**METRO RESERVATION SYSTEM**

**(BIOMETRIC BASED)**

*A Report*

*Submitted in partial fulfillment of the Requirements*

*for the COURSE*

**DATABASE MANAGEMENT SYSTEMS**

**By**

### VOOTLA KRISHNA SAI SRINIVAS <1602-21-737-026>

**Under the guidance of Ms B. Leelavathy**



### Department of Information Technology Vasavi College of Engineering (Autonomous) (Affiliated to Osmania University) Ibrahimbagh, Hyderabad-31

**2022-2023**

BONAFIDE CERTIFICATE

This is to certify that this project report titled

***‘METRO RESERVATION SYSTEM (BIOMETRIC BASED)’*** is a project work of **Vootla Krishna Sai Srinivas** bearing roll no. 1602-21-737-026 who carried out this project under my supervision in the IV semester for the academic year 2022- 2023.

Signature Signature

External Examiner Internal Examiner

**ABSTRACT**

The Metro Reservation System (biometric based) is a cutting-edge solution designed to address the problems associated with traditional paper-based ticketing systems at metro stations. The system provides a more efficient and convenient method for passengers to book and manage their metro reservations, by using biometric authentication to grant access to reservation details.

The system allows passengers to view train schedules, reserve seats, and pay for their reservations using a secure payment gateway. By implementing a biometric-based system, passengers can quickly and securely access their reservation details without the need for physical tickets or manual checks. Additionally, the system provides real-time updates on train schedules, seat availability, and station information, allowing passengers to plan their journey more effectively.

The Metro Reservation System (biometric based) not only enhances the overall passenger experience but also increases the efficiency and revenue of the metro service provider. The system reduces the workload for metro employees, allowing them to focus on other aspects of providing quality service to passengers. This innovative system is a step towards a more efficient and sustainable public transportation system.

# **Requirement Analysis**

## **List of Tables**:

* Trains
* Reservations
* Stations
* Passengers

**List of Attributes with their Domain Types:**

**Trains**

* TRAIN\_ID NOT NULL NUMBER(5)
* TRAIN\_NAME VARCHAR2(15)
* ROUTE\_ID NUMBER(5)
* MAX\_SEATS NOT NULL NUMBER(3)
* ARRIVAL NOT NULL VARCHAR2(5)
* DEPARTURE NOT NULL VARCHAR2(5)

**Reservations**

* RESERVATIONS\_ID NOT NULL NUMBER(10)
* TRAIN\_ID NUMBER(5)
* BIOMETRIC\_ID NOT NULL NUMBER(10)
* RESERVATION\_TIME VARCHAR2(5)
* SEAT\_NUMBER VARCHAR2(3)
* TRAIN\_FARE FLOAT(10)

**Stations**

* STATION\_ID NOT NULL NUMBER(5)
* STATION\_NAME VARCHAR2(20)
* LATITUDE FLOAT(10)
* LONGITUDE FLOAT(10)
* NO\_OF\_EMPL NOT NULL NUMBER(3)

**Passengers**

* BIOMETRIC\_ID NOT NULL NUMBER(10)
* F\_NAME VARCHAR2(20)
* L\_NAME VARCHAR2(20)
* E\_MAIL VARCHAR2(50)
* ADDRESS VARCHAR2(50)
* MOBILE\_NUMBER NUMBER(10)

**AIM AND PRIORITY OF THE PROJECT**

To create a **Java GUI-based** desktop application that performs the functions

of a Metro Reservation system by taking the biometric of the customer during the

booking and allocate the seat(s) after the completion of payment. It takes values

like Biometric ID, First Name, Last Name, Email, Address, Mobile Number, etc

through forms which are then updated in the database using JDBC connectivity.

**ARCHITECTURE AND TECHNOLOGY**

### Software used:

Java, Oracle 11g Database, Java SE version 14, Run SQL.

### Java SWING:

**Java SWING** is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

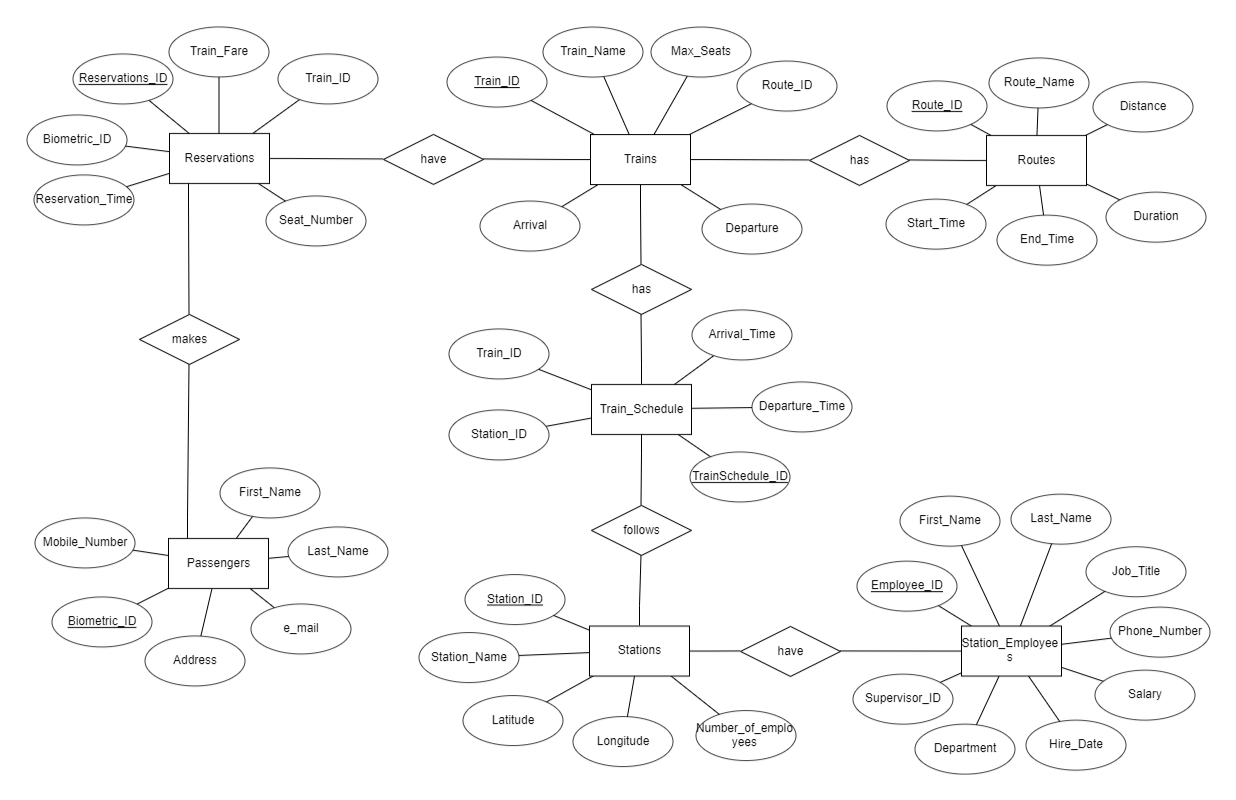
Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

### SQL:

Structure Query Language(SQL) is a database query language used for storing and managing data in **Relational** DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

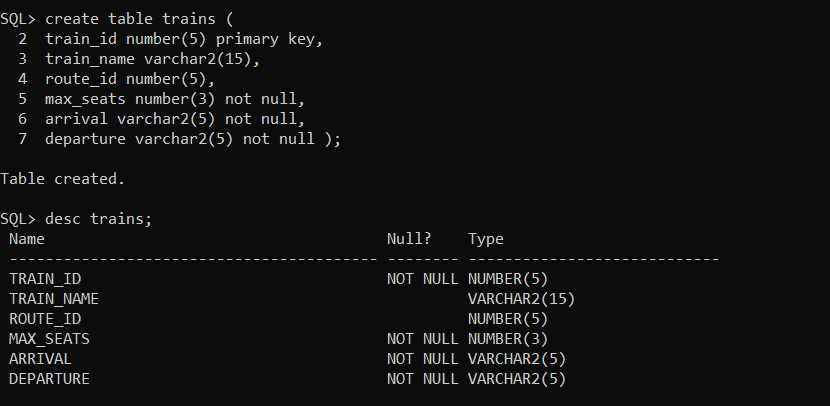
**DESIGN**

### Entity Relationship Diagram

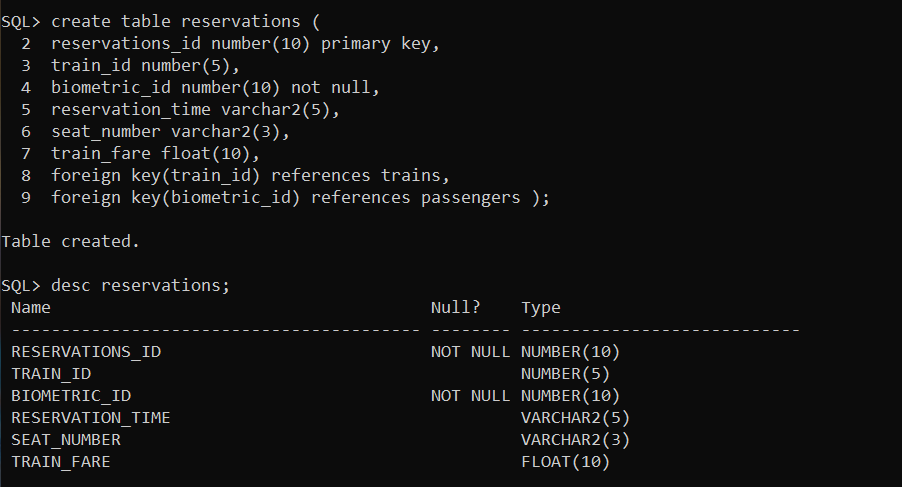


**TABLES CREATED IN SQL:**

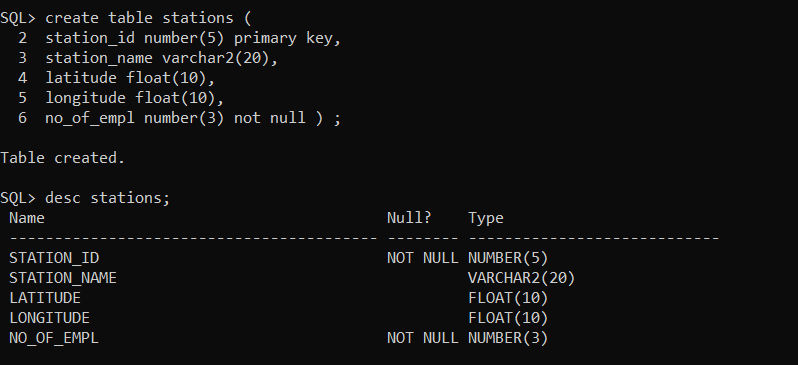
**1.** Trains Table



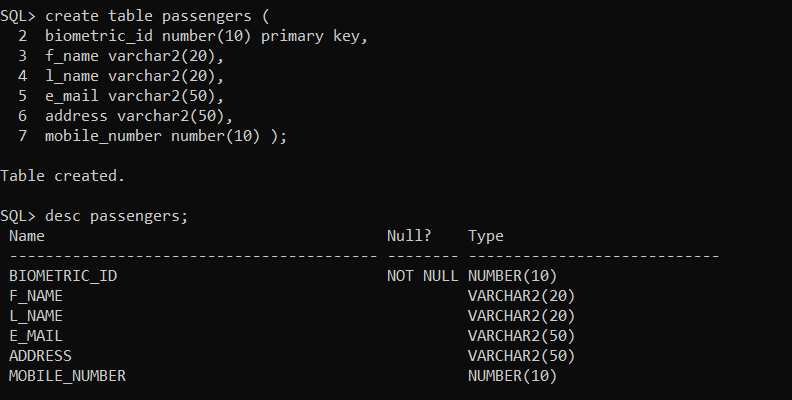
**2.** Reservations Table



**3.** Stations Table



**4.** Passengers Table



**DATABASE DESIGN:**

SQL> SELECT \* FROM TAB;

TNAME TABTYPE CLUSTERID

------------------------------ ------- ----------

PASSENGERS TABLE

RESERVATIONS TABLE

STATIONS TABLE

TRAINS TABLE

SQL> DESC PASSENGERS;

Name Null? Type

----------------------------------------- -------- ----------------------------

BIOMETRIC\_ID NOT NULL NUMBER(10)

F\_NAME VARCHAR2(20)

L\_NAME VARCHAR2(20)

E\_MAIL VARCHAR2(50)

ADDRESS VARCHAR2(50)

MOBILE\_NUMBER NUMBER(10)

SQL> DESC RESERVATIONS;

Name Null? Type

----------------------------------------- -------- ----------------------------

RESERVATIONS\_ID NOT NULL NUMBER(10)

TRAIN\_ID NUMBER(5)

BIOMETRIC\_ID NOT NULL NUMBER(10)

RESERVATION\_TIME VARCHAR2(5)

SEAT\_NUMBER VARCHAR2(3)

TRAIN\_FARE FLOAT(10)

SQL> DESC STATIONS;

Name Null? Type

----------------------------------------- -------- ----------------------------

STATION\_ID NOT NULL NUMBER(5)

STATION\_NAME VARCHAR2(20)

LATITUDE FLOAT(10)

LONGITUDE FLOAT(10)

NO\_OF\_EMPL NOT NULL NUMBER(3)

SQL> DESC TRAINS;

Name Null? Type

----------------------------------------- -------- ----------------------------

TRAIN\_ID NOT NULL NUMBER(5)

TRAIN\_NAME VARCHAR2(15)

ROUTE\_ID NUMBER(5)

MAX\_SEATS NOT NULL NUMBER(3)

ARRIVAL NOT NULL VARCHAR2(5)

DEPARTURE NOT NULL VARCHAR2(5)

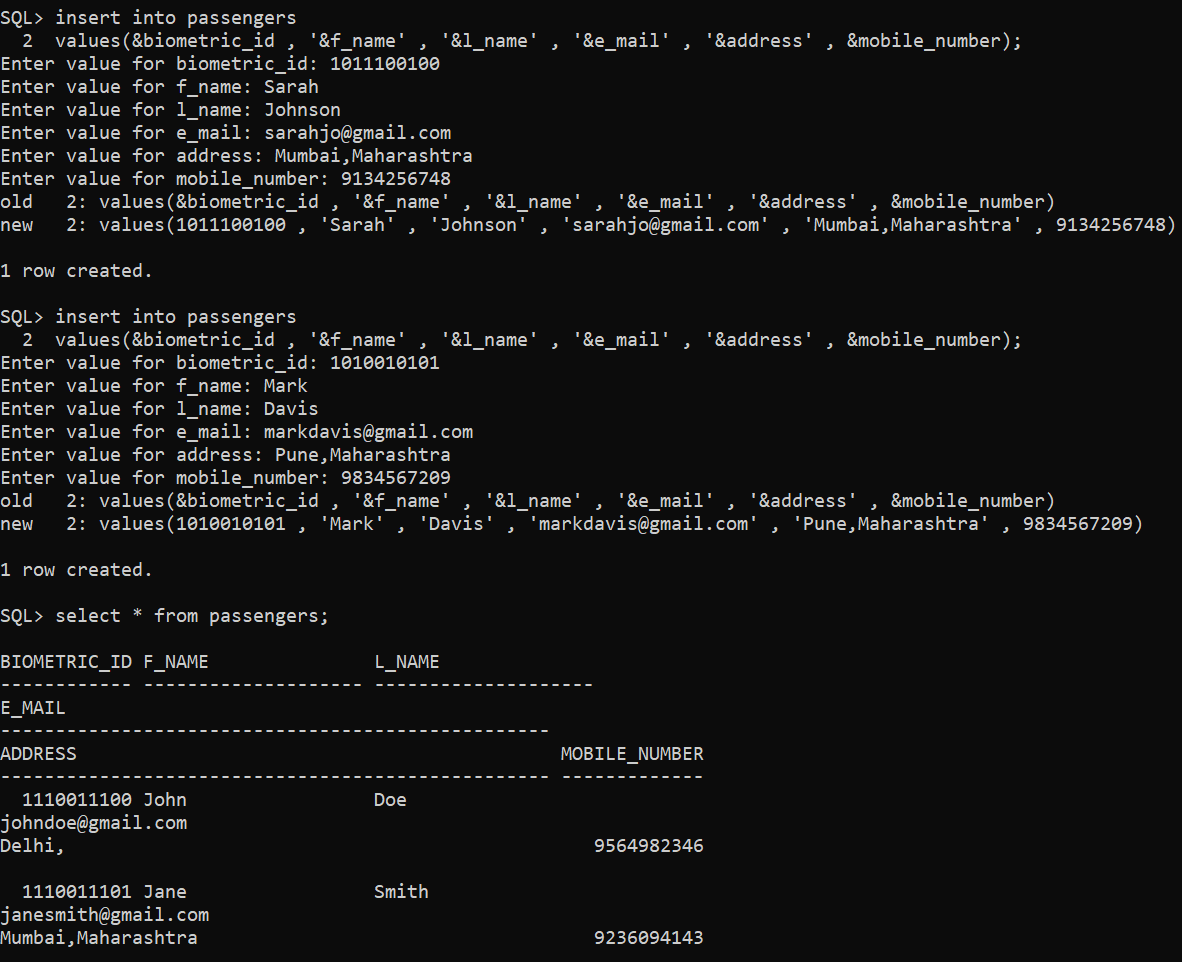
### DML Operations

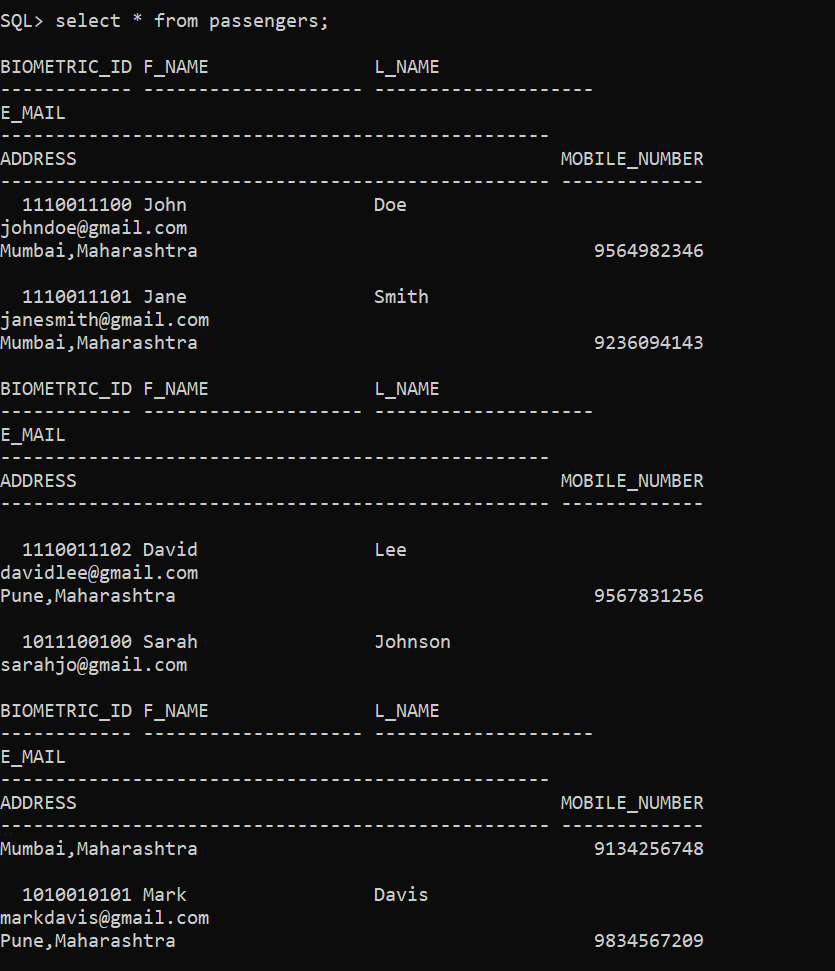
1.Insert values into Passengers:

**QUERY:**

insert into passengers values(&biometric\_id,'&f\_name','&l\_name',’&e\_mail’,'&address',&mobile\_number);

**OUTPUT:**





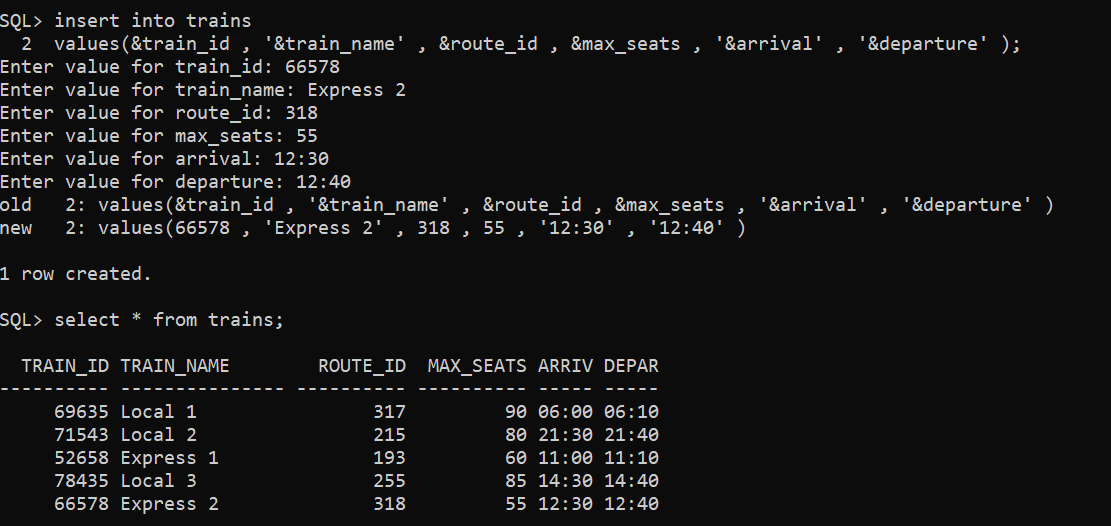
2.Insert values into Trains:

**QUERY:**

insert into trains

values(&train\_id, '&train\_name', &route\_id, &max\_seats ,’&arrival’ , ‘&departure’);

**OUTPUT:**



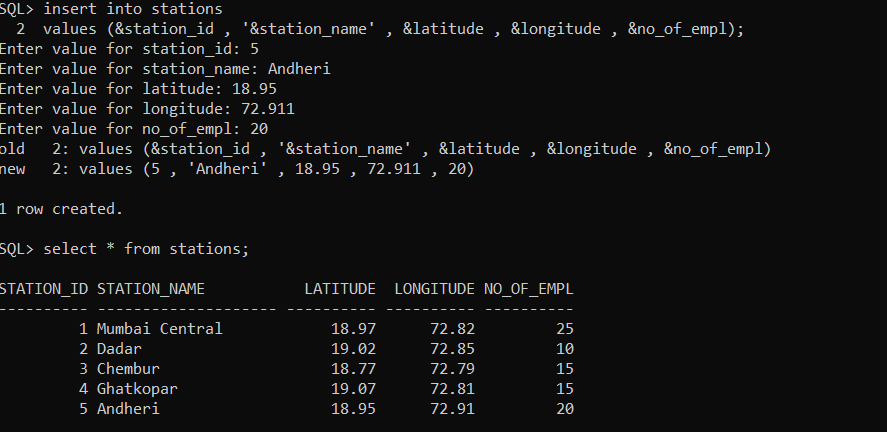
3.Inserting values into Stations:

**QUERY:**

insert into stations

values(&station\_id,'&station\_name',&latitude,&longitude,&no\_of\_empl);

**OUTPUT:**

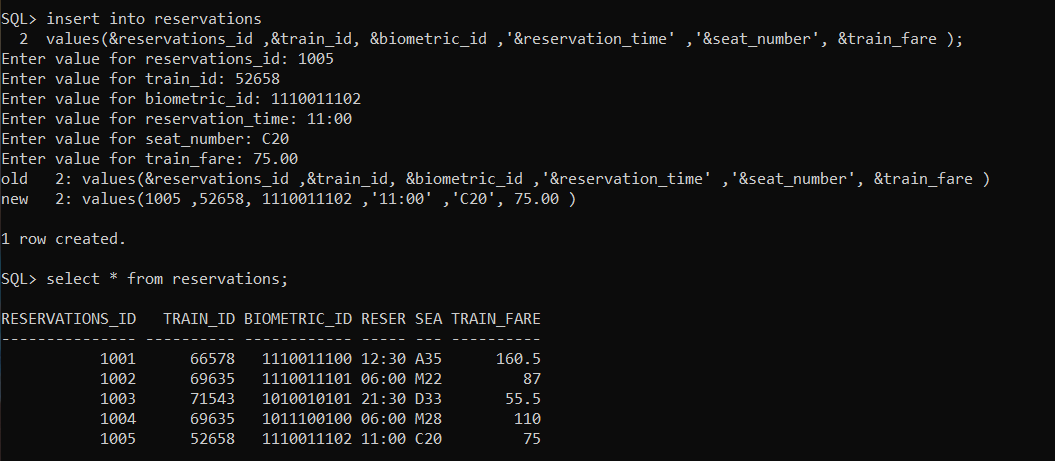
****

4.Insert values into Reservations:

**QUERY:**

insert into reservations values(&reservations\_id,&train\_id,&biometric\_id,'&reservation\_time','&seat\_number',&train\_fare);

**OUTPUT:**



**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:

(JDBC\_Insert.java)

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

public class JDBC\_Insert {

public static void main(String[] args)

{

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","krishna","vasavi");

PreparedStatement psmt = con.prepareStatement("insert into emp values(?,?,?)");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

while(true)

{

System.out.print("Enter Emp Id: ");

int eno = Integer.parseInt(br.readLine());

System.out.print("Enter Emp Name: ");

String ename = br.readLine();

System.out.print("Enter Emp Salary: ");

double esal = Double.parseDouble(br.readLine());

psmt.setInt(1, eno);

psmt.setString(2, ename);

psmt.setDouble(3, esal);

int count = psmt.executeUpdate();

if(count>0)

{

System.out.println(count+" record inserted");

}

else

{

System.out.println("No record inserted");

}

System.out.println("Do you want to continue inserting: ");

String ch = br.readLine();

if(ch.equalsIgnoreCase("no"))

break;

}

}

catch(Exception e)

{

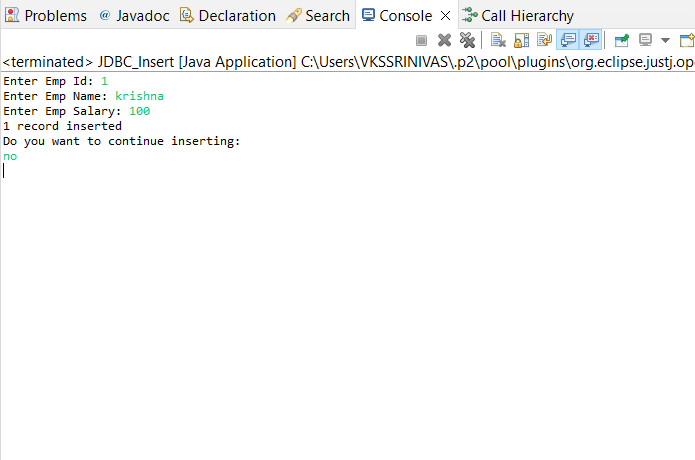
e.printStackTrace();

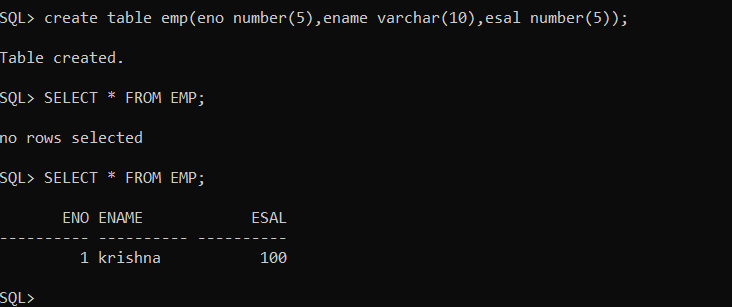
}

}

}

OUTPUT:





**Program:  
(ReservationSystemGUI.java)**

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

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

**import** java.sql.\*;

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

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

**import** javax.swing.table.DefaultTableModel;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.stream.Collectors;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**public** **class** ReservationSystemGUI **extends** JFrame {

**private** Connection connection;

**private** Statement statement;

**private** DefaultTableModel tableModel;

**private** JTable reservationsTable;

**private** JPanel welcomePanel;

**private** JPanel reservationsPanel ;

**private** JPanel createPanel;

**private** JPanel insertPanel;

**private** JPanel updatePanel;

**private** JPanel modifyPanel;

**private** JPanel DeletePanel;

**public** ReservationSystemGUI() {

initializeDatabase();

initComponents();

setLocationRelativeTo(**null**);

}

**private** **void** initializeDatabase() {

String DB\_URL = "jdbc:oracle:thin:@localhost:1521:XE";

String DB\_USERNAME = "krishna";

String DB\_PASSWORD = "vasavi";

**try** {

connection = DriverManager.*getConnection*(DB\_URL, DB\_USERNAME, DB\_PASSWORD);

statement = connection.createStatement();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** initComponents() {

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

setTitle("Reservation System");

setSize(900, 400);

// Create menu bar

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

setJMenuBar(menuBar);

// Create 'Home' menu

JMenu homeMenu = **new** JMenu("Home");

menuBar.add(homeMenu);

// Create 'Home' menu item

JMenuItem homeItem = **new** JMenuItem("Home");

homeMenu.add(homeItem);

// Create 'Admin' menu

JMenu adminMenu = **new** JMenu("Admin");

menuBar.add(adminMenu);

// Create 'View All Reservations' menu item

JMenuItem viewAllReservationsItem = **new** JMenuItem("View All Reservations");

adminMenu.add(viewAllReservationsItem);

// Create 'View Reservations by Train ID' menu item

JMenuItem viewReservationsByTrainIdItem = **new** JMenuItem("View Reservations by Train ID");

adminMenu.add(viewReservationsByTrainIdItem);

// Create 'User' menu

JMenu userMenu = **new** JMenu("User");

menuBar.add(userMenu);

// Create 'View your Reservation(s)' menu item

JMenuItem viewUserReservationsItem = **new** JMenuItem("View your Reservation(s)");

userMenu.add(viewUserReservationsItem);

// Create 'Insert/Create new Reservation(s)' menu item

JMenuItem insertReservationItem = **new** JMenuItem("Insert/Create new Reservation(s)");

userMenu.add(insertReservationItem);

// Create 'Modify/Update Detail(s)' menu item

JMenuItem modifyDetailsItem = **new** JMenuItem("Modify/Update Reservation Detail(s)");

userMenu.add(modifyDetailsItem);

// Create 'Delete/Cancel your Reservation(s)' menu item

JMenuItem deleteReservationItem = **new** JMenuItem("Delete/Cancel your Reservation(s)");

userMenu.add(deleteReservationItem);

// Create 'Forgot Biometric ID?' menu item

JMenuItem forgotBiometricIdItem = **new** JMenuItem("Forgot Biometric ID?");

userMenu.add(forgotBiometricIdItem);

// Create 'Create new User' menu item

JMenuItem createNewUserItem = **new** JMenuItem("Create new User(s)");

menuBar.add(createNewUserItem);

// Create 'Exit' menu item

JMenuItem exitItem = **new** JMenuItem("Exit");

menuBar.add(exitItem);

// Add action listener to 'Home' menu item

homeItem.addActionListener(e -> displayWelcomePanel());

// Add action listeners to menu items

viewAllReservationsItem.addActionListener(e -> viewAllReservations());

viewReservationsByTrainIdItem.addActionListener(e -> viewReservationsByTrainId());

viewUserReservationsItem.addActionListener(e -> viewUserReservations());

insertReservationItem.addActionListener(e -> insertReservation());

modifyDetailsItem.addActionListener(e -> modifyDetails());

deleteReservationItem.addActionListener(e -> deleteReservation());

forgotBiometricIdItem.addActionListener(e -> forgotBiometricId());

createNewUserItem.addActionListener(e -> createNewUser());

exitItem.addActionListener(e -> System.*exit*(0)); // Close the application

displayWelcomePanel();

setVisible(**true**);

}

**//Home has 1 option**

**//1.1 Home**

**//1.1 Home**

**private** **void** displayWelcomePanel() {

removePreviousPanel(); // Remove all components from the content pane

// Create a welcome panel

welcomePanel = **new** JPanel();

welcomePanel.setLayout(**new** BoxLayout(welcomePanel, BoxLayout.***Y\_AXIS***));

// Create welcome label

JLabel welcomeLabel = **new** JLabel("Welcome to the Metro Based Reservation System using Biometric ID");

welcomeLabel.setAlignmentX(Component.***CENTER\_ALIGNMENT***);

welcomeLabel.setFont(**new** Font("Arial", Font.***BOLD***, 16));

// Create note label

JLabel noteLabel = **new** JLabel("NOTE:");

noteLabel.setAlignmentX(Component.***CENTER\_ALIGNMENT***);

noteLabel.setFont(**new** Font("Arial", Font.***BOLD***, 14));

// Create fingerprint label

JLabel fingerprintLabel = **new** JLabel("Please keep your Right hand Thumb on the fingerprint scanner for the generation of Biometric ID for later use and then select from the above options.");

fingerprintLabel.setAlignmentX(Component.***CENTER\_ALIGNMENT***);

fingerprintLabel.setFont(**new** Font("Arial", Font.***PLAIN***, 12));

// Create note content label

JLabel noteContentLabel = **new** JLabel("The reservation made on a particular day (i.e., on the day of traveling on the metro) is only valid for that day and future ticket reservations are not possible.");

noteContentLabel.setAlignmentX(Component.***CENTER\_ALIGNMENT***);

noteContentLabel.setFont(**new** Font("Arial", Font.***PLAIN***, 12));

// Create train number table label

JLabel trainNumberLabel = **new** JLabel("Please check the Train\_Number's table to identify under which route your destination is present:");

trainNumberLabel.setAlignmentX(Component.***CENTER\_ALIGNMENT***);

// Create table with route and cities

String[] columnNames = {"Route\_No", "Train\_Fare" ,"Cities"};

Object[][] data = {

{"52658", "75" ,"City-1.1,City-1.2,..."},

{"66578", "160.5" ,"City-2.1,City-2.2,..."},

{"69635", "87" ,"City-3.1,City-3.2,..."},

{"71543", "55.5" ,"City-4.1,City-4.2,..."}

};

JTable table = **new** JTable(data, columnNames);

table.setAlignmentX(Component.***CENTER\_ALIGNMENT***);

// Create thank you label

JLabel thankYouLabel = **new** JLabel("\*\*\*THANK YOU\*\*\*");

thankYouLabel.setAlignmentX(Component.***CENTER\_ALIGNMENT***);

thankYouLabel.setFont(**new** Font("Arial", Font.***BOLD***, 16));

// Add components to the welcome panel

welcomePanel.add(Box.*createVerticalStrut*(20));

welcomePanel.add(welcomeLabel);

welcomePanel.add(Box.*createVerticalStrut*(10));

welcomePanel.add(fingerprintLabel);

welcomePanel.add(Box.*createVerticalStrut*(10));

welcomePanel.add(noteLabel);

welcomePanel.add(noteContentLabel);

welcomePanel.add(Box.*createVerticalStrut*(10));

welcomePanel.add(trainNumberLabel);

welcomePanel.add(table.getTableHeader());

welcomePanel.add(table);

welcomePanel.add(Box.*createVerticalStrut*(10));

welcomePanel.add(thankYouLabel);

welcomePanel.add(Box.*createVerticalStrut*(20));

// Add the welcome panel to the content pane

getContentPane().add(welcomePanel, BorderLayout.***CENTER***);

revalidate(); // Refresh the frame

repaint(); // Repaint the frame

}

**private** **void** removePreviousPanel()

{

**if**(welcomePanel!=**null**)

{

getContentPane().remove(welcomePanel);

}

**if**(reservationsPanel!=**null**)

{

getContentPane().remove(reservationsPanel);

}

**if**(createPanel!=**null**)

{

getContentPane().remove(createPanel);

}

**if**(insertPanel!=**null**)

{

getContentPane().remove(insertPanel);

}

}

**//Admin has 2 options  
//2.1 View all Reservations**

**//2.2 View Reservations by Train ID**

**//2.1 View all Reservations**

**private** **void** viewAllReservations() {

// Add your implementation here

JOptionPane.*showMessageDialog*(**this**, "View All Reservations option selected");

// Clear the content pane

removePreviousPanel();

**try** {

// Execute a query to fetch all reservations

String query = "SELECT \* FROM reservations";

ResultSet resultSet = statement.executeQuery(query);

// Create a table model with the column names

String[] columnNames = {"Reservations ID", "Train ID", "Biometric ID", "Reservation Time", "Seat Number", "Train Fare"};

tableModel = **new** DefaultTableModel(columnNames, 0);

// Populate the table model with reservation data

**while** (resultSet.next()) {

**int** reservationId = resultSet.getInt("Reservations\_ID");

**int** trainId = resultSet.getInt("Train\_ID");

**int** biometricId = resultSet.getInt("Biometric\_ID");

String reservationTime = resultSet.getString("Reservation\_Time");

String seatNumber = resultSet.getString("Seat\_Number");

**double** trainFare = resultSet.getDouble("Train\_Fare");

Object[] rowData = {reservationId, trainId, biometricId, reservationTime, seatNumber, trainFare};

tableModel.addRow(rowData);

}

// Create a JTable with the table model

reservationsTable = **new** JTable(tableModel);

// Create a scroll pane and add the table to it

JScrollPane scrollPane = **new** JScrollPane(reservationsTable);

// Add the scroll pane to the content pane

reservationsPanel = **new** JPanel(**new** BorderLayout());

reservationsPanel.add(scrollPane, BorderLayout.***CENTER***);

getContentPane().add(reservationsPanel, BorderLayout.***CENTER***);

// Refresh the frame

revalidate();

repaint();

// Close the result set

resultSet.close();

} **catch** (SQLException e) {

e.printStackTrace();

JOptionPane.*showMessageDialog*(**this**, "An error occurred while retrieving reservations.", "Error", JOptionPane.***ERROR\_MESSAGE***);

}

}

**//2.2 View Reservations by Train ID**

**private** JTextField trainIdTextField;

**private** JTextArea outputTextArea;

**private** **void** viewReservationsByTrainId() {

removePreviousPanel();

// Add your implementation here

JOptionPane.*showMessageDialog*(**this**, "View Reservations by Train ID option selected");

// Create UI components

JFrame frame = **new** JFrame("View Reservations by Train ID");

JLabel trainIdLabel = **new** JLabel("Enter Train ID:");

trainIdTextField = **new** JTextField();

trainIdTextField.setPreferredSize(**new** Dimension(200, 50));

JButton submitButton = **new** JButton("Submit");

JButton exitButton = **new** JButton("Exit");

JScrollPane tableScrollPane = **new** JScrollPane();

reservationsTable = **new** JTable();

// Configure UI layout

frame.setLayout(**new** BorderLayout());

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

frame.setSize(600, 400);

// Create a table model

DefaultTableModel tableModel = **new** DefaultTableModel();

tableModel.addColumn("Reservation ID");

tableModel.addColumn("Train ID");

tableModel.addColumn("Biometric ID");

tableModel.addColumn("Reservation Time");

tableModel.addColumn("Seat Number");

tableModel.addColumn("Train Fare");

reservationsTable.setModel(tableModel);

tableScrollPane.setViewportView(reservationsTable);

// Add components to the frame

JPanel inputPanel = **new** JPanel(**new** FlowLayout());

inputPanel.add(trainIdLabel);

inputPanel.add(trainIdTextField);

inputPanel.add(submitButton);

inputPanel.add(exitButton);

frame.add(inputPanel, BorderLayout.***NORTH***);

frame.add(tableScrollPane, BorderLayout.***CENTER***);

// Define the action listener for the submit button

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

@Override

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

// Get the train ID from the text field

**int** trainId = Integer.*parseInt*(trainIdTextField.getText());

// Retrieve reservations by train ID from the database

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna" , "vasavi")) {

String query = "SELECT \* FROM reservations WHERE Train\_ID = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, trainId);

ResultSet resultSet = statement.executeQuery();

// Clear existing table data

tableModel.setRowCount(0);

// Add retrieved reservations to the table model

**while** (resultSet.next()) {

**int** reservationId = resultSet.getInt("Reservations\_ID");

**int** retrievedTrainId = resultSet.getInt("Train\_ID");

**int** biometricId = resultSet.getInt("Biometric\_ID");

String reservationTime = resultSet.getString("Reservation\_Time");

String seatNumber = resultSet.getString("Seat\_Number");

**double** trainFare = resultSet.getDouble("Train\_Fare");

Object[] rowData = {reservationId, retrievedTrainId, biometricId, reservationTime, seatNumber, trainFare};

tableModel.addRow(rowData);

}

} **catch** (SQLException ex) {

ex.printStackTrace();

}

}

});

// Define the action listener for the exit button

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

@Override

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

frame.dispose(); // Close the JFrame

}

});

// Add the scroll pane to the content pane

insertPanel = **new** JPanel(**new** BorderLayout());

/\*insertPanel.add(trainIdLabel);

insertPanel.add(trainIdTextField);

insertPanel.add(submitButton);\*/

insertPanel.add(tableScrollPane, BorderLayout.***CENTER***);

getContentPane().add(insertPanel, BorderLayout.***CENTER***);

// Show the frame

frame.setVisible(**true**);

revalidate();

repaint();

}

**//User has 5 options**

**//3.1 View your Reservation(s)**

**//3.2 Insert/Create new Reservation(s)**

**//3.3 Modify/Update Reservation Detail(s)**

**//3.4 Delete/Cancel your Reservation(s)**

**//3.5 Forgot Biometric ID?**

**//3.1 View your Reservation(s)**

**private** **void** viewUserReservations() {

removePreviousPanel();

// Add your implementation here

JOptionPane.*showMessageDialog*(**this**, "View My Reservations option selected");

// Create UI components

JFrame frame = **new** JFrame("View User Reservations");

JLabel biometricIdLabel = **new** JLabel("Enter Biometric ID:");

JTextField biometricIdTextField = **new** JTextField();

biometricIdTextField.setPreferredSize(**new** Dimension(200, 50));

JButton submitButton = **new** JButton("Submit");

JButton exitButton = **new** JButton("Exit");

JTable reservationsTable = **new** JTable();

JScrollPane tableScrollPane = **new** JScrollPane(reservationsTable);

// Configure UI layout

frame.setLayout(**new** BorderLayout());

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

frame.setSize(600, 400);

// Create a table model

DefaultTableModel tableModel = **new** DefaultTableModel();

tableModel.addColumn("Reservation ID");

tableModel.addColumn("Train ID");

tableModel.addColumn("Biometric ID");

tableModel.addColumn("Reservation Time");

tableModel.addColumn("Seat Number");

tableModel.addColumn("Train Fare");

reservationsTable.setModel(tableModel);

// Add components to the frame

JPanel inputPanel = **new** JPanel(**new** FlowLayout());

inputPanel.add(biometricIdLabel);

inputPanel.add(biometricIdTextField);

inputPanel.add(submitButton);

inputPanel.add(exitButton);

frame.add(inputPanel, BorderLayout.***NORTH***);

frame.add(tableScrollPane, BorderLayout.***CENTER***);

// Define the action listener for the submit button

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

@Override

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

// Get the biometric ID from the text field

**int** biometricId = Integer.*parseInt*(biometricIdTextField.getText());

// Retrieve reservations by biometric ID from the database

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna", "vasavi")) {

String query = "SELECT \* FROM reservations WHERE Biometric\_ID = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, biometricId);

ResultSet resultSet = statement.executeQuery();

// Clear existing table data

tableModel.setRowCount(0);

// Add retrieved reservations to the table model

**while** (resultSet.next()) {

**int** reservationId = resultSet.getInt("Reservations\_ID");

**int** trainId = resultSet.getInt("Train\_ID");

**int** retrievedBiometricId = resultSet.getInt("Biometric\_ID");

String reservationTime = resultSet.getString("Reservation\_Time");

String seatNumber = resultSet.getString("Seat\_Number");

**double** trainFare = resultSet.getDouble("Train\_Fare");

Object[] rowData = {reservationId, trainId, retrievedBiometricId, reservationTime, seatNumber, trainFare};

tableModel.addRow(rowData);

}

// If no matching records found, display a message

**if** (tableModel.getRowCount() == 0) {

JOptionPane.*showMessageDialog*(frame, "No matching records found", "No Records", JOptionPane.***INFORMATION\_MESSAGE***);

}

} **catch** (SQLException ex) {

ex.printStackTrace();

}

}

});

// Define the action listener for the exit button

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

@Override

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

frame.dispose(); // Close the JFrame

}

});

// Add the scroll pane to the content pane

insertPanel = **new** JPanel(**new** BorderLayout());

/\*insertPanel.add(biometricIdLabel);

insertPanel.add(biometricIdTextField);

insertPanel.add(submitButton);\*/

insertPanel.add(tableScrollPane, BorderLayout.***CENTER***);

getContentPane().add(insertPanel, BorderLayout.***CENTER***);

// Show the frame

frame.setVisible(**true**);

revalidate();

repaint();

}

**//3.2 Insert/Create new Reservation(s)**

**private** **void** insertReservation() {

removePreviousPanel();

// Add your implementation here

JOptionPane.*showMessageDialog*(**this**, "Insert/Create New Reservation(s) option selected");

// Create UI components

JFrame frame = **new** JFrame("Insert Reservation");

JLabel reservationIdLabel = **new** JLabel("Enter Reservation ID:");

JTextField reservationIdTextField = **new** JTextField();

reservationIdTextField.setPreferredSize(**new** Dimension(200, 50));

JLabel trainIdLabel = **new** JLabel("Enter Train ID:");

JTextField trainIdTextField = **new** JTextField();

trainIdTextField.setPreferredSize(**new** Dimension(200, 50));

JLabel biometricIdLabel = **new** JLabel("Enter Biometric ID:");

JTextField biometricIdTextField = **new** JTextField();

biometricIdTextField.setPreferredSize(**new** Dimension(200, 50));

JLabel reservationTimeLabel = **new** JLabel("Enter Reservation Time:");

JTextField reservationTimeTextField = **new** JTextField();

reservationTimeTextField.setPreferredSize(**new** Dimension(200, 50));

JLabel seatNumberLabel = **new** JLabel("Enter Seat Number:");

JTextField seatNumberTextField = **new** JTextField();

seatNumberTextField.setPreferredSize(**new** Dimension(200, 50));

JLabel trainFareLabel = **new** JLabel("Enter Train Fare:");

JTextField trainFareTextField = **new** JTextField();

trainFareTextField.setPreferredSize(**new** Dimension(200, 50));

JButton submitButton = **new** JButton("Submit");

JButton exitButton = **new** JButton("Exit");

JTable reservationsTable = **new** JTable();

JScrollPane tableScrollPane = **new** JScrollPane(reservationsTable);

// Configure UI layout

frame.setLayout(**new** BorderLayout());

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

frame.setSize(600, 400);

/\* Create a table model

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Reservation ID");

tableModel.addColumn("Train ID");

tableModel.addColumn("Biometric ID");

tableModel.addColumn("Reservation Time");

tableModel.addColumn("Seat Number");

tableModel.addColumn("Train Fare");

reservationsTable.setModel(tableModel);\*/

// Add components to the frame

JPanel inputPanel = **new** JPanel(**new** GridLayout(7, 2));

inputPanel.add(reservationIdLabel);

inputPanel.add(reservationIdTextField);

inputPanel.add(trainIdLabel);

inputPanel.add(trainIdTextField);

inputPanel.add(biometricIdLabel);

inputPanel.add(biometricIdTextField);

inputPanel.add(reservationTimeLabel);

inputPanel.add(reservationTimeTextField);

inputPanel.add(seatNumberLabel);

inputPanel.add(seatNumberTextField);

inputPanel.add(trainFareLabel);

inputPanel.add(trainFareTextField);

inputPanel.add(submitButton);

inputPanel.add(exitButton);

frame.add(inputPanel, BorderLayout.***NORTH***);

frame.add(tableScrollPane, BorderLayout.***CENTER***);

// Define the action listener for the submit button

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

@Override

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

// Get the input values from the text fields

**int** reservationId = 0;

**int** trainId = 0;

**int** biometricId = 0;

String reservationTime = "";

String seatNumber = "";

**double** trainFare = 0.0;

**try** {

reservationId = Integer.*parseInt*(reservationIdTextField.getText());

trainId = Integer.*parseInt*(trainIdTextField.getText());

biometricId = Integer.*parseInt*(biometricIdTextField.getText());

reservationTime = reservationTimeTextField.getText();

seatNumber = seatNumberTextField.getText();

trainFare = Double.*parseDouble*(trainFareTextField.getText());

} **catch** (NumberFormatException ex) {

JOptionPane.*showMessageDialog*(frame, "Invalid details entered. Please enter valid numeric values.", "Invalid Details", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

// Validate seat number

**if** (seatNumber.isEmpty()) {

JOptionPane.*showMessageDialog*(frame, "Seat number cannot be empty", "Invalid Seat Number", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

// Validate train fare

Double[] fares = {75.0, 110.0, 55.5, 87.0, 160.5};

List<Double> fareList = Arrays.*stream*(fares).collect(Collectors.*toList*());

**if** (!fareList.contains(trainFare)) {

JOptionPane.*showMessageDialog*(frame, "Invalid train fare entered. Please enter a valid fare from the list: 75.0, 55.5, 87.0, 160.5 for Routes 52658 ,71543 ,69635 and 66578 respectively.", "Invalid Train Fare", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

// Validate train ID

**if** (!isValidTrainId(trainId)) {

JOptionPane.*showMessageDialog*(frame, "Invalid Train ID. Please enter a valid Train ID.", "Invalid Train ID", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

// Insert reservation into the database

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna", "vasavi")) {

String query = "INSERT INTO reservations (Reservations\_ID, Train\_ID, Biometric\_ID, Reservation\_Time, Seat\_Number, Train\_Fare) " +

"VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, reservationId);

statement.setInt(2, trainId);

statement.setInt(3, biometricId);

statement.setString(4, reservationTime);

statement.setString(5, seatNumber);

statement.setDouble(6, trainFare);

**int** rowsAffected = statement.executeUpdate();

// If the reservation was inserted successfully, display a success message

**if** (rowsAffected > 0) {

JOptionPane.*showMessageDialog*(frame, "Reservation inserted successfully", "Success", JOptionPane.***INFORMATION\_MESSAGE***);

} **else** {

JOptionPane.*showMessageDialog*(frame, "Failed to insert reservation", "Error", JOptionPane.***ERROR\_MESSAGE***);

}

// Clear existing table data

tableModel.setRowCount(0);

} **catch** (SQLException ex) {

ex.printStackTrace();

}

}

});

// Define the action listener for the exit button

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

@Override

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

frame.dispose(); // Close the JFrame

}

});

// Show the frame

frame.setVisible(**true**);

}

**//3.3 Modify/Update Reservation Detail(s)**

**private** **void** modifyDetails() {

removePreviousPanel();

// Add your implementation here

JOptionPane.*showMessageDialog*(**this**, "Modify/Update Reservation Detail(s) option selected");

// Create UI components

JFrame frame = **new** JFrame("Modify Reservation Details");

JLabel reservationIdLabel = **new** JLabel("Enter Reservation ID:");

JTextField reservationIdTextField = **new** JTextField();

JLabel trainIdLabel = **new** JLabel("Enter Train ID:");

JTextField trainIdTextField = **new** JTextField();

JLabel biometricIdLabel = **new** JLabel("Enter Biometric ID:");

JTextField biometricIdTextField = **new** JTextField();

JLabel reservationTimeLabel = **new** JLabel("Enter Reservation Time:");

JTextField reservationTimeTextField = **new** JTextField();

JLabel seatNumberLabel = **new** JLabel("Enter Seat Number:");

JTextField seatNumberTextField = **new** JTextField();

JLabel trainFareLabel = **new** JLabel("Enter Train Fare:");

JTextField trainFareTextField = **new** JTextField();

JButton modifyButton = **new** JButton("Modify");

JButton exitButton = **new** JButton("Exit");

JScrollPane tableScrollPane = **new** JScrollPane();

JTable reservationsTable = **new** JTable();

// Configure UI layout

frame.setLayout(**new** BorderLayout());

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

frame.setSize(600, 400);

/\* Create a table model

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Reservation ID");

tableModel.addColumn("Train ID");

tableModel.addColumn("Biometric ID");

tableModel.addColumn("Reservation Time");

tableModel.addColumn("Seat Number");

tableModel.addColumn("Train Fare");

reservationsTable.setModel(tableModel);

tableScrollPane.setViewportView(reservationsTable);\*/

// Add components to the frame

JPanel inputPanel = **new** JPanel(**new** GridLayout(7, 2));

inputPanel.add(reservationIdLabel);

inputPanel.add(reservationIdTextField);

inputPanel.add(trainIdLabel);

inputPanel.add(trainIdTextField);

inputPanel.add(biometricIdLabel);

inputPanel.add(biometricIdTextField);

inputPanel.add(reservationTimeLabel);

inputPanel.add(reservationTimeTextField);

inputPanel.add(seatNumberLabel);

inputPanel.add(seatNumberTextField);

inputPanel.add(trainFareLabel);

inputPanel.add(trainFareTextField);

inputPanel.add(modifyButton);

inputPanel.add(exitButton);

frame.add(inputPanel, BorderLayout.***NORTH***);

frame.add(tableScrollPane, BorderLayout.***CENTER***);

// Define the action listener for the modify button

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

@Override

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

// Get input values

**int** reservationId = Integer.*parseInt*(reservationIdTextField.getText());

**int** trainId = Integer.*parseInt*(trainIdTextField.getText());

**int** biometricId = Integer.*parseInt*(biometricIdTextField.getText());

String reservationTime = reservationTimeTextField.getText();

String seatNumber = seatNumberTextField.getText();

**double** trainFare = Double.*parseDouble*(trainFareTextField.getText());

// Validate input values

**if** (!isValidReservationId(reservationId) || !isValidTrainFare(trainFare) || !isValidBiometricId(biometricId) ||

!isValidTrainId(trainId) /\*|| !isValidReservationTime(reservationTime) || !isValidSeatNumber(seatNumber)\*/) {

JOptionPane.*showMessageDialog*(frame, "Invalid details entered. Please check your input.", "Invalid Details", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

// Modify reservation details in the database

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna", "vasavi")) {

String query = "UPDATE reservations SET Train\_ID = ?, Biometric\_ID = ?, Reservation\_Time = ?, Seat\_Number = ?, Train\_Fare = ? WHERE Reservations\_ID = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, trainId);

statement.setInt(2, biometricId);

statement.setString(3, reservationTime);

statement.setString(4, seatNumber);

statement.setDouble(5, trainFare);

statement.setInt(6, reservationId);

**int** rowsAffected = statement.executeUpdate();

**if** (rowsAffected > 0) {

JOptionPane.*showMessageDialog*(frame, "Modification successful.", "Success", JOptionPane.***INFORMATION\_MESSAGE***);

} **else** {

JOptionPane.*showMessageDialog*(frame, "Reservation ID not found.", "Error", JOptionPane.***ERROR\_MESSAGE***);

}

} **catch** (SQLException ex) {

ex.printStackTrace();

}

}

});

// Define the action listener for the exit button

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

@Override

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

frame.dispose();

}

});

// Show the frame

frame.setVisible(**true**);

}

**//3.4 Delete/Cancel your Reservation(s)**

**private** **void** deleteReservation() {

removePreviousPanel();

// Add your implementation here

JOptionPane.*showMessageDialog*(**this**, "Delete/Cancel Your Reservation(s) option selected");

// Create UI components

JFrame frame = **new** JFrame("Delete Reservation");

JLabel biometricIdLabel = **new** JLabel("Enter Biometric ID:");

JTextField biometricIdTextField = **new** JTextField();

biometricIdTextField.setPreferredSize(**new** Dimension(200, 50));

JLabel trainIdLabel = **new** JLabel("Enter Reservation ID:");

JTextField trainIdTextField = **new** JTextField();

trainIdTextField.setPreferredSize(**new** Dimension(200, 50));

JButton submitButton = **new** JButton("Submit");

JButton exitButton = **new** JButton("Exit");

JTable reservationsTable = **new** JTable();

JScrollPane tableScrollPane = **new** JScrollPane(reservationsTable);

// Configure UI layout

frame.setLayout(**new** BorderLayout());

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

frame.setSize(600, 400);

/\* Create a table model

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Reservation ID");

tableModel.addColumn("Train ID");

tableModel.addColumn("Biometric ID");

tableModel.addColumn("Reservation Time");

tableModel.addColumn("Seat Number");

tableModel.addColumn("Train Fare");

reservationsTable.setModel(tableModel);\*/

// Add components to the frame

JPanel inputPanel = **new** JPanel(**new** GridLayout(3, 2));

inputPanel.add(biometricIdLabel);

inputPanel.add(biometricIdTextField);

inputPanel.add(trainIdLabel);

inputPanel.add(trainIdTextField);

inputPanel.add(submitButton);

inputPanel.add(exitButton);

frame.add(inputPanel, BorderLayout.***NORTH***);

frame.add(tableScrollPane, BorderLayout.***CENTER***);

// Define the action listener for the submit button

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

@Override

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

// Get the input values from the text fields

**int** biometricId = Integer.*parseInt*(biometricIdTextField.getText());

**int** trainId = Integer.*parseInt*(trainIdTextField.getText());

// Delete reservations from the database

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna", "vasavi")) {

String query = "DELETE FROM reservations WHERE Biometric\_ID = ? AND Reservations\_ID = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, biometricId);

statement.setInt(2, trainId);

**int** rowsAffected = statement.executeUpdate();

// If any rows were deleted, display a success message

**if** (rowsAffected > 0) {

JOptionPane.*showMessageDialog*(frame, "Reservation deleted successfully", "Success", JOptionPane.***INFORMATION\_MESSAGE***);

} **else** {

JOptionPane.*showMessageDialog*(frame, "No matching reservations found", "No Reservations", JOptionPane.***INFORMATION\_MESSAGE***);

}

// Clear existing table data

tableModel.setRowCount(0);

} **catch** (SQLException ex) {

ex.printStackTrace();

}

}

});

// Define the action listener for the exit button

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

@Override

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

frame.dispose(); // Close the JFrame

}

});

// Show the frame

frame.setVisible(**true**);

}

**//3.5 Forgot Biometric ID?**

**private** **void** forgotBiometricId() {

removePreviousPanel();

// Add your implementation here

JOptionPane.*showMessageDialog*(**this**, "Forgot Biometric ID? option selected");

// Create UI components

JFrame frame = **new** JFrame("Forgot Biometric ID");

JLabel firstNameLabel = **new** JLabel("Enter First Name:");

JTextField firstNameTextField = **new** JTextField();

firstNameTextField.setPreferredSize(**new** Dimension(200, 50));

JLabel lastNameLabel = **new** JLabel("Enter Last Name:");

JTextField lastNameTextField = **new** JTextField();

lastNameTextField.setPreferredSize(**new** Dimension(200, 50));

JLabel mobileNumberLabel = **new** JLabel("Enter Mobile Number:");

JTextField mobileNumberTextField = **new** JTextField();

mobileNumberTextField.setPreferredSize(**new** Dimension(200, 50));

JButton submitButton = **new** JButton("Submit");

JButton exitButton = **new** JButton("Exit");

JTable passengersTable = **new** JTable();

JScrollPane tableScrollPane = **new** JScrollPane(passengersTable);

// Configure UI layout

frame.setLayout(**new** BorderLayout());

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

frame.setSize(600, 400);

//Create a table model

DefaultTableModel tableModel = **new** DefaultTableModel();

tableModel.addColumn("First Name");

tableModel.addColumn("Last Name");

tableModel.addColumn("Mobile Number");

tableModel.addColumn("Biometric ID");

passengersTable.setModel(tableModel);

// Add components to the frame

JPanel inputPanel = **new** JPanel(**new** GridLayout(4, 2));

inputPanel.add(firstNameLabel);

inputPanel.add(firstNameTextField);

inputPanel.add(lastNameLabel);

inputPanel.add(lastNameTextField);

inputPanel.add(mobileNumberLabel);

inputPanel.add(mobileNumberTextField);

inputPanel.add(submitButton);

inputPanel.add(exitButton);

frame.add(inputPanel, BorderLayout.***NORTH***);

frame.add(tableScrollPane, BorderLayout.***CENTER***);

// Define the action listener for the submit button

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

@Override

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

// Get the input values from the text fields

String firstName = firstNameTextField.getText();

String lastName = lastNameTextField.getText();

String mobileNumber = mobileNumberTextField.getText();

// Validate the input values

**if** (firstName.isEmpty() || lastName.isEmpty() || mobileNumber.isEmpty()) {

JOptionPane.*showMessageDialog*(frame, "Invalid details: Please enter all the details", "Invalid Input", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

**if** (mobileNumber.length() < 10 || !mobileNumber.matches("\\d+")) {

JOptionPane.*showMessageDialog*(frame, "Invalid details: Please enter a valid mobile number", "Invalid Input", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

// Retrieve passenger details from the database

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna", "vasavi")) {

String query = "SELECT \* FROM passengers WHERE F\_Name = ? AND L\_Name = ? AND Mobile\_Number = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, firstName);

statement.setString(2, lastName);

statement.setString(3, mobileNumber);

ResultSet resultSet = statement.executeQuery();

// Clear existing table data

tableModel.setRowCount(0);

// Add retrieved passenger details to the table model

**while** (resultSet.next()) {

String retrievedFirstName = resultSet.getString("F\_Name");

String retrievedLastName = resultSet.getString("L\_Name");

String retrievedMobileNumber = resultSet.getString("Mobile\_Number");

**int** retrievedBiometricId = resultSet.getInt("Biometric\_ID");

Object[] rowData = {retrievedFirstName, retrievedLastName, retrievedMobileNumber, retrievedBiometricId};

tableModel.addRow(rowData);

}

// If no matching records found, display a message

**if** (tableModel.getRowCount() == 0) {

JOptionPane.*showMessageDialog*(frame, "No matching records found", "No Records", JOptionPane.***INFORMATION\_MESSAGE***);

}

} **catch** (SQLException ex) {

ex.printStackTrace();

}

}

});

// Define the action listener for the exit button

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

@Override

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

frame.dispose(); // Close the JFrame

}

});

// Add the scroll pane to the content pane

insertPanel = **new** JPanel(**new** BorderLayout());

/\*insertPanel.add(frame);\*/

/\*insertPanel.add(firstNameLabel);

insertPanel.add(firstNameTextField);

insertPanel.add(lastNameLabel);

insertPanel.add(lasttNameTextField);

insertPanel.add(mobileNumberLabel);

insertPanel.add(mobileNumberTextField);

insertPanel.add(submitButton);\*/

insertPanel.add(tableScrollPane, BorderLayout.***CENTER***);

getContentPane().add(insertPanel, BorderLayout.***CENTER***);

// Show the frame

frame.setVisible(**true**);

revalidate();

repaint();

}

**//Create new User(s) has no options**

**//4 Create new User(s)**

**private** JTextField biometricIdField;

**private** JTextField firstNameField;

**private** JTextField lastNameField;

**private** JTextField emailField;

**private** JTextField addressField;

**private** JTextField mobileNumberField;

**private** **void** createNewUser() {

// Clear the content pane

removePreviousPanel();

// Set the layout manager

createPanel = **new** JPanel();

createPanel.setLayout(**new** GridLayout(7, 2, 5, 5));

// Create text labels

JLabel biometricIdLabel = **new** JLabel("Biometric ID:");

JLabel firstNameLabel = **new** JLabel("First Name:");

JLabel lastNameLabel = **new** JLabel("Last Name:");

JLabel emailLabel = **new** JLabel("Email:");

JLabel addressLabel = **new** JLabel("Address:");

JLabel mobileNumberLabel = **new** JLabel("Mobile Number:");

// Create text fields

biometricIdField = **new** JTextField();

firstNameField = **new** JTextField();

lastNameField = **new** JTextField();

emailField = **new** JTextField();

addressField = **new** JTextField();

mobileNumberField = **new** JTextField();

// Create submit button

JButton submitButton = **new** JButton("Submit");

// Add components to the content pane

createPanel.add(biometricIdLabel);

createPanel.add(biometricIdField);

createPanel.add(firstNameLabel);

createPanel.add(firstNameField);

createPanel.add(lastNameLabel);

createPanel.add(lastNameField);

createPanel.add(emailLabel);

createPanel.add(emailField);

createPanel.add(addressLabel);

createPanel.add(addressField);

createPanel.add(mobileNumberLabel);

createPanel.add(mobileNumberField);

createPanel.add(submitButton);

// Register submit button action listener

submitButton.addActionListener(e -> {

// Get the input values

**int** biometricId;

String firstName, lastName, email, address, mobileNumber;

**try** {

biometricId = Integer.*parseInt*(biometricIdField.getText());

firstName = firstNameField.getText();

lastName = lastNameField.getText();

email = emailField.getText();

address = addressField.getText();

mobileNumber = mobileNumberField.getText();

} **catch** (NumberFormatException ex) {

// Show warning message if any field is empty

JOptionPane.*showMessageDialog*(createPanel, "Fill all the details,(Biometric ID should consist of 0's and 1's) and Please try again!!!", "Warning", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

// Call a method to add the user to the database

addUserToDatabase(biometricId, firstName, lastName, email, address, mobileNumber);

});

getContentPane().add(createPanel, BorderLayout.***CENTER***);

// Repaint the content pane

revalidate();

repaint();

}

**private** **void** addUserToDatabase(**int** biometricId, String firstName, String lastName, String email, String address, String mobileNumber) {

// Check if all details are entered

**if** (biometricId <= 0 || firstName.isEmpty() || lastName.isEmpty() || email.isEmpty() || address.isEmpty() || mobileNumber.isEmpty()) {

JOptionPane.*showMessageDialog*(**this**, "Fill all the details and Please try again!!!", "Warning", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

// Establish a database connection

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna" , "vasavi")) {

// Create a prepared statement

String query = "INSERT INTO passengers (Biometric\_ID, F\_Name, L\_Name, e\_mail, Address, Mobile\_Number) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, biometricId);

statement.setString(2, firstName);

statement.setString(3, lastName);

statement.setString(4, email);

statement.setString(5, address);

statement.setString(6, mobileNumber);

// Execute the query

statement.executeUpdate();

// Show success message

JOptionPane.*showMessageDialog*(**this**, "User added successfully!", "Success", JOptionPane.***INFORMATION\_MESSAGE***);

// Clear the text fields

biometricIdField.setText("");

firstNameField.setText("");

lastNameField.setText("");

emailField.setText("");

addressField.setText("");

mobileNumberField.setText("");

// Close the statement

statement.close();

} **catch** (SQLException ex) {

ex.printStackTrace();

// Show error message

JOptionPane.*showMessageDialog*(**this**, "An error occurred while adding the user.", "Error", JOptionPane.***ERROR\_MESSAGE***);

}

}

**private** **boolean** isValidReservationId(**int** reservationId) {

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna", "vasavi")) {

String query = "SELECT \* FROM reservations WHERE Reservations\_ID = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, reservationId);

ResultSet resultSet = statement.executeQuery();

**return** resultSet.next(); // Returns true if a row with the reservation ID exists in the table

} **catch** (SQLException ex) {

ex.printStackTrace();

**return** **false**;

}

}

**private** **boolean** isValidTrainFare(**double** trainFare) {

**double**[] validFares = {75.0, 110.0, 55.5, 87.0, 160.5};

**for** (**double** fare : validFares) {

**if** (fare == trainFare) {

**return** **true**; // Train fare is valid

}

}

**return** **false**; // Train fare is invalid

}

**private** **boolean** isValidBiometricId(**int** biometricId) {

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna", "vasavi")) {

String query = "SELECT COUNT(\*) FROM passengers WHERE Biometric\_ID = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, biometricId);

ResultSet resultSet = statement.executeQuery();

**if** (resultSet.next()) {

**int** count = resultSet.getInt(1);

**return** count > 0; // Return true if the count is greater than 0 (biometric ID exists)

}

} **catch** (SQLException ex) {

ex.printStackTrace();

}

**return** **false**; // Biometric ID is invalid (error occurred while checking or no rows found)

}

**private** **boolean** isValidTrainId(**int** trainId) {

**try** (Connection connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:XE", "krishna", "vasavi")) {

String query = "SELECT COUNT(\*) FROM trains WHERE Train\_ID = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, trainId);

ResultSet resultSet = statement.executeQuery();

**if** (resultSet.next()) {

**int** count = resultSet.getInt(1);

**return** count > 0; // Return true if the count is greater than 0 (train ID exists)

}

} **catch** (SQLException ex) {

ex.printStackTrace();

}

**return** **false**; // Train ID is invalid (error occurred while checking or no rows found)

}

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

SwingUtilities.*invokeLater*(ReservationSystemGUI::**new**);

}

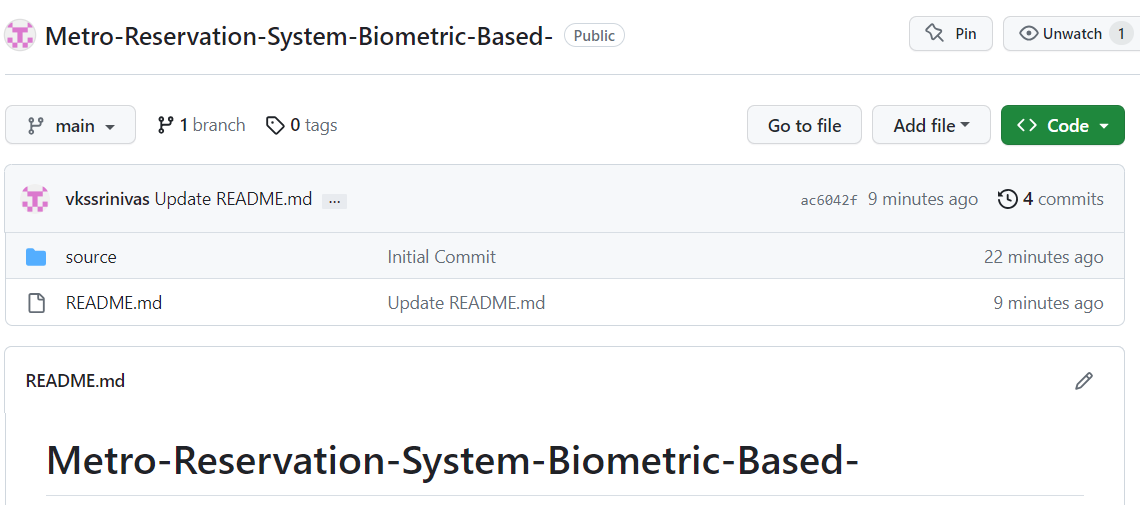
}

**GitHub Link and Folder Structure**

**Link:**

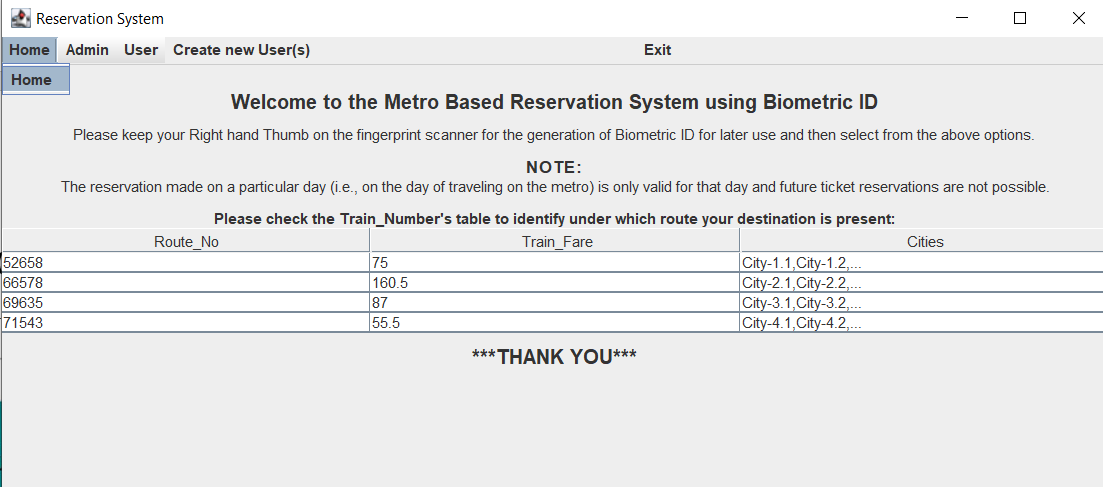
https://github.com/vkssrinivas/Metro-Reservation-System-Biometric-Based-

**Folder Structure:**

****

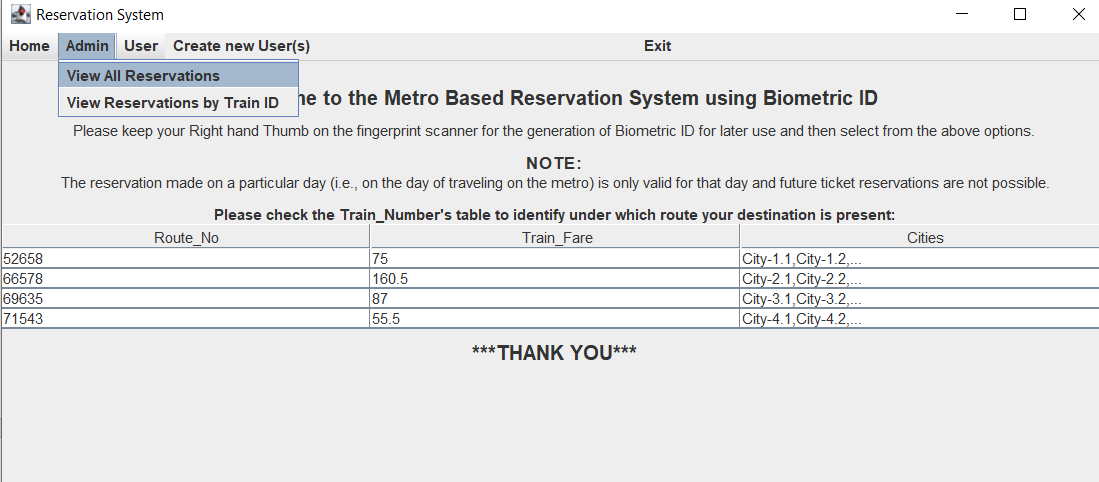
**TESTING**

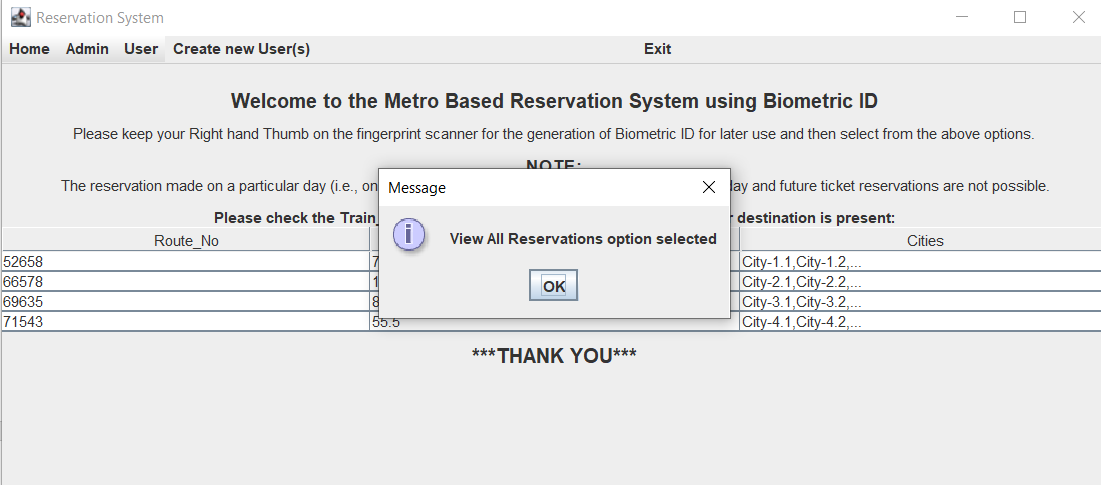
METRO RESERVATION SYSTEM INTRODUCTION/HOME PAGE:

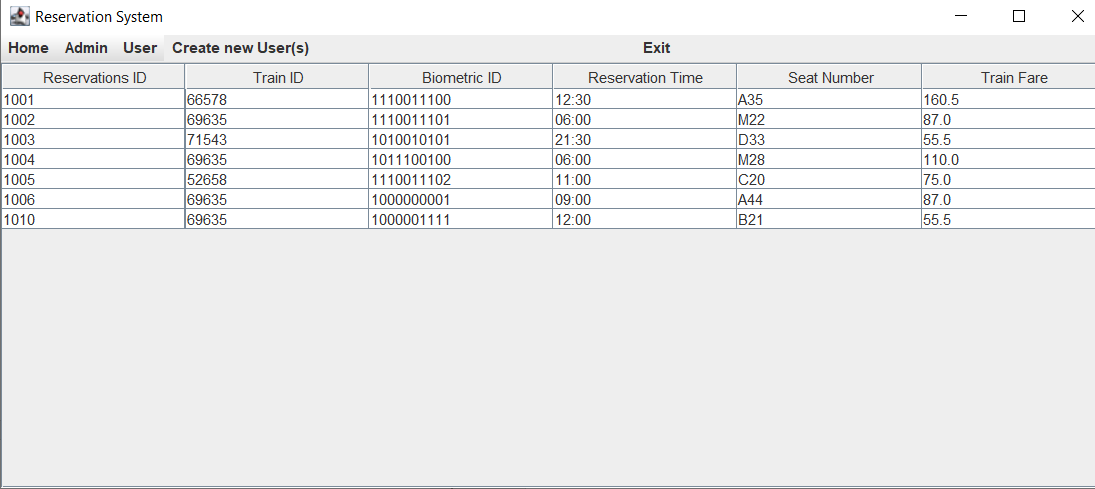


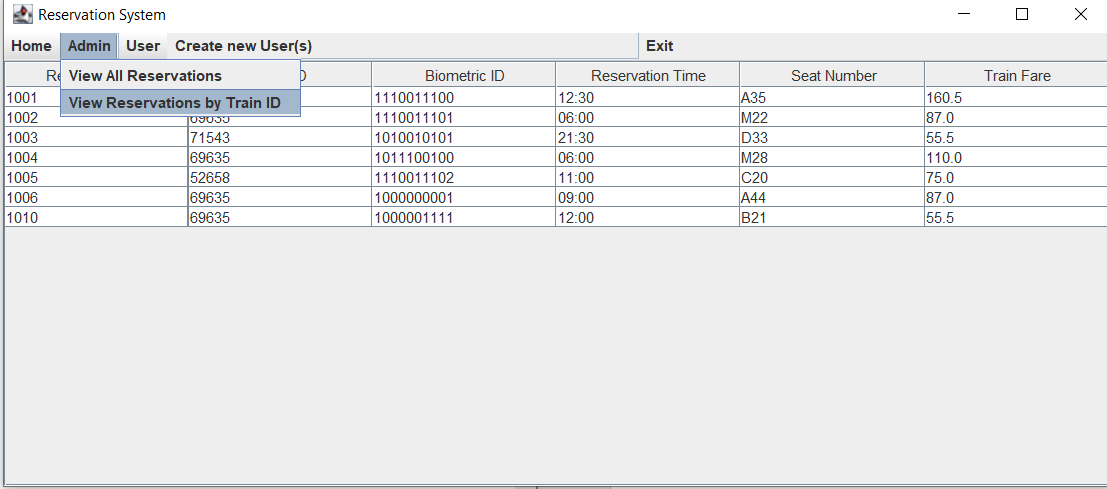
1. ADMIN PAGE:

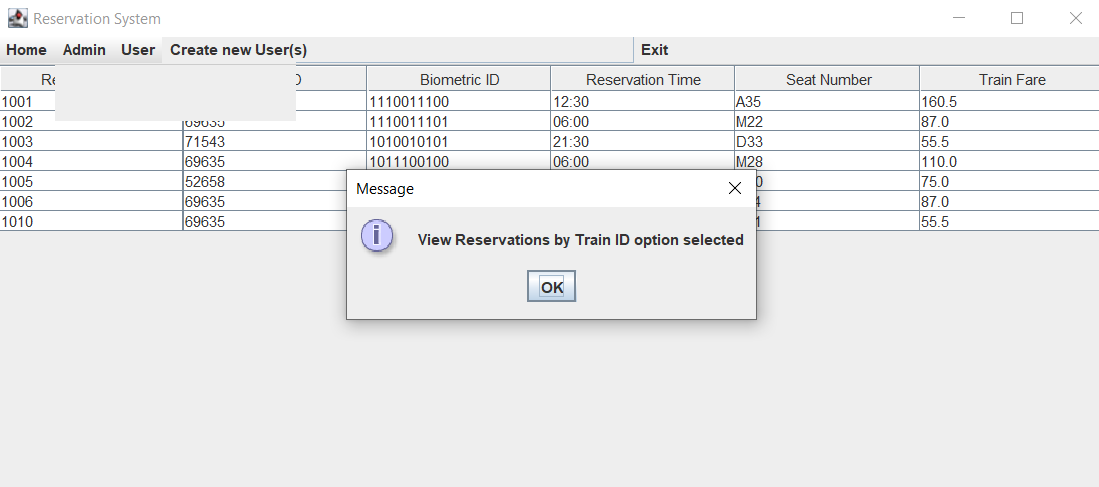
1.1 VIEW ALL RESERVATIONS

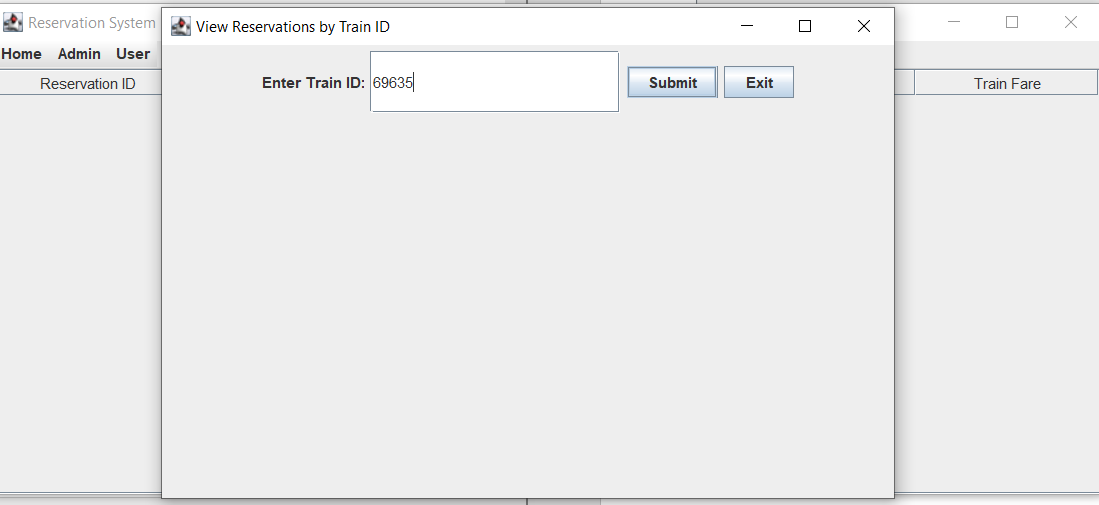


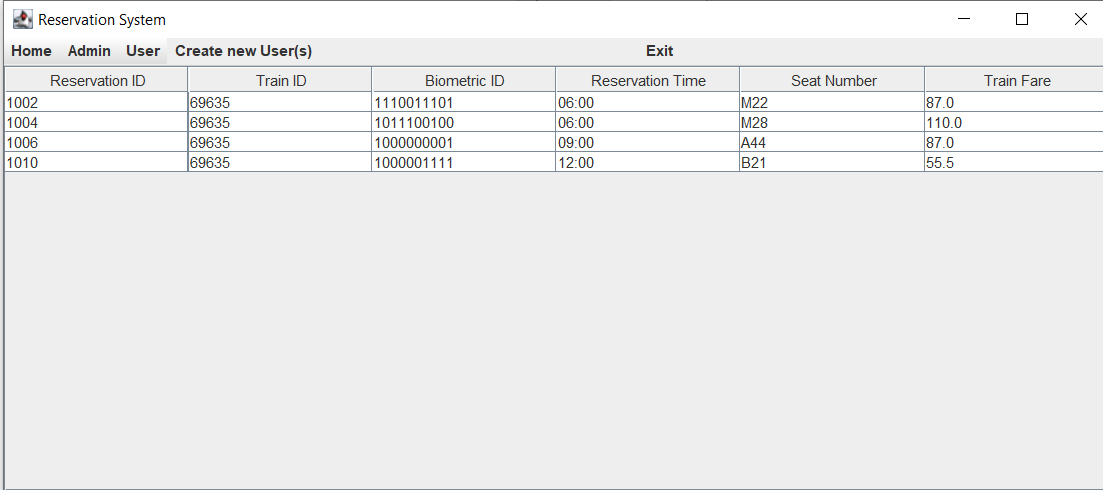


1.2 VIEW RESERVATIONS BY TRAIN ID

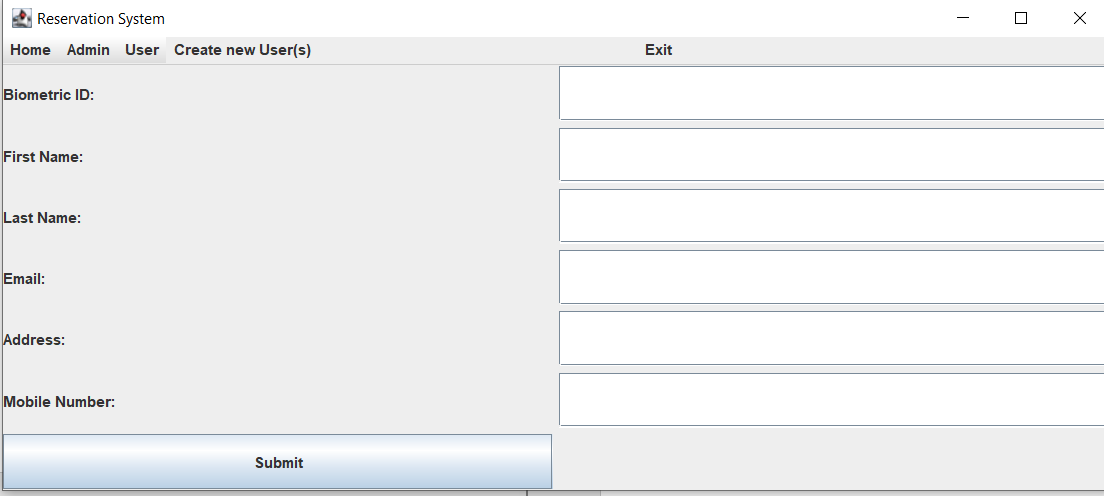




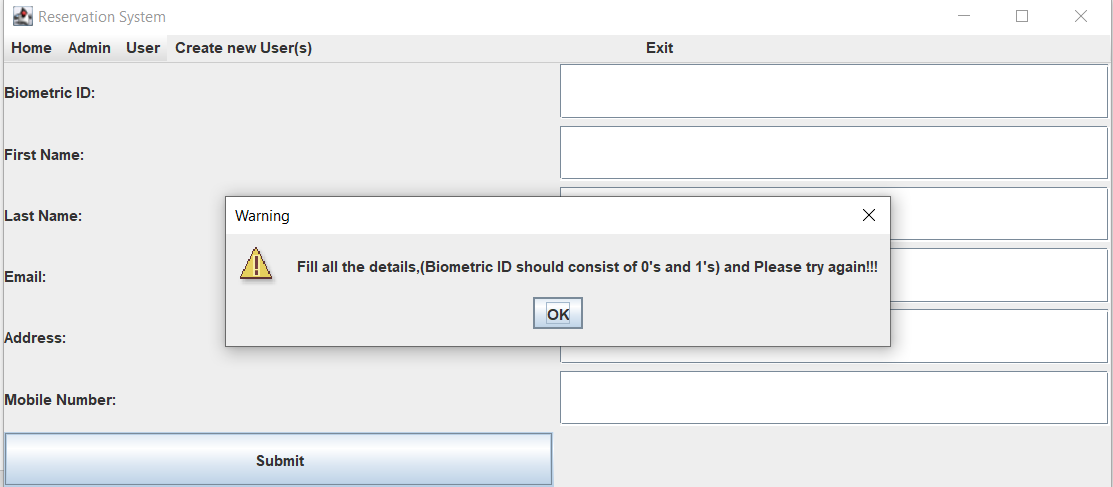




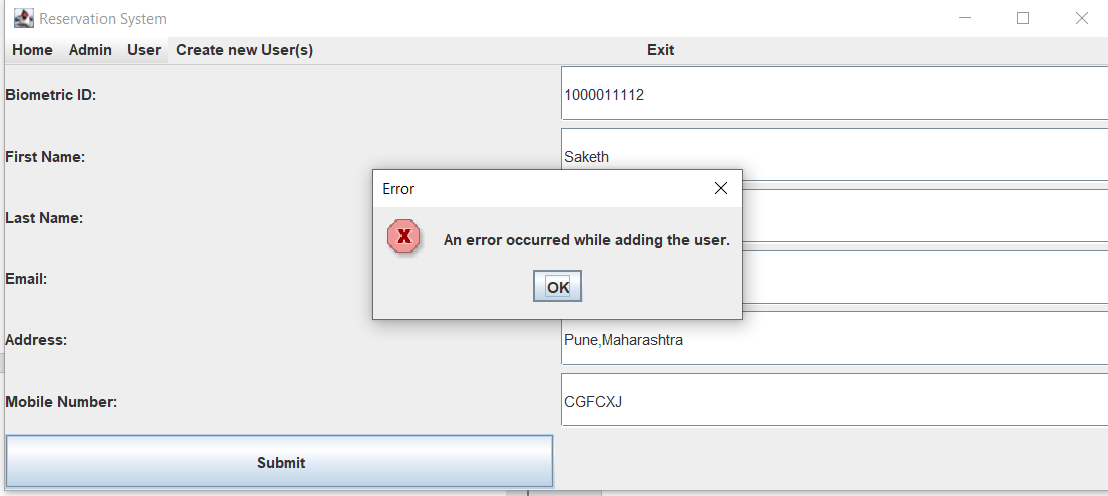
3.CREATE NEW USER(S) PAGE:



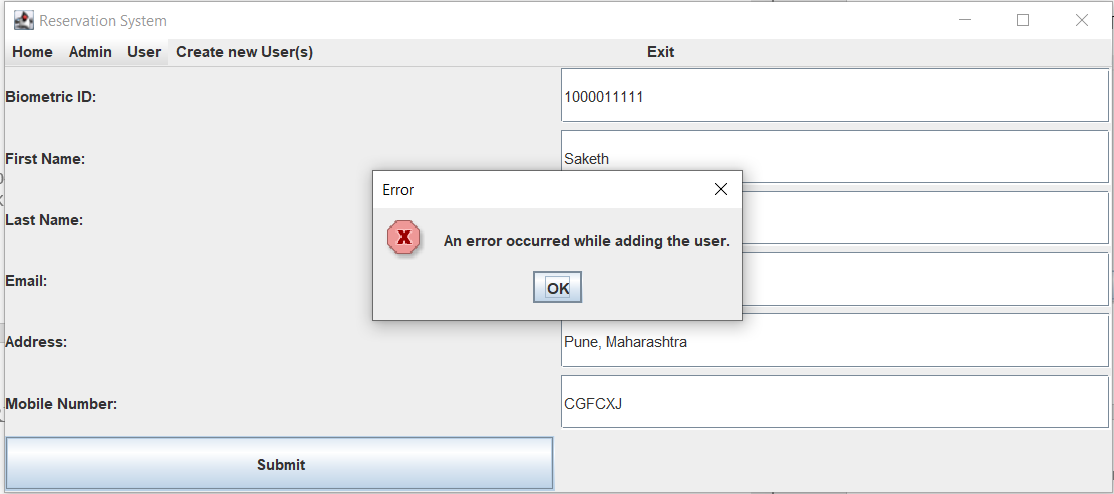
SUBMIT WITHOUT INFORMATION



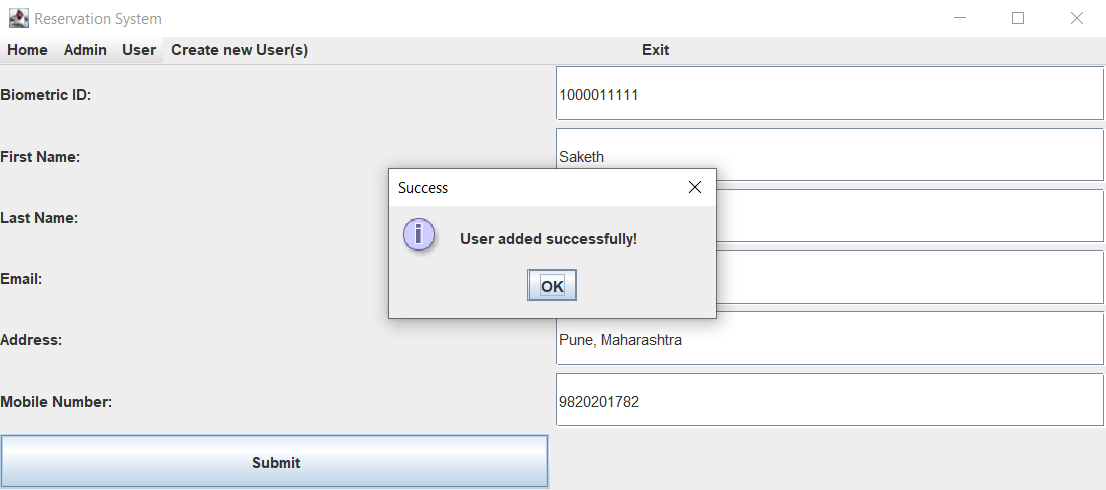
INCORRECT BIOMETRIC ID AND MOBILE NUMBER



INCORRECT MOBILE NUMBER

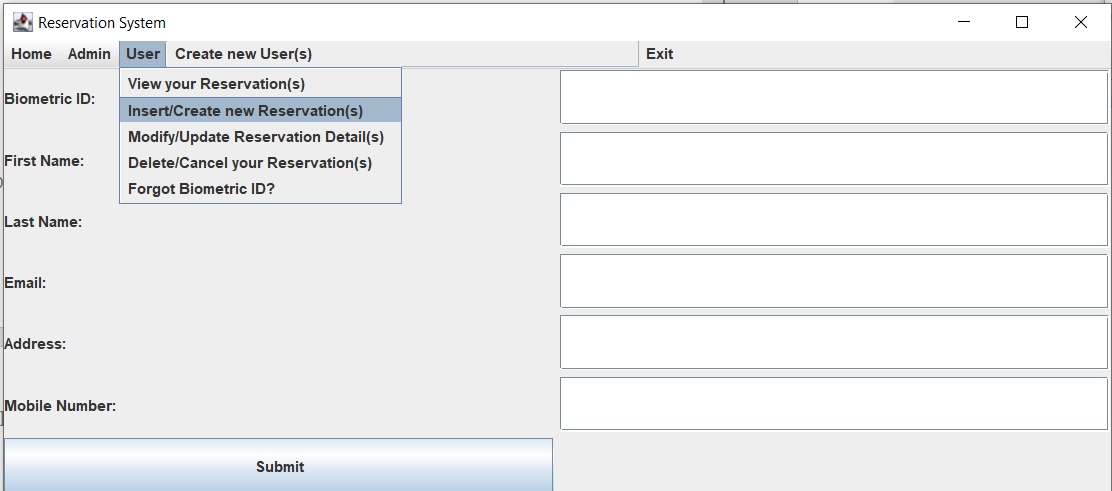


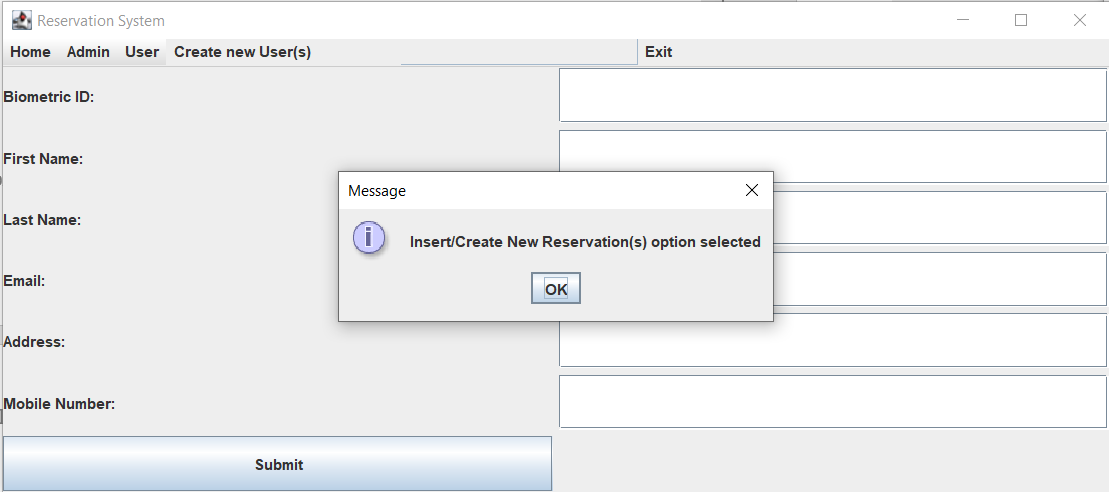
NO ERROR



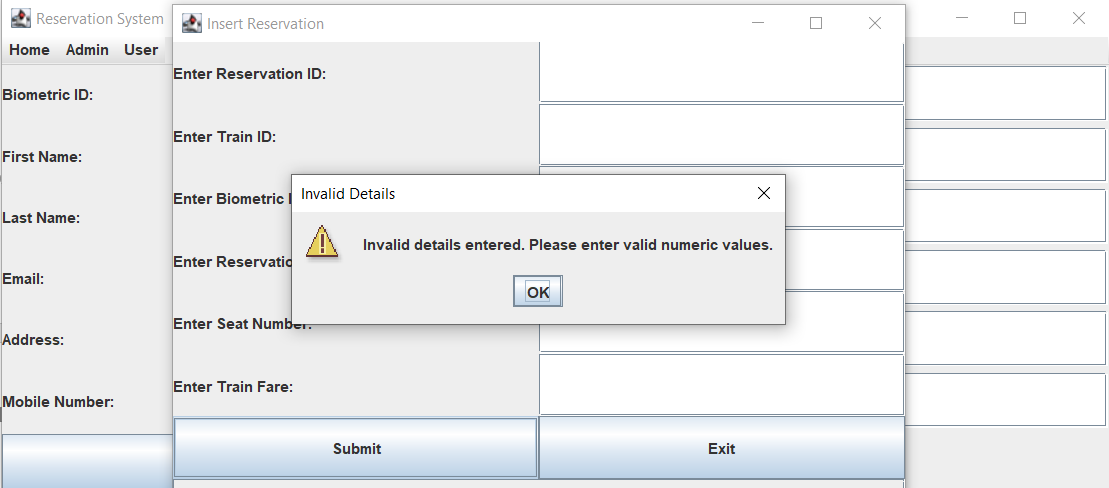
4. USER PAGE

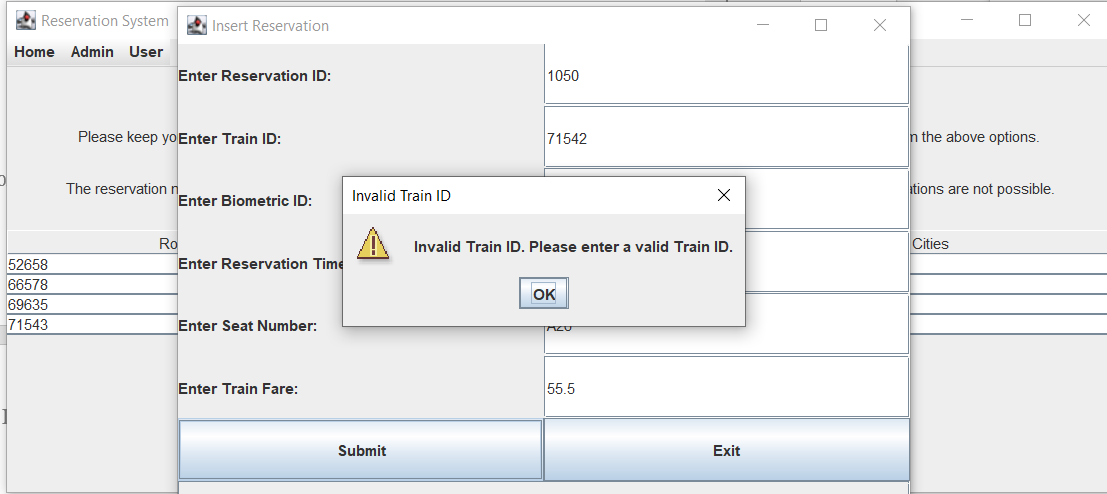
4.1 INSERT/CREATE NEW RESERVATION



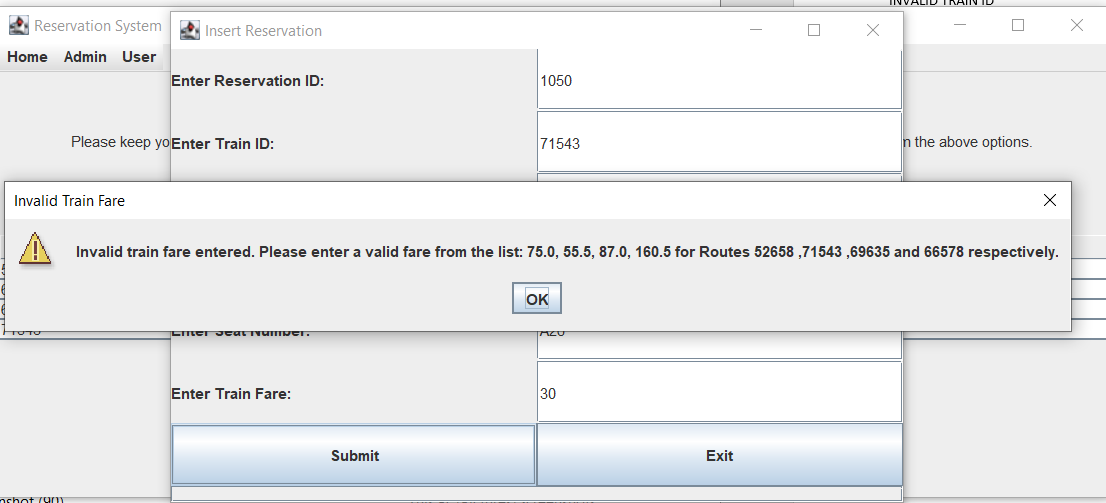


SUBMIT WITHOUT INFORMATION

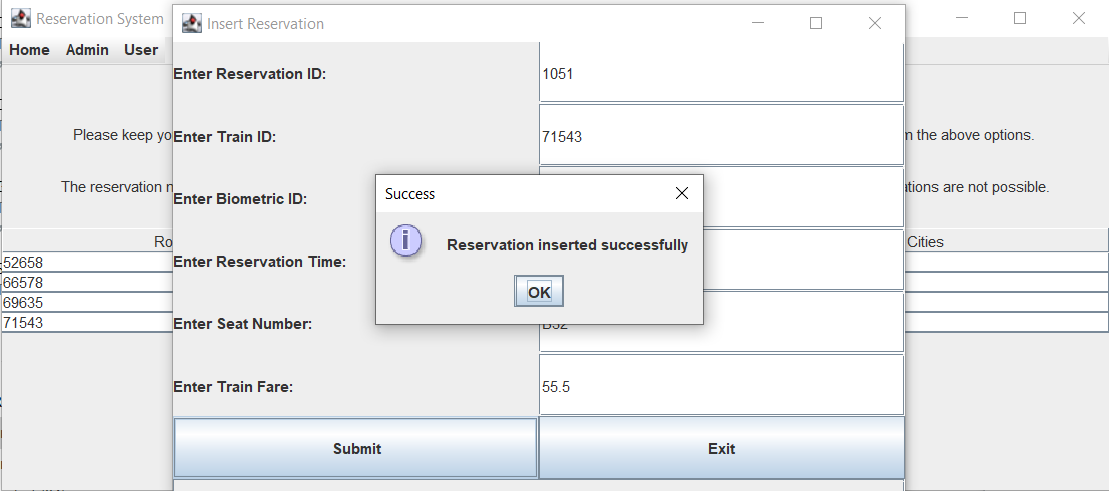
INVALID TRAIN ID



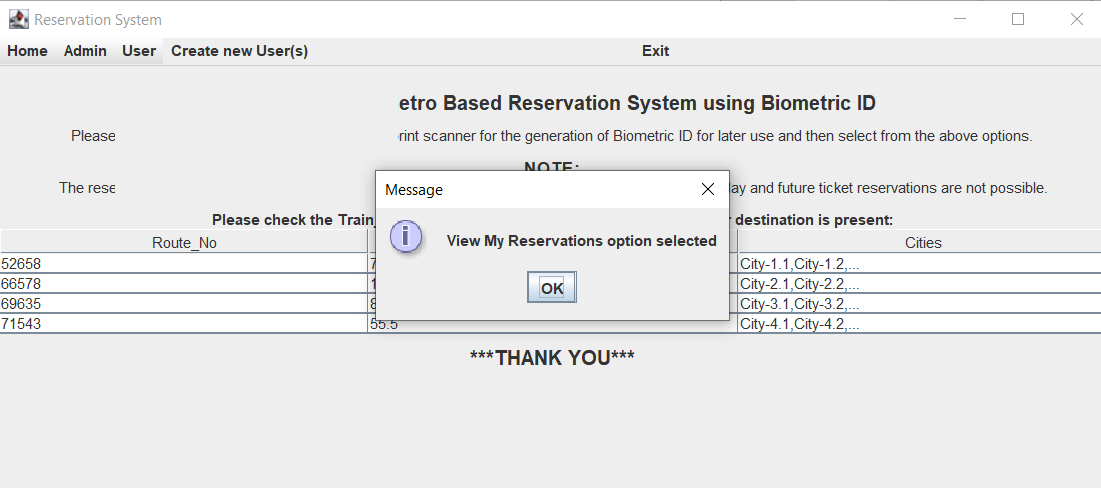
INVALID TRAIN FARE CORRESPONDING TO TRAIN ID



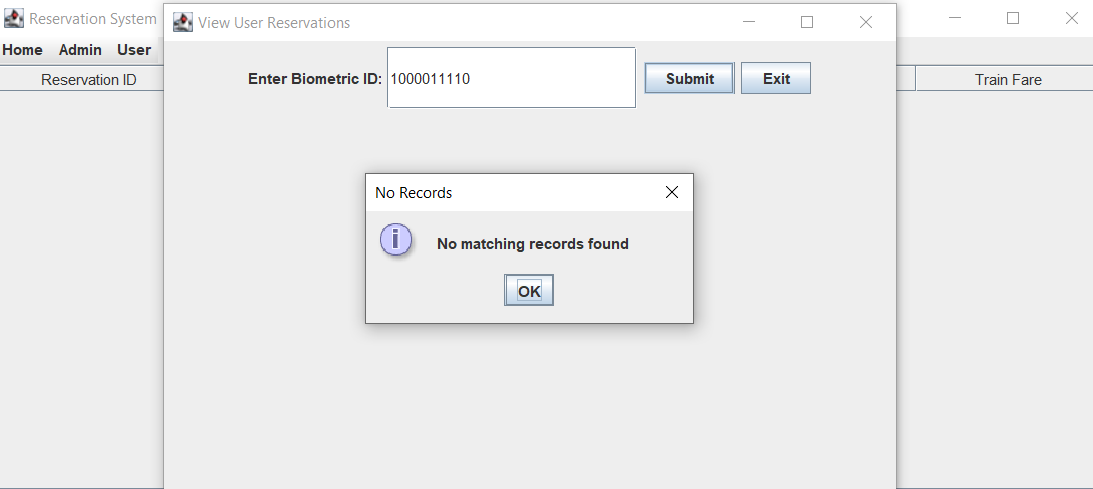
SUCCESFUL RESERVATION



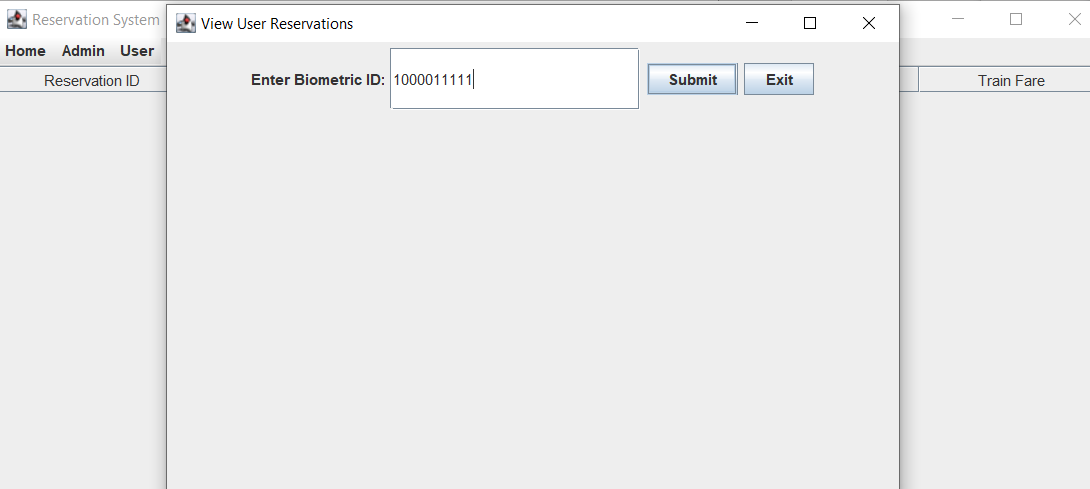
4.2 VIEW YOUR RESERVATIONS

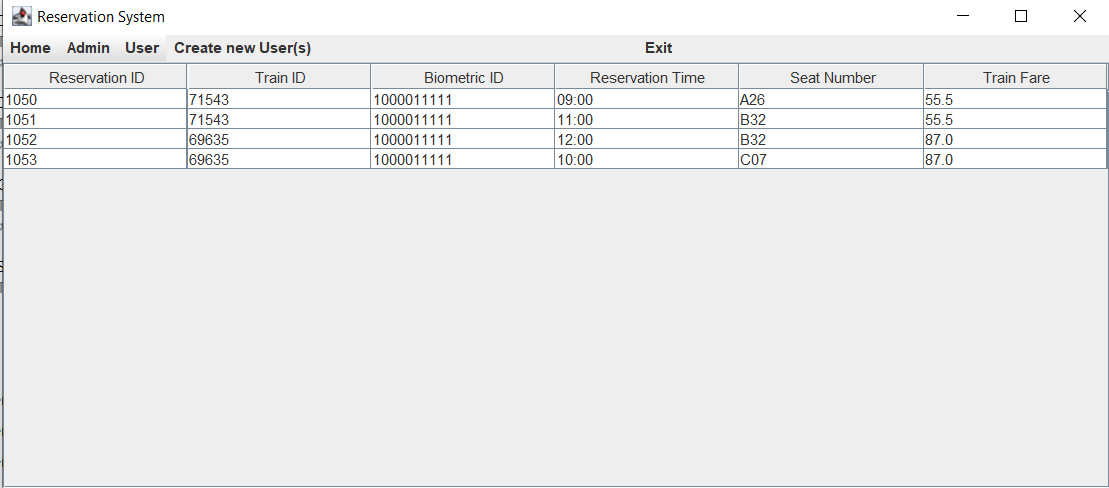


INCORRECT BIOMETRIC ID

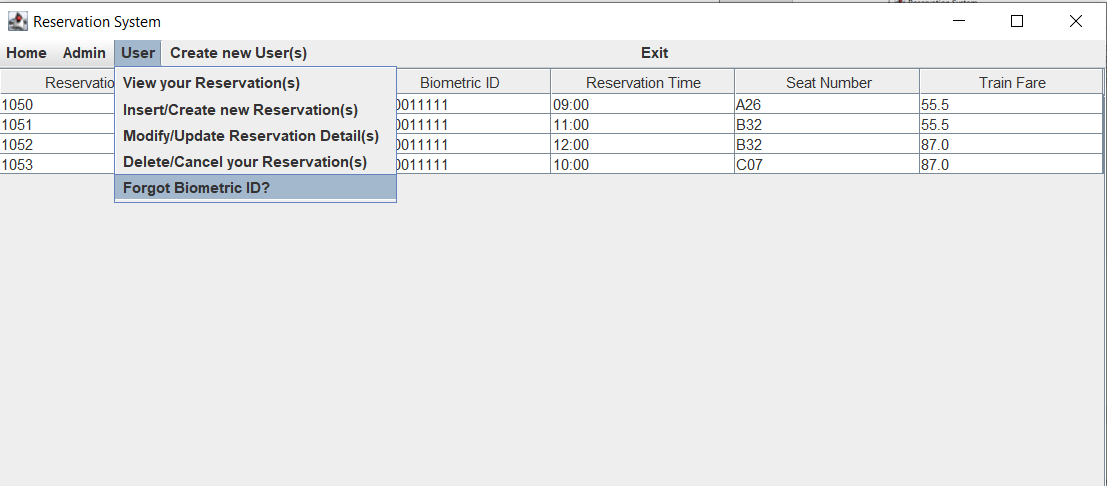


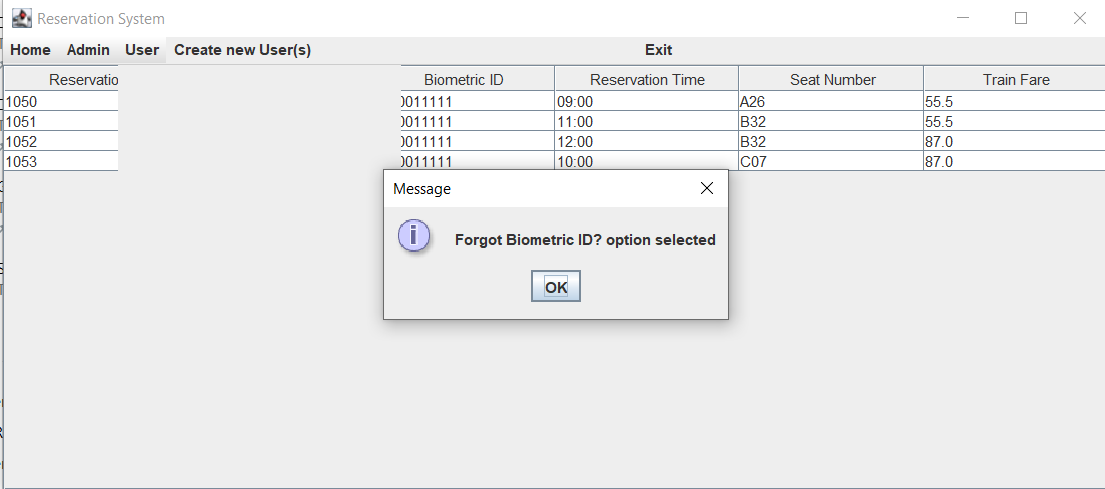
CORRECT BIOMETRIC ID



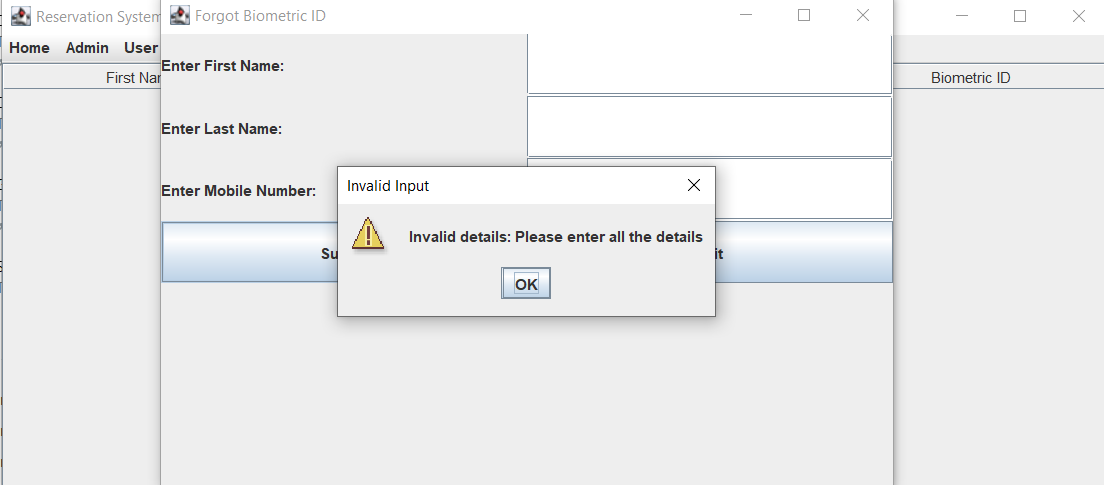


4.3 FORGOT BIOMETRIC ID ?

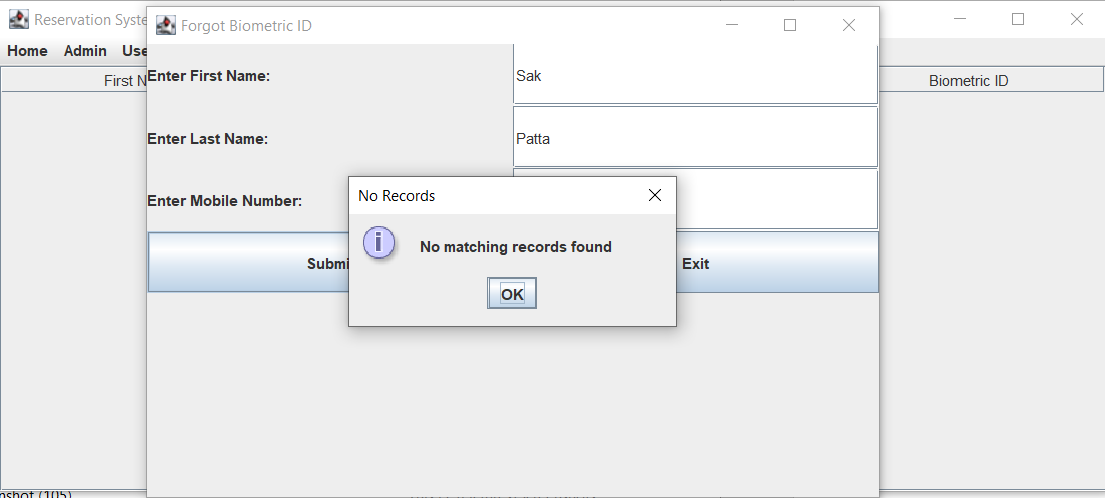




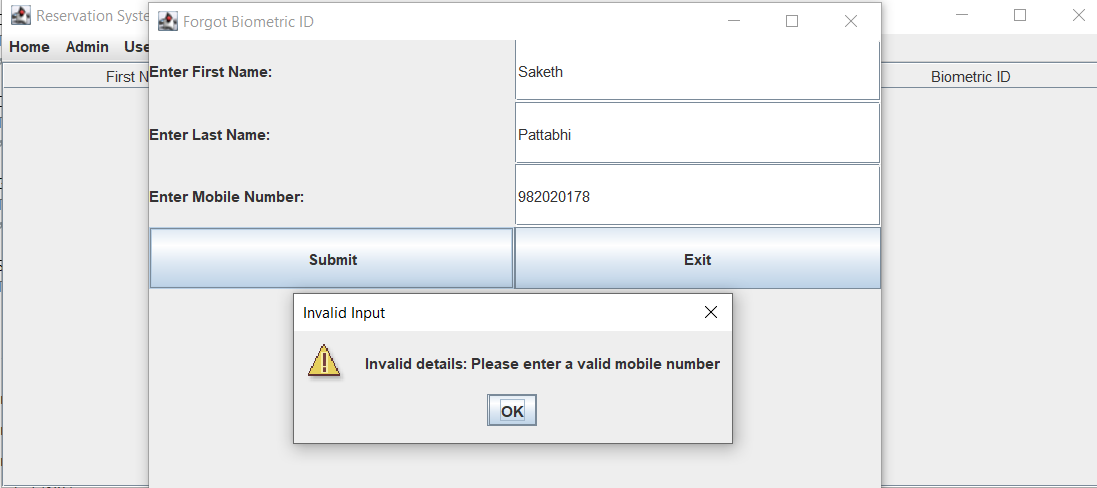
SUBMIT WITHOUT INFORMATION



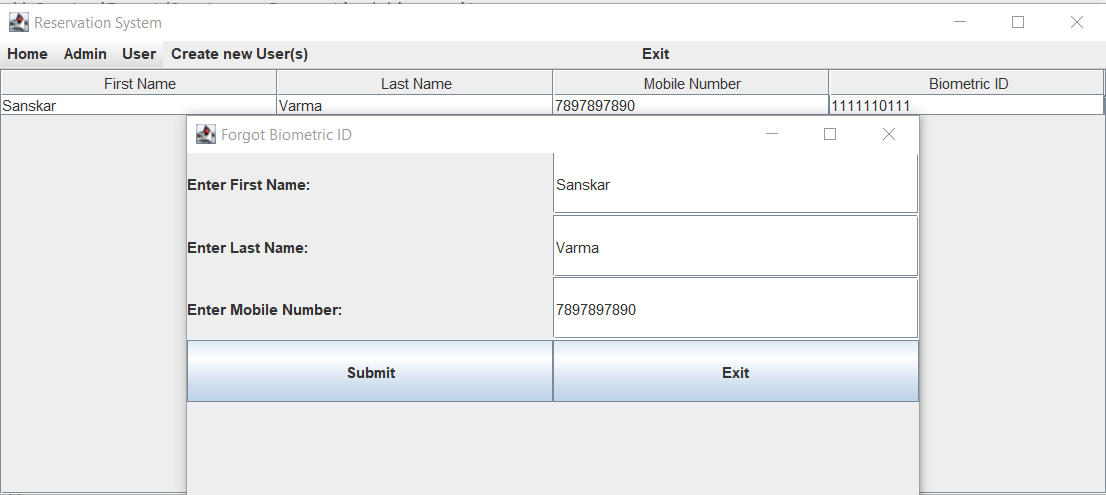
INVALID FIRST NAME/LAST NAME



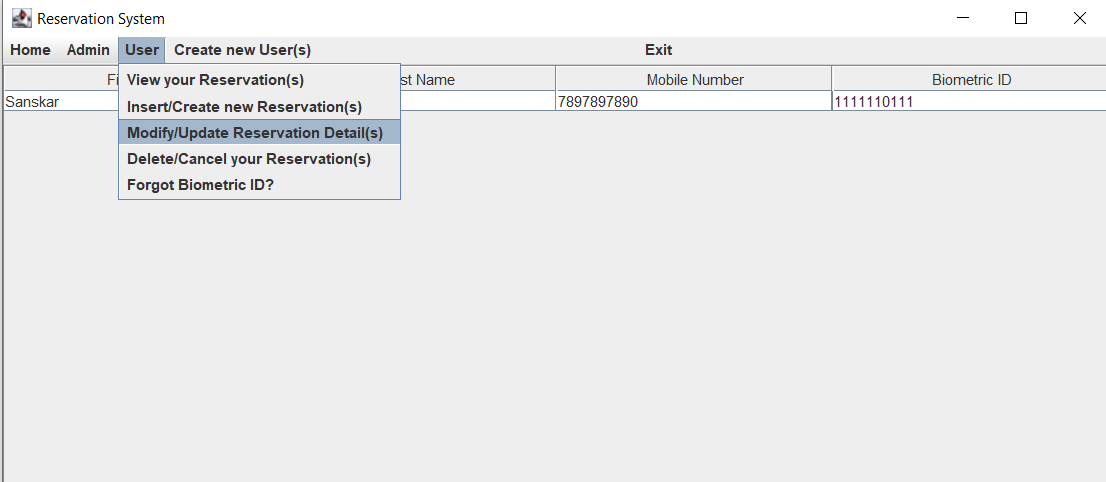
INVALID MOBILE NUMBER

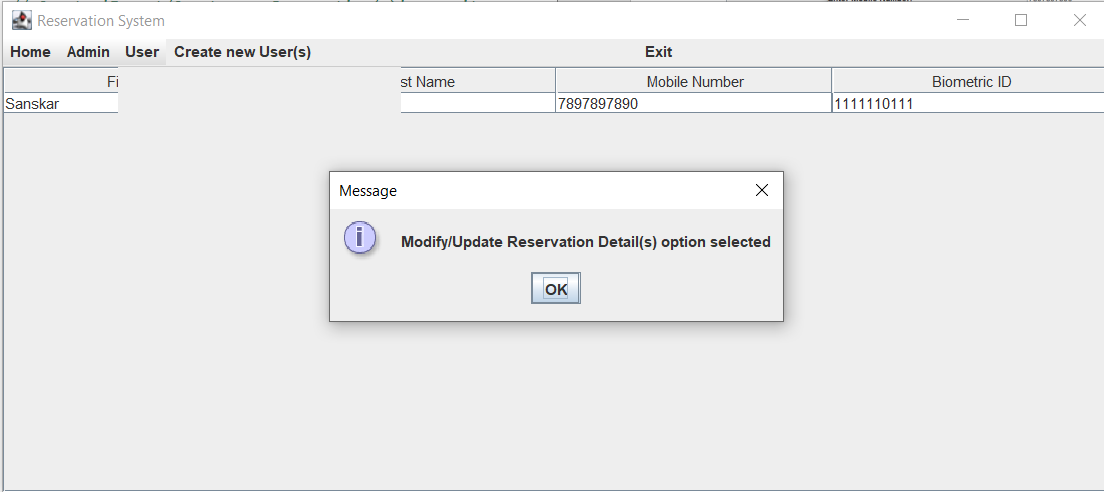


VALID DETAILS

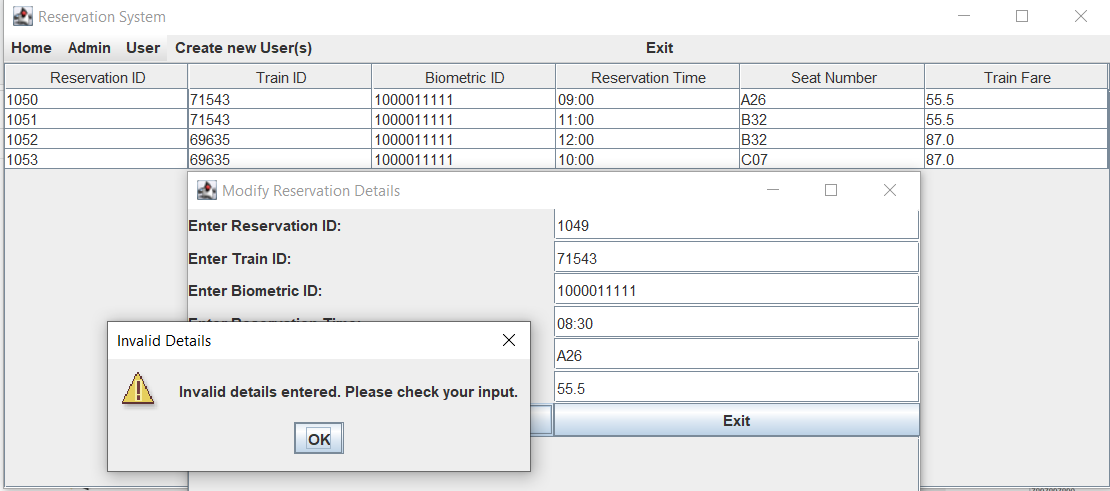


4.4 MODIFY/UPDATE RESERVATION DETAIL(S)

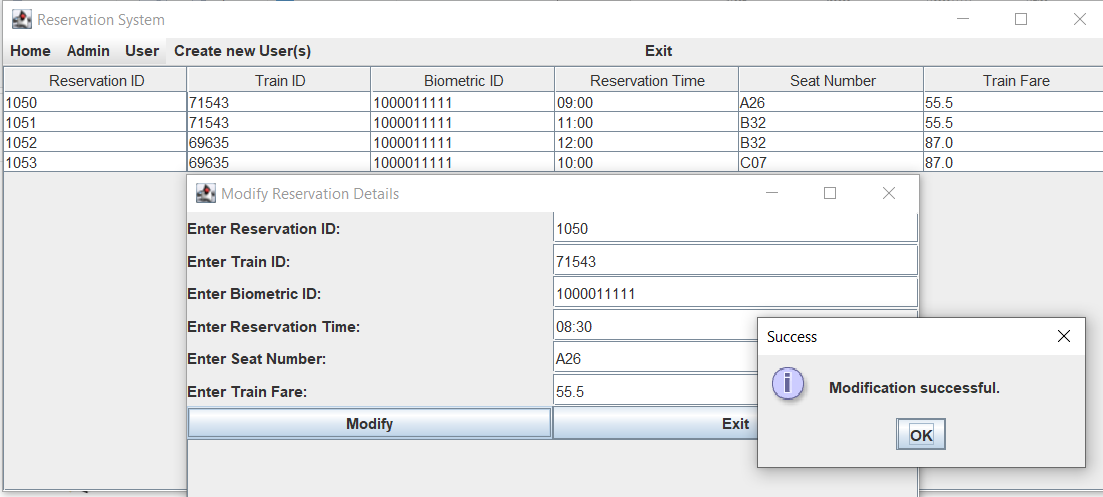




INVALID/INCORRECT DETAILS

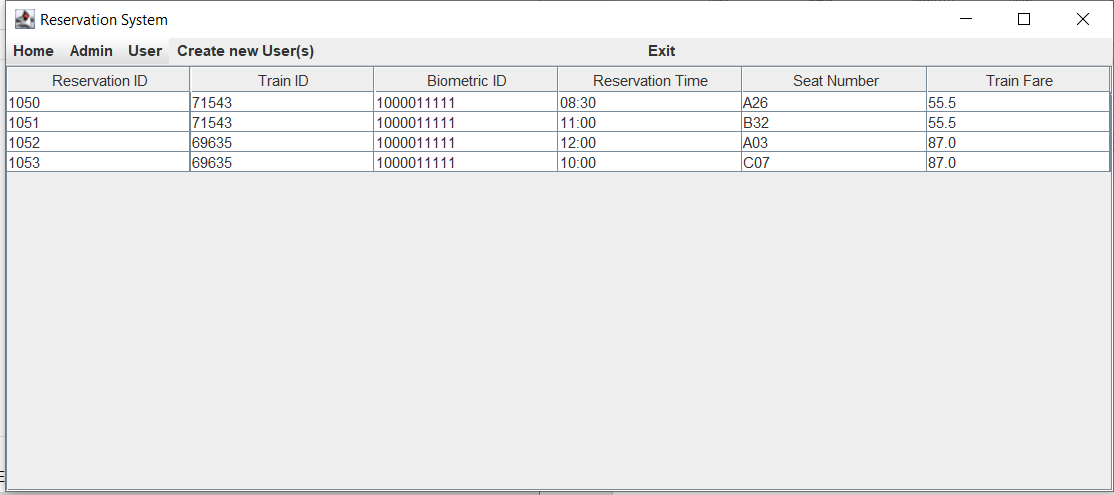


VALID DETAILS

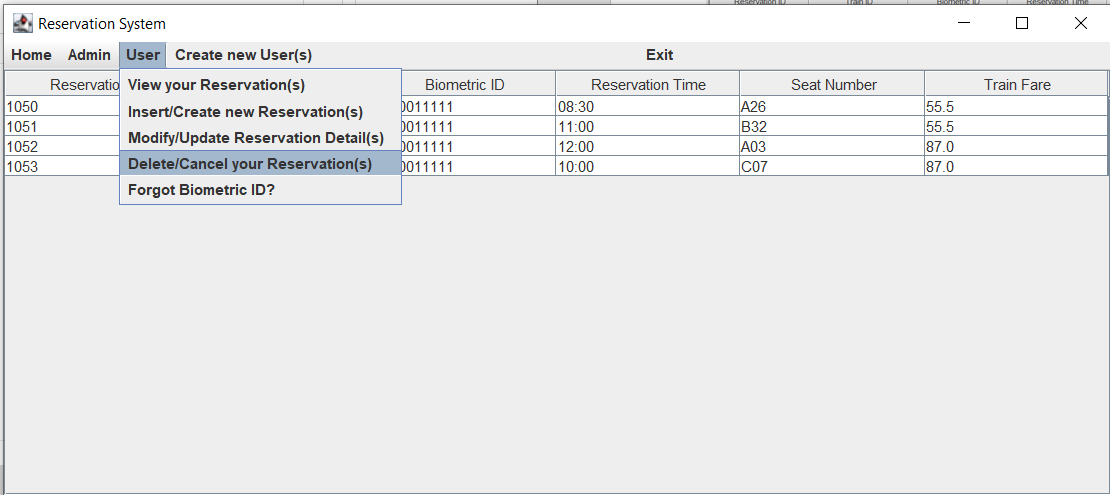


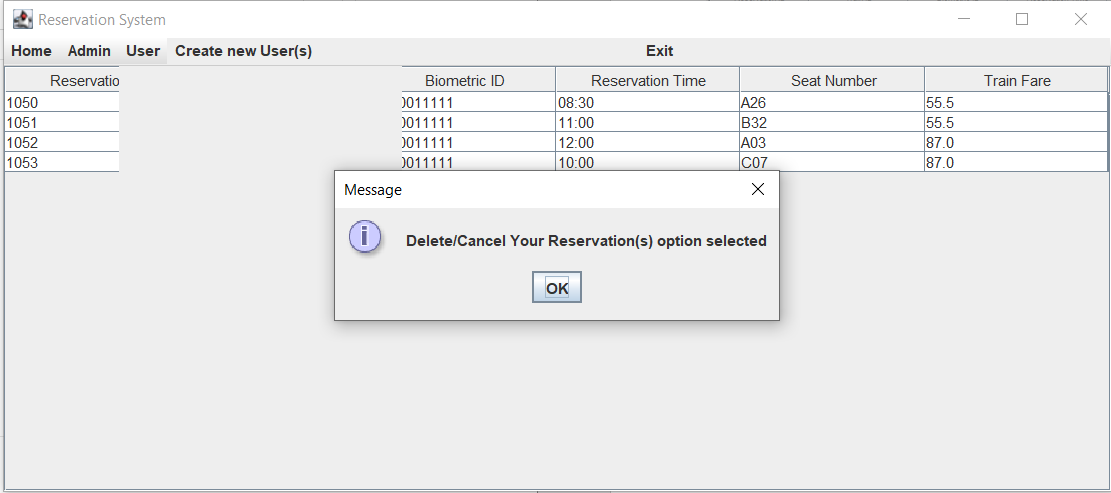
AFTER UPDATION

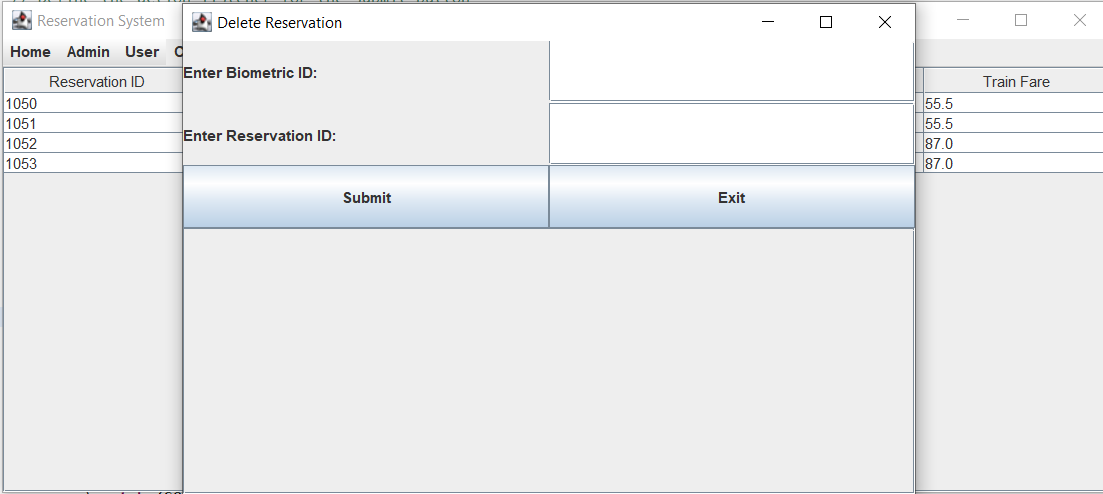
(VIEW YOUR RESERVATION(S))



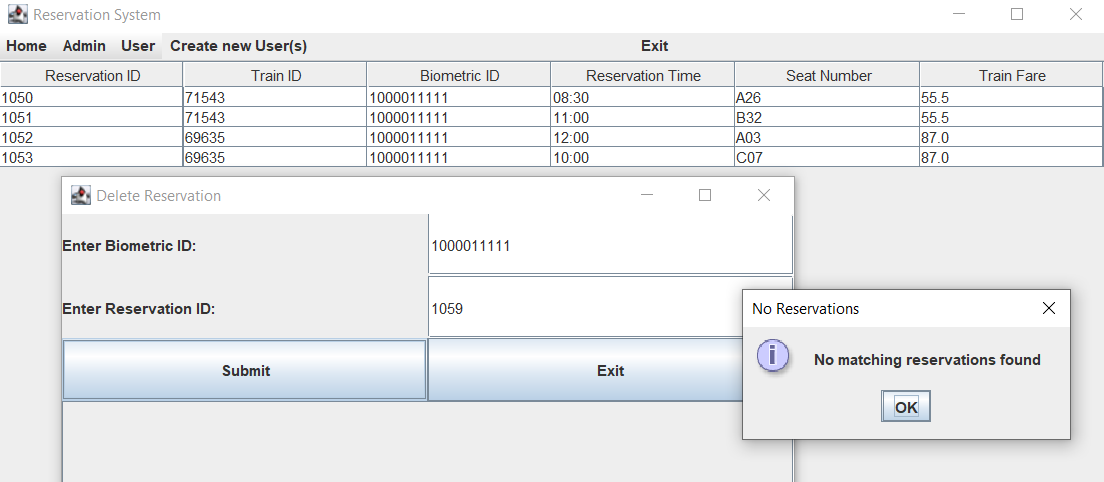
4.5 DELETE/CANCEL YOUR RESERVATION



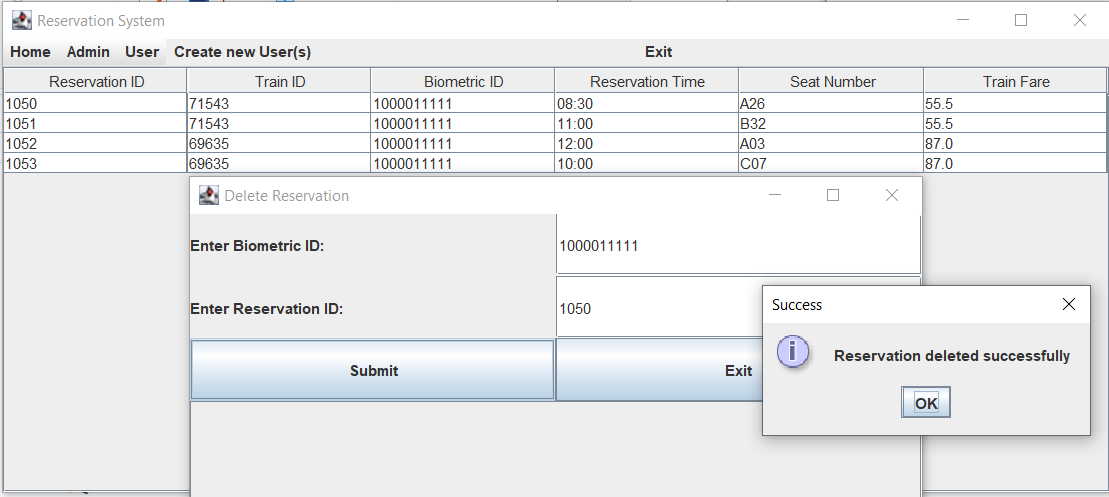




INCORRECT/INVALID DETAILS

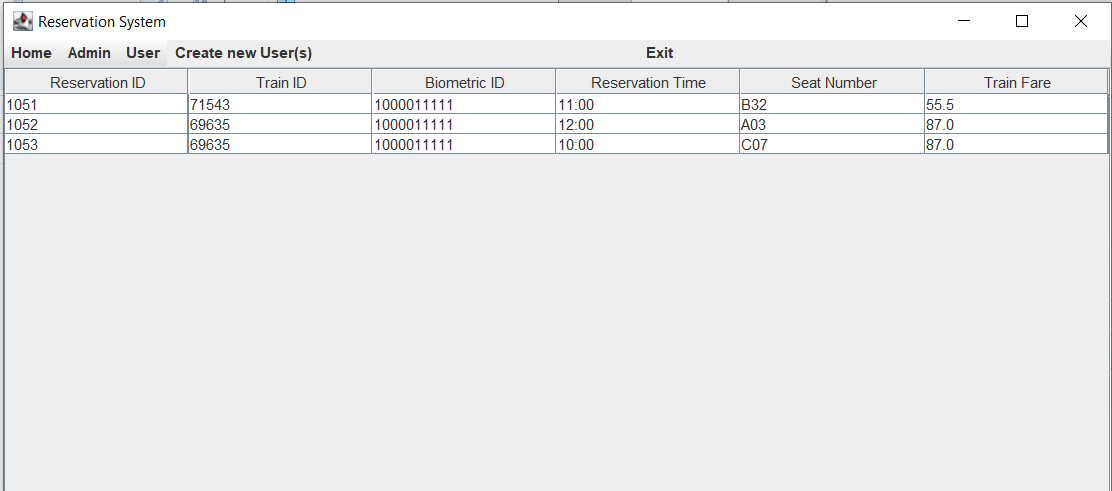


VALID DETAILS



AFTER DELETION

(VIEW YOUR RESERVATION(S))



**RESULTS**

I have successfully completed the mini-project **“METRO RESERVATION SYSTEM- (BIOMETRIC BASED)” .**

**DISCUSSION AND FUTURE WORK**

* This project contains the basic interaction of giving information by the users for booking a metro ticket. It has a very basic user interface.
* Future scope would be to make the UI more appealing by using graphics. We can also add more trains, more metro stations, more junction-metro stations i.e; a metro station which is common to two or more lines. Better ways to display tables, generate biometric\_id.
* More tables/functionality can be added for the Admin side, so that he can understand for a metro station which train arrives at what time, how many employees are working in the station, how many shops/vendors are present for lease, etc.
* We can also think of including a feedback system to allow the users to leave their valuable feedback after using this app. Making this feedback to be publicly viewable, would attract many more users to use this app.

**REFERENCES**

* https://docs.oracle.com/javase/7/docs/api/
* [https://www.javatpoint.com/java-swing](http://www.javatpoint.com/java-swing)
* https://stackoverflow.com/