# RAILWAY RESERVATION SYSTEM

A MINI PROJECT REPORT

SUBMITTED BY


| | |
|---|---|
| **ABHIGNYA P** | **2116221701002** |
| **ABINAUV R** | **2116221701003** |
| **RAHUL S** | **2116221701044** |
| **RITHIKA H** | **2116221701047** |


In partial fulfillment for the award of the degree of

**BACHELOR OF ENGINEERING**
**IN**
**COMPUTER SCIENCE AND DESIGN**


**RAJALAKSHMI ENGINEERING COLLEGE**
**THANDALAM**
**CHENNAI – 602105**



**ANNA UNIVERSITY: CHENNAI 600625**

# BONAFIDE CERTIFICATE

Certified that this project report **"RAILWAY RESERVATION SYSTEM"** is the Bonafide work of **"ABHIGNYA P (2116221701002), ABINAUV R (2116221701003**), **RAHUL S (2116221701044)** AND **RITHIKA H (2116221701047)"** who carried out the project work under my supervision.

**SIGNATURE**

**Mr.Uma Maheswar Rao MA.,MFA in Design .,**
Professor and Head,
Computer Science and Design,
Rajalakshmi Engineering College,
Thndalam, Chennai – 602105.

**SIGNATURE**

**Mr.Vijaykumar M.Tech.,**
Asst. Professor (SS),
Computer Science and Design,
Rajalakshmi Engineering College,
Thandalam, Chennai – 602105.

**EXTERNAL EXAMINER**

**INTERNAL EXAMINER**

## ACKNOWLEGEMENT

We are highly obliged in taking the opportunity to thank our Chairman **Mr. S. Meganathan**, Chairperson **Dr.Thangam Meganathan** and our Principal **Dr.S.N.Murugesan** for providing all the facilities which are required to carry out this project work.

We are ineffably indebted to our H.O.D **Mr.Uma Maheswar Rao M.A., MFA.,** for his conscientious guidance and encouragement to make this project a recognizable one.

We are extremely thankful to our faculty **Mr.Vijaykumar M.Tech.,** for his valuable guidance and indefatigable support and extend our heartfelt thanks to all the teaching and non-teaching staff of **Computer Science & Design department** who helped us directly or indirectly in the completion of this project successfully.

At last but not least gratitude goes to our friends who helped us compiling the project and finally to god who made all things possible.

Any omission in this brief acknowledgement doesn't mean lack of gratitude.

**ABHIGNYA** **2116221701002**

**ABINAUV R** **2116221701003**

**RAHUL S** **2116221701044**

**RITHIKA H** **2116221701047**

## ABSTRACT

The "**Railway Reservation System**" is designed to simplify and enhance the ticket booking process for passengers. Traditional manual methods were prone to delays, errors, and miscommunication, especially during high-demand periods. This system addresses these challenges by providing a centralized digital platform for managing reservations efficiently. Key features include real-time updates on seat availability, a simple booking process, and the ability to search and filter trains based on routes. Passengers can easily view their reservations, track booking status, and cancel tickets when necessary. By ensuring accurate and instant communication, the system minimizes errors and enhances transparency. Additionally , the platform optimizes seat allocation, ensuring fair and equitable distribution among passengers. With a focus on user experience, the interface is designed to be intuitive and accessible, making it suitable for all users. The system also supports robust database management to handle multiple concurrent operations seamlessly. This modernized approach revolutionizes railway ticketing, offering convenience, reliability, and efficiency to passengers while reducing the operational burden on railway authorities.

# TABLE OF CONTENTS


**Page No.**

**1. INTRODUCTION** **6**

1.1 Python

1.2 SQLite

**2. SYSTEM SPECIFICATION** **8**

2.1 Hardware and software specifications

**3. SOFTWARE DESCRIPTION** **9**

3.1 **STREAMLIT**

3.2 **FEATURES**

3.3 **PYTHON**

3.4 **VISUAL STUDIO CODE (VS Code)**

**4. PROJECT DESCRIPTION** **11**

4.1 Module Description

4.1.1 **HOME**

4.1.2 **BOOK TICKET**

4.1.3 **RESERVATION PAGE**

4.1.4 **CANCEL TICKET**

4.1.5 **CONTACT SUPERVISOR**

4.1.6 **PASSENGER**

4.1.7 **TRAIN**

**5. IMPLEMENTATION** **13**

5.1 Source code

5.2 Screen Shots

**6. CONCLUSION** **21**

**7. BIBLIOGRAPHY** **22**

# CHAPTER – 1

## INTRODUCTION

The project discussed here, is implemented using the concepts of PYTHON and SQL.

### 1.1 PYTHON

Python is a versatile, high-level programming language known for its simplicity, readability, and broad applicability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, making it a popular choice for developers worldwide. Python emphasizes code readability and efficiency, allowing developers to focus on problem-solving rather than syntax. For the **Railway Reservation System,** Python was chosen as the primary programming language due to its robust library ecosystem and ease of use. The system employs **Streamlit** , an open-source Python library, to create an intuitive and interactive web-based interface. Streamlit allows developers to build and deploy data-driven web applications with minimal effort, enhancing user experience and productivity. This system integrates Python's extensive database management capabilities with **SQLite**, ensuring efficient handling of passenger data, train schedules, and reservations. The combination of

Python and Streamlit enables real-time updates, dynamic visualizations, and user-friendly navigation. First introduced in 1991 by Guido van Rossum , Python has since become one of the most widely used programming languages, celebrated for its versatility and extensive community support. Its use in the Railway Reservation System demonstrates its ability to handle complex tasks, ensuring an efficient, reliable, and scalable solution for ticket booking and management.

**1.2 SQL (STRUCTURED QUERY LANGUAGE)**

SQL is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

Originally based upon relational algebra and tuple relational calculus. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control.

We are using **SQLite** for our project:

**SQLite** (Structured Query Language – Lite) is a lightweight, self-contained, serverless database engine used for efficient data management. It is integrated into our **Railway Reservation System** to store and handle passenger details, train schedules, reservations, and supervisor information.SQLite's minimal setup and robust functionality make it an ideal choice for ensuring data consistency and reliability. Its support for concurrent operations allows real-time updates to seat availability and bookings. With SQLite, the system achieves seamless database management, enhancing overall efficiency and performance.

# CHAPTER – 2

## SYSTEM SPECIFICATION

## 2.1 HARDWARE AND SOFTWARE SPECIFICATIONS

The following hardware and software are required:

| criterion | Description |
|---|---|
| **OS version** | Microsoft® Windows® 7/8/10 (32-bit or 64-bit) Mac® OS X® 10.10 (Yosemite) or higher, up to 10.13 (macOS High Sierra) GNOME or KDE desktop Linux (64 bit capable of running 32-bit applications)(GNU C Library (glibc) 2.19+) |
| **RAM** | 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator |
| **Disk space** | 2 GB of available disk space minimum,<br>4 GB recommended (500 MB for IDE + 1.5 GB for Android<br><br>SDK and emulator system image) |
| **Python version** | Python 3.13.0 |
| **Screen resolution** | 1280×800 minimum screen resolution |

# CHAPTER - 3

## SOFTWARE DESCRIPTION:

### 3.1 Python

Python is a versatile, high-level programming language known for its simplicity, readability, and broad applicability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, making it an ideal choice for developing web-based applications like the **Railway Reservation System**. Python's extensive library support ensures efficient integration with databases, frameworks, and visualization tools, enabling seamless backend and frontend development.

### 3.2 Streamlit

Streamlit is an open-source Python library specifically designed for creating interactive web applications with minimal effort. It is widely used for data-driven projects due to its simplicity and rapid deployment capabilities. Streamlit eliminates the need for extensive HTML or JavaScript coding, allowing developers to focus on functionality. Features such as live data updates, drag-and-drop widgets, and user-friendly interfaces make it a key tool for building the **Railway Reservation System**, ensuring real-time booking and database interactions.

### 3.3 Visual Studio Code (VS Code)

VS Code is a lightweight yet powerful source code editor developed by Microsoft. It supports a wide range of programming languages, including Python, and offers features like intelligent code completion, debugging, and Git integration. The editor's flexibility, combined with its vast library of extensions, provides an optimal environment for developing and testing applications. It was used in the **Railway Reservation System** project for writing, debugging, and maintaining Python and Streamlit code.

### 3.4 FEATURES

The following features were utilized in the development of the **Railway Reservation System**:

- Seamless Python and Streamlit integration for backend and frontend functionality.
- A user-friendly interface with real-time updates for train schedules and seat availability.
- Simplified debugging and code management through VS Code's built-in tools.
- Database management using SQLite for efficient storage and retrieval of reservations, train details, and passenger information.
- Rapid application development with Streamlit’s widget-based design and deployment capabilities.
- Cross-platform support, ensuring compatibility on Windows, macOS, and Linux systems.

Together, these tools and technologies provided a robust foundation for building an efficient and user-centric railway reservation system.

# CHAPTER - 4

## PROJECT DESCRIPTION

The **Railway Reservation System** is a web-based application developed using **Python** and **Streamlit** , designed to simplify train ticket booking and management. It allows users to search for trains, book tickets, view reservations, and cancel bookings with real-time updates on seat availability. The system is backed by an **SQLite** database that stores train schedules, passenger data, and reservation details. It offers a user-friendly interface for both passengers and administrators, enabling efficient management of bookings. The platform also includes features for viewing all reservations and contacting the supervisor for assistance. The project aims to enhance user experience and optimize train reservation operations.

## 4.1 MODULE DESCRIPTION

1. **Home**
2. **Book Ticket**
3. **Reservation Page**
4. **Cancel Ticket**
5. **Contact Supervisor**
6. **Passenger**
7. **Train**

### 4.1.1 HOME

The Home module displays a list of available trains, their details, and their current seat availability. Users can easily navigate through the platform and access other functionalities like booking tickets, viewing reservations, and more.

### 4.1.2 BOOK TICKET

This module allows passengers to book tickets by selecting a train, entering passenger details, and choosing available seats. It enables users to complete their booking and view confirmation details such as seat number and train schedule.

### 4.1.3 RESERVATION PAGE

The Reservation Page provides users with an overview of all their active

bookings. Passengers can view detailed information about each reservation, including train name, seat number, and travel dates.

### 4.1.4 CANCEL TICKET

This module allows passengers to cancel an existing reservation. Users can search for their reservation by ID and remove it from the system if necessary. It also updates the seat availability in real-time.

### 4.1.5 CONTACT SUPERVISOR

The Contact Supervisor module provides users with the contact information of the supervisor, including name, phone number, and email, for assistance with any booking or system-related issues.

### 4.1.6 PASSENGER

The Passenger module allows passengers to create profiles, store personal information, and manage their bookings, including adding new passengers for ticket reservations.

### 4.1.7 TRAIN

The Train module enables administrators to manage and update the train schedules, available seats, and other relevant details in the system. It ensures that real-time data is available for passengers to make informed decisions.

# CHAPTER - 5

## IMPLEMENTATION

### 5.1 Source code

```
import streamlit as st
import sqlite3
from datetime import datetime
import base64
import time

# Convert image to base64 for background
def get_base64_of_bin_file(file_path):
    with open(file_path, 'rb') as f:
        data = f.read()
    return base64.b64encode(data).decode('utf-8')

# Load and encode the background image
background_image =
get_base64_of_bin_file("C:/Users/rehit/OneDrive/Desktop/trains.jpg")

# Custom CSS for styling
st.markdown(
    f"""
    <style>
    .stApp {{
        background-image: url("{background_image}");
        background-size: cover;
        background-position: center;
        color: #FFFFFF;
    }}
    .sidebar .sidebar-content {{
        background-color: #2E86C1;
        color: white;
    }}
    .stButton > button {{
        background-color: #1ABC9C;
        color: white;
        border-radius: 5px;
        height: 40px;
        width: 100px;
    }}
    .stButton > button:hover {{
        background-color: #16A085;
        color: white;
```

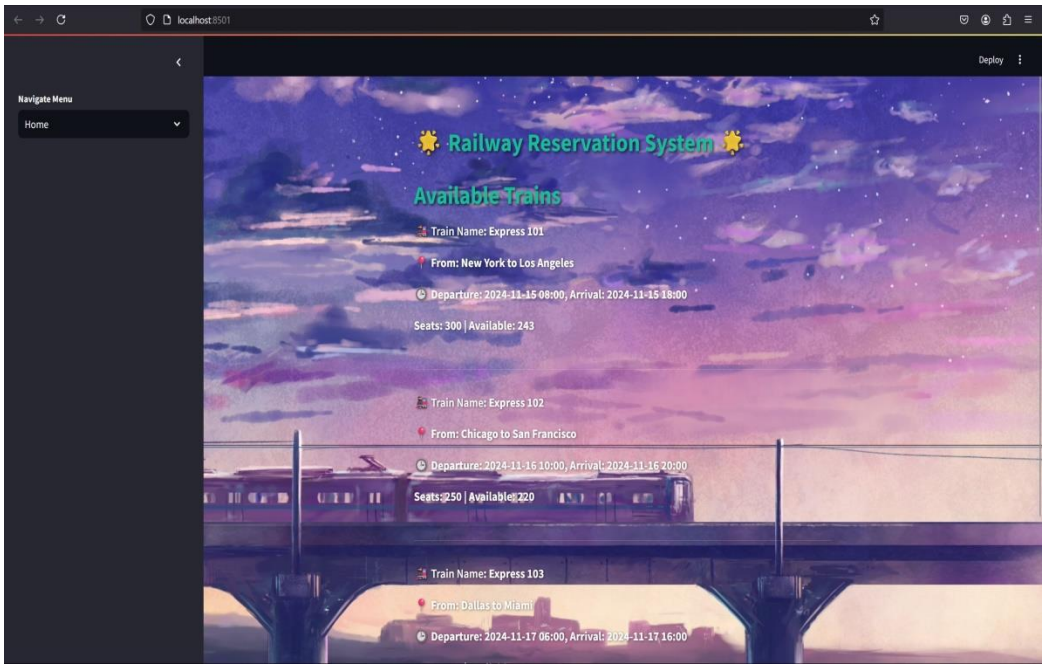
```
    }}
    h1, h2, h3, h4, h5, h6 {{
        color: #1ABC9C;
        font-weight: bold;
        text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.7);
        font-size: 36px; /* Larger font size */
    }}
    p, li {{
        font-weight: bold;
        text-shadow: 1px 1px 3px rgba(0, 0, 0, 0.5);
        font-size: 18px; /* Larger font size */
    }}
    </style>
    """,
    unsafe_allow_html=True,
)

# Enhanced Database connection with retry and WAL mode
def get_db_connection():
    retries = 10
    while retries > 0:
        try:
            conn = sqlite3.connect('railway.db', timeout=10)
            conn.row_factory = sqlite3.Row
            conn.execute("PRAGMA journal_mode=WAL;")
            conn.execute("PRAGMA busy_timeout = 3000;")
            return conn
        except sqlite3.OperationalError as e:
            if "database is locked" in str(e):
                time.sleep(0.5)
                retries -= 1
            else:
                raise
    raise Exception("Failed to acquire database connection after several
retries.")

# Function to fetch all trains
def get_all_trains():
    with get_db_connection() as conn:
        return conn.execute("SELECT * FROM train").fetchall()

# Function to fetch a specific train by ID
def get_train_by_id(train_id):
    with get_db_connection() as conn:
        return conn.execute("SELECT * FROM train WHERE id = ?",
(train_id,)).fetchone()

# Function to book a ticket
```

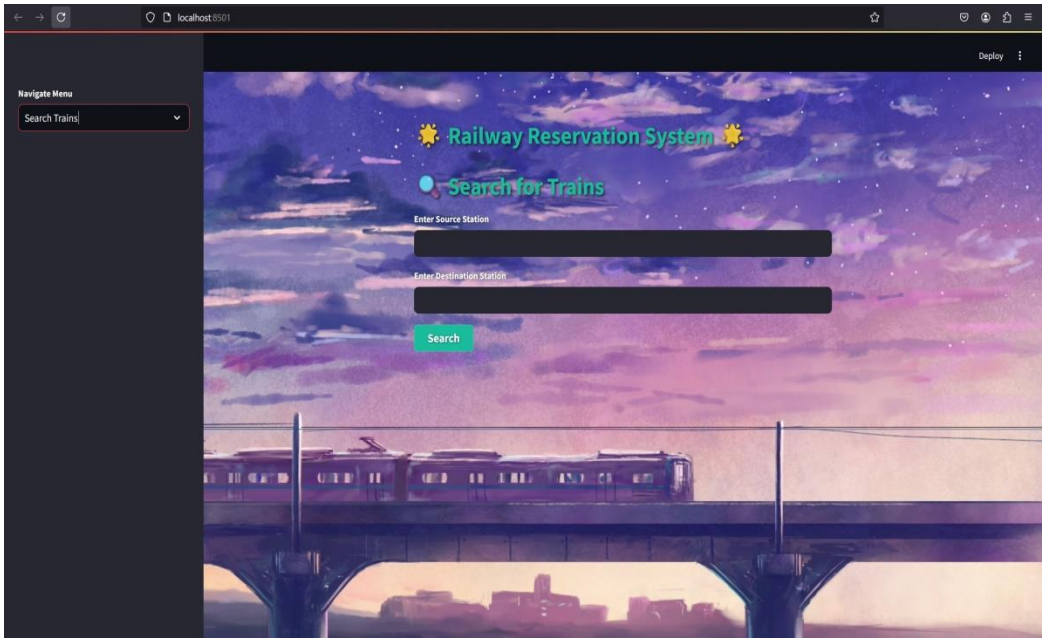
```
def book_ticket(train_id, passenger_id, seat_number, booking_date):
    with get_db_connection() as conn:
        conn.execute(
            "INSERT INTO reservation (train_id, passenger_id, seat_number,
booking_date) VALUES (?, ?, ?, ?)",
            (train_id, passenger_id, seat_number, booking_date)
        )
        conn.execute("UPDATE train SET available_seats = available_seats - 1
WHERE id = ?", (train_id,))
        conn.commit()

# Function to add a new passenger
def add_passenger(name, age, gender, contact_info):
    with get_db_connection() as conn:
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO passenger (name, age, gender, contact_info) VALUES
(?, ?, ?, ?)",
            (name, age, gender, contact_info)
        )
        return cursor.lastrowid

# Function to get reservations
def get_reservations():
    with get_db_connection() as conn:
        return conn.execute("SELECT * FROM reservation").fetchall()

# Function to check if a reservation exists
def check_reservation_exists(reservation_id):
    with get_db_connection() as conn:
        return conn.execute("SELECT * FROM reservation WHERE id = ?",
(reservation_id,)).fetchone()

# Function to get supervisor information
def get_supervisor_info():
    with get_db_connection() as conn:
        return conn.execute("SELECT * FROM supervisor").fetchone()

# Streamlit App
st.title("🚆 Railway Reservation System 🚆")

menu = ["Home", "Search Trains", "Book Ticket", "Cancel Ticket", "View
Reservations", "Contact Supervisor"]
choice = st.sidebar.selectbox("Navigate Menu", menu)

if choice == "Home":
    st.header("Available Trains")
    trains = get_all_trains()
```

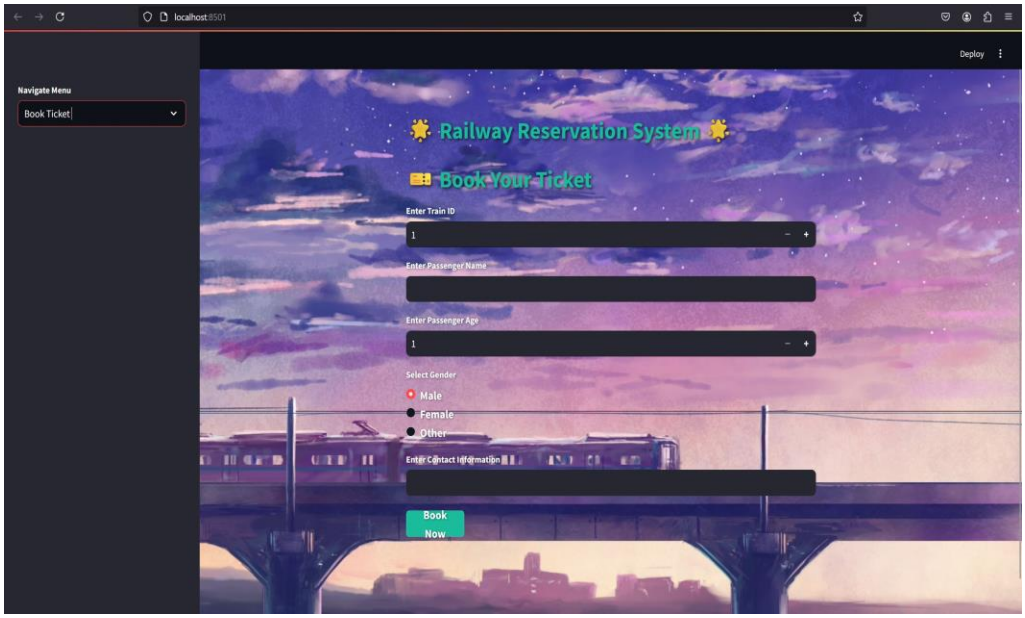
```
    for train in trains:
        st.write(f"🚆 **Train Name**: {train['name']}")
        st.write(f"📍 From: {train['source']} to {train['destination']}")
        st.write(f"🕒 Departure: {train['departure_time']}, Arrival:
{train['arrival_time']}")
        st.write(f"Seats: {train['total_seats']} | Available:
{train['available_seats']}")
        st.markdown("---")

elif choice == "Search Trains":
    st.header("🔍 Search for Trains")
    source = st.text_input("Enter Source Station")
    destination = st.text_input("Enter Destination Station")
    if st.button("Search"):
        with get_db_connection() as conn:
            query = "SELECT * FROM train WHERE source = ? AND destination = ?"
            results = conn.execute(query, (source, destination)).fetchall()
        if results:
            for train in results:
                st.write(f"🚆 **Train Name**: {train['name']} | Seats
Available: {train['available_seats']}")
        else:
            st.error("No trains found!")

elif choice == "Book Ticket":
    st.header("🎟 Book Your Ticket")
    train_id = st.number_input("Enter Train ID", min_value=1)
    passenger_name = st.text_input("Enter Passenger Name")
    passenger_age = st.number_input("Enter Passenger Age", min_value=1)
    passenger_gender = st.radio("Select Gender", ["Male", "Female", "Other"])
    contact_info = st.text_input("Enter Contact Information")

    if st.button("Book Now"):
        passenger_id = add_passenger(passenger_name, passenger_age,
passenger_gender, contact_info)
        train = get_train_by_id(train_id)

        if train and train['available_seats'] > 0:
            seat_number = train['total_seats'] - train['available_seats'] + 1
            booking_date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            book_ticket(train_id, passenger_id, seat_number, booking_date)
            st.success(f"🎉 Ticket booked! Your Seat Number is {seat_number}.")
        else:
            st.error("No seats available.")

elif choice == "Cancel Ticket":
    st.header("❌ Cancel a Ticket")
    reservation_id = st.number_input("Enter Reservation ID", min_value=1)
```

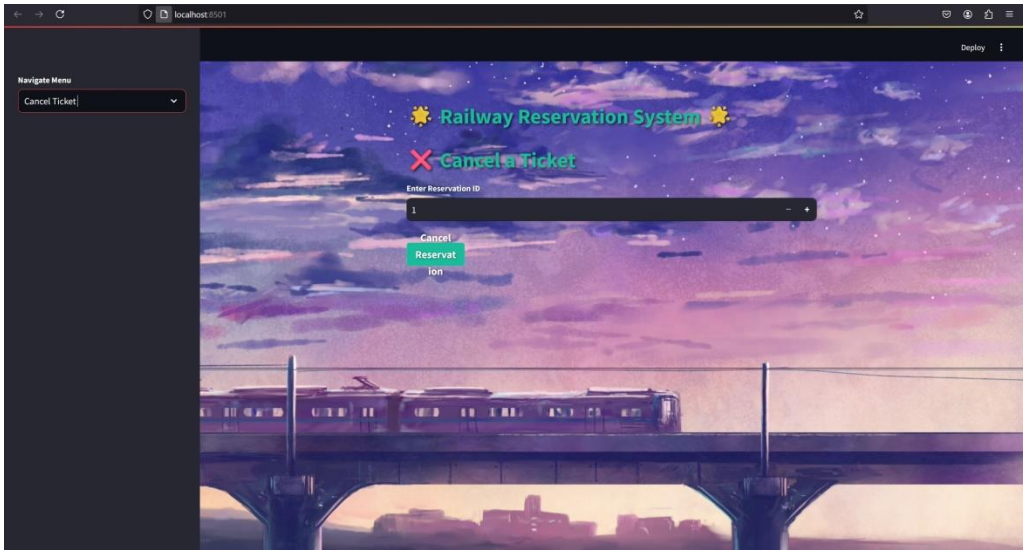
```
    if st.button("Cancel Reservation"):
        reservation = check_reservation_exists(reservation_id)
        if reservation:
            with get_db_connection() as conn:
                conn.execute("DELETE FROM reservation WHERE id = ?",
(reservation_id,))
                conn.commit()
            st.success("Reservation cancelled successfully.")
        else:
            st.error("Reservation ID does not exist.")

elif choice == "View Reservations":
    st.header("📋 All Reservations")
    reservations = get_reservations()
    if reservations:
        for res in reservations:
            st.write(f"🎟️ Reservation ID: {res['id']} | Train ID:
{res['train_id']} | Seat Number: {res['seat_number']}")
            st.markdown("---")
    else:
        st.info("No reservations found.")

elif choice == "Contact Supervisor":
    st.header("📞 Contact Supervisor")
    supervisor = get_supervisor_info()
    if supervisor:
        st.write(f"👨‍💼 Name: {supervisor['name']}")
        st.write(f"📱 Phone: {supervisor['phone']}")
        st.write(f"📧 Email: {supervisor['email']}")
    else:
        st.info("Supervisor information not found.")
```

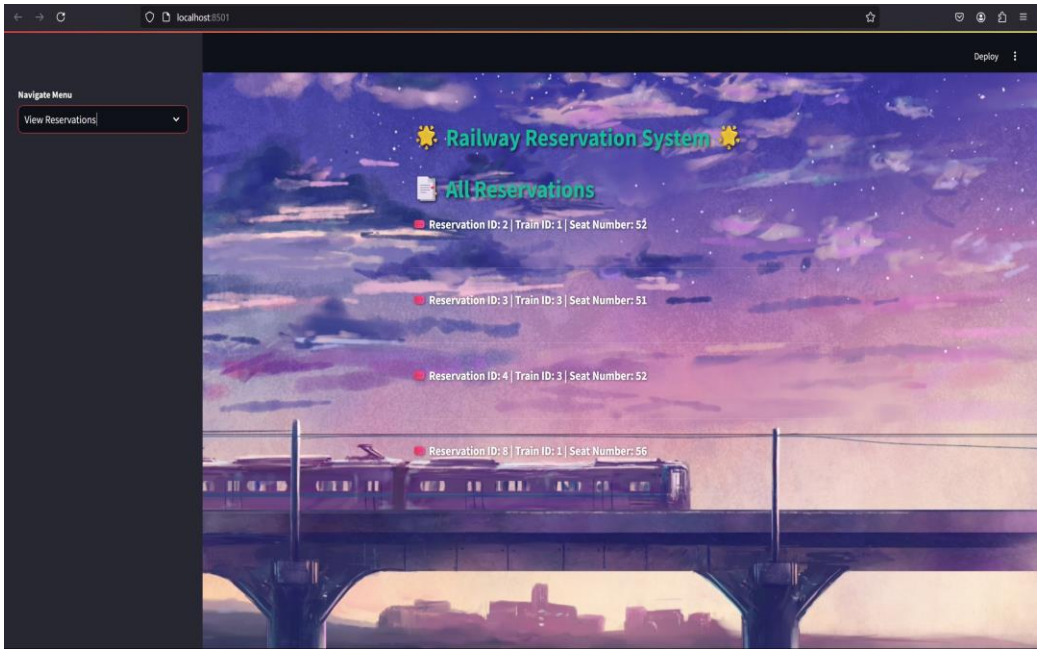
## 5.2 Screen shots

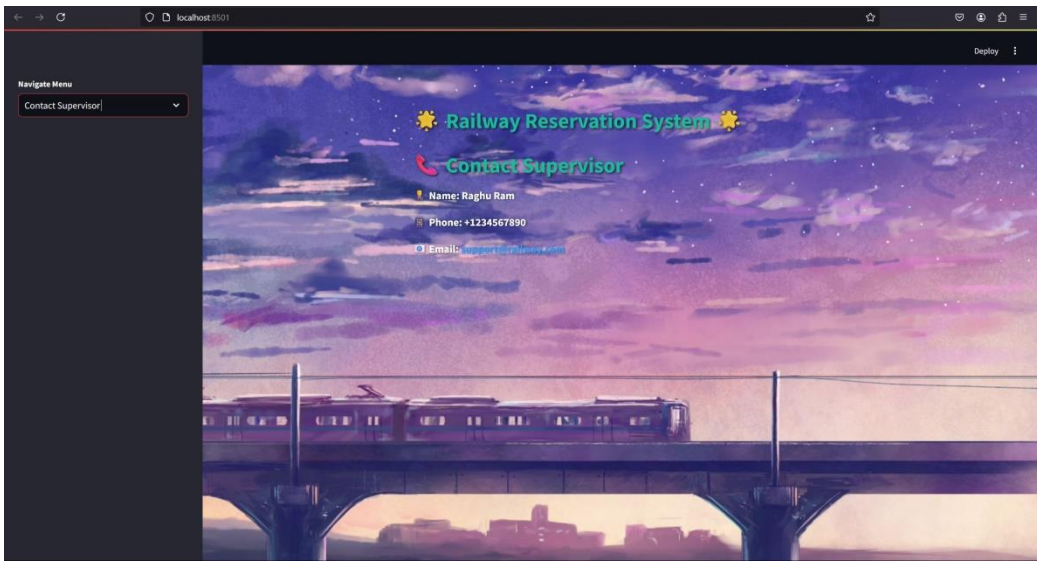



localhost:8501
Deploy
Navigate Menu
Book Ticket
Railway Reservation System
Book Your Ticket
Enter Train ID
1
Enter Passenger Name
Enter Passenger Age
1
Select Gender
Male
Female
Other
Enter Contact Information
Book Now


localhost:8501
Deploy
Navigate Menu
Cancel Ticket
Railway Reservation System
Cancel a Ticket
Enter Reservation ID
1
Cancel Reservation


localhost:8501
Deploy
Navigate Menu
View Reservations
Railway Reservation System
All Reservations
Reservation ID: 2 | Train ID: 1 | Seat Number: 52
Reservation ID: 3 | Train ID: 3 | Seat Number: 51
Reservation ID: 4 | Train ID: 3 | Seat Number: 52
Reservation ID: 8 | Train ID: 1 | Seat Number: 56


localhost:8501
Deploy
Navigate Menu
Contact Supervisor
Railway Reservation System
Contact Supervisor
Name: Raghu Ram
Phone: +1234567890
Email:

# CHAPTER 6

## CONCLUSION

Technology has greatly enhanced the efficiency and convenience of various systems, offering improved accuracy, security, and ease of maintenance. In the case of the **Railway Reservation System** , the integration of modern technologies like Python and Streamlit has transformed traditional train booking methods, making them faster, more reliable, and user-friendly. With features such as real-time seat availability, easy booking and cancellation, and an intuitive interface, this system provides a seamless experience for both passengers and administrators. As technology continues to evolve, such systems will become even more efficient, providing cost-effective solutions that meet the needs of users while simplifying management tasks for railway operators.

# CHAPTER – 7

## BIBLIOGRAPHY:

Website references

http://www.youtube.com

http://www.wikipedia.com