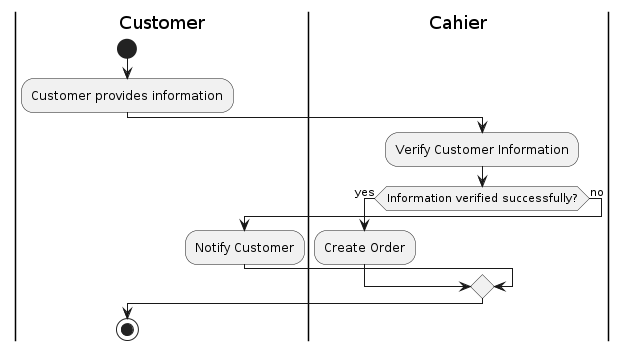
**Customer Registration**

**1 - Description:**

The `CustomerManagementForm` class is responsible for managing customer information within a GUI application. It extends the `SimpleForm` class and provides functionalities to interact with customer data, including adding, updating, and searching for customers.

**2 - Activity diagram**



**2 - Class Variables:**

- `Citymap`: A HashMap that maps city names to their corresponding IDs.

- Type: `HashMap<String, Integer>`

**3 - Constructor:**

- `CustomerManagementForm()`

- Description: Initializes the form components, loads city data, and populates the customer table.

- Parameters: None

**4 - Methods:**

**clean()**

- Description: Resets all input fields and buttons to their initial state.

- Access Modifier: Public

- Return Type: Void

| public void clean() {  jTextField2.setText("");  jTextField3.setText("");  jTextField4.setText("");  jTextField5.setText("");  jTextField6.setText("");  jTextField7.setText("");  jTextField1.setText("");  jComboBox1.setSelectedIndex(0);  jTable1.setEnabled(true);  jButton1.setEnabled(true);  jButton3.setEnabled(true);  } |
| --- |

**loadcity()**

- Description: Loads city data from the database into the city combo box and populates the `Citymap`.

- Access Modifier: Private

- Return Type: Void

| private void loadcity() {  try {  //search  ResultSet rs = MySQL.execute("SELECT \* FROM `city`");  Vector v = new Vector();  v.add("Select");  while (rs.next()) {  v.add(rs.getString("city\_name"));  Citymap.put(rs.getString("city\_name"), rs.getInt("idCity"));  }  DefaultComboBoxModel m = new DefaultComboBoxModel(v);  jComboBox1.setModel(m);  } catch (Exception e) {  LOGGER.log(Level.SEVERE, "Error loading cities", e);  }  } |
| --- |

**Customers(String column, String order, String Search)**

- Description: Retrieves customer data from the database based on the search criteria and populates the customer table.

- Access Modifier: Private

- Parameters:

- `column`: The column to sort by

- `order`: The sorting order (ASC or DESC)

- `Search`: The search string

- Return Type: Void

private void Customers(String column, String order, String Search) {

try {

ResultSet rs = MySQL.execute("SELECT \* FROM `customer` INNER JOIN `city` ON `customer`.`City\_id`= `city`.`idCity` WHERE `mobile` LIKE '" + Search + "%' OR `fname` LIKE '" + Search + "%' OR `lname` LIKE '" + Search + "%' OR `address` LIKE '" + Search + "%' OR `NIC` LIKE '" + Search + "%' OR `email` LIKE '" + Search + "%' ORDER BY `" + column + "` " + order + " ");

DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

model.setRowCount(0);

while (rs.next()) {

Vector<String> v = new Vector();

v.add(rs.getString("customer\_id"));

v.add(rs.getString("fname"));

v.add(rs.getString("lname"));

v.add(rs.getString("mobile"));

v.add(rs.getString("address"));

v.add(rs.getString("city.city\_name"));

v.add(rs.getString("NIC"));

v.add(rs.getString("email"));

model.addRow(v);

}

jTable1.setModel(model); // Set the model outside the loop

} catch (Exception e) {

LOGGER.log(Level.SEVERE, "Error loading customers", e);

}

}

**sendSMS(String recipient, String senderId, String message)**

- Description: Sends an SMS notification using a third-party API.

- Access Modifier: Public

- Parameters:

- `recipient`: The recipient's phone number

- `senderId`: The sender's ID

- `message`: The message content

- Return Type: Void

public static void sendSMS(String recipient, String senderId, String message) {

try {

// Set the API endpoint URL

String apiUrl = "https://sms.send.lk/api/v3/sms/send";

// Set your API access token

String apiToken = "1940|kWL80z5lZmj0Ad1gKoD0oLaQ8HILC2iuNOiKP6R8 ";

// Create JSON payload

String payload = String.format("{\"recipient\": \"%s\", \"sender\_id\": \"%s\", \"message\": \"%s\"}", recipient, senderId, message);

// Create URL object

URL url = new URL(apiUrl);

// Open HTTP connection

HttpURLConnection con = (HttpURLConnection) url.openConnection();

// Set request method

con.setRequestMethod("POST");

// Set request headers

con.setRequestProperty("Authorization", "Bearer " + apiToken);

con.setRequestProperty("Accept", "application/json");

con.setRequestProperty("Content-Type", "application/json");

// Enable output and set payload

con.setDoOutput(true);

try (DataOutputStream wr = new DataOutputStream(con.getOutputStream())) {

wr.write(payload.getBytes(StandardCharsets.UTF\_8));

}

// Read response

try (BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()))) {

String inputLine;

StringBuilder response = new StringBuilder();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

// Print response

System.out.println(response.toString());

}

// Close connection

con.disconnect();

} catch (Exception e) {

LOGGER.log(Level.SEVERE, "Error sending SMS", e);

}

}

**Event Handlers:**

- `jTextField1KeyReleased(java.awt.event.KeyEvent evt)`

- `jTable1MouseClicked(java.awt.event.MouseEvent evt)`

- `jButton1ActionPerformed(java.awt.event.ActionEvent evt)`

- `jButton2ActionPerformed(java.awt.event.ActionEvent evt)`

- Additional event handlers for form interactions.

**Helper Methods:**

- `search()`: Performs a search based on the selected sort order and the text in the search field.

**private void search() {**

**int sort = jComboBox2.getSelectedIndex();**

**if (sort == 0) {**

**Customers("customer\_id", "ASC", jTextField1.getText());**

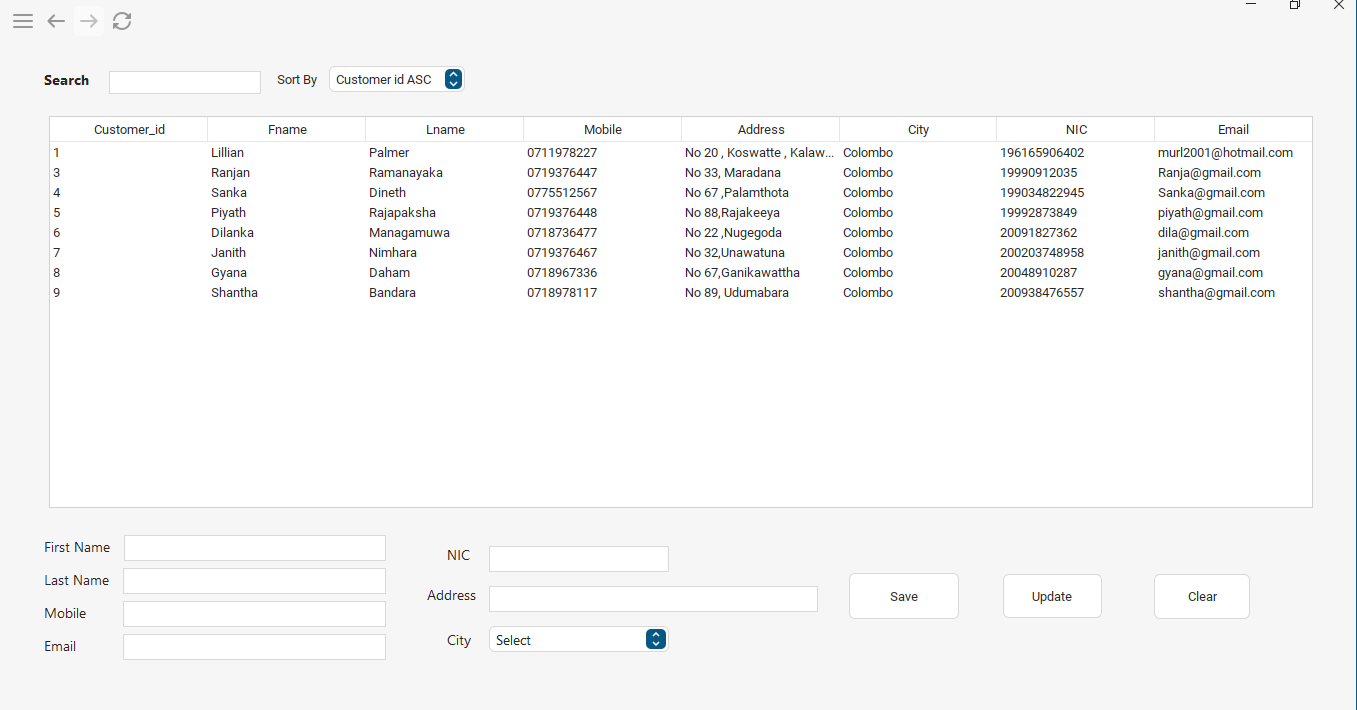
**} else if (sort == 1) {**

**Customers("customer\_id", "DESC", jTextField1.getText());**

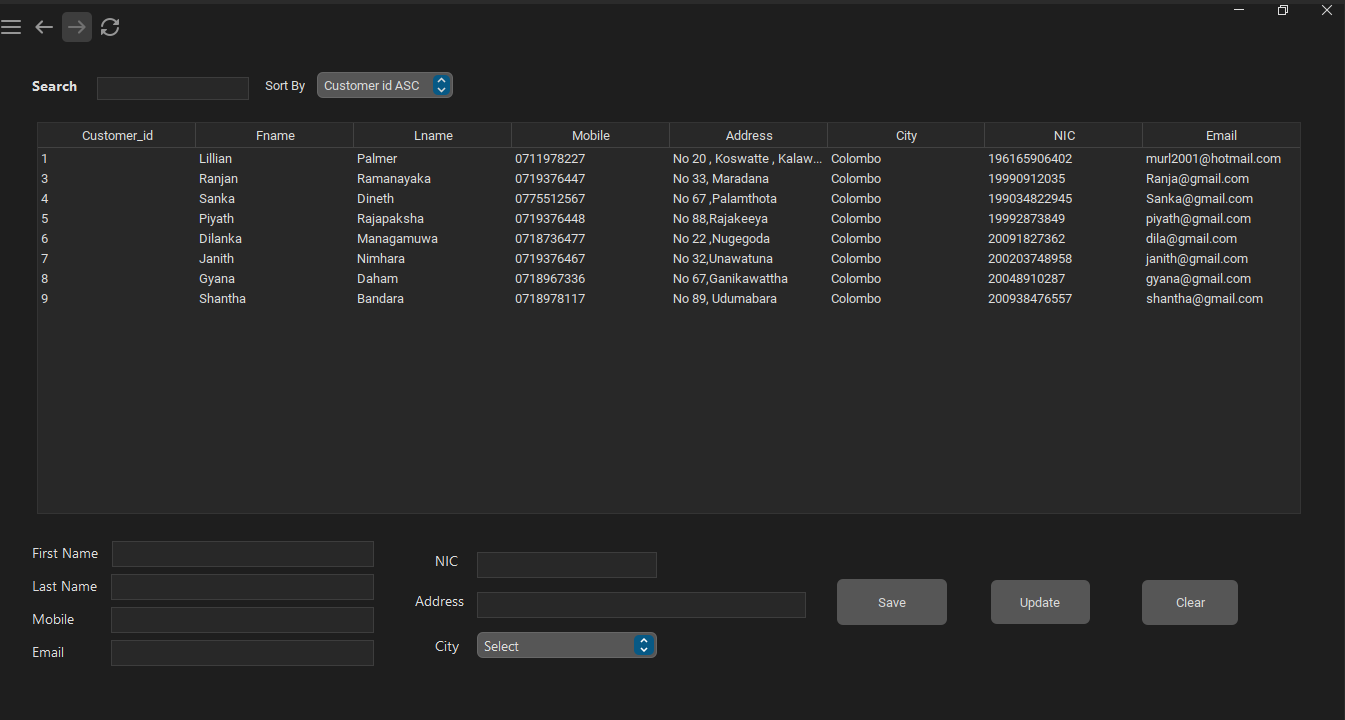
**}**

**}**

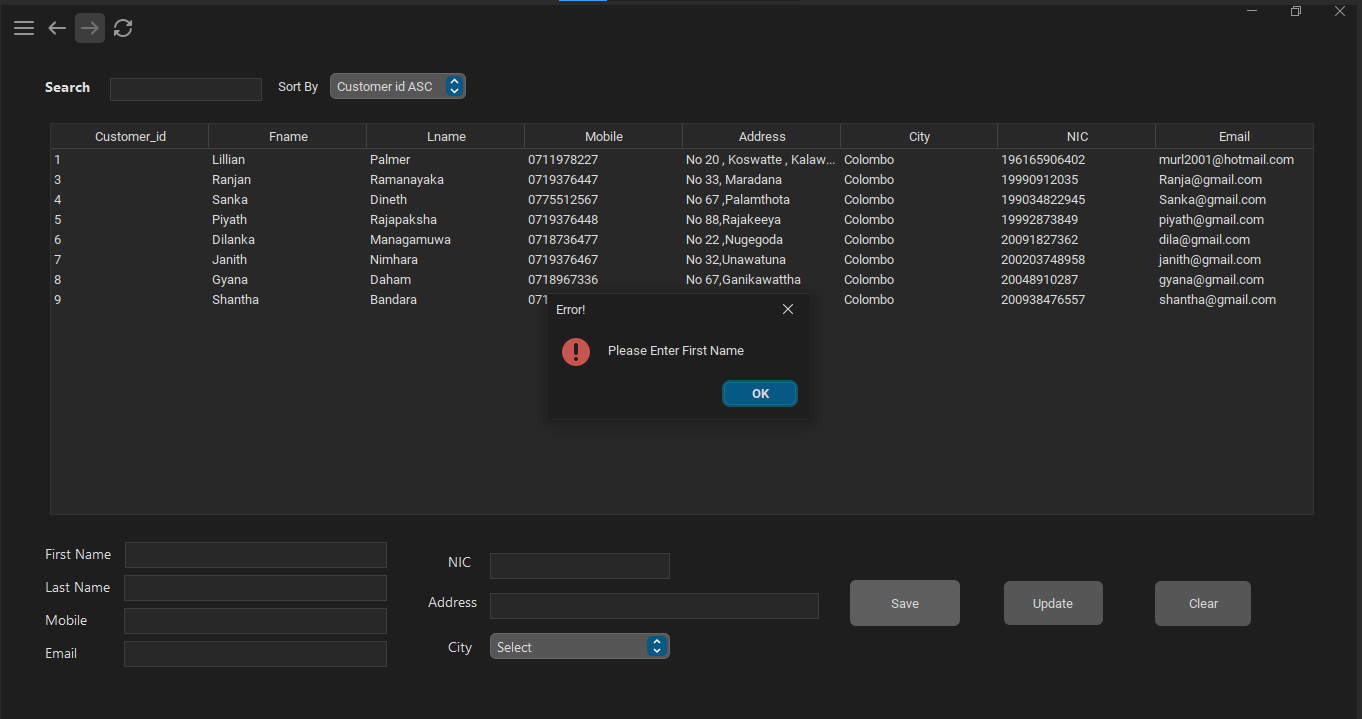
**5 - GUI**



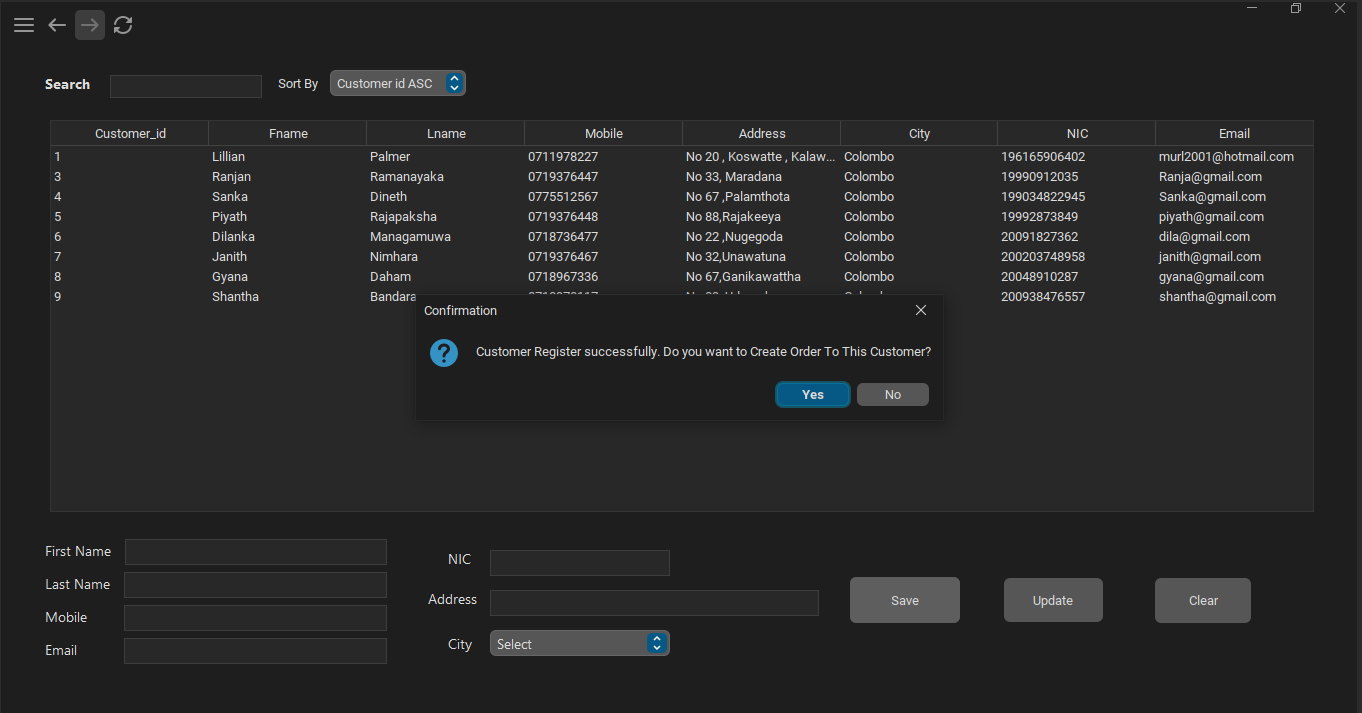
Img 1 - Light Theme Interface



Img 2 - Dark Theme Interface



Img 3 - Error Handling



Img 4 - Confirmation Message