PROJECT REPORT OF EXPLORATORY PROJECT (EP)

ON

Traffic Signal Simulation System

Traffic Signal Simulation and Vehicle Flow Management

submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Sneha (2210990856)

Harsh (2210991612)

Jinny Kapur (2210990462)

Khushi (2210991796)

Supervised By:

Roshni Mali

Lead Trainer,

BridgeLabz

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHITKARA UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY

CHITKARA UNIVERSITY, PUNJAB, INDIA

CONTENTS

ACKNOWLEDGEMENT

It is our pleasure to be indebted to various individuals who directly or indirectly contributed to the development of this project and who influenced our thinking, learning and effort throughout the course of this study.

We express our sincere gratitude to the university for providing us the opportunity to undertake this Exploratory Project as a part of the curriculum and for offering us an academic environment that continuously supports learning and innovation.

We are extremely thankful to **Roshni Mali**, Associate Professor, Chitkara University, Punjab, for his constant support, cooperation, motivation and valuable guidance throughout the duration of the project. His encouragement, expert suggestions and constructive feedback played a crucial role in helping us successfully complete this work.

We also extend our sincere appreciation to all the trainers of **BridgeLabz** for their continuous support, valuable insights and assistance during the execution of our project, which helped us improve the quality of work and deepen our understanding.

Lastly, we would like to express our heartfelt thanks to the almighty and our parents for their moral support, encouragement and blessings. We are also grateful to our friends for sharing their suggestions and companionship throughout this journey, which helped us refine our work and stay motivated.

Sneha Chhabra	Harsh Dhiman	Jinny Kapur	Khushi
2210990856	2210991612	2210990462	2210991796

DECLARATION

I hereby certify that the work being presented in this project report entitled **"Traffic Signal Simulation System Using Java"**, submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering (Computer Science and Engineering)** at **Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India**, is an authentic record of our own work carried out under the supervision of **Roshni Mali**. The matter presented in this report has not been submitted to any other institute/university for the award of any degree.

Place: Rajpura

Date: _____

Signatures of Students :

Sneha Chhabra (2210990856)	Harsh Dhiman (2210991612)	Jinny Kapur (2210990462)	Khushi (2210991796)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief.

(Roshni Mali)
Lead Trainer

BridgeLabz

3. ABSTRACT

The challenge of **efficient traffic control** in growing urban areas, marked by increasing vehicle numbers and limited infrastructure, demands advanced solutions beyond traditional **fixed-timer traffic signals**. This project introduces a **Traffic Signal Simulation and Vehicle Flow Management System** built with **Java Multithreading** to address this need. Simulation-based models are crucial for replicating **real-time traffic behaviour**, enabling researchers to safely analyse and test signal timing strategies, predict **queue build-ups**, and enhance **vehicle throughput** without costly on-field experiments.

Our system models a signalized road intersection using two parallel, synchronized threads:

1. **Signal Controller Thread:** Manages the cyclic transition between **red, green, and yellow** signal states.
2. **Vehicle Generator Thread:** Randomly creates vehicles and adds them to a queue.

Vehicles progress only during the **green** phase and accumulate during the **red and yellow** phases, accurately simulating urban traffic flow. **Thread Synchronization** is used for all transitions and queue length updates, ensuring **logical consistency** and preventing **race conditions** during concurrent execution.

All key data, including signal changes and waiting vehicle counts, are logged in **real time** using a **CSV-based logging mechanism**. This dataset can be utilized to evaluate signal timing efficiency, study the impact of vehicle density on **congestion**, and serve as a foundation for developing future **smart and adaptive traffic control algorithms**.

Ultimately, this research demonstrates the practical application of **multithreading** and **concurrency control** in simulating complex real-world traffic systems, providing a **scalable platform** for future advancements like emergency vehicle prioritization, density-based switching, and AI-enabled traffic prediction.

Keywords

Traffic Simulation, Multithreading, Java, Traffic Signal Cycle, Vehicle Flow, CSV Logging, Concurrency, Thread Synchronization, Smart Traffic Systems

4. INTRODUCTION

Urban transportation is one of the core pillars of modern infrastructure, directly influencing economic growth, productivity and quality of life. However, with rapid urbanization, increasing vehicle ownership and expanding city landscapes, traffic congestion has become a major challenge for metropolitan as well as developing regions. The absence of adaptive traffic mechanisms results in unnecessary waiting time at intersections, increased fuel consumption, longer travel durations and heightened stress among road users. Traditional traffic signals operate on static timers rather than fluctuating real-time traffic variations. For example, when the signal remains red despite having no vehicles approaching from the opposite side, time and fuel are wasted. These inefficiencies highlight the need for advanced control models that are able to understand, predict and regulate road traffic systematically.

Simulation is an important tool for studying traffic flow and developing intelligent solutions without implementing changes physically on roads. By creating virtual representations of intersections, signal cycles and vehicle behaviours, researchers can analyze the direct effect of signal timing, waiting queues and traffic density on road efficiency. Simulation-driven analysis reduces risk, accelerates development and supports data-driven decision-making before real-world deployment. In the field of computer science, simulation also becomes a practical application of concurrency, synchronization, timing operations and parallel execution — forming an excellent bridge between theoretical knowledge and real-world scenario modelling.

The present project introduces a **Traffic Signal Simulation and Vehicle Flow Management System using Java Multithreading**, which replicates the behaviour of a standard signalized intersection. The model is designed around two main concurrent processes — (i) the traffic signal controller and (ii) the vehicle generator. The signal controller cyclically transitions between red, green and yellow phases based on a predefined timing pattern. Simultaneously, the vehicle generator produces vehicles at randomized intervals to create natural variations in vehicle density. Vehicles are allowed to pass the intersection only when the signal is Green, while they accumulate during red and yellow phases, similar to real-world behaviour. Both threads share a common state (signal condition and number of waiting vehicles), necessitating careful synchronization to avoid race conditions and inconsistent results.

A structured logging mechanism records each signal transition along with timestamp and queue length in a .csv file. These logs can later be used to evaluate performance, visualize congestion trends and generate statistical insights about traffic behaviour. The simulation thus provides a controlled platform for experimentation and future extensions such as adaptive signal switching, emergency-vehicle priority clearance, machine-learning-based timing prediction and IoT-assisted live input.

4.1 Background

Traffic engineering has historically relied on fixed-cycle signal timers, mathematical models and field trials for optimizing traffic flow. While effective in certain cases, these methods fail to capture sudden increases in traffic density or unpredictable queue formations. Researchers have increasingly turned to simulation-based systems to model real-time traffic patterns with accuracy. Such systems help to visualize delays, evaluate multiple timing schemes and test the feasibility of smart traffic models without affecting real road users.

In the domain of computing, Java has remained a preferred language for simulation modelling due to its cross-platform execution, thread management capabilities and reliable in-built concurrency support. By leveraging multithreading, developers can create virtual environments where independent activities — such as vehicle arrival and signal switching — occur simultaneously, just as they do in reality. This makes traffic simulation an ideal platform for practising and understanding concurrency, synchronization and modular programming.

4.2 Problem Statement

Traffic signals operating through static and fixed timing cycles are unable to adapt to varying traffic flow conditions, resulting in long queues, unnecessary waiting times, fuel consumption and reduced intersection efficiency. There is a need for a cost-effective and safe simulation environment that can replicate real-time traffic behaviour and allow researchers to test and evaluate different signal timings, queue responses and flow patterns without deploying physical systems on the road.

4.3 Objective

- To simulate the behaviour of a traffic signal and vehicle flow using Java multithreading.
- To model realistic queue formation and vehicle departure based on signal phases.
- To maintain synchronization between concurrent threads managing signal and vehicle states.
- To record vehicle counts and signal transitions in real time for analytical evaluation.
- To provide a scalable base model for future extensions including adaptive and intelligent traffic control algorithms.

5.METHODOLOGY

The methodology adopted for this project focuses on systematically designing, implementing and evaluating a virtual simulation of traffic signal behaviour and vehicle flow using Java multithreading. The workflow follows a structured development approach that ensures accuracy, modularity, and real-time responsiveness. Each phase of the methodology contributes to transforming a conceptual traffic scenario into a fully synchronized and executable simulation model.

5.1 System Requirement Analysis:

The methodology begins with identification of the functional requirements of a basic signalized intersection. The essential operations include defining the behaviour of traffic lights, generating vehicles at varying time intervals, enabling vehicles to pass only during a green signal, and logging the queue size throughout the simulation. Non-functional requirements were also evaluated, focusing on multithreading efficiency, shared state consistency, scalability, and responsiveness of the simulation. The requirement analysis highlighted the necessity of concurrency management to achieve parallel processing that accurately represents real-world traffic dynamics.

5.2 System Design

Following requirement analysis, a detailed software architecture was designed based on the object-oriented programming paradigm. The system was divided into independent and reusable modules to simplify management and future extensibility.

- The **Signal** module maintains the current state of the traffic signal and performs controlled transitions between red, yellow, and green phases.
- The **Vehicle** module represents each vehicle with a unique ID and vehicle type.
- The **TrafficService** module manages two concurrent threads — one responsible for the signal cycle and another handling vehicle generation and flow.
- The **LogService** and **FileService** modules manage real-time data storage to produce an analysable dataset for performance review.
  This modular design supports abstraction, encapsulation, and low coupling between components.

5.3 Multithreading and Synchronization Strategy

The core of the methodology lies in implementing multithreading to simulate real-time independent processes. The **signal controller thread** governs the transition between signal states at predefined time durations, while the **vehicle thread** continuously generates vehicles at randomized intervals. Both threads operate simultaneously and interact with shared resources, including signal state and vehicle queue length. To prevent inconsistent data access between threads, Java synchronization techniques were adopted.

- synchronized methods ensure safe updates to the signal state.
- AtomicInteger prevents race conditions when modifying vehicle count.
  This strategy ensures accuracy in queue size tracking and guarantees deterministic behaviour of the simulation even under concurrent execution.

5.4 Data Logging and Monitoring

To allow empirical evaluation of the simulation, a logging methodology was incorporated using CSV-based storage. Each signal transition is recorded along with real-time timestamps and vehicle count. This method converts the simulation into measurable and verifiable data, enabling later performance analysis and visualization. The log file can be used to plot traffic patterns, identify queue rise and fall trends across signal cycles, and test modified timing algorithms.

5.5 Implementation and Testing

Once multithreading and synchronization mechanisms were established, the simulation was implemented in Java and tested using multiple iterative cycles. Console-based monitoring ensured that signal transitions, queuing and vehicle departures were executed logically and sequentially. Testing involved validating:

- correctness of thread execution order
- consistency of shared state
- accuracy of logged values
- stability of simulation across long durations

Stress testing was also conducted by increasing vehicle arrival frequency to ensure system resilience against high-density flow conditions.

5.6 Validation and Refinement

After functional testing, the logged data was examined to validate queue behaviour with respect to signal duration. Observations were matched with expected traffic behaviour to confirm reliability of the simulation. Refinements were made to improve timing precision, readability of console results, and modularity of code structure. The refined model now supports scalability for future extensions such as machine-learning-based traffic optimization or dynamic signal timing.

6. TOOLS AND TECHNOLOGY

The Traffic Signal Simulation and Vehicle Flow Management System is developed using programming technologies and development tools that support concurrency, modularity, and

real-time data recording. Each tool and technology contributes to thread stability, timing accuracy, system scalability, and performance. The following technologies were used during the development of the project:

6.1 Java (JDK 8 or above)

Java is the core programming language used for building the entire simulation. It provides strong support for multithreading, platform independence and an extensive set of built-in libraries for time handling, thread control and file operations. Its stability and reliability make it highly suitable for real-time simulation environments.

6.2 Java Multithreading

The simulation is driven by two independent threads: one handling the traffic signal cycle and the other generating vehicles. Multithreading enables both operations to run simultaneously, closely mirroring actual traffic behaviour where signal transitions and vehicle arrival happen concurrently without blocking each other.

6.3 Synchronization and Atomic Data Handling

Since multiple threads interact with shared variables, synchronization mechanisms ensure data safety. The synchronized keyword prevents conflicting access to the signal state, and the use of AtomicInteger ensures thread-safe increments and decrements of the vehicle count. These techniques eliminate race conditions and guarantee consistent system behaviour under parallel execution.

6.4 CSV Logging using FileWriter

To enable future analysis and visualization, all signal transitions are logged into a CSV file along with timestamps and the number of waiting vehicles. Java's FileWriter class is used to record these logs efficiently. This ensures that insights regarding traffic density and signal timing performance can be obtained later.

6.5 Object-Oriented Programming (OOP) Architecture

The system is designed using OOP principles such as modularity, abstraction and reusability. Classes like Signal, Vehicle, TrafficService, LogService and FileService encapsulate separate responsibilities, making the code cleaner, scalable and easy to maintain. New features can be added without altering the core structure.

6.6 Integrated Development Environments (IntelliJ IDEA / Eclipse / VS Code)

Modern IDEs were used for coding, debugging and execution. These tools provide features such as thread monitoring, console output tracking, error tracing and project

structure management which significantly improved the development and testing process.

7. IMPLEMENTATION

The implementation phase focuses on converting the conceptual and architectural design of the simulation into a fully functional software application using Java. The system is developed using an object–oriented approach in which each major component of the simulation is represented as an independent class with a dedicated responsibility. This modular structure not only ensures readability and reusability but also supports long-term scalability of the project.

7.1 Package Structure

The project is organized into three logical packages:

- **com.traffic.main** → contains the main execution entry point responsible for initializing and starting the simulation.
- **com.traffic.model** → consists of core simulation models including Signal and Vehicle.
- **com.traffic.service** → contains service-level classes such as TrafficService, LogService, and FileService that implement the signal cycle, vehicle flow and logging procedures.

This structural separation ensures clear responsibility distribution among components and reduces inter-dependency.

7.2 Main Execution Module

The simulation begins from the TrafficSimulation class located in the *main* package. This module performs:

- Initialization of the Signal object.
- Creation of the TrafficService instance.
- Invocation of the traffic simulation thread scheduler.
- Support for user-driven termination through console input.

This module plays a similar role to a controller that orchestrates the complete simulation run while providing user control.

7.3 Signal Module

The Signal class represents the core of the traffic light system. It defines:

- Different signal states (RED, YELLOW, GREEN) using an enumeration.
- Methods for reading and modifying the current signal.
- Synchronization control to prevent concurrent conflicting updates.

Signal state transitions are triggered by the service layer based on predefined durations, ensuring realistic traffic light behaviour.

7.4 Vehicle Module

The Vehicle class models the vehicles arriving at the intersection. It consists of:

- A unique vehicle ID.
- A vehicle type selected randomly from (CAR, BUS, TRUCK, MOTORCYCLE). The vehicle module does not manage queue movement directly; instead, queue count tracking is handled centrally by LogService.

7.5 Traffic Service

TrafficService is the **heart of the simulation** and handles concurrency. It contains two independent threads:

- **Signal Controller Thread** — cycles through red, green and yellow signal states based on timing intervals (e.g., 5 sec Red, 7 sec Green, 2 sec Yellow).
- **Vehicle Flow Thread** — generates vehicles at random time intervals and increments queue count.

During the Green phase, the vehicle thread permits a dynamically determined number of vehicles to exit the queue, creating realistic arrival–departure behaviour.

7.6 Logging and Storage

The simulation includes a built-in logging mechanism managed by two classes:

- LogService — generates log entries capturing timestamp, current signal, and total queued vehicles.
- FileService — writes the log data to a .csv file and ensures the log directory exists.

This mechanism produces analyzable datasets representing congestion patterns and signal performance.

7.7 Concurrency and Data Integrity

Because two threads operate simultaneously on shared data, the implementation uses:

- synchronized methods in Signal to prevent race conditions during state transitions.
- AtomicInteger in LogService to guarantee thread-safe increment/decrement operations on the queue counter.

This ensures accurate logging and realistic queue dynamics even under concurrent execution.

7.8 Testing and Debugging

During implementation, console monitoring was used to:

- Track real-time signal transitions.
- Verify queue behaviour.
- Validate logging output.

Multiple test cycles were conducted with varying sleep intervals and arrival rates to ensure consistency under different traffic densities.

8.RESULTS

The execution of the Traffic Signal Simulation and Vehicle Flow Management System produced significant outcomes that highlight the behaviour of vehicles and congestion patterns under automated signal regulation. The simulation successfully demonstrated how changes in

signal duration directly influence queue formation and traffic discharge rate at an intersection. Two independent processes — signal transition and vehicle arrival — ran concurrently without conflict, validating the accuracy of multithreading and synchronization mechanisms implemented in the system.

One of the most notable findings was the direct relationship between signal phase duration and queue length. During the Red phase, the number of vehicles consistently increased as arrivals continued without any departures. During the Yellow phase, the queue stabilized briefly due to a shorter transition interval. During the Green phase, a noticeable decline in queue length was observed as multiple vehicles were allowed to depart simultaneously. This cyclic rise and fall of vehicle count closely resembled real-world traffic behaviour, proving the model's effectiveness in emulating real scenarios.

The logging system resulted in a data-driven understanding of traffic behaviour. Each signal change was recorded with a timestamp and corresponding vehicle count in a CSV file, enabling post-simulation evaluation and supporting further research. Analysis of the recorded data revealed predictable oscillation patterns in traffic load that could form the foundation for future enhancements such as adaptive timing and prediction-based signal switching. Additionally, the system remained stable even during stress conditions with increased vehicle arrival rates, affirming the reliability of synchronization and thread-safety mechanisms.

Overall, the simulation achieved its intended purpose by replicating realistic intersection behaviour, creating measurable datasets, and validating the effects of timing control on congestion. The outcomes of this project confirm that Java multithreading can serve as an efficient platform for developing intelligent traffic management simulations that support future extensions for smart city applications.

9. CONCLUSION AND FUTURE SCOPE

The Traffic Signal Simulation and Vehicle Flow Management System demonstrates that multithreading in Java provides an effective and efficient approach to modelling real-time traffic behaviour at an intersection. The simulation successfully replicated realistic vehicle flow patterns in response to traffic signal cycles, validating the relationship between signal duration

and queue formation. The concurrent execution of the traffic light controller and vehicle generator threads functioned smoothly without interference, proving the stability of synchronization techniques and shared-data handling mechanisms. The logging module further strengthened the analytical value of the simulation by creating a structured dataset representing signal transitions and queue changes over time.

The findings clearly suggest that the duration of the green phase plays a crucial role in minimizing traffic congestion, while extended red phases contribute to increased queue buildup. The simulation not only achieved its intended goal of realistically modelling a signalized intersection but also provided a platform capable of producing measurable and analysable insights. Moreover, the modular object-oriented structure of the project ensures ease of extension, adaptability and long-term usability for research and educational purposes. Overall, the project reinforces that software-based traffic simulation is a powerful tool for analysing traffic systems without the cost or risk of real-world experimentation.

Future Scope

While the current simulation successfully models basic signal-vehicle interactions, there are several possibilities for enhancement that can expand the system's functionality and real-world relevance:

- **Density-based adaptive signal timing:** The system can be extended to dynamically adjust the duration of red, green and yellow phases based on real-time vehicle density instead of using fixed time intervals.
- **Emergency vehicle priority:** Features can be added to identify high-priority vehicles such as ambulances or fire trucks and grant them immediate signal clearance.
- **Multiple-intersection simulation:** The model can be extended to simulate a network of interconnected traffic signals rather than a single intersection, enabling analysis of area-wide traffic coordination.
- **Machine Learning / AI-assisted prediction:** Historical log data can be utilized to train models that predict queue buildup and automatically adjust signal timings for improved efficiency.
- **Graphical user interface (GUI):** A visual interface can be introduced to display vehicle movement and signal transitions in real time, making the simulation more interactive and user-friendly.
- **IoT and real-time sensor integration:** The simulation can be connected to live input sources—such as traffic cameras or vehicle sensors—to create real-time measurable smart-traffic systems.

10. REFERENCES

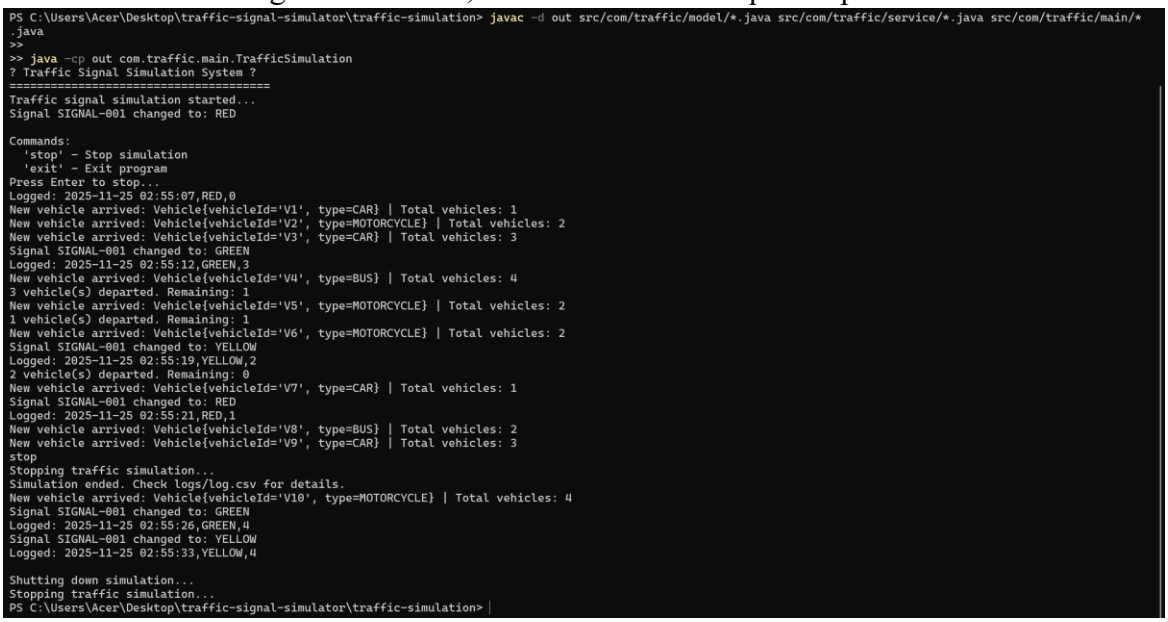
- Oracle Java Documentation – Multithreading and Concurrency. Retrieved from: https://docs.oracle.com/javase/

- Oracle Java Documentation – FileWriter and I/O Classes. Retrieved from: https://docs.oracle.com/javase/
- GeeksforGeeks – Java Multithreading and Synchronization Concepts.
- TutorialsPoint – Java Threading, Sleep(), and Synchronization.
- Stack Overflow – Community discussions on handling concurrent threads and race conditions.
- IEEE Xplore – Research papers on traffic simulation and signal optimization.
- IRJET Journal – Studies on Intelligent Traffic Management using Traffic Signal Modelling.
- Web Resources
    - https://www.javatpoint.com
    - https://www.geeksforgeeks.org
    - https://www.tutorialspoint.com
    - https://stackoverflow.com

11.APPENDIX

11.1 Appendix – A : Console Output of Simulation

This appendix contains the sample console output generated during the execution of the Traffic Signal Simulation and Vehicle Flow Management System. The output indicates real-time signal transitions, vehicle arrivals and queue updates.



11.2 Appendix – B : log.csv Sample Data

Below is a sample of the data recorded during the simulation. The complete file is stored in /logs/log.csv.

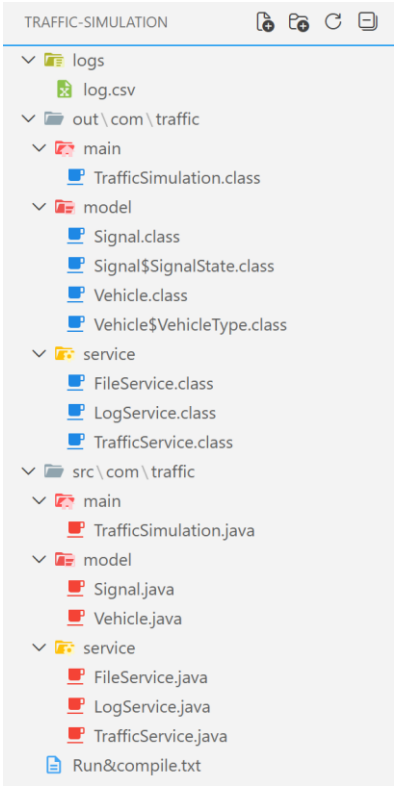
log.csv

logs > log.csv

	A	B	C
123	Timestamp	SignalState	VehicleCount
124	2025-11-25 02:55:07	RED	0
125	2025-11-25 02:55:12	GREEN	3
126	2025-11-25 02:55:19	YELLOW	2
127	2025-11-25 02:55:21	RED	1
128	2025-11-25 02:55:26	GREEN	4
129	2025-11-25 02:55:33	YELLOW	4
130			

11.3 Appendix – C : Class Diagram / Project Screenshots

- Project structure from IDE



- UML/Class Diagram