AUTOMATED RENT INVOICE GENERATOR

**Abstract**

This project involves the design and implementation of a rental cab management system that charges rent based on the monthly usage of the vehicle. The system will be designed using the entity-relationship (ER) model and implemented using SQL. The system will include tables for users, vehicles, rentals, payments, and locations. The design will incorporate domain types, referential integrity constraints, and security features to ensure data confidentiality and integrity. The logical database design will include DDL operations for all tables with outputs, and the enforcement of primary and foreign key constraints. DML operations will be performed with outputs.

**List of tables**:

* User table
* Vehicle table
* Rental table
* Payments table
* Location table

**Attributes and Domain Types**:

User table:

* user\_id NUMBER(10)
* name VARCHAR(50)
* contact\_information VARCHAR(100)
* payment\_method VARCHAR(50)

Vehicle table:

* vehicle\_id NUMBER(10)
* model VARCHAR(50)
* rental\_rate DECIMAL(6,2)

Rental table:

* rental\_id NUMBER(10)
* user\_id NUMBER(10)
* vehicle\_id NUMBER(10)
* start\_date DATE
* end\_date DATE
* distance\_traveled NUMBER(10)

Payments table

* payment\_id NUMBER(10)
* rental\_id NUMBER(10)
* amount DECIMAL(6,2)
* payment\_date DATE,

Location table

* location\_id NUMBER(10)
* name VARCHAR(50)
* address VARCHAR(100)

Key constraints:

User table:

* user\_id - PRIMARY KEY

Vehicle table

* vehicle\_id - PRIMARY KEY

Rental table

* rental\_id - PRIMARY KEY
* user\_ID - FOREIGN KEY

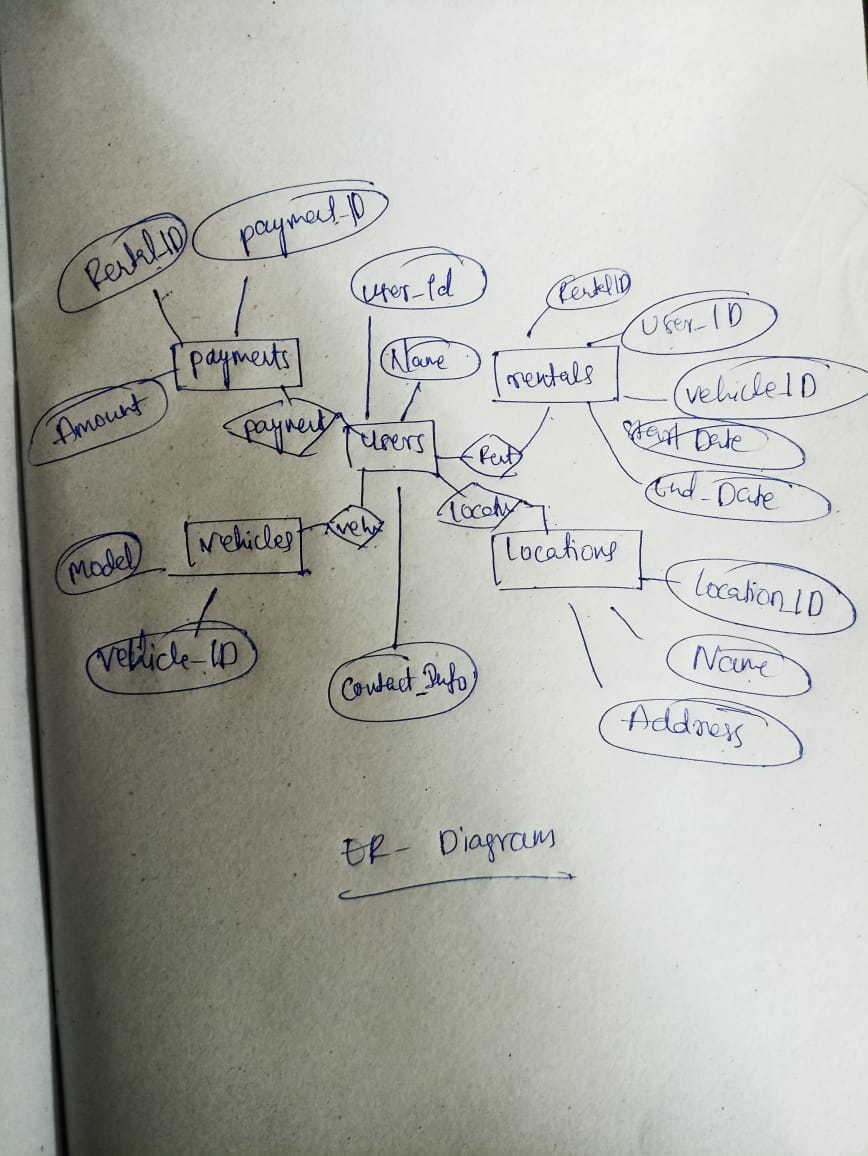
Payments table

* payment\_id INT PRIMARY KEY

Location table

* location\_id INT PRIMARY KEY

**ER Diagram:**



**DDL COMMANDS:**

User and vehicle tableText

Description automatically generated

Rental and payment tableText

Description automatically generatedlocations tableText

Description automatically generated

DML COMMANDS:

User table:Text

Description automatically generatedText

Description automatically generated

Vehicle tableText

Description automatically generated

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedlocation table:Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

**IMPLEMENTATION**

**JAVA-SQL Connectivity using 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 = "saicharan";

private static String password = "saicharan";

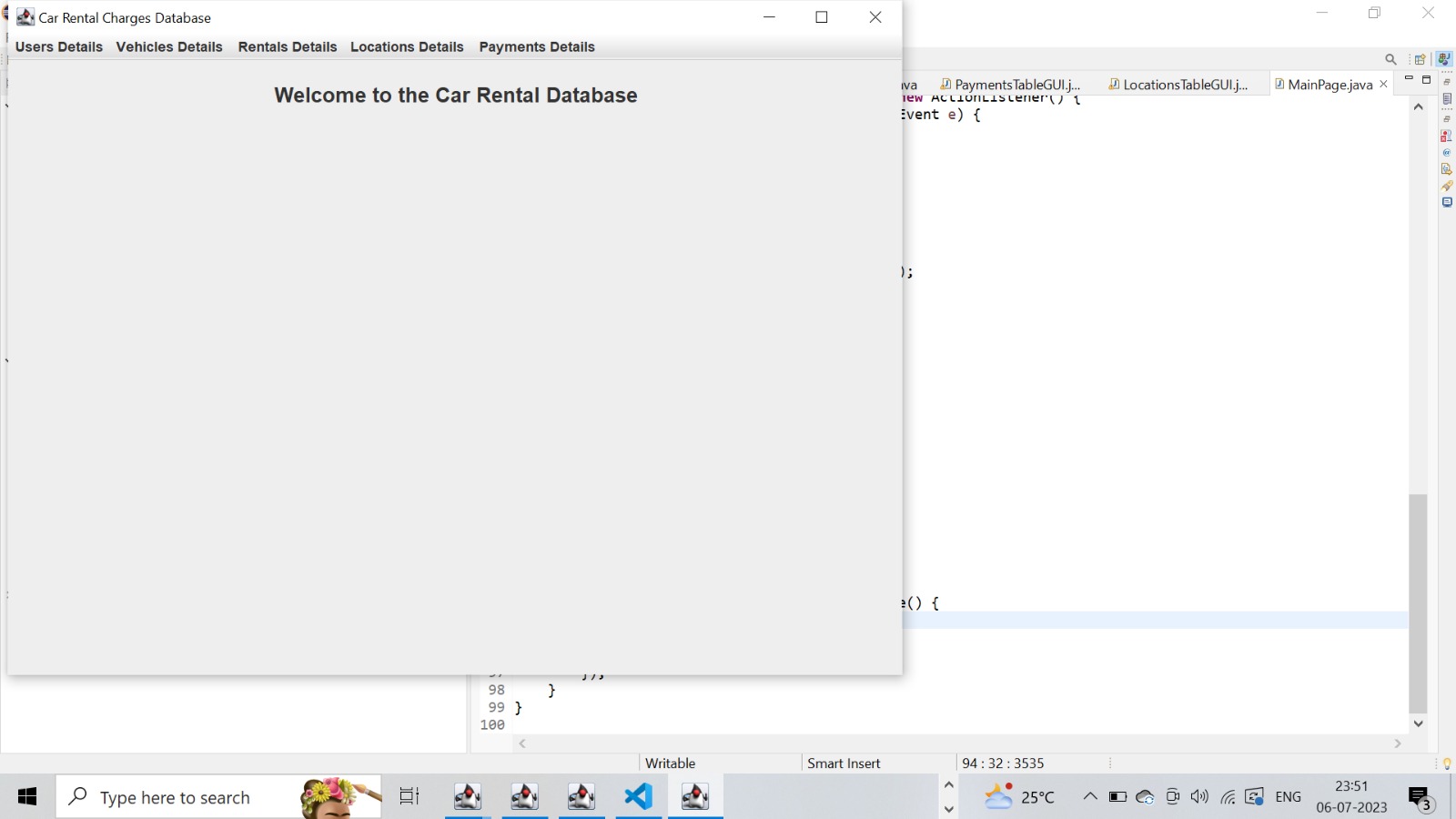
private static Connection con;

public static Connection getConnection () throws Exception [ con =

DriverManager.getConnection (url, username, password);

return con;

**MAIN PAGE**



**CODE:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class MainPage extends JFrame {

private static final long serialVersionUID = 1L;

public MainPage() {

// Set frame properties

setTitle("Car Rental Charges Database");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create label

JLabel welcomeLabel = new JLabel("Welcome to the Car Rental Database");

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 menu bar

JMenuBar menuBar = new JMenuBar();

// Create menus

JMenu usersMenu = new JMenu("Users Details");

JMenu vehiclesMenu = new JMenu("Vehicles Details");

JMenu rentalsMenu = new JMenu("Rentals Details");

JMenu locationsMenu = new JMenu("Locations Details");

JMenu paymentsMenu = new JMenu("Payments Details");

// Create menu items for Users menu

JMenuItem viewUsersDetails = new JMenuItem("View Users Details");

viewUsersDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new UsersTableGUI();

}

});

// Create menu items for Vehicles menu

JMenuItem viewVehiclesDetails = new JMenuItem("View Vehicles Details");

viewVehiclesDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new VehiclesTableGUI();

}

});

// Create menu items for Rentals menu

JMenuItem viewRentalsDetails = new JMenuItem("View Rentals Details");

viewRentalsDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new RentalsTableGUI();

}

});

// Create menu items for Locations menu

JMenuItem viewLocationsDetails = new JMenuItem("View Locations Details");

viewLocationsDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new LocationsTableGUI();

}

});

// Create menu items for Payments menu

JMenuItem viewPaymentsDetails = new JMenuItem("View Payments Details");

viewPaymentsDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new PaymentsTableGUI();

}

});

// Add menu items to respective menus

usersMenu.add(viewUsersDetails);

vehiclesMenu.add(viewVehiclesDetails);

rentalsMenu.add(viewRentalsDetails);

locationsMenu.add(viewLocationsDetails);

paymentsMenu.add(viewPaymentsDetails);

// Add menus to the menu bar

menuBar.add(usersMenu);

menuBar.add(vehiclesMenu);

menuBar.add(rentalsMenu);

menuBar.add(locationsMenu);

menuBar.add(paymentsMenu);

// Set the menu bar

setJMenuBar(menuBar);

// Set frame size and visibility

setSize(800, 600);

setVisible(true);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

new MainPage();

}

});

}

}

**USER PAGE:**

**A screenshot of a computer

Description automatically generated**

**CODE:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class UsersTableGUI extends JFrame {

private JTextField txtUserID, txtName, txtContactInfo, txtPaymentMethod;

private JTable tblUsers;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public UsersTableGUI() {

initializeUI();

connectToDatabase();

displayUsers();

}

private void initializeUI() {

txtUserID = new JTextField();

txtName = new JTextField();

txtContactInfo = new JTextField();

txtPaymentMethod = 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("Name:"), gbc);

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtUserID, gbc);

gbc.gridy++;

panel.add(txtName, gbc);

gbc.gridy++;

panel.add(txtContactInfo, gbc);

gbc.gridy++;

panel.add(txtPaymentMethod, 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("Users Table");

pack();

setLocationRelativeTo(null);

setVisible(true);

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

super.windowClosing(e);

disconnectFromDatabase();

}

});

}

private void connectToDatabase() {

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

String username = "saicharan";

String password = "saicharan";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void disconnectFromDatabase() {

try {

if (connection != null && !connection.isClosed()) {

connection.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertUser() {

int userID = Integer.parseInt(txtUserID.getText());

String name = txtName.getText();

String contactInfo = txtContactInfo.getText();

String paymentMethod = txtPaymentMethod.getText();

try {

String query = "INSERT INTO Users (user\_ID, name, contact\_information, payment\_method) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, userID);

statement.setString(2, name);

statement.setString(3, contactInfo);

statement.setString(4, paymentMethod);

statement.executeUpdate();

clearFields();

displayUsers();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyUser() {

int selectedRow = tblUsers.getSelectedRow();

if (selectedRow >= 0) {

int userID = Integer.parseInt(txtUserID.getText());

String name = txtName.getText();

String contactInfo = txtContactInfo.getText();

String paymentMethod = txtPaymentMethod.getText();

try {

String query = "UPDATE Users SET name=?, contact\_information=?, payment\_method=? WHERE user\_ID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, name);

statement.setString(2, contactInfo);

statement.setString(3, paymentMethod);

statement.setInt(4, 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) {

int userID = Integer.parseInt(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.setInt(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 name = resultSet.getString("name");

String contactInfo = resultSet.getString("contact\_information");

String paymentMethod = resultSet.getString("payment\_method");

users.add(new User(userID, name, contactInfo, paymentMethod));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"User ID", "Name", "Contact Information", "Payment Method"});

for (User user : users) {

model.addRow(new Object[]{user.getUserID(), user.getName(), user.getContactInfo(), user.getPaymentMethod()});

}

tblUsers.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectUser() {

int selectedRow = tblUsers.getSelectedRow();

if (selectedRow >= 0) {

int userID = Integer.parseInt(tblUsers.getValueAt(selectedRow, 0).toString());

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

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

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

txtUserID.setText(String.valueOf(userID));

txtName.setText(name);

txtContactInfo.setText(contactInfo);

txtPaymentMethod.setText(paymentMethod);

}

}

private void clearFields() {

txtUserID.setText("");

txtName.setText("");

txtContactInfo.setText("");

txtPaymentMethod.setText("");

}

private static class User {

private int userID;

private String name;

private String contactInfo;

private String paymentMethod;

public User(int userID, String name, String contactInfo, String paymentMethod) {

this.userID = userID;

this.name = name;

this.contactInfo = contactInfo;

this.paymentMethod = paymentMethod;

}

public int getUserID() {

return userID;

}

public String getName() {

return name;

}

public String getContactInfo() {

return contactInfo;

}

public String getPaymentMethod() {

return paymentMethod;

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(UsersTableGUI::new);

}

}

**VEHICLES:**

**A screenshot of a computer

Description automatically generated**

**CODE:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class VehiclesTableGUI extends JFrame {

private JTextField txtVehicleId, txtMake, txtModel, txtRentalRate;

private JTable tblVehicles;

private JButton btnAdd, btnModify, btnDelete;

private Connection connection;

public VehiclesTableGUI() {

initializeUI();

connectToDatabase();

displayVehicles();

}

private void initializeUI() {

txtVehicleId = new JTextField();

txtMake = new JTextField();

txtModel = new JTextField();

txtRentalRate = new JTextField();

tblVehicles = new JTable();

tblVehicles.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblVehicles.getSelectionModel().addListSelectionListener(e -> selectVehicle());

JScrollPane scrollPane = new JScrollPane(tblVehicles);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

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("Vehicle ID:"), gbc);

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtVehicleId, gbc);

gbc.gridy++;

panel.add(txtMake, gbc);

gbc.gridy++;

panel.add(txtModel, gbc);

gbc.gridy++;

panel.add(txtRentalRate, 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);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

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

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

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

setTitle("Vehicles");

pack();

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

private void connectToDatabase() {

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

String username = "saicharan";

String password = "saicharan";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertVehicle() {

String vehicleId = txtVehicleId.getText();

String make = txtMake.getText();

String model = txtModel.getText();

String rentalRate = txtRentalRate.getText();

try {

String query = "INSERT INTO Vehicles (vehicle\_id, make, model, rental\_rate) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, vehicleId);

statement.setString(2, make);

statement.setString(3, model);

statement.setString(4, rentalRate);

statement.executeUpdate();

clearFields();

displayVehicles();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyVehicle() {

int selectedRow = tblVehicles.getSelectedRow();

if (selectedRow >= 0) {

String vehicleId = txtVehicleId.getText();

String make = txtMake.getText();

String model = txtModel.getText();

String rentalRate = txtRentalRate.getText();

try {

String query = "UPDATE Vehicles SET make=?, model=?, rental\_rate=? WHERE vehicle\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, make);

statement.setString(2, model);

statement.setString(3, rentalRate);

statement.setString(4, vehicleId);

statement.executeUpdate();

clearFields();

displayVehicles();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

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

}

}

private void deleteVehicle() {

int selectedRow = tblVehicles.getSelectedRow();

if (selectedRow >= 0) {

String vehicleId = tblVehicles.getValueAt(selectedRow, 0).toString();

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

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

try {

String query = "DELETE FROM Vehicles WHERE vehicle\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, vehicleId);

statement.executeUpdate();

clearFields();

displayVehicles();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

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

}

}

private void displayVehicles() {

try {

String query = "SELECT \* FROM Vehicles";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Vehicle> vehicles = new ArrayList<>();

while (resultSet.next()) {

String vehicleId = resultSet.getString("vehicle\_id");

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

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

String rentalRate = resultSet.getString("rental\_rate");

vehicles.add(new Vehicle(vehicleId, make, model, rentalRate));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Vehicle ID", "Make", "Model", "Rental Rate"});

for (Vehicle vehicle : vehicles) {

model.addRow(new String[]{vehicle.getVehicleId(), vehicle.getMake(), vehicle.getModel(), vehicle.getRentalRate()});

}

tblVehicles.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectVehicle() {

int selectedRow = tblVehicles.getSelectedRow();

if (selectedRow >= 0) {

String vehicleId = tblVehicles.getValueAt(selectedRow, 0).toString();

String make = tblVehicles.getValueAt(selectedRow, 1).toString();

String model = tblVehicles.getValueAt(selectedRow, 2).toString();

String rentalRate = tblVehicles.getValueAt(selectedRow, 3).toString();

txtVehicleId.setText(vehicleId);

txtMake.setText(make);

txtModel.setText(model);

txtRentalRate.setText(rentalRate);

}

}

private void clearFields() {

txtVehicleId.setText("");

txtMake.setText("");

txtModel.setText("");

txtRentalRate.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(VehiclesTableGUI::new);

}

private class Vehicle {

private String vehicleId;

private String make;

private String model;

private String rentalRate;

public Vehicle(String vehicleId, String make, String model, String rentalRate) {

this.vehicleId = vehicleId;

this.make = make;

this.model = model;

this.rentalRate = rentalRate;

}

public String getVehicleId() {

return vehicleId;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public String getRentalRate() {

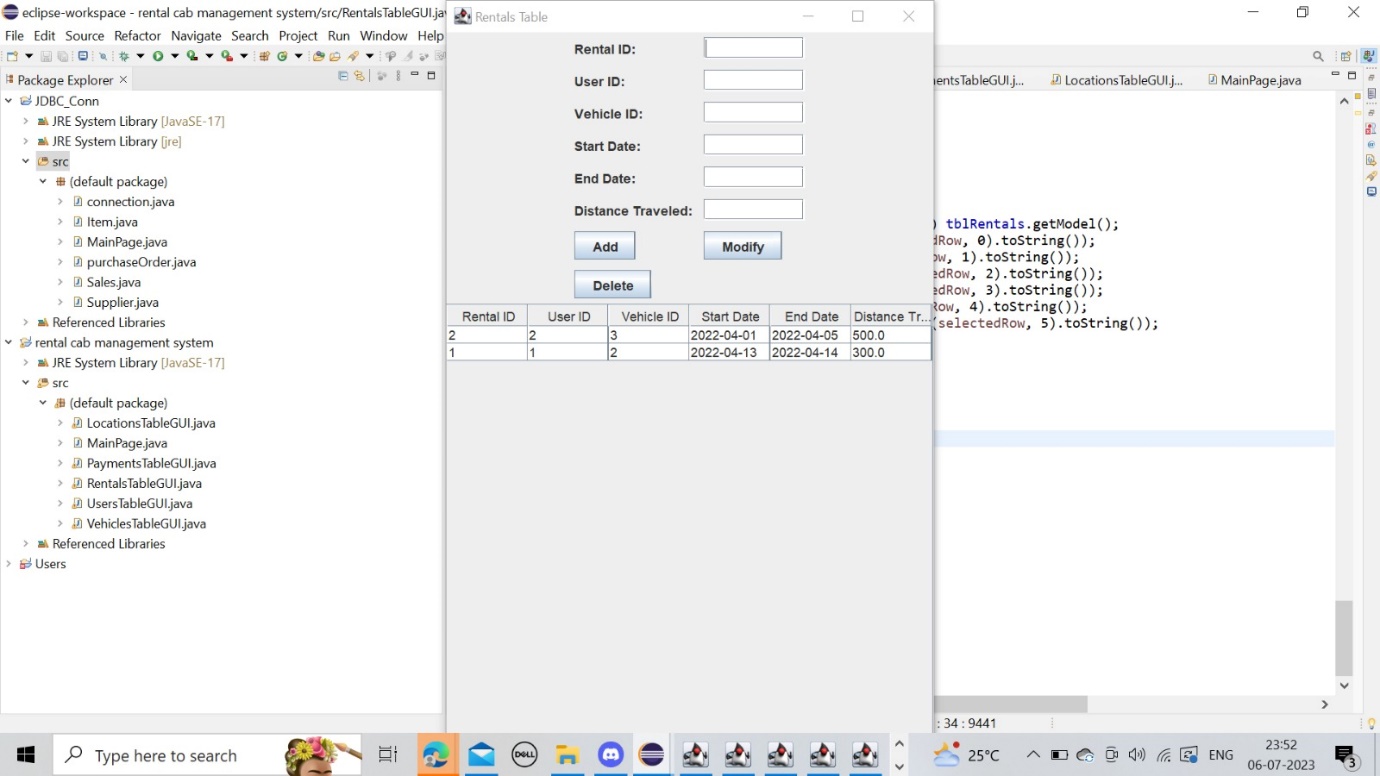
return rentalRate;

}

}

}

**RENTALS:**

****

**CODE:**

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 RentalsTableGUI extends JFrame {

private JTextField txtRentalId;

private JTextField txtUserId;

private JTextField txtVehicleId;

private JTextField txtStartDate;

private JTextField txtEndDate;

private JTextField txtDistanceTraveled;

private JButton btnAdd;

private JButton btnModify;

private JButton btnDelete;

private JTable tblRentals;

private JScrollPane scrollPane;

private Connection connection;

public RentalsTableGUI() {

initializeGUI();

connectToDatabase();

displayRentals();

}

private void initializeGUI() {

setTitle("Rentals Table");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

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

GridBagConstraints constraints = new GridBagConstraints();

constraints.gridx = 0;

constraints.gridy = 0;

constraints.anchor = GridBagConstraints.WEST;

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

panel.add(new JLabel("Rental ID:"), constraints);

txtRentalId = new JTextField(10);

constraints.gridx = 1;

panel.add(txtRentalId, constraints);

constraints.gridy = 1;

constraints.gridx = 0;

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

txtUserId = new JTextField(10);

constraints.gridx = 1;

panel.add(txtUserId, constraints);

constraints.gridy = 2;

constraints.gridx = 0;

panel.add(new JLabel("Vehicle ID:"), constraints);

txtVehicleId = new JTextField(10);

constraints.gridx = 1;

panel.add(txtVehicleId, constraints);

constraints.gridy = 3;

constraints.gridx = 0;

panel.add(new JLabel("Start Date:"), constraints);

txtStartDate = new JTextField(10);

constraints.gridx = 1;

panel.add(txtStartDate, constraints);

constraints.gridy = 4;

constraints.gridx = 0;

panel.add(new JLabel("End Date:"), constraints);

txtEndDate = new JTextField(10);

constraints.gridx = 1;

panel.add(txtEndDate, constraints);

constraints.gridy = 5;

constraints.gridx = 0;

panel.add(new JLabel("Distance Traveled:"), constraints);

txtDistanceTraveled = new JTextField(10);

constraints.gridx = 1;

panel.add(txtDistanceTraveled, constraints);

btnAdd = new JButton("Add");

constraints.gridy = 6;

constraints.gridx = 0;

panel.add(btnAdd, constraints);

btnModify = new JButton("Modify");

constraints.gridx = 1;

panel.add(btnModify, constraints);

btnDelete = new JButton("Delete");

constraints.gridy = 7;

constraints.gridx = 0;

panel.add(btnDelete, constraints);

add(panel, BorderLayout.NORTH);

tblRentals = new JTable();

scrollPane = new JScrollPane(tblRentals);

add(scrollPane, BorderLayout.CENTER);

pack();

setLocationRelativeTo(null);

btnAdd.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

addRental();

}

});

btnModify.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

modifyRental();

}

});

btnDelete.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

deleteRental();

}

});

tblRentals.getSelectionModel().addListSelectionListener(e -> selectRental());

}

private void connectToDatabase() {

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

String username = "saicharan";

String password = "saicharan";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void displayRentals() {

try {

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT \* FROM Rentals");

List<Object[]> rows = new ArrayList<>();

while (resultSet.next()) {

Object[] row = {

resultSet.getInt("rental\_id"),

resultSet.getInt("user\_id"),

resultSet.getInt("vehicle\_id"),

resultSet.getDate("start\_date"),

resultSet.getDate("end\_date"),

resultSet.getDouble("distance\_traveled")

};

rows.add(row);

}

String[] columnNames = {"Rental ID", "User ID", "Vehicle ID", "Start Date", "End Date", "Distance Traveled"};

DefaultTableModel model = new DefaultTableModel(rows.toArray(new Object[0][]), columnNames);

tblRentals.setModel(model);

statement.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void addRental() {

try {

int rentalId = Integer.parseInt(txtRentalId.getText());

int userId = Integer.parseInt(txtUserId.getText());

int vehicleId = Integer.parseInt(txtVehicleId.getText());

Date startDate = Date.valueOf(txtStartDate.getText());

Date endDate = Date.valueOf(txtEndDate.getText());

double distanceTraveled = Double.parseDouble(txtDistanceTraveled.getText());

String query = "INSERT INTO Rentals (rental\_id, user\_id, vehicle\_id, start\_date, end\_date, distance\_traveled) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, rentalId);

statement.setInt(2, userId);

statement.setInt(3, vehicleId);

statement.setDate(4, startDate);

statement.setDate(5, endDate);

statement.setDouble(6, distanceTraveled);

statement.executeUpdate();

statement.close();

clearInputFields();

displayRentals();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyRental() {

try {

int rentalId = Integer.parseInt(txtRentalId.getText());

int userId = Integer.parseInt(txtUserId.getText());

int vehicleId = Integer.parseInt(txtVehicleId.getText());

Date startDate = Date.valueOf(txtStartDate.getText());

Date endDate = Date.valueOf(txtEndDate.getText());

double distanceTraveled = Double.parseDouble(txtDistanceTraveled.getText());

String query = "UPDATE Rentals SET user\_id = ?, vehicle\_id = ?, start\_date = ?, end\_date = ?, distance\_traveled = ? WHERE rental\_id = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, userId);

statement.setInt(2, vehicleId);

statement.setDate(3, startDate);

statement.setDate(4, endDate);

statement.setDouble(5, distanceTraveled);

statement.setInt(6, rentalId);

statement.executeUpdate();

statement.close();

clearInputFields();

displayRentals();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void deleteRental() {

try {

int rentalId = Integer.parseInt(txtRentalId.getText());

String query = "DELETE FROM Rentals WHERE rental\_id = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, rentalId);

statement.executeUpdate();

statement.close();

clearInputFields();

displayRentals();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectRental() {

int selectedRow = tblRentals.getSelectedRow();

if (selectedRow >= 0) {

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

txtRentalId.setText(model.getValueAt(selectedRow, 0).toString());

txtUserId.setText(model.getValueAt(selectedRow, 1).toString());

txtVehicleId.setText(model.getValueAt(selectedRow, 2).toString());

txtStartDate.setText(model.getValueAt(selectedRow, 3).toString());

txtEndDate.setText(model.getValueAt(selectedRow, 4).toString());

txtDistanceTraveled.setText(model.getValueAt(selectedRow, 5).toString());

}

}

private void clearInputFields() {

txtRentalId.setText("");

txtUserId.setText("");

txtVehicleId.setText("");

txtStartDate.setText("");

txtEndDate.setText("");

txtDistanceTraveled.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new RentalsTableGUI().setVisible(true);

}

});

}

}

**LOCATIONS:**

**A screenshot of a computer

Description automatically generated**

**CODE:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class LocationsTableGUI extends JFrame {

private JTextField txtLocationId, txtName, txtAddress;

private JTable tblLocations;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public LocationsTableGUI() {

initializeUI();

connectToDatabase();

displayLocations();

}

private void initializeUI() {

txtLocationId = new JTextField();

txtName = new JTextField();

txtAddress = 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("Name:"), gbc);

gbc.gridy++;

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

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtLocationId, gbc);

gbc.gridy++;

panel.add(txtName, gbc);

gbc.gridy++;

panel.add(txtAddress, 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("Locations Table");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

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

String username = "saicharan";

String password = "saicharan";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertLocation() {

String locationId = txtLocationId.getText();

String name = txtName.getText();

String address = txtAddress.getText();

try {

String query = "INSERT INTO locations (location\_id, name, address) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, locationId);

statement.setString(2, name);

statement.setString(3, address);

statement.executeUpdate();

clearFields();

displayLocations();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyLocation() {

int selectedRow = tblLocations.getSelectedRow();

if (selectedRow >= 0) {

String locationId = txtLocationId.getText();

String name = txtName.getText();

String address = txtAddress.getText();

try {

String query = "UPDATE locations SET name=?, address=? WHERE location\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, name);

statement.setString(2, address);

statement.setString(3, 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 locations 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 locations";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

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

while (resultSet.next()) {

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

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

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

locations.add(new Location(locationId, name, address));

}

DefaultTableModel model = new DefaultTableModel();

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

for (Location location : locations) {

model.addRow(new String[]{location.getLocationId(), location.getName(), location.getAddress()});

}

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 name = tblLocations.getValueAt(selectedRow, 1).toString();

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

txtLocationId.setText(locationId);

txtName.setText(name);

txtAddress.setText(address);

}

}

private void clearFields() {

txtLocationId.setText("");

txtName.setText("");

txtAddress.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(LocationsTableGUI::new);

}

private class Location {

private String locationId;

private String name;

private String address;

public Location(String locationId, String name, String address) {

this.locationId = locationId;

this.name = name;

this.address = address;

}

public String getLocationId() {

return locationId;

}

public String getName() {

return name;

}

public String getAddress() {

return address;

}

}

}

**PAYMENTS:**

A screenshot of a computer

Description automatically generated

**CODE:**

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 PaymentsTableGUI extends JFrame {

private JTextField txtPaymentId;

private JTextField txtRentalId;

private JTextField txtAmount;

private JTextField txtPaymentDate;

private JButton btnAdd;

private JButton btnModify;

private JButton btnDelete;

private JTable tblPayments;

private JScrollPane scrollPane;

private Connection connection;

public PaymentsTableGUI() {

initializeGUI();

connectToDatabase();

displayPayments();

}

private void initializeGUI() {

setTitle("Payments Table");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

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

GridBagConstraints constraints = new GridBagConstraints();

constraints.gridx = 0;

constraints.gridy = 0;

constraints.anchor = GridBagConstraints.WEST;

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

panel.add(new JLabel("Payment ID:"), constraints);

txtPaymentId = new JTextField(10);

constraints.gridx = 1;

panel.add(txtPaymentId, constraints);

constraints.gridy = 1;

constraints.gridx = 0;

panel.add(new JLabel("Rental ID:"), constraints);

txtRentalId = new JTextField(10);

constraints.gridx = 1;

panel.add(txtRentalId, constraints);

constraints.gridy = 2;

constraints.gridx = 0;

panel.add(new JLabel("Amount:"), constraints);

txtAmount = new JTextField(10);

constraints.gridx = 1;

panel.add(txtAmount, constraints);

constraints.gridy = 3;

constraints.gridx = 0;

panel.add(new JLabel("Payment Date:"), constraints);

txtPaymentDate = new JTextField(10);

constraints.gridx = 1;

panel.add(txtPaymentDate, constraints);

btnAdd = new JButton("Add");

constraints.gridy = 4;

constraints.gridx = 0;

panel.add(btnAdd, constraints);

btnModify = new JButton("Modify");

constraints.gridx = 1;

panel.add(btnModify, constraints);

btnDelete = new JButton("Delete");

constraints.gridy = 5;

constraints.gridx = 0;

panel.add(btnDelete, constraints);

add(panel, BorderLayout.NORTH);

tblPayments = new JTable();

scrollPane = new JScrollPane(tblPayments);

add(scrollPane, BorderLayout.CENTER);

pack();

setLocationRelativeTo(null);

btnAdd.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

addPayment();

}

});

btnModify.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

modifyPayment();

}

});

btnDelete.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

deletePayment();

}

});

tblPayments.getSelectionModel().addListSelectionListener(e -> selectPayment());

}

private void connectToDatabase() {

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

String username = "saicharan";

String password = "saicharan";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void displayPayments() {

try {

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT \* FROM payments");

List<Object[]> rows = new ArrayList<>();

while (resultSet.next()) {

Object[] row = new Object[4];

row[0] = resultSet.getInt("payment\_id");

row[1] = resultSet.getInt("rental\_id");

row[2] = resultSet.getDouble("amount");

row[3] = resultSet.getString("payment\_date");

rows.add(row);

}

Object[][] rowData = new Object[rows.size()][];

for (int i = 0; i < rows.size(); i++) {

rowData[i] = rows.get(i);

}

String[] columnNames = {"Payment ID", "Rental ID", "Amount", "Payment Date"};

DefaultTableModel model = new DefaultTableModel(rowData, columnNames);

tblPayments.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void addPayment() {

String paymentId = txtPaymentId.getText();

String rentalId = txtRentalId.getText();

String amount = txtAmount.getText();

String paymentDate = txtPaymentDate.getText();

try {

String sql = "INSERT INTO payments (payment\_id, rental\_id, amount, payment\_date) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, paymentId);

statement.setString(2, rentalId);

statement.setString(3, amount);

statement.setString(4, paymentDate);

statement.executeUpdate();

displayPayments();

clearFields();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyPayment() {

int selectedRowIndex = tblPayments.getSelectedRow();

if (selectedRowIndex == -1) {

JOptionPane.showMessageDialog(this, "No payment selected.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String paymentId = txtPaymentId.getText();

String rentalId = txtRentalId.getText();

String amount = txtAmount.getText();

String paymentDate = txtPaymentDate.getText();

try {

String sql = "UPDATE payments SET payment\_id = ?, rental\_id = ?, amount = ?, payment\_date = ? WHERE payment\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, paymentId);

statement.setString(2, rentalId);

statement.setString(3, amount);

statement.setString(4, paymentDate);

statement.setString(5, paymentId);

statement.executeUpdate();

displayPayments();

clearFields();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void deletePayment() {

int selectedRowIndex = tblPayments.getSelectedRow();

if (selectedRowIndex == -1) {

JOptionPane.showMessageDialog(this, "No payment selected.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String paymentId = txtPaymentId.getText();

try {

String sql = "DELETE FROM payments WHERE payment\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, paymentId);

statement.executeUpdate();

displayPayments();

clearFields();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectPayment() {

int selectedRowIndex = tblPayments.getSelectedRow();

if (selectedRowIndex != -1) {

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

String paymentId = model.getValueAt(selectedRowIndex, 0).toString();

String rentalId = model.getValueAt(selectedRowIndex, 1).toString();

String amount = model.getValueAt(selectedRowIndex, 2).toString();

String paymentDate = model.getValueAt(selectedRowIndex, 3).toString();

txtPaymentId.setText(paymentId);

txtRentalId.setText(rentalId);

txtAmount.setText(amount);

txtPaymentDate.setText(paymentDate);

}

}

private void clearFields() {

txtPaymentId.setText("");

txtRentalId.setText("");

txtAmount.setText("");

txtPaymentDate.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new PaymentsTableGUI().setVisible(true);

}

});

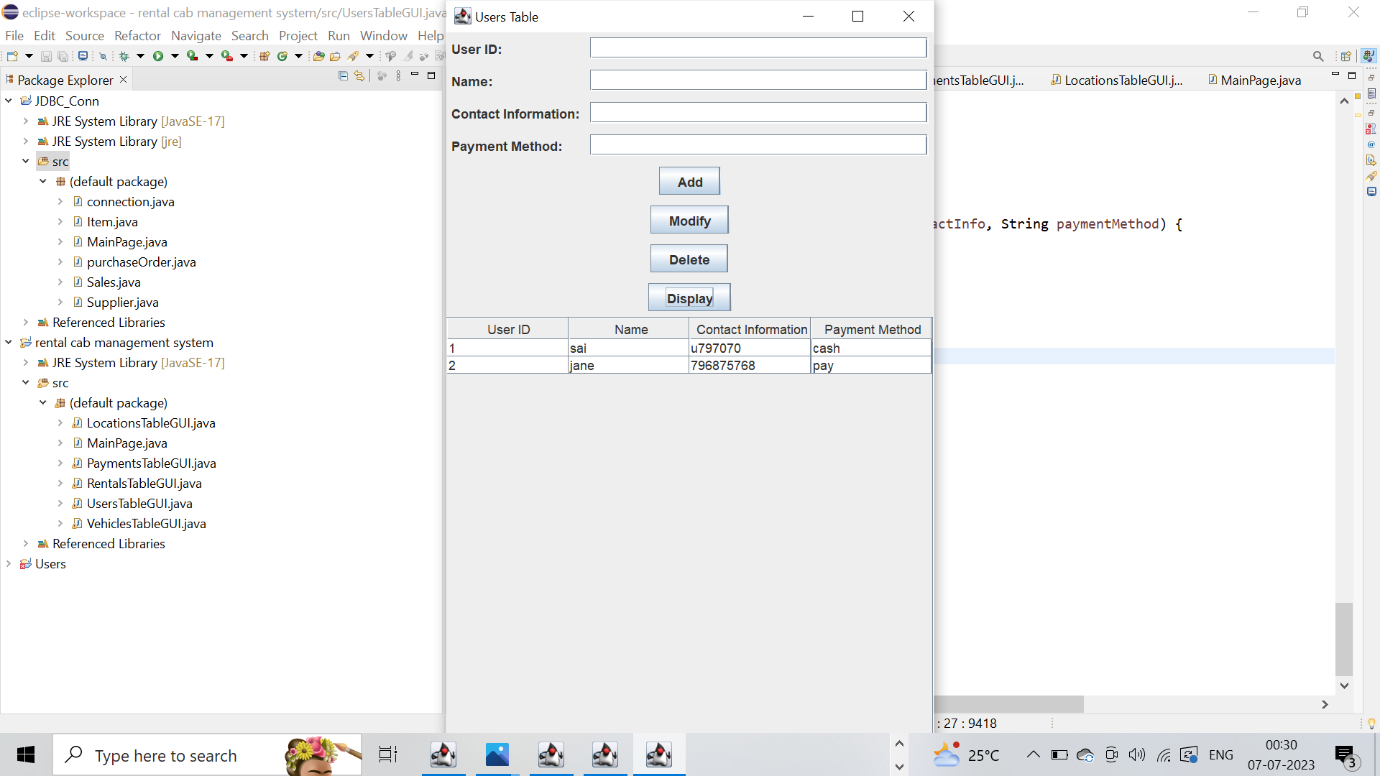
}

}

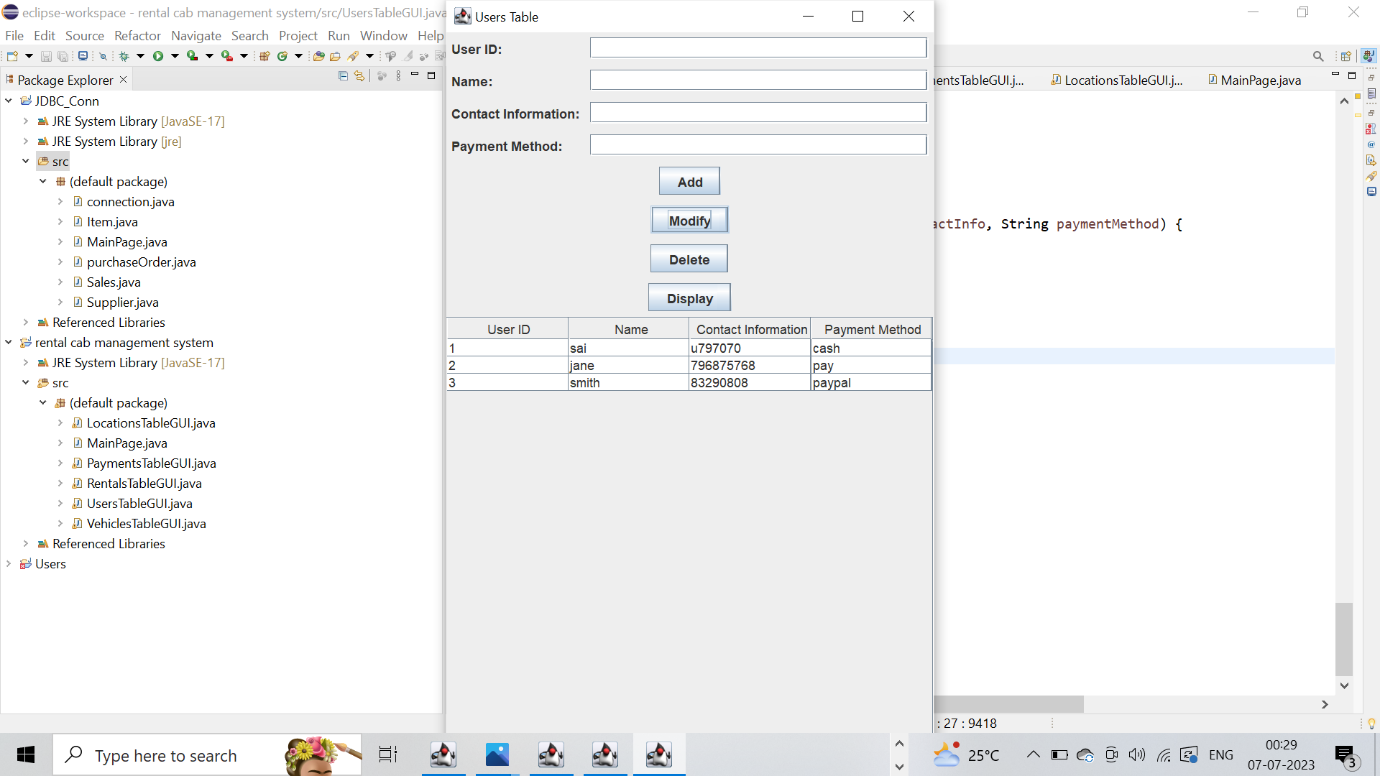
**TESTING:**

**USER PAGE:**

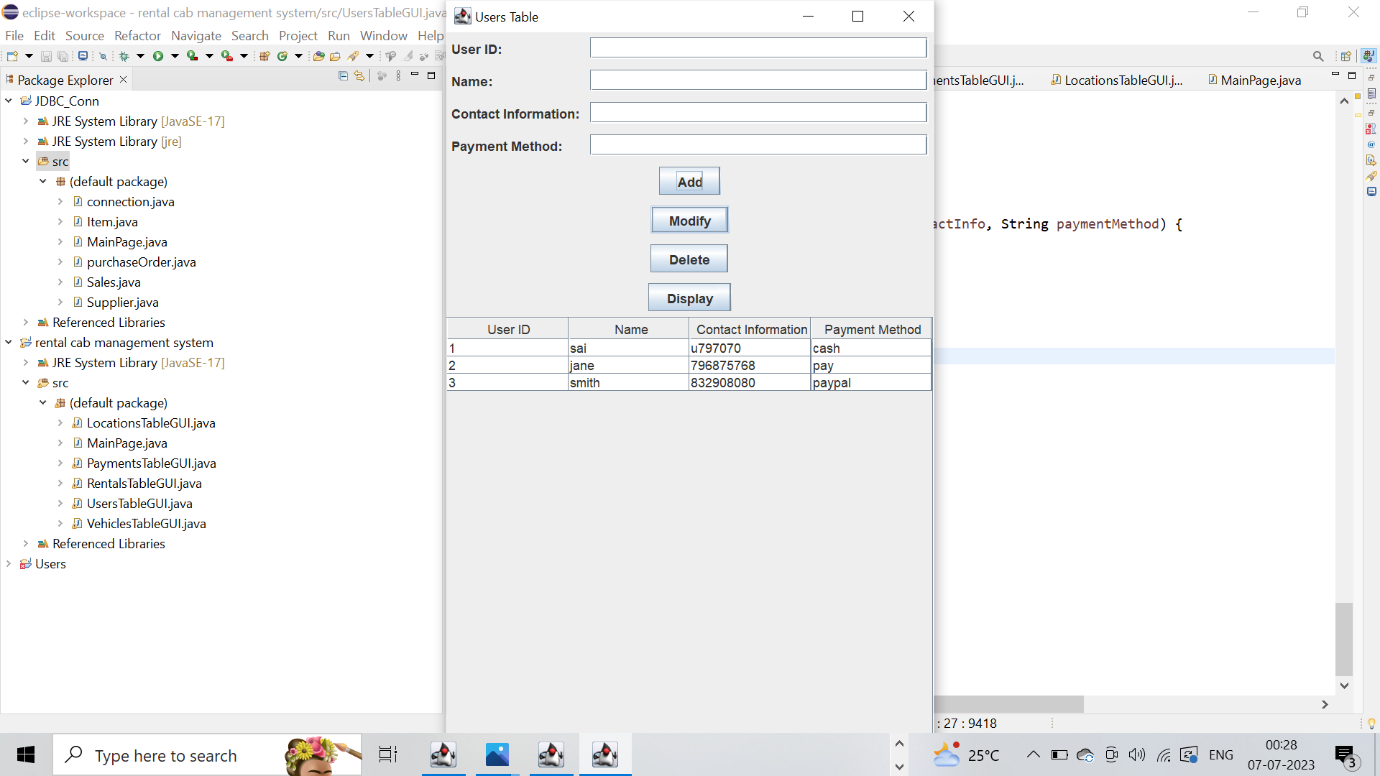
**AFTER INSERTION:**

****

**MODIFY:**

****

**DELETE:**

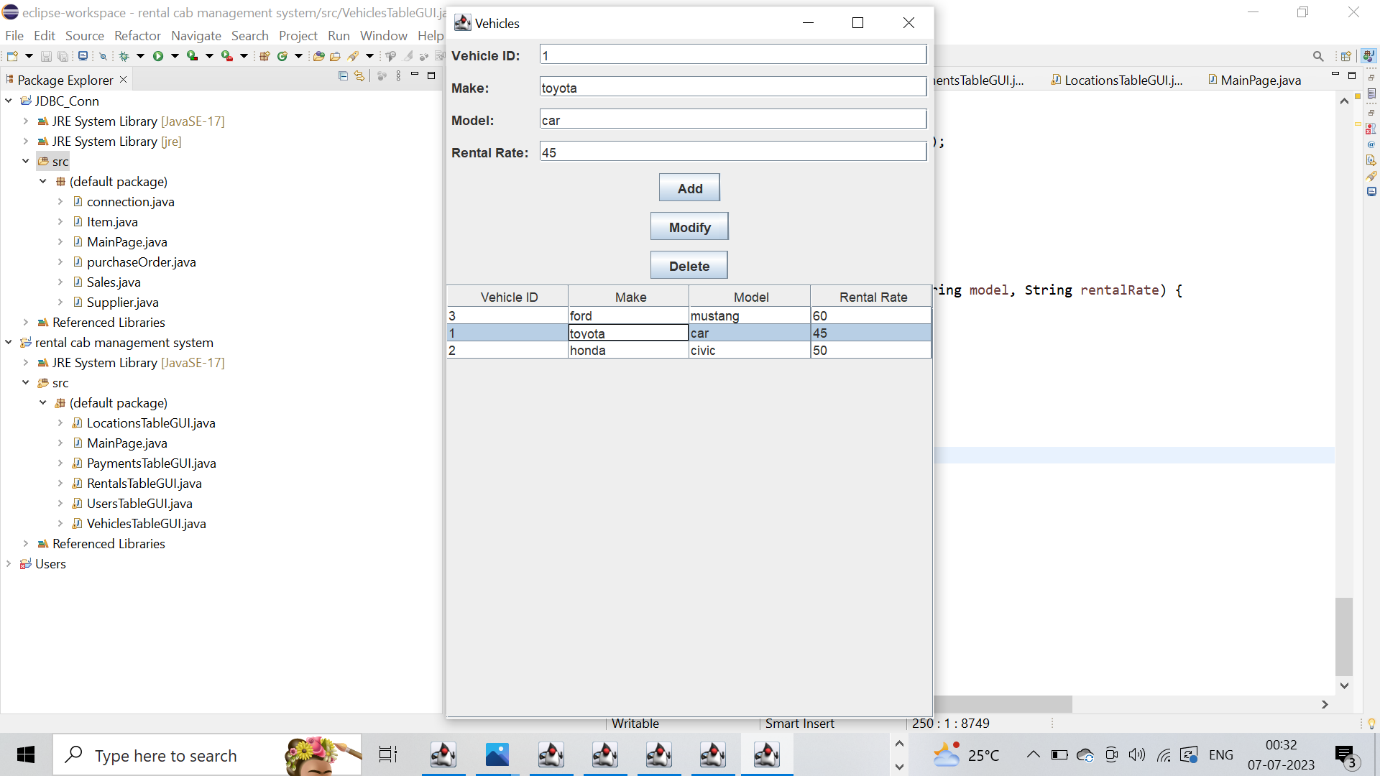
****

**VEHICLES PAGE:**

**INSERTION:**

**A screenshot of a computer

Description automatically generated**

**MODIFY:**

**DELETE:A screenshot of a computer

Description automatically generated**

**RENTALS PAGE:**

**INSERTIONA screenshot of a computer

Description automatically generated**

**MODIFY:A screenshot of a computer

Description automatically generated**

**DELETE:A screenshot of a computer

Description automatically generated**

**MODIFY:A screenshot of a computer

Description automatically generated**

**PAYMENTS PAGE:**

**INSERTION:A screenshot of a computer

Description automatically generated**

**MODIFY:A screenshot of a computer

Description automatically generated**

**RESULT**

I have successfully completed my DBMS project ‘Rental cab management system’.

**Benefits of Car Rental Cab Management System:**

Streamlined Booking Process: Customers can easily book a cab online or through a mobile app, eliminating the need for manual bookings. This streamlines the process and saves time for both customers and rental companies.

Efficient Fleet Management: The system allows rental companies to manage their fleet effectively. They can track vehicle availability, monitor maintenance schedules, and optimize vehicle allocation to ensure maximum utilization and profitability.

Online Reservations and Payments: Customers can make reservations and payments online, offering convenience and flexibility. It reduces the hassle of cash transactions and allows for secure online payment options.

**Future Work on Car Rental Cab Management System**:

Integration with Ride-Sharing Services: Integrating the system with popular ride-sharing platforms can expand the customer base and provide additional revenue streams for rental companies.

Mobile App Enhancements: Developing a user-friendly mobile app can enhance the customer experience, allowing them to book, track, and manage their rentals on the go.

IoT Integration: Implementing Internet of Things (IoT) technologies can enable advanced vehicle monitoring, predictive maintenance, and automated check-in/out processes for customers.

Dynamic Pricing: Implementing dynamic pricing algorithms based on factors like demand, seasonality, and vehicle availability can optimize revenue and improve profitability.

Integration with Smart Locking Systems: Integrating with smart locking systems can enable customers to access and start rental vehicles using their smartphones, enhancing convenience and security.

Enhancing Customer Support: Implementing chatbots or AI-powered customer support systems can provide instant assistance to customers, resolving queries and issues more efficiently.

Sustainability Initiatives: Implementing environmentally friendly options such as electric or hybrid vehicles in the rental fleet can align with sustainability goals and attract eco-conscious customers.

**References:**

● https://docs.oracle.com/javase/7/docs/api/

● https://www.javatpoint.com/java-swing

● <https://stackoverflow.com/>

**Git-Hub LINK:**

<https://github.com/saicharan-044/rentalcab-rental-management-system>

A screenshot of a computer

Description automatically generated