

# **RAILWAY MANAGEMENT SYSTEM**

## **A MINI PROJECT REPORT**

**Submitted by**

**SANJANA SHREE S 220701245**

**SHAKTHI PRIYA V 220701259**

In partial fulfillment for the award of the degree of

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE**

**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**

**THANDALAM**

**CHENNAI – 602105**

**2023-2024**

# **BONAFIDE CERTIFICATE**

Certified that this project report

**"RAILWAY MANAGEMENT SYSTEM"**

Is the bonafide work of

**SANJANA SHREE S (220701245)**

**SHAKTHI PRIYA V (220701259)**

Who carried out the project work under my supervision.

**Submitted for the Practical Examination held on**

**SIGNATURE**

**Mrs.K. MahesMeena**

**Assistant Professor(SG),**

**Computer Science and Engineering**

**Rajalakshmi Engineering College**

# **ABSTRACT**

The Railway management system (RMS) is a comprehensive software solution designed to streamline and enhance the efficiency of railway operations. This system integrates various functions including ticket booking, scheduling, checking, availability, and resources management into a unified platform. It aims to provide real time information to passengers, optimize train schedules, and ensure the effective utilization of resources.

The RMS is built with a user-friendly interfaces for passengers to easily books tickets, check train statues, know about delays or changes in schedules.

# **TABLE OF CONTENTS**

## **1. INTRODUCTION**

1.1 INTRODUCTION

1.2 OBJECTIVES

1.3 MODULES

## **2. SURVEY OF TECHNOLOGIES**

2.1 SOFTWARE DESCRIPTION

2.2 LANGUAGES

2.2.1 MYSQL

2.2.2 PYTHON

## **3. REQUIREMENTS AND ANALYSIS**

3.1 REQUIREMENT SPECIFICATION

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

3.3 ARCHITECTURE DIAGRAM

3.4 ER DIAGRAM

3.5 NORMALIZATION

## **4. PROGRAM CODE**

## **5. RESULTS AND DISCUSSION**

## **6. CONCLUSION**

## **7. REFERENCES**

# 1.INTRODUCTION

## 1.1 INTRODUCTION

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 main aim is to develop a data base maintaining records of different trains, train status and passengers. 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. 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 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.

## 1.2 OBJECTIVES

- Efficient ticketing system
- Availability checking
- Schedule optimization
- Passenger information system
- Data Maintenance
- Resource Maintenance

## 1.3 MODULES

User login page : To add the passenger those who visit the system, into the database.

Book Ticket : To book the ticket.

Cancel Ticket : To cancel the ticket.

## 2. SURVEY OF TECHNOLOGIES

### 2.1 SOFTWARE DESCRIPTION:

**Ticketing System:** RMS includes a ticketing module that facilitates the booking, reservation, and cancellation of tickets for passengers. It provides various options for ticket purchase, such as online booking portals, mobile apps, and ticket counters at railway stations. The system manages seat availability, fare calculation, and passenger information efficiently.

**Train Scheduling and Management:** This module handles the scheduling of trains, including route planning, allocation of time slots, and coordination of train movements. It ensures optimal utilization of railway infrastructure while minimizing conflicts and delays. Train schedules can be dynamically adjusted based on factors like maintenance, weather conditions, and unforeseen events.

**Passenger Information System (PIS):** PIS delivers real-time information to passengers regarding train schedules, delays, platform assignments, and other relevant updates. It may include digital displays at

stations, mobile apps, and automated announcements to keep passengers informed throughout their journey.

**Safety and Security:** RMS includes features to enhance safety and security across the railway network. This may involve surveillance cameras, access control systems, and emergency response protocols to mitigate risks and respond to incidents promptly. Additionally, the system may analyze operational data to identify potential safety hazards and implement preventive measures.

**Revenue Management:** The software handles financial transactions related to ticket sales, freight shipments, and other revenue streams. It generates reports on revenue generation, passenger traffic, and operational costs to support financial planning and decision-making. Integration with accounting systems ensures accurate financial management and compliance with regulatory requirements.

**Analytics and Reporting:** RMS collects and analyzes vast amounts of data generated by railway operations to derive actionable insights. This includes performance metrics such as on-time performance, passenger satisfaction, and resource utilization. Advanced analytics techniques, such as predictive modeling and data



visualization, enable operators to optimize operations and improve service quality over time.

## 2.2 LANGUAGES

- MYSQL
- PYTHON

### MYSQL:

In a Railway Management System (RMS) project, MYSQL can be utilized in various aspects to efficiently manage and organize railway-related data. Here's how MySQL can be used in different modules of an RMS project:

#### 1. Ticketing System:

- MYSQL can store information about available seats, ticket reservations, passenger details, and ticket pricing.
- Tables can be created to manage ticket booking transactions, including seat assignments, payment status, and booking timestamps.

## 2. Train Scheduling and Management:

- MySQL can maintain schedules for trains, including departure and arrival times, route details, and platform assignments.
- Data related to train routes, stops, and distances between stations can be stored in MySQL tables.

## 3. Passenger Information System (PIS):

- MySQL can be used to store real-time data on train statuses, delays, and platform announcements.
- Information about station facilities, amenities, and services can be stored in MySQL tables for passenger reference.

## 4. Safety and Security:

- MySQL can store data related to safety inspections, incident reports, and security protocols.
- Tables can be created to track surveillance footage, access control logs, and emergency response procedures.

## 5. Analytics and Reporting:

- MySQL can store historical data for analysis, such as passenger traffic, on-time performance, and revenue trends. Tables can be created to generate reports and visualizations for

stakeholders, management, and regulatory authorities.

## PYTHON:

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers, such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

### 1. Web Development:

- Python frameworks such as DJANGO or Flask can be used to build the backend of web applications for ticket booking systems, passenger information systems, and administrative dashboards.

### 2. Data Processing and Analysis:

- Python libraries such as Pandas, NUMPY, and SCIPY can be used for processing and analyzing large datasets related to train schedules, passenger demographics, and revenue management.

### 3. Automation:

- Python scripts can automate routine tasks in the RMS, such as generating reports, updating train schedules, and sending notifications to passengers about delays or cancellations.
- Automated scripts can be developed to synchronize data between different modules of the RMS, ensuring consistency and accuracy.

### 4. Integration with MYSQL Database:

- Python's MYSQL Connector library allows developers to connect Python applications with MYSQL databases, enabling the retrieval, insertion, and manipulation of data.
- Python scripts can execute SQL queries to retrieve information from MYSQL tables, update records, or perform data analysis tasks.

### 5. API Development:

- Python can be used to develop restful APIs (Application Programming Interfaces) to expose functionality of the RMS to external systems or mobile applications.

## **3.REQUIREMENT AND ANALYSIS**

### **3.1 REQUIREMENT SPECIFICATION**

Designing a Railway Management System (RMS) involves a comprehensive analysis of the requirements to ensure smooth operations, safety, and customer satisfaction. Here's an outline of the requirements and analysis for such a system:

Functional Requirements:

#### **1. Train Scheduling and Routing:**

- Ability to schedule trains, allocate routes, and manage timetables efficiently.
- Support for automatic route optimization and adjustment for unexpected events.

#### **2. Ticket Booking and Reservation:**

- Online and offline ticket booking facilities for passengers.
- Seat reservation, cancellation, and modification functionalities.

- Integration with payment gateways for secure transactions.

### 3. Passenger Information System:

- Real-time information on train schedules, delays, cancellations, and platform changes.
- Integration with mobile apps, websites, and display boards at stations.

### 4. Maintenance and Repair:

- Scheduling and tracking of maintenance tasks for trains, tracks, and infrastructure.
- Integration with inventory management for spare parts and supplies.

### 5. Security and Safety:

- Surveillance and monitoring systems onboard trains and at stations.
- Emergency communication systems for passengers and staff.
- Integration with law enforcement agencies for security alerts and responses.

## Non-functional Requirements:

### 1. Performance:

- High availability and reliability to ensure uninterrupted service.
- Scalability to handle peak loads during holidays and special events.

## 2. Security:

- Data encryption and secure communication protocols to protect passenger information and financial transactions.
- Access control mechanisms to prevent unauthorized access to sensitive systems.

## 3. Usability:

- Intuitive user interfaces for both passengers and railway staff.
- Accessibility features for passengers with disabilities.

## 4. Interoperability:

- Integration with other railway systems, transportation networks, and third-party services.

## 5. Regulatory Compliance:

- Adherence to safety regulations, industry standards, and government policies.
- Regular audits and inspections to ensure compliance.

## 3.2 HARDWARE AND SOFTWARE REQUIREMENTS

### Hardware:

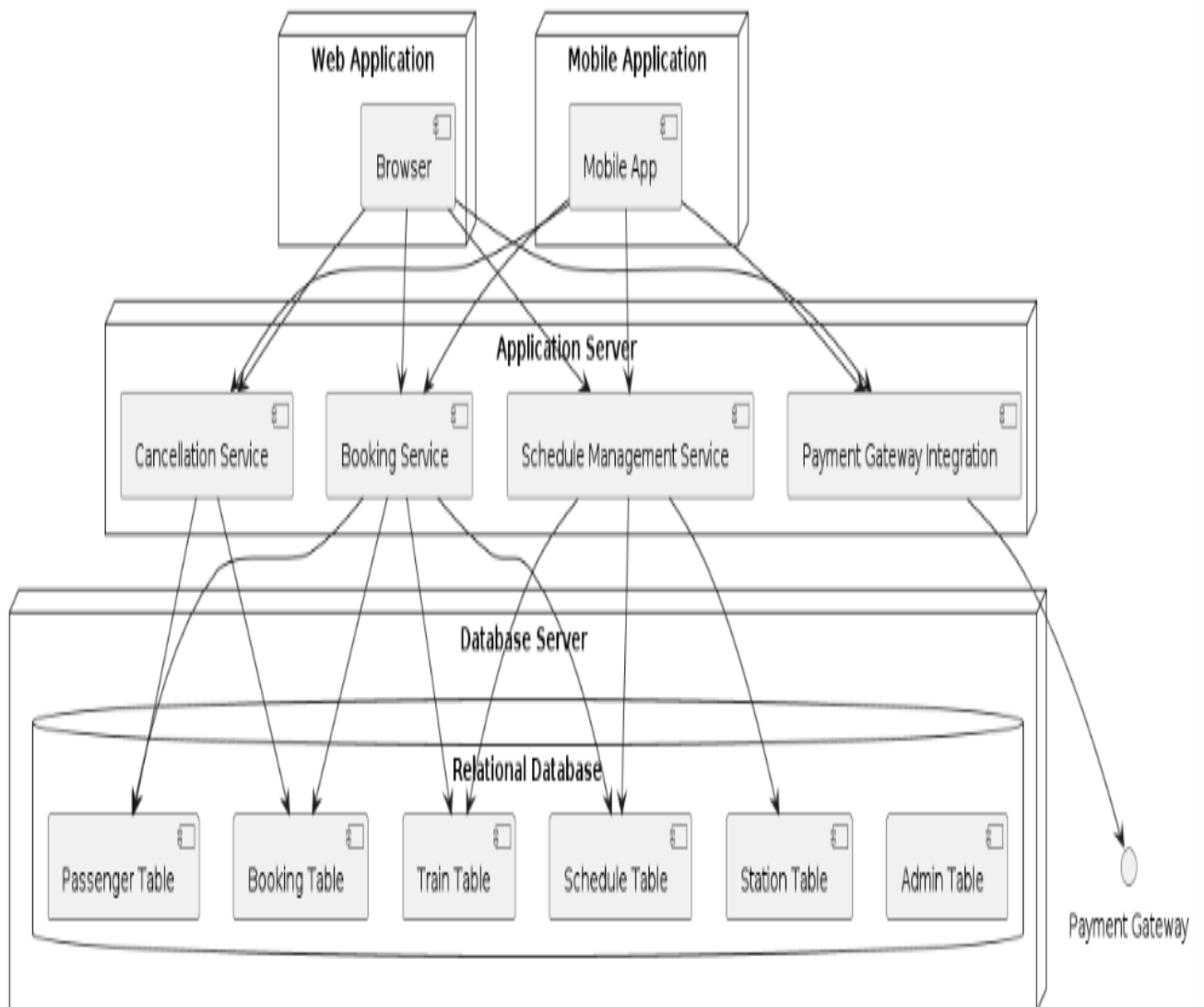
- i) Server with minimum 4GB RAM and 100GB storage.
- ii) Client devices with internet access.

### Software:

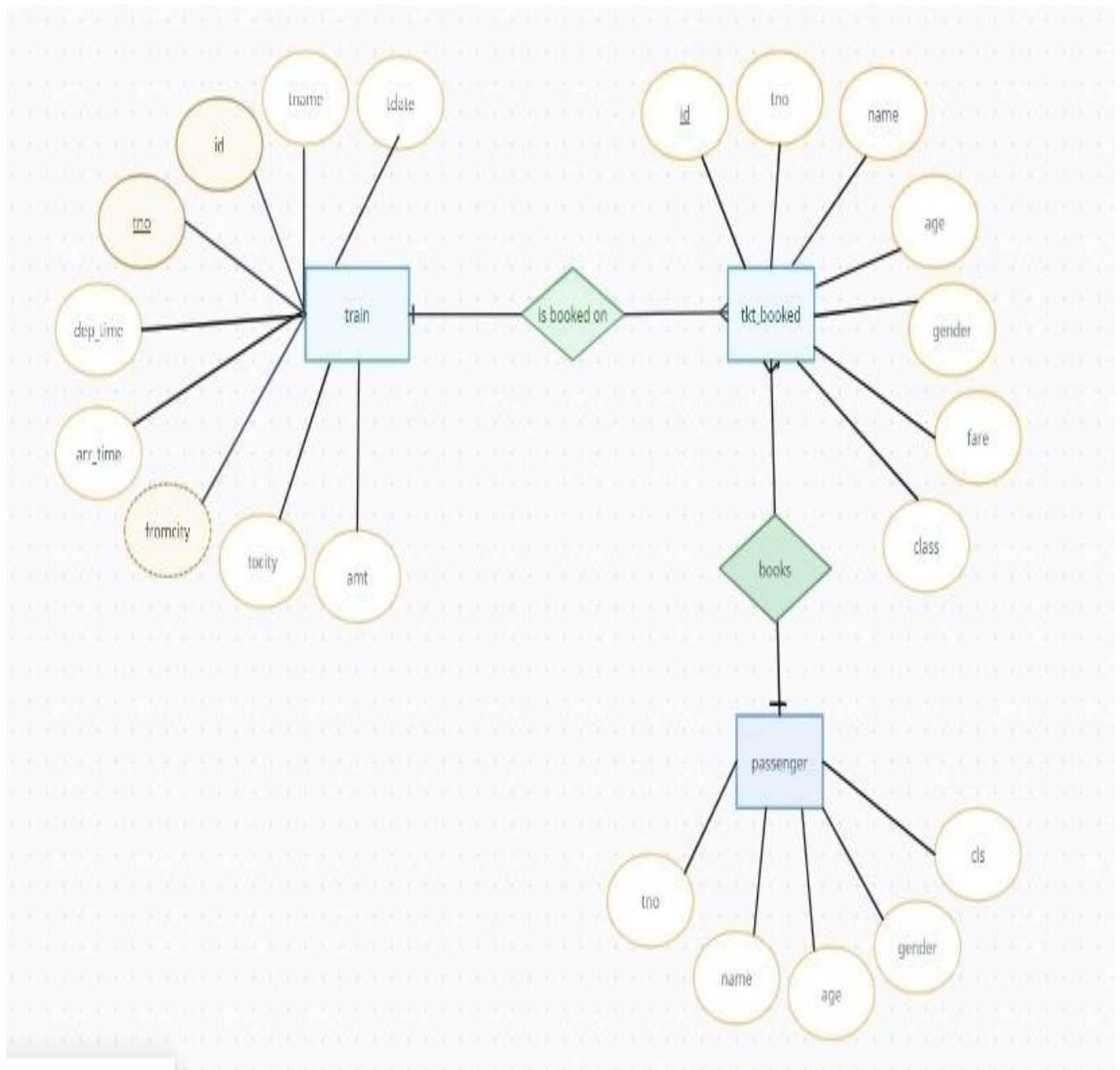
- i) Operating System : Windows/Linux
- ii) Web Server : Apache
- iii) Database : MySQL
- iv) Languages : Python
- v) Libraries : Tkinter



### 3.3 ARCHITECTURE DIAGRAM



## 3.4 ER-DIAGRAM



## 3.5 NORMALIZATION

The database is normalized to reduce redundancy and ensure data integrity. The normalization process involves organizing the database into tables and columns according to the following forms:

1st Normal Form (1NF): Ensure that the table contains only atomic values and each column contains unique values.

2nd Normal Form (2NF): Remove partial dependencies by ensuring that each non-key attribute is fully functionally dependent on the primary key.

3rd Normal Form (3NF): Remove transitive dependencies to ensure that non-key attributes are not dependent on other non-key attributes.

# MYSQL DATABASE TABLES

- Train
- Tkt\_booked
- Passenger

## STRUCTURE OF THE TABLES

```
mysql> desc train;
```

| Field    | Type         | Null | Key | Default | Extra          |
|----------|--------------|------|-----|---------|----------------|
| id       | int          | NO   | PRI | NULL    | auto_increment |
| fromcity | varchar(255) | NO   |     | NULL    |                |
| tocty    | varchar(255) | NO   |     | NULL    |                |
| tno      | int          | NO   |     | NULL    |                |
| tname    | varchar(255) | NO   |     | NULL    |                |
| tdate    | varchar(255) | NO   |     | NULL    |                |
| arr_time | varchar(255) | NO   |     | NULL    |                |
| dep_time | varchar(255) | NO   |     | NULL    |                |
| amt      | int          | NO   |     | NULL    |                |

9 rows in set (0.09 sec)

```
mysql> desc tkt_booked;
```

| Field  | Type         | Null | Key | Default | Extra          |
|--------|--------------|------|-----|---------|----------------|
| id     | int          | NO   | PRI | NULL    | auto_increment |
| tno    | int          | NO   |     | NULL    |                |
| name   | varchar(255) | NO   |     | NULL    |                |
| age    | int          | NO   |     | NULL    |                |
| gender | varchar(10)  | NO   |     | NULL    |                |
| fare   | int          | NO   |     | NULL    |                |
| class  | varchar(255) | NO   |     | NULL    |                |

7 rows in set (0.00 sec)

```
mysql> desc passenger;
```

| Field  | Type         | Null | Key | Default | Extra |
|--------|--------------|------|-----|---------|-------|
| tno    | int          | NO   |     | NULL    |       |
| name   | varchar(100) | NO   |     | NULL    |       |
| age    | int          | NO   |     | NULL    |       |
| gender | varchar(10)  | NO   |     | NULL    |       |
| cls    | varchar(50)  | NO   |     | NULL    |       |

5 rows in set (0.00 sec)

```
mysql>
```

# RECORDED DATA

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 371
Server version: 8.4.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql> use railwayDB
Database changed
mysql> show tables;
+-----+
| Tables_in_railwaydb |
+-----+
| passenger            |
| tkt_booked           |
| train                |
+-----+
3 rows in set (0.19 sec)

mysql> select * from train;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | fromcity | tocity | tno | tname      | tdate   | arr_time | dep_time | amt |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2332 | Delhi    | Chennai | 1121 | Chennai Express | July 23,2024 | 18:00 | 18:00 | 1000 |
| 2345 | Mumbai  | Madgoan | 10001 | Mandovi      | June 23,2024 | 17:00 | 10:00 | 650 |
| 2546 | Chennai | Hyderabad | 10101 | Hyderabad Express | Oct 18,2024 | 09:00 | 21:00 | 1050 |
| 2756 | Chennai | Bengaluru | 12027 | Shatabdi     | Sept 28,2024 | 22:00 | 03:00 | 500 |
| 2766 | Vijayawada | Chennai | 12077 | Jan Shatabdi  | Jun 28,2024 | 23:00 | 06:00 | 900 |
+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.07 sec)
```

```
5 rows in set (0.07 sec)

mysql> select * from tkt_booked;
+-----+-----+-----+-----+-----+-----+
| id | tno | name      | age | gender | fare | class |
+-----+-----+-----+-----+-----+-----+
| 9  | 12027 | Smitha    | 13  | Female | 500  | Second Class |
| 10 | 12027 | Rajalakshmi | 41  | Female | 500  | Second Class |
| 13 | 10101 | Reena     | 24  | Female | 1050 | First Class |
| 14 | 10101 | Raveena   | 19  | Female | 1050 | Second Class |
| 16 | 10101 | Raja      | 33  | Male   | 1050 | Second Class |
| 17 | 12027 | Rocky     | 27  | Male   | 500  | First Class |
| 19 | 12027 | Rakesh    | 27  | Male   | 500  | First Class |
| 20 | 10101 | Rajesh    | 20  | Male   | 1050 | First Class |
| 22 | 12027 | Rashmika  | 20  | Female | 500  | First Class |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql> select * from passenger;
+-----+-----+-----+-----+-----+
| tno | name      | age | gender | cls |
+-----+-----+-----+-----+-----+
| 1121 | ddsss     | 12  | f      | second class |
| 10101 | Reena     | 24  | Female | First Class |
| 10101 | Raena     | 19  | Female | Second Class |
| 12027 | Dhanasekar | 43  | Male   | Second Class |
| 12027 | Naveen    | 20  | Male   | Second Class |
| 10101 | Raveena   | 19  | Female | Second Class |
| 10101 | Raveena   | 19  | Female | Second Class |
| 1121 | ddsss     | 12  | f      | Second Class |
| 10101 | Raja      | 33  | Male   | Second Class |
| 12027 | Rocky     | 27  | Male   | First Class |
| 12027 | Ravi      | 27  | Male   | First Class |
| 12027 | Rakesh    | 27  | Male   | First Class |
| 10101 | Rajesh    | 20  | Male   | First Class |
| 10101 | Sachin    | 27  | Male   | Second Class |
| 10101 | Ravi      | 27  | Male   | Second Class |
| 1121 | Sinaaj    | 30  | Male   | Second Class |
| 2332 | Rashmika  | 20  | Female | First Class |
+-----+-----+-----+-----+-----+
17 rows in set (0.05 sec)
```

## 4.PROGRAM CODE

```
from tkinter import *
from tkinter import messagebox
import mysql.connector
from mysql.connector import Error
def connect_db():
    try:
        con = mysql.connector.connect(
            host='localhost',
            database='railwayDB',
            user='root',
            password='sanjana@0806'
        )
        if con.is_connected():
            return con
    except Error as e:
```

```

        messagebox.showerror("Error", f"Error: {e}")

    return None

def handle_book_ticket(tno_entry, pname_entry, page_entry,
pgender_entry, pclass_entry):

    book_ticket(

        tno_entry.get(), pname_entry.get(), page_entry.get(),
pgender_entry.get(), pclass_entry.get()

    )

def book_ticket(tno, name, age, gender, cls):

    try:

        cursor.execute("SELECT amt FROM train WHERE tno=%s",
(tno,))

        result = cursor.fetchone()

        if result:

            fare = result[0]

            cursor.execute(

                "INSERT INTO tkt_booked (tno, name, age, gender,
fare, class) VALUES (%s, %s, %s, %s, %s, %s)",

                (tno, name, age, gender, fare, cls))

```

```

con.commit()

    messagebox.showinfo("Success", "Ticket booked
successfully")

else:

    messagebox.showerror("Error", "Train number not
found")

except Error as e:

    messagebox.showerror("Error", f"Error: {e}")

def cancel_ticket(tno, name, age, gender, cls):

    try:

        cursor.execute("SELECT * FROM tkt_booked WHERE
tno=%s AND name=%s AND age=%s AND gender=%s AND
class=%s",

            (tno, name, age, gender, cls))

        result = cursor.fetchone()

        if result:

            cursor.execute("DELETE FROM tkt_booked WHERE
tno=%s AND name=%s AND age=%s AND gender=%s AND
class=%s",

                (tno, name, age, gender, cls))

```



```
        con.commit()

        messagebox.showinfo("Success", "Ticket canceled
successfully")

    else:

        messagebox.showerror("Error", "Ticket not found")

except Error as e:

    messagebox.showerror("Error", f"Error: {e}")

def add_passenger(tno,name,age,gender,cls):

    try:

        cursor.execute("INSERT INTO passenger
(tno,name,age,gender,cls) VALUES (%s, %s, %s, %s, %s)"
(tno,name,age,gender,cls))

        con.commit()

        messagebox.showinfo("Success", "Passenger added
successfully")

    except Error as e:

        #messagebox.showerror("Error", f"Error: {e}")

        messagebox.showerror("Error", "All fields are required")

    return
```

```

    passenger_info.set(f"Train No: {tno}, Name: {name}, Age:
{age}, Gender: {gender}, Class: {cls}")

def handle_booking():

    if passenger_info.get() != "":

        handle_book_ticket(tno_entry, pname_entry, page_entry,
pgender_entry, pclass_entry)

    else:

        messagebox.showerror("Error", "No passenger information
added")

def handle_cancelling():

    if passenger_info.get() != "":

        cancel_ticket(tno_entry.get(), pname_entry.get(),
page_entry.get(), pgender_entry.get(), pclass_entry.get())

    else:

        messagebox.showerror("Error", "No passenger information
added")

def main():

    global con, cursor, tno_entry, pname_entry, page_entry,
pgender_entry, pclass_entry, passenger_info

    con = connect_db()

```

```
if con:

    cursor = con.cursor()

    root = Tk()

    root.title("Railway Ticket Booking System")

    root.geometry("800x600")

    Label(root, text="Railway Ticket Booking System",
font=("Arial", 20, "bold")).pack(pady=20)

    # Frame for adding a passenger

    add_frame = Frame(root)

    add_frame.pack(pady=10)

    Label(add_frame, text="Passenger Details", font=("Arial",
16, "bold")).grid(row=0, columnspan=2, pady=10)

    Label(add_frame, text="Train No").grid(row=1, column=0,
pady=5)

    tno_entry = Entry(add_frame)

    tno_entry.grid(row=1, column=1, pady=5)

    Label(add_frame, text="Name").grid(row=2, column=0,
pady=5)

    pname_entry = Entry(add_frame)
```

```
    pname_entry.grid(row=2, column=1, pady=5)

    Label(add_frame, text="Age").grid(row=3, column=0,
pady=5)

    page_entry = Entry(add_frame)

    page_entry.grid(row=3, column=1, pady=5)

    Label(add_frame, text="Gender").grid(row=4, column=0,
pady=5)

    pgender_entry = Entry(add_frame)

    pgender_entry.grid(row=4, column=1, pady=5)

    Label(add_frame, text="Class").grid(row=5, column=0,
pady=5)

    pclass_entry = Entry(add_frame)

    pclass_entry.grid(row=5, column=1, pady=5)

    # Button to add passenger

    Button(add_frame, text="Add Passenger",
command=lambda:add_passenger(tno_entry.get(),
pname_entry.get(), page_entry.get(),

    pgender_entry.get(), pclass_entry.get())).grid(row=6,
columnspan=2, pady=10)

    # Label to display added passenger info
```

```
passenger_info = StringVar()

Label(root, textvariable=passenger_info, font=("Arial", 14),
fg="blue").pack(pady=10)

# Frame for booking and canceling tickets

action_frame = Frame(root)

action_frame.pack(pady=10)

# Button to book ticket

Button(action_frame, text="Book Ticket",
command=handle_booking).grid(row=0, column=0, padx=20,
pady=10)

# Button to cancel ticket

Button(action_frame, text="Cancel Ticket",
command=handle_cancelling).grid(row=0, column=1, padx=20,
pady=10)

root.mainloop()

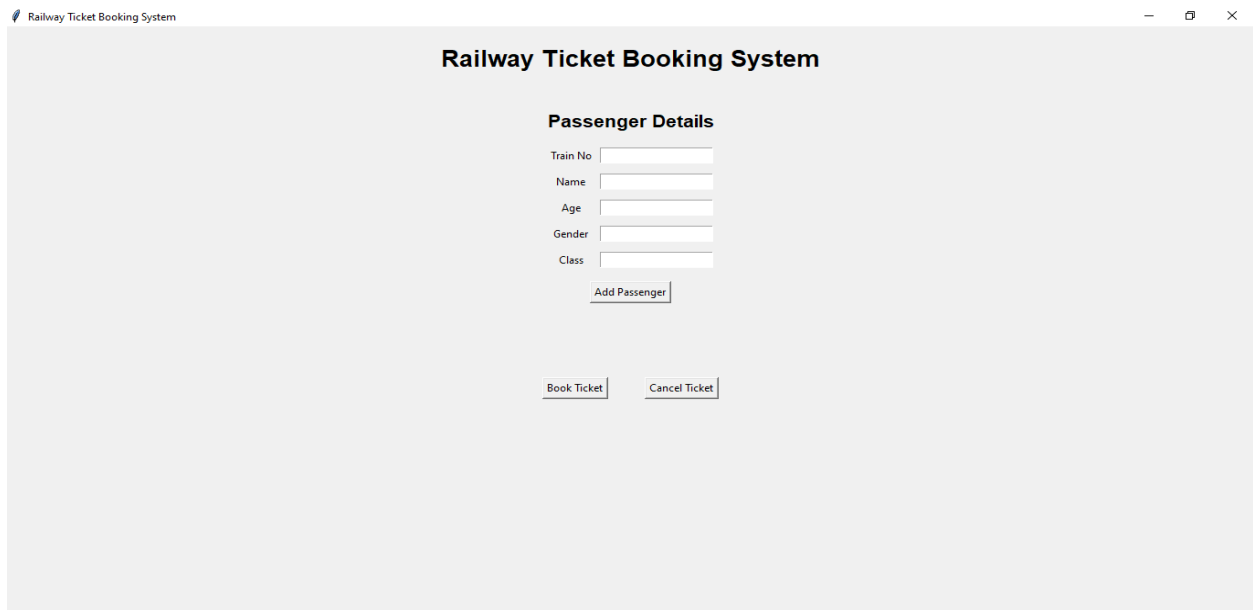
cursor.close()

con.close()

if __name__ == "__main__":
    main()
```

# 5.RESULTS AND DISCUSSION

## HOME PAGE



The screenshot displays the home page of a 'Railway Ticket Booking System'. The page has a light gray background. At the top left, there is a small logo and the text 'Railway Ticket Booking System'. At the top right, there are three small icons: a minus sign, a square, and an 'X'. The main heading 'Railway Ticket Booking System' is centered at the top. Below this, the section 'Passenger Details' is centered. It contains five input fields: 'Train No', 'Name', 'Age', 'Gender', and 'Class'. Below these fields is a button labeled 'Add Passenger'. At the bottom of the form, there are two buttons: 'Book Ticket' and 'Cancel Ticket'.

Railway Ticket Booking System

**Passenger Details**

Train No

Name

Age

Gender

Class

# ADDING THE PASSENGERS

Railway Ticket Booking System

### Passenger Details

|          |             |
|----------|-------------|
| Train No | 10101       |
| Name     | Tina        |
| Age      | 24          |
| Gender   | Female      |
| Class    | First Class |

Success

Passenger added successfully

OK

# DISPLAYING THE DETAILS

Railway Ticket Booking System

### Passenger Details

|          |             |
|----------|-------------|
| Train No | 10101       |
| Name     | Tina        |
| Age      | 24          |
| Gender   | Female      |
| Class    | First Class |

Add Passenger

Train No: 10101, Name: Tina, Age: 24, Gender: Female, Class: First Class

Book Ticket Cancel Ticket

# BOOKING TICKET

Railway Ticket Booking System

### Passenger Details

|          |             |
|----------|-------------|
| Train No | 10101       |
| Name     | Tina        |
| Age      | 24          |
| Gender   | Female      |
| Class    | First Class |

Train No: 10101, Name: Tina, Age: 24, Gender: Female, Class: First Class

Success

Ticket booked successfully

OK

# CANCELLING TICKET

Railway Ticket Booking System

### Passenger Details

|          |             |
|----------|-------------|
| Train No | 10101       |
| Name     | Tina        |
| Age      | 24          |
| Gender   | Female      |
| Class    | First Class |

Train No: 10101, Name: Tina, Age: 24, Gender: Female, Class: First Class

Success

Ticket canceled successfully

OK



## **6.CONCLUSION**

In conclusion, the successful implementation of an RMS represents a significant milestone in the evolution of railway systems, unlocking new levels of efficiency, safety, and passenger satisfaction. As technology continues to advance and evolve, the potential for innovation and improvement within the realm of railway management remains boundless, promising a future of smarter, safer, and more sustainable transportation networks.

# 7.REFERENCES

Database System Concepts by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan

This book provides a solid foundation in database concepts and design.

Learning MySQL by Seyed M.M. Tahaghoghi and Hugh Williams  
A detailed guide to understanding and using MySQL effectively.

## **MySQL Documentation**

Official MySQL Documentation

Provides extensive information on MySQL setup, configuration, and usage.

## **Python and MySQL**

Python MySQL Connector Documentation

Official MySQL Connector/Python Documentation

Learn how to connect your Python application to a MySQL database.

Automate the Boring Stuff with Python by Al Sweigart

While this book covers a range of Python topics, it includes practical examples of database interaction.

## **Python and Tkinter**

Python GUI Programming with Tkinter by Alan D. Moore

This book covers the basics and advanced topics of creating GUI applications with Tkinter.

## **Tkinter Documentation**

Official Documentation

Provides details on how to use Tkinter for building GUIs in Python.

## **Project Tutorials and Examples**

1.GeeksforGeeks: Railway Reservation System Project

Railway Reservation System using Python and MySQL

2.YouTube Tutorials

Python Tkinter and MySQL Projects

There are many YouTube channels offering step-by-step tutorials on similar projects.

3.FreeCodeCamp: Tkinter Course

Tkinter Course - Create Graphic User Interfaces in Python Tutorial