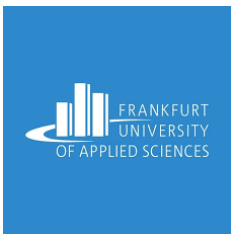
FRA-UAS
DEPARTMENT OF COMPUTER SCIENCE



REAL-TIME TRAFFIC SIMULATION WITH JAVA

SiMpFi

Instructor: Prof. Dr.-Eng. Ghadi Mahmoudi

Student 1: Kenji Bischoff - 1402720
Student 2: Nguyen Ha Khai - 1650329
Student 3: Nguyen Duy Khanh - 1648306
Student 4: To Huynh Phuc - 1651780
Student 5: Ninh Hoang Quan - 1650910

Frankfurt am Main, 11/2025

Abstract

This report provides progress, purpose, and other detailed informations about simpfi, a real-time traffic simulation app in which users can simulate real-world transportation scenarios and export useful statistics.

List of Tables

List of Figures

Contents

1 Introduction

1.1 Context

SUMO - Simulation of Urban MObility - is a traffic simulation tool used to simulate urban transportation scenarios. The application is helpful for designers, researchers who design road network and evaluate features such as CO_2 levels, traffic congestion rate, etc.

1.2 Main Issue

Although providing useful functionalities for traffic simulation, SUMO is difficult to utilize effectively. In other words, only users with programming knowledge can benefit greatly from the software.
This disadvantage can be observed by the fact that users are required to grasp a basic understanding of SUMO's configurations such as XML files and how to run command lines.

1.3 Purpose and Goals

In order to address the problem mentioned above, the team aims to create a software that leverages SUMO's features but maintains the simplicity of user interface. That is, people can perform simple operations such as drag-and-drop and clicking buttons to retrieve meaningful detailed information and represent complicated traffic networks. Consequently, more and more designs can be sketched and created quickly, which potentially speeds up one's studies or transportation planning. Furthermore, an increased number of designs means that the urban area may have better transportation plans, thereby reducing traffic congestion and CO_2 emissions.
Regarding study goals, the team plans to master in Object-Oriented Programming concepts and gain more experience with teamwork, writing documents such as javadoc and academic reports.

2 Project Overview

2.1 Description

The software **simpfi**, which stands for **Si**mulation **Ma**p Traf**fi**c, is a traffic simulation tool used to create and monitor simulated representation of real-world networks. For clarity, the application's core is based on SUMO, but the difference is that it is developed to suit a wider range of target users from non-coders to experts.

2.2 Technology Stack

There are three main components the team uses for the project: **Java**, **TraaS**, and **SUMO**. Additionally, the team uses **GitHub** to create a shared repository for members to submit codes, and **Notion** to distribute tasks.

2.3 Team Roles

The below table shows the main role of each member during the milestone 1, note that although these are specific-domain roles, the team tends to overlap the work and responsibilities in order to fully experience how to work in real-world projects:

ID	Name	Main Role
1402720	Kenji Bischoff	Documentation and Analysis
1650329	Nguyen Ha Khai	Documentation and Designer
1648306	Nguyen Duy Khanh	Frontend
1651780	To Huynh Phuc	Coordinator and Technical Lead
1650910	Ninh Hoang Quan	Backend

Table 1: Team Roles

2.4 Time Plan

For the detailed time plan, readers can see our activities in this Notion page: `https://www.notion.so/2a62b60367bb80a58886c6713e08ba54?v=2a62b60367bb80db8153000caff77c02&source=copy_link`

3 Architecture Diagram

Here is the architecture diagram of the project:

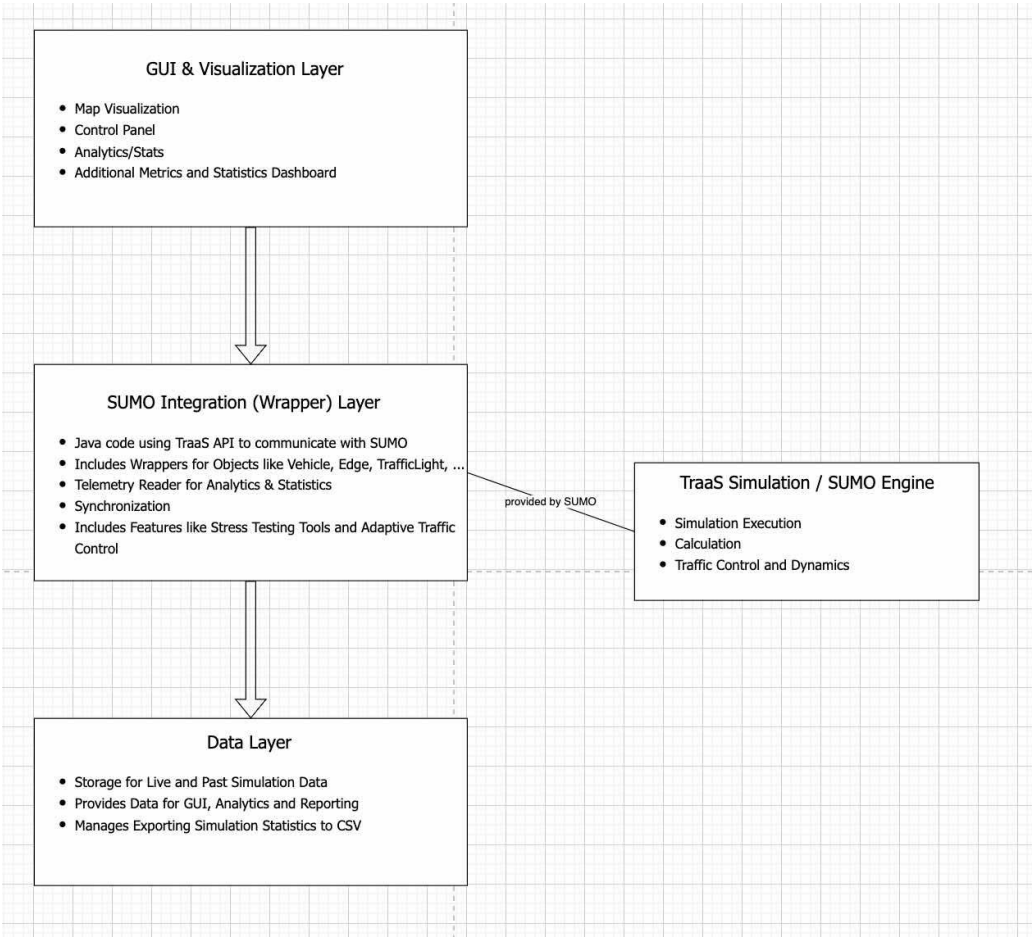


Figure 1: Architecture Diagram

From the diagram, three main layers can be observed: GUI & Visualization, SUMO Integration, and Data. Each layer provides separate methods in order to guarantee modularity and coherence in the system's design.

3.1 GUI & Visualization Layer

The main task of this layer is to present an user-friendly UI for users. In particular, it includes map visualization, multiple panels such as **Inject** so that users can add vehicles to the network. Additionally, users can move and scale the map by using keyboard shortcuts like `Ctrl =`(`command =`) to increase its size.

3.2 SUMO Integration (Wrapper) Layer

The layer is where TraCI connection takes place. Put differently, it is the interface where SUMO and wrapper classes communicate with each other. From there, the simulation can be executed, logical operations like numerical calculations can be performed, and network controls can be handled flawlessly.

3.3 Data Layer

As its name suggests, this layer handle data-related operations such as live storage and past storage, which is essential for exporting meaningful statistics. Also, the layer is developed to ensure the consistency and integrity of data throughout different parts of the system.

4 Class Design

4.1 Overview

Besides core wrapper classes, the team decided to draw the user interface by defining and connecting classes such as buttons, panels,... As a result, the number of classes was up to 30, which would be excessively complicated and difficult to understand if the team sketch them in one single class diagram. Therefore, a wrapper class diagram was designed to ensure the conciseness and readability.

4.2 Wrapper Class Diagram

The class diagram is shown as follows:

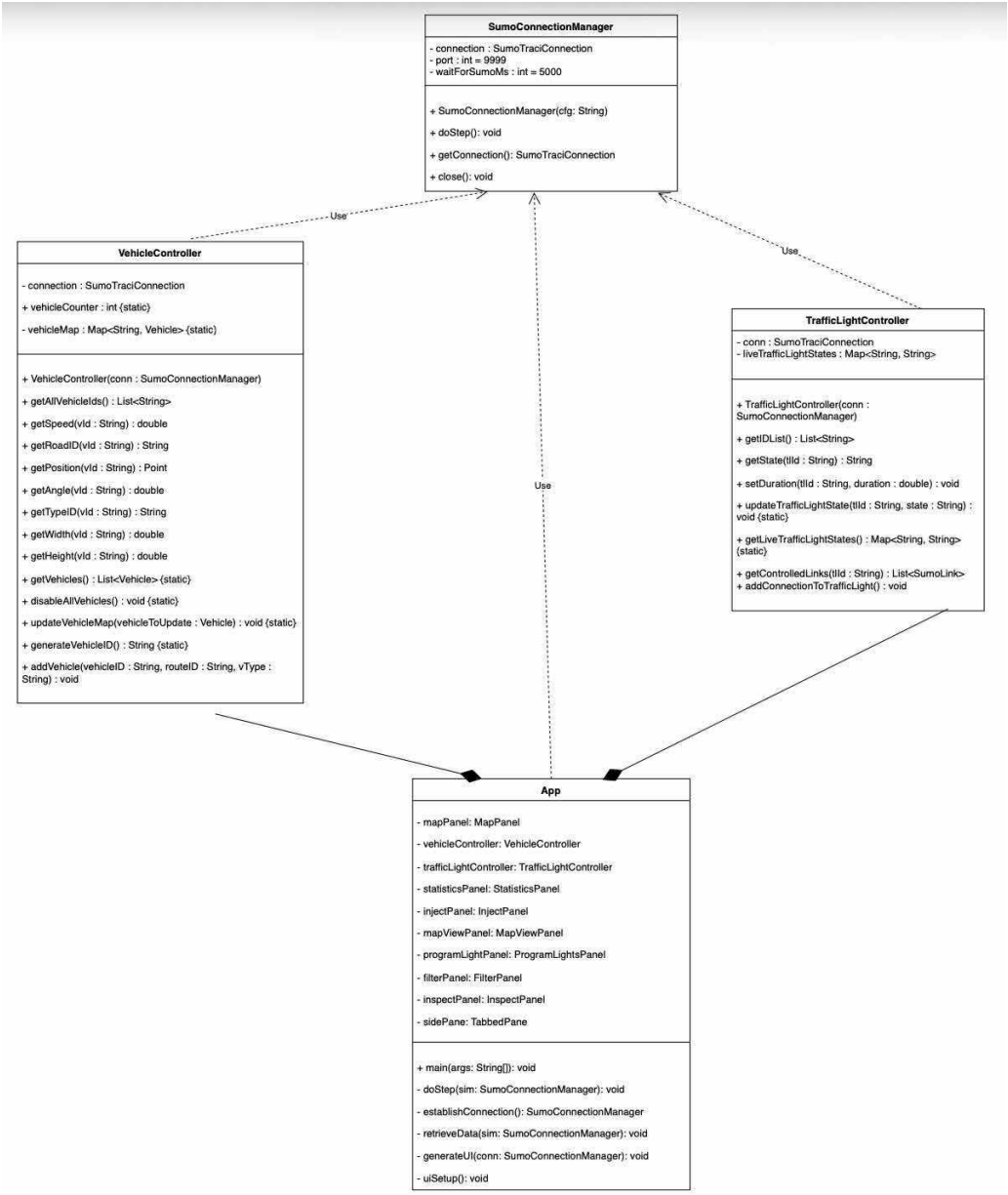


Figure 2: Wrapper Class Diagram

4.3 Class Responsibilities

- **VehicleController**: its main role is to process and retrieve detailed information regarding vehicles. To achieve this, the class has a `SumoTraciConnection` attribute `connection` used to actively connect to SUMO so that getters such as `getSpeed()`, `getRoadID()` can be implemented. Additionally, `vehicleCounter` and `vehicleMap` are developed to keep tracks of

vehicles number and mapping. Furthermore, the class provides several useful methods like `addVehicle()` for users to perform vehicle-related operations.

- **TrafficLightController**: as its name suggests, the class deals with traffic light objects, especially in the real-time scenarios since it has a live traffic light mapping: `liveTrafficLightStates`. Similar to `VehicleController`, the class establishes a connection with SUMO using the attribute `connection` and therefore is able to implement getters such as `getIDList()`, `getState()`. Besides, `updateTrafficLightState()`, `getLiveTrafficLightStates()` are provided to maintain the logic of SUMO traffic lights in the Simpfi's map.

- **SumoConnectionManager**: is the class that creates and manages the TraCI connection between SUMO and the project's app. This is achieved by passing a XML configuration file to its constructor and then the program can initialize the connection by `getConnection()` method and close it safely later using `close()`. Moreover, it provides an *important* method `doStep()` which advances the simulation by one timestep.

- **App**: is where the main function resides. Particularly, it connects the frontend (`mapPanel`) and the backend (`vehicleController` and `trafficLightController`); defines the TraCI connection; and handles the connection loop so that the software can function properly.

5 GUI Mockups

Below is the intended final user interface of the project:

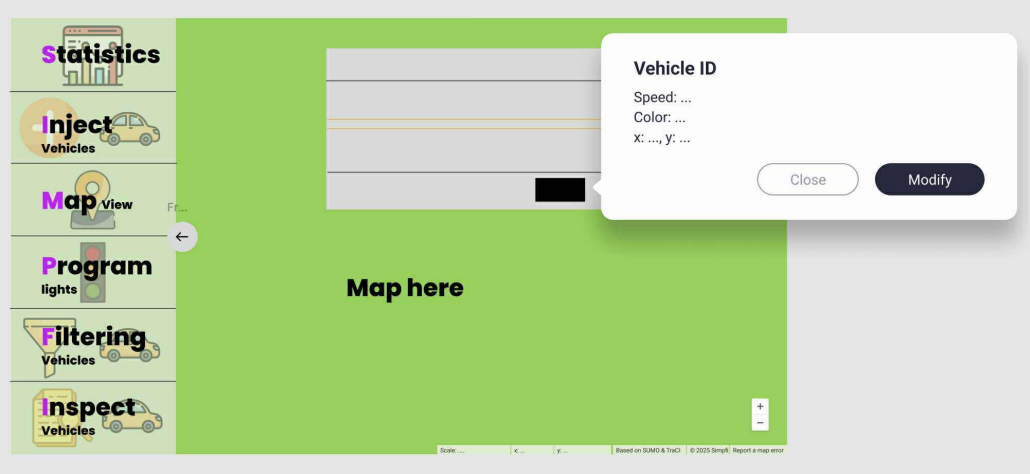


Figure 3: User Interface

6 User Guide

6.1 Introduction

6.1.1 Overview

This application provides a simple graphical interface for controlling, visualizing, and analyzing SUMO traffic simulations. Users can inject traffic, program traffic lights, and monitor simulation statistics in real time.

6.1.2 Target Audience

Our project is aimed at users working with traffic and urban mobility simulations, such as:

- Designers.
- Transportation Researchers.
- Environmental Analysts.
- Students.

6.1.3 Features

- Real Time Map Visualization.
- UI including:
  - Vehicle Injection
  - Vehicle Inspection and Filtering
  - Traffic Light Programming
  - Statistics Panel
  - Map View settings

6.2 Setup

6.2.1 Software Requirements

1. Java 17 (JDK) or higher.
2. SUMO software.
3. IDE for Java.

6.2.2 Installation Guide

Make sure that you download the versions compatible with your operating system.

1. Install Java Development Kit (JDK from oracle).
2. Install SUMO (Simulation of Urban MObility).
3. Install an IDE of your choice to run the Java project (Eclipse, VSCode, IntelliJ, ...).
4. Install `flatlaf-3.6.2.jar`.

6.2.3 Project Setup

The last step of the setup is to download our project by cloning it from GitHub. After that, open it up in your IDE.
For running the application, simply run the class "App.java".

6.3 Getting Started

After successfully setting up and running the application, you will find yourself directly on the User Interface, including various panels on the left side:

1. Statistics
2. Inject
   The Inject Panel is used to create a new Vehicle and place it on Route. It displays two mandatory fields — Vehicle Type and Route — that the user must set to place a vehicle in the simulation. From the dropdown menu, select the vehicle type and its route. By pressing "Adding vehicle" you inject the vehicle on the selected route.
3. Map View
4. Program Lights
5. Filter
6. Inspect

On the right hand side you see the map, which you can move with the arrow keys. You can also zoom with `Ctrl =` to zoom in as well as `Ctrl -` to zoom out.

7 Conclusion

From an objective perspective, the team is progressing at a reasonable rate and therefore plausibly finish the project within the target date.