ATA BASE MANAGEMENT SYSTEM

ASSIGNMENT-1

Project Name: “Asset Tracker”

**ABSTRACT**

Asset Tracker is a database management system designed to help organizations manage their assets efficiently. The system allows the organization to track the location, status, and maintenance history of their assets. Asset Tracker is a web-based application that can be accessed from anywhere at any time. The system is user-friendly and has an intuitive interface that makes it easy to use.

The database is designed to store information about various types of assets such as computers, furniture, vehicles, and equipment. The system allows the user to add, update, and delete assets from the database. It also allows the user to generate reports on the status and location of the assets.

The system uses various tables to store information about the assets, including the asset ID, name, type, location, and maintenance history. The user table stores information about the users who have access to the system. The maintenance table stores information about the maintenance performed on the assets. The location table stores information about the location of the assets.

Asset Tracker helps organizations to streamline their asset management processes, reduce costs, and increase efficiency. By using Asset Tracker, organizations can keep track of their assets, reduce downtime, and ensure that their assets are in good working condition. Overall, Asset Tracker is an essential tool for organizations looking to improve their asset management practices.

List Of Tables:

1.User Table

2.Asset Table

3.Location Table

4.Assignment Table

5.Maintenance Table

Attributes and Domain Types:

User Table:

1.user\_id number

2.username varchar2(50)

3.password varchar2(50)

4.email varchar2(50)

5.phone varchar2(50)

Asset Table:

1.asset\_id number

2.asset\_name varchar2(50)

3.asset\_type varchar2(50)

4.purchase\_date date

5.purchase\_cost float

Location Table:

1.location\_id number

2.location\_name varchar2(50)

3.address varchar2(50)

4.city varchar2(50)

5.state varchar2(50)

6.country varchar2(50)

Assignment Table:

1.assignment\_id number

2.asset\_id number

3.uder\_id number

4.assigned\_date date

5.return\_date date

6.location\_id number

Maintenance Table:

1.Maintenance\_id number

2.asset\_id number

3.maintenance\_date date

4.maintenance\_description varchar2(100)

5.maintenance\_cost float

Key Constraints:

User Table:

1.user\_id-primary key

Asset Table:

1.asset\_id-primary key

Location Table:

1.location\_id-primary key

Assignment Table:

1.assignment\_id-primary key

2.asset\_id-foreign key

1.assignment\_id-primary key

2.location\_id-foreign key

3.user\_id-foreign key

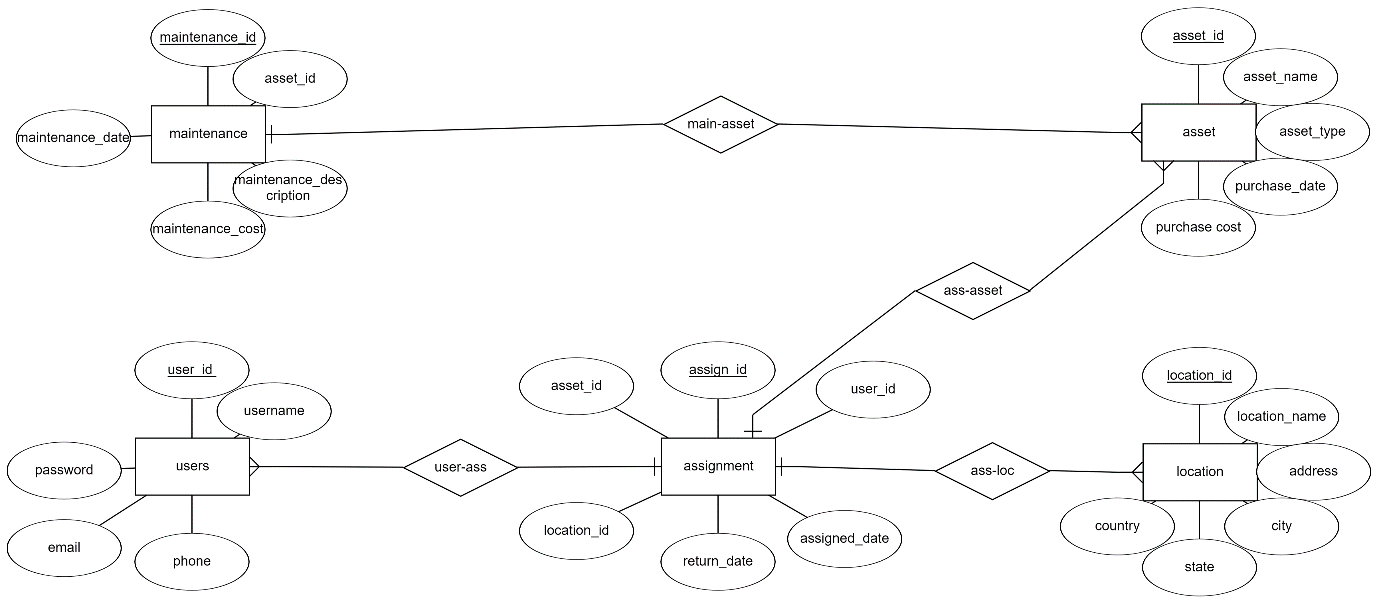
4.location\_id-foreign key

Maintenance Table:

1.maintenance\_id-primary key

2.asset\_id-foreign key

ER DIAGRAM:



**DDL COMMANDS:**

**1)CREATE:**

User Table:

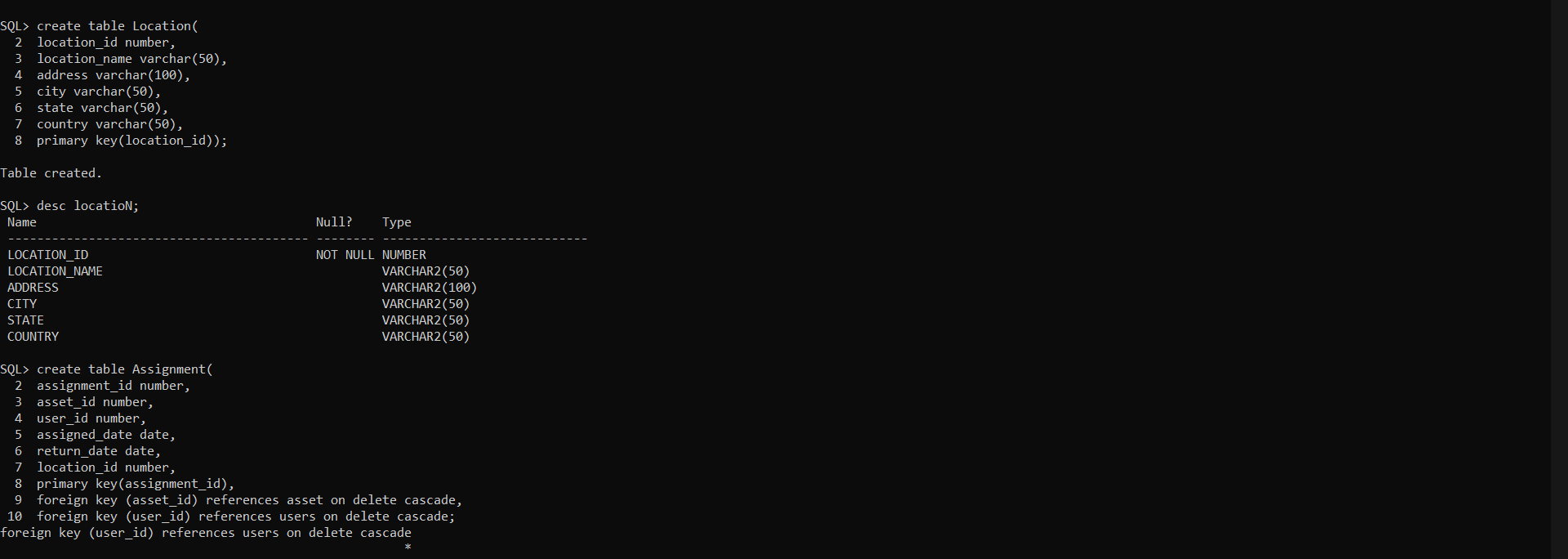


Asset Table:





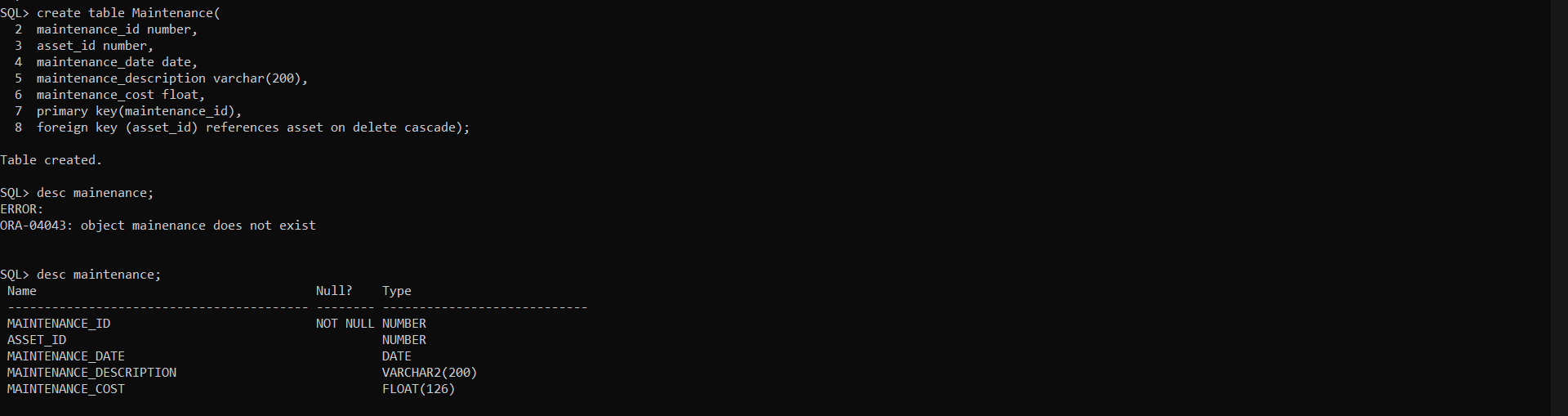
Location:



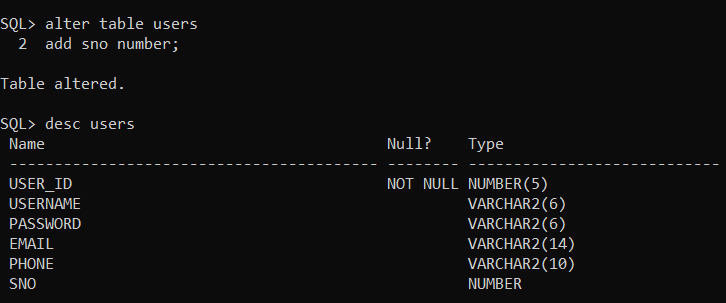
Assignment:



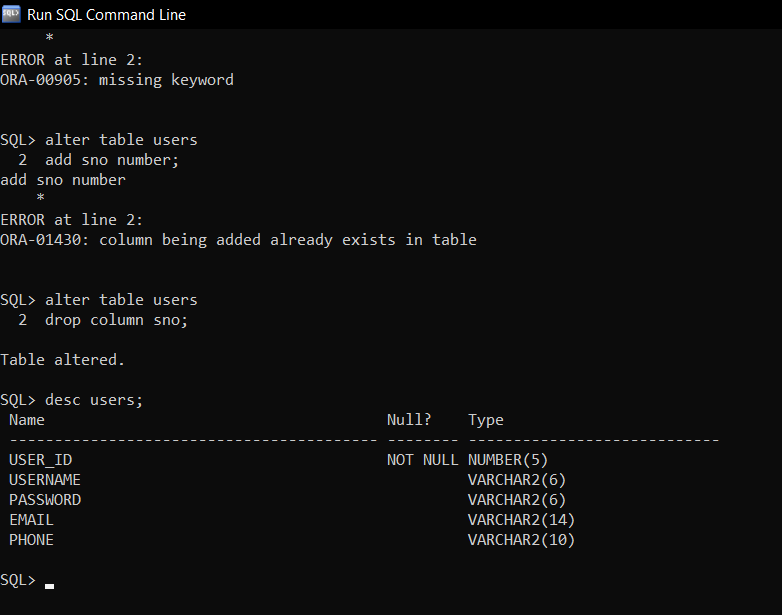
Maintenance:



**2)ALTER:**

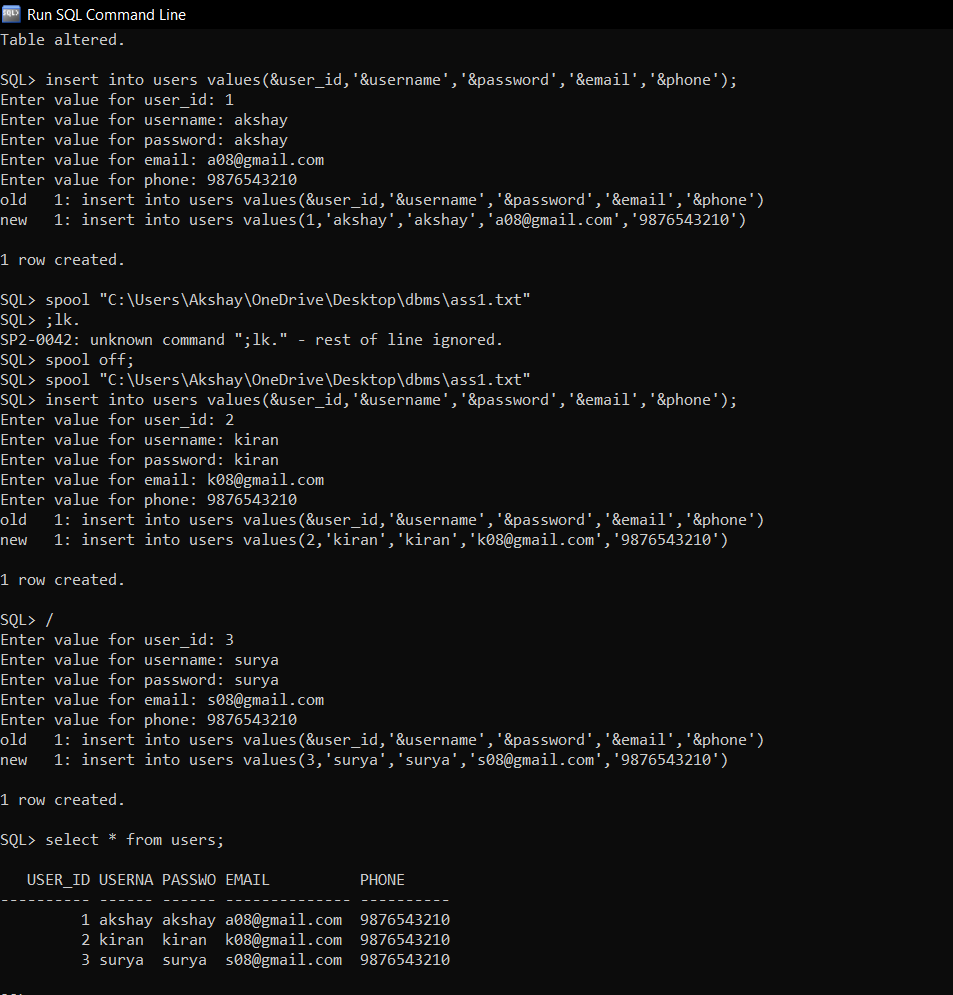


**3)DROP:**

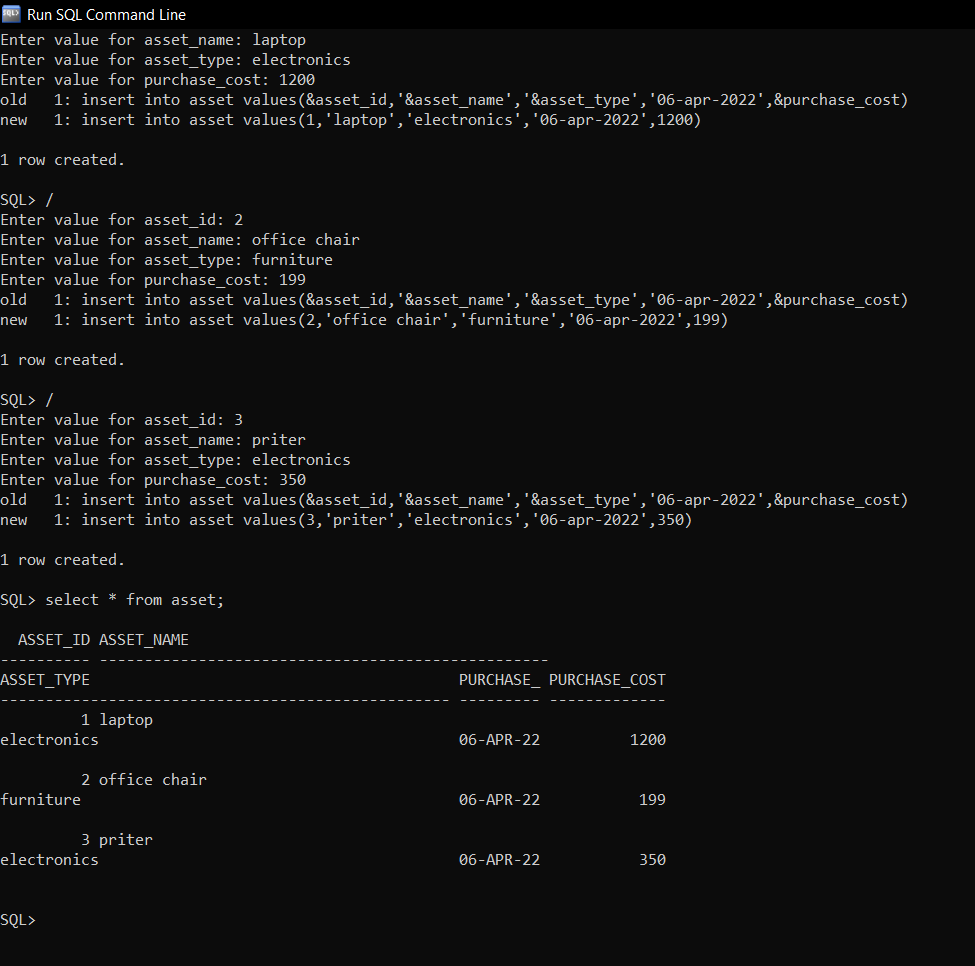


**DML COMMANDS:**

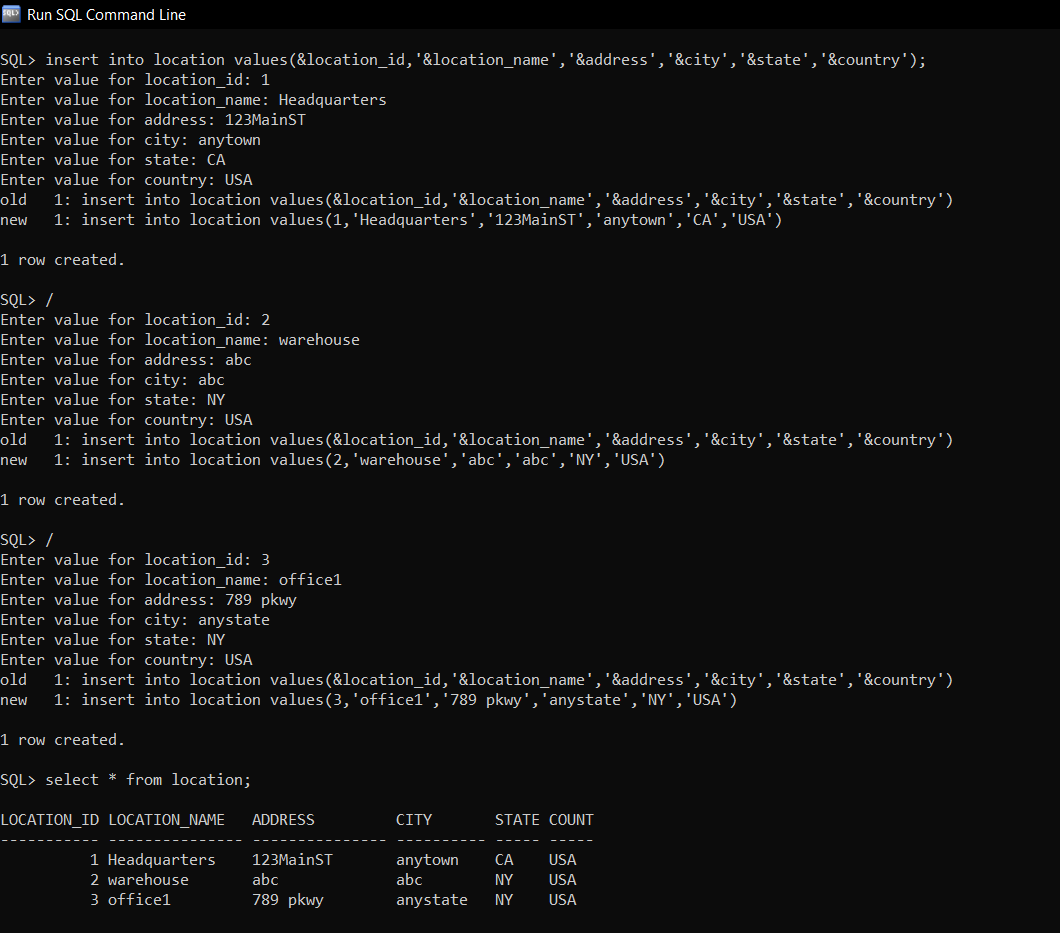
User Table:



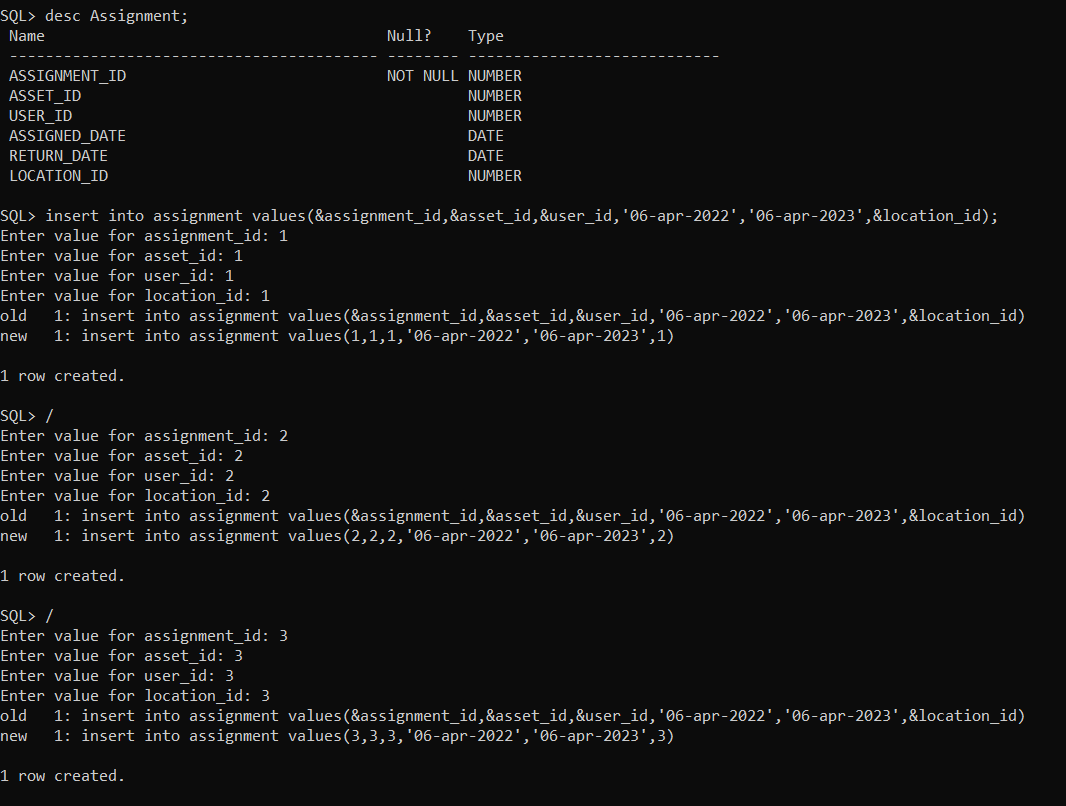
Asset Table:

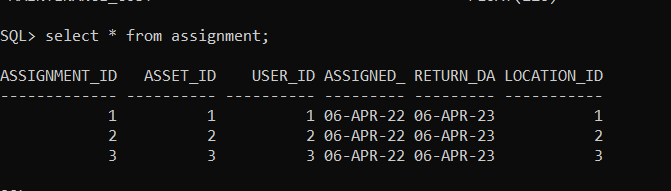


Location:

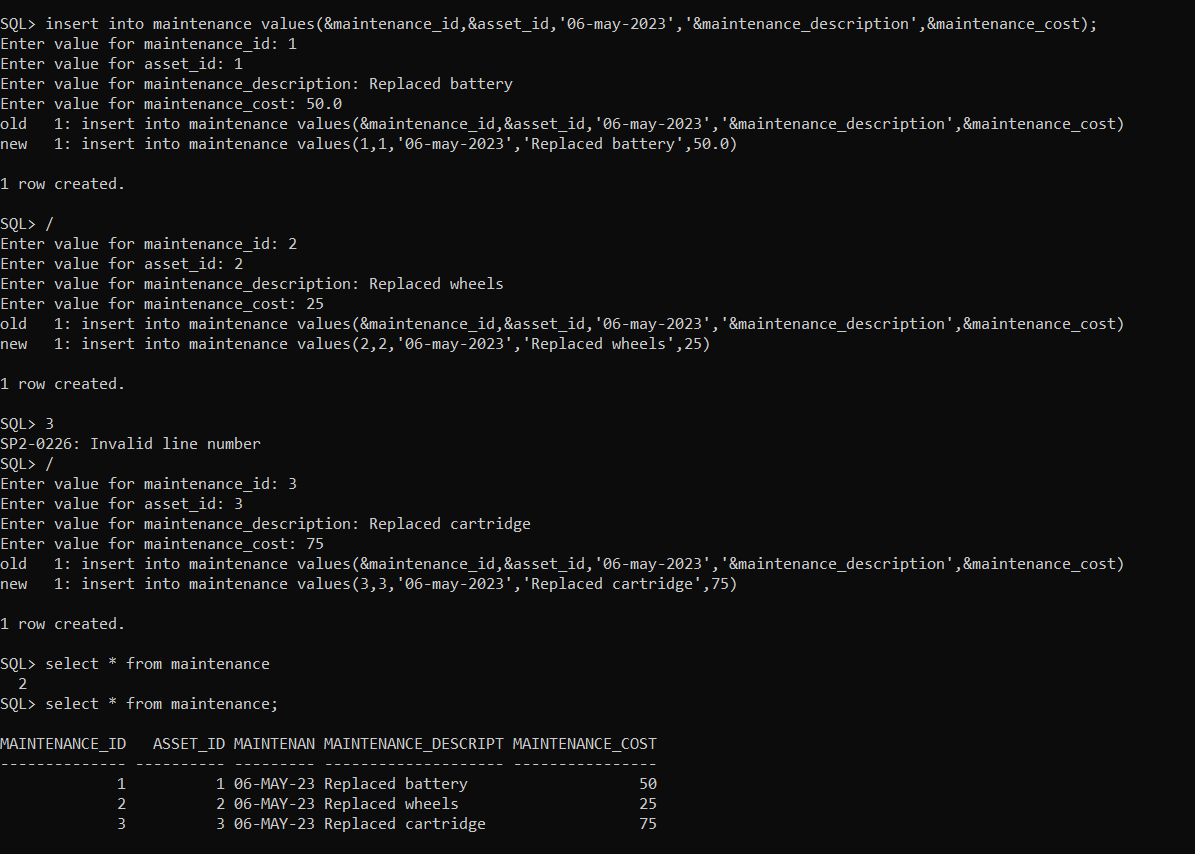


Assignment table:





Maintenance:



**IMPLEMENTATION**

**JAVA- SQL Connectivity using JDBC:**

**Java Database Connectivity (JDBC)** is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

package main;

import java.sql.\*;

public class ConnectionManager (

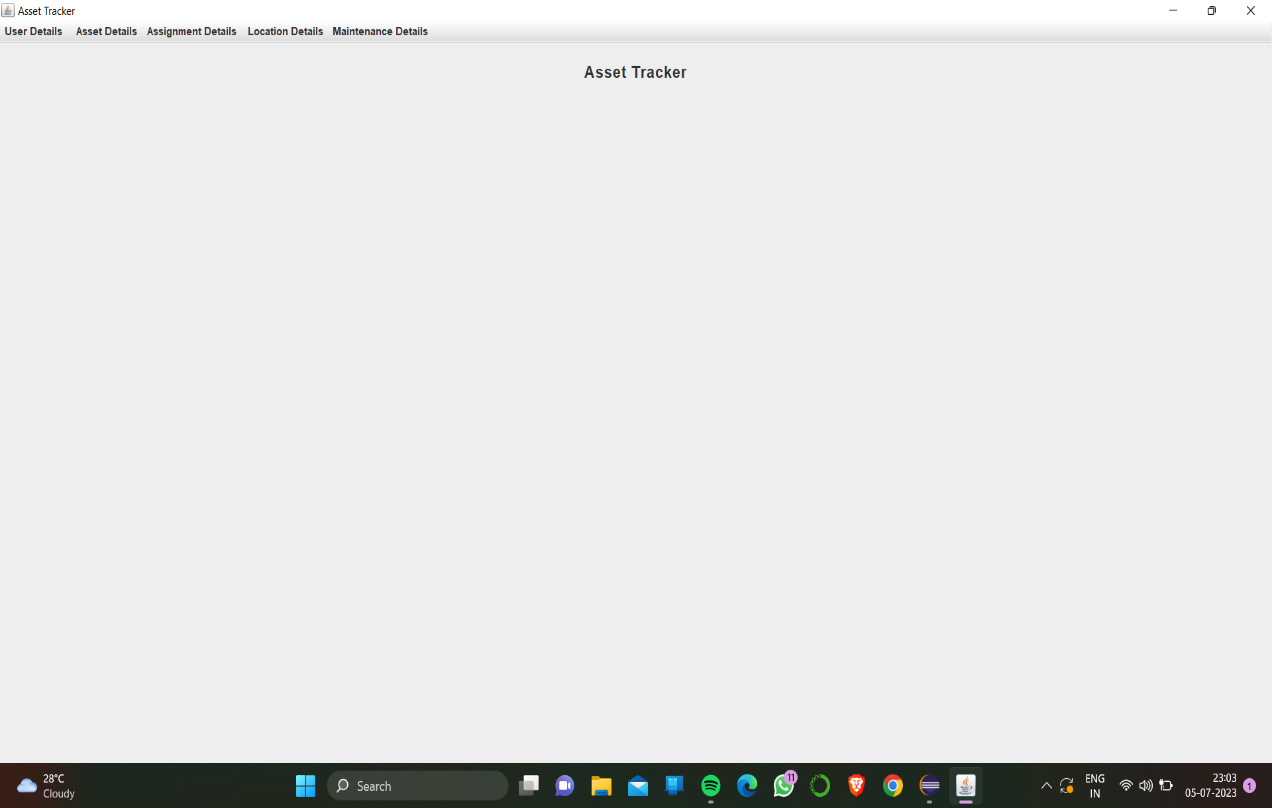
private static String url = "jdbc:oracle: thin:@localhost:1521:xe"; private static String username = "akshayi";

private static String password = "akshay2003";

private static Connection con;

public static Connection getConnectionn() throws Exception [ con = DriverManager.getConnection (url, username, password);

return con;

**Main Page:** 

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**public class MainPage extends JFrame {**

**/\***

**\***

**\*/**

**private static final long serialVersionUID = 1L;**

**//private JButton retrieveMarksButton;**

**public MainPage() {**

**// Set frame properties**

**setTitle("Asset Tracker");**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**// Create label**

**JLabel welcomeLabel = new JLabel("Asset Tracker");**

**welcomeLabel.setFont(new Font("Arial", Font.BOLD, 18));**

**welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);**

**welcomeLabel.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0));**

**add(welcomeLabel, BorderLayout.NORTH);**

**// Create panel for the button**

**/\*JPanel buttonPanel = new JPanel();**

**retrieveMarksButton = new JButton("Retrieve Marks");**

**buttonPanel.add(retrieveMarksButton);**

**\*/**

**// Create menu bar**

**JMenuBar menuBar = new JMenuBar();**

**// Create menus**

**JMenu UserMenu = new JMenu("User Details");**

**JMenu AssetMenu = new JMenu(" Asset Details");**

**JMenu AssignmentMenu = new JMenu("Assignment Details");**

**JMenu LocationMenu = new JMenu("Location Details");**

**JMenu MaintenanceMenu = new JMenu("Maintenance Details");**

**// Create menu item for student menu**

**JMenuItem viewUserDetails = new JMenuItem("View User Details");**

**viewUserDetails.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**new UserTableGUI();**

**}**

**});**

**// Create menu item for course menu**

**JMenuItem viewAssetDetails = new JMenuItem("View Asset Details");**

**viewAssetDetails.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**new AssetTableGUI();**

**}**

**});**

**// Create menu item for enrollment menu**

**JMenuItem viewAssignmentDetails = new JMenuItem("View Assignment Details");**

**viewAssignmentDetails.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**new AssignmentTableGUI();**

**}**

**});**

**// Create menu item for semester menu**

**JMenuItem viewLocationDetails = new JMenuItem("View Location Details");**

**viewLocationDetails.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**new LocationTableGUI();**

**}**

**});**

**// Create menu item for grade menu**

**JMenuItem viewMaintenanceDetails = new JMenuItem("View Maintenance Details");**

**viewMaintenanceDetails.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**new MaintenanceTableGUI();**

**}**

**});**

**// Add menu items to respective menus**

**UserMenu.add(viewUserDetails);**

**AssetMenu.add(viewAssetDetails);**

**AssignmentMenu.add(viewAssignmentDetails);**

**LocationMenu.add(viewLocationDetails);**

**MaintenanceMenu.add(viewMaintenanceDetails);**

**// Add menus to the menu bar**

**menuBar.add(UserMenu);**

**menuBar.add(AssetMenu);**

**menuBar.add(AssignmentMenu);**

**menuBar.add(LocationMenu);**

**menuBar.add(MaintenanceMenu);**

**// Set the menu bar**

**setJMenuBar(menuBar);**

**// Add the button panel to the frame**

**// add(buttonPanel, BorderLayout.CENTER);**

**// Set button action for "Retrieve Marks"**

**/\*retrieveMarksButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**new Retreive();**

**}**

**});\*/**

**// Add window listener to handle maximizing the window**

**addWindowStateListener(new WindowStateListener() {**

**public void windowStateChanged(WindowEvent e) {**

**if ((e.getNewState() & Frame.MAXIMIZED\_BOTH) == Frame.MAXIMIZED\_BOTH) {**

**System.out.println("Window maximized");**

**} else {**

**System.out.println("Window not maximized");**

**}**

**}**

**});**

**// Set frame size and visibility**

**setSize(800, 600);**

**setVisible(true);**

**}**

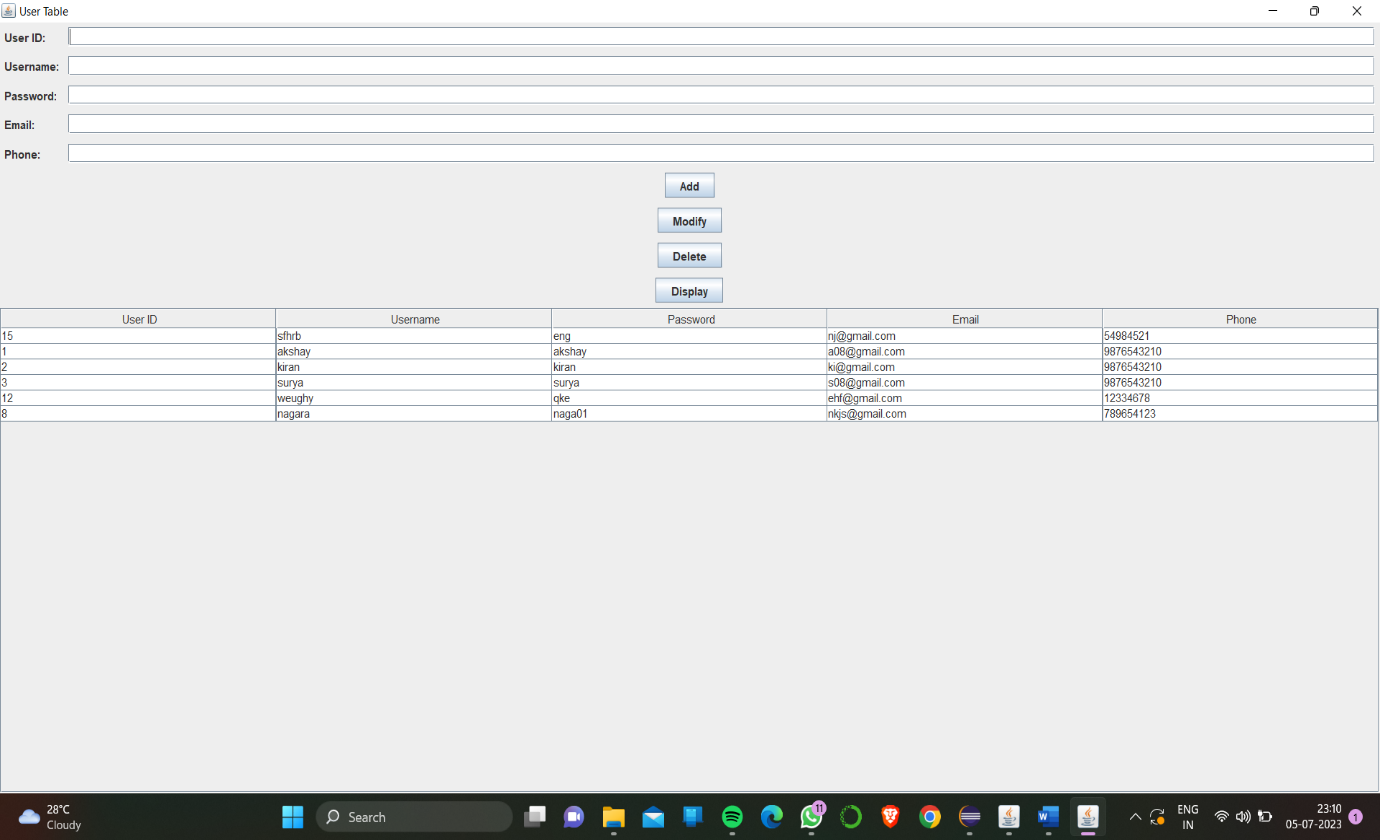
**public static void main(String[] args) {**

**new MainPage();**

**}**

**}**

**User table:**



import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class UserTableGUI extends JFrame {

private JTextField txtUserId, txtUsername, txtPassword, txtEmail, txtPhone;

private JTable tblUsers;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public UserTableGUI() {

initializeUI();

connectToDatabase();

displayUsers();

}

private void initializeUI() {

txtUserId = new JTextField();

txtUsername = new JTextField();

txtPassword = new JTextField();

txtEmail = new JTextField();

txtPhone = new JTextField();

tblUsers = new JTable();

tblUsers.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblUsers.getSelectionModel().addListSelectionListener(e -> selectUser());

JScrollPane scrollPane = new JScrollPane(tblUsers);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("User ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Username:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Password:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Email:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Phone:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtUserId, gbc);

gbc.gridy++;

panel.add(txtUsername, gbc);

gbc.gridy++;

panel.add(txtPassword, gbc);

gbc.gridy++;

panel.add(txtEmail, gbc);

gbc.gridy++;

panel.add(txtPhone, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertUser());

btnModify.addActionListener(e -> modifyUser());

btnDelete.addActionListener(e -> deleteUser());

btnDisplay.addActionListener(e -> displayUsers());

setTitle("User Table");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "akshay";

String password = "akshay2003";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertUser() {

String userId = txtUserId.getText();

String username = txtUsername.getText();

String password = txtPassword.getText();

String email = txtEmail.getText();

String phone = txtPhone.getText();

try {

String query = "INSERT INTO users (user\_id, username, password, email, phone) VALUES (?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, userId);

statement.setString(2, username);

statement.setString(3, password);

statement.setString(4, email);

statement.setString(5, phone);

statement.executeUpdate();

clearFields();

displayUsers();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyUser() {

int selectedRow = tblUsers.getSelectedRow();

if (selectedRow >= 0) {

String userId = txtUserId.getText();

String username = txtUsername.getText();

String password = txtPassword.getText();

String email = txtEmail.getText();

String phone = txtPhone.getText();

try {

String query = "UPDATE users SET username=?, password=?, email=?, phone=? WHERE user\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, username);

statement.setString(2, password);

statement.setString(3, email);

statement.setString(4, phone);

statement.setString(5, userId);

statement.executeUpdate();

clearFields();

displayUsers();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select a user to modify.");

}

}

private void deleteUser() {

int selectedRow = tblUsers.getSelectedRow();

if (selectedRow >= 0) {

String userId = tblUsers.getValueAt(selectedRow, 0).toString();

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this user?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM users WHERE user\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, userId);

statement.executeUpdate();

clearFields();

displayUsers();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select a user to delete.");

}

}

private void displayUsers() {

try {

String query = "SELECT \* FROM users";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

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

while (resultSet.next()) {

int userId = resultSet.getInt("user\_id");

String username = resultSet.getString("username");

String password = resultSet.getString("password");

String email = resultSet.getString("email");

String phone = resultSet.getString("phone");

users.add(new User(userId, username, password, email, phone));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"User ID", "Username", "Password", "Email", "Phone"});

for (User user : users) {

model.addRow(new Object[]{user.getUserId(), user.getUsername(), user.getPassword(), user.getEmail(), user.getPhone()});

}

tblUsers.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectUser() {

int selectedRow = tblUsers.getSelectedRow();

if (selectedRow >= 0) {

int userId = (int) tblUsers.getValueAt(selectedRow, 0);

String username = tblUsers.getValueAt(selectedRow, 1).toString();

String password = tblUsers.getValueAt(selectedRow, 2).toString();

String email = tblUsers.getValueAt(selectedRow, 3).toString();

String phone = tblUsers.getValueAt(selectedRow, 4).toString();

txtUserId.setText(String.valueOf(userId));

txtUsername.setText(username);

txtPassword.setText(password);

txtEmail.setText(email);

txtPhone.setText(phone);

}

}

private void clearFields() {

txtUserId.setText("");

txtUsername.setText("");

txtPassword.setText("");

txtEmail.setText("");

txtPhone.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(UserTableGUI::new);

}

private class User {

private int userId;

private String username;

private String password;

private String email;

private String phone;

public User(int userId, String username, String password, String email, String phone) {

this.userId = userId;

this.username = username;

this.password = password;

this.email = email;

this.phone = phone;

}

public int getUserId() {

return userId;

}

public String getUsername() {

return username;

}

public String getPassword() {

return password;

}

public String getEmail() {

return email;

}

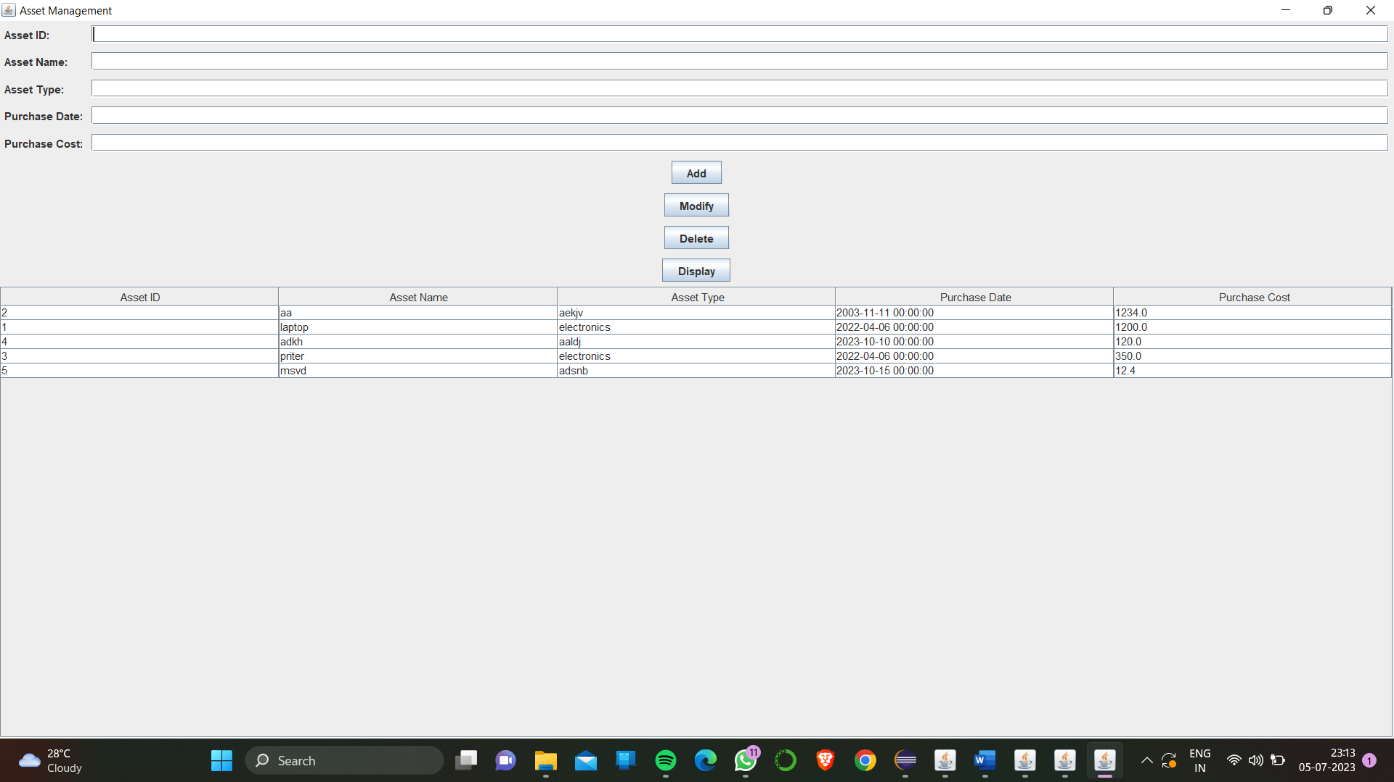
public String getPhone() {

return phone;

}

}

}Asset Table:



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.List;

public class AssetTableGUI extends JFrame {

private JTextField txtAssetId, txtAssetName, txtAssetType, txtPurchaseDate, txtPurchaseCost;

private JTable tblAssets;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public AssetTableGUI() {

initializeUI();

connectToDatabase();

displayAssets();

}

private void initializeUI() {

txtAssetId = new JTextField();

txtAssetName = new JTextField();

txtAssetType = new JTextField();

txtPurchaseDate = new JTextField();

txtPurchaseCost = new JTextField();

tblAssets = new JTable();

tblAssets.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblAssets.getSelectionModel().addListSelectionListener(e -> selectAsset());

JScrollPane scrollPane = new JScrollPane(tblAssets);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("Asset ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Asset Name:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Asset Type:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Purchase Date:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Purchase Cost:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtAssetId, gbc);

gbc.gridy++;

panel.add(txtAssetName, gbc);

gbc.gridy++;

panel.add(txtAssetType, gbc);

gbc.gridy++;

panel.add(txtPurchaseDate, gbc);

gbc.gridy++;

panel.add(txtPurchaseCost, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertAsset());

btnModify.addActionListener(e -> modifyAsset());

btnDelete.addActionListener(e -> deleteAsset());

btnDisplay.addActionListener(e -> displayAssets());

setTitle("Asset Management");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "akshay";

String password = "akshay2003";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertAsset() {

int assetId = Integer.parseInt(txtAssetId.getText());

String assetName = txtAssetName.getText();

String assetType = txtAssetType.getText();

String purchaseDate = txtPurchaseDate.getText();

float purchaseCost = Float.parseFloat(txtPurchaseCost.getText());

try {

String query = "INSERT INTO asset (asset\_id, asset\_name, asset\_type, purchase\_date, purchase\_cost) VALUES (?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, assetId);

statement.setString(2, assetName);

statement.setString(3, assetType);

statement.setString(4, purchaseDate);

statement.setFloat(5, purchaseCost);

statement.executeUpdate();

clearFields();

displayAssets();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyAsset() {

int selectedRow = tblAssets.getSelectedRow();

if (selectedRow >= 0) {

int assetId = Integer.parseInt(txtAssetId.getText());

String assetName = txtAssetName.getText();

String assetType = txtAssetType.getText();

String purchaseDate = txtPurchaseDate.getText();

float purchaseCost = Float.parseFloat(txtPurchaseCost.getText());

try {

String query = "UPDATE asset SET asset\_name=?, asset\_type=?, purchase\_date=?, purchase\_cost=? WHERE asset\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, assetName);

statement.setString(2, assetType);

statement.setString(3, purchaseDate);

statement.setFloat(4, purchaseCost);

statement.setInt(5, assetId);

statement.executeUpdate();

clearFields();

displayAssets();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select an asset to modify.");

}

}

private void deleteAsset() {

int selectedRow = tblAssets.getSelectedRow();

if (selectedRow >= 0) {

int assetId = Integer.parseInt(tblAssets.getValueAt(selectedRow, 0).toString());

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this asset?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM asset WHERE asset\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, assetId);

statement.executeUpdate();

clearFields();

displayAssets();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select an asset to delete.");

}

}

private void displayAssets() {

try {

String query = "SELECT \* FROM asset";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Asset> assets = new ArrayList<>();

while (resultSet.next()) {

int assetId = resultSet.getInt("asset\_id");

String assetName = resultSet.getString("asset\_name");

String assetType = resultSet.getString("asset\_type");

String purchaseDate = resultSet.getString("purchase\_date");

float purchaseCost = resultSet.getFloat("purchase\_cost");

assets.add(new Asset(assetId, assetName, assetType, purchaseDate, purchaseCost));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Asset ID", "Asset Name", "Asset Type", "Purchase Date", "Purchase Cost"});

for (Asset asset : assets) {

model.addRow(new Object[]{asset.getAssetId(), asset.getAssetName(), asset.getAssetType(), asset.getPurchaseDate(), asset.getPurchaseCost()});

}

tblAssets.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectAsset() {

int selectedRow = tblAssets.getSelectedRow();

if (selectedRow >= 0) {

int assetId = Integer.parseInt(tblAssets.getValueAt(selectedRow, 0).toString());

String assetName = tblAssets.getValueAt(selectedRow, 1).toString();

String assetType = tblAssets.getValueAt(selectedRow, 2).toString();

String purchaseDate = tblAssets.getValueAt(selectedRow, 3).toString();

float purchaseCost = Float.parseFloat(tblAssets.getValueAt(selectedRow, 4).toString());

txtAssetId.setText(Integer.toString(assetId));

txtAssetName.setText(assetName);

txtAssetType.setText(assetType);

txtPurchaseDate.setText(purchaseDate);

txtPurchaseCost.setText(Float.toString(purchaseCost));

}

}

private void clearFields() {

txtAssetId.setText("");

txtAssetName.setText("");

txtAssetType.setText("");

txtPurchaseDate.setText("");

txtPurchaseCost.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(AssetTableGUI::new);

}

private class Asset {

private int assetId;

private String assetName;

private String assetType;

private String purchaseDate;

private float purchaseCost;

public Asset(int assetId, String assetName, String assetType, String purchaseDate, float purchaseCost) {

this.assetId = assetId;

this.assetName = assetName;

this.assetType = assetType;

this.purchaseDate = purchaseDate;

this.purchaseCost = purchaseCost;

}

public int getAssetId() {

return assetId;

}

public String getAssetName() {

return assetName;

}

public String getAssetType() {

return assetType;

}

public String getPurchaseDate() {

return purchaseDate;

}

public float getPurchaseCost() {

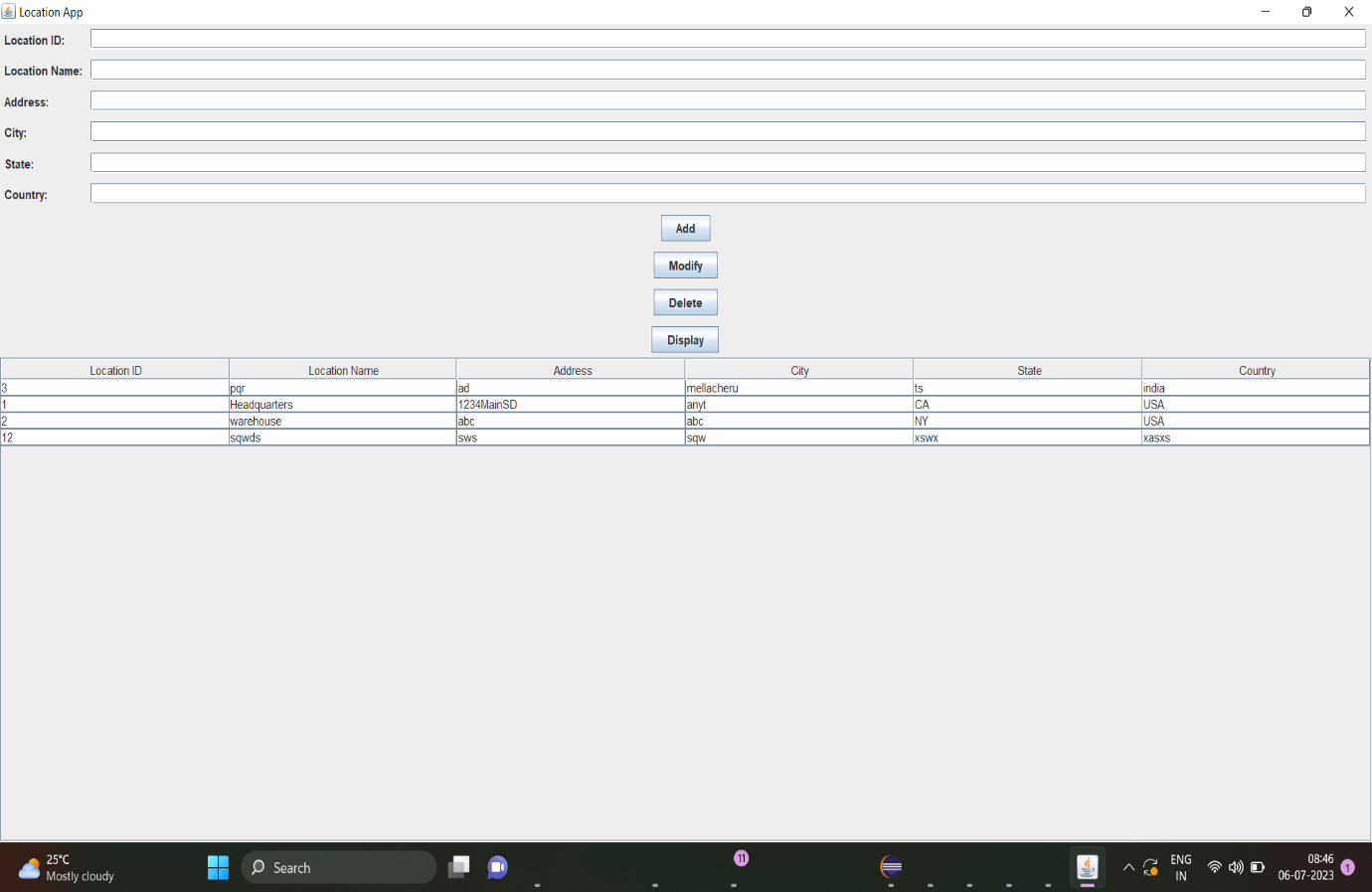
return purchaseCost;

}

}

}

**Location Table:**



import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class LocationTableGUI extends JFrame {

private JTextField txtLocationId, txtLocationName, txtAddress, txtCity, txtState, txtCountry;

private JTable tblLocations;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public LocationTableGUI() {

initializeUI();

connectToDatabase();

displayLocations();

}

private void initializeUI() {

txtLocationId = new JTextField();

txtLocationName = new JTextField();

txtAddress = new JTextField();

txtCity = new JTextField();

txtState = new JTextField();

txtCountry = new JTextField();

tblLocations = new JTable();

tblLocations.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblLocations.getSelectionModel().addListSelectionListener(e -> selectLocation());

JScrollPane scrollPane = new JScrollPane(tblLocations);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("Location ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Location Name:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Address:"), gbc);

gbc.gridy++;

panel.add(new JLabel("City:"), gbc);

gbc.gridy++;

panel.add(new JLabel("State:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Country:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtLocationId, gbc);

gbc.gridy++;

panel.add(txtLocationName, gbc);

gbc.gridy++;

panel.add(txtAddress, gbc);

gbc.gridy++;

panel.add(txtCity, gbc);

gbc.gridy++;

panel.add(txtState, gbc);

gbc.gridy++;

panel.add(txtCountry, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertLocation());

btnModify.addActionListener(e -> modifyLocation());

btnDelete.addActionListener(e -> deleteLocation());

btnDisplay.addActionListener(e -> displayLocations());

setTitle("Location App");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "akshay";

String password = "akshay2003";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertLocation() {

String locationId = txtLocationId.getText();

String locationName = txtLocationName.getText();

String address = txtAddress.getText();

String city = txtCity.getText();

String state = txtState.getText();

String country = txtCountry.getText();

try {

String query = "INSERT INTO location (location\_id, location\_name, address, city, state, country) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, locationId);

statement.setString(2, locationName);

statement.setString(3, address);

statement.setString(4, city);

statement.setString(5, state);

statement.setString(6, country);

statement.executeUpdate();

clearFields();

displayLocations();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyLocation() {

int selectedRow = tblLocations.getSelectedRow();

if (selectedRow >= 0) {

String locationId = txtLocationId.getText();

String locationName = txtLocationName.getText();

String address = txtAddress.getText();

String city = txtCity.getText();

String state = txtState.getText();

String country = txtCountry.getText();

try {

String query = "UPDATE location SET location\_name=?, address=?, city=?, state=?, country=? WHERE location\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, locationName);

statement.setString(2, address);

statement.setString(3, city);

statement.setString(4, state);

statement.setString(5, country);

statement.setString(6, locationId);

statement.executeUpdate();

clearFields();

displayLocations();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select a location to modify.");

}

}

private void deleteLocation() {

int selectedRow = tblLocations.getSelectedRow();

if (selectedRow >= 0) {

String locationId = tblLocations.getValueAt(selectedRow, 0).toString();

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this location?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM location WHERE location\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, locationId);

statement.executeUpdate();

clearFields();

displayLocations();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select a location to delete.");

}

}

private void displayLocations() {

try {

String query = "SELECT \* FROM location";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Location> locations = new ArrayList<>();

while (resultSet.next()) {

String locationId = resultSet.getString("location\_id");

String locationName = resultSet.getString("location\_name");

String address = resultSet.getString("address");

String city = resultSet.getString("city");

String state = resultSet.getString("state");

String country = resultSet.getString("country");

locations.add(new Location(locationId, locationName, address, city, state, country));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Location ID", "Location Name", "Address", "City", "State", "Country"});

for (Location location : locations) {

model.addRow(new String[]{location.getLocationId(), location.getLocationName(), location.getAddress(), location.getCity(), location.getState(), location.getCountry()});

}

tblLocations.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectLocation() {

int selectedRow = tblLocations.getSelectedRow();

if (selectedRow >= 0) {

String locationId = tblLocations.getValueAt(selectedRow, 0).toString();

String locationName = tblLocations.getValueAt(selectedRow, 1).toString();

String address = tblLocations.getValueAt(selectedRow, 2).toString();

String city = tblLocations.getValueAt(selectedRow, 3).toString();

String state = tblLocations.getValueAt(selectedRow, 4).toString();

String country = tblLocations.getValueAt(selectedRow, 5).toString();

txtLocationId.setText(locationId);

txtLocationName.setText(locationName);

txtAddress.setText(address);

txtCity.setText(city);

txtState.setText(state);

txtCountry.setText(country);

}

}

private void clearFields() {

txtLocationId.setText("");

txtLocationName.setText("");

txtAddress.setText("");

txtCity.setText("");

txtState.setText("");

txtCountry.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(LocationTableGUI::new);

}

private class Location {

private String locationId;

private String locationName;

private String address;

private String city;

private String state;

private String country;

public Location(String locationId, String locationName, String address, String city, String state, String country) {

this.locationId = locationId;

this.locationName = locationName;

this.address = address;

this.city = city;

this.state = state;

this.country = country;

}

public String getLocationId() {

return locationId;

}

public String getLocationName() {

return locationName;

}

public String getAddress() {

return address;

}

public String getCity() {

return city;

}

public String getState() {

return state;

}

public String getCountry() {

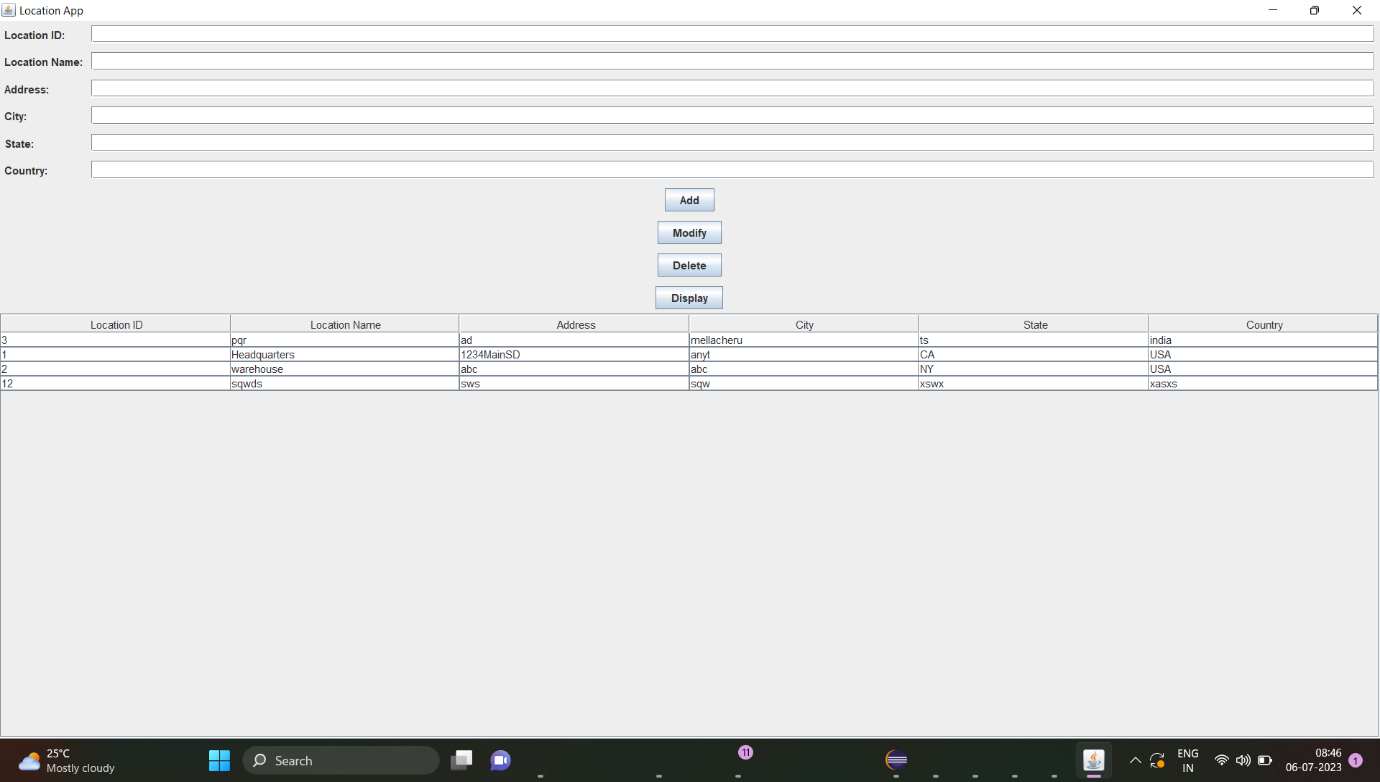
return country;

}

}

}

**Assignment Table:**



import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class AssignmentTableGUI extends JFrame {

private JTextField txtAssignmentId, txtAssetId, txtUserId, txtAssignedDate, txtReturnDate, txtLocationId;

private JTable tblAssignments;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public AssignmentTableGUI() {

initializeUI();

connectToDatabase();

displayAssignments();

}

private void initializeUI() {

txtAssignmentId = new JTextField();

txtAssetId = new JTextField();

txtUserId = new JTextField();

txtAssignedDate = new JTextField();

txtReturnDate = new JTextField();

txtLocationId = new JTextField();

tblAssignments = new JTable();

tblAssignments.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblAssignments.getSelectionModel().addListSelectionListener(e -> selectAssignment());

JScrollPane scrollPane = new JScrollPane(tblAssignments);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("Assignment ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Asset ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("User ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Assigned Date:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Return Date:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Location ID:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtAssignmentId, gbc);

gbc.gridy++;

panel.add(txtAssetId, gbc);

gbc.gridy++;

panel.add(txtUserId, gbc);

gbc.gridy++;

panel.add(txtAssignedDate, gbc);

gbc.gridy++;

panel.add(txtReturnDate, gbc);

gbc.gridy++;

panel.add(txtLocationId, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertAssignment());

btnModify.addActionListener(e -> modifyAssignment());

btnDelete.addActionListener(e -> deleteAssignment());

btnDisplay.addActionListener(e -> displayAssignments());

setTitle("Assignments App");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "akshay";

String password = "akshay2003";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertAssignment() {

String assignmentId = txtAssignmentId.getText();

String assetId = txtAssetId.getText();

String userId = txtUserId.getText();

String assignedDate = txtAssignedDate.getText();

String returnDate = txtReturnDate.getText();

String locationId = txtLocationId.getText();

try {

String query = "INSERT INTO assignment (assignment\_id, asset\_id, user\_id, assigned\_date, return\_date, location\_id) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, assignmentId);

statement.setString(2, assetId);

statement.setString(3, userId);

statement.setString(4, assignedDate);

statement.setString(5, returnDate);

statement.setString(6, locationId);

statement.executeUpdate();

clearFields();

displayAssignments();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyAssignment() {

int selectedRow = tblAssignments.getSelectedRow();

if (selectedRow >= 0) {

String assignmentId = txtAssignmentId.getText();

String assetId = txtAssetId.getText();

String userId = txtUserId.getText();

String assignedDate = txtAssignedDate.getText();

String returnDate = txtReturnDate.getText();

String locationId = txtLocationId.getText();

try {

String query = "UPDATE assignment SET asset\_id=?, user\_id=?, assigned\_date=?, return\_date=?, location\_id=? WHERE assignment\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, assetId);

statement.setString(2, userId);

statement.setString(3, assignedDate);

statement.setString(4, returnDate);

statement.setString(5, locationId);

statement.setString(6, assignmentId);

statement.executeUpdate();

clearFields();

displayAssignments();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select an assignment to modify.");

}

}

private void deleteAssignment() {

int selectedRow = tblAssignments.getSelectedRow();

if (selectedRow >= 0) {

String assignmentId = tblAssignments.getValueAt(selectedRow, 0).toString();

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this assignment?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM assignment WHERE assignment\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, assignmentId);

statement.executeUpdate();

clearFields();

displayAssignments();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select an assignment to delete.");

}

}

private void displayAssignments() {

try {

String query = "SELECT \* FROM assignment";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Assignment> assignments = new ArrayList<>();

while (resultSet.next()) {

String assignmentId = resultSet.getString("assignment\_id");

String assetId = resultSet.getString("asset\_id");

String userId = resultSet.getString("user\_id");

String assignedDate = resultSet.getString("assigned\_date");

String returnDate = resultSet.getString("return\_date");

String locationId = resultSet.getString("location\_id");

assignments.add(new Assignment(assignmentId, assetId, userId, assignedDate, returnDate, locationId));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Assignment ID", "Asset ID", "User ID", "Assigned Date", "Return Date", "Location ID"});

for (Assignment assignment : assignments) {

model.addRow(new String[]{assignment.getAssignmentId(), assignment.getAssetId(), assignment.getUserId(), assignment.getAssignedDate(), assignment.getReturnDate(), assignment.getLocationId()});

}

tblAssignments.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectAssignment() {

int selectedRow = tblAssignments.getSelectedRow();

if (selectedRow >= 0) {

String assignmentId = tblAssignments.getValueAt(selectedRow, 0).toString();

String assetId = tblAssignments.getValueAt(selectedRow, 1).toString();

String userId = tblAssignments.getValueAt(selectedRow, 2).toString();

String assignedDate = tblAssignments.getValueAt(selectedRow, 3).toString();

String returnDate = tblAssignments.getValueAt(selectedRow, 4).toString();

String locationId = tblAssignments.getValueAt(selectedRow, 5).toString();

txtAssignmentId.setText(assignmentId);

txtAssetId.setText(assetId);

txtUserId.setText(userId);

txtAssignedDate.setText(assignedDate);

txtReturnDate.setText(returnDate);

txtLocationId.setText(locationId);

}

}

private void clearFields() {

txtAssignmentId.setText("");

txtAssetId.setText("");

txtUserId.setText("");

txtAssignedDate.setText("");

txtReturnDate.setText("");

txtLocationId.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(AssignmentTableGUI::new);

}

private class Assignment {

private String assignmentId;

private String assetId;

private String userId;

private String assignedDate;

private String returnDate;

private String locationId;

public Assignment(String assignmentId, String assetId, String userId, String assignedDate, String returnDate, String locationId) {

this.assignmentId = assignmentId;

this.assetId = assetId;

this.userId = userId;

this.assignedDate = assignedDate;

this.returnDate = returnDate;

this.locationId = locationId;

}

public String getAssignmentId() {

return assignmentId;

}

public String getAssetId() {

return assetId;

}

public String getUserId() {

return userId;

}

public String getAssignedDate() {

return assignedDate;

}

public String getReturnDate() {

return returnDate;

}

public String getLocationId() {

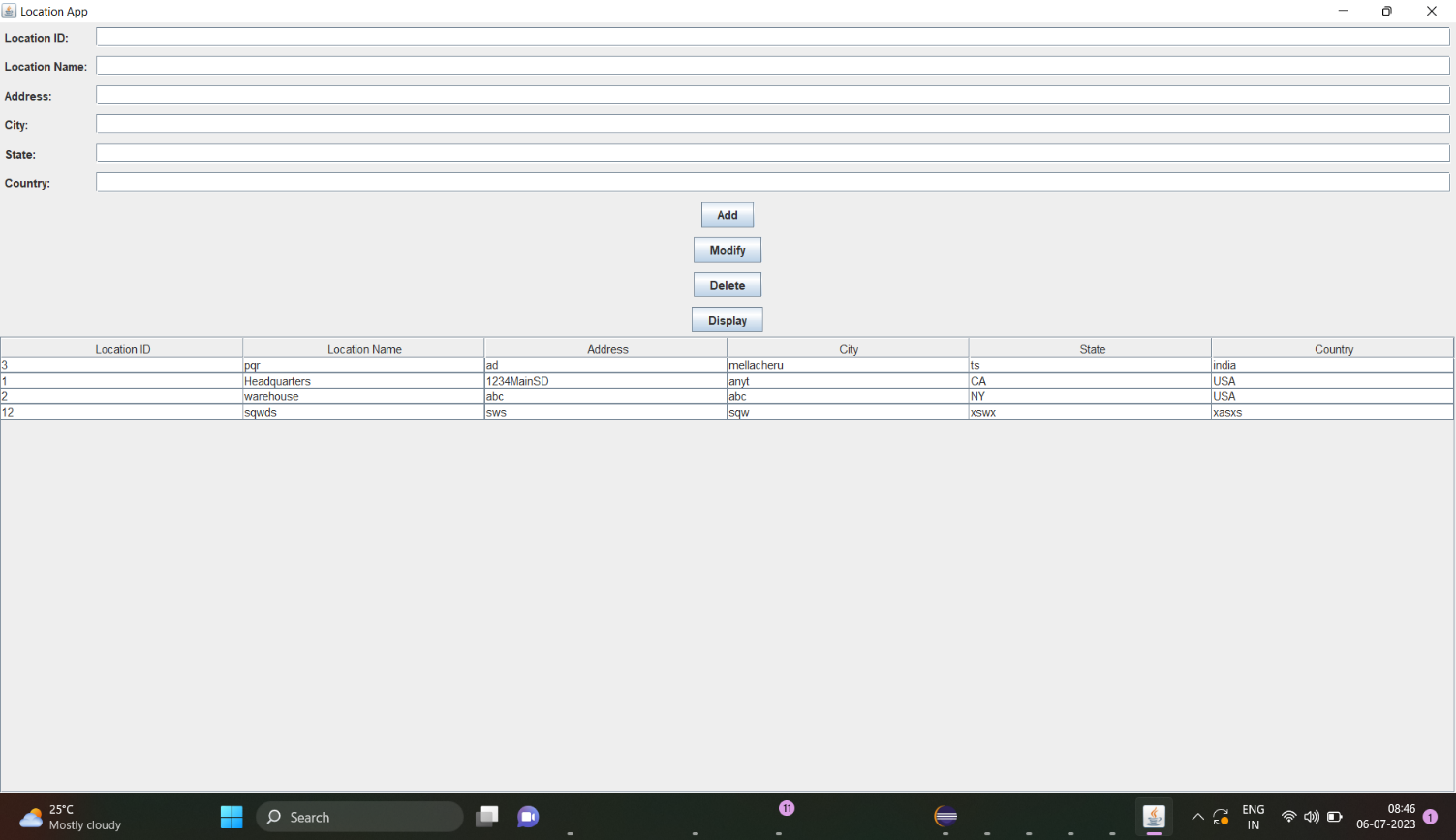
return locationId;

}

}

}

**Maintenance Table:**



import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class MaintenanceTableGUI extends JFrame {

private JTextField txtMaintenanceId, txtAssetId, txtMaintenanceDate, txtMaintenanceDescription;

private JTable tblMaintenance;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public MaintenanceTableGUI() {

initializeUI();

connectToDatabase();

displayMaintenance();

}

private void initializeUI() {

txtMaintenanceId = new JTextField();

txtAssetId = new JTextField();

txtMaintenanceDate = new JTextField();

txtMaintenanceDescription = new JTextField();

tblMaintenance = new JTable();

tblMaintenance.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblMaintenance.getSelectionModel().addListSelectionListener(e -> selectMaintenance());

JScrollPane scrollPane = new JScrollPane(tblMaintenance);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("Maintenance ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Asset ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Maintenance Date:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Maintenance Description:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtMaintenanceId, gbc);

gbc.gridy++;

panel.add(txtAssetId, gbc);

gbc.gridy++;

panel.add(txtMaintenanceDate, gbc);

gbc.gridy++;

panel.add(txtMaintenanceDescription, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertMaintenance());

btnModify.addActionListener(e -> modifyMaintenance());

btnDelete.addActionListener(e -> deleteMaintenance());

btnDisplay.addActionListener(e -> displayMaintenance());

setTitle("Maintenance App");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "akshay";

String password = "akshay2003";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertMaintenance() {

String maintenanceId = txtMaintenanceId.getText();

String assetId = txtAssetId.getText();

String maintenanceDate = txtMaintenanceDate.getText();

String maintenanceDescription = txtMaintenanceDescription.getText();

try {

String query = "INSERT INTO Maintenance (maintenance\_id, asset\_id, maintenance\_date, maintenance\_description) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, maintenanceId);

statement.setString(2, assetId);

statement.setString(3, maintenanceDate);

statement.setString(4, maintenanceDescription);

statement.executeUpdate();

clearFields();

displayMaintenance();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyMaintenance() {

int selectedRow = tblMaintenance.getSelectedRow();

if (selectedRow >= 0) {

String maintenanceId = txtMaintenanceId.getText();

String assetId = txtAssetId.getText();

String maintenanceDate = txtMaintenanceDate.getText();

String maintenanceDescription = txtMaintenanceDescription.getText();

try {

String query = "UPDATE Maintenance SET asset\_id=?, maintenance\_date=?, maintenance\_description=? WHERE maintenance\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, assetId);

statement.setString(2, maintenanceDate);

statement.setString(3, maintenanceDescription);

statement.setString(4, maintenanceId);

statement.executeUpdate();

clearFields();

displayMaintenance();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select a maintenance entry to modify.");

}

}

private void deleteMaintenance() {

int selectedRow = tblMaintenance.getSelectedRow();

if (selectedRow >= 0) {

String maintenanceId = tblMaintenance.getValueAt(selectedRow, 0).toString();

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this maintenance entry?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM Maintenance WHERE maintenance\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, maintenanceId);

statement.executeUpdate();

clearFields();

displayMaintenance();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select a maintenance entry to delete.");

}

}

private void displayMaintenance() {

try {

String query = "SELECT \* FROM Maintenance";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<MaintenanceEntry> maintenanceEntries = new ArrayList<>();

while (resultSet.next()) {

String maintenanceId = resultSet.getString("maintenance\_id");

String assetId = resultSet.getString("asset\_id");

String maintenanceDate = resultSet.getString("maintenance\_date");

String maintenanceDescription = resultSet.getString("maintenance\_description");

maintenanceEntries.add(new MaintenanceEntry(maintenanceId, assetId, maintenanceDate, maintenanceDescription));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Maintenance ID", "Asset ID", "Maintenance Date", "Maintenance Description"});

for (MaintenanceEntry maintenanceEntry : maintenanceEntries) {

model.addRow(new String[]{maintenanceEntry.getMaintenanceId(), maintenanceEntry.getAssetId(), maintenanceEntry.getMaintenanceDate(), maintenanceEntry.getMaintenanceDescription()});

}

tblMaintenance.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectMaintenance() {

int selectedRow = tblMaintenance.getSelectedRow();

if (selectedRow >= 0) {

String maintenanceId = tblMaintenance.getValueAt(selectedRow, 0).toString();

String assetId = tblMaintenance.getValueAt(selectedRow, 1).toString();

String maintenanceDate = tblMaintenance.getValueAt(selectedRow, 2).toString();

String maintenanceDescription = tblMaintenance.getValueAt(selectedRow, 3).toString();

txtMaintenanceId.setText(maintenanceId);

txtAssetId.setText(assetId);

txtMaintenanceDate.setText(maintenanceDate);

txtMaintenanceDescription.setText(maintenanceDescription);

}

}

private void clearFields() {

txtMaintenanceId.setText("");

txtAssetId.setText("");

txtMaintenanceDate.setText("");

txtMaintenanceDescription.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(MaintenanceTableGUI::new);

}

private class MaintenanceEntry {

private String maintenanceId;

private String assetId;

private String maintenanceDate;

private String maintenanceDescription;

public MaintenanceEntry(String maintenanceId, String assetId, String maintenanceDate, String maintenanceDescription) {

this.maintenanceId = maintenanceId;

this.assetId = assetId;

this.maintenanceDate = maintenanceDate;

this.maintenanceDescription = maintenanceDescription;

}

public String getMaintenanceId() {

return maintenanceId;

}

public String getAssetId() {

return assetId;

}

public String getMaintenanceDate() {

return maintenanceDate;

}

public String getMaintenanceDescription() {

return maintenanceDescription;

}

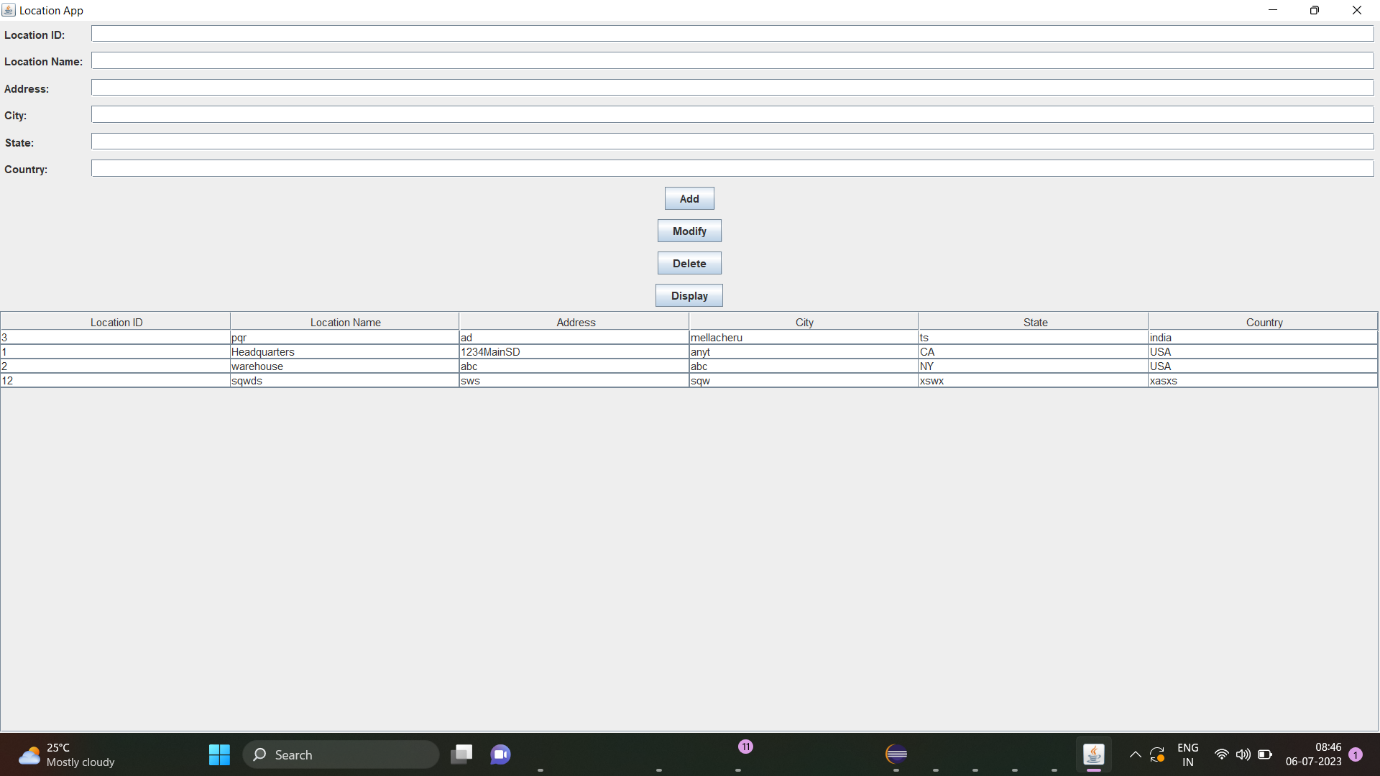
}

}

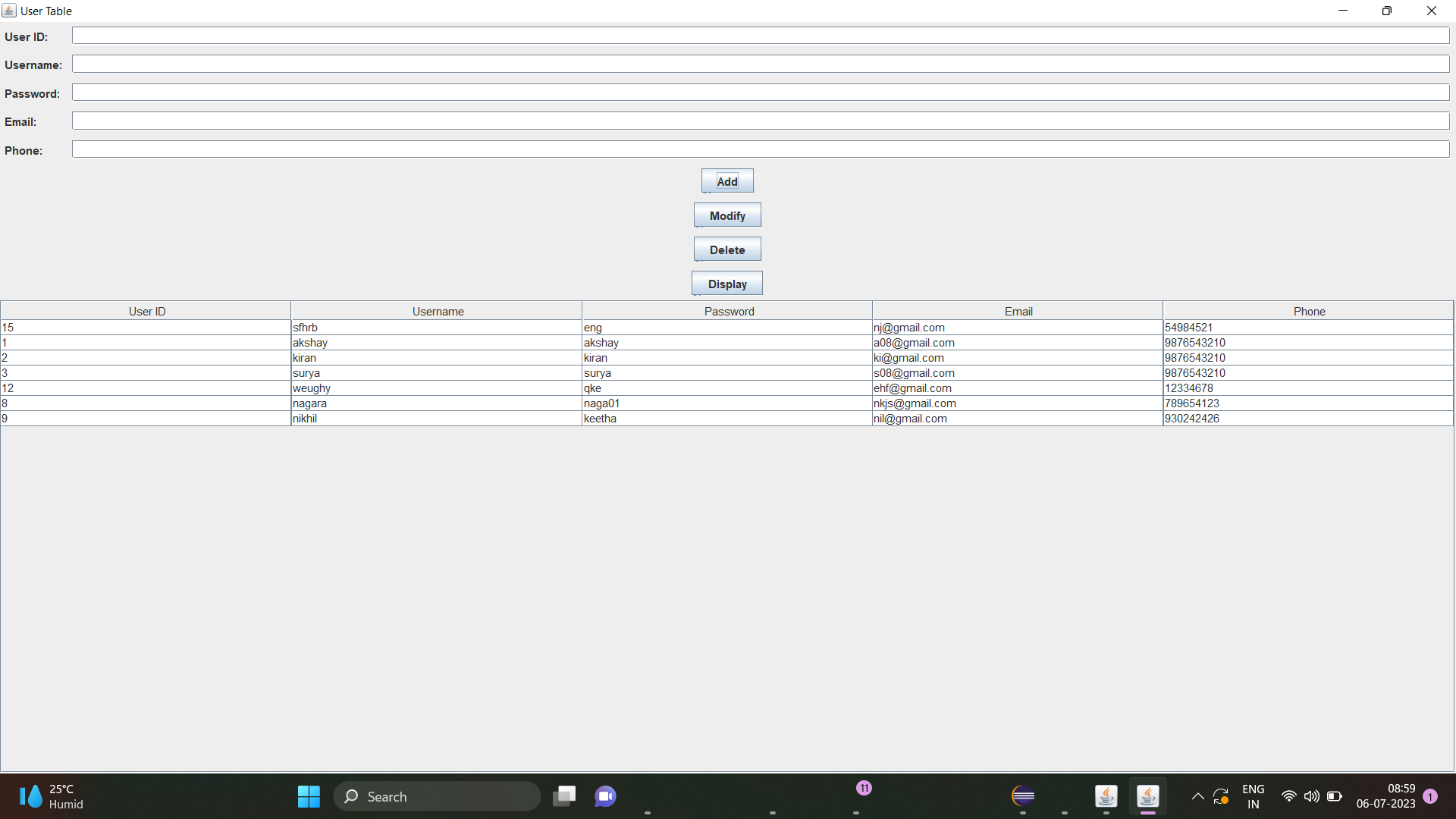
**TESTING**

User page:

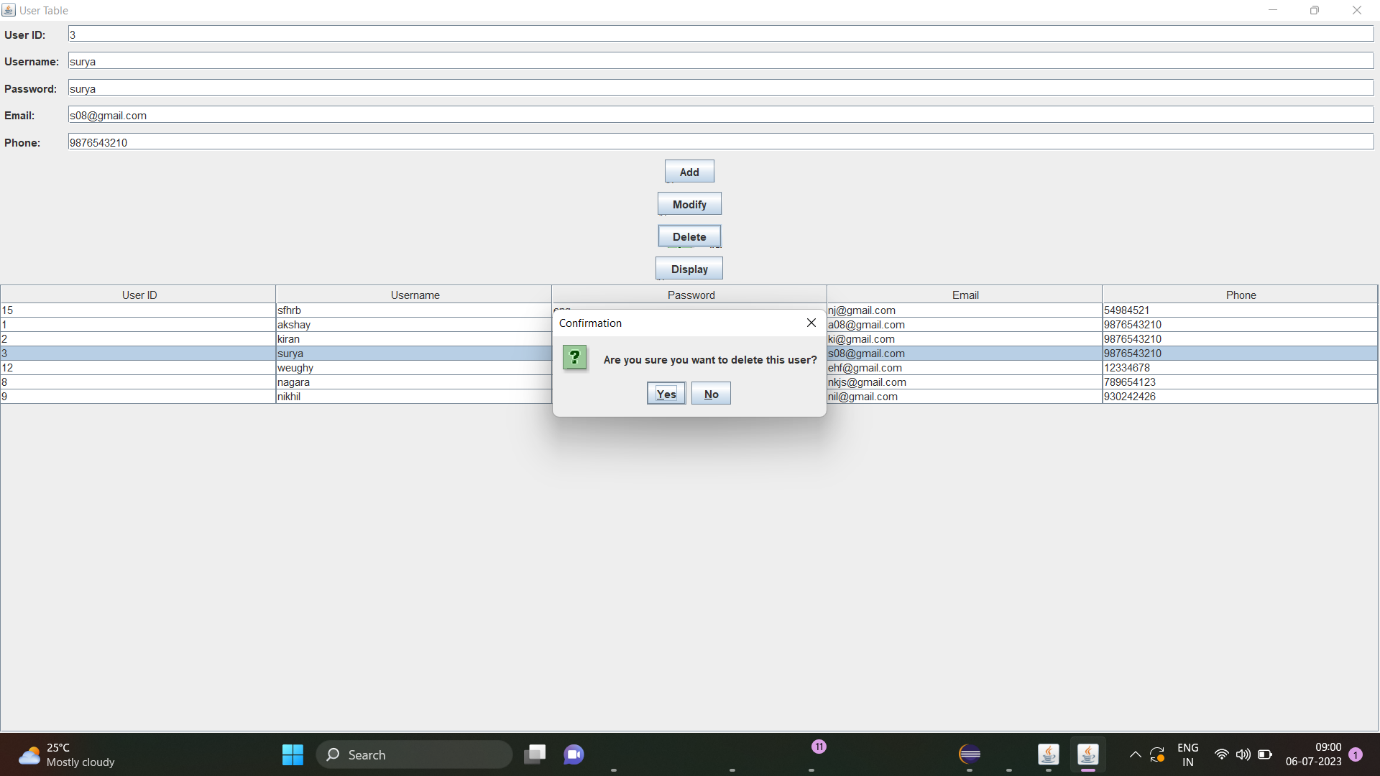
Before insertion into table :



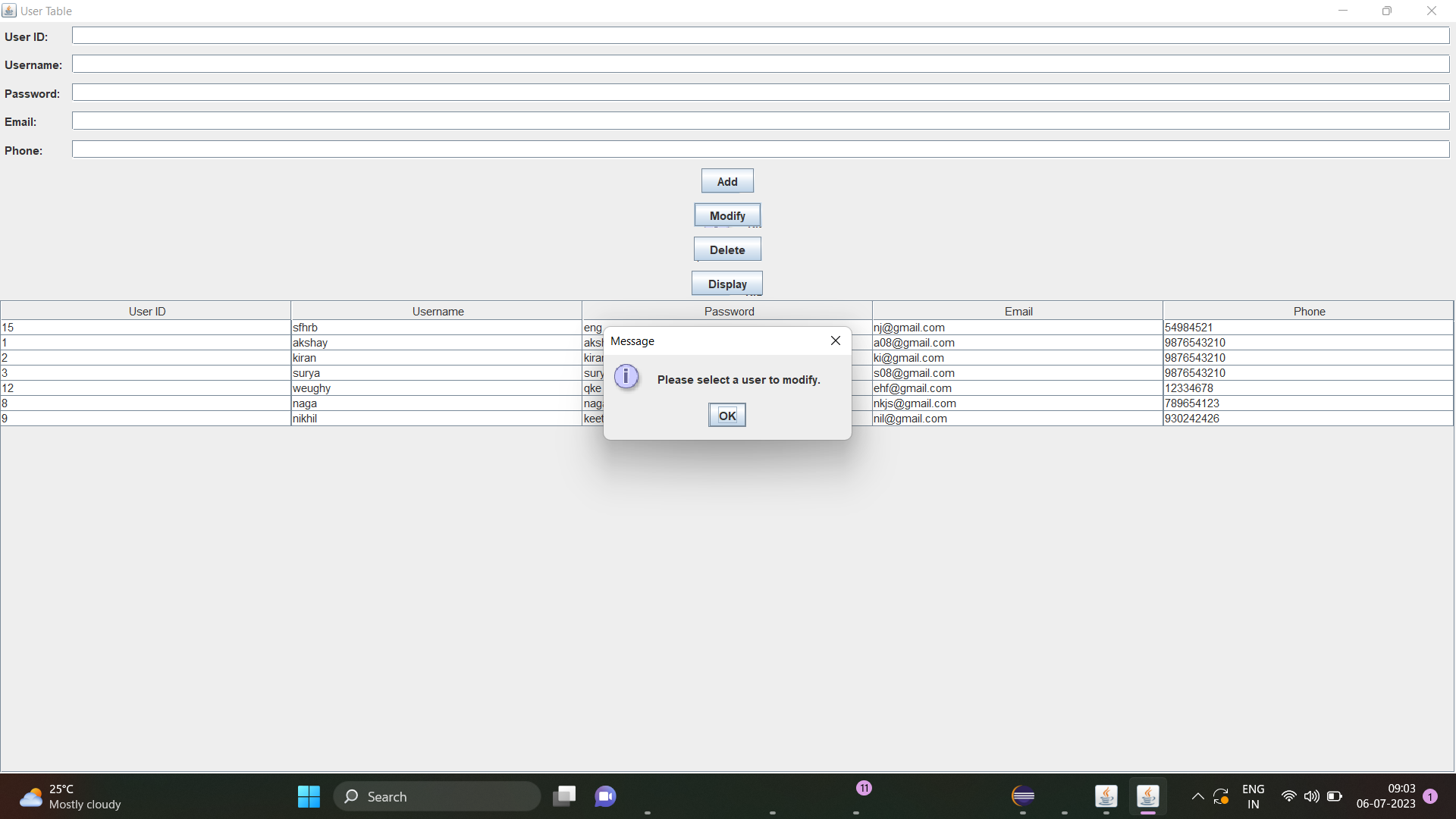
After insertion into table:



Deletion:

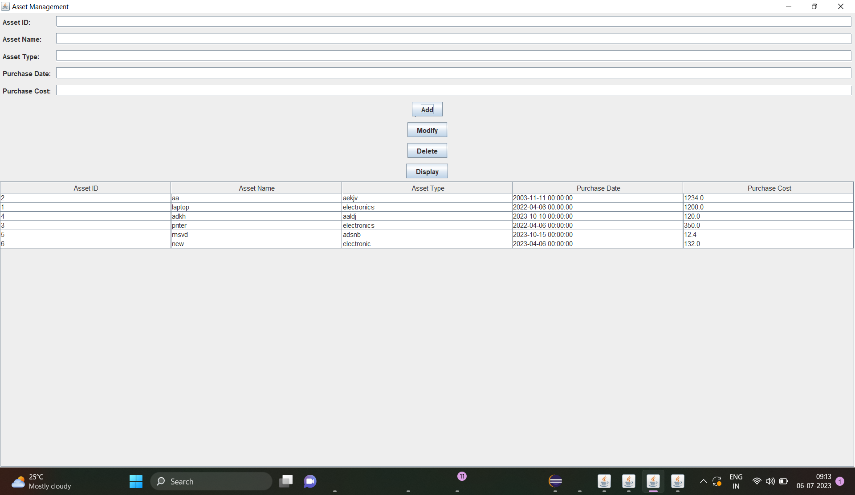


Modification:

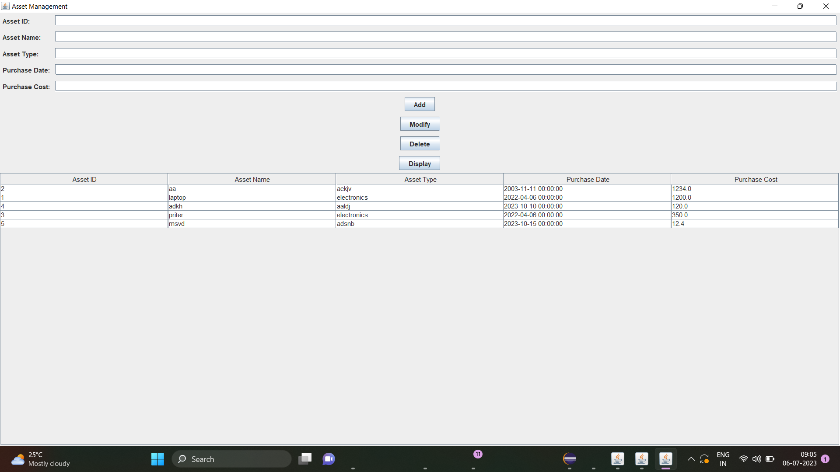


**Asset Management Page:**

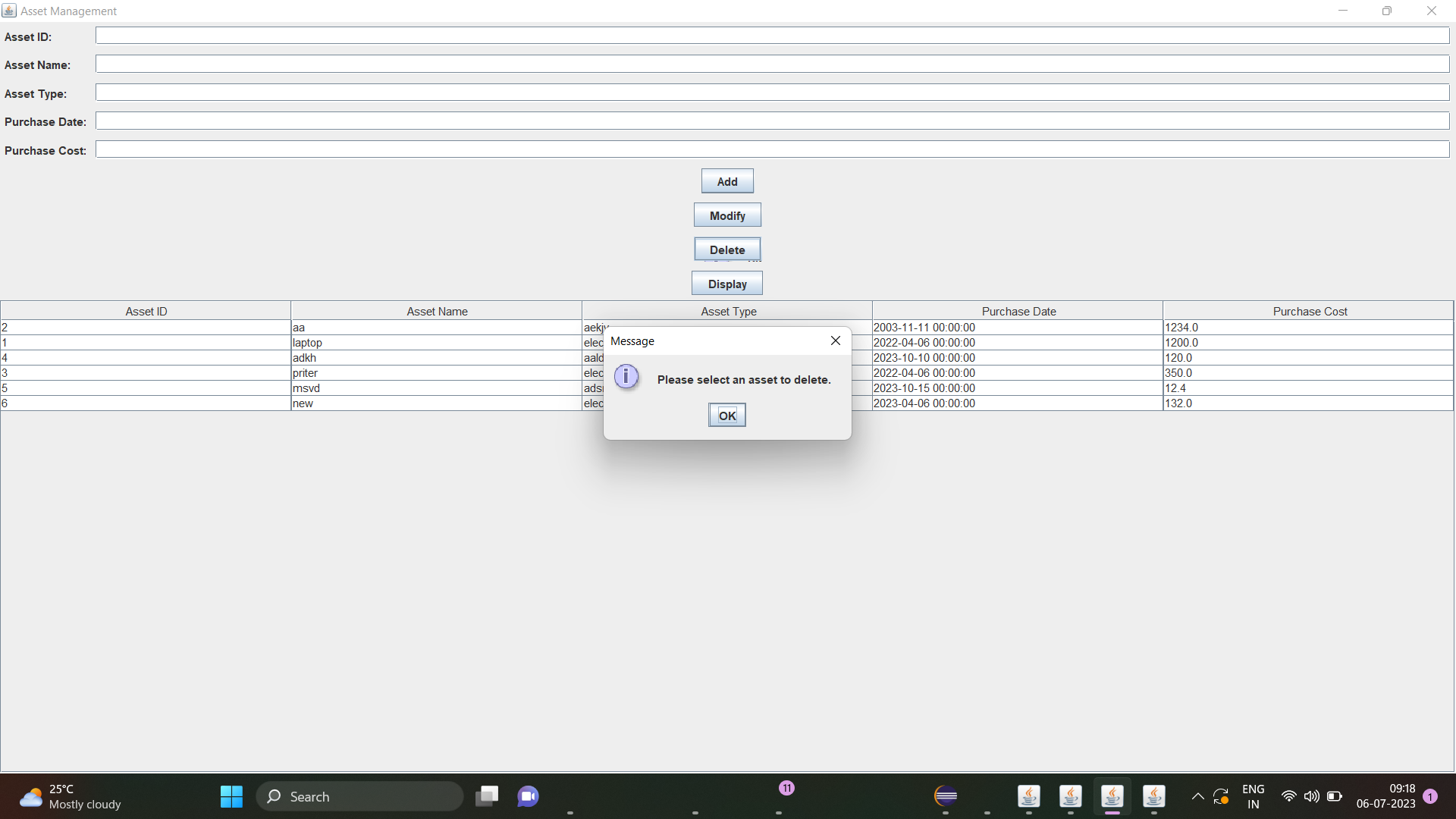
Before insertion:



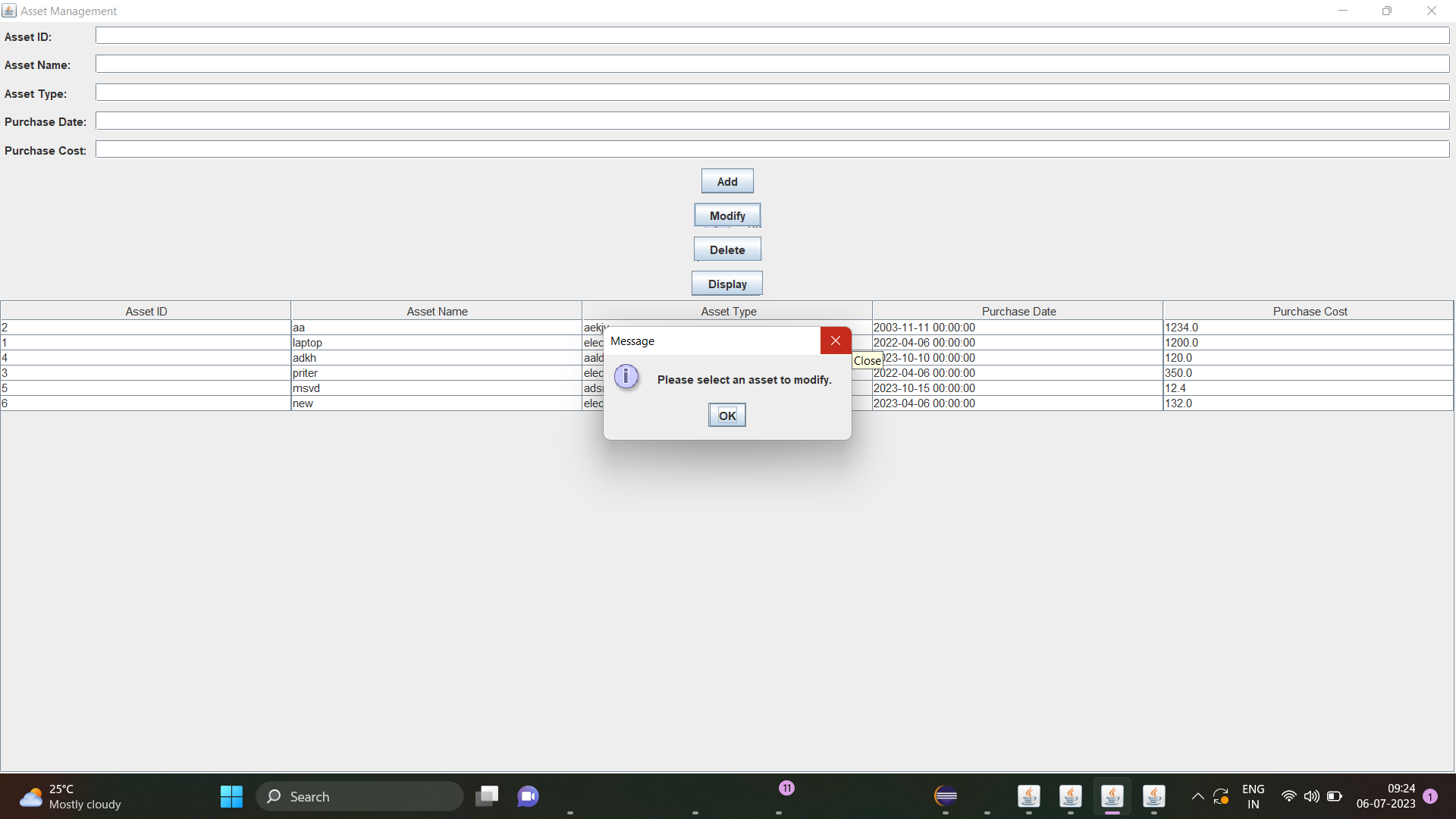
After Insertion:



Deletion:

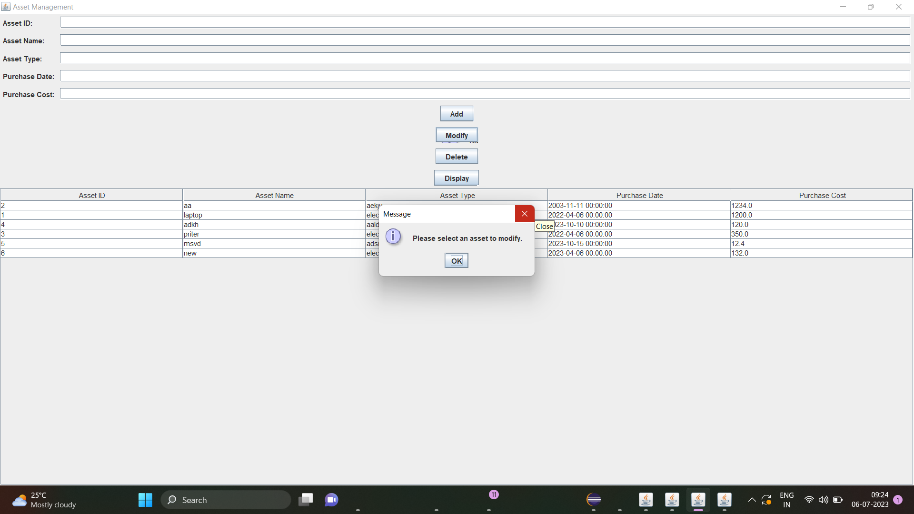


Modification:

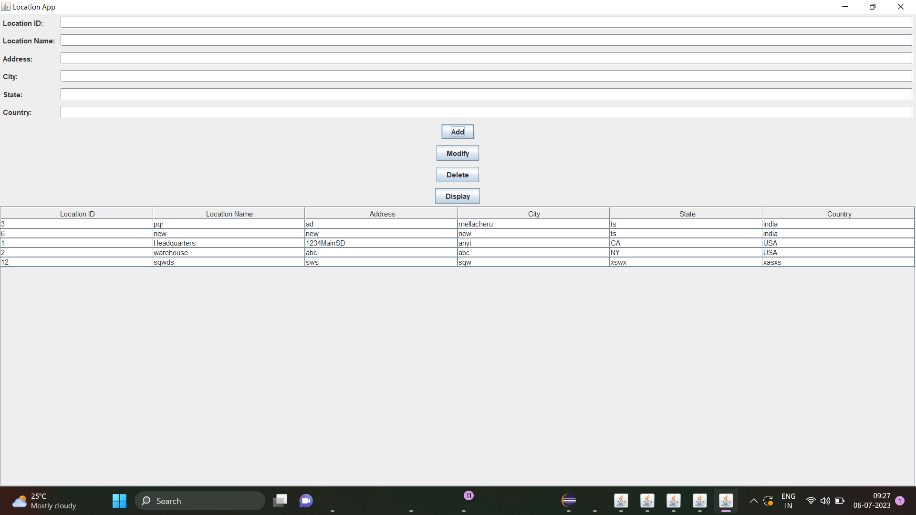


**Location Page:**

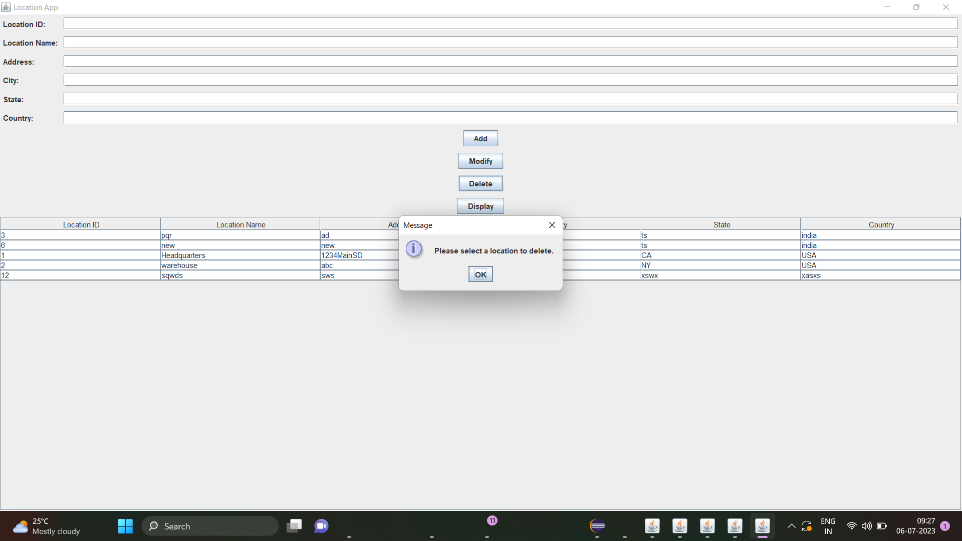
Before insertion:



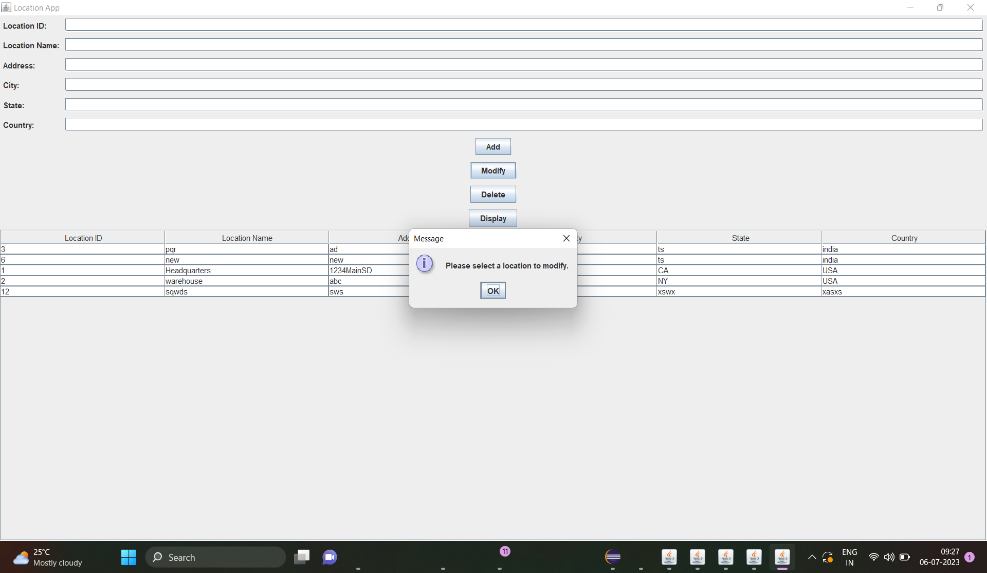
After insertion:



Deletion:

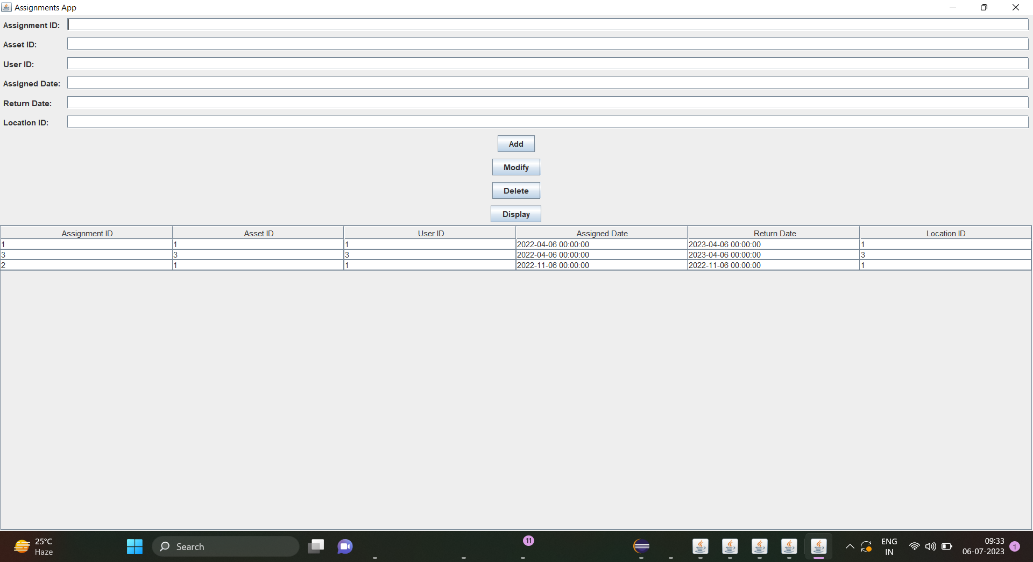


Modification:

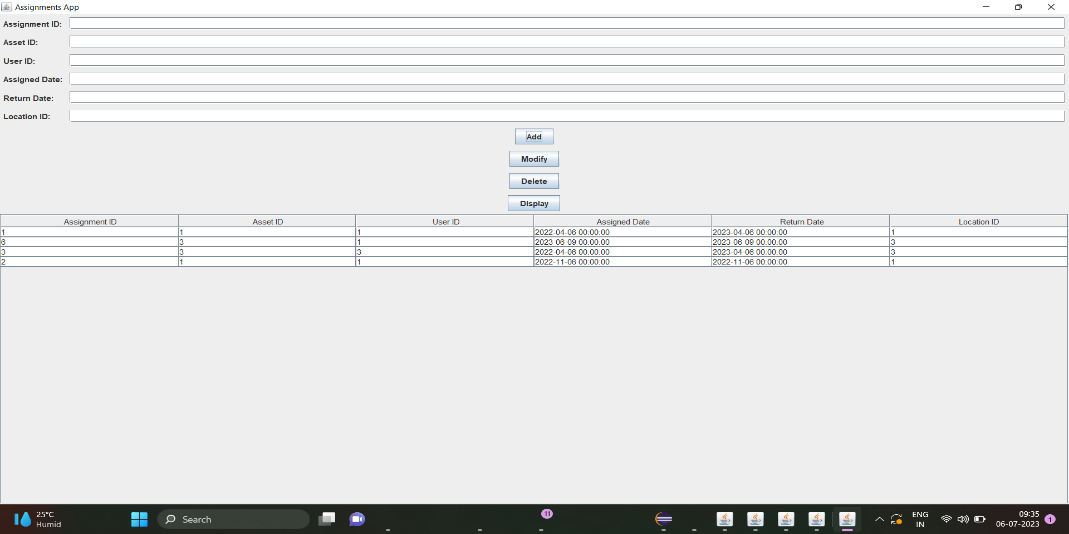


**Assignment page:**

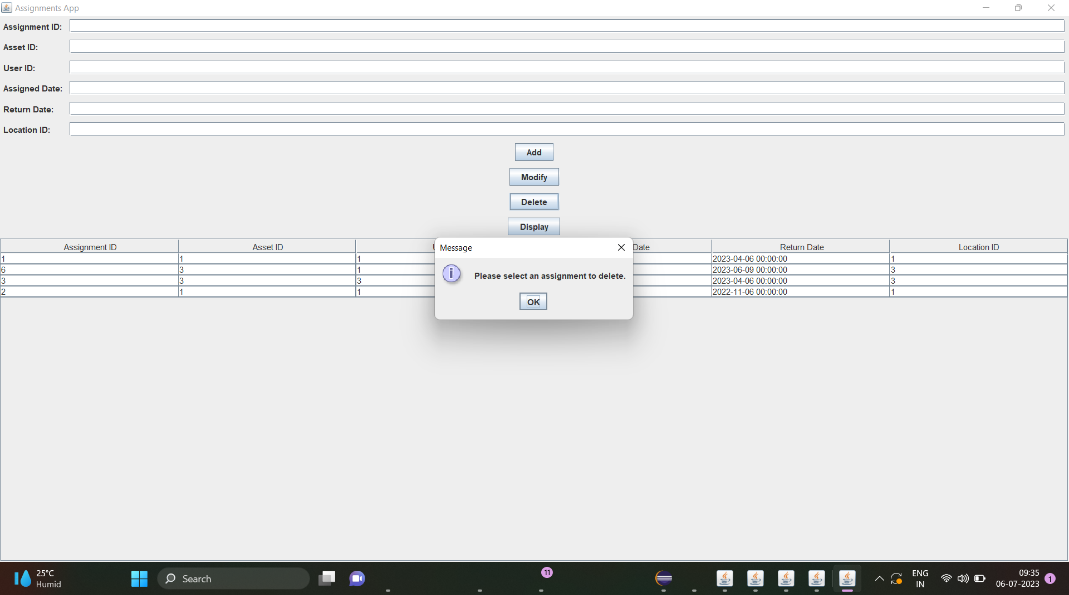
Before insertion:



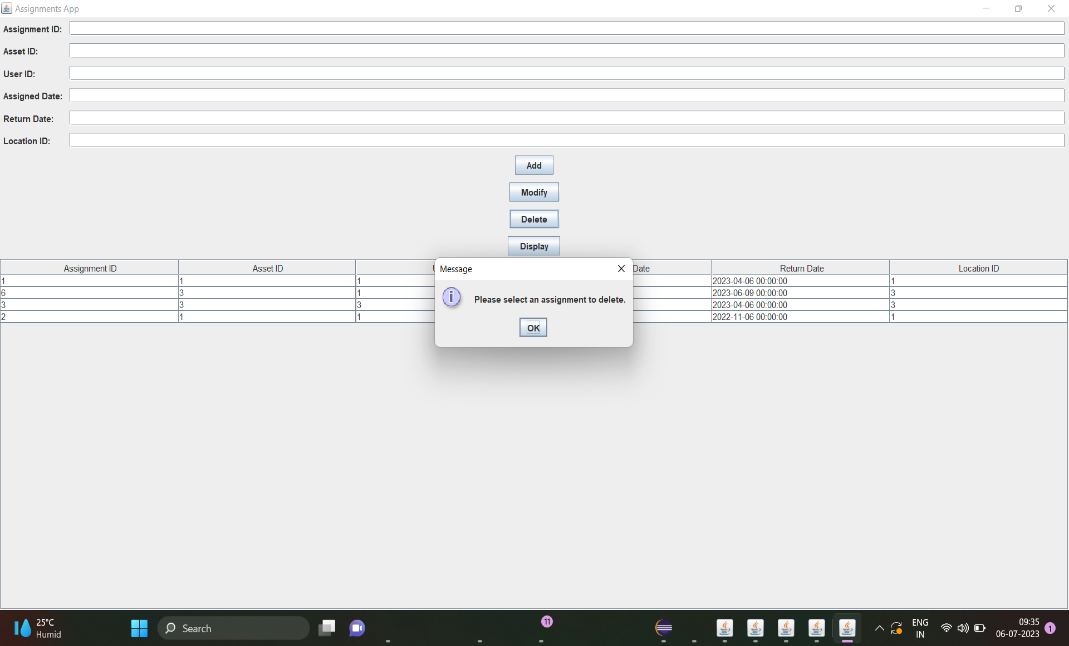
After insertion:



Deletion:

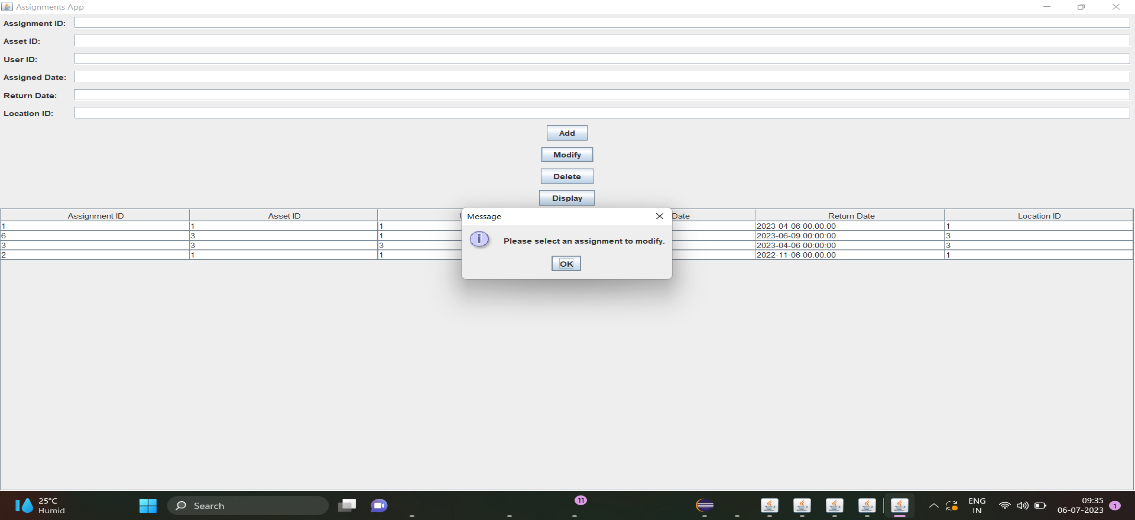


Modification:

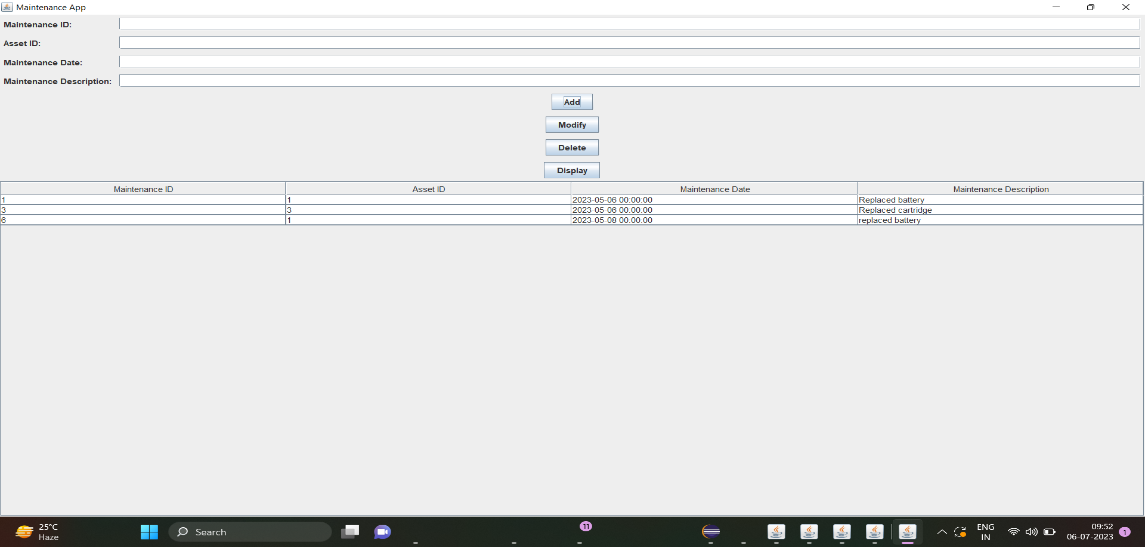


**Maintenance page:**

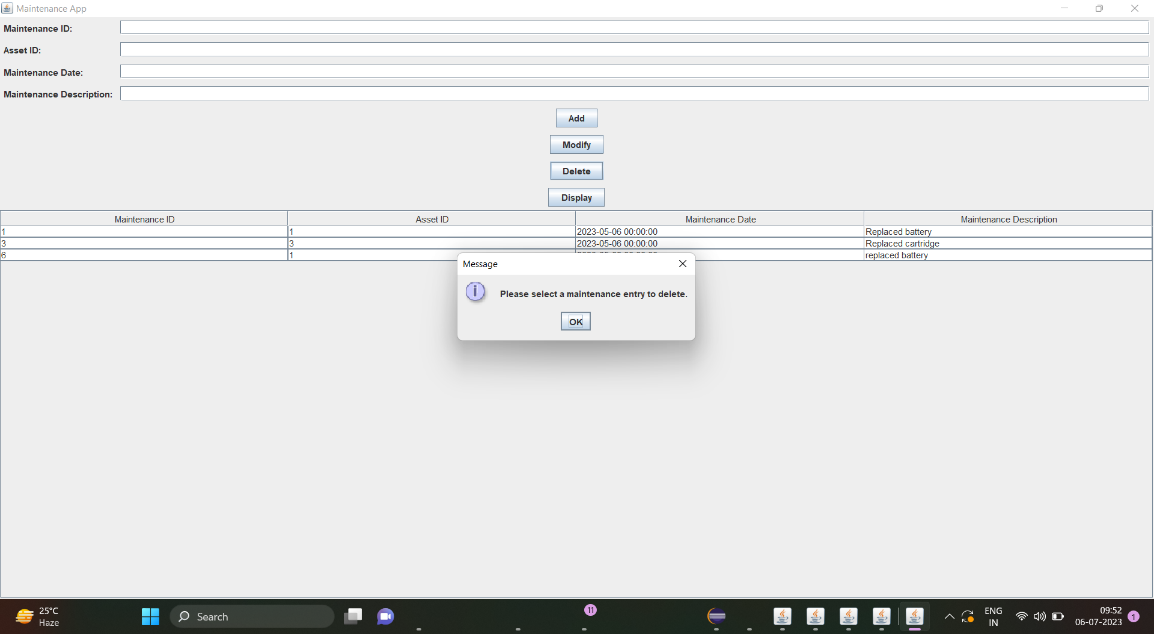
Before insertion:



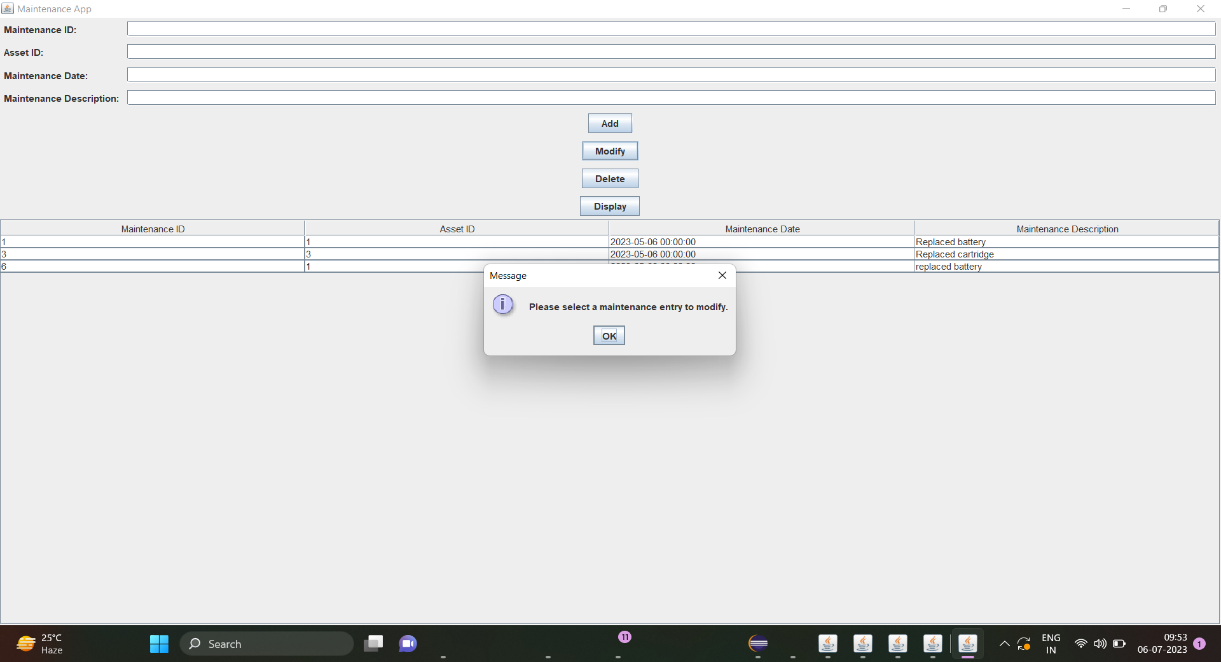
After Insertion:



Deletion:



Modification:



RESULT

I have successfully completed my DBMS project ‘***Asset Tracker***’.

**DISCUSSION AND FUTURE WORK**

My project on asset tracking has been successful in developing a robust system for monitoring and managing assets within an organization. Through our discussions, we have identified several areas for future work and improvement. Firstly, we plan to enhance the tracking capabilities by incorporating advanced technologies such as RFID and GPS for real-time asset location and movement monitoring. Additionally, we aim to integrate the asset tracker with existing enterprise resource planning (ERP) systems to streamline asset data management and enable seamless integration with other business processes. Furthermore, we will explore the potential of incorporating machine learning algorithms to analyze asset usage patterns, predict maintenance needs, and optimize asset allocation. Lastly, we plan to collaborate with stakeholders and gather feedback to continuously improve the user interface and user experience, ensuring that the asset tracking system remains user-friendly and intuitive. Through these future efforts, we aim to provide organizations with an even more powerful and comprehensive asset tracking solution.

**References**

. <https://docs.oracle.com/javase/7/docs/api/>

. <https://www.javatpoint.com/java-swing>

. <https://stackoverflow.com/>