

Florida International University
School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Traffic Simulator 2.0

Team Members: Matt Thomson, Rolando Carralero

Product Owner(s): Kianoosh G. Boroojeni

Instructor: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) 2016 *Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

This document presents the information necessary to gain a good understanding of the web server, API, and client simulation app.

In the near future, we will see autonomous e-vehicles which self-navigate themselves based on the traffic flow of streets. Under these circumstances, an open network platform is how the vehicles should be navigated such that they can avoid/mitigate traffic congestion in densely populated areas. This problem can't be solved using classic network routing problems as the traffic pattern of vehicles is oblivious (unknown) or highly stochastic. Our product provides a smarter navigation experience with real-time smart grid routing analytics for driverless cars to disperse traffic in the most beneficial way for society.

Table of Contents

INTRODUCTION	
CURRENT SYSTEM	5
PURPOSE OF NEW SYSTEM	5
USER STORIES	
IMPLEMENTED USER STORIES	6
PENDING USER STORIES	17
PROJECT PLAN	
HARDWARE AND SOFTWARE RESOURCES	16
SPRINTS PLAN	19
<i>Sprint 1</i>	19
<i>Sprint 2</i>	19
<i>Sprint 3</i>	20
<i>Sprint 4</i>	21
SYSTEM DESIGN	
ARCHITECTURAL PATTERNS	22
SYSTEM AND SUBSYSTEM DECOMPOSITION	22
DESIGN PATTERNS	23
SYSTEM VALIDATION	24
APPENDIX	31
APPENDIX A - UML DIAGRAMS	31
APPENDIX B - USER INTERFACE DESIGN	36
APPENDIX C - SPRINT REPORTS	37
APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS	40

INTRODUCTION

Current System

Version 1.0 of the Traffic Simulator was a collection of three major parts, algorithms, web server api, and simulation webpage. The web server was written in Python 3.5 utilizing Flask. The algorithms were written in Python 3.6+. And the simulation webpage was written in JavaScript utilizing Leaflet and MapQuest.

Purpose of New System

The first major priority of the new system is to get all of the code into a single web application. To do this the system will be unified under a NodeJS and ExpressJS web server. The front end handling the simulation display will be served by the web server as an AngularJS application. The back end API and algorithms will be handled by ExpressJS and a Python wrapper to run the algorithms in their native language of Python.

The new system will also display a side by side comparison with Google Map's Navigation. The simulation will run off of Google Maps and no longer be coupled with Leaflet and MapQuest.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

User Story #237 - Rewrite API in NodeJS

As a developer, I want to rewrite the 1.0 API in NodeJS, because of experience and familiarity in software and to remove some complications that arose from running Flask.

Acceptance Criteria

The API is supported by a NodeJS web server running ExpressJS that supports the following endpoints.

1. [POST] /graph – application/json
2. [POST] /path – application/json
3. [GET] / – (simulation)

Use Case

- Name: Post /graph endpoint
- Actor: Client App or 3rd Party
- Description:
 - Actor makes a post request to /api/v2/graph endpoint
 - Sends adjacency matrix in json format as adjMatrix
 - Initializes graph #238
 - Responds with success message

Use Case

- Name: Post /path/dijkstra endpoint
- Actor: Client App or 3rd Party
- Description:
 - Actor makes a post request to /api/v2/path/dijkstra endpoint
 - Sends source and destination as lat/lng pairs
 - Utilizes Dijkstra's algorithm #245

- Responds with array of lat/lng pairs

Use Case

- Name: Get / endpoint
- Actor: Client App
- Description:
 - Actor makes a get request to / endpoint
 - Responds with client app

User Story #238 - [API] Initialize Graph

As a developer, I want to be able to initialize my own graph, because I want to be able to customize the graph in which I use for my application.

Acceptance Criteria

1. [POST] /graph accepts geojson and generates the graph

Use Case

- Name: Initialize Graph
- Actor: Client App or 3rd Party
- Precondition:
 - Adjacency Matrix is properly formatted.
- Description:
 - Actor makes a post request to /api/v2/graph endpoint
 - Sends adjacency matrix in json format as adjMatrix
 - Spawns child python process to initialize the graph and return computations provided by version 1.0.
 - Responds success message

User Story #239 - geojson formatting

As a developer, I want to be able to utilize the common geojson format for graph representation, because it is a commonly utilized standard for representing geographical features.

Acceptance Criteria

1. Convert geojson data to a usable graph representation.

Use Case

- Name: Convert GeoJson
- Actor: Client App or 3rd Party

- Precondition:
 - GeoJson is properly formatted.
- Description:
 - Actor makes a post request to /api/v2/geo endpoint
 - Sends geojson
 - Utilizes version 1.0's algorithm to convert geojson to adjacency matrix in json format.
 - Responds with adjacency matrix.

User Story #240 - [API] Use graph

As a developer, I want to be able to use my initialized graph, because I want to be able to use the graph in which I specify.

Acceptance Criteria

1. After initialization, the initialized graph is used for the algorithms.

User Story #243 - [API] Python Wrapper

As a developer, I want to be able to write algorithms in Python, because Python provides useful numeric computation libraries such as numpy.

Acceptance Criteria

1. Python algorithms run from NodeJS.

Use Case

- Name: Python Wrapper
- Actor: System
- Precondition:
 - Data required for pending algorithm is formatted properly.
- Description:
 - System invokes a python algorithm through its respective NodeJS wrapper.
 - NodeJS spawns a child python process and performs I/O to exchange information.
 - NodeJS wrapper returns appropriate data.

User Story #245 - [API] Use Dijkstra Algorithm

As a developer, I want to be able to utilize the Dijkstra Navigation algorithm, because I want to use this algorithm in my pathing decisions for my application.

Acceptance Criteria

1. Uses the Dijkstra algorithm to determine path.

Use Case

- Name: Dijkstra
- Actor: Client App or 3rd Party
- Precondition:
 - Graph is initialized already.
- Description:
 - Actor makes a post request to /api/v2/path/dijkstra endpoint
 - Sends source and destination as lat/lng pairs
 - Python algorithm from version 1.0 calculates the results.
 - Responds with array of lat/lng pairs representing the path between source and destination.

User Story #264 - [Simulator] Add input fields to enter the data for the maps simulation

As a user, I want to be able to enter my origin address and my destination address to create my trip so I can visualize possible routes

Acceptance Criteria

1. Have 3 input fields, to enter the trip data, source, destination and trip time.
2. User must be able to enter as many trips as desired.
3. Show trips entered in a textarea.

Use Case

- Name: Enter data to input fields
- Actor: User
- Preconditions:
 - Web page running at localhost:8080
 - User is at the homepage
- Description:
 - User enters input to source address field
 - User enters input to destination address field
 - User enters input to time field
 - User selects addTrip button to print the trip created

User Story #266 - [Simulator] Add a reset button to reset data entered and results

As a user I want to be able to reset the information entered before, either because I made a mistake, or because I just want to enter new input to make the calculations.

Acceptance Criteria

1. Reset button should return the input fields and textarea showing the trips, back to its initial state

Use Case

- Name: Clear all data from input fields and list of trips
- Actor: User
- Preconditions:
 - User is at the homepage
 - User has entered some data in the input fields already and/or print some trips
- Description:
 - User select Reset button
 - System responds by clearing all data entered.

User Story #268 - [Simulator] add radio buttons to switch algorithms

As a user I want to have the opportunity of selecting which algorithm I want to use to get the best route for my entered trip and compare it with the google maps algorithms.

Acceptance Criteria

1. Radio buttons with the 2 algorithms and being able to display the algorithm selected by the user in the Oblivious Navigation Map

User Story #277 - [Simulator] Setting the app with angularJS

As a developer, I would like to set the app to work with angularJs so in the future will be easier to create routes and components and the web page will be more responsive and faster.

Acceptance Criteria

1. Having components working with AngularJs
2. Be able to separate the web page in different components (views)

User Story #278 - [Simulator] Create a submit button to send data entered and display car paths and simulations

As a user I want to save every input trip entered and submit it when all the information required is completed with a button, so it can be displayed on the maps.

Acceptance Criteria

1. Displaying all the information recollected in the input fields such as source and destination address for every trip in the maps and showing the car animations when clicking the submit button

Use Case

- Name: Submit data entered by user
- Actor: User
- Preconditions:
 - User is at the homepage
 - User has entered at least one trip with the three fields required: source, destination and time.
 - Algorithm selected for Oblivious map navigation.
- Description:
 - User selects Submit button
 - System responds by displaying route/s for each trip entered in the maps.

User Story #276 - [API] Utilize Sessions

As a developer, I want to utilize sessions to store graph information, because I want to be able to quickly import/export graphs after initialization.

Acceptance Criteria

1. Server sessions are utilized to store graph information after initialization.

User Story #279 - [Simulation] Moving Cars

As a user, I want to be able to see the cars moving along the path chosen by the algorithm.

Acceptance Criteria

1. Cars move smoothly from source to destination following the given path.
2. Cars move in a base unit of time (1 second) from point A to point B regardless of distance.
3. Cars disappear one second after reaching destination.

Use Case

- Name: Car Animation
- Actor: User
- Preconditions:
 - User has submitted at least one trip.
- Description:

- A Google Maps marker will appear at the specified location based on the chosen algorithm.
- The marker will animate to the next point in the path after 1 second.
- The marker will disappear after 1 second after reaching its destination.

User Story #265 - Split window pane for the 2 maps

As a user I want to see and compare the resultant routes from google maps(map displayed on the left) and from oblivious navigation algorithms(map displayed on the right), so I can decided which route is better for me.

Acceptance Criteria

1. Two maps displayed in the screen, one map to show google navigation algorithm, the second map to show oblivious navigation algorithm

User Story #269 - [Simulator] Switch from MapQuest to Google Maps

As a developer I want to use a more clean, simple and complete mapping service.

Acceptance Criteria

1. Use google map instead quest Map for the mapping service
2. More clean, simple and complete mapping service

User Story #287 - Show routes in google map navigation based on user inputs

As a user I want to see the best route calculated by google map navigation and be able to compare with results from oblivious navigation maps.

Acceptance Criteria

1. Showing routes from source address to destination address entered by users on the google map

Use Case

- Name: Show routes for trips
- Actor: User
- Preconditions:
 - User is at the homepage
- Description:
 - User enters at least one trip with the three fields required: source, destination and time.

- User selects add Trip button
- System responds by showing/printing trip added in the textarea.
- User selects Submit button.
- System responds by displaying routes for each trip on the maps.

User Story #280 - Car Paths

As a user, I want to see the correct path chosen by my choice of algorithms, because I want to see the difference between algorithms.

Acceptance Criteria

1. Utilizes the server API for getting paths.
2. Animates the car along the path

Use Case

- Name: Showing car paths using Dijkstra algorithm
- Actor: User
- Preconditions:
 - User already entered at least one trip with the three fields required: source, destination and time.
- Description:
 - User selects Dijkstra radio button
 - User selects Submit button
 - System responds by displaying car paths determined by Dijkstra algorithm and car animations along the paths on the Oblivious map navigation

Use Case

- Name: Showing car paths using Oblivious algorithm
- Actor: User
- Preconditions:
 - User already entered at least one trip with the three fields required: source, destination and time.
- Description:
 - User selects Oblivious radio button
 - User selects Submit button
 - System responds by displaying car paths determined by Oblivious algorithm and car animations along the paths on the Oblivious map navigation

User Story #282 - GeoJson for roads

As a developer, I want to be able to convert geojson into a graph with vertices indicated by (lat, lng) coordinates, because I need to be able to plot car paths and animations.

Acceptance Criteria

1. Takes input of geojson and outputs a set of vertices and paths.

Use Case

- Name: Intersections
- Actor: Developer
- Preconditions:
 - Actor has geojson for roads
- Description:
 - Actor submits the geojson for the roads.
 - System provides a file download for the intersections.

User Story #289 - Implement autocomplete feature from ng-map for input fields

As a user I want to be able to use autocomplete when typing the addresses to create my trip so I can avoid mistakes and enter not existing addresses.

Acceptance Criteria

1. Start showing real addresses as user is typing.
2. Make sure users enter existing address and do not make errors when typing.

Use Case

- Name: Showing car paths using Oblivious algorithm
- Actor: User
- Preconditions:
 - User is at the homepage
- Description:
 - User starts typing an address in the input field for origin address or destination address
 - System responds with autocomplete functionality showing many options to select.
 - User selects the the right address from the options displayed or finishes typing the full address

User Story #286 - Separate frontend files from server files

As a developer I want to be able to have my angularJs files set in a client folder, separate from the server files, so as the app grows everything remains organized and easy to find.

Acceptance Criteria

1. Cleaner more consistent code

User Story #290 - Implement Car Simulation for google map navigation

As a user I want to see cars moving through the route or routes calculated by google maps based in the trips I entered.

Acceptance Criteria

1. Cars move from source to destination through the route

Use Case

- Name: Showing car paths using google map routes
- Actor: User
- Preconditions:
 - User already entered at least one trip with the three fields required: source, destination and time.
- Description:
 - User selects Add Trip button
 - System responds by displaying the data entered for the trip on the textarea
 - User selects Submit button
 - System responds by displaying car paths or routes determined by google map algorithm and car animations along the paths on the Google map navigation

User Story #292 - Set different times(relative times) for each trip entered

As a user I would like to set different times for each trip created(relative time between trips) so making that way the maps animations more realistic. After a trip time ends, the next trip start running.

Acceptance Criteria

1. User being able to enter integer numbers as inputs for each trip time.
2. Next trip on the list starts just after the previous trip time finish.

User Story #293 - Add button with the functionality of adding trips

As a user I would like to add as many trips as I want to my list of trips before showing the routes and car animations in the maps(use submit button for this functionality). This way there are not animations running in the maps if I have not finished to enter the trips.

Acceptance Criteria

1. Showing a trip with the three fields (source, destination and time) in the textarea every time the user click the button add trip and there is user input for the three fields.
2. Repeat last trip entered data if there is not user input for the three fields and add trip button is clicked.
3. Not showing animations on the maps when the add trip button is clicked. Separate functionality from submit button.

Use Case

- Name: Print trips
- Actor: User
- Preconditions:
 - User is at the homepage
- Description:
 - User enters at least one trip with the three fields required: source, destination and time.
 - User selects Add Trip button
 - System responds by displaying the data entered for the trip on the textarea

User Story #294 - Make the reset button to also clear maps data and animations

As a user I would like to clear all animations showed in the maps when I click the reset button. Now it just clear input entered by the user and the list of trips created.

Acceptance Criteria

1. After clicking reset button return both maps to their initial states (no animations)

Pending User Stories

- [Algo] Adjacency Matrix to Adjacency List #291
- [API] Admin Dashboard #249
- [API] Admin Login #250
- [API] API Key: CRUD #248
- [API] API Keys
- [API] Database #253
- [API] Developer Dashboard #251
- [API] Developer Login #252
- [API] Local API Key #246
- Create Contact Us form #267

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

HARDWARE:

- Windows or MacOS based computer

SOFTWARE:

- NodeJS v6
- NPM v3
- Javascript Libraries
 - ExpressJS
 - AngularJS
 - Google Maps
 - Python-shell
 - Superagent
 - WebPack v2
- Python 3 (dependencies from v1.0)
 - Numpy
 - NetworkX
 - Cython

Sprints Plans

Sprint 1

After discussion, the velocity of the team has yet to be determined, because this is the first sprint.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #237 Rewrite API in NodeJS 4 points
- User Story #238 Initialize Graph 8 points
- User Story #239 geojson formatting 8 points
- User Story #240 Use Graph 4 points
- User Story #243 Python Wrapper 8 points
- User Story #244 Use Oblivious Algorithm 4 points
- User Story #245 Use Dijkstra's Algorithm 4 points
- User Story #264 Add panel to input data 16 points
- User Story #266 Add a reset button 8 points
- User Story #267 Create contact us page 16 points

The team members indicated their willingness to work on the following user stories.

- Matt Thomson
 - User Story #237 Rewrite API in NodeJS 4 points
 - User Story #238 Initialize Graph 8 points
 - User Story #239 geojson formatting 8 points
 - User Story #240 Use Graph 4 points
 - User Story #243 Python Wrapper 8 points
 - User Story #244 Use Oblivious Algorithm 4 points
 - User Story #245 Use Dijkstra's Algorithm 4 points
 - 40 points

Rolando Carralero

- User Story #264 Add panel to input data 16 points
- User Story #266 Add a reset button 8 points
- User Story #267 Create contact us page 16 points
- 40 points

Sprint 2

After discussion, the velocity of the team were estimated to be 64 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- Google Navigation
- Google Maps for our paths
- Integrate Input and API
 - Validate
 - Convert to JSON
- Batch input
- Add moving 'cars' to maps
- Graph data to session storage on server

The team members indicated their willingness to work on the following user stories.

- Matt Thomson
 - #269 Add moving 'cars' to maps
 - #267 Graph data to session storage on server
- Rolando Carralero
 - #269 Google Navigation
 - #287 Google Maps for our paths
 - #Integrate Input and API
 - Validate
 - Convert to
 - #285 Batch input
 - #267 Create Contact us page

Sprint 3

After discussion, the velocity of the team were estimated to be 32 hours. Product owner wants an update in 1 week.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- Google Navigation car animation
- Car animations use server API
- Possible input for both maps

The team members indicated their willingness to work on the following user stories.

- Matt Thomson
 - #280 Car animations use server API
 - #283 Road builder helper program for geojson

- #282 geoJson for roads
 - #283 Road builder
 - #284 Converting algo
- Rolando Carralero
 - #290 Google Navigation car animation
 - #286 Separate frontend files from server files
 - #289 Implement autocomplete feature for user inputs

Sprint 4

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- Set each trip created to work with different initial time(relative time)
- Remove second input field (textarea) and include a new button to add new trips (just submit after adding all trips)
- input for both maps
- Sync car animations between maps
- Build input vertices and correlate lat/lng to addresses via google

The team members indicated their willingness to work on the following user stories.

- Matt Thomson
 - Sync car animations between maps
 - Build input vertices and correlate lat/lng to addresses via google
- Rolando Carralero
 - #292 Set each trip created to work with different initial time (relative time)
 - #293 Remove second input field (textarea) and include a new button to add new trips (just submit after adding all trips)
 - #294 Make reset button to also remove all animation from maps

SYSTEM DESIGN

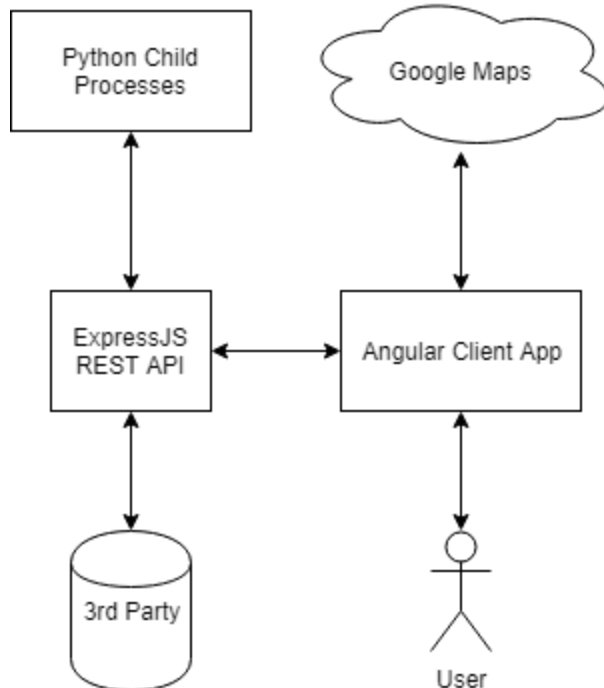
This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

Client-Server

In this application we used client-server architecture. The client make a request to the server and the server return response to that request. This architectural decision allows the server/algorithms code to live in an optimized computing environment.

System and Subsystem Decomposition



1. The frontend (client) is composed as an AngularJS application.
2. The backend (server) is composed of NodeJS, ExpressJS and static files in JSON format for our datasets.

Design Patterns

We used the following design patterns in our project

- **Singleton.**

The singleton patterns allow us to limit the number of instantiations of a class to one. We used “require” to create singletons in our application. This way a module only exist as a single instance, it does not matter how many times the module is required.

SYSTEM VALIDATION

Unit tests are essential to test the functionality within the application. All test cases in this section were used to validate the code for each User Story.

Test case ID: Traffic_simulator_Setup
Description/Summary of Test: <ul style="list-style-type: none">• A user is navigating localhost:8080
Pre-condition: <ul style="list-style-type: none">• Server and Node Js are all running
Expected Results: <ul style="list-style-type: none">• The traffic simulator home page is displayed
Actual Result: <ul style="list-style-type: none">• The traffic simulator home page is displayed
Status : Pass

Test case ID: Enter_data_001
Description/Summary of Test: <ul style="list-style-type: none">• A user is entering data into the three input fields with valid values.

Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page
Expected Results: <ul style="list-style-type: none">• Data is successfully submitted and car paths displayed on the maps
Actual Result: <ul style="list-style-type: none">• Data is successfully submitted and car paths displayed on the maps
Status : Pass

Test case ID: Enter_data_002
Description/Summary of Test: <ul style="list-style-type: none">• A user is entering data into the three input fields with invalid values for either the origin address or the destination address. (User entering no existing addresses)
Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page
Expected Results: <ul style="list-style-type: none">• car paths are not displayed on the maps since invalid origin address or destination was entered
Actual Result: <ul style="list-style-type: none">• car paths are not displayed on the maps
Status : Pass

Test case ID: Enter_data_003
Description/Summary of Test: <ul style="list-style-type: none">• A user is entering data into the three input fields with invalid value for the time
Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page
Expected Results: <ul style="list-style-type: none">• Car paths are not displayed on the maps since invalid value for the time was entered(ex. Enter a string as the time value)
Actual Result: <ul style="list-style-type: none">• car paths are not displayed on the maps
Status : Pass

Test case ID: Add_Trip_001
Description/Summary of Test: <ul style="list-style-type: none">• A user enters valid values into the origin and destination input fields but does not enter a value into the time input field and clicks the add trip button
Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page
Expected Results: <ul style="list-style-type: none">• Trip created and printed on the textarea with with a time value of 0 and the addresses entered by the user

Actual Result: <ul style="list-style-type: none">• Trip created and printed on the textarea with a time value of 0 and the addresses entered by the user
Status : Pass

Test case ID: Add_Trip_002
Description/Summary of Test: <ul style="list-style-type: none">• A user enters valid values into the destination address and time input fields but does not enter a value into the origin address input field and clicks the add trip button
Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page
Expected Results: <ul style="list-style-type: none">• Trip created and printed on the textarea with the current location as a value for the origin address and the other two values correctly entered by the user
Actual Result: <ul style="list-style-type: none">• Trip created and printed on the textarea with the current location as a value for the origin address and the other two values correctly entered by the user
Status : Pass

Test case ID: Add_Trip_003
Description/Summary of Test: <ul style="list-style-type: none">• A user enters valid values into the origin address and time input fields but does not enter a value into the destination address input field and clicks the add trip button

Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page
Expected Results: <ul style="list-style-type: none">• Trip created and printed on the textarea with the current location as a value for the destination address and the other two values correctly entered by the user
Actual Result: <ul style="list-style-type: none">• Trip created and printed on the textarea with the current location as a value for the destination address and the other two values correctly entered by the user
Status : Pass

Test case ID: Add_Trip_004
Description/Summary of Test: <ul style="list-style-type: none">• A user does not enter values into any of the three input fields and clicks the add trip button
Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page
Expected Results: <ul style="list-style-type: none">• Trip created and printed on the textarea with the current location as a value for the origin address and destination address and 0 as a value for the time.
Actual Result: <ul style="list-style-type: none">• Trip created and printed on the textarea with the current location as a value for the origin address and destination address and 0 as a value for the time.
Status : Pass

Test case ID: Add_Trip_005
Description/Summary of Test: <ul style="list-style-type: none">• A user enters valid values the three input fields and clicks the add trip button
Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page
Expected Results: <ul style="list-style-type: none">• Trip created and printed on the textarea with the three valid values entered by the user
Actual Result: <ul style="list-style-type: none">• Trip created and printed on the textarea with the three valid values entered by the user
Status : Pass

Test case ID: Select_Algorithm_001
Description/Summary of Test: <ul style="list-style-type: none">• User creates a trip or some trips filling out the required fields. User selects Dijkstra algorithm to determine car paths. Then user selects Submit button
Pre-condition: <ul style="list-style-type: none">• User is at Traffic Simulator home page• User created at least one trip

<p>Expected Results:</p> <ul style="list-style-type: none">• Car paths are determined by Dijkstra algorithm and displayed on the oblivious map navigation.
<p>Actual Result:</p> <ul style="list-style-type: none">• Car paths are determined by Dijkstra algorithm and displayed on the oblivious map navigation.
<p>Status : Pass</p>

<p>Test case ID: Clear_data_001</p>
<p>Description/Summary of Test:</p> <ul style="list-style-type: none">• A user is clicking the reset button to remove all data entered and trips created
<p>Pre-condition:</p> <ul style="list-style-type: none">• User already created a trip or entered data into any of the three input fields
<p>Expected Results:</p> <ul style="list-style-type: none">• All data is erased from input fields and the textarea
<p>Actual Result:</p> <ul style="list-style-type: none">• All data is erased from input fields and the textarea
<p>Status : Pass</p>

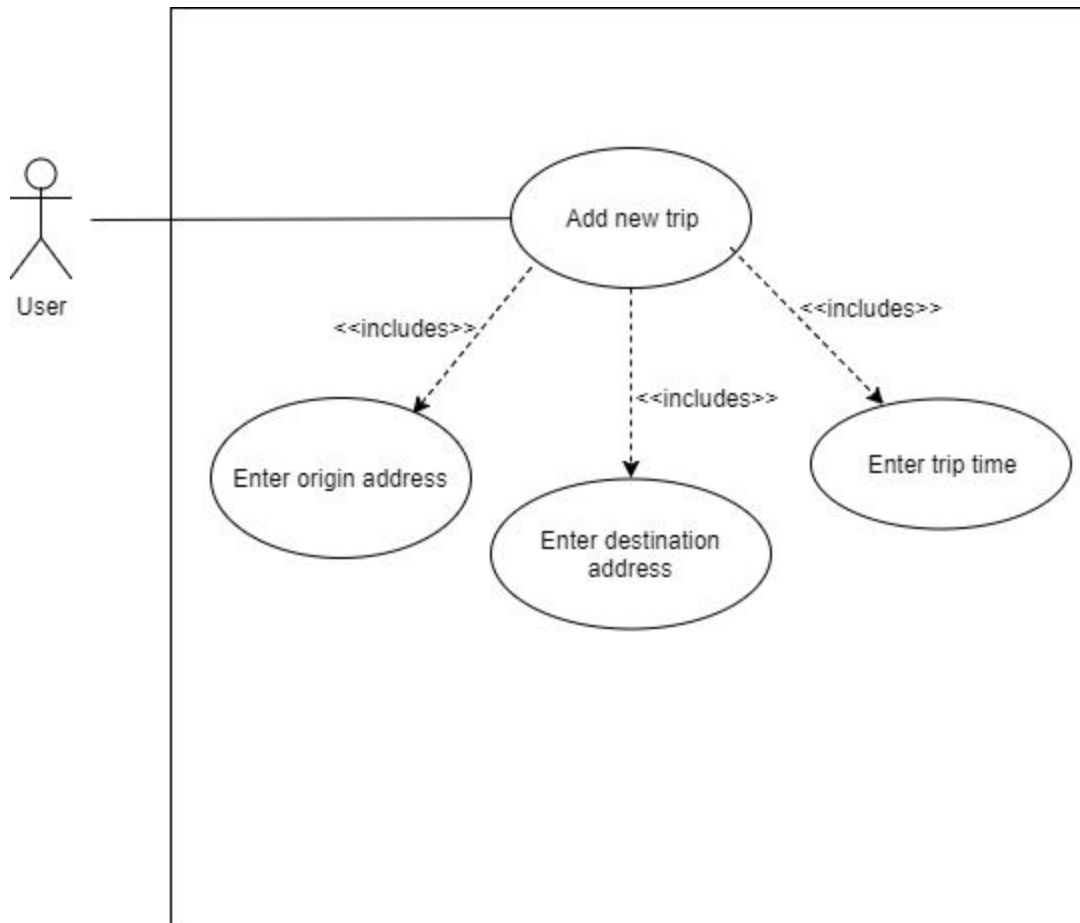
APPENDIX**Appendix A - UML Diagrams**

Figure 1: Use case diagram, part of user stories #264 and #293-Enter data and add Trip

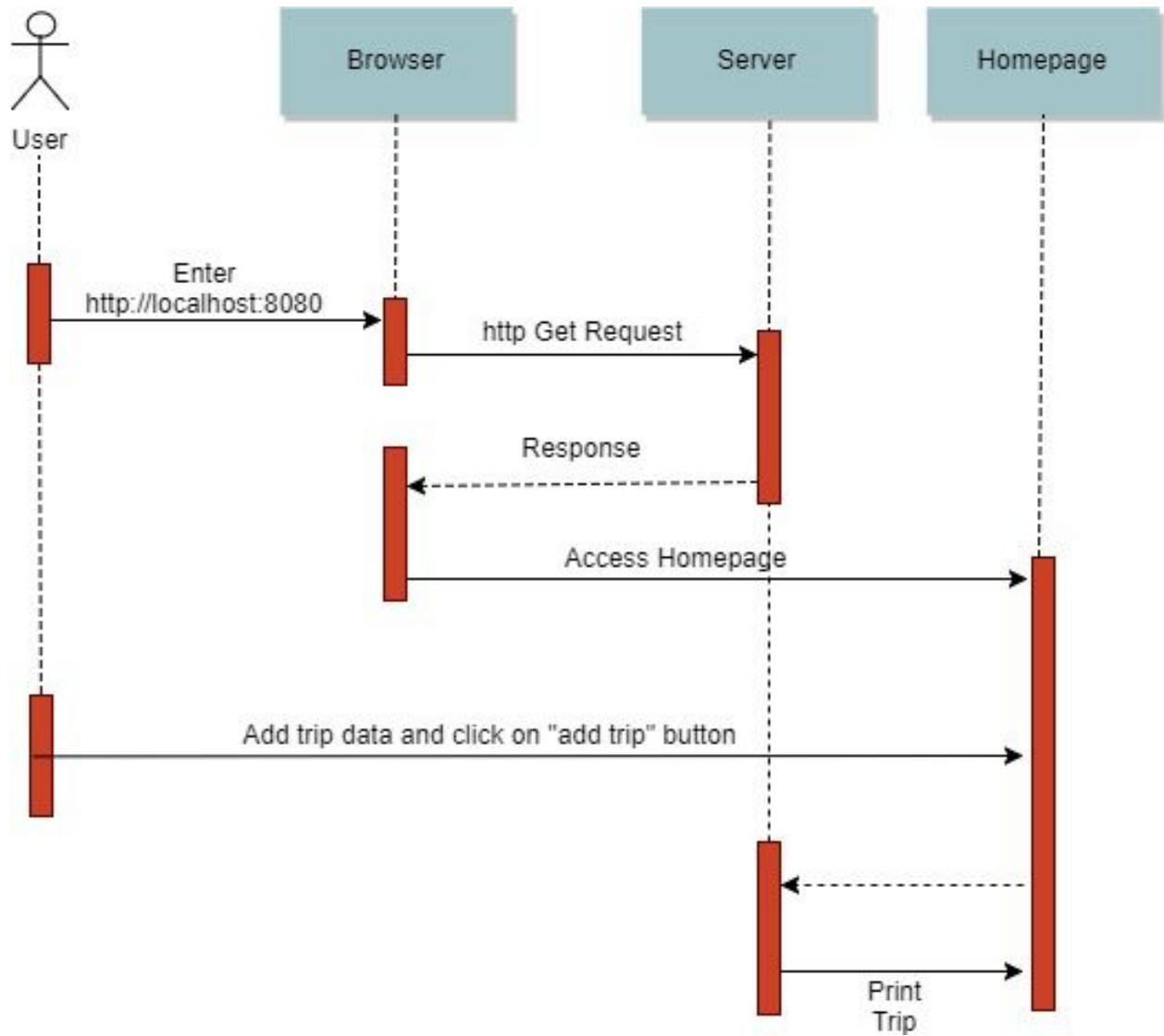


Figure 2: Sequence diagram, part of user stories #264 and #293-Enter data and add Trip

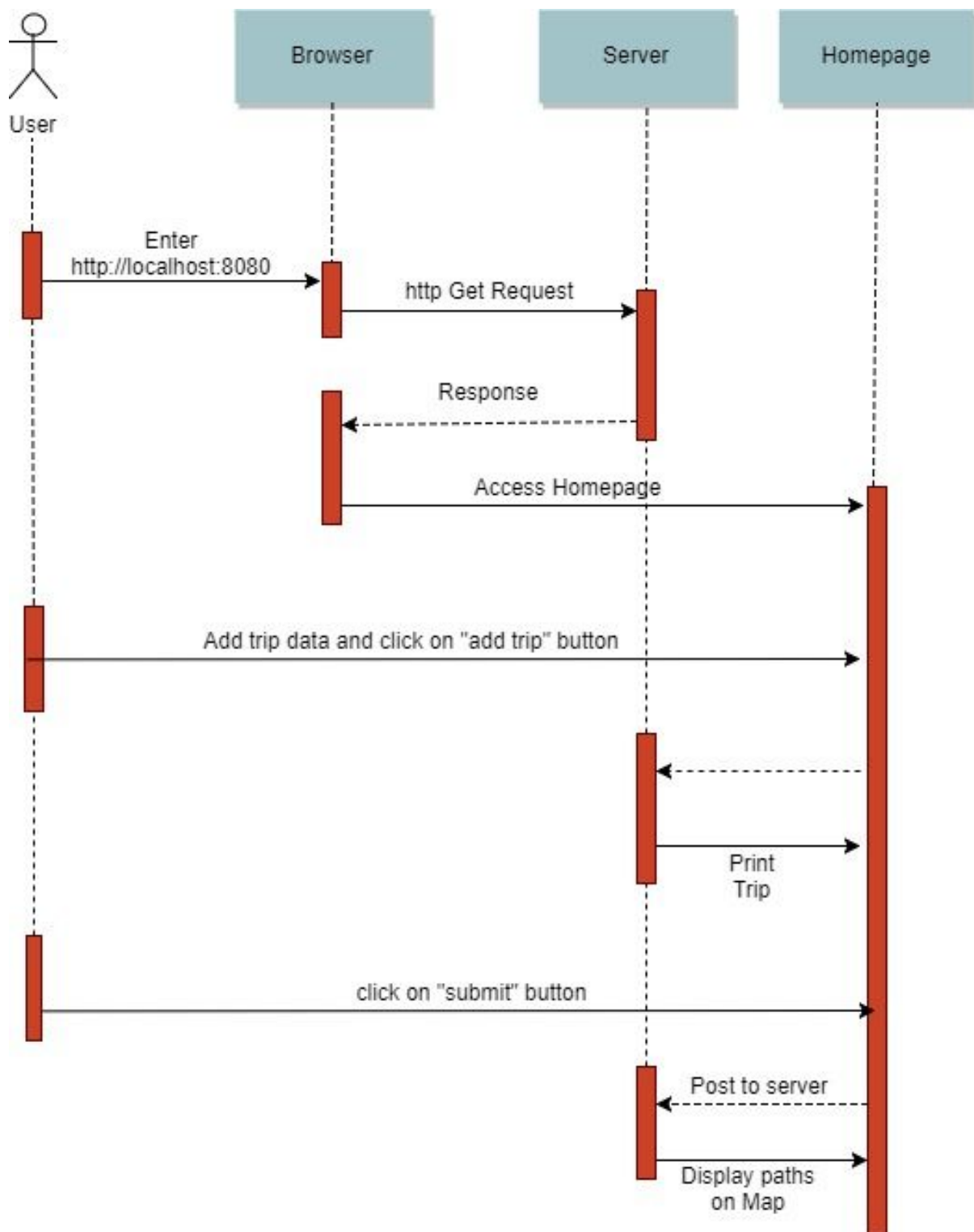


Figure 3: Sequence diagram, part of user stories #278 and #287-Create submit button and display routes on google map.

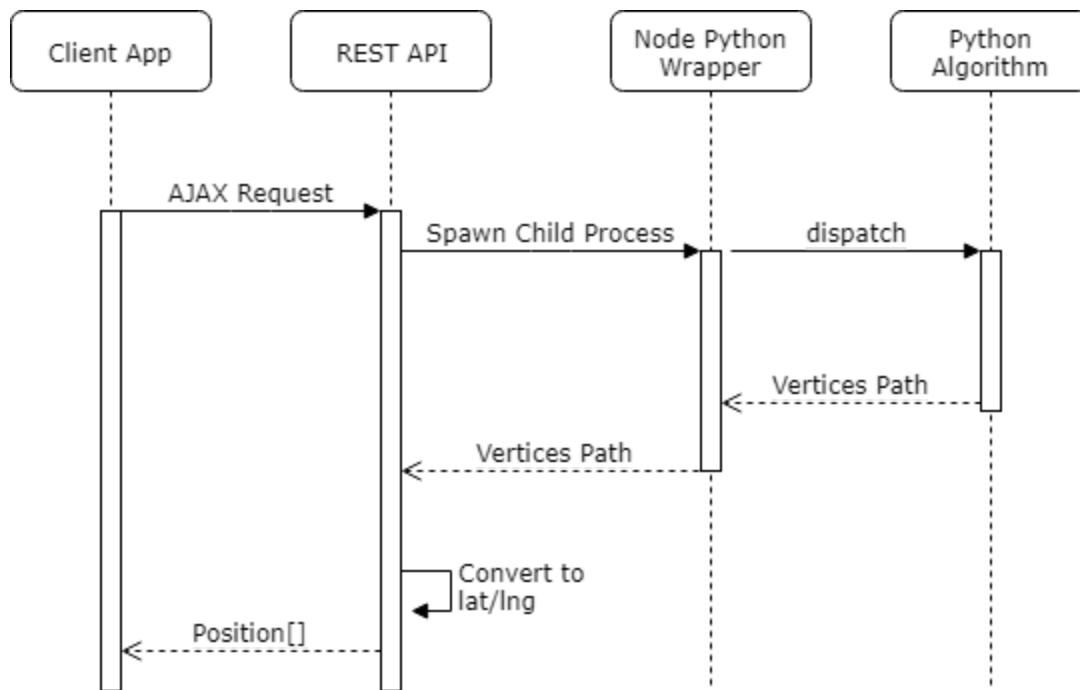


Figure 4: Sequence diagram for algorithm access.

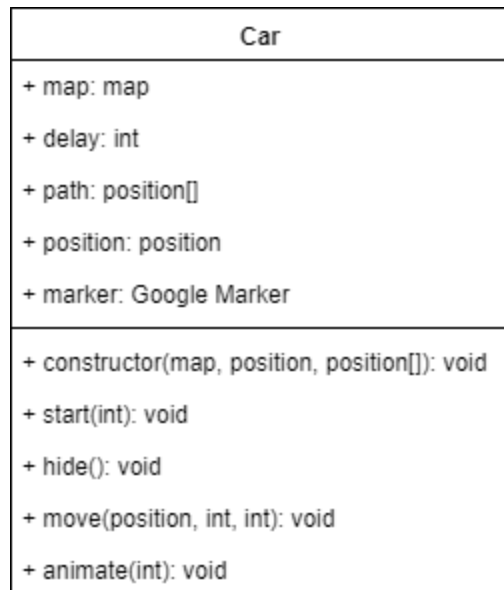
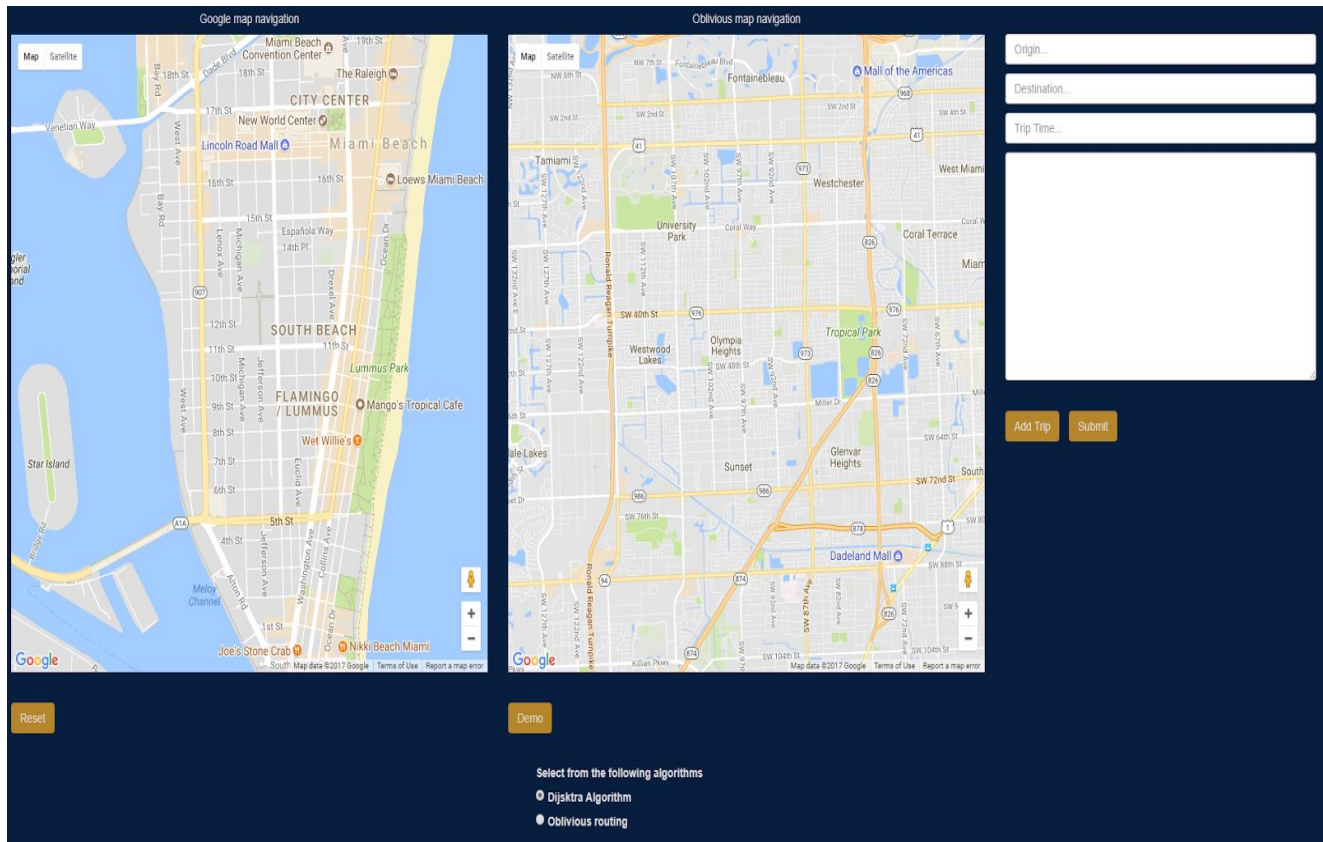


Figure 5: Class diagram for car animations

Appendix B - User Interface Design

Figure #1 Homepage



Appendix C - Sprint Reports

Sprint Review

Sprint 1

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

- User Story #237 Rewrite API in NodeJS 4pts
- User Story #238 Initialize Graph 8pts
- User Story #239 geojson formatting 8pts
- User Story #240 Use Graph 4pts
- User Story #243 Python Wrapper 8pts
- User Story #245 Use Dijkstra's Algorithm 4pts
- User Story #264 Add panel to input data 16 pts
- User Story #266 Add a reset button 8 pts
- User Story #277 Setting the app with AngularJs 8pts

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- User Story #267 Create contact us page 16 pts
 - Because it isn't a priority
- User Story #244 Use Oblivious Algorithm 4pts
 - Because it doesn't work

Sprint 2

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

- #269 Google Navigation
- #287 Google Maps for our paths
- #285 Batch input
- #279 Add moving 'cars' to maps
- #276 Graph data to session storage on server

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- Integrate Input and API

Sprint 3

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

- Google Navigation car animation
- Car animations use server API

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- Possible input for both maps
- Synchronize both maps
- Bound direction to built geojson.

Sprint Retrospective

Sprint 1

What went wrong?

- Did we do a good job estimating our team's velocity?
 - I think we did a good job estimating our team's velocity. There was a story that we were off on, but overall we were fairly accurate.
- Did we do a good job estimating the points (time required) for each user story?
 - Some of user stories were overestimated and underestimated based on the unknown complexity of building off of the previous team's code.
- Did each team member work as scheduled?
 - Yes

What went right?

- Our daily scrum times and meetings were efficient and well done.

How to address the issues in the next sprint?

- How to improve the process?
 - Our process was good this sprint. Meetings were on time and efficient.
- How to improve the product?
 - Completely move to Google Maps. Display "cars" on Google Maps from Google Navigation and our API.

Sprint 2

What went wrong?

- Did we do a good job estimating our team's velocity?
 - Not as good as the first sprint, because there were a lot of unforeseen complications.
- Did we do a good job estimating the points (time required) for each user story?
 - No, for the same reason as the velocity issue.
- Did each team member work as scheduled?
 - Yes

What went right?

- Our daily scrum times and meetings were efficient and well done.

How to address the issues in the next sprint?

- How to improve the process?
 - Our process was good this sprint. Meetings were on time and efficient.
- How to improve the product?
 - Animating Google's Navigation and integrating our server api algorithms.

Sprint 3

What went wrong?

- Did we do a good job estimating our team's velocity?
 - I think we did a good job estimating our team's velocity.
- Did we do a good job estimating the points (time required) for each user story?
 - Our time estimations were a bit different, since the product owner wished to have something ready mid sprint, but overall it went well.
- Did each team member work as scheduled?
 - Yes

What went right?

- Our daily scrum times and meetings were efficient and well done.

How to address the issues in the next sprint?

- How to improve the process?
 - Our process was good this sprint. Meetings were on time and efficient.
- How to improve the product?
 - Use relative time as input for every trip, Set each trip to have different initial time and animate it with only one car.
 - Bound the possible inputs to the geojson that was generated last sprint
 - Synchronize maps

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings, Wishlist Document and other documents

Installation/Maintenance

1. Install NodeJS v6+ from <https://nodejs.org/en/>
2. Clone <https://github.com/FIU-SCIS-Senior-Projects/Traffic-Simulator-2.0> repository
3. Navigate to the Code folder in terminal or cmd.
4. Run `npm install` to install all dependencies.
5. Run `npm run build` to compile the app.
6. Run `npm start` to start the web server.
7. Open the browser and go to <http://localhost:8080>

Shortcomings

We were working on version 2.0 of the Traffic Simulator project. During this semester we were able to implement new features and improve the functionality of the system. However there are some features we were not able to work on.

- Algorithms take a long time to initialize.
- Unable to improve algorithms functionality. Some algorithms are not fully implemented

Wishlist

- Implement a new design for the car animations on the maps. (Now using google maps marker)
- Add a contact us page