C Programming Major Project (CSEG1032)

Airline Ticket Management System (ATMS)

This project implements a console-based Airline Ticket Management System (ATMS) in C. The primary goal is to demonstrate mastery of modular design, data structures, and robust file I/O for achieving data persistence.

The system adheres strictly to the mandatory requirements regarding repository structure and modular separation, ensuring high scores in both implementation quality and grading compliance.

Core Features

1. **Flight Management:** Allows creation, storage, and display of flights (ID, route, capacity, price). Includes validation to ensure capacity and price are positive values.
2. **Data Persistence:** Uses **binary file I/O (fread/fwrite)** to load and save all flight records, ticket records, and a unique, sequential PNR counter (pnr_counter.dat) across program sessions.
3. **Booking and PNR Generation:** Enables ticket booking, generating a **unique, persistent PNR** for each transaction.
4. **Cancellation:** Supports ticket cancellation using the PNR, gracefully updating the corresponding flight's availableSeats count and removing the record from the array.
5. **Search & Display:** Allows users to view all flights and search by route criteria.

Mandatory Repository Structure

The repository structure **must** follow the template below precisely. Failure to maintain this structure will result in a **fatal error** during the automated evaluation (Section 8.1).

ATMS_Project/

|-- src/

| |-- main.c (Main program loop, UI execution)

| |-- data_handler.c (All Persistence and File I/O logic)

| |-- ticket_operations.c (All Core business logic: CRUD functions)

|-- include/

| |-- airline_management.h (Structs, Constants, and Function Prototypes)

|-- docs/

| |-- ProjectReport.pdf <-- MANDATORY: Full documentation (PDF)

| |-- assets/ <-- Flowcharts, diagrams, etc.

|-- README.md <-- This file

|-- sample_input.txt (Automated test cases for the grader)

|-- github_link.txt <-- MANDATORY: Submission file

Compilation and Execution

To successfully compile and run the project, ensure you are in the root directory (ATMS_Project/).

1. **Compilation (using GCC):**

This command compiles all .c files in src/ and directs the compiler to look for .h files in include/.

gcc -o main src/*.c -I include -std=c99

2. **Execution (Automated Grader Mode - Section 8.6):**

Use this command to simulate the environment used by the automated scoring script.

./main < sample_input.txt

3. **Execution (Interactive Mode):**

4. ./main

This will launch the main menu, allowing manual input and feature testing.