ISSN: 2834-4928

Research Article

Journal of Electrical Electronics Engineering

Creating a GUI-Based Train Ticket Booking System in Python using MySQL

Jishnu Teja Dandamudi*

Amrita School of Artificial Intelligence, Coimbatore, Amrita Vishwa Vidyapeetham, India

***Corresponding Author**
Jishnu Teja Dandamudi, Amrita School of Artificial Intelligence, Coimbatore, Amrita Vishwa Vidyapeetham, India.

Submitted: 2025, May 12; **Accepted:** 2025, Jun 02; **Published:** 2025, Jun 11

Citation: Dandamudi, J. T. (2025). Creating a GUI-Based Train Ticket Booking System in Python using MySQL. *J Electrical Electron Eng, 4*(3), 01-10.

Abstract

In today's digital era, students often resort to replicating pre-existing projects, limiting opportunities for innovation and genuine learning. This project aims to design and develop a novel Train Ticket Booking System using Python's Tkinter for the front-end and MySQL for the back-end. The system is a self-driven initiative, created without referencing any prior implementations, with the objective of learning new skills and addressing real-world challenges in railway reservation systems. The project seeks to digitize and streamline the processes involved in railway inquiry and ticket reservation. In Admin Mode, the system enables functionalities such as adding trains, assigning seats, defining train routes, generating reservation charts, and deleting trains. In User Mode, it allows ticket booking, PNR inquiry, and viewing booking history. The software supports storing and managing multiple train records while offering intuitive options for modifications. Key features include detailed train information, such as travel duration, arrival and departure timings, and historical ticket bookings. By leveraging a structured database, the system enhances data integrity and retrieval efficiency, offering a comprehensive solution for railway management. This project not only demonstrates practical application development but also emphasizes learning through creation, contributing to the broader goal of fostering innovation in student-led initiatives.

Keywords: Python, Python Tkinter GUI, MySQL, Text Files, File Handling, DDL, DML

1. Introduction

• Interest

The interest that made me do this paper is to create an app like structure cum interface for train ticket booking using pure Python.

• Python

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by

Guido van Rossum and first released in 1991. Python is widely used for web development, data analysis, scientific computing, artificial intelligence, machine learning, automation, and more.

• Key Features of Python

Easy to Learn and Use: Python has a simple syntax similar to English, making it accessible for beginners.

Interpreted Language: Python executes code line by line, which makes debugging easier.

High-Level Language: Python abstracts complex details of the computer's operations, allowing developers to focus on programming logic.

Dynamic Typing: Variable types are determined at runtime, eliminating the need for explicit declarations.

Extensive Libraries: Python has a rich set of libraries and frameworks, such as NumPy, Pandas, TensorFlow, Django, and Flask.

Cross-Platform: Python runs on various platforms (Windows, macOS, Linux)

Community Support: Python has a large, active community, ensuring plenty of resources, documentation, and support.

• Common Uses of Python

Web Development: Using frameworks like Django and Flask.

Data Science and Machine Learning: With libraries like NumPy, Pandas, Scikit-learn, and TensorFlow.

Automation/Scripting: Writing scripts to automate repetitive tasks.

Game Development: Using libraries like Pygame.
Scientific and Numeric Computing: With tools like SciPy and Matplotlib.
Software Development: For building desktop and mobile applications.
Internet of Things (IoT): Using Raspberry Pi and other microcontrollers.

• MySQL

MySQL is a popular open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage and interact with databases. It was originally developed by MySQL AB in 1995 and later acquired by Oracle Corporation in 2010.

• Key Features of MySQL

Relational Database: MySQL organizes data into tables consisting of rows and columns, making it easy to relate and query data.
SQL Support: It uses SQL as the standard language for querying and managing data.
Open Source: MySQL is freely available under the GNU General Public License (GPL), with additional commercial versions offered by Oracle.
Cross-Platform: Runs on various operating systems, including Windows, macOS, Linux, and Unix.
Scalability: Suitable for small-scale applications and large-scale systems with high-volume data.
Security: Features like user authentication, data encryption, and access control ensure robust security.
High Performance: Optimized for speed and efficiency in data transactions and complex queries.
Replication: Supports database replication for backup, high availability, and load balancing.

• Common Uses of MySQL

Web Applications: Powers databases for websites and applications like WordPress, Facebook, and Twitter.
E-Commerce: Supports online stores by managing product catalogs, user information, and transactions.
Data Warehousing: Used to analyze large datasets.
Content Management Systems (CMS): Integrates with CMS platforms like Drupal, Joomla, and Magento.
Enterprise Applications: Manages data for internal systems like CRM or ERP tools.

• How MySQL Works

Server-Client Model: MySQL operates as a server and listens to requests from client applications.
Storage Engine Architecture: Supports multiple storage engines, like InnoDB (for transactions and data integrity) and MyISAM (optimized for speed).
SQL Queries: Allows users to create, read, update, and delete data using SQL commands.

• Why Use MySQL

It is fast, reliable, and easy to use.
Integrates seamlessly with programming languages like PHP, Python, and Java.
Provides excellent community support and documentation.

Refer to Figure 1.

2. Working Methodology

A. Backend - MySQL

• Create a database with name rrs
• Create Table 1 with the following data storage attributes as shown in Figure 2
• Create Table 2 with the following data storage attributes as shown in Figure 3

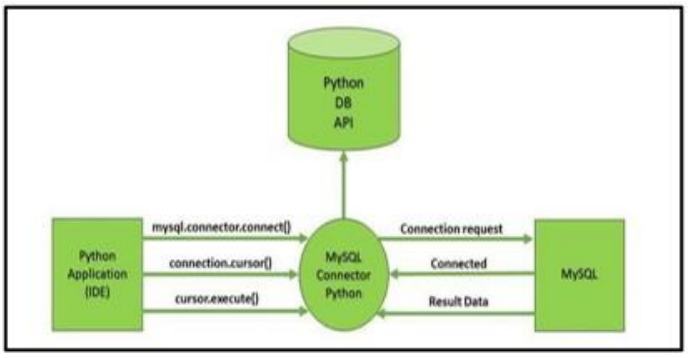

Figure 1: MySQL with Python

B. Admin Mode

• Add a Train

The Python code implements a GUI-based train management system using Tkinter and a MySQL database, allowing admins to add new train details interactively. Admins can input the train number, name, and running days through a user-friendly interface with labeled entry fields. The program checks if the train number already exists in the database to prevent duplicates; if it doesn't, it dynamically creates a unique table for the train to store station details such as arrival and departure times. The train details, including its number, name, and running days, are then stored in a main trains table. The GUI provides feedback through success

or error messages, ensuring smooth user interaction. The design features a consistent aesthetic with a defined color scheme and font style, and the system supports scalability by dynamically handling new train additions without predefining their structure.

• Assign Seats

The program provides a graphical user interface (GUI) using tkinter for assigning seat allocations to a specific train number in a railway database. Admins input the train number and specify the number of coaches for Sleeper (SL), Second AC (2A), and Third AC (3A). The inputs are validated to ensure they are numeric and non-negative. Upon submission, the program updates the database with the entered values. Feedback is displayed to inform the user of successful updates or any errors encountered during the operation. The GUI employs labels, entry fields, and buttons for interaction, with clear error messages and success notifications using messagebox.

• Add Train Route

The code implements a Tkinter-based GUI application for managing train route details, allowing admins to input and store information such as station codes, station names, arrival times, and departure times into a MySQL database. The interface includes input fields for train number and other route details, validating train numbers against existing database tables. It employs dynamic SQL

```
mysql> desc trains;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| train_no     | int          | NO   | PRI | NULL    |       |
| train_name   | char(50)     | YES  |     | NULL    |       |
| running_days | char(200)    | YES  |     | NULL    |       |
| coaches      | varchar(500) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

Figure 2: Trains Table Structure in MySQL

```
mysql> desc user_bookings;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| username       | char(30)    | YES  |     | NULL    |       |
| start_datetime | date        | YES  |     | NULL    |       |
| end_datetime   | date        | YES  |     | NULL    |       |
| train_no       | int         | YES  |     | NULL    |       |
| pnr            | int         | YES  |     | NULL    |       |
| booking_status | char(30)    | YES  |     | NULL    |       |
| current_status | char(30)    | YES  |     | NULL    |       |
| class          | char(30)    | YES  |     | NULL    |       |
| name           | varchar(30) | YES  |     | NULL    |       |
| age            | int         | YES  |     | NULL    |       |
| gender         | char(30)    | YES  |     | NULL    |       |
| board          | char(30)    | YES  |     | NULL    |       |
| destination    | char(30)    | YES  |     | NULL    |       |
| status         | char(50)    | YES  |     | NULL    |       |
| train_name     | char(30)    | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
15 rows in set (0.02 sec)
```

Figure 3: User Bookings Table Structure in MySQL

queries to insert data into the respective train's route table while providing user feedback on successful operations. Enhancements such as input validation, error handling for database interactions, and better UI layout are used to improve robustness, user experience, and security.

• Reservation Chart

The code defines a Tkinter-based GUI application for displaying a train reservation chart. The reservationChart() function creates a new window where admins can input a train number and submit it to fetch reservation details. Upon submission, the reservation_details() function queries a database (user_bookings) to retrieve and display passenger information, including name, gender, age, PNR, berth, boarding, and destination details, formatted in a readable layout. Error handling is included to manage invalid inputs or database issues, and the UI dynamically updates for new inputs. An "Exit" button allows users to close the reservation chart window.

• Delete a Train

The deleteTrain function provides a graphical interface for deleting train records from a database. Admins input the train number, which is validated to ensure it is numeric and corresponds to an existing record. The system confirms the deletion action before proceeding. The deleting_train function performs two key operations: removing the train's associated table and deleting its entry from the main trains table. Input validation, error handling, and user feedback are implemented to enhance security and usability. If errors occur, appropriate messages are displayed. The design includes a success message upon completion and an exit button to close the window.

C. User Mode

• Book a Ticket

The provided code defines a Tkinter-based GUI application for booking train tickets in a railway reservation system. It allows users to find trains between a boarding and destination station for a specified date, check train availability based on running days, and book tickets. The system dynamically queries a MySQL database to retrieve train schedules, routes, and seat availability. After booking, it generates a PNR and assigns a booking status, confirming the reservation. The ticket details, including journey information, passenger details, and booking status, are saved to a text file and displayed to the user. The program also ensures basic validations such as date and route consistency while incorporating dynamic SQL queries for flexibility.

• PNR Enquiry

The program is a GUI-based PNR Checker implemented using tkinter. It allows users to input a PNR number and retrieve booking details from a database. Upon entering a PNR, the application queries the database to fetch associated details. If the PNR is marked as "cancelled," a cancellation message is displayed. Otherwise, it shows train details, passenger information, booking status, and current status in the interface. The interface includes labels for displaying messages and a button to close the window. Enhancements include error handling for invalid inputs and database exceptions, organized UI with responsive labels, and secure parameterized queries to prevent SQL injection.

• Cancel Ticket

The program is a tkinter-based GUI application for canceling train tickets. It provides a window where users can enter a PNR to cancel a ticket. On submitting the PNR, it updates the database to mark the ticket's status and current status as "Cancelled" using SQL queries. The interface confirms successful cancellation through a message and offers an "Exit" button to close the cancellation window. The application uses parameterized SQL queries to ensure security and a responsive design for ease of use.

• Booking History

The program is a tkinter-based GUI application to display the booking history of users. It queries the database to retrieve all records from the user_bookings table and formats the information into a readable structure. Each booking's details, including username, train information, journey dates, PNR, passenger details, and ticket status, are written to a text file named booking history. txt. Once the data is saved, the text file is automatically opened using the default system application. A confirmation message is displayed in the GUI, and an "Exit" button allows users to close the window. The program could benefit from input validation, better error handling, and pagination for large datasets.

3. Result

A. Main Interface:

Refer to Figure 4

B. Admin Interface:

Refer to Figure 5

C. Admin Entering Details before User booking a ticket:

1) Add a Train:
Refer to Figure 6
2) Assign Seats:
Refer to Figure 7
3) Add Train Route:
Refer to Figure 8

D. User Interface

Refer to Figure 9

Book a Ticket:

Refer to Figure 10
Refer to Figure 11

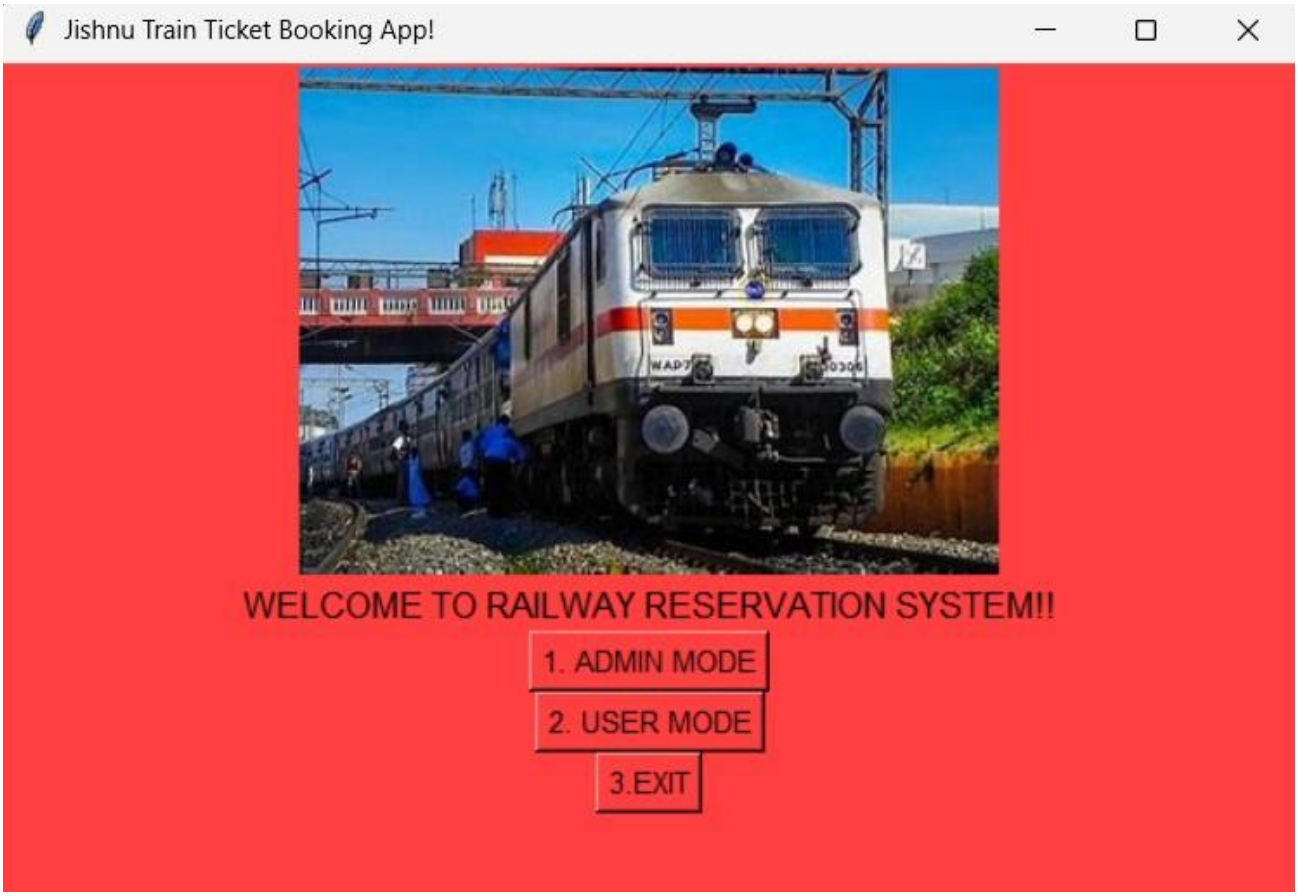

Figure 4: Main Interface

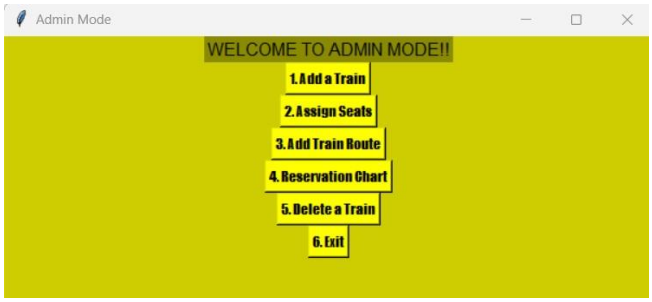

Figure 5: Admin Mode

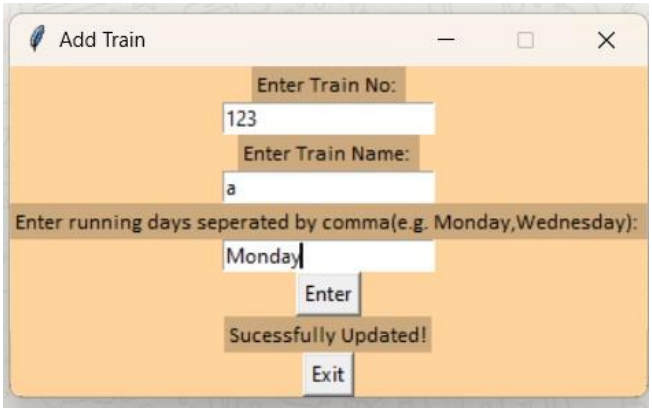

Figure 6: Add a Train

Refer to Figure 12

• **PNR Enquiry**

Refer to Figure 13

• **Booking History**

Refer to Figure 14

• **Cancel Ticket**

Refer to Figure 15

Figure 6: Add a Train

E. Admin Finding Details after User booking a ticket

• **Reservation Chart**

Refer to Figure 16

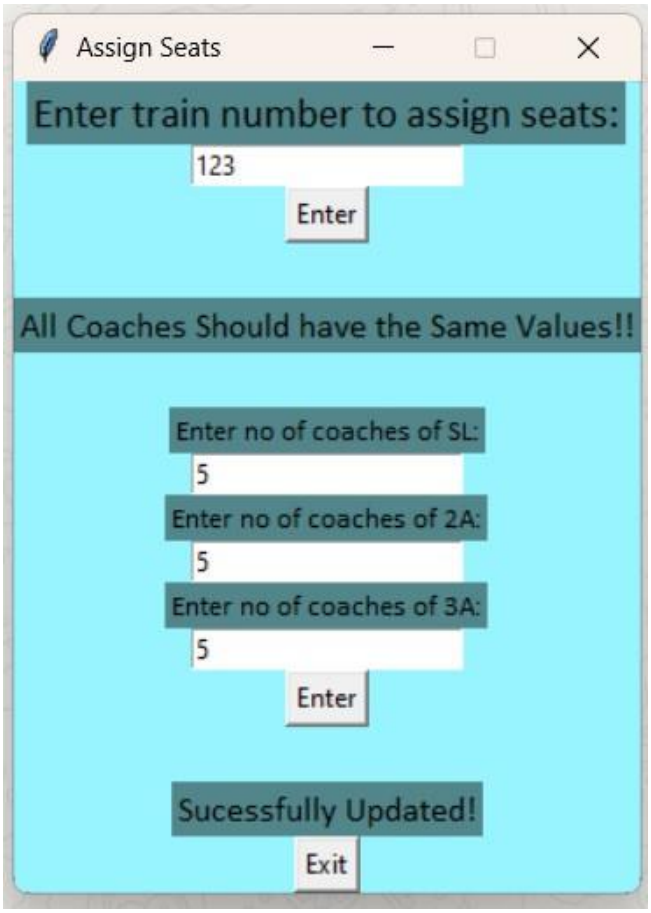

Figure 7: Assigning Seats

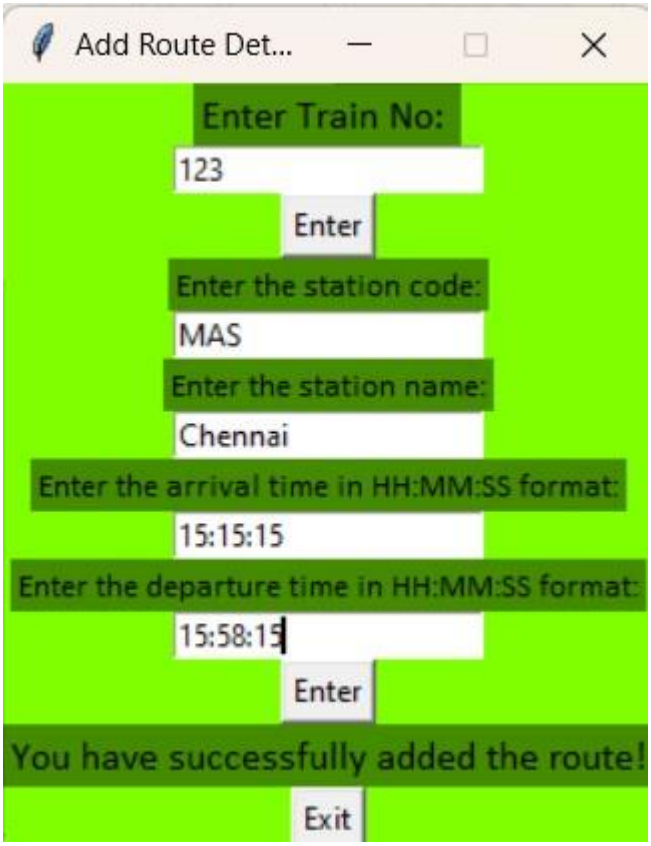

Figure 8: Add Train Route

Figure 9: User Mode

Delete a Train
Refer to Figure 17

F. MySQL updation

• Database after train creation by admin
Refer to Figure 18
• Description of the new train created in database
Refer to Figure 19
• User Bookings after booking ticket from user mode in Figure 9.
User Mode database
Refer to Figure 20
• User Bookings after cancelling the booked ticket from user mode in database
Refer to Figure 21
• Database after admin deleting the train
Refer to Figure 22

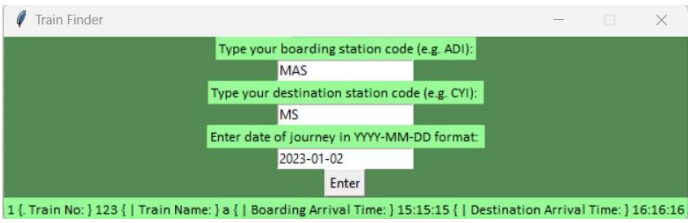

Figure 10: Trains Enquiry

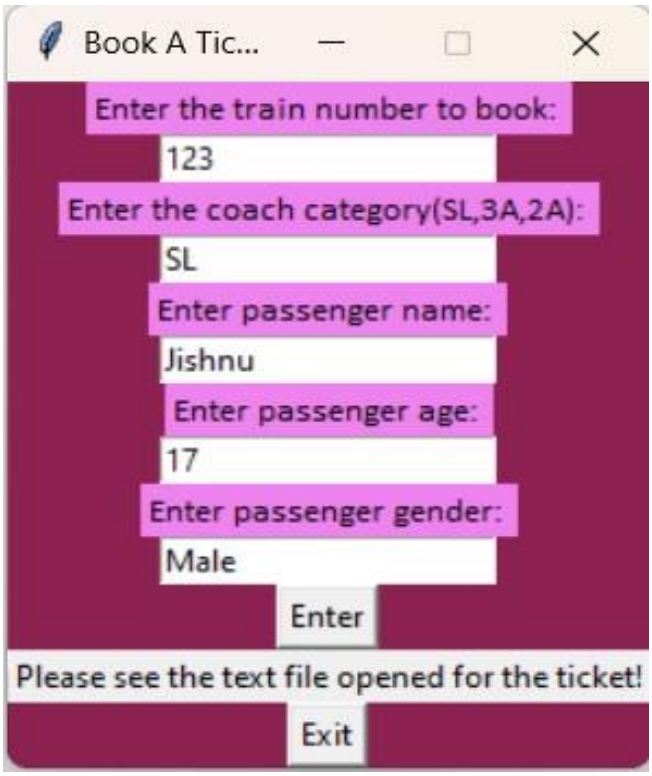

Figure 11: User Booking Ticket

Figure 13: PNR

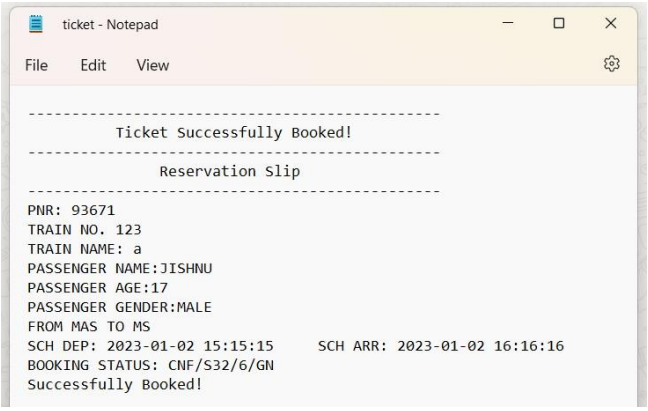

Figure 12: Confirmed Ticket in Text Format

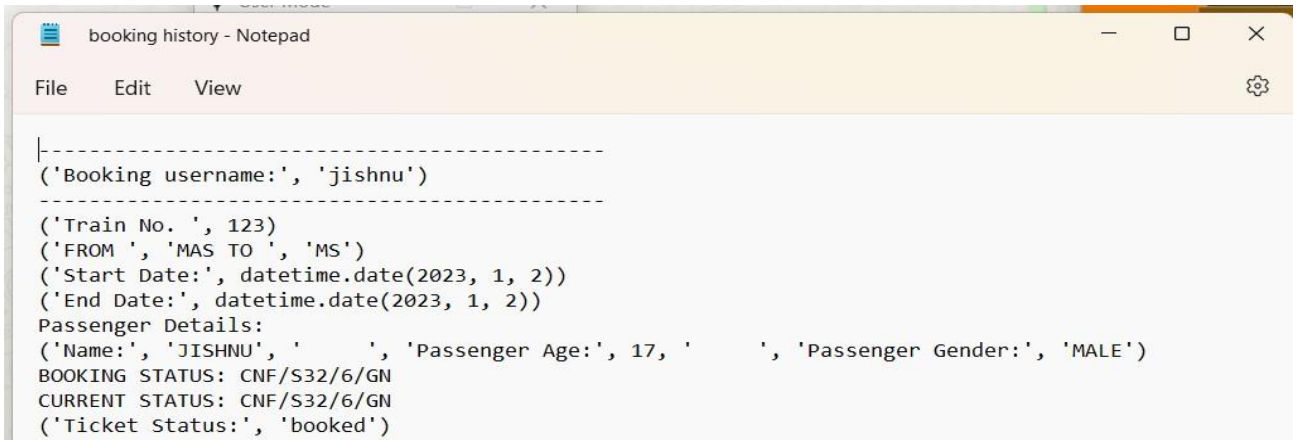

Figure 14: Booking History

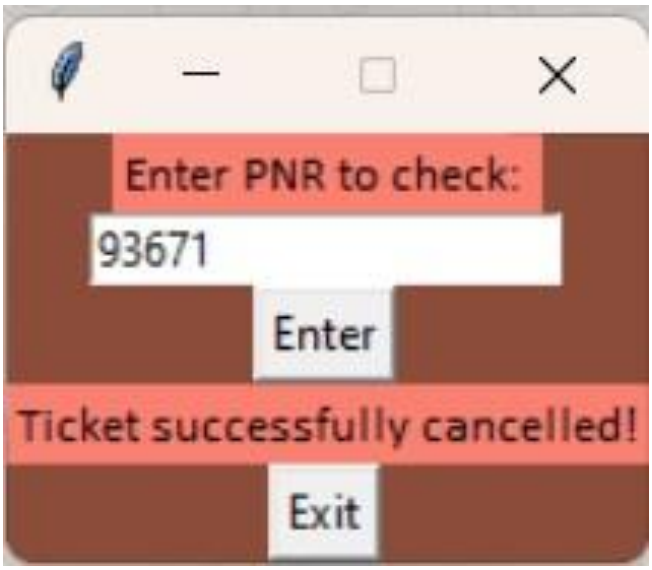

Figure 15: Cancel Ticket

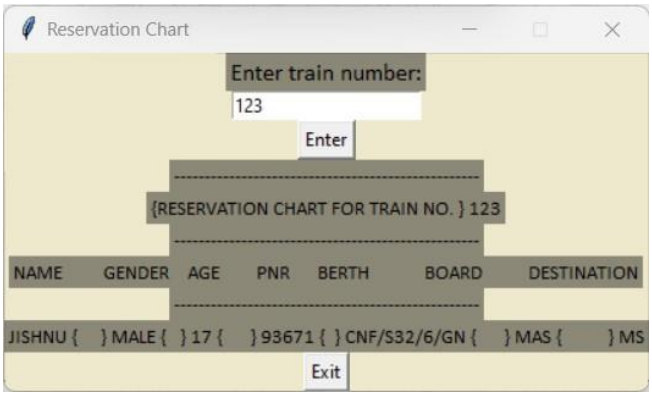

Figure 16: Reservation Chart

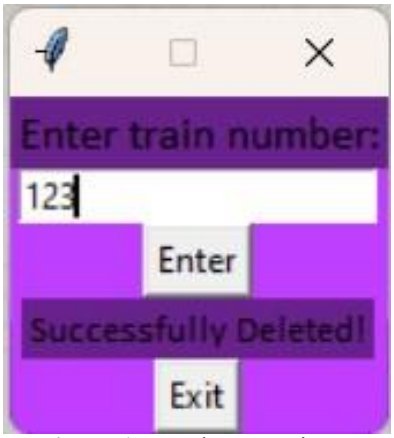

Figure 17: Delete a Train

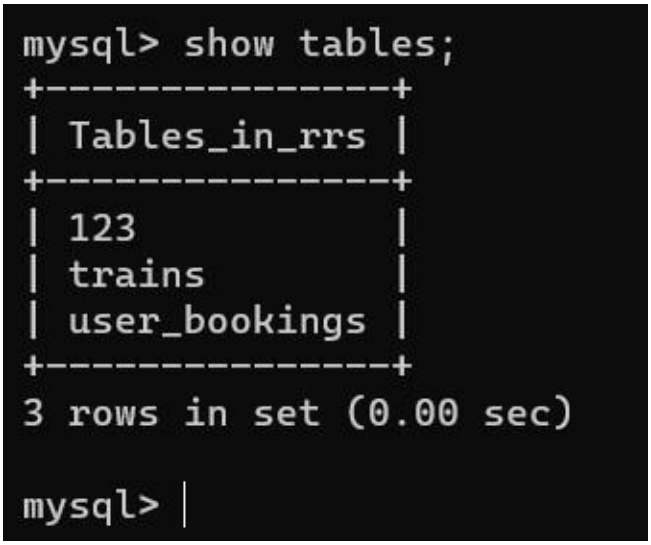

Figure 18: Database After Train Creation by Admin

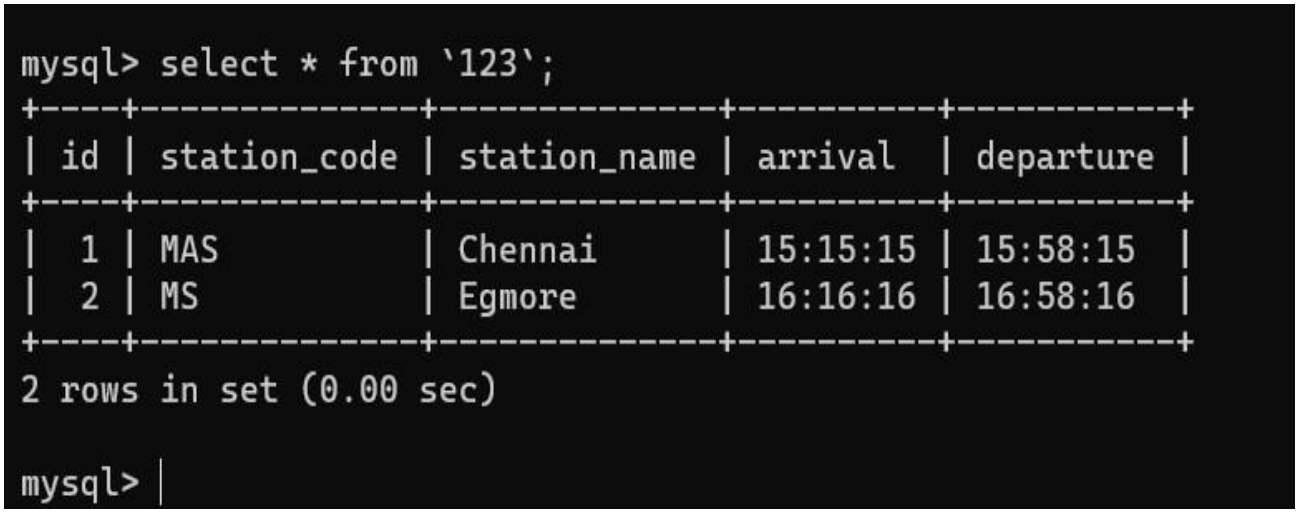

Figure 19: Description of the New Train Created in Database

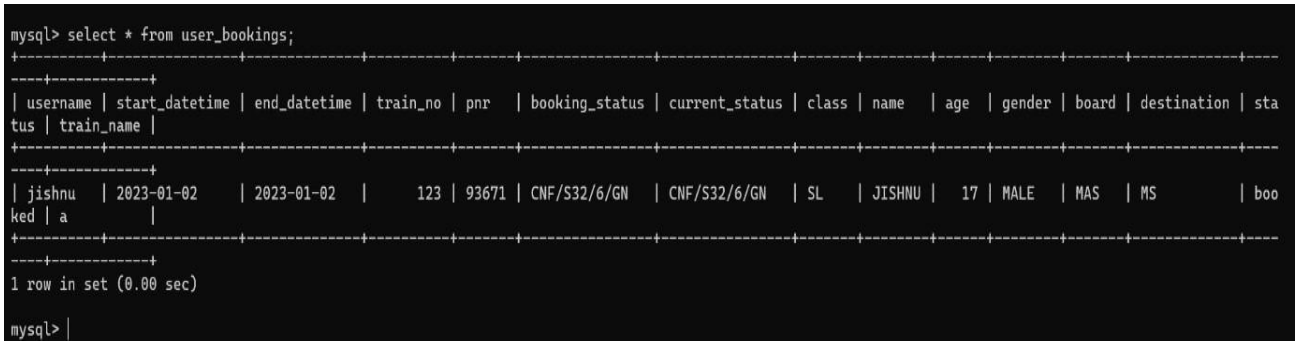

Figure 20: User Bookings After Booking Ticket from User Mode in Database

```
mysql> select * from user_bookings;
+----------+----------------+--------------+----------+-------+----------------+----------------+-------+--------+------+--------+-------+-------------+----
-------+------------+
| username | start_datetime | end_datetime | train_no | pnr   | booking_status | current_status | class | name   | age  | gender | board | destination | sta
tus    | train_name |
+----------+----------------+--------------+----------+-------+----------------+----------------+-------+--------+------+--------+-------+-------------+----
-------+------------+
| jishnu   | 2023-01-02     | 2023-01-02   |      123 | 93671 | CNF/S32/6/GN   | CNF/S32/6/GN   | SL    | JISHNU |   17 | MALE   | MAS   | MS          | can
celled | a          |
+----------+----------------+--------------+----------+-------+----------------+----------------+-------+--------+------+--------+-------+-------------+----
-------+------------+
1 row in set (0.00 sec)

mysql>
```

Figure 21: User Bookings After Cancelling the Booked Ticket from User Mode in Database

```
mysql> show tables;
+---------------+
| Tables_in_rrs |
+---------------+
| trains        |
| user_bookings |
+---------------+
2 rows in set (0.00 sec)

mysql>
```

Figure 22: Database After Admin Deleting the Train

4. Conclusion

Enquiry cum Reservation System can be used by railway institutions to maintain their record of train easily. Achievinthis objective is difficult using the manual system as the information is scattered, can be redundant, and collecting relevant information may be very time consuming [1-4]. All these problems are solved by this project. This system helps in maintaining the information of trains of the organization. It can be easily accessed by any higher authority who has been given access by the admin and kept safe for a longer period of time without any change from the user.

Author Contribution

The authors has created a extensive database and GUI for stroing the train informations, ticket booking, etc and had
shown how the ticket booking is working in the backend for the users.

References

1. Christudas, B., & Christudas, B. (2019). MySQL. Practical Microservices Architectural Patterns: Event-Based Java Microservices with Spring Boot and Spring Cloud, 877-884.
2. Van Rossum, G., & Drake, F. L. (2003). An introduction to Python (p. 115). Bristol: Network Theory Ltd.
3. Triani, R. A., & Schouten, F. S. (2023). Analysis of The Application of Online Ticket Booking Application Access By KAI. *Journal of Business Studies and Management Review, 7*(1), 132-138.
4. Van Rossum, G., & Drake Jr, F. L. (1995). Python tutorial (Vol. 620). Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica.