

# 1. Project Overview

## 1.1 Introduction

The SUMO Traffic System project aims to build a Java-based control and visualization layer for the SUMO (Simulation of Urban Mobility) traffic simulator. SUMO provides a powerful microscopic traffic simulation engine, but lacks a user-friendly Java interface.

This project bridges that gap by using **libtraci**, SUMO's official API, to enable real-time communication between Java and the simulation engine.

Milestone 1 focuses on laying the technical and conceptual foundation for the full application: setting up the architecture, defining core components, and demonstrating a working SUMO connection from Java.

---

## 1.2 Objectives of Milestone 1

Milestone 1 sets the groundwork for later development. Its core objectives are:

- Establish a working SUMO Java libtraci integration
- Create the initial system architecture
- Design class structures for later modules (vehicles, traffic lights, controllers)
- Develop GUI mockups for map, controls, and dashboard
- Prepare documentation and repository structure
- Define team roles and the time plan for the remaining milestones

These deliverables ensure that the team has a clear strategy before implementing advanced features.

---

## 1.3 System Description

The system consists of a Java application that communicates with SUMO using libtraci.

Key capabilities built in Milestone 1 include:

- Loading SUMO configuration files
- Starting SUMO from Java (portable installation)
- Executing simulation steps programmatically
- Preparing future modules such as VehicleManager, TrafficLightManager, and SimulationController
- Designing GUI concepts for visualization and control

This foundation allows the project to evolve toward an interactive traffic control interface with analytics and dynamic manipulation.

---

## 1.4 Architecture Summary

The system is structured into several layers:

### 1. SUMO Integration Layer

Handles communication with SUMO via libtraci: - Starting/closing SUMO - Loading `.sumocfg` files - Stepping through simulation time - Reading vehicle and traffic light states

### 2. Core Logic Layer

Contains the main program modules (planned for Milestone 2-3): - Simulation-Controller - VehicleManager - TrafficLightManager  
These components translate GUI and logic commands into SUMO actions.

### 3. GUI Layer (Future)

Based on Swing/JavaFX and designed in Milestone 1 through mockups.  
It will include: - Map visualization - Control panel - Statistics dashboard

### 4. Utility Layer

Handles configuration paths, data formatting, exporting, and support tools.  
This modular structure ensures scalability and clean separation of concerns.

---

## 1.5 Technology Stack

- **Programming Language:** Java 21
- **Simulation Engine:** SUMO 1.25.0 (portable installation)
- **Communication:** libtraci (Java bindings)
- **IDE:** IntelliJ IDEA
- **Version Control:** Git + GitHub
- **GUI Framework:** Swing (JavaFX optional)

The stack is chosen for stability, portability, and compatibility with the course requirements.

---

## 1.6 Milestone 1 Deliverables (Completed)

- GitHub repository with `.gitignore` and documentation
- Portable SUMO setup inside the project

- Working SUMO  $\longleftrightarrow$  Java integration via libtraci
- Example demo (load config, run simulation steps)
- System architecture diagram
- Class design for core modules
- GUI mockups for map, control panel, dashboard
- Technology stack summary
- Time plan for Milestones 2 and 3
- Defined team roles

Milestone 1 is fully completed.

---

## 1.7 Team Roles

- **Ilias** – Repository management, documentation, project overview
- **Selim** – Java development, SUMO integration, demo implementation
- **Yilin** – Java development, SUMO integration, demo implementation
- **Enes** – GUI mockups (map, dashboard, control view)
- **Alex** – System architecture and class design

All members contribute to testing, discussion, and refinement.

---

## 1.8 Summary

Milestone 1 successfully delivers the core planning, architecture, and technical integration required for the SUMO Traffic System. With the communication pipeline, design foundations, and structural planning complete, the project is prepared to advance into Milestone 2, which will focus on functional features such as vehicle/light control, basic GUI development, and initial analytics.