Semester Project: Transport Network System

Purpose

This project is designed to simulate the development of a real-world software system in a collaborative environment. It challenges your understanding of core data structures (graphs, heaps, hash maps, queues, stacks, trees) and emphasizes code modularity, interface design, and progressive system building. By working in teams, you'll practice dividing responsibilities, integrating components, and maintaining clean, reusable code across stages.

Assignment

You and your team have been hired as software engineers at TransRoute Systems, a logistics company working to modernize the country's transportation infrastructure. Your job is to build the core routing engine that powers everything from emergency supply delivery to long-haul freight scheduling.

The Ministry of Transport has provided you with:

- A large network of cities and roads
- Real-time traffic updates
- A stream of delivery requests that must be managed and audited

Your system must:

- Parse and model this map efficiently
- Handle live traffic conditions and route recalculations
- Track and replay delivery events for audit and optimization

You won’t build the UI or GPS; your job is deeper: you are building the graph layer, the algorithms that plan the best paths, the data structures that hold the system together. It must be fast, modular, and testable via command-line input/output.

By the end of this project, your engine will be able to:

1. Model a full country-wide road network
2. Respond to dynamic traffic conditions in real-time
3. Schedule, track, and audit deliveries over time

Welcome to the team. Your data structures are about to carry the weight of a nation.

This is a team project structured in 3 progressive modules. Each group (3 to 4 students) will collaboratively build and evolve a transport network system using core data structures. You must design clear module boundaries, practice code reuse, and submit code that works via command-line input/output. Your code will be tested with input files, your scripts must read these files and print correct results.

Module 1: Graph Builder

You will build a weighted, directed graph based on a text file describing cities and roads.

Team Tasks:

- Design a Graph class with add_node, add_edge, remove_node, remove_edge
- Implement to_adjacency_list() for output
- Parse structured input files and handle errors cleanly
- Define clear interfaces for future modules to use

Below is an example of how the input .txt (100 cities, 600 roads) file will look like:

```
CITIES
City1
City2
City3
City4
ROADS
City1 City2 5
City2 City3 3
City3 City4 4
City1 City4 10
```

Below is an example of how the output will look like. You may remove City or keep it.

```
City1: City2(5), City4(10)
City2: City3(3)
City3: City4(4)
```

And you will run your script as: `python3 graph_builder#.py input1.txt`

The # is referencing your group number.

Module 2: Traffic & Priority Query System

You will simulate live traffic conditions and support shortest-path queries.

Team Tasks:

- Extend graph to support dynamic edge weights via a traffic map (hash map)
- Use Dijkstra's with a heap for shortest path
- Implement K_PATHS using priority queues or iterative Dijkstra
- Provide formatted CLI output (e.g., -> arrows and cost summaries)

Below is an example of how the input .txt file (100 traffic reports, 50 queries) will look like:

```
TRAFFIC_REPORT City1 City2 +3
TRAFFIC_REPORT City3 City4 -2
QUERY SHORTEST_PATH City1 City4
QUERY K_PATHS City1 City4 2
```

Below is an example of how the output will look like:

```
SHORTEST_PATH City1 City4: City1 -> City2 -> City3 -> City4 (cost: 12)
K_PATHS City1 City4:
1) City1 -> City2 -> City3 -> City4 (12)
2) City1 -> City4 (13)
```

And you will run your script as:

```
python3 graph_query#.py input1.txt commands2.txt
```

The # is referencing your group number.

Module 3: Delivery Scheduler with History Tracking

You will manage a delivery task queue and track route history for auditing.

Team Tasks:

- Use a queue for delivery scheduling
- Log completed routes into a searchable structure (BST, sorted list, etc.)
- Implement an undo stack to roll back actions
- Enable time-range queries over the history

Below is an example of how the input .txt file (50 delivery tasks with timestamps, history, and undo commands) will look like:

```
SCHEDULE DELIVERY City1->City4 at 9:00
SCHEDULE DELIVERY City2->City3 at 9:15
RECORD_HISTORY
UNDO_LAST
QUERY_HISTORY BETWEEN 9:00 9:30
```

Below is an example of how the output will look like:

```
Scheduled: City1->City4 at 9:00
Scheduled: City2->City3 at 9:15
Recorded history
Undid last action
History between 9:00 and 9:30:
- City1->City4 at 9:00
```

And you will run your script as:

```
Python3 graph_schedule#.py schedule3.txt
```

The # is referencing your group number.

Team Presentation

Each team will create a short PowerPoint and present on the work you did. Elaborate on the procedure, structures, and setup you did to achieve this project.. Please keep the presentation no more than 10 minutes. Below are the required slides:

1. Team Introduction: Names, roles, and responsibilities

2. System Architecture: Diagram of how modules/components interact
3. Data Structures Used: Graphs, heaps, queues, trees, etc. why each was chosen
4. Technical Highlights: Challenges and interesting solutions per module
5. Sample Inputs/Outputs: Screenshots or CLI demos
6. Lessons Learned: Workflow insights, debugging stories, teamwork takeaways
7. (Optional) Any custom features or extensions added

This is where you will submit your README file with member names and assigned tasks, contribution breakdown, summary of tbe designs. I will be grading this on professionalism, ease of reading, and polish.

Deliverables

- 3 python scripts: graph_builder#.py, graph_query#.py, graph_schedule#.py
- Team README.
- Presentation
- Only one member from the team needs to submit it. Please do not submit multiples.

Module 1

100 points maximum

	40	25	20	5	0
Functionality	Graph parsing, structure, and output are complete, correct, modular, and efficient	Most features correct; code is modular, with minor errors	Functional but lacks reusability or has edge case failures	Basic implementation with clear flaws	Major parts missing or incorrectly structured
Modularity & Code Reuse	Code is clean, DRY, and reused across modules without duplication	Mostly reused; some duplication or weak abstraction	Partial reuse with copy/paste logic	Modules feel disconnected or inconsistent	Each module rebuilt from scratch, no shared code
Group Contract			Submitted		Not submitted

-10 point: multiple submissions

Module 2

100 points maximum

	50	25	10	5	0
Functionality	Fully supports shortest and k-path queries with clear, correct, formatted outputs	Algorithms work but k-paths may be incorrect or inefficient	Output correct but formatting unclear or no CLI support	One query correct; second partially or wrongly implemented	Neither query is correctly implemented
Code Reuse and Modularity	Code is clean, DRY, and reused across modules without duplication	Mostly reused; some duplication or weak abstraction	Partial reuse with copy/paste logic	Modules feel disconnected or inconsistent	Each module rebuilt from scratch, no shared code

- 10 points: multiple submissions

Final Presentation (Module 1, 2, 3)

100 points maximum

	40	20	15	10	5	0
Module 3		Delivery queue, history log, undo, and time-range queries are fully functional and cleanly designed	3 of 4 major features implemented correctly	2 features work, others stubbed or incorrectly implemented	At least 1 working, but implementation is hard to follow	Bare minimum scheduling with broken or missing logic
Code Reuse and Modularity		Code is clean, DRY, and reused across modules without duplication	Mostly reused; some duplication or weak abstraction	Partial reuse with copy/paste logic	Modules feel disconnected or inconsistent	Each module rebuilt from scratch, no shared code
Presentation Itself (see rubric below)	See rubric below					
Documentation		README with roles, structure diagrams, sample outputs, lessons learned	Most content included; a bit light on technical clarity	Some missing or vague content; limited explanation	incomplete or poorly structured	no one could explain key components

-10 point: missing readme

Final Presentation

40 points maximum

Nonverbal Skills	**4 – Exceptional**	**3 – Admirable**	**2 – Acceptable**	**1 – Poor**
Eye Contact	Holds attention of entire audience with the use of direct eye contact, seldom looks at notes or slides.	Consistent use of direct eye contact with audience, but still returns to notes.	Displayed minimal eye contact with audience, while reading mostly from notes.	No eye contact with audience, as entire report is read from notes.
Body Language	Movements seem fluid and help the audience visualize.	Made movements or gestures that enhance the articulation.	Very little movement or descriptive gestures.	No movement or descriptive gestures.
Poise	Displays relaxed, self-confident nature about self, with no-mistakes	Makes minor mistakes, but quickly recovers from them; displays little or no tension.	Displays mild tension; has trouble recovering from mistakes.	Tension and nervous is obvious; has trouble recovering from mistakes.

Verbal Skills	**4 – Exceptional**	**3 – Admirable**	**2 – Acceptable**	**1 – Poor**
Enthusiasm	Demonstrates a strong, positive feeling about topic during entire presentation	Occasionally shows positive feelings about topic.	Shows some negativity toward topic presented.	Shows absolutely no interest in topic presented.
Speaking Skills	Uses a clear voice and speaks at a good pace so audience members can hear presentation. Does not read off slides.	Presenter's voice is clear. The pace is a little slow or fast at times. Most audience members can hear presentation.	Presenter's voice is low. The pace is much too rapid/slow. Audience members have difficulty hearing presentation.	Presenter mumbles, talks very fast, and speaks too quietly for majority of students to hear and understand.

Timing	**4 – Exceptional**	**3 – Admirable**	**2 – Acceptable**	**1 – Poor**
Length of Presentation	Within two minutes of allotted time +/-.	Within four minutes of allotted time +/-.	Within six minutes of allotted time +/-.	Too long or too short; ten or more minutes above or below allotted time

Content	**4 – Exceptional**	**3 – Admirable**	**2 – Acceptable**	**1 – Poor**
Subject Knowledge	An abundance of material clearly related to the research is presented.	Sufficient information within many good points made, uneven balance and little	There is a great deal of information that is not clearly integrated or	Goal of research unclear, information included that does not support research

	Points are clearly made and evidence is used to support claims.	consistency	connected to the research.	claim in any way.
Organization	Information is presented in a logical and interesting sequence which audience can follow. Flows well.	Information is presented in logical sequence which audience can follow.	Audience has difficulty following presentation because the presentation jumps around and lacks clear transition.	Audience cannot understand presentation because there is no sequence of information.
Visuals	Excellent visuals that are tied into the overall story of the research	Appropriate visuals are used and explained by the speaker.	Visuals used but not explains or put in context.	Little or no visuals, too much text on slides.
Mechanics	Presentation has no misspellings or grammatical errors.	Presentation has no more than two misspellings and/or grammatical errors.	Presentation has three misspellings and/or grammatical errors.	Presentation has many spellings and/or grammatical errors