Introduction to Computing – Lab

Faculty of Information Technology & Computer Science

Campus Bus Route & Seat Allocation System

Course Instructor: Fraz Aslam

Project Overview

The Campus Bus Route & Seat Allocation System is a console-based application that simulates how a university manages its campus transport service. The system maintains a fixed set of bus routes, buses assigned to those routes, and seat allocation for students using the service. It stores route details, bus capacity, current seat occupancy, and student allocation records. The program supports registering students to routes, assigning seats, updating allocations, canceling allocations, searching records, detecting conflicts (overbooking, duplicate student allocations, invalid route/bus references), and generating summary reports for transport administration. The project is fully menu-driven and ensures students demonstrate solid command of static arrays, char arrays, loops, nested loops, and if/else logic only.

Features to Implement (Requirements / Deliverables)

1) Route & Bus Setup

- The system must manage a fixed number of routes (example: 5 to 10 routes).
- Each route must include at minimum:
    - **Route ID** (integer, unique)
    - **Route Name** (char array)
    - **Start Point** (char array)
    - **End Point** (char array)
    - **Total Stops Count** (integer)

- Each route must have one bus assigned in the system records, including:
    - **Bus ID** (integer, unique)
    - **Route ID** (integer reference)
    - **Bus Capacity (Seats)** (integer)
    - **Current Allocated Seats Count** (integer)
- The system must prevent duplicate Route IDs and duplicate Bus IDs.
- The system must ensure that each bus record references a valid Route ID.

2) Display Routes and Bus Status

- The system must display all routes along with their bus assignment and seat status, including:
    - Route ID, route name, start/end, stops count
    - Bus ID, capacity, allocated seats, remaining seats
- The display must handle the case where routes/buses are not yet initialized.

3) Student Registration for Transport (Seat Allocation)

- The system must allow allocating a seat to a student for a specific route.
- Each student allocation record must include at minimum:
    - **Allocation ID** (integer, unique)
    - **Student ID** (integer, unique within active allocations)
    - **Student Name** (char array)
    - **Department** (char array)
    - **Semester** (integer)
    - **Contact Number** (char array)
    - **Route ID** (integer)
    - **Bus ID** (integer)
    - **Seat Number** (integer)
    - **Fee Status** (integer-coded, e.g., 0=Unpaid, 1=Paid)
- The system must enforce rules such as:
    - Route must exist
    - Bus must exist and must match the chosen route
    - Seat allocation must not exceed bus capacity
    - Student must not already have an active seat allocation
    - Seat number must be valid and must not already be assigned to another student on the same bus
- The system must clearly indicate success or failure and must not create partial/invalid records.

4) Update Student Allocation Record

- The system must allow updating a student's allocation using **Student ID** or **Allocation ID** as the key (system policy must be consistent).
- Update must support modifying at least:
    - Student name, department, semester, contact number

    - Fee status
    - Route/bus transfer (only if capacity and seat availability rules remain satisfied)
    - Seat number change (must remain within capacity and not collide with another student's seat)
- The system must clearly report if the record does not exist or the update violates rules.

5) Cancel Seat Allocation

- The system must allow cancellation/removal of an allocation record using the selected key.
- After cancellation:
    - The seat must become available again
    - The bus allocated seat count must update correctly
    - The allocation record must no longer appear in listings/search results
- The system must handle cancellation of non-existing records safely.

6) Search and Listing Operations

The system must provide searching/listing options including at least:

- Search by **Student ID** (show full allocation details)
- List all allocated students for a **Route ID**
- List all allocated students for a **Bus ID**
- List all students by **Fee Status** (paid/unpaid)
- List all students of a specific **Department**
  All outputs must be readable and must handle "not found" cases correctly.

7) Seat Map / Occupancy Display

- The system must display an occupancy view for a selected bus showing:
    - Total capacity
    - Which seat numbers are occupied and which are free (based on stored allocations)
- The output must be generated from stored allocation data and must reflect the current system state correctly.

8) Validation / Conflict Detection

The system must provide checks that can be run from the menu, including at least:

- Detect duplicate Student IDs across active allocations
- Detect duplicate seat numbers assigned on the same bus
- Detect allocations that reference invalid Route ID or invalid Bus ID
- Detect allocations where seat number is out of bus capacity range
- Detect bus records whose allocated seat count does not match actual allocation records
  A readable validation report must be printed listing issues found.

9) Reports / Summaries

The system must generate summary reports such as:

- Total routes, total buses, total active allocations
- Route-wise allocated seats and remaining seats
- Paid vs unpaid allocation counts
- Department-wise transport usage counts
- Bus(es) with highest occupancy and lowest occupancy
  Reports must be computed using loops and displayed clearly.

10) Exit

- Provide an option to exit the program gracefully.

Constraints (Must Follow)

- **Allowed library:** iostream only
- **Control structures:** if / else, loops (for, while, do-while), including nested loops
- **Data structures allowed:**
  - Static **1D int arrays** only (no dynamic arrays)
  - Static **1D char arrays** only (no string type)
- **Not allowed:**
  - No string library or std::string
  - No built-in string functions (e.g., anything from cstring)
  - No user-defined functions
  - No dynamic memory (new, malloc, vectors, etc.)
- Program must be fully menu-driven and demonstrate repeated operations until exit.

Sample Output Menu (Console)

```
==================================================
   Bus Route & Seat Allocation System
==================================================

1) Initialize / Reset Routes & Buses
2) Add / Replace Route Records
3) Add / Replace Bus Records
```

```
4) Display Routes & Bus Seat Status
5) Allocate Seat to Student (Transport Registration)
6) Update Student Allocation Record
7) Cancel Seat Allocation
8) Search / Listings
9) Display Bus Seat Occupancy View
10) Validate Records & Detect Conflicts
11) Generate Reports / Summaries
0) Exit
-------------------------------------------
Enter your choice:
```