Background:

1.LPS

Logic Production Systems is a LPS is a logic-based framework for programmingdatabases and AI (intelligent agent) applications[1]. LPS combines logic program-ming (as in Prolog) and production systems as a model of human thinking.A production system are computer language that are widely employed for represent-ing the process that operate in models of cognitive systems. In a productions system,all of the instructions take the formIF << condition >>, T HEN << actions >>. Thatis to say if certain conditions are satisfied then take the specified actions abbreviatedasC−> A[2]. LPS tries to close the gap between the logic-based system and statetransitions system using reactive rules and logic programming. Computation gener-ates a model of world described by a logic programs (beliefs) to make reactive rules(goal) true.

2.Traffic Regulation

In cities, where the number of vehicles continuously increases faster than the avail-able traffic infrastructure to support them, congestion is a difficult issue to deal withand it becomes even worse in case of car accidents. However, among all these trans-portation facilities, automotive vehicles are still the most adopted due to its comfortand practicality More than one half of the students(54%)had been involved in RTAs;22% out of these had been injured and 13% had been admitted into hospital for 9.3days. More than 50% of them indicated that speed was the main cause of the RTA2

CONTENTS2. SUMMARY OF PROPOSAL

1. Roundabouts are where number of junctions meets from different direction.

Roundabout are design for number of juctions meets from different dirctions.Roundabouts are designed to keep the traffic flow in clockwise. The circularroad is a one-way street and maybe made up of one or more lanes. So whenapproaching to enter a roundabouts thiking of joinning a one way road. Thetraffic entering the roundabouts must give way to traffic already on the round-abouts approaching from the right For country which the main road is left theirit is usually anticlock wise for the roundabouts.

2. T-junction

3. cross-junction

3.LSP.JS, LPS Studio, LPS Cli

LPS.JS is the JavaScript interpreter most of the function works fine except the peer to peer function.

LPS studio is the visualisation tool for LPS program. Since the p2p function is not working at this moment. P2P services is considered to be re-implemented in the lps visualisation system. Because the visualization is not part of the LPS program and the visualisation as treated as a set of fluent which run current with the LPS program it has a certain of problems that encountered. Including the conflict between the print statement in the LPS program. The malfunctioning in the recursion call etc.

LPS CLI tool gives the interface of testing and running the LPS interpreter in command line level including. It is a lower level of running time implement of lps.js. However the code is not documented at all, so it takes a long time to understand the logic in the code. Code is not documented either.

LPS web version due to the outdated dependency the whole program cannot be used anymore. The main diagram visualization will crash. Also, the code is not documented.

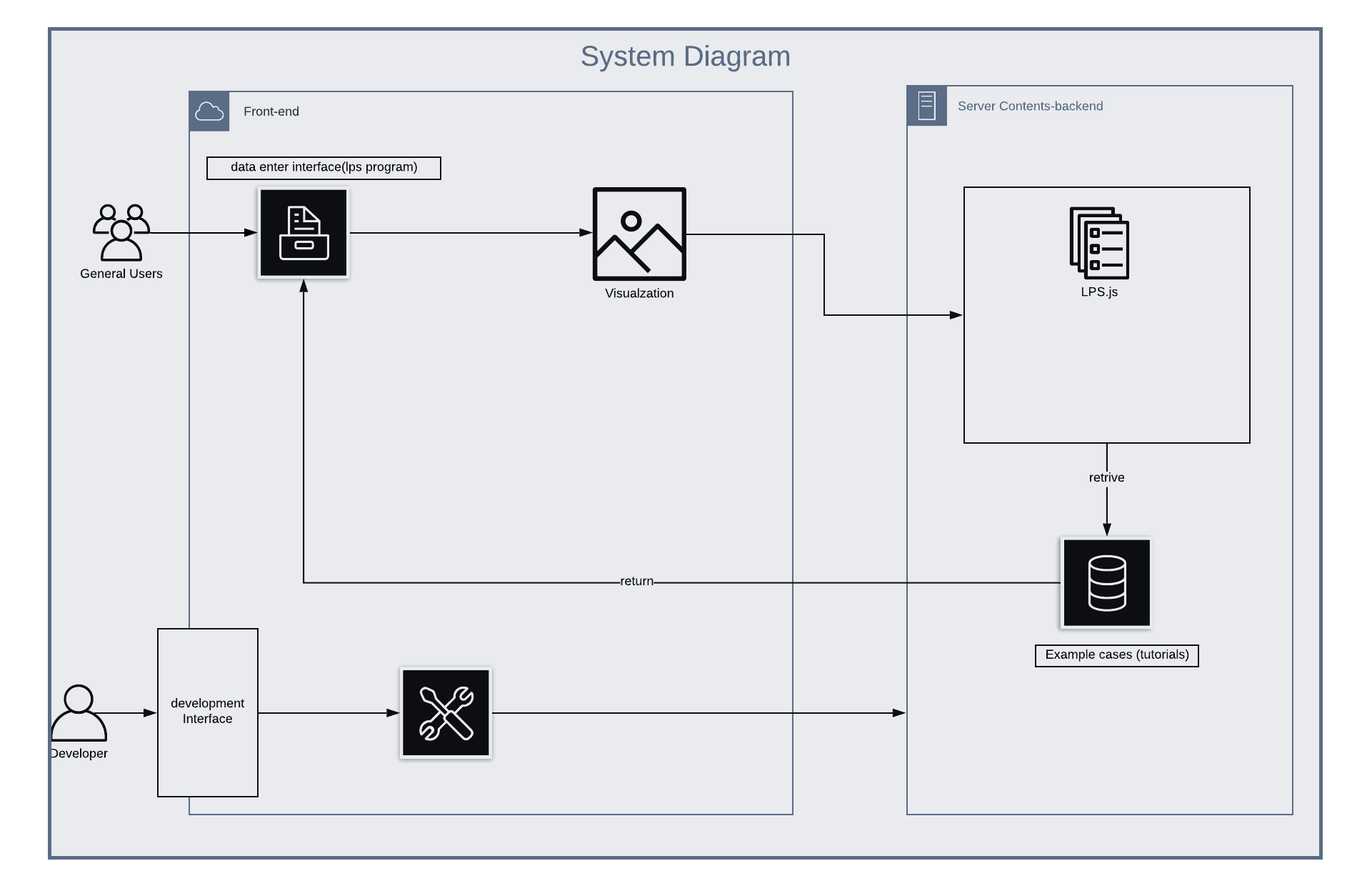
The testing function which has already implemented are not useful. Since all the testing is testing the class types none of the test function is about logic and functionality. Also, there most of the tests are aiming to check whether there is the specification file generated using lps-generate-spec function which makes no sense to LPS visualization.

Summary of proposal:

The main objective of this project is to implement a visualization tool for LPS interpreter in JavaScript. Since all the previous code is not documented and based on the point in the background section, spending time to fix the bugs which left from a lot of legacy code is not desirable. A new program should be developed, because abandoning all the previous code and change all the structure is very time consuming and risky. This project intends to only use the function which works fine in LPS.js and re-implement the visualization in the front-end side. Moreover, a tutorial and a well-documented code will also be available. If time allows the tutorial will be hosted on a server, otherwise it will be at least on a static web.

To achieve this objective, the following was required: Understand the LPS syntax and semantics

* Understand the LPS.js interpreter where the backend will be running.
* Understand Node.js runtime where the backend will be tested and running on.
* Design an easy-to-use and well documented front end and back end.
* Design a suitable system for the LPS visualization
* Preparing different cases of the different scenarios for traffic as described in section

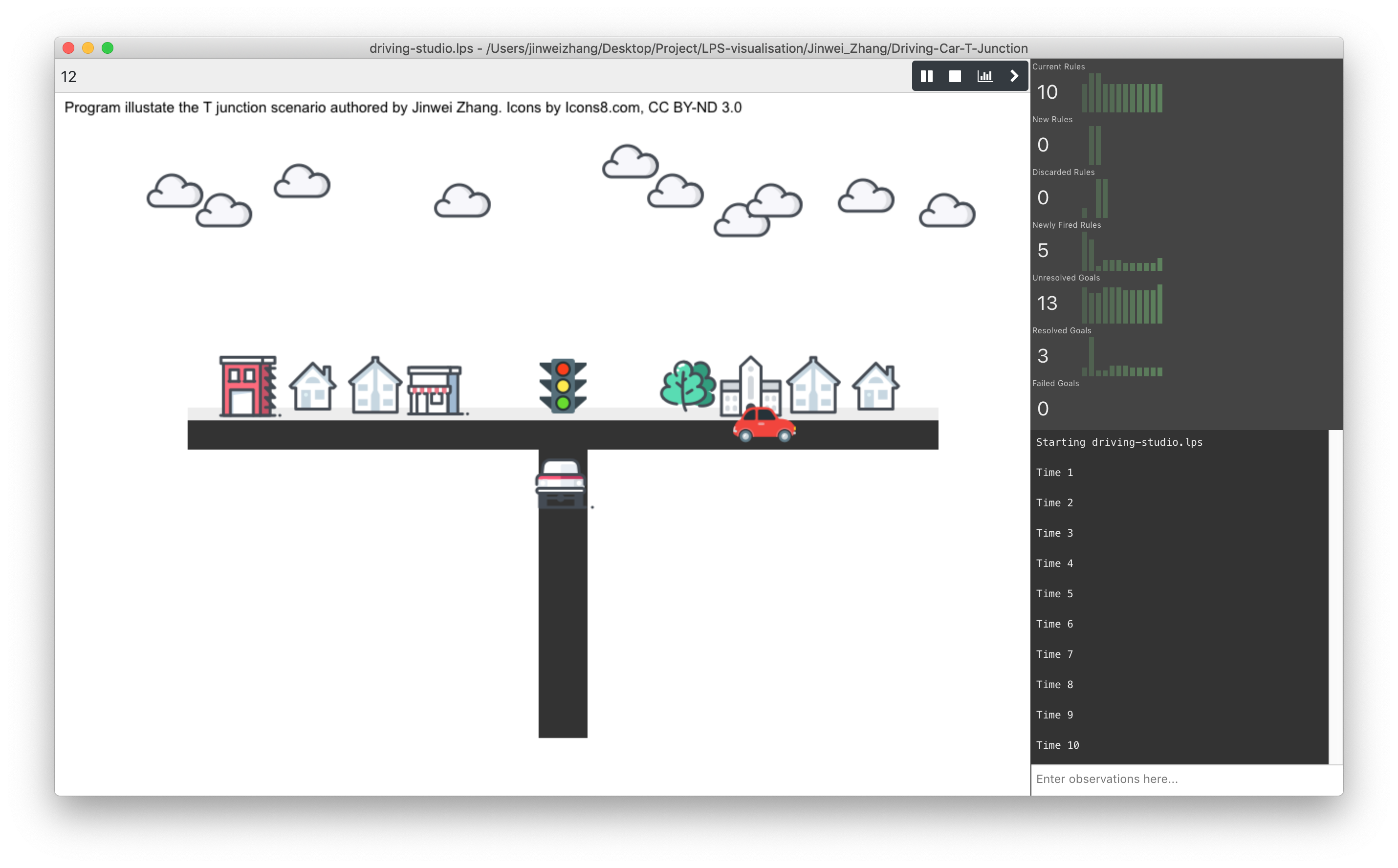


This is the system diagram: the basic flow is that the user has input only to the LPS program strict to the traffic visualization. Once the front side has the LPS code the code will be transfer to the LPS.js interpreter then phrase to AST-node. An asynchronous indicating function call will be called only the backend start to generate AST-node node. Error handling needs to be done to indicate the user whether the program is valid. Once the backend finished running the valid LPS program, the user will be promoted a visualisation function button that user will be able to visualise the LPS code.

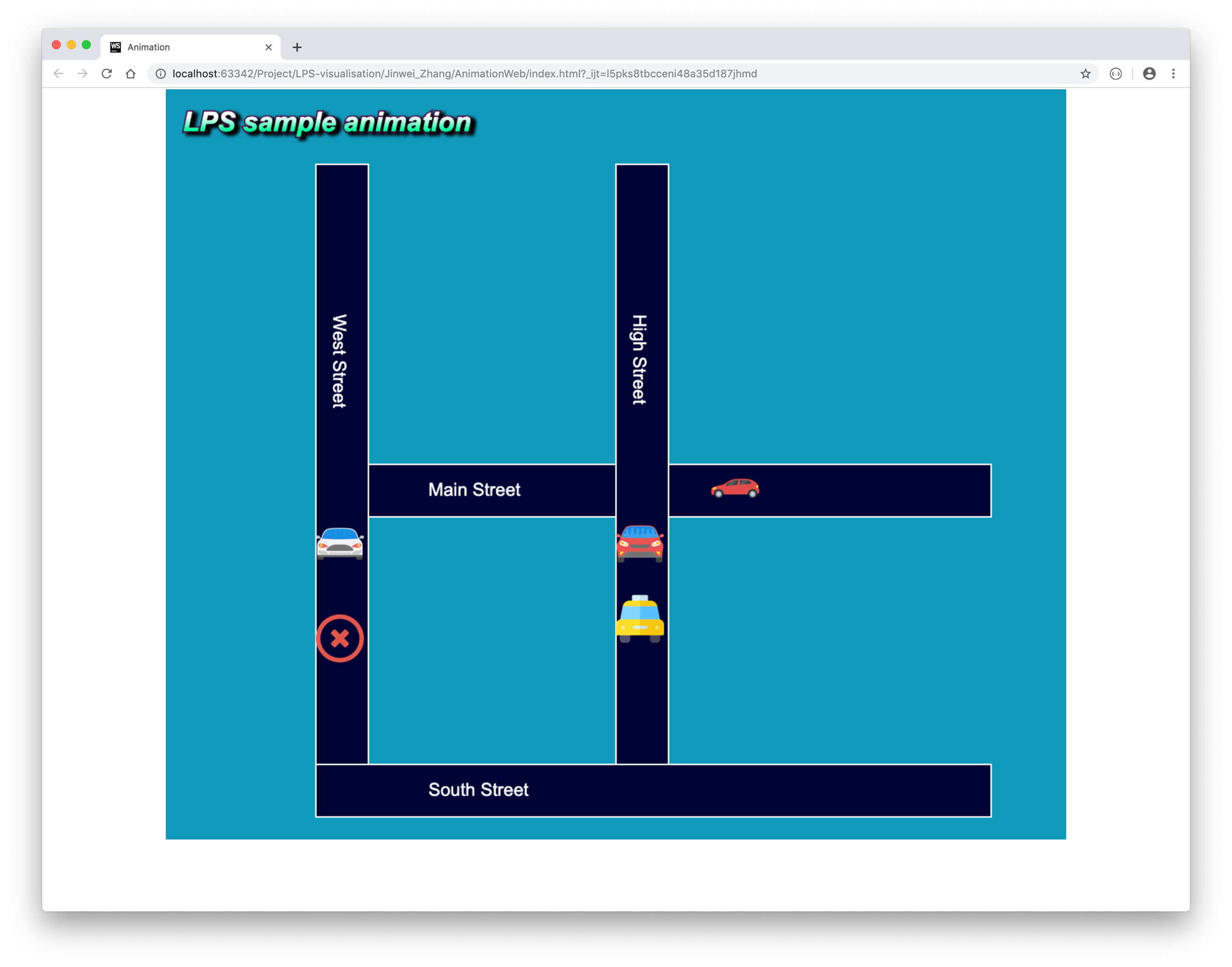
Intermediate report:

Some works as been done so far:

* Research about the traffic rules and regulation
* Simple visuzation case base on LPS studio show as follow:
* 



* Research about the classic front-end design which from up the idea of using pixi.js to do visualization. A simple case demo has given as follow:



* Most of the time spend on finding bugs in LPS.js LPS web, LPS Cli p2p tracking services. However most of bugs are easy to avoid and hard to change because the high coupled structure and un document code.

Evaluation: To summary the problem and avoid the problems in the future, A good project should at least fulfill the point below.

* Usability

Easy to use up to an educational level. People have no programming experience should be able to program in LPS and visualise the code after going through the tutorial provided. LPS visitation should have well documented code.

* Efficiency

This means the code should is efficient to development low coupled and low coherence. Program should be hosted on a server and running in a reasonable amount of time. Also it should be effienct to be extended

* Reusability

As we have desctibe the problem about the previous visualization tool LPS.studio and the lps web version. It should be easy to extend. All the back-end development should be based on the lps.ls. Code for LPS.js is not documented. Development might be too quick to documented the code hence there are too many over-line function and useless variable and unnecessary function call-back etc. LPS visualization should keep the code as simple as possible also well documented. Testing should be formed to testing some key boundary as well the unit testing.