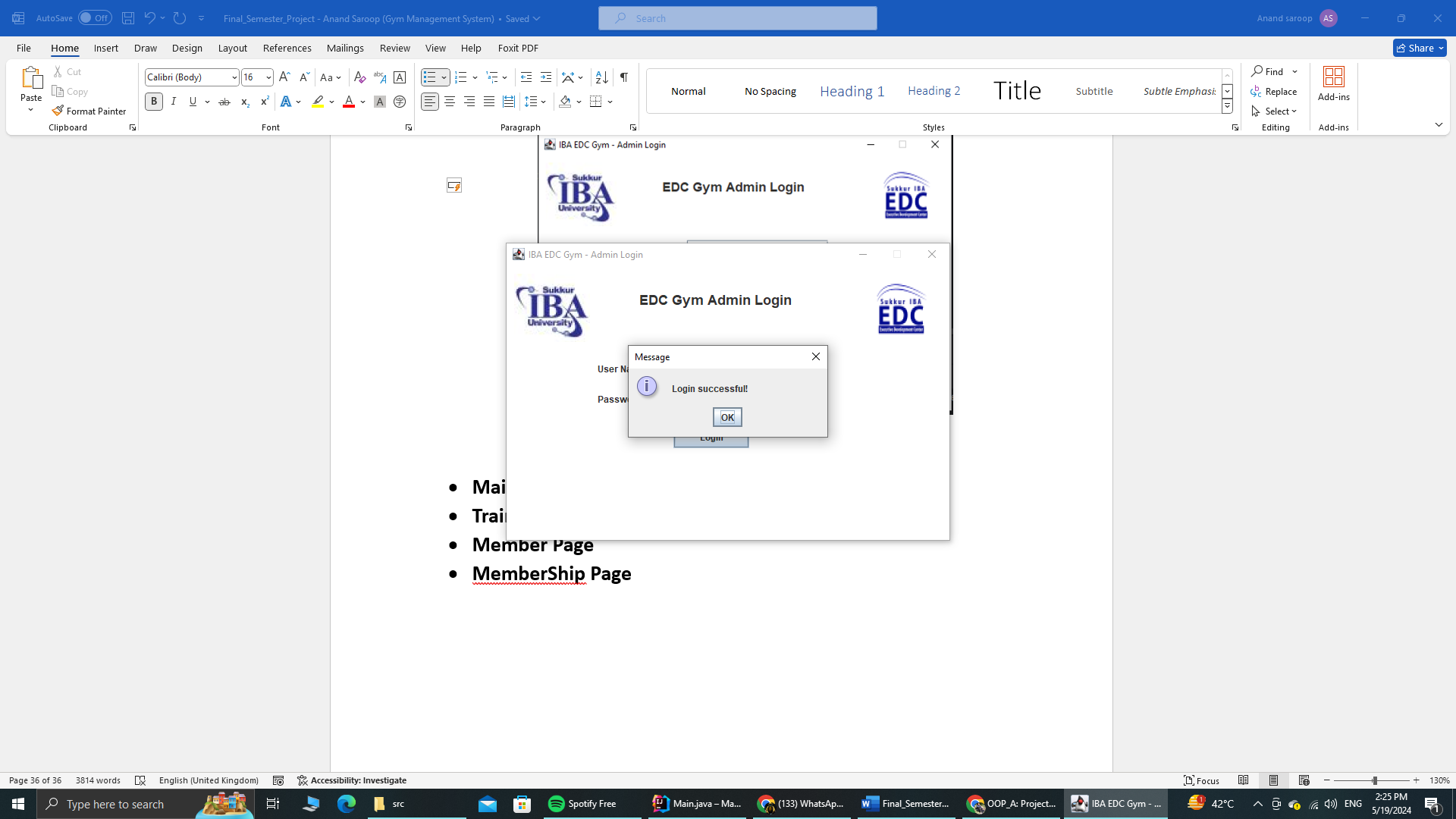
**Packages Used :**

* javax.swing.\*;
* javax.swing.table.DefaultTableModel;
* java.awt.\*;
* java.awt.event.ActionEvent;
* java.awt.event.ActionListener;
* java.io.File;
* java.sql.\*;
* java.util.Vector;

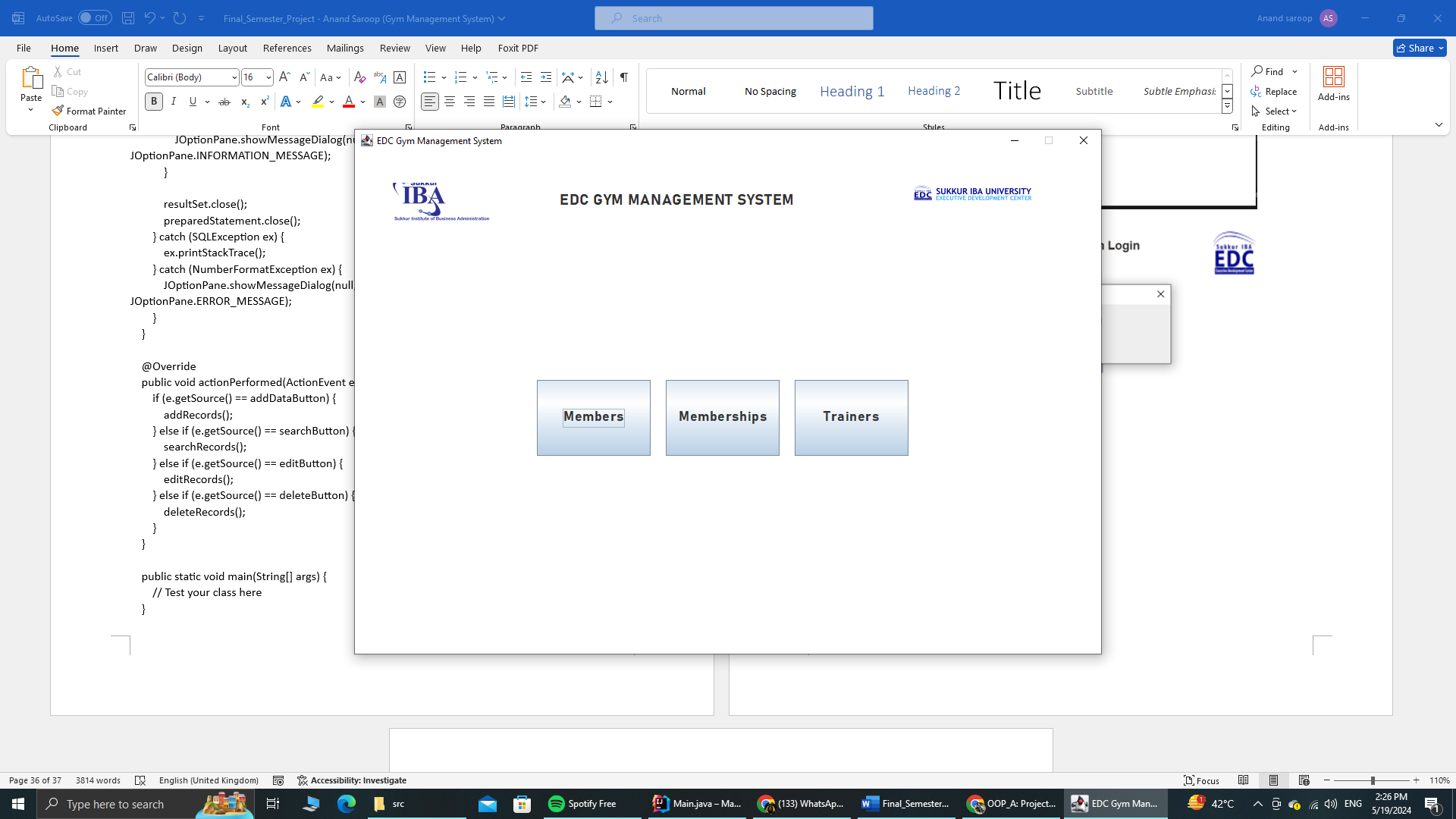
**Java Source Code :**   
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.File;  
import java.sql.\*;  
import java.util.Vector;  
  
  
class Main  
{  
 private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/GymManagement";  
 private static final String USERNAME = "root";  
 private static final String PASSWORD = "root";  
 private Connection con;  
  
 Statement statement;  
 Main()  
 {  
 try  
 {  
 con = DriverManager.getConnection(JDBC\_URL, USERNAME, PASSWORD);  
 statement = con.createStatement();  
 System.out.println("DataBase Connected Successfully!");  
 }  
 catch(SQLException e)  
 {  
 e.printStackTrace();  
 }  
  
  
  
 new Login(con,statement);  
  
 }  
 public static void main(String[] args)  
 {  
 new Main();  
  
  
 }  
  
}  
  
  
class Login extends JFrame implements ActionListener  
{  
  
 JLabel User = new JLabel("User Name");  
 JLabel Password = new JLabel("Password");  
 JTextField UserF = new JTextField();  
 JPasswordField PasswordF = new JPasswordField();  
 JPanel LoginPanel = new JPanel();  
 JPanel GUIPanel = new JPanel();  
  
 JButton Login = new JButton("Login");  
 Connection con;  
 Statement statement;  
 Login(Connection con,Statement statement)  
 {  
 this.con = con;  
 this.statement = statement;  
  
 setGUIcomponents();  
 setLoginFrame();  
 setLoginComponents();  
  
  
  
 }  
  
 public void setLoginFrame()  
 {  
  
  
  
  
 this.setTitle("IBA EDC Gym - Admin Login");  
 this.setSize(600,400);  
 this.setVisible(true);  
 this.setLayout(null);  
 this.setResizable(false);  
 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 this.setLocationRelativeTo(null);  
 this.getContentPane().setBackground(Color.WHITE);  
 GUIPanel.setBounds(0, 0, 600, 100);  
 LoginPanel.setBounds(0, 100, 500, 200);  
  
 this.add(GUIPanel);  
 this.add(LoginPanel);  
 }  
  
 public void setLoginComponents()  
 {  
 LoginPanel.setLayout(null);  
 LoginPanel.setBackground(Color.WHITE);  
  
 LoginPanel.add(User);  
 LoginPanel.add(UserF);  
  
  
 LoginPanel.add(Password);  
 LoginPanel.add(PasswordF);  
 LoginPanel.add(Login);  
 Login.addActionListener(this);  
  
  
 User.setBounds(120, 20, 100, 30);  
 UserF.setBounds(210, 20, 200, 30);  
 Password.setBounds(120, 60, 100, 30);  
 PasswordF.setBounds(210, 60, 200, 30);  
 Login.setBounds(220, 110, 100, 30);  
  
  
  
 }  
 ImageIcon loadImage(String path, int width, int height) {  
 File imageFile = new File(path);  
 if (imageFile.exists()) {  
 ImageIcon icon = new ImageIcon(path);  
 Image img = icon.getImage();  
 Image resizedImage = img.getScaledInstance(width, height, Image.SCALE\_SMOOTH);  
 return new ImageIcon(resizedImage);  
 } else {  
 System.out.println("Image file not found: " + path);  
 return new ImageIcon();  
 }  
 }  
  
 public void setGUIcomponents() {  
 ImageIcon ibaLogo = loadImage("src\\IBA.jpg", 100, 100);  
 JLabel imageIBA = new JLabel(ibaLogo);  
  
 ImageIcon edcLogo = loadImage("src\\EDC.png", 100, 100);  
 JLabel imageEDC = new JLabel(edcLogo);  
  
 GUIPanel.setBackground(Color.WHITE);  
 GUIPanel.setLayout(null);  
 GUIPanel.setVisible(true);  
  
 // Add components to guiPanel  
 imageIBA.setBounds(10, 10, 100, 100);  
 imageEDC.setBounds(470, 10, 100, 100);  
  
  
 // Add components to guiPanel  
  
  
 JLabel heading = new JLabel("EDC Gym Admin Login", JLabel.CENTER);  
 heading.setFont(new Font("Arial", Font.BOLD, 18));  
 heading.setBounds(150, 30, 250, 30);  
  
 GUIPanel.add(imageIBA);  
 GUIPanel.add(imageEDC);  
 GUIPanel.add(heading);  
 }  
  
 public void UserValidity()  
 {  
 String username = UserF.getText();  
 String password = new String(PasswordF.getPassword());  
 if (username.equals("root") && password.equals("root")) {  
  
 JOptionPane.showMessageDialog(this, "Login successful!");  
 dispose();  
 new MainPage(con,statement);  
  
 } else {  
 // Incorrect username or password  
 JOptionPane.showMessageDialog(this, "Incorrect username or password!", "Error", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
 public void actionPerformed(ActionEvent e)  
 {  
 if (e.getSource() == Login)  
 {  
 UserValidity();  
 }  
 }  
}  
  
  
  
  
class MainPage extends JFrame implements ActionListener  
{  
 JButton Trainer = new JButton("Trainers");  
 JButton Members = new JButton("Members");  
 JButton MemberShip = new JButton("Memberships");  
 JPanel ButtonPanel = new JPanel();  
 JPanel GUIPanel = new JPanel();  
 Connection connection;  
 Statement statement;  
 MainPage(Connection con, Statement statement )  
 {  
 connection = con;  
 this.statement = statement;  
 setGUI();  
 setButtons();  
 setMainFrame();  
  
 }  
 public void setMainFrame()  
 {  
  
  
  
 this.getContentPane().setBackground(Color.white);  
 this.setTitle("EDC Gym Management System");  
 this.setSize(1000,700);  
 this.setVisible(true);  
 this.setLayout(null);  
 this.setResizable(false);  
 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 this.setLocationRelativeTo(null);  
  
  
  
 this.add(ButtonPanel);  
 this.add(GUIPanel);  
  
 }  
  
 public void setButtons()  
 {  
 Trainer.setBounds(570, 0,150,100);  
 Trainer.setFont(new Font("Bahnschrift",Font.BOLD,18));  
 Members.setBounds(230, 0,150,100);  
 Members.setFont(new Font("Bahnschrift",Font.BOLD,18));  
 MemberShip.setBounds(400, 0,150,100);  
 MemberShip.setFont(new Font("Bahnschrift",Font.BOLD,18));  
  
 Trainer.addActionListener(this);  
 Members.addActionListener(this);  
 MemberShip.addActionListener(this);  
  
 ButtonPanel.setBackground(Color.white);  
 ButtonPanel.setBounds(10,300, 800, 400);  
 ButtonPanel.setLayout(null);  
 ButtonPanel.add(Members);  
 ButtonPanel.add(MemberShip);  
 ButtonPanel.add(Trainer);  
  
 }  
  
 public void setGUI()  
 {  
  
 ImageIcon EDClogo = new ImageIcon("src\\EDC2.png");  
 ImageIcon IBAlogo = new ImageIcon("src\\IBA1.png");  
  
  
 JLabel Heading = new JLabel("EDC GYM MANAGEMENT SYSTEM");  
 JLabel ImageEDC = new JLabel();  
 JLabel ImageIBA = new JLabel();  
 Heading.setFont(new Font("Bahnschrift",Font.BOLD,20));  
 ImageEDC.setIcon(EDClogo);  
 ImageIBA.setIcon(IBAlogo);  
 ImageEDC.setSize(300, 50);  
 ImageIBA.setSize(300, 50);  
 Heading.setSize(350, 50);  
 ImageEDC.setLocation(730, 30);  
 ImageIBA.setLocation(50, 40);  
 Heading.setLocation(270, 40);  
  
  
 GUIPanel.setLocation(10,10);  
 GUIPanel.setSize(800,100);  
 GUIPanel.setLayout(null);  
 GUIPanel.add(ImageEDC);  
 GUIPanel.add(ImageIBA);  
 GUIPanel.add(Heading);  
 GUIPanel.setBackground(Color.WHITE);  
 this.add(GUIPanel);  
  
  
  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if(e.getSource() == Trainer )  
 {  
 new Trainers(connection,statement);  
 }  
 if(e.getSource() == Members )  
 {  
 new Members(connection,statement);  
 }  
 if(e.getSource() == MemberShip )  
 {  
 new Memberships(connection,statement);  
 }  
 }  
}  
  
  
  
  
class Trainers extends JFrame implements ActionListener  
{  
 JLabel trainerIDLabel;  
 JLabel trainerNameLabel;  
 JLabel trainerAgeLabel;  
 JLabel trainerShiftLabel;  
 JLabel trainerSalaryLabel;  
 JTable trainerTable;  
 DefaultTableModel tableModel;  
 JTextField trainerIDField;  
 JTextField trainerNameField;  
 JTextField trainerAgeField;  
 JComboBox<String> trainerShiftBox;  
 JTextField trainerSalaryField;  
 JPanel components = new JPanel();  
 JPanel Table = new JPanel();  
 JPanel querybuttons = new JPanel();  
  
 JPanel GUIPanel = new JPanel();  
  
  
 JButton AddData = new JButton("Add Record");  
 JButton Search = new JButton("Search Record");  
 JButton Edit = new JButton("Edit Record");  
 JButton Delete = new JButton("Delete Record");  
  
  
 Connection con;  
 Statement statement;  
 Trainers(Connection con, Statement statement)  
 {  
 this.con = con;  
 this.statement = statement;  
 ComponentCreation();  
 setTrainerFrame();  
 setComponents();  
 setCRUDButtons();  
 setHeading();  
 ReadAllData();  
 setTable();  
 }  
  
 public void setTrainerFrame()  
 {  
 this.add(components);  
 this.add(Table);  
 this.add(querybuttons);  
 this.setTitle("Trainer Management ");  
 this.setSize(1300,1000);  
 this.setVisible(true);  
 this.setLayout(null);  
 this.setResizable(false);  
 this.getContentPane().setBackground(Color.white);  
 this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 this.setLocationRelativeTo(null);  
 this.setVisible(true);  
 this.setLayout(null);  
 }  
  
 public void ComponentCreation()  
 {  
 trainerIDLabel = new JLabel("Trainer ID");  
 trainerNameLabel = new JLabel("Trainer Name");  
 trainerAgeLabel = new JLabel("Trainer Age");  
 trainerShiftLabel = new JLabel("Trainer Shift");  
 trainerSalaryLabel = new JLabel("Trainer Salary");  
 trainerIDField = new JTextField();  
 trainerNameField = new JTextField();  
 trainerAgeField = new JTextField();  
 trainerShiftBox = new JComboBox<>(new String[]{"Morning", "Afternoon", "Evening", "Night"});  
 trainerSalaryField = new JTextField();  
 String[] columnNames = {"Trainer ID", "Trainer Name", "Trainer Age", "Trainer Shift", "Trainer Salary"};  
 tableModel = new DefaultTableModel(columnNames, 0);  
 trainerTable = new JTable(tableModel);  
  
 }  
 public void setComponents()  
 {  
  
 components.setSize(400,300);  
 components.setLayout(null);  
 components.setLocation(100,100);  
 components.setBackground(Color.white);  
  
 trainerIDLabel.setBounds(50,60,130,30);  
 trainerIDField.setBounds(150,60,200,30);  
 trainerNameLabel.setBounds(50,110,130,30);  
 trainerNameField.setBounds(150,110,200,30);  
 trainerAgeLabel.setBounds(50,160,130,30);  
 trainerAgeField.setBounds(150,160,200,30);  
 trainerShiftLabel.setBounds(50,210,130,30);  
 trainerShiftBox.setBounds(150,210,200,30);  
 trainerSalaryLabel.setBounds(50,260,130,30);  
 trainerSalaryField.setBounds(150,260,200,30);  
  
  
  
  
  
 components.add(trainerIDLabel);  
 components.add(trainerNameLabel);  
 components.add(trainerAgeLabel);  
 components.add(trainerShiftLabel);  
 components.add(trainerSalaryLabel);  
 components.add(trainerIDField);  
 components.add(trainerNameField);  
 components.add(trainerAgeField);  
 components.add(trainerShiftBox);  
 components.add(trainerSalaryField);  
  
 }  
 public void setTable()  
 {  
 Table.setSize(1000 ,400);  
 Table.setLocation(30,500);  
 Table.setLayout(null);  
 Table.setVisible(true);  
  
 JScrollPane scrolltable = new JScrollPane(trainerTable);  
  
 trainerTable.setBounds(0,0,1000,400);  
 scrolltable.setBounds(0,0,1000,400);  
 Table.add(scrolltable);  
  
 }  
 public void setHeading() {  
 ImageIcon EDClogo = new ImageIcon("src\\EDC2.png");  
 ImageIcon IBAlogo = new ImageIcon("src\\IBA1.png");  
  
  
 JLabel Heading = new JLabel("EDC GYM TRAINER MANAEGEMENT");  
 JLabel ImageEDC = new JLabel();  
 JLabel ImageIBA = new JLabel();  
 Heading.setFont(new Font("Bahnschrift",Font.BOLD,30));  
 ImageEDC.setIcon(EDClogo);  
 ImageIBA.setIcon(IBAlogo);  
 ImageEDC.setSize(300, 50);  
 ImageIBA.setSize(300, 50);  
 Heading.setSize(550, 50);  
 ImageEDC.setLocation(1000, 30);  
 ImageIBA.setLocation(50, 40);  
 Heading.setLocation(370, 40);  
  
  
 GUIPanel.setLocation(10,10);  
 GUIPanel.setSize(1200,100);  
 GUIPanel.setLayout(null);  
 GUIPanel.add(ImageEDC);  
 GUIPanel.add(ImageIBA);  
 GUIPanel.add(Heading);  
 GUIPanel.setBackground(Color.WHITE);  
 this.add(GUIPanel);  
 }  
  
 public void setCRUDButtons()  
 {  
  
 querybuttons.setBackground(Color.white);  
 querybuttons.setSize(900 ,70);  
 querybuttons.setLocation(30,400);  
 querybuttons.setLayout(null);  
 querybuttons.setVisible(true);  
  
  
 AddData.setBounds(50,30,150,30);  
 Search.setBounds(210,30,150,30);  
 Edit.setBounds(370,30,150,30);  
 Delete.setBounds(530,30,150,30);  
  
  
 AddData.addActionListener(this);  
 Search.addActionListener(this);  
 Delete.addActionListener(this);  
 Edit.addActionListener(this);  
  
 querybuttons.add(AddData);  
 querybuttons.add(Search);  
 querybuttons.add(Edit);  
 querybuttons.add(Delete);  
  
 }  
 public void clearFields() {  
 trainerIDField.setText("");  
 trainerNameField.setText("");  
 trainerAgeField.setText("");  
 trainerShiftBox.setSelectedIndex(0); // Reset shift box to the first item  
 trainerSalaryField.setText("");  
 }  
  
 public void refreshTableData() {  
  
 tableModel.setRowCount(0);  
 ReadAllData();  
  
 }  
  
  
 public void ReadAllData()  
 {  
  
 try {  
 System.out.println("Connected to Database Successfully!");  
 ResultSet r = statement.executeQuery("Select \* from trainer");  
 while (r.next()) {  
 Vector v = new Vector();  
 v.add(r.getInt(1));  
 v.add(r.getString(2));  
 v.add(r.getInt(3));  
 v.add(r.getString(4));  
 v.add(r.getDouble(5));  
 tableModel.addRow(v);  
  
 }  
  
 } catch (SQLException e) {  
 System.out.println(e.getMessage());  
 }  
 }  
 public void addRecords() {  
 String query = "INSERT INTO trainer (TrainerID,TrainerName,TrainerAge,TrainerShift,TrainerSalary) VALUES (?,?,?,?,?)";  
  
  
 try {  
 int id = Integer.parseInt(trainerIDField.getText());  
 String name = trainerNameField.getText();  
 int age = Integer.parseInt(trainerAgeField.getText());  
 String shift = (String) (trainerShiftBox.getSelectedItem());  
 int salary = Integer.parseInt(trainerSalaryField.getText());  
  
  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
 preparedStatement.setString(2, name);  
 preparedStatement.setInt(3, age);  
 preparedStatement.setString(4, shift);  
 preparedStatement.setDouble(5, salary);  
 preparedStatement.executeUpdate();  
 JOptionPane.showMessageDialog(null, "Record Added Successfully : \n" , "Record Added", JOptionPane.PLAIN\_MESSAGE);  
  
  
 refreshTableData();  
 clearFields();  
  
 preparedStatement.close();  
  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (NumberFormatException e) {  
 System.out.println("Invalid input format: " + e.getMessage());  
 }  
  
  
 }  
 public void deleteRecords()  
 {  
 int id = Integer.parseInt(trainerIDField.getText());  
  
 try {  
 // Query to delete the record for the given ID  
 String query = "DELETE FROM trainer WHERE TrainerID = ?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
  
 // Execute the delete query  
 int rowsDeleted = preparedStatement.executeUpdate();  
  
 if (rowsDeleted > 0) {  
 // If rows are deleted, clear the form fields and refresh the table data  
 JOptionPane.showMessageDialog(null, "Record deleted successfully", "Delete Result", JOptionPane.INFORMATION\_MESSAGE);  
 clearFields();  
 refreshTableData();  
 } else {  
 // If no rows are deleted (i.e., no record found for the given ID), display a message  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Delete Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
 // Close the statement  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid ID format", "Delete Result", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
  
 public void editRecords()  
 {  
 int id = Integer.parseInt(trainerIDField.getText());  
  
 try {  
  
 String query = "UPDATE trainer SET TrainerName=?, TrainerAge=?, TrainerShift=?, TrainerSalary=? WHERE TrainerID=?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setString(1, trainerNameField.getText());  
 preparedStatement.setInt(2, Integer.parseInt(trainerAgeField.getText()));  
 preparedStatement.setString(3, (String) trainerShiftBox.getSelectedItem());  
 preparedStatement.setDouble(4, Double.parseDouble(trainerSalaryField.getText()));  
 preparedStatement.setInt(5, id);  
  
  
 int rowsUpdated = preparedStatement.executeUpdate();  
  
 if (rowsUpdated > 0) {  
  
 JOptionPane.showMessageDialog(null, "Record updated successfully", "Update Result", JOptionPane.INFORMATION\_MESSAGE);  
 refreshTableData();  
 } else {  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Update Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid input format", "Update Result", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
  
  
 public void searchRecords()  
 {  
 int id = Integer.parseInt(trainerIDField.getText());  
  
 try {  
 // Query to select data for the given ID  
 String query = "SELECT \* FROM trainer WHERE TrainerID = ?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
 ResultSet resultSet = preparedStatement.executeQuery();  
  
 // If a record is found, display the data on the form  
 if (resultSet.next()) {  
 trainerNameField.setText(resultSet.getString("TrainerName"));  
 trainerAgeField.setText(String.valueOf(resultSet.getInt("TrainerAge")));  
 trainerShiftBox.setSelectedItem(resultSet.getString("TrainerShift"));  
 trainerSalaryField.setText(String.valueOf(resultSet.getDouble("TrainerSalary")));  
 } else {  
 // If no record is found for the given ID, display a message  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Search Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
 // Close the result set and statement  
 resultSet.close();  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid ID format", "Search Result", JOptionPane.ERROR\_MESSAGE);  
 }  
  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() == AddData)  
 {  
 addRecords();  
 }  
 else if (e.getSource() == Search)  
 {  
 searchRecords();  
 }  
 else if (e.getSource() == Edit)  
 {  
 editRecords();  
 }  
 else if (e.getSource() == Delete)  
 {  
 deleteRecords();  
 }  
 }  
}  
class Members extends JFrame implements ActionListener {  
 JLabel memberIDLabel;  
 JLabel memberNameLabel;  
 JLabel memberAgeLabel;  
 JLabel membershipIDLabel;  
 JLabel feesStatusLabel;  
 JLabel trainerIDLabel;  
 JLabel shiftLabel;  
 JTable memberTable;  
 DefaultTableModel tableModel;  
 JTextField memberIDField;  
 JTextField memberNameField;  
 JTextField memberAgeField;  
 JTextField membershipIDField;  
 JTextField trainerIDField;  
 JComboBox<String> feesStatusBox;  
 JComboBox<String> shiftBox;  
 JPanel components = new JPanel();  
 JPanel Table = new JPanel();  
 JPanel querybuttons = new JPanel();  
 JPanel GUIPanel = new JPanel();  
  
 JButton AddData = new JButton("Add Record");  
 JButton Search = new JButton("Search Record");  
 JButton Edit = new JButton("Edit Record");  
 JButton Delete = new JButton("Delete Record");  
  
 Connection con;  
 Statement statement;  
  
 Members(Connection con, Statement statement) {  
 this.con = con;  
 this.statement = statement;  
 ComponentCreation();  
 setMemberFrame();  
 setComponents();  
 setCRUDButtons();  
 setHeading();  
 ReadAllData();  
 setTable();  
 }  
  
 public void setMemberFrame() {  
 this.add(components);  
 this.add(Table);  
 this.add(querybuttons);  
 this.setTitle("EDC GYM MEMBER MANAEGEMENT");  
 this.setSize(1400, 1000);  
 this.getContentPane().setBackground(Color.WHITE);  
 this.setVisible(true);  
 this.setLayout(null);  
 this.setResizable(false);  
 this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 this.setLocationRelativeTo(null);  
 }  
  
 public void ComponentCreation() {  
 memberIDLabel = new JLabel("Member ID");  
 memberNameLabel = new JLabel("Member Name");  
 memberAgeLabel = new JLabel("Member Age");  
 membershipIDLabel = new JLabel("Membership ID");  
 feesStatusLabel = new JLabel("Fees Status");  
 trainerIDLabel = new JLabel("Trainer ID");  
 shiftLabel = new JLabel("Shift");  
  
 memberIDField = new JTextField();  
 memberNameField = new JTextField();  
 memberAgeField = new JTextField();  
 membershipIDField = new JTextField();  
 trainerIDField = new JTextField();  
 feesStatusBox = new JComboBox<>(new String[]{"Paid", "Unpaid"});  
 shiftBox = new JComboBox<>(new String[]{"Morning", "Afternoon", "Evening", "Night"});  
  
 String[] columnNames = {"Member ID", "Member Name", "Member Age", "Membership ID", "Fees Status", "Trainer ID", "Shift"};  
 tableModel = new DefaultTableModel(columnNames, 0);  
 memberTable = new JTable(tableModel);  
 }  
  
 public void setComponents() {  
 components.setSize(500, 280);  
 components.setLayout(null);  
 components.setLocation(50, 115);  
 components.setBackground(Color.WHITE);  
  
 memberIDLabel.setBounds(50, 10, 130, 30);  
 memberIDField.setBounds(200, 10, 200, 30);  
 memberNameLabel.setBounds(50, 50, 130, 30);  
 memberNameField.setBounds(200, 50, 200, 30);  
 memberAgeLabel.setBounds(50, 90, 130, 30);  
 memberAgeField.setBounds(200, 90, 200, 30);  
 membershipIDLabel.setBounds(50, 130, 130, 30);  
 membershipIDField.setBounds(200, 130, 200, 30);  
 feesStatusLabel.setBounds(50, 170, 130, 30);  
 feesStatusBox.setBounds(200, 170, 200, 30);  
 trainerIDLabel.setBounds(50, 210, 130, 30);  
 trainerIDField.setBounds(200, 210, 200, 30);  
 shiftLabel.setBounds(50, 250, 130, 30);  
 shiftBox.setBounds(200, 250, 200, 30);  
  
 components.add(memberIDLabel);  
 components.add(memberNameLabel);  
 components.add(memberAgeLabel);  
 components.add(membershipIDLabel);  
 components.add(feesStatusLabel);  
 components.add(trainerIDLabel);  
 components.add(shiftLabel);  
 components.add(memberIDField);  
 components.add(memberNameField);  
 components.add(memberAgeField);  
 components.add(membershipIDField);  
 components.add(feesStatusBox);  
 components.add(trainerIDField);  
 components.add(shiftBox);  
 }  
  
 public void setTable() {  
 Table.setSize(1300, 400);  
 Table.setLocation(30, 500);  
 Table.setLayout(null);  
 Table.setVisible(true);  
  
 JScrollPane scrolltable = new JScrollPane(memberTable);  
  
 memberTable.setBounds(0, 0, 1300, 400);  
 scrolltable.setBounds(0, 0, 1300, 400);  
 Table.add(scrolltable);  
 }  
  
 public void setHeading() {  
 ImageIcon EDClogo = new ImageIcon("src\\EDC2.png");  
 ImageIcon IBAlogo = new ImageIcon("src\\IBA1.png");  
  
  
 JLabel Heading = new JLabel("EDC GYM MEMBER MANAEGEMENT");  
 JLabel ImageEDC = new JLabel();  
 JLabel ImageIBA = new JLabel();  
 Heading.setFont(new Font("Bahnschrift",Font.BOLD,30));  
 ImageEDC.setIcon(EDClogo);  
 ImageIBA.setIcon(IBAlogo);  
 ImageEDC.setSize(300, 50);  
 ImageIBA.setSize(300, 50);  
 Heading.setSize(500, 50);  
 ImageEDC.setLocation(1100, 30);  
 ImageIBA.setLocation(50, 40);  
 Heading.setLocation(360, 40);  
  
  
 GUIPanel.setLocation(10,10);  
 GUIPanel.setSize(1300,100);  
 GUIPanel.setLayout(null);  
 GUIPanel.add(ImageEDC);  
 GUIPanel.add(ImageIBA);  
 GUIPanel.add(Heading);  
 GUIPanel.setBackground(Color.WHITE);  
 this.add(GUIPanel);  
 }  
  
 public void setCRUDButtons() {  
 querybuttons.setSize(900, 70);  
 querybuttons.setLocation(30, 400);  
 querybuttons.setLayout(null);  
 querybuttons.setVisible(true);  
 querybuttons.setBackground(Color.white);  
  
 AddData.setBounds(50, 30, 150, 30);  
 Search.setBounds(210, 30, 150, 30);  
 Edit.setBounds(370, 30, 150, 30);  
 Delete.setBounds(530, 30, 150, 30);  
  
 AddData.addActionListener(this);  
 Search.addActionListener(this);  
 Delete.addActionListener(this);  
 Edit.addActionListener(this);  
  
 querybuttons.add(AddData);  
 querybuttons.add(Search);  
 querybuttons.add(Edit);  
 querybuttons.add(Delete);  
 }  
  
 public void clearFields() {  
 memberIDField.setText("");  
 memberNameField.setText("");  
 memberAgeField.setText("");  
 membershipIDField.setText("");  
 feesStatusBox.setSelectedIndex(0);  
 trainerIDField.setText("");  
 shiftBox.setSelectedIndex(0);  
 }  
  
 public void refreshTableData() {  
 tableModel.setRowCount(0);  
 ReadAllData();  
 }  
  
 public void ReadAllData() {  
 try {  
 System.out.println("Connected to Database Successfully!");  
 ResultSet r = statement.executeQuery("Select \* from member");  
 while (r.next()) {  
 Vector v = new Vector();  
 v.add(r.getInt(1));  
 v.add(r.getString(2));  
 v.add(r.getInt(3));  
 v.add(r.getInt(4));  
 v.add(r.getBoolean(5) ? "Paid" : "Unpaid");  
 v.add(r.getInt(6));  
 v.add(r.getString(7));  
 tableModel.addRow(v);  
 }  
 } catch (SQLException e) {  
 System.out.println(e.getMessage());  
 }  
 }  
  
 public void addRecords() {  
 String query = "INSERT INTO member (MemID, MemName, MemAge, MembershipID, FeesStatus, TrainerID, Shift) VALUES (?,?,?,?,?,?,?)";  
  
 try {  
 int id = Integer.parseInt(memberIDField.getText());  
 String name = memberNameField.getText();  
 int age = Integer.parseInt(memberAgeField.getText());  
 int membershipID = Integer.parseInt(membershipIDField.getText());  
 boolean feesStatus = feesStatusBox.getSelectedItem().equals("Paid");  
 int trainerID = Integer.parseInt(trainerIDField.getText());  
 String shift = (String) (shiftBox.getSelectedItem());  
  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
 preparedStatement.setString(2, name);  
 preparedStatement.setInt(3, age);  
 preparedStatement.setInt(4, membershipID);  
 preparedStatement.setBoolean(5, feesStatus);  
 preparedStatement.setInt(6, trainerID);  
 preparedStatement.setString(7, shift);  
 preparedStatement.executeUpdate();  
 JOptionPane.showMessageDialog(null, "Record Added Successfully", "Record Added", JOptionPane.PLAIN\_MESSAGE);  
  
 refreshTableData();  
 clearFields();  
  
 preparedStatement.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (NumberFormatException e) {  
 System.out.println("Invalid input format: " + e.getMessage());  
 }  
 }  
  
 public void deleteRecords() {  
 int id = Integer.parseInt(memberIDField.getText());  
  
 try {  
 // Query to delete the record for the given ID  
 String query = "DELETE FROM member WHERE MemID = ?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
  
 // Execute the delete query  
 int rowsDeleted = preparedStatement.executeUpdate();  
  
 if (rowsDeleted > 0) {  
 // If rows are deleted, clear the form fields and refresh the table data  
 JOptionPane.showMessageDialog(null, "Record deleted successfully", "Delete Result", JOptionPane.INFORMATION\_MESSAGE);  
 clearFields();  
 refreshTableData();  
 } else {  
 // If no rows are deleted (i.e., no record found for the given ID), display a message  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Delete Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
 // Close the statement  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid ID format", "Delete Result", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
 public void editRecords() {  
 int id = Integer.parseInt(memberIDField.getText());  
  
 try {  
 String query = "UPDATE member SET MemName=?, MemAge=?, MembershipID=?, FeesStatus=?, TrainerID=?, Shift=? WHERE MemID=?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setString(1, memberNameField.getText());  
 preparedStatement.setInt(2, Integer.parseInt(memberAgeField.getText()));  
 preparedStatement.setInt(3, Integer.parseInt(membershipIDField.getText()));  
 preparedStatement.setBoolean(4, feesStatusBox.getSelectedItem().equals("Paid"));  
 preparedStatement.setInt(5, Integer.parseInt(trainerIDField.getText()));  
 preparedStatement.setString(6, (String) shiftBox.getSelectedItem());  
 preparedStatement.setInt(7, id);  
  
 int rowsUpdated = preparedStatement.executeUpdate();  
  
 if (rowsUpdated > 0) {  
 JOptionPane.showMessageDialog(null, "Record updated successfully", "Update Result", JOptionPane.INFORMATION\_MESSAGE);  
 refreshTableData();  
 } else {  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Update Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid input format", "Update Result", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
 public void searchRecords() {  
 int id = Integer.parseInt(memberIDField.getText());  
  
 try {  
 // Query to select data for the given ID  
 String query = "SELECT \* FROM member WHERE MemID = ?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
 ResultSet resultSet = preparedStatement.executeQuery();  
  
 // If a record is found, display the data on the form  
 if (resultSet.next()) {  
 memberNameField.setText(resultSet.getString("MemName"));  
 memberAgeField.setText(String.valueOf(resultSet.getInt("MemAge")));  
 membershipIDField.setText(String.valueOf(resultSet.getInt("MembershipID")));  
 feesStatusBox.setSelectedItem(resultSet.getBoolean("FeesStatus") ? "Paid" : "Unpaid");  
 trainerIDField.setText(String.valueOf(resultSet.getInt("TrainerID")));  
 shiftBox.setSelectedItem(resultSet.getString("Shift"));  
 } else {  
 // If no record is found for the given ID, display a message  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Search Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
 // Close the result set and statement  
 resultSet.close();  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid ID format", "Search Result", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() == AddData) {  
 addRecords();  
 } else if (e.getSource() == Search) {  
 searchRecords();  
 } else if (e.getSource() == Edit) {  
 editRecords();  
 } else if (e.getSource() == Delete) {  
 deleteRecords();  
 }  
 }  
}  
class Memberships extends JFrame implements ActionListener {  
 JLabel membershipIDLabel;  
 JLabel membershipNameLabel;  
 JLabel cardioLabel;  
 JLabel bodybuildingLabel;  
 JLabel personalTrainerLabel;  
 JLabel membershipFeesLabel;  
 JTable membershipTable;  
 DefaultTableModel tableModel;  
 JTextField membershipIDField;  
 JTextField membershipNameField;  
 JCheckBox cardioCheckBox;  
 JCheckBox bodybuildingCheckBox;  
 JCheckBox personalTrainerCheckBox;  
 JTextField membershipFeesField;  
 JPanel components = new JPanel();  
 JPanel tablePanel = new JPanel();  
 JPanel queryButtons = new JPanel();  
 JPanel GUIPanel = new JPanel();  
  
 JButton addDataButton = new JButton("Add Record");  
 JButton searchButton = new JButton("Search Record");  
 JButton editButton = new JButton("Edit Record");  
 JButton deleteButton = new JButton("Delete Record");  
  
 Connection con;  
 Statement statement;  
  
 Memberships(Connection con, Statement statement) {  
 this.con = con;  
 this.statement = statement;  
 createComponents();  
 setMembershipFrame();  
 setComponents();  
 setCRUDButtons();  
 setHeading();  
 readAllData();  
 setTable();  
 }  
  
 public void setMembershipFrame() {  
 this.add(components);  
 this.add(tablePanel);  
 this.add(queryButtons);  
 this.setTitle("EDC Gym Membership Management");  
 this.setSize(1300, 1000);  
 this.setVisible(true);  
 this.getContentPane().setBackground(Color.WHITE);  
 this.setLayout(null);  
 this.setResizable(false);  
 this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 this.setLocationRelativeTo(null);  
 }  
  
 public void createComponents() {  
 membershipIDLabel = new JLabel("Membership ID");  
 membershipNameLabel = new JLabel("Membership Name");  
 cardioLabel = new JLabel("Cardio");  
 bodybuildingLabel = new JLabel("Bodybuilding");  
 personalTrainerLabel = new JLabel("Personal Trainer");  
 membershipFeesLabel = new JLabel("Membership Fees");  
  
 membershipIDField = new JTextField();  
 membershipNameField = new JTextField();  
 cardioCheckBox = new JCheckBox();  
 bodybuildingCheckBox = new JCheckBox();  
 personalTrainerCheckBox = new JCheckBox();  
 membershipFeesField = new JTextField();  
  
 String[] columnNames = {"Membership ID", "Membership Name", "Cardio", "Bodybuilding", "Personal Trainer", "Membership Fees"};  
 tableModel = new DefaultTableModel(columnNames, 0);  
 membershipTable = new JTable(tableModel);  
 }  
  
 public void setComponents() {  
 components.setSize(450, 300);  
 components.setLayout(null);  
 components.setLocation(50, 130);  
 components.setBackground(Color.white);  
  
 membershipIDLabel.setBounds(50, 10, 130, 30);  
 membershipIDField.setBounds(200, 10, 200, 30);  
 membershipNameLabel.setBounds(50, 60, 130, 30);  
 membershipNameField.setBounds(200, 60, 200, 30);  
 cardioLabel.setBounds(50, 110, 130, 30);  
 cardioCheckBox.setBounds(200, 110, 30, 30);  
 bodybuildingLabel.setBounds(50, 160, 130, 30);  
 bodybuildingCheckBox.setBounds(200, 160, 30, 30);  
 personalTrainerLabel.setBounds(50, 210, 130, 30);  
 personalTrainerCheckBox.setBounds(200, 210, 30, 30);  
 membershipFeesLabel.setBounds(50, 260, 130, 30);  
 membershipFeesField.setBounds(200, 260, 200, 30);  
  
 components.add(membershipIDLabel);  
 components.add(membershipNameLabel);  
 components.add(cardioLabel);  
 components.add(bodybuildingLabel);  
 components.add(personalTrainerLabel);  
 components.add(membershipFeesLabel);  
 components.add(membershipIDField);  
 components.add(membershipNameField);  
 components.add(cardioCheckBox);  
 components.add(bodybuildingCheckBox);  
 components.add(personalTrainerCheckBox);  
 components.add(membershipFeesField);  
 }  
  
 public void setTable() {  
 tablePanel.setSize(1000, 400);  
 tablePanel.setLocation(30, 550);  
 tablePanel.setLayout(null);  
 tablePanel.setVisible(true);  
  
 JScrollPane scrollTable = new JScrollPane(membershipTable);  
  
 membershipTable.setBounds(0, 0, 1000, 400);  
 scrollTable.setBounds(0, 0, 1000, 400);  
 tablePanel.add(scrollTable);  
 }  
  
 public void setHeading() {  
 ImageIcon EDClogo = new ImageIcon("src\\EDC2.png");  
 ImageIcon IBAlogo = new ImageIcon("src\\IBA1.png");  
  
  
 JLabel Heading = new JLabel("EDC GYM MEMBER MANAEGEMENT");  
 JLabel ImageEDC = new JLabel();  
 JLabel ImageIBA = new JLabel();  
 Heading.setFont(new Font("Bahnschrift",Font.BOLD,30));  
 ImageEDC.setIcon(EDClogo);  
 ImageIBA.setIcon(IBAlogo);  
 ImageEDC.setSize(300, 50);  
 ImageIBA.setSize(300, 50);  
 Heading.setSize(500, 50);  
 ImageEDC.setLocation(1100, 30);  
 ImageIBA.setLocation(50, 40);  
 Heading.setLocation(360, 40);  
  
  
 GUIPanel.setLocation(10,10);  
 GUIPanel.setSize(1300,100);  
 GUIPanel.setLayout(null);  
 GUIPanel.add(ImageEDC);  
 GUIPanel.add(ImageIBA);  
 GUIPanel.add(Heading);  
 GUIPanel.setBackground(Color.WHITE);  
 this.add(GUIPanel);  
 }  
  
 public void setCRUDButtons() {  
 queryButtons.setSize(900, 70);  
 queryButtons.setLocation(30, 460);  
 queryButtons.setLayout(null);  
 queryButtons.setVisible(true);  
 queryButtons.setBackground(Color.white);  
 addDataButton.setBounds(50, 30, 150, 30);  
 searchButton.setBounds(210, 30, 150, 30);  
 editButton.setBounds(370, 30, 150, 30);  
 deleteButton.setBounds(530, 30, 150, 30);  
  
 addDataButton.addActionListener(this);  
 searchButton.addActionListener(this);  
 deleteButton.addActionListener(this);  
 editButton.addActionListener(this);  
  
 queryButtons.add(addDataButton);  
 queryButtons.add(searchButton);  
 queryButtons.add(editButton);  
 queryButtons.add(deleteButton);  
 }  
  
 public void clearFields() {  
 membershipIDField.setText("");  
 membershipNameField.setText("");  
 cardioCheckBox.setSelected(false);  
 bodybuildingCheckBox.setSelected(false);  
 personalTrainerCheckBox.setSelected(false);  
 membershipFeesField.setText("");  
 }  
  
 public void refreshTableData() {  
 tableModel.setRowCount(0);  
 readAllData();  
 }  
  
 public void readAllData() {  
 try {  
 ResultSet rs = statement.executeQuery("SELECT \* FROM MemberShipPlan");  
 while (rs.next()) {  
 Vector<Object> row = new Vector<>();  
 row.add(rs.getInt("MemberShipID"));  
 row.add(rs.getString("MemberShipName"));  
 row.add(rs.getBoolean("Cardio"));  
 row.add(rs.getBoolean("Bodybuilding"));  
 row.add(rs.getBoolean("PersonalTrainer"));  
 row.add(rs.getDouble("MemberShipFees"));  
 tableModel.addRow(row);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void addRecords() {  
 String query = "INSERT INTO MemberShipPlan (MemberShipID, MemberShipName, Cardio, Bodybuilding, PersonalTrainer, MemberShipFees) VALUES (?, ?, ?, ?, ?, ?)";  
  
 try {  
 int id = Integer.parseInt(membershipIDField.getText());  
 String name = membershipNameField.getText();  
 boolean cardio = cardioCheckBox.isSelected();  
 boolean bodybuilding = bodybuildingCheckBox.isSelected();  
 boolean personalTrainer = personalTrainerCheckBox.isSelected();  
 double fees = Double.parseDouble(membershipFeesField.getText());  
  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
 preparedStatement.setString(2, name);  
 preparedStatement.setBoolean(3, cardio);  
 preparedStatement.setBoolean(4, bodybuilding);  
 preparedStatement.setBoolean(5, personalTrainer);  
 preparedStatement.setDouble(6, fees);  
 preparedStatement.executeUpdate();  
 JOptionPane.showMessageDialog(null, "Record Added Successfully", "Record Added", JOptionPane.PLAIN\_MESSAGE);  
  
 refreshTableData();  
 clearFields();  
  
 preparedStatement.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (NumberFormatException e) {  
 JOptionPane.showMessageDialog(null, "Invalid input format", "Error", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
 public void deleteRecords() {  
 int id = Integer.parseInt(membershipIDField.getText());  
  
 try {  
 String query = "DELETE FROM MemberShipPlan WHERE MemberShipID = ?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
  
 int rowsDeleted = preparedStatement.executeUpdate();  
  
 if (rowsDeleted > 0) {  
 JOptionPane.showMessageDialog(null, "Record deleted successfully", "Delete Result", JOptionPane.INFORMATION\_MESSAGE);  
 clearFields();  
 refreshTableData();  
 } else {  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Delete Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid ID format", "Delete Result", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
 public void editRecords() {  
 int id = Integer.parseInt(membershipIDField.getText());  
  
 try {  
 String query = "UPDATE MemberShipPlan SET MemberShipName=?, Cardio=?, Bodybuilding=?, PersonalTrainer=?, MemberShipFees=? WHERE MemberShipID=?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setString(1, membershipNameField.getText());  
 preparedStatement.setBoolean(2, cardioCheckBox.isSelected());  
 preparedStatement.setBoolean(3, bodybuildingCheckBox.isSelected());  
 preparedStatement.setBoolean(4, personalTrainerCheckBox.isSelected());  
 preparedStatement.setDouble(5, Double.parseDouble(membershipFeesField.getText()));  
 preparedStatement.setInt(6, id);  
  
 int rowsUpdated = preparedStatement.executeUpdate();  
  
 if (rowsUpdated > 0) {  
 JOptionPane.showMessageDialog(null, "Record updated successfully", "Update Result", JOptionPane.INFORMATION\_MESSAGE);  
 refreshTableData();  
 } else {  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Update Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid input format", "Update Result", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
 public void searchRecords() {  
 int id = Integer.parseInt(membershipIDField.getText());  
  
 try {  
 String query = "SELECT \* FROM MemberShipPlan WHERE MemberShipID = ?";  
 PreparedStatement preparedStatement = con.prepareStatement(query);  
 preparedStatement.setInt(1, id);  
 ResultSet resultSet = preparedStatement.executeQuery();  
  
 if (resultSet.next()) {  
 membershipNameField.setText(resultSet.getString("MemberShipName"));  
 cardioCheckBox.setSelected(resultSet.getBoolean("Cardio"));  
 bodybuildingCheckBox.setSelected(resultSet.getBoolean("Bodybuilding"));  
 personalTrainerCheckBox.setSelected(resultSet.getBoolean("PersonalTrainer"));  
 membershipFeesField.setText(String.valueOf(resultSet.getDouble("MemberShipFees")));  
 } else {  
 JOptionPane.showMessageDialog(null, "No record found for ID: " + id, "Search Result", JOptionPane.INFORMATION\_MESSAGE);  
 }  
  
 resultSet.close();  
 preparedStatement.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Invalid ID format", "Search Result", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() == addDataButton) {  
 addRecords();  
 } else if (e.getSource() == searchButton) {  
 searchRecords();  
 } else if (e.getSource() == editButton) {  
 editRecords();  
 } else if (e.getSource() == deleteButton) {  
 deleteRecords();  
 }  
 }  
  
 public static void main(String[] args) {  
 // Test your class here  
 }  
}  
**Output :**

* **Login Page**
* **User name: root password : root**

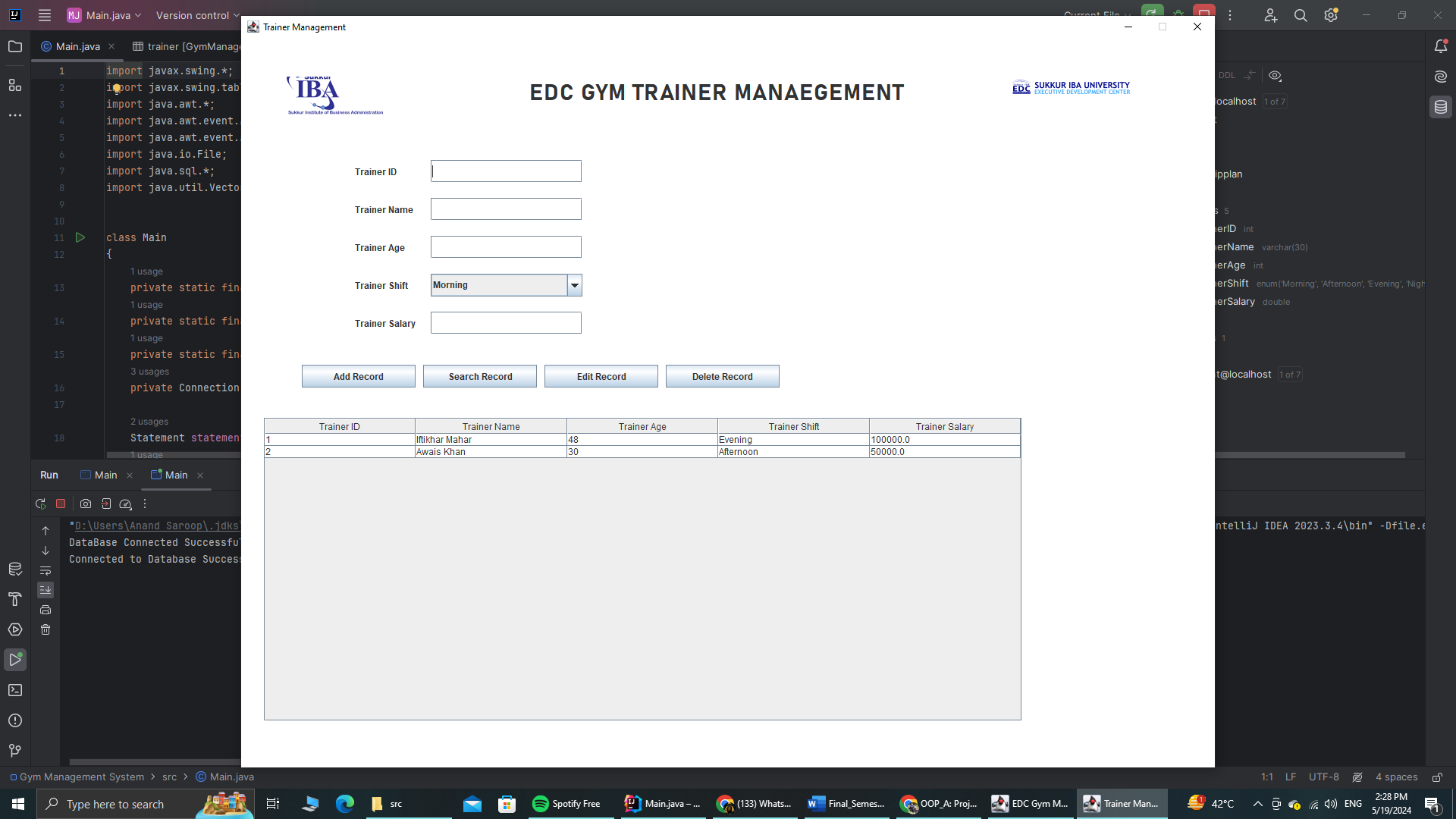




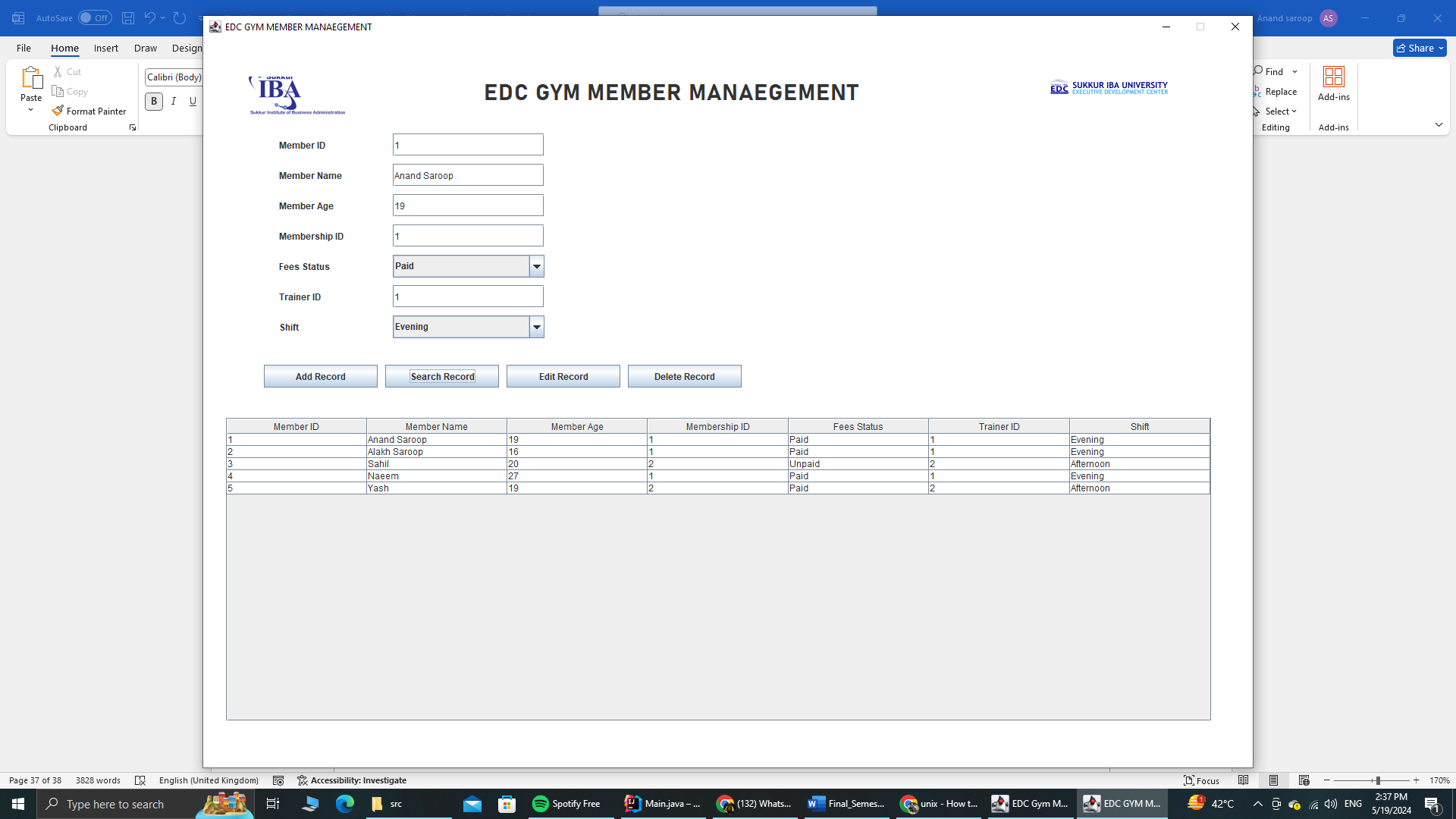
* **Main Page**



* **Trainer Page**



* **Member Page**



* **MemberShip Page**

Sorry Heading should edc gym membership management

But it is edc gym member management. Mistakenly written

