Department: **Computer Science** Program: **BS(CS)**

**Data Structures and Algorithms**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Announced date:10/06/2025** | | | **Due Date: 30/06/2025** | | **Total Marks = 20** |
| **Complex Computing Problem (CCP)** | | | | | |
| **Mapped CLO** | **SDG** | | **Complex Problem Solving Mapped** | | |
| CLO3 | 4 | | WP1 (Depth of knowledge required)  WP2 (Range of conflicting requirements required)  WP3 (Depth of analysis required)  WP4 (Familiarity of Issues) | | |

**COMPLEX COMUTING PROBLEM(REPORT)**

# **TITLE: EFFICIENT TRAFFIC MANAGEMENT SYSTEM SIMULATION USING DATA STRUCTURES**

**INSTRUCTOR’S NAME: ISRAR ALI**

**SLOT: TUESDAY&THURSDAY(2:30-3:45)**

**SUBMITTED BY:**

**MAHNOOR(68689)  
 SANIA IRFAN(68664)  
 DIYA PAHOOJA(68699)**

### **1. INTRODUCTION**

Urban traffic congestion remains one of the critical challenges in modern cities, impacting commute times, fuel consumption, and emergency response efficiency. This project aims to simulate a dynamic traffic management system that models vehicle movement and traffic light control at multiple intersections. Leveraging core data structures and algorithms, the system enhances traffic flow, reduces congestion, and ensures the prioritized movement of emergency vehicles.

### **2. PROBLEM STATEMENT**

The increasing volume of urban traffic requires intelligent systems that adapt in real-time. The objective of this project is to develop a simulation tool that optimizes traffic signal control and vehicle flow at city intersections. The system utilizes data structures such as queues, priority queues, and graphs to simulate real-world traffic behavior and analyze performance under dynamic conditions.

### **3. OBJECTIVES**

* Simulate vehicle flow at city intersections using efficient data structures.
* Design a priority-based vehicle clearance mechanism for emergency response.
* Model road networks using graph structures.
* Implement adaptive traffic signal control based on congestion.
* Provide a visual interface using JavaFX to monitor and control the simulation in real time.

### **4. TOOLS AND TECHNOLOGIES**

|  |  |
| --- | --- |
| **Component** | **Description** |
| * Programming | * Java (OOP, Collections Framework) |
| * UI Framework | * JavaFX (for GUI development) |
| * IDE | * Eclipse IDE |
| * Data Structures | * Queue, PriorityQueue, HashMap, Graph |
| * Algorithms | * Dynamic light adjustment; Dijkstra (planned) |

### **5. METHODOLOGY**

#### **5.1 Road Network Design**

A directed graph is used to model intersections and connecting roads, with each edge representing a road segment and associated travel time.

#### **5.2 Vehicle Handling**

Vehicles are generated with attributes such as ID and emergency status. They are stored in normal or priority queues at each intersection depending on their urgency.

#### **5.3 Traffic Signal Control**

Traffic light timing is adjusted dynamically based on the length of the queues. The green and red light durations are calculated to maximize throughput and minimize waiting time.

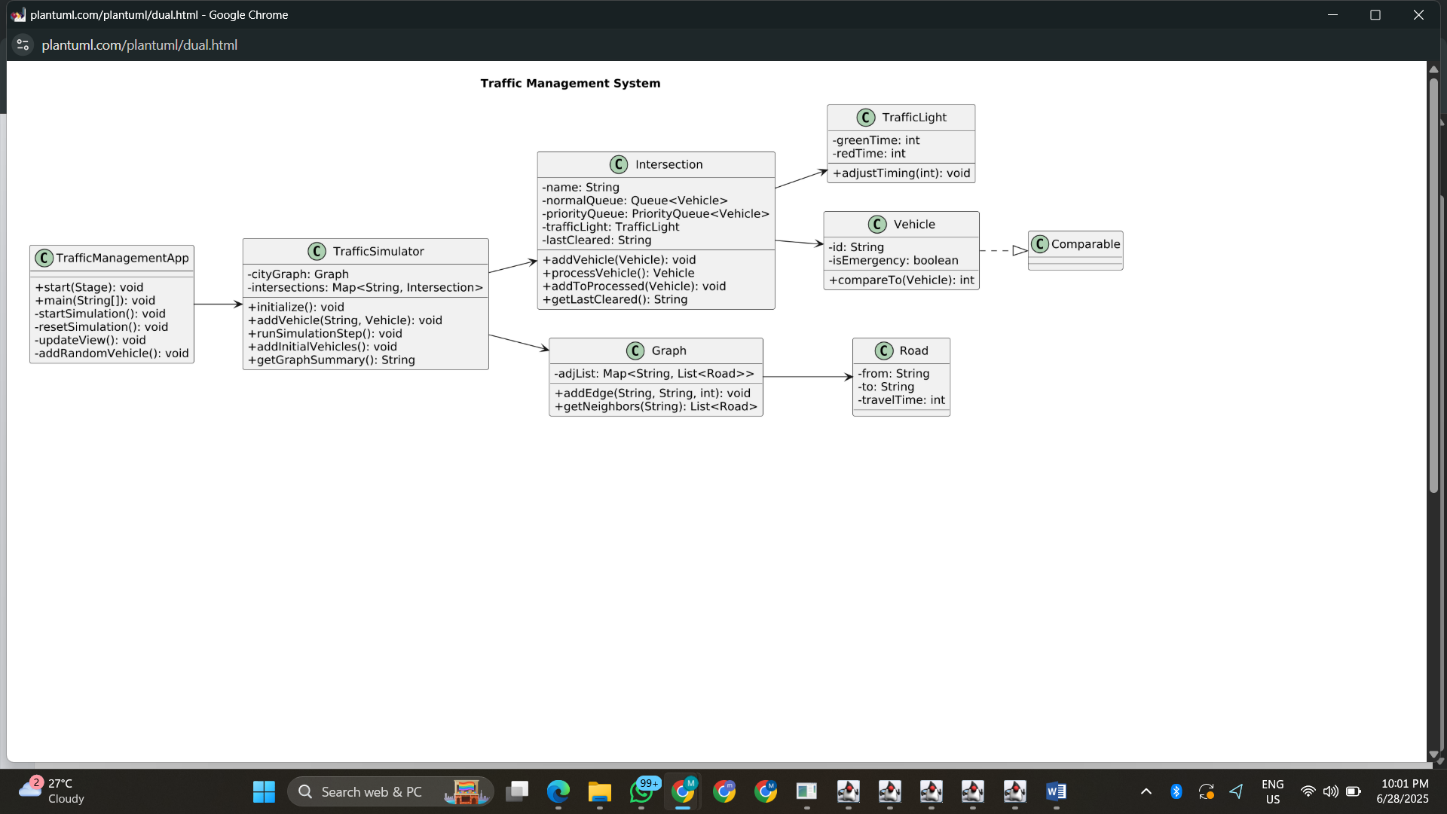
#### **5.4 Simulation Cycle**

The simulation runs in timed steps. At each step, intersections attempt to process vehicles based on queue status and traffic light timing. Emergency vehicles are prioritized automatically.

#### **5.5 Visualization**

A JavaFX-based interface visualizes each intersection in real-time, showing:

* Queue sizes (normal and emergency)
* Signal states (green/red durations)
* Recently cleared vehicle IDs
* Global statistics such as total and emergency vehicles cleared.

**UML DIAGRAM:**

### **6. KEY FEATURES**

* Priority-based vehicle dispatching.
* Real-time adjustment of signal timings based on congestion.
* Visual dashboard with per-intersection updates.
* Random vehicle arrival simulation.
* Emergency response handling using PriorityQueue.

### **7. DATA STRUCTURES OVERVIEW**

|  |  |
| --- | --- |
| Structure | **Application** |
| Queue | Manage regular vehicles at intersections |
| PriorityQueue | Prioritize emergency vehicles |
| Graph | Represent road network using adjacency lists |
| HashMap | Store and access intersections efficiently |

### **8. OUTPUT OVERVIEW**

The simulation provides a detailed console and GUI output:

* **Console:** Logs each simulation step with vehicle clearance, queue status, and traffic light timing.
* **GUI Dashboard:** Displays:
  + Intersections (A, B, C, D)
  + Vehicle queues
  + Signal timings (Green/Red)
  + Cleared vehicle ID
  + Summary statistics

### **9. CHALLENGES ENCOUNTERED**

* Maintaining synchronization between GUI rendering and backend simulation logic.
* Implementing dynamic signal timing logic without starving lower-priority vehicles.
* Designing an object-oriented structure to keep code modular and scalable.

### **10. FUTURE ENHANCEMENTS**

* Integrate Dijkstra’s algorithm for rerouting during high congestion.
* Add animations to visualize vehicle movement using image icons.
* Incorporate real-time data feeds and user-driven scenarios.
* Expand network to support 50+ intersections and 10,000+ vehicles.

### **11. CONCLUSION**

This project effectively demonstrates how core concepts from data structures and algorithms can be applied to solve a real-world problem in traffic management. The simulation offers a scalable, priority-aware, and dynamically adjustable system capable of adapting to traffic demands. The integration of JavaFX enhances user interaction and facilitates comprehensive visualization.

### **12. REFERENCES**

* Data Structures and Algorithms, Java Collections API
* JavaFX Documentation – Oracle
* Real-time Simulation Techniques in Java
* Dijkstra’s Algorithm: Shortest Path Analysis

## **USER GUIDE**

### **JavaFX Traffic Management Simulator - User Manual**

### **1. System Requirements**

* **Java SDK 8 or above**
* **JavaFX library (bundled with SDK or separately added)**
* **Eclipse IDE (recommended)**

### **2. Launching the Application**

1. Open Eclipse and import the project.
2. Run TrafficManagementApp.java.
3. The GUI will appear.

### **3. Main Interface Overview**

* **▶ Start Simulation:** Starts the simulation (15 time steps).
* **🔁 Reset Simulation:** Resets everything to the initial state.
* **🛣️ Road Connections:** Shows road network (A→B, A→D, B→C, D→C).
* **Simulation Cards:** Show real-time data of each intersection.
* **📊 Stats:** Show total and emergency vehicles cleared.

### **4. Simulation Behavior**

* Every second, the simulator:
  + Processes vehicles at intersections.
  + Updates queue sizes.
  + Adjusts traffic light timings.
  + Logs output to console and updates GUI.
  + Occasionally injects random vehicles (normal/emergency).

### **5. How Emergency Handling Works**

* Emergency vehicles are added to a PriorityQueue.
* These vehicles are always cleared before normal ones.
* This mimics real-world emergency passage rules.

### **6. Reset Function**

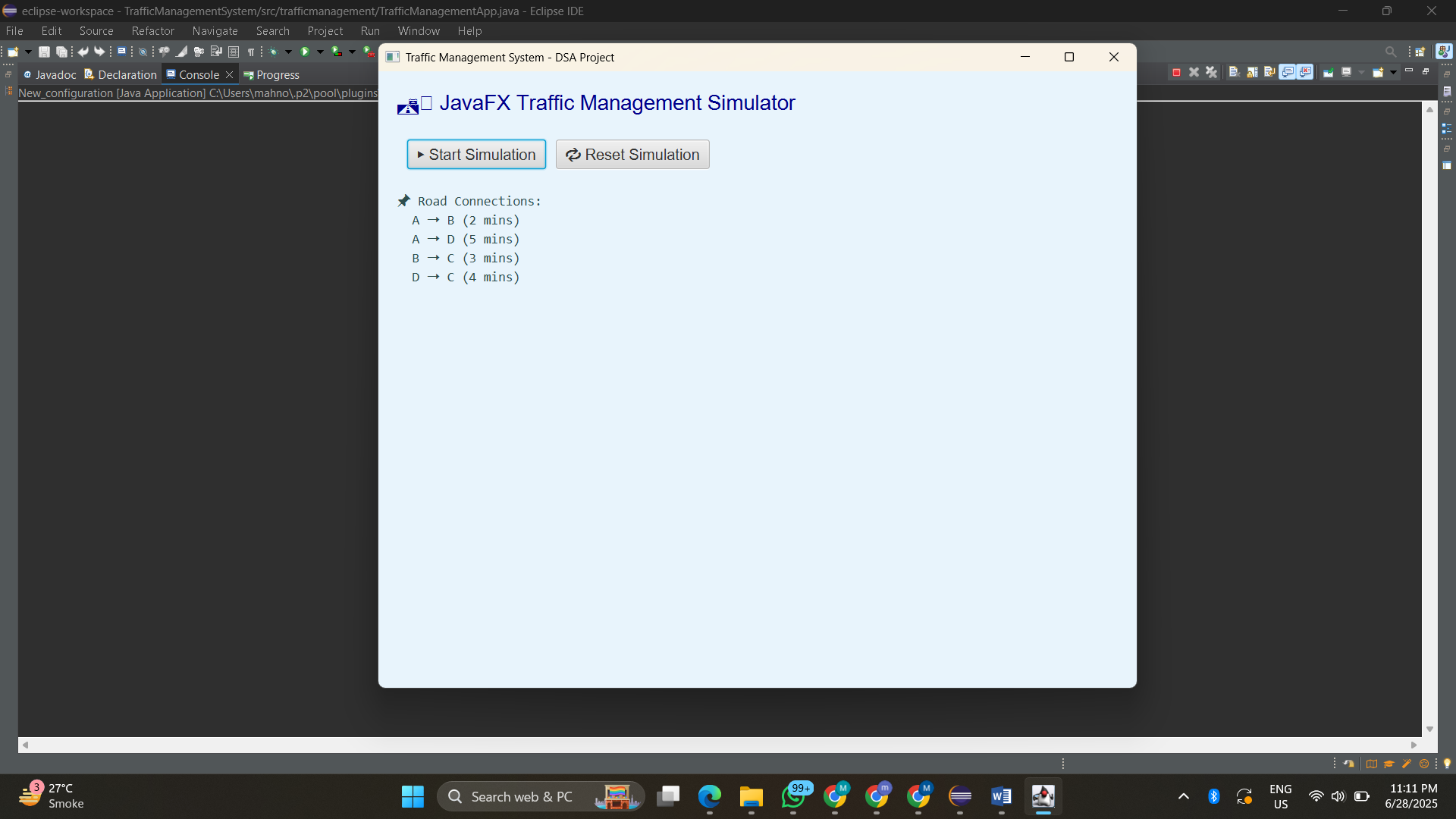
Clicking **Reset** will:

* Reinitialize roads and intersections.
* Reload initial vehicles.
* Clear the simulation view and stats.

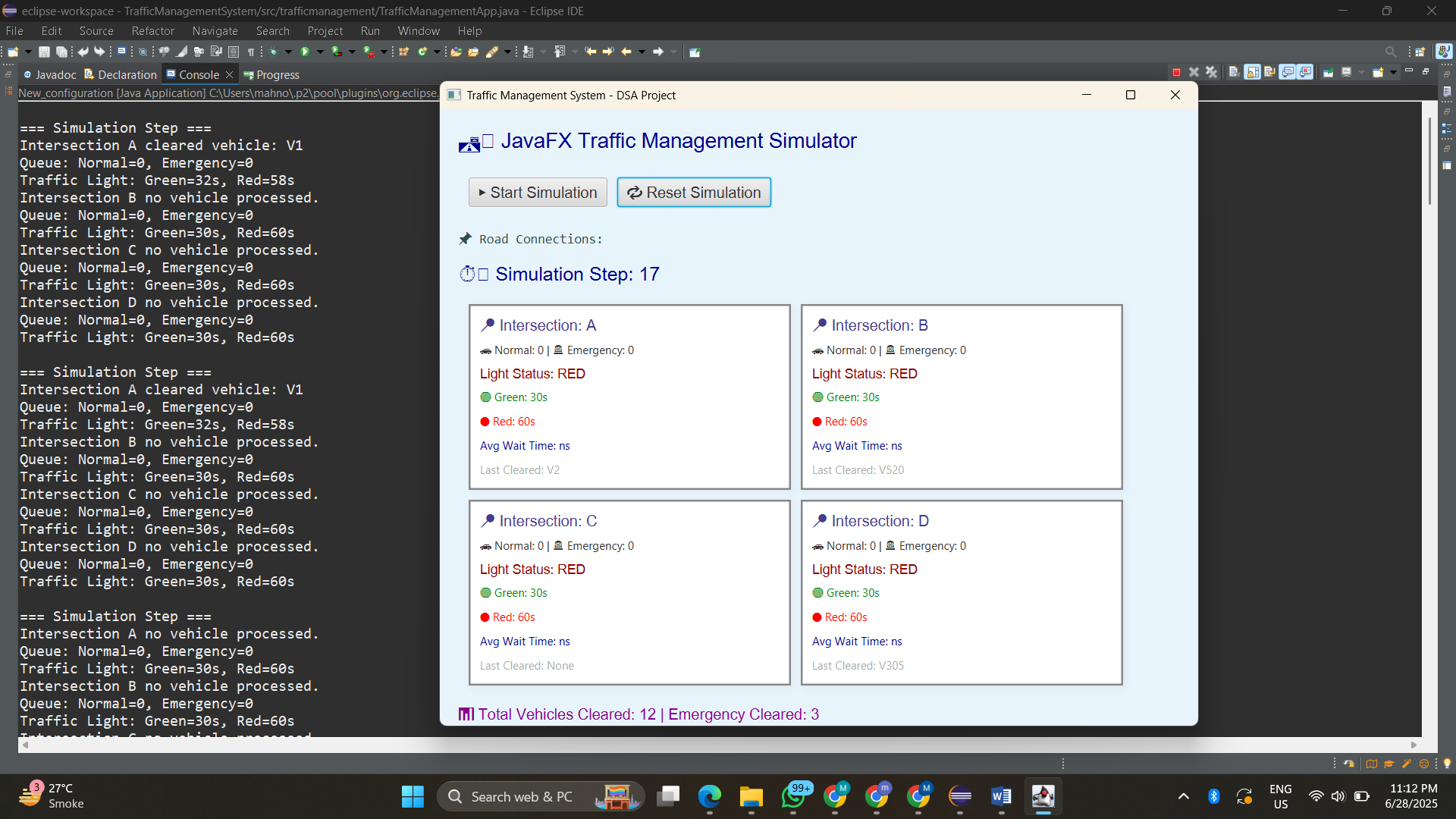
### **7. Sample Outputs**

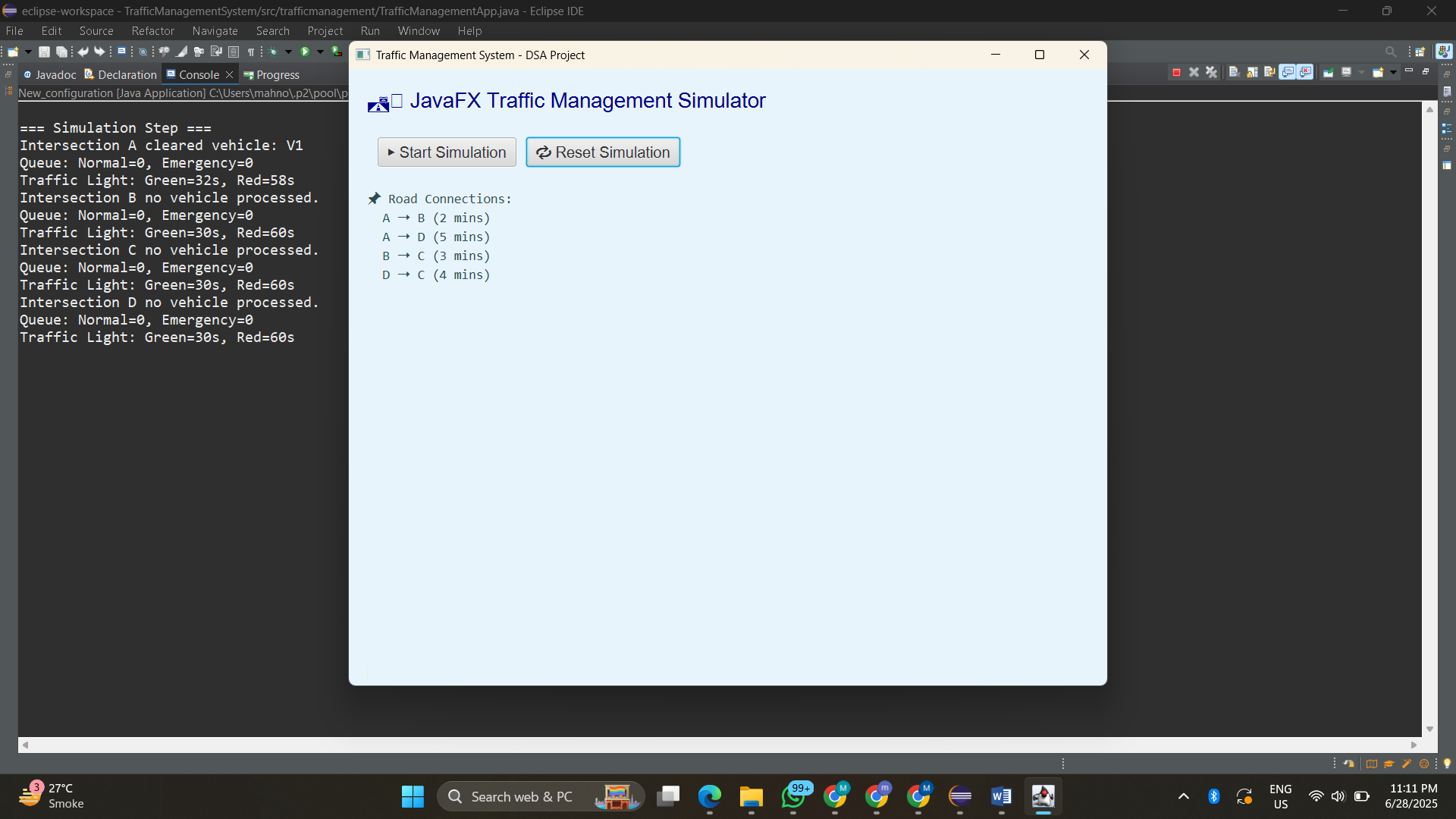
Refer to the attached screenshots:

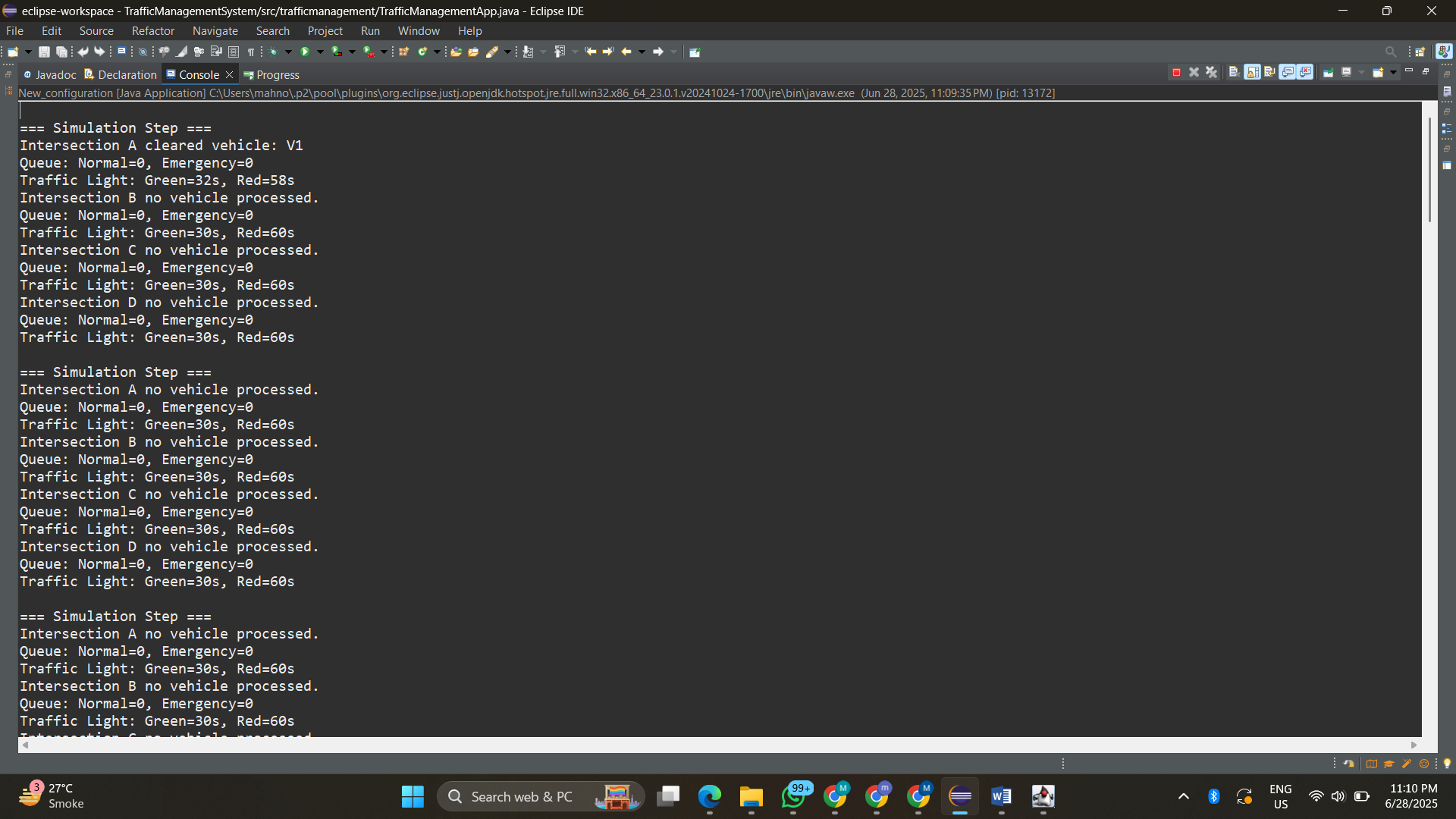
* **Screenshot 1:** Initial setup and simulation start.



* **Screenshot 2:** Ongoing simulation with vehicle stats and light states.

****

* **Screenshot 3:** Processed simulation to Restart again.
* **Screenshot 4:** Console Output.



### **8. Known Limitations**

* Vehicle movement is only simulated logically (no animation).
* Average wait time is shown as "ns" (not yet computed).