Project Report: Simple Train Management System

Field	Detail
Project Title	Simple Train Management System (STMS)
Developed By	Pawnesh Bhatt
Registration Number	25BCE10744
Course	B.Tech. Computer Science and Engineering, First Year
Submission Date	21 November,2025
Programming Language	Python

1. Abstract

This project, the Simple Train Management System (STMS), was developed as a core requirement for the first-year computer programming course. The primary goal was to apply foundational programming concepts, including functions, data structures (lists and dictionaries), to solve a real-world data management problem. The STMS is a console-based application that allows users to book tickets, check availability, and display the current status of the trains in a simulated environment. The system successfully demonstrates proficiency in modular programming and basic data persistence using local storage.

2. Introduction

2.1 Problem Statement

Efficiently managing data for a small number of trains—including their numbers, names, routes, and seat availability—requires a structured approach. Without a system, tracking bookings and capacity manually is prone to errors. The project addresses the need for a simple, automated solution to handle these basic operations via an interactive console interface.

2.2 Objectives

1. To design and implement a functional, menu-driven command-line interface (CLI) using Python.
2. To utilize fundamental Python data structures (Lists and Dictionaries) for storing and retrieving train and booking information.
3. To apply modular programming by breaking down functionality into dedicated functions.
4. To implement core features like adding new trains, booking tickets

3. System Design and Analysis

3.1 Technology Stack

- **Programming Language:** Python 3.x (chosen for its readability and simplicity, making it ideal for a first-year project).
- **Data Structure:** Lists of Python Dictionaries (to represent a collection of trains, where each train is an object/dictionary with structured attributes).

- **User Interface:** Command Line Interface (CLI).

3.2 System Modules and Features

The STMS is structured around a main loop and several functions, each handling a specific use case:

Module	Description	Key Functionality
Main Menu	Provides the user interface and navigates between features.	Display menu, handle user input.
Train Management	Manages the static data of the trains.	Add New Train, Delete Train Record.
Booking & Availability	Handles transaction-based changes to train data.	Book Ticket (decrements available seats), Check Availability.
Reporting	Displays current system information.	View All Trains, Search Train by Number.

3.3 Data Structure Schema

Train data is stored as a list of dictionaries, where a typical train object looks like this:

```
trains = [
   {"train_no": "101", "train_name": "Express A", "source": "Delhi", "destination": "Mumbai", "seats": 55},
   {"train_no": "102", "train_name": "Superfast B", "source": "Kolkata", "destination": "Chennai", "seats": 34},
   {"train_no": "103", "train_name": "Mail C", "source": "Jaipur", "destination": "Pune", "seats": 43},
   {"train_no": "104", "train_name": "Shatabdi Express", "source": "NDLS", "destination": "BPL", "seats": 50, },
   {"train_no": "105", "train_name": "Rajdhani Express", "source": "BPL", "destination": "NDLS", "seats": 50,},
   {"train_no": "106", "train_name": "Intercity Express", "source": "JHS", "destination": "LKO", "seats": 100, },


]


]
```

4. Implementation Details

4.1 Core Logic and Python Concepts Used

1. **Functions:** The entire project is organized using functions (view_train(), search_train(), book_seat()) to ensure code reusability and maintainability.

2. **Conditional Logic and Loops:** if/else statements are extensively used for menu navigation, input validation, and decision-making (e.g., checking if enough seats are available before booking). for loops are used to iterate over the train_data list.
3. **Dictionary Manipulation:** Core operations involve accessing, updating, and adding key-value pairs within the train dictionaries (e.g., decreasing the available_seats value upon a successful booking).

4.2 Code Snippet Example (Booking)

The booking logic is a critical part of the project, demonstrating data integrity checks:

```
def book_ticket(train_number, seats_to_book):
    """Attempts to book tickets and updates the train's capacity."""
    def book_seat():
    print("\n--- Book Seat ---")
    num = input("Enter Train Number: ")


    for t in trains:
        if t["train_no"] == num:
            if t["seats"] > 0:
                t["seats"] -= 1
                print("Seat booked successfully!")
                print(PNR.generate_pnr())
                print(f"Remaining Seats: {t['seats']}\n")
            else:
                print("No seats available.\n")
            return


    print("Train not found.\n")
```

5. Results and Testing

The STMS provides a fully interactive and robust CLI experience for basic management tasks.

6. Conclusion and Future Scope

6.1 Conclusion

The Simple Train Management System project successfully met all its defined objectives. It served as an excellent hands-on exercise for solidifying fundamental Python concepts, including data structure

handling, function definition, and control flow. The project proves the ability to translate real-world requirements into a functional, albeit simple, software application.

6.2 Future Enhancements

The system is limited by its console interface and lack of persistent data storage (data is lost upon program exit). Potential improvements include:

1. **Persistent Storage:** Integrate a database (e.g., SQLite) or use JSON/CSV file I/O to save data between sessions.
2. **Graphical User Interface (GUI):** Migrate from CLI to a desktop application using libraries like Tkinter or PyQt for improved user experience.
3. **Advanced Features:** Implement cancellation logic, dynamic pricing, and user authentication/login.

End of Report