

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor in
Technology Degree in Information Technology

By
RAUSHAN KUMAR
Registration No.: 39120092



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

JEPPIAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI-600119, TAMILNADU

MARCH 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **RAUSHAN KUMAR (Reg. No: 39120092)** who carried out the project entitled "**Railway Reservation System**" under my supervision from January 2022 to March 2022

Internal Guide

Mr Jeberson Ratna Raj

DECLARATION:

I, **RAUSHAN KUMAR** hereby declare that the project entitled “**Railway reservation System**” completed by me under the guidance of **Mr Jeberson Ratna Raj** is submitted in partial fulfilment of the requirements for the award of Bachelor in Technology Degree in Information Technology.

RAUSHAN KUMAR

DATE: 10/03/2022

DIGITAL SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** at **Sathyabama Institute** for their encouragement and guidance in the completion of this project.

I would like to express my sincere gratitude to my Project Guide **Mr Jeberson Ratna Raj**, for her valuable guidance, suggestions and constant encouragement which paved the way for the successful completion of my project work.

I would also like to express my humble appreciation towards the staff over at **HCL** & especially our trainer, **Sankaran Vaidyanathan** who held a guiding hand at every turn throughout the completion of my Machine Learning course.

I wish to express my thanks to all teaching and non-teaching staff members of the **Department of Information Technology** who were helpful in many ways throughout the duration of the project.

TRAINING CERTIFICATE

PREFACE

I have made this report file on the topic “**Railway Reservation System**”; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic. My efforts and wholehearted co-operation of each and every one has ended on a successful note. I express my sincere gratitude to my guide Dr Jeberson and Sankaran Vaidyanathan who assisted me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

ABSTRACT

The railway reservation system facilitates the passengers to enquiry about the trains available on the basis of source and destination, booking and cancellation of tickets, enquiry about the status of the booked ticket, etc. The aim of case study is to design and develop a data base maintaining records of different trains, train status and passengers. This project contains introduction to the railways reservation system. It is the computerized system of reserving the seats of train seats in advance. It is mainly used for a long route. Online reservation has made the process for the reservation of seats very much easier than ever before

INTRODUCTION

In our country India, there are number of counters for the reservation of the seats and one can easily make reservations and get tickets. Railway reservation system, has described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization it better utilization of resources . Administrator of the project, with the help of a password, can enter new train record, display all train records, modify train records and delete train records. The record of train includes its number, name, source, destination, and days on which it is available, whereas record of train status includes dates for which tickets can be booked, total number of seats available, and number of seats already booked.

OBJECTIVE

Railway ticket reservation will help the passengers to book their tickets for their journey, search train between two stations and get details of particular train schedules including their fare details. To use this system, the user have to approach the Admin. The main objective of this login module is to keep each passengers safe which will not allowed to make fraud and make their transactions secure. After making registration passengers have to enter the source and destination station name and the system will able to provide the list of all available trains for particular destination and their arrival and departure time and date. passengers will be provided with choice to select the seat numbers as per their location, passengers will be provided with total fare charges and mode of payment which they want to select for final payment. And mode of payment which they want to select for making final payment.

The main objective of the Railway Reservation System is to manage the details of Train, Booking, Payment, Seat, Ticket. It manages all the information about Train, Customer, Ticket, and Train. The project is totally built at administrative end and thus only the administrator is guaranteed the access.

How is DBMS used in railway reservation system?

In the rail route reservation framework, the information base is needed to store the record or information of ticket appointments, status about train's appearance. Additionally, if trains get late, individuals become acquainted with it through the information base update.

Aim And Scope of present investigation

AIM

Reservation clerk is a person to create and cancel reservation by entering login password. The system cant only reserve and cancel ticket but can also manage the train details train timing and can change according to the requirement. The main purpose of this system is:

- i) Creating reservation
- ii) Cancel reservation
- iii) View reservation status
- iv) View train schedule
- v) Update train schedule
- vi) Update reservation details

SCOPE

The scope of this system in creating reservation is that, from any railway station we can create reservation, which is updated automatically in all the stations. Hence there is no confusion to reservation clerk. This can be possible by maintaining global databse. Clerks present at different station can access this global databaseand clerk can easily understand the remaining reservation seats.

Advantage:

The system is so secured and clerk and manager can utilize it. Nobody can able to access the information without admin permission because of providing login facility to the system.

Software and Hardware Requirement:

SOFTWARE REQUIREMENT

- ☐ Windows/Mac Based System
- ☐ Necessary Java Libraries
- ☐ XAMPP
- ☐ phpMyAdmin
- ☐ NetBeans
- ☐ Eclipse (part 2)
- ☐ JDBC connector to mySQL
- ☐ Apache server

Windows/Mac System

Windows or Mac operating based system for personal computers or any other internet of things devices.

ALGORITHM USED

- i) I used java NetBeans for jframe work and customized that front end it according to the requirements.
- ii) Then connected database of train route, reservation and customer details using XAMPP which include apache server and JDBC connector to SQL.
- iii) Then customized the source code for admin page to add train details and passenger details for their reservation to the data base.
- iv) Steps we followed:
 - We need to install java net beans in our system and set it up.
 - Once done, we select our best algorithm for continuation of this project.
 - Then we created a file for login page where admin will login with their user name and password.
 - Then we will move to our main page where admin will get the option to add train details or reserve the ticket for passengers.
 - Then we will start the Apache and SQL. After starting sql we will click admin which will redirect us to phpMyAdmin where we will create our database of train details and reservation.
 - After clicking "Add train" button admin directed to the add train page where he can add edit or delete train details in data base. If he clicked the reserve button he can reserve the tickets according to passenger requirement

Conclusion and Research

Railway Reservation system has been created using different java libraries, class, front end frame work, SQL, Array list(part 2), Linked List(part 2). I created a admin page from where we can add train details(like adding trains for new route, managing there fare and the new train time etc) and reserve ticket seats for according to the customer demand.

REFERENCES

- www.google.com
- www.geeksforgeeks.com
- [And other informative websites for different java libraries](#)

APPENDIX:

CODE SCREENSHOT

Setting passwords for admin:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    if(txtUsername.getText().trim().isEmpty() && txtPassword.getText().trim().isEmpty())  
    {  
        JOptionPane.showMessageDialog(this, "Blank not Allowed");  
    }  
    else if(txtUsername.getText().equals("Admin") && txtPassword.getText().equals("123"))  
    {  
        Main m = new Main();  
        this.hide();  
        m.setVisible(true);  
    }  
    else  
    {  
        JOptionPane.showMessageDialog(this, "UserName or Password Do not match");  
    }  
}
```

Buttons to add train and reservation on main page:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    addtrain a= new addtrain();  
    a.setVisible(true);  
  
}
```

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    reservation r =new reservation();  
    r.setVisible(true);  
}
```

Connecting to my database server:

```
public addtrain() {
    initComponents();
    Connect();
    Train_Details();
}

Connection con;
PreparedStatement pst;
ResultSet rs;

public void Connect()
{
    try {
        Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost/railway","root","");
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(addtrain.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(addtrain.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

To add Trains details to my database:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String tno = txttno.getText();
    String trainname = tname.getText();
    String splace = tstartplace.getText();
    String dplace = tdplace.getText();
    String price = txtprice.getText();

    try {
        pst = con.prepareStatement("insert into traindetails(tno,tname,startplace,destinationplace,price)values(?,?,?,?,?)");
        pst.setString(1, tno);
        pst.setString(2, trainname);
        pst.setString(3, splace);
        pst.setString(4, dplace);
        pst.setString(5, price);
        int k=pst.executeUpdate();

        if(k==1)
        {
            JOptionPane.showMessageDialog(this, "Record Adedd");
        }
        else
        {
            JOptionPane.showMessageDialog(this, "Record Failed");
        }
        txttno.setText("");
        tname.setText("");
        tstartplace.setText("");
        tdplace.setText("");
        txtprice.setText("");
        txttno.requestFocus();
        Train_Details();
    } catch (SQLException ex) {
```

To Update the train records:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String tno = txttno.getText();  
    String trainname = tname.getText();  
    String splace = tstartplace.getText();  
    String dplace = tdplace.getText();  
    String price = txtprice.getText();  
    try {  
        pst = con.prepareStatement("update traindetails set tname = ?,startplace = ?,destinationplace = ?,price = ? where tno = ?");  
        pst.setString(1, trainname);  
        pst.setString(2, splace);  
        pst.setString(3, dplace);  
        pst.setString(4, price);  
        pst.setString(5, tno);  
        int k=pst.executeUpdate();  
        if(k==1){  
            JOptionPane.showMessageDialog(this, "Record Updated");  
        }else{  
            JOptionPane.showMessageDialog(this, "Record Failed");  
        }  
        txttno.setText("");  
        tname.setText("");  
        tstartplace.setText("");  
        tdplace.setText("");  
        txtprice.setText("");  
        txttno.requestFocus();  
        Train_Details();  
        jButton2.setEnabled(true);  
    } catch (SQLException ex) {  
        Logger.getLogger(addtrain.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

To Delete train details from database:

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String tno = txtno.getText();  
  
    try {  
        pst = con.prepareStatement("delete from traindetails where tno = ?");  
        pst.setString(1, tno);  
        int k=pst.executeUpdate();  
        if(k==1)  
        {  
            JOptionPane.showMessageDialog(this, "Record Deleted");  
        }  
        else  
        {  
            JOptionPane.showMessageDialog(this, "Record Failed");  
        }  
        txtno.setText("");  
        tname.setText("");  
        tstartplace.setText("");  
        tdplace.setText("");  
        txtprice.setText("");  
  
        txtno.requestFocus();  
        Train_Details();  
        jButton2.setEnabled(true);  
    } catch (SQLException ex) {  
        Logger.getLogger(addtrain.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

To reset train data:

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Train_Details();  
    jButton2.setEnabled(true);  
}
```

To show train details on admin page:

```
public void Train_Details()
{
    try {
        pst =con.prepareStatement("select * from traindetails");
        rs = pst.executeQuery();

        ResultSetMetaData rsd = rs.getMetaData();
        int c;
        c = rsd.getColumnCount();
        DefaultTableModel d = (DefaultTableModel)jTable1.getModel();
        d.setRowCount(0);

        while(rs.next())
        {
            Vector v =new Vector();
            for(int i=0; i<=c; i++)
            {
                v.add(rs.getString("tno"));
                v.add(rs.getString("tname"));
                v.add(rs.getString("startplace"));
                v.add(rs.getString("destinationplace"));
                v.add(rs.getString("price"));
            }
            d.addRow(v);
        }

    } catch (SQLException ex) {
        Logger.getLogger(addtrain.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

To load start place on reservation page:

```
public void LoadStartplace()
{
    try
    {
        pst=con.prepareStatement("SELECT startplace FROM traindetails");
        rs=pst.executeQuery();
        txtsplace.removeAllItems();
        while(rs.next())
        {
            txtsplace.addItem(rs.getString(1));
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

To load end place on reservation page:

```
public void LoadEndplace()
{
    try
    {
        pst=con.prepareStatement("SELECT destinationplace FROM traindetails");
        rs=pst.executeQuery();
        txtplace.removeAllItems();
        while(rs.next())
        {
            txtplace.addItem(rs.getString(1));
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

To search trains between start place and end place:

```
public void load()
{
    String splace = txtsplace.getSelectedItem().toString();
    String eplace = txtplace.getSelectedItem().toString();

    try
    {
        pst=con.prepareStatement("SELECT * FROM traindetails where startplace = ? and destinationplace = ?");
        pst.setString(1, splace);
        pst.setString(2, eplace);
        rs = pst.executeQuery();

        if(rs.next()==true)
        {
            txttno.setText(rs.getString(1));
            txttname.setText(rs.getString(2));
            txtprice.setText(rs.getString(5));
        }
        else
        {
            JOptionPane.showMessageDialog(this,"Record Not Found");
        }
    }

    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

To evaluate total price(by pressing ENTER key):

```
private void txtticketKeyPressed(java.awt.event.KeyEvent evt) {  
    // TODO add your handling code here:  
  
    int price,ticket,tot;  
    if(evt.getKeyCode()== KeyEvent.VK_ENTER)  
    {  
        price=Integer.parseInt(txtprice.getText());  
        ticket=Integer.parseInt(txtticket.getText());  
  
        tot = price * ticket;  
  
        txttotal.setText(String.valueOf(tot));  
    }  
}
```

To add reservation details to database:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String pno = txtpno.getText();  
    String splace = txtsplace.getSelectedItem().toString();  
    String eplace = txteplace.getSelectedItem().toString();  
    String tno = txttno.getText();  
    String tname = txttname.getText();  
    String price = txtprice.getText();  
    SimpleDateFormat date_form = new SimpleDateFormat("yyyy-MM-dd");  
    String date=date_form.format(txtdate.getDate());  
    String ticket = txtticket.getText();  
    String total = txttotal.getText();  
    try {  
        pst = con.prepareStatement("insert into reservation(pno,splace,eplace,tno,tname,price,date,ticket,tot) values(?,?,?,?,?,?,?,?)");  
        pst.setString(1, pno);  
        pst.setString(2, splace);  
        pst.setString(3, eplace);  
        pst.setString(4, tno);  
        pst.setString(5, tname);  
        pst.setString(6, price);  
        pst.setString(7, date);  
        pst.setString(8, ticket);  
        pst.setString(9, total);  
        int k=pst.executeUpdate();  
        if(k==1){  
            JOptionPane.showMessageDialog(this, "Record Adedd");  
        }  
        else{  
            JOptionPane.showMessageDialog(this, "Record Failed");  
        }  
    } catch (SQLException ex) {  
        Logger.getLogger(addtrain.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

OUTPUT SCREENSHOT

Login Page:



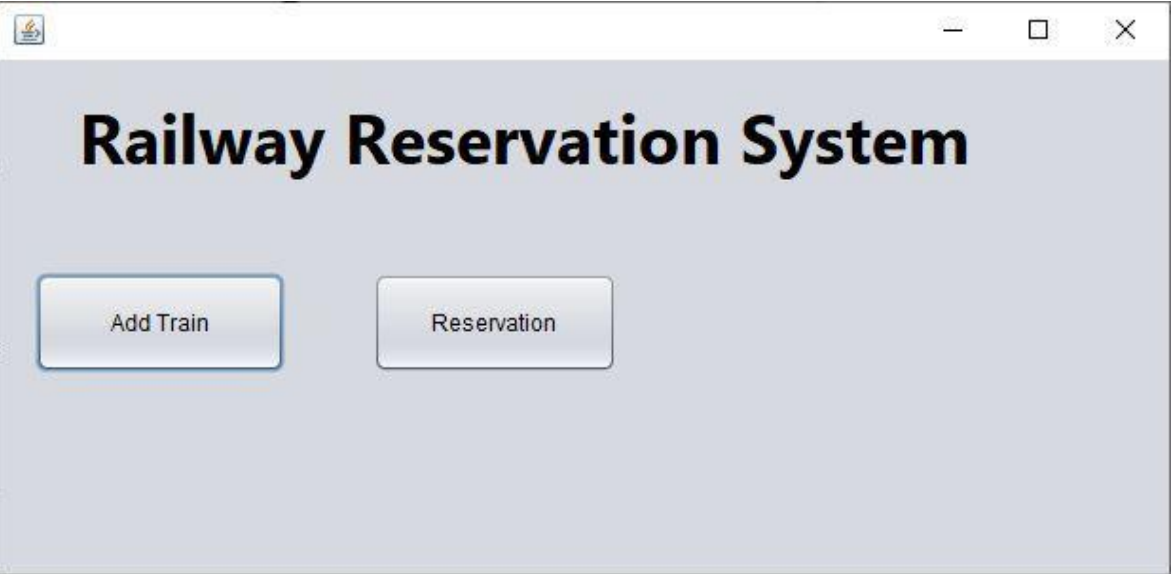
A screenshot of a web application window titled "Login". The window has a light gray background and a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area is a darker gray rectangle. At the top center of this area is the word "Login" in a large, bold, black font. Below it is a white rectangular form containing two input fields. The first field is labeled "UserName" and contains the text "Admin". The second field is labeled "Password" and contains three asterisks "***". Below the form are two buttons: "Login" and "Cancel", both with a light gray gradient and rounded corners.

Login

UserName

Password

Main Page:



A screenshot of a web application window titled "Railway Reservation System". The window has a light gray background and a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area is a darker gray rectangle. At the top center of this area is the text "Railway Reservation System" in a large, bold, black font. Below this text are two buttons: "Add Train" and "Reservation", both with a light gray gradient and rounded corners.

Railway Reservation System

Adding Train details to database:

Add Train

Train No

12001

Train Name

ProjectEXPRESS

Startplace

delhi

Destinationplace

Mumbai

Price

1700

TrainNo	Train Name	Start Place	D Place	Price
11221	Gxpress	Coimbatore	Chennai	1600
12345	TExpress	Chennai	OOTy	1200
27001	saptkranti	Patna	Delhi	2345
		Trichi	Banglore	1300

Message

 Record Adedd

OK

AddTrain

Edit

Delete

Reset

Train details:

TrainNo	Train Name	Start Place	D Place	Price
11221	Gxpress	Coimbatore	Chennai	1600
12001	ProjectEXPR...	delhi	Mumbai	1700
12345	TExpress	Chennai	OOTy	1200
27001	saptkranti	Patna	Delhi	2345
54321	MExpress	Trichi	Banglore	1300

Reservation page:

Reservation

Passenger No

6

Start Place

delhi

End Place

Mumbai

Train No

12001

Train Name

ProjectEXPRESS

Price

1700

Date

Apr 15, 2022

No of Tickets

5

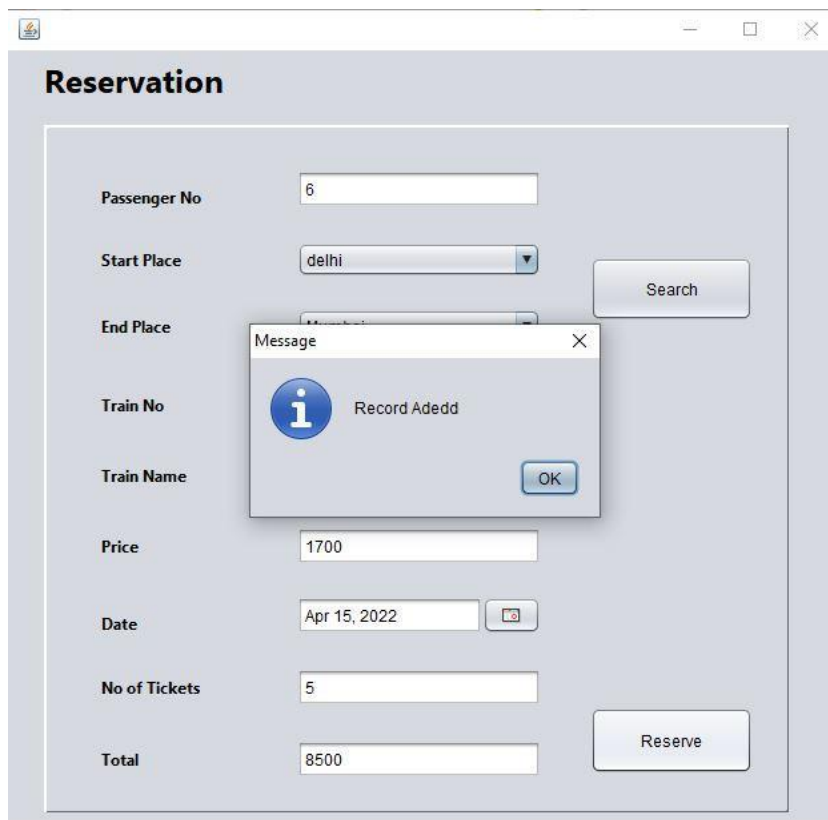
Total

8500

Search

Reserve

Adding passenger details to database:



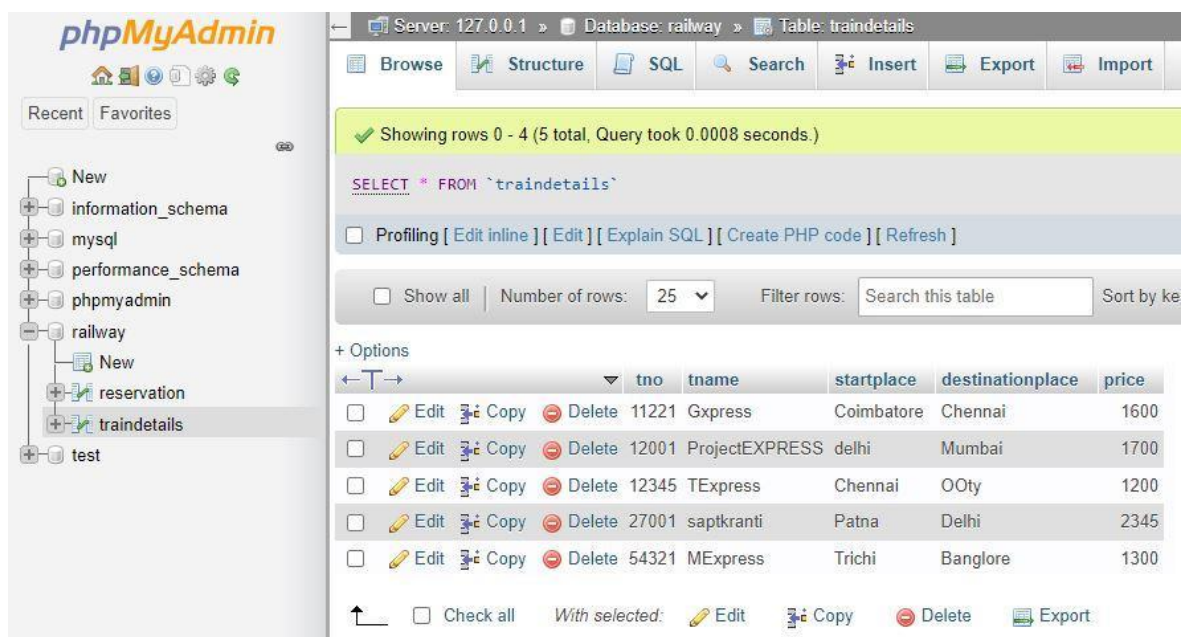
The screenshot shows a web application window titled "Reservation". It contains a form with the following fields and values:

- Passenger No: 6
- Start Place: delhi
- End Place: (empty)
- Train No: (empty)
- Train Name: (empty)
- Price: 1700
- Date: Apr 15, 2022
- No of Tickets: 5
- Total: 8500

There are two buttons: "Search" and "Reserve". A modal message box is displayed in the center with the text "Record Added" and an "OK" button.

My DataBase for Train details and Passenger details:

Train database:

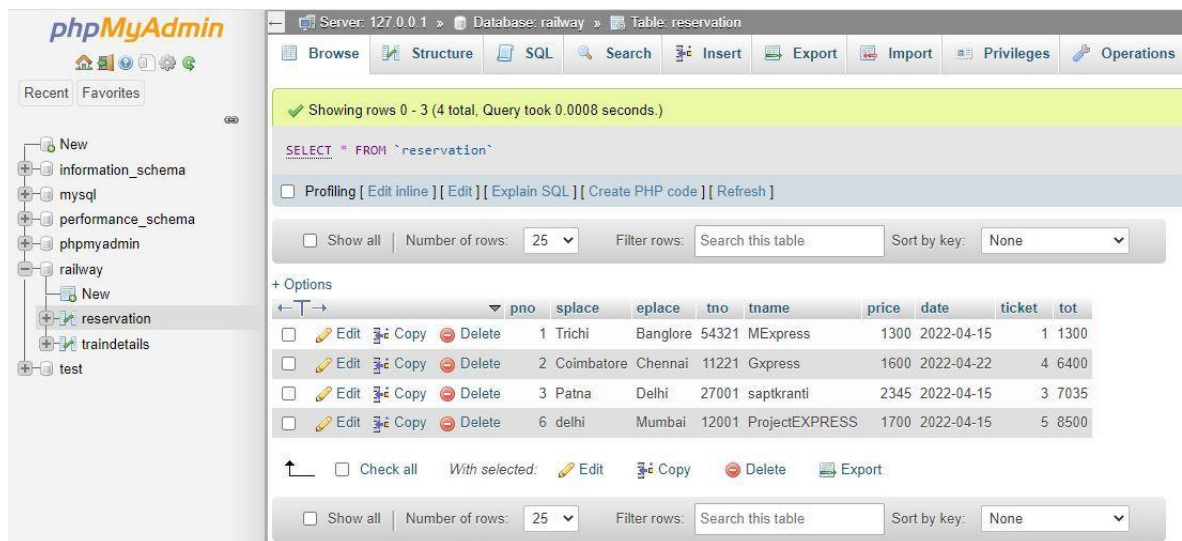


The screenshot shows the phpMyAdmin interface. The left sidebar shows the database structure, with 'railway' selected and 'traindetails' highlighted. The main area displays the 'traindetails' table with the following data:

	tno	tname	startplace	destinationplace	price
☐	11221	Gxpress	Coimbatore	Chennai	1600
☐	12001	ProjectEXPRESS	delhi	Mumbai	1700
☐	12345	TExpress	Chennai	OOty	1200
☐	27001	saptkranti	Patna	Delhi	2345
☐	54321	MExpress	Trichi	Banglore	1300

At the bottom, there are options to 'Check all', 'With selected', 'Edit', 'Copy', 'Delete', and 'Export'.

reservation database:



Server: 127.0.0.1 > Database: railway > Table: reservation

Showing rows 0 - 3 (4 total, Query took 0.0008 seconds.)

SELECT * FROM `reservation`

Options: Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

	pno	splace	eplace	tno	tname	price	date	ticket	tot
☐ Edit Copy Delete	1	Trichi	Banglore	54321	MExpress	1300	2022-04-15	1	1300
☐ Edit Copy Delete	2	Coimbatore	Chennai	11221	Gxpress	1600	2022-04-22	4	6400
☐ Edit Copy Delete	3	Patna	Delhi	27001	saptkranti	2345	2022-04-15	3	7035
☐ Edit Copy Delete	6	delhi	Mumbai	12001	ProjectEXPRESS	1700	2022-04-15	5	8500

Options: Check all | With selected: Edit Copy Delete Export

Options: Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

PART 2

Honestly the project of railway resrvatuon system which I am reporting in part 2 was my primary project earlier. But this second(part 2) project doesn't contain any database. I entered all the required user data like there name destination boarding point etc manualu using array list and linked list. This was not up to the mark so I came up with this new project which I had given in above part of this report.

INTRODUCTION:

In this project I used the concept of Object oriented. I majorly used arrayList and Linked List to enter the data manually. The admin have to give the passenger details in code only. I didn't used any database to store any data in this project.

Code:

Created a class for train details:

```
public class Train {
    private int trainNumber;
    private String trainName;
    private String fromStation;
    private String toStation;
    private int seatsAvailabe;
    private int fare;
    private LocalDate doj;
    public Train() {
        super();
    }
    public Train(int trainNumber, String trainName, String fromStation, String toStation, int seatsAvailabe, int fare,
        LocalDate doj) {
        super();
        this.trainNumber = trainNumber;
        this.trainName = trainName;
        this.fromStation = fromStation;
        this.toStation = toStation;
        this.seatsAvailabe = seatsAvailabe;
        this.fare = fare;
        this.doj = doj;
    }
}
```

Getting and setting the value for train details:

```

public int getTrainNumber() {
    return trainNumber;
}
public void setTrainNumber(int trainNumber) {
    this.trainNumber = trainNumber;
}
public String getTrainName() {
    return trainName;
}
public void setTrainName(String trainName) {
    this.trainName = trainName;
}
public String getFromStation() {
    return fromStation;
}
public void setFromStation(String fromStation) {
    this.fromStation = fromStation;
}
public String getToStation() {
    return toStation;
}
public void setToStation(String toStation) {
    this.toStation = toStation;
}
public int getSeatsAvailabe() {
    return seatsAvailabe;
}
public void setSeatsAvailabe(int seatsAvailabe) {
    this.seatsAvailabe = seatsAvailabe;
}
public int getFare() {
    return fare;
}
public void setFare(int fare) {
    this.fare = fare;
}
public LocalDate getDoj() {
    return doj;
}
public void setDoj(LocalDate doj) {
    this.doj = doj;
}
}

```

Setitng the value of train details using linked list:

```

private static List<Train> allTrains = new LinkedList<>();

static {
    allTrains.add(new Train(101,"Train-one","Hyderabad","Bangalore",100,800,LocalDate.now()));
    allTrains.add(new Train(102,"Train-two","Bangalore","Hyderabad",100,800,LocalDate.now()));
    allTrains.add(new Train(103,"Train-three","Hyderabad","Bangalore",80,600,LocalDate.now()));
    allTrains.add(new Train(104,"Train-four","Bangalore","Hyderabad",80,600,LocalDate.now()));
    allTrains.add(new Train(105,"Train-five","Hyderabad","Chennai",100,900,LocalDate.now()));
    allTrains.add(new Train(106,"Train-six","Chennai","Hyderabad",100,800,LocalDate.now()));
}

```

Giving the total fare according the customer requirement:

```

public static void bookTickets(int trainNumber, List<Passenger> passengerList) {
    int numofSeats = passengerList.size();
    /*Passenger tempPassenger = passengerList.get(0);|
    BankAccount account = tempPassenger.getBankAccount();*/

    BankAccount account = passengerList.get(0).getBankAccount();
    Train tempTrain = findTrain(trainNumber);
    int fare = tempTrain.getFare();
    int totalFare = fare * numofSeats;
    account.withdraw(totalFare);

    tempTrain.setSeatsAvailabe(tempTrain.getSeatsAvailabe()-numofSeats);

    TicketService.addNewTicket(trainNumber, passengerList);
}

```

Giving the passenger details:

```

System.out.println("Now searching for trains...");
TrainService.searchTrainsBetweenStations("Hyderabad","Bangalore", LocalDate.now(), 10);

BankAccount b1 = new BankAccount();
b1.setActNumber(999);
b1.setActBalance(10000);

Passenger p1 = new Passenger(10, "Amit", 69, b1);
Passenger p2 = new Passenger(20, "Ankit", 38, b1);
Passenger p3 = new Passenger(10, "Sumit", 39, b1);

int initialBalance = b1.getActBalance();

List<Passenger>passengerList = new LinkedList<>();

passengerList.add(p1);
passengerList.add(p2);
passengerList.add(p3);

System.out.println("Now booking tickets for 3 passengers in train number 101...");

```

Giving the account details for final payment:

```

public class BankAccount {
    private int actNumber;
    private int actBalance;
    public int getActNumber() {
        return actNumber;
    }
    public void setActNumber(int actNumber) {
        this.actNumber = actNumber;
    }
    public int getActBalance() {
        return actBalance;
    }
    public void setActBalance(int actBalance) {
        this.actBalance = actBalance;
    }
    public void deposit(int amount) {
        actBalance += amount;
    }
    public void withdraw(int amount) {
        actBalance -= amount;
    }
    public void showBalance() {
        System.out.println("Act Balance = " +actBalance);
    }
}

```

Displaying the final output for ticket details and payment:

```

Train tempTrain = tempTicket.getBookedTrain();
List<Passenger> passengerList = tempTicket.getPassengerList();
int totalFare = tempTrain.getFare() * passengerList.size();

System.out.println("**** YOUR BOOKING DETAILS ****");
System.out.println("PNR=>" + tempTicket.getPnr() + " Tr.No" + tempTrain.getTrainNumber());
System.out.println("Train Name :" + tempTrain.getTrainName());
System.out.println("From Station: " + tempTrain.getFromStation() + " To Station :" + tempTrain.getToStation());
System.out.println("DOJ:" + tempTrain.getDoj() + " Fare:" + tempTrain.getFare() + " Total Fare:" + totalFare);
System.out.println("\nPassenger info below\n");

System.out.println("Passenger Name    Age");
for(Passenger p : passengerList) {
    System.out.println(p.getPassengerName()+"\t"+p.getAge());
}

```

OUTPUT:

Passenger info below

Passenger Name	Age
----------------	-----

Amit	69
------	----

Ankit	38
-------	----

Sumit	39
-------	----

Tr.No	Train Name	From Station	To Station	Doj	Seats Available	Fa
101	Train-one	Hyderabad	Bangalore	2022-04-11	97	800
103	Train-three	Hyderabad	Bangalore	2022-04-11	80	600

Initial balance = 10000
Balance after booking = 7600

My GITHUB account link:

<https://github.com/08raushan/Railway-reservation-system>