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;

Public class GymManagementSystem extends JFrame {

Private static final long serialVersionUID = 1L;

Private JPanel contentPane;

Private JTextField firstNameField;

Private JTextField lastNameField;

Private JTextField emailField;

Private JTextField membershipTypeField;

Private JTable table;

Private DefaultTableModel tableModel;

// Database URL

Private static final String DB\_URL = “jdbc:ucanaccess://C:/Users/linta/Documents/GymManagementSystem.accdb”;

Public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

Public void run() {

Try {

GymManagementSystem frame = new GymManagementSystem();

Frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

Public GymManagementSystem() {

// Frame settings

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 600, 400);

contentPane = new JPanel();

contentPane.setLayout(new BorderLayout());

setContentPane(contentPane);

// Create the panel for input fields

JPanel inputPanel = new JPanel();

inputPanel.setLayout(new GridLayout(5, 2, 5, 5));

inputPanel.add(new JLabel(“First Name:”));

firstNameField = new JTextField();

inputPanel.add(firstNameField);

inputPanel.add(new JLabel(“Last Name:”));

lastNameField = new JTextField();

inputPanel.add(lastNameField);

inputPanel.add(new JLabel(“Email:”));

emailField = new JTextField();

inputPanel.add(emailField);

inputPanel.add(new JLabel(“Membership Type:”));

membershipTypeField = new JTextField();

inputPanel.add(membershipTypeField);

contentPane.add(inputPanel, BorderLayout.NORTH);

// Create the buttons panel

JPanel buttonPanel = new JPanel();

buttonPanel.setLayout(new FlowLayout());

JButton addButton = new JButton(“Add Member”);

buttonPanel.add(addButton);

JButton updateButton = new JButton(“Update Member”);

buttonPanel.add(updateButton);

JButton deleteButton = new JButton(“Delete Member”);

buttonPanel.add(deleteButton);

contentPane.add(buttonPanel, BorderLayout.SOUTH);

// Create table to display members

tableModel = new DefaultTableModel();

tableModel.addColumn(“First Name”);

tableModel.addColumn(“Last Name”);

tableModel.addColumn(“Email”);

tableModel.addColumn(“Membership Type”);

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

contentPane.add(scrollPane, BorderLayout.CENTER);

// Load all members from the database

loadMembersFromDatabase();

// Add Member Button Action

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String firstName = firstNameField.getText();

String lastName = lastNameField.getText();

String email = emailField.getText();

String membershipType = membershipTypeField.getText();

// Create a new member and insert into the database

insertMember(firstName, lastName, email, membershipType);

// Update the table

loadMembersFromDatabase();

// Clear input fields

clearFields();

}

});

// Update Member Button Action

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int selectedRow = table.getSelectedRow();

if (selectedRow != -1) {

String firstName = firstNameField.getText();

String lastName = lastNameField.getText();

String email = emailField.getText();

String membershipType = membershipTypeField.getText();

Int memberId = (Integer) tableModel.getValueAt(selectedRow, 0);

// Update member in the database

updateMember(memberId, firstName, lastName, email, membershipType);

// Update the table

loadMembersFromDatabase();

// Clear input fields

clearFields();

} else {

JOptionPane.showMessageDialog(null, “Please select a member to update”);

}

}

});

// Delete Member Button Action

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int selectedRow = table.getSelectedRow();

if (selectedRow != -1) {

int memberId = (Integer) tableModel.getValueAt(selectedRow, 0);

// Delete member from the database

deleteMember(memberId);

// Update the table

loadMembersFromDatabase();

} else {

JOptionPane.showMessageDialog(null, “Please select a member to delete”);

}

}

});

}

// Method to load members from the database and update the table

Private void loadMembersFromDatabase() {

ArrayList<Member> members = getMembersFromDatabase();

updateTable(members);

}

// Method to update the table with the current members list

Private void updateTable(ArrayList<Member> members) {

// Clear the current table data

tableModel.setRowCount(0);

// Add members to the table

For (Member member : members) {

tableModel.addRow(new Object[] { member.getId(), member.getFirstName(), member.getLastName(), member.getEmail(), member.getMembershipType() });

}

}

// Method to clear input fields

Private void clearFields() {

firstNameField.setText(“”);

lastNameField.setText(“”);

emailField.setText(“”);

membershipTypeField.setText(“”);

}

// Method to get members from the database

Private ArrayList<Member> getMembersFromDatabase() {

ArrayList<Member> members = new ArrayList<>();

Try (Connection conn = DriverManager.getConnection(DB\_URL)) {

String query = “SELECT \* FROM Members”;

PreparedStatement pst = conn.prepareStatement(query);

ResultSet rs = pst.executeQuery();

While (rs.next()) {

Int id = rs.getInt(“MemberID”);

String firstName = rs.getString(“FirstName”);

String lastName = rs.getString(“LastName”);

String email = rs.getString(“Email”);

String membershipType = rs.getString(“MembershipType”);

Members.add(new Member(id, firstName, lastName, email, membershipType));

}

} catch (SQLException e) {

e.printStackTrace();

}

Return members;

}

// Method to insert a new member into the database

Private void insertMember(String firstName, String lastName, String email, String membershipType) {

Try (Connection conn = DriverManager.getConnection(DB\_URL)) {

String query = “INSERT INTO Members (FirstName, LastName, Email, MembershipType) VALUES (?, ?, ?, ?)”;

PreparedStatement pst = conn.prepareStatement(query);

Pst.setString(1, firstName);

Pst.setString(2, lastName);

Pst.setString(3, email);

Pst.setString(4, membershipType);

Pst.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

// Method to update a member in the database

Private void updateMember(int id, String firstName, String lastName, String email, String membershipType) {

Try (Connection conn = DriverManager.getConnection(DB\_URL)) {

String query = “UPDATE Members SET FirstName = ?, LastName = ?, Email = ?, MembershipType = ? WHERE MemberID = ?”;

PreparedStatement pst = conn.prepareStatement(query);

Pst.setString(1, firstName);

Pst.setString(2, lastName);

Pst.setString(3, email);

Pst.setString(4, membershipType);

Pst.setInt(5, id);

Pst.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

// Method to delete a member from the database

Private void deleteMember(int id) {

Try (Connection conn = DriverManager.getConnection(DB\_URL)) {

String query = “DELETE FROM Members WHERE MemberID = ?”;

PreparedStatement pst = conn.prepareStatement(query);

Pst.setInt(1, id);

Pst.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

// Member class representing the data model

Class Member {

Private int id;

Private String firstName;

Private String lastName;

Private String email;

Private String membershipType;

Public Member(int id, String firstName, String lastName, String email, String membershipType) {

This.id = id;

This.firstName = firstName;

This.lastName = lastName;

This.email = em