NAME-: RUDAR PARTAP SINGH

SAP ID-:590026152

COURSE-: BTECH CSE

Simple Console-Based Train Ticket Booking System

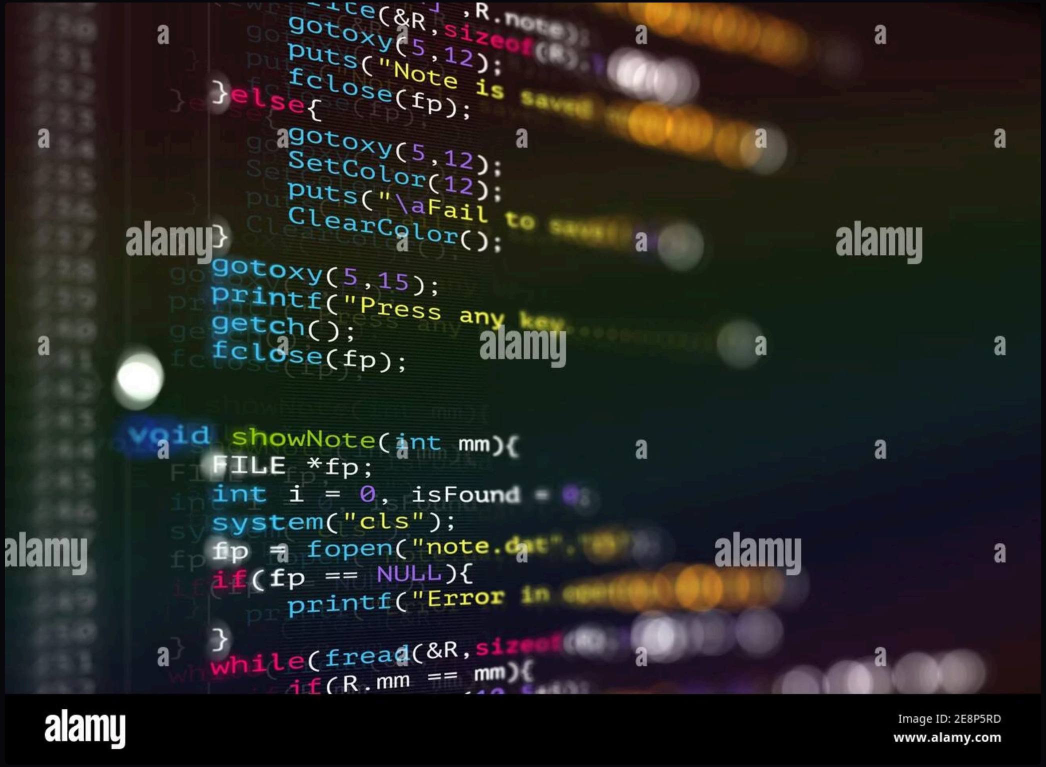
A C Programming Project demonstrating fundamental concepts through a practical application. This system simulates the core functionality of train ticket reservation using console-based interaction, modular function design, and structured data handling.

Project Overview

The Challenge

Create a functional command-line utility that simulates essential train ticket booking operations. This project bridges the gap between theoretical C programming knowledge and real-world application development.

The system demonstrates how basic C constructs—functions, arrays, pointers, and structured input/output—can be combined to create a practical, user-friendly booking interface.



```
    ite(&R,sizeof(R.note)
    gotoxy(5,12);
    puts("Note is saved
    fclose(fp);
}else{
    gotoxy(5,12);
    SetColor(12);
    puts("\aFail to save
    ClearColor();
}
gotoxy(5,15);
printf("Press any key
getch();
fclose(fp);

void showNote(int mm){
FILE *fp;
int i = 0, isFound = 0
system("cls");
fp = fopen("note.dat","
if(fp == NULL){
    printf("Error in
}
while(fread(&R,sizeof(
    if(R.mm == mm){
```

System Capabilities

Passenger Details

Collects essential information including name, age, and journey endpoints

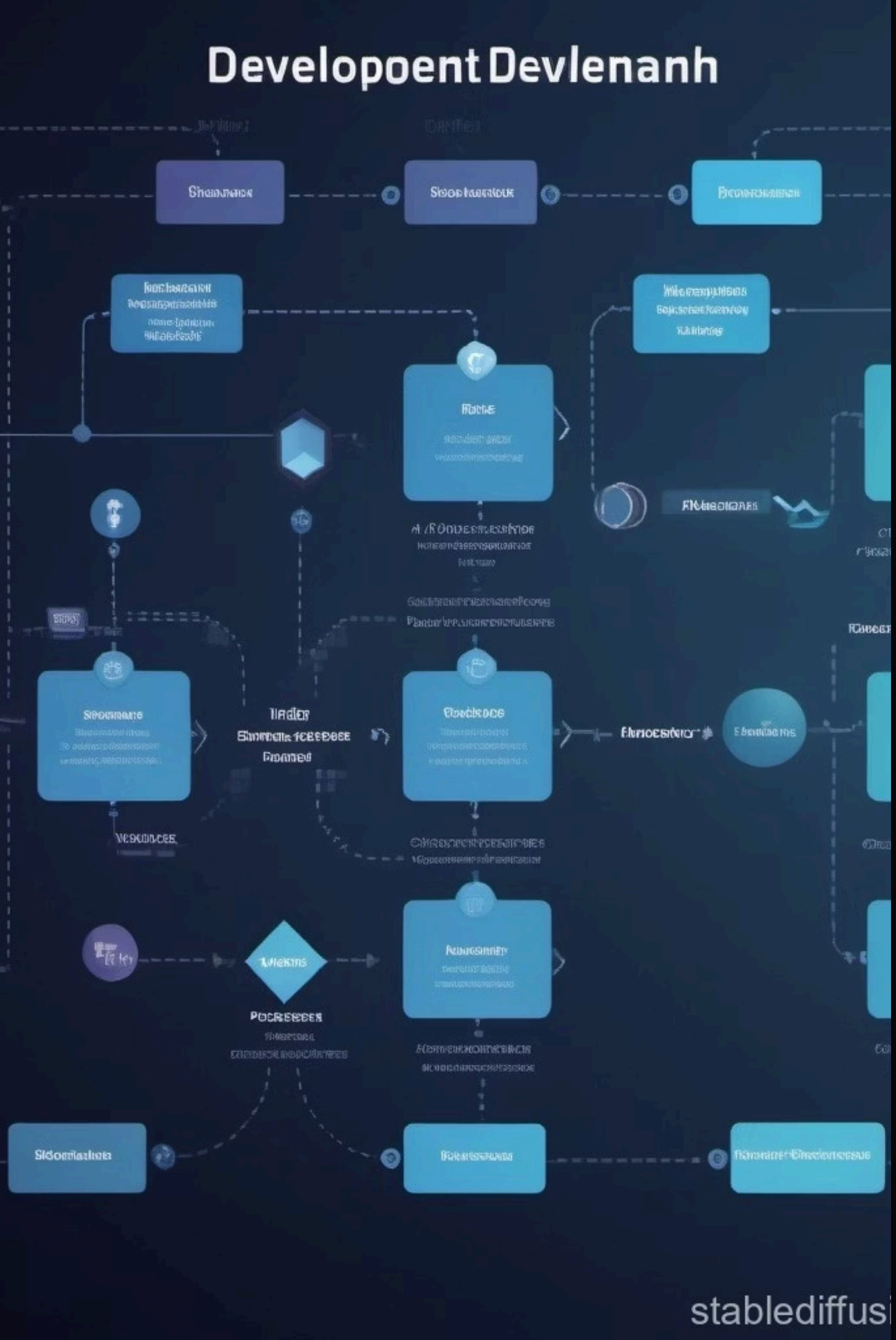
Train Display

Shows available trains dynamically based on selected route

Booking Summary

Generates comprehensive confirmation with all travel details

The system accepts six key inputs: passenger name, age, source station, destination station, train number, and ticket quantity. It processes this information through modular functions to produce a professional booking confirmation.



Program Flow Architecture

Input Collection

getPassengerDetails() gathers user information with safe string handling

Train Display

showTrains() presents available options for the selected route

Booking Process

processBooking() generates the final confirmation summary

Safe String Input Handling

The Challenge

String input in C requires careful handling to prevent buffer overflow and newline character issues. Using scanf() for strings can lead to unpredictable behavior with spaces.

The Solution

The system implements fgets() combined with a custom removeNewline() utility function. This ensures safe input capture while eliminating trailing newline characters that could corrupt data display.

```
void removeNewline(char *str) {
 str[strcspn(str, "\n")] = 0;
}

void getPassengerDetails(char name[],
 int *age, char source[],
 char destination[]) {
 printf("\nEnter passenger name: ");
 fgets(name, 50, stdin);
 removeNewline(name);

 printf("Enter age: ");
 scanf("%d", age);
 getchar(); // Clear buffer
}
```

Modular Function Architecture

getPassengerDetails()

Handles all user input collection with pointer parameters for age and character arrays for string data. Demonstrates pass-by-reference concepts.

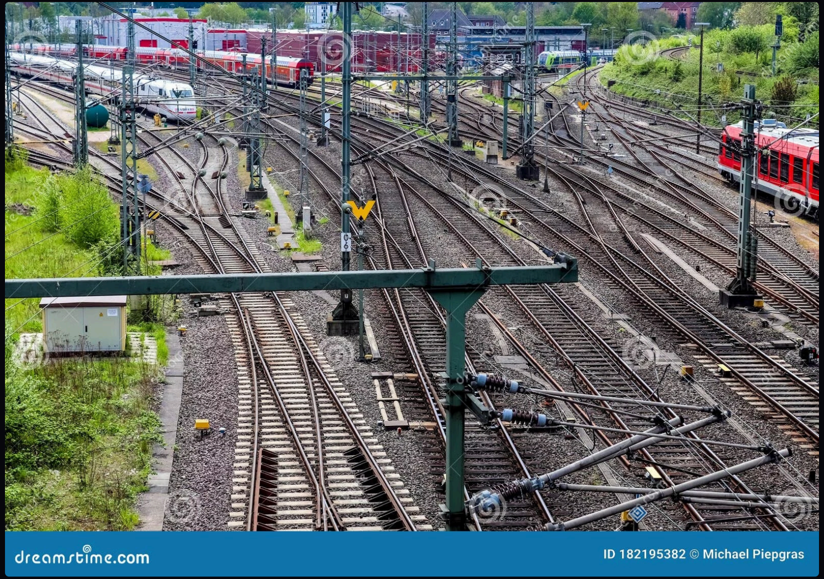
showTrains()

Displays hardcoded train options dynamically using user's source and destination inputs. Shows practical string parameter usage.

processBooking()

Consolidates all collected data into a formatted booking summary. Demonstrates structured output generation and data presentation.

Dynamic Train Display Implementation



Context-Aware Output

The showTrains() function demonstrates advanced string formatting by incorporating user input directly into the display output. Rather than showing generic train listings, it presents options specifically for the user's selected route.

```
void showTrains(char source[],
 char destination[]) {
 printf("\nAvailable Trains:\n");
 printf("1. 12345 - Express A ");
 printf("(%s    %s)\n",
 source, destination);
 printf("2. 67890 - Express B ");
 printf("(%s    %s)\n",
 source, destination);
}
```

01
AI-ML Enabled QA and Testing Services
Shift-Left Testing
Test Automation at Scale
04
TestOps & Quality Engineering
Cybersecurity Testing
Test Automation for Mobile and IoT
Software Testing & Quality Assurance Trends

Testing Results

1

Test Scenario

Passenger: John Doe (Age 35)
Route: Delhi → Mumbai
Train: 67890
Tickets: 2

2

Expected Output

System should display trains for Delhi-Mumbai route and generate a complete booking summary with all entered details formatted correctly.

3

Actual Results

Program executed successfully! All inputs were captured accurately, train display showed correct route, and booking summary matched expectations perfectly.

Key Learning Outcomes

Function Design

Understanding modular programming through function creation, parameter passing, and return value handling. Learning to break complex problems into manageable, reusable components.

Safe Input Handling

Implementing robust string input methods using `fgets()` and buffer management. Understanding the importance of input validation in preventing program crashes.

Pointers & Arrays

Practical application of pointer concepts for pass-by-reference parameters and character array manipulation for string operations.

Structured Output

Creating professional-looking formatted output using printf statements with proper alignment and presentation techniques.

Future Enhancement Roadmap

Data Persistence

Implement file handling using .txt or .csv files to permanently store train information and booking records, enabling data retrieval across program sessions.

Seat Availability

Add logic to track and update available seats per train, implementing real-time availability checks and preventing overbooking scenarios.

Payment Processing

Integrate basic payment calculation and processing functionality, including fare computation based on distance and ticket class.

GUI Development

Migrate from console to graphical interface using libraries like GTK or Qt, providing enhanced user experience with visual elements and mouse interaction.

Project Success: This system successfully demonstrates core C programming concepts through a practical, real-world application. It provides a solid foundation for understanding function-based architecture and user interaction in console applications.