CS221 – Data Structures & Algorithms

Semester Project (Fall 2025)

Project Title: Intelligent Transport Network Management System (ITNMS)

1. Introduction

The purpose of this semester project is to provide students with hands-on experience in applying core Data Structures and Algorithms (DSA) to design and implement a complex, real-world software system. The **Intelligent Transport Network Management System (ITNMS)** simulates a smart city transportation network, allowing users to manage vehicles, passengers, routes, stations, ticketing, shortest paths, and traffic operations.

This project integrates **all major concepts of CS221**, including arrays, linked lists, queues, stacks, trees, graphs, hashing, and a complete set of searching and sorting algorithms.

2. Project Description

Students will build an interactive system that models the functioning of a city-wide transport network similar to metro/bus routing systems or GPS navigation applications.

The system will:

- Maintain station and vehicle information
- Manage routes and connections
- Allow shortest path computations
- Handle queues for passengers
- Store system history using stacks
- Use hashing for fast identification
- Provide sorting & searching modules
- Generate analytical reports

This project will be completed **individually** or in **teams of maximum 4 students**.

3. Core Functional Requirements

3.1 Data Structure Requirements (Mandatory)

Concept	Required Implementation
Arrays	Store static data (e.g., station list, fare list)
Linked Lists	Used in hash table chaining, dynamic queues
Stacks	Undo/redo, navigation history
Queues	Ticketing system, passenger waiting line
Trees (BST/AVL)	Store route or station metadata
Heaps (Priority Queue)	Fastest vehicle, traffic prioritization
Graphs	Main transport network (adjacency list/matrix)
Hash Tables	Vehicle/passenger lookup
Searching Algorithms	Linear, Binary
Sorting Algorithms	Bubble, Selection, Insertion, Quick, Merge, Heap
Algorithm Complexity	Time & space analysis required

4. System Modules

4.1 Route & Station Management (Graphs)

- Add / Delete Station
- Add / Delete Route (edge)
- Display all connected stations
- Perform BFS & DFS
- Find **Shortest Path** using Dijkstra
- Generate **Minimum Spanning Tree** (MST)
- Detect cycles in the network

4.2 Passenger Ticketing System (Queues)

Implement a **FIFO queue** for ticket requests:

- Passenger enters queue
- Display queue
- Process next passenger
- Circular Queue or Linked-List Queue allowed

4.3 Vehicle Database (Hashing + Linked Lists)

Use a hash table to store and retrieve vehicles:

- Insert vehicle
- Search vehicle
- Remove vehicle
- Handle collisions (preferably chaining)

4.4 History & Undo Operations (Stacks)

Use a **stack** to store operations such as:

- Last visited station
- Actions history
- Undo last action

4.5 Searching & Sorting Module

User selects the algorithm to apply on a dataset.

Searching Algorithms

- Linear Search
- Binary Search

Sorting Algorithms

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort
- Counting/Radix Sort (optional)

Each must include:

- Best/Worst/Average time complexity
- Space complexity
- Sample dataset execution

4.6 Analytics & Reporting (Advanced DSA)

Optional but recommended:

- Most crowded station (hash frequency count)
- Busiest route (graph edge weight statistics)
- Fastest vehicle assignment (min-heap)
- Traffic density prediction (heap sorting)
- Daily usage trends (BST traversal)

5. Technical Requirements

- Programming language: **C++**
- Use object-oriented design
- Use separate classes for each module
- Code must be modular, readable, and documented

- Provide complexity analysis for each algorithm used
- Use meaningful variable/class names

6. Deliverables

Students must submit:

6.1 Source Code Folder

- Properly commented
- Separate class files & modules
- README with instructions

6.2 Project Report (30–40 pages)

The report must include:

Cover Page

- Title
- Course name
- Student name & ID

Chapters

i. **Introduction**

ii. **System Requirement Specifications (SRS)**

iii. **System Design**
   - Flowcharts
   - UML diagrams
   - Data structure diagrams

iv. **Implementation**
   - Code snippets
   - Explanation of chosen data structures

v. **Algorithm Analysis**
   - Complexity of all algorithms

vi. **Testing & Results**

vii. **Conclusion & Limitations**

viii. **References**

6.3 Demo Video (5–10 mins)

Explaining:

- System navigation
- Key DSA implementations
- Shortest path demonstration
- Queue/Stack/Hash operations

7. Constraints & Rules

- No use of external libraries for graph algorithms (except I/O).
- All data structures must be implemented **manually**.
- Searching and sorting must be coded by students (not built-in).
- Collaboration allowed only within group.
- Plagiarism will result in **zero marks**.

8. Grading Breakdown (100 Marks)

Component	**Marks**
Graph Implementation & Algorithms	20
Queues & Stacks Functionality	10
Hash Table Design	10
Searching & Sorting Library	15

Component	Marks
Tree/Heap Implementations	10
Code Quality & Modularity	10
Report Quality	15
Demo Presentation/Viva	10

10. Conclusion

This project is designed to push students beyond simple coding tasks and help them understand the real-world application of Data Structures and Algorithms. By implementing every major DSA concept inside a coherent, challenging system, students will gain practical skills similar to those used in navigation systems, traffic control, and intelligent transport planning.