Harrison Mitchell

SDD Major Work

2017

Table of Contents

Defining and Understanding

Data dictionary

Data name	Data type	Format	Display size	Description	Example	Range
ShowVehicles	String Array	XXX,XXX...	N/A	Which vehicles should be shown on the map, edited via settings	[“SydneyTrains”, “NSWTrains”]	[] to [“SydneyTrains”, “NSWTrains”, “Buses”, “LightRail”, “Ferries”]
URLbase	String	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	35	The base URL for API calls	'https://api.transport.nsw.gov.au/v1'	N/A
URLposSydneyTrains	String	XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	29	The location of the API off base	'/gtfs/vehiclepos/sydneytrains'	N/A
URLposNSWTrains	String	XXXXXXXXXXXXXXXXXXXXXXXXXX	26	The location of the API off base	'/gtfs/vehiclepos/nswtrains’	N/A
URLposLightRail	String	XXXXXXXXXXXXXXXXXXXXXXXXXX	26	The location of the API off base	'/gtfs/vehiclepos/lightrail’	N/A
URLposFerries	String	XXXXXXXXXXXXXXXXXXXXXXXX	24	The location of the API off base	'/gtfs/vehiclepos/ferries'	N/A
URLposBuses	String	XXXXXXXXXXXXXXXXXXXXXX	22	The location of the API off base	'/gtfs/vehiclepos/buses’	N/A
APIkey	String	“apikey l7”	43	The key	'apikey	N/A

		+ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX		needed to access the TransportData API	l7xxdd534960a59c41b2b645de58f795a81b'	
APIkey	String	“key=” + XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	44	The key needed to access GoogleMaps API	‘key=AIzaSyCD1-Q0ATIDu6QmY6UsOfjOSIY2IyfYoyU’	N/A
CallWait	Int	NNN - NNNNNN	3 - 6	How often should the API be called in ms?	1000	100-100000
Zoom	int	N - NN	1 - 2	How zoomed should the Google Map be?	10	3-20
Center	Float Array	[NN.NN, NN.NN]	8 - 10	Where should the map be centred?	[-33.85, 151.2]	[-180, -180] - [180, 180]
disableDefaultUI	Boolean	True/False	4 - 5	Should GoogleMaps UI be simplified?	True	True/False
VechicleImage	String	“img/” + XXX + “dot.png”	15 - 18	Location to image used for displaying vehicle	'img/yellowdot.png'	Yellow, black, green, red, blue
MarkerPosition	Float Array	[NN.NN, NN.NN]	8 - 10	Where should the currently focused vehicle's	[-33.85, 151.25]	[-180, -180] - [180, 180]

				icon be drawn		
MarkerMap	Map	XXXXXXX	7	Which map should the marker be drawn to?	Default	N/A
DotFillColor	String	“#” + “***”	4	What color should the marker be?	'#00F'	#000 - #fff
DotFillOpacity	Float	N.N	1 - 2	How transparent should the marker be?	1	0-1
DotRotation	Float	N - NNN	1 - 3	How rotated should the marker be?	41.6	0-360
DotScale	Float	N.N	1 - 2	How big should the marker be?	1	0-1

Outline of requirements and specifications

Developer’s perspective

- Write efficient algorithms (not wasting too much processing power)
- Ensure documentation is helpful and specific
- Ensure UI scales well to user’s display resolution
- Ensure the correct data types are used
- Ensure errors are caught and users are served a more understandable error
- Consider and select the most applicable software development approach
- Make sure quality assurance is met
- Ensure documentation is kept up to date
- Ensure variables are reflective of the data dictionary
- Ensure variables are named well

User’s perspective

- UI will use appropriate and relative text for buttons
- Error messages will be specific and use neutral language
- Tooltip messages will be succinct and helpful
- All messages will be gender neutral
- UI will scale well to the users’ display resolution
- Site will load in < 2 seconds
- Vehicles will update every 20 seconds
- Buttons will always perform a noticeable action
- Clicking on a vehicle will have its data displayed within 2 seconds
- Removing a vehicle type from the map should take < .5 seconds

IPO chart

API response to boolean

Input	Processing	Output
Response	Check if response is 400	IsAuthenticated boolean

Authenticate with GoogleMapsAPI

Input	Processing	Output
API key	Send a request to the Google Maps API server Wait for a response	Response

Authenticate with NSWTransportAPI

Input	Processing	Output
API key	Send a request to the NSWTransport API server Wait for a response	Response

Update vehicle positions

Input	Processing	Output

Array of vehicle positions	Remove all previous vehicles from map Decipher what vechicle type the current vehicle is and the marker color for it Get the latitude and longitude from the array of vehicles and parse it into a point on the map Send the Google Maps API the marker color and position	Display vehicles

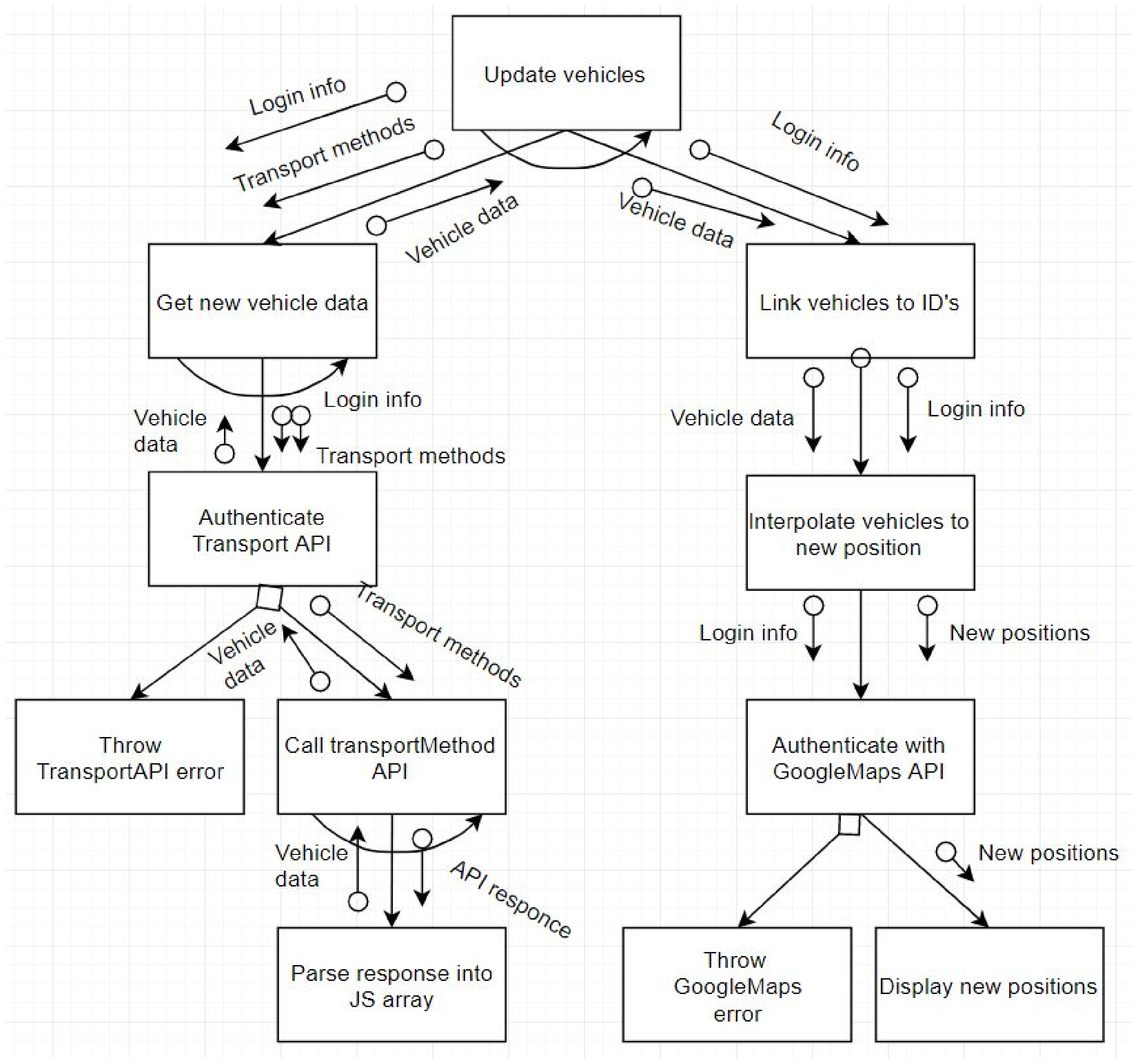
Inquire about new vehicle positions

Input	Processing	Output
IsAuthenticated boolean URL for vehicle type	If we are authenticated ask the API to return vehicle data for the specific vehicle	Array of vehicle data

Interpolate vehicle positions

Input	Processing	Output
Array of old vehicle positions Array of new vehicle positions from API	Calculate several points between the two points and send the update vehicle API each subsequent step 100ms apart	Location (10 times, once every .1 second)

Structure Chart



Planning and Designing

Research into program creation

Standard Algorithms

I'll need to manipulate strings in order to extract relevant data from the API response. I'll need to be able to open and close relative files when I need to find stop locations for a particular route in the shapes file. Finally, I'll be using binary search to find the relevant ID for a shape in the shapes file. Binary search almost always allows the data to be found faster, which makes for a better product.

Comparison to Similar Products

Appearance is very similar to Google Maps, this allows the user to become more comfortable faster if they have used Google Maps in the past. Functionality wise, Google Maps does not show live positions of vehicles nor are shapes of routes shown.
Tripview is an app endorsed by NSW transport as it shows your bus/train etc… as it goes along its route and shows you your vehicle on a map in real time, however the position update is largely infrequent. My program updates positions 3 times faster as well as showing every vehicle whilst Tripview shows your singular, specific vehicle. Route shapes are also not shown on the app whilst stop locations are shown, similar to my program. Tripview is a mobile app whilst mine is a desktop app, inherently meaning my program will look different proportion wise. Tripview uses a different map provider, hence the map will look different between the two programs.

Platform considerations

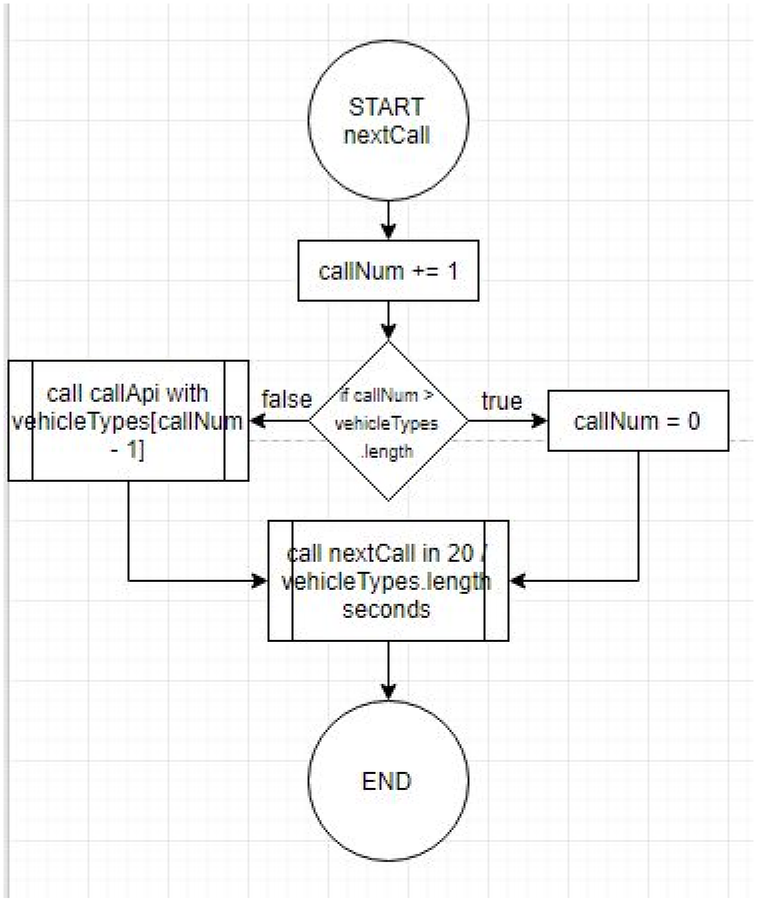
I have chosen to make a browser-based program due to it's cross platform nature. My website will be opened with browsers on desktop computers. The languages (JS/Node, HTML/CSS) I'm using to run this program also work best with browser-based programs as they were developed for it. This project isn't too large and working with HTML/CSS allows changes to be seen faster than if I were to make a GUI for a desktop application. A desktop launch is desirable since there are can be many vehicles at one time interpolating between positions which can be tough on the GPU/CPU. This load might not be handled well on a phone/app..

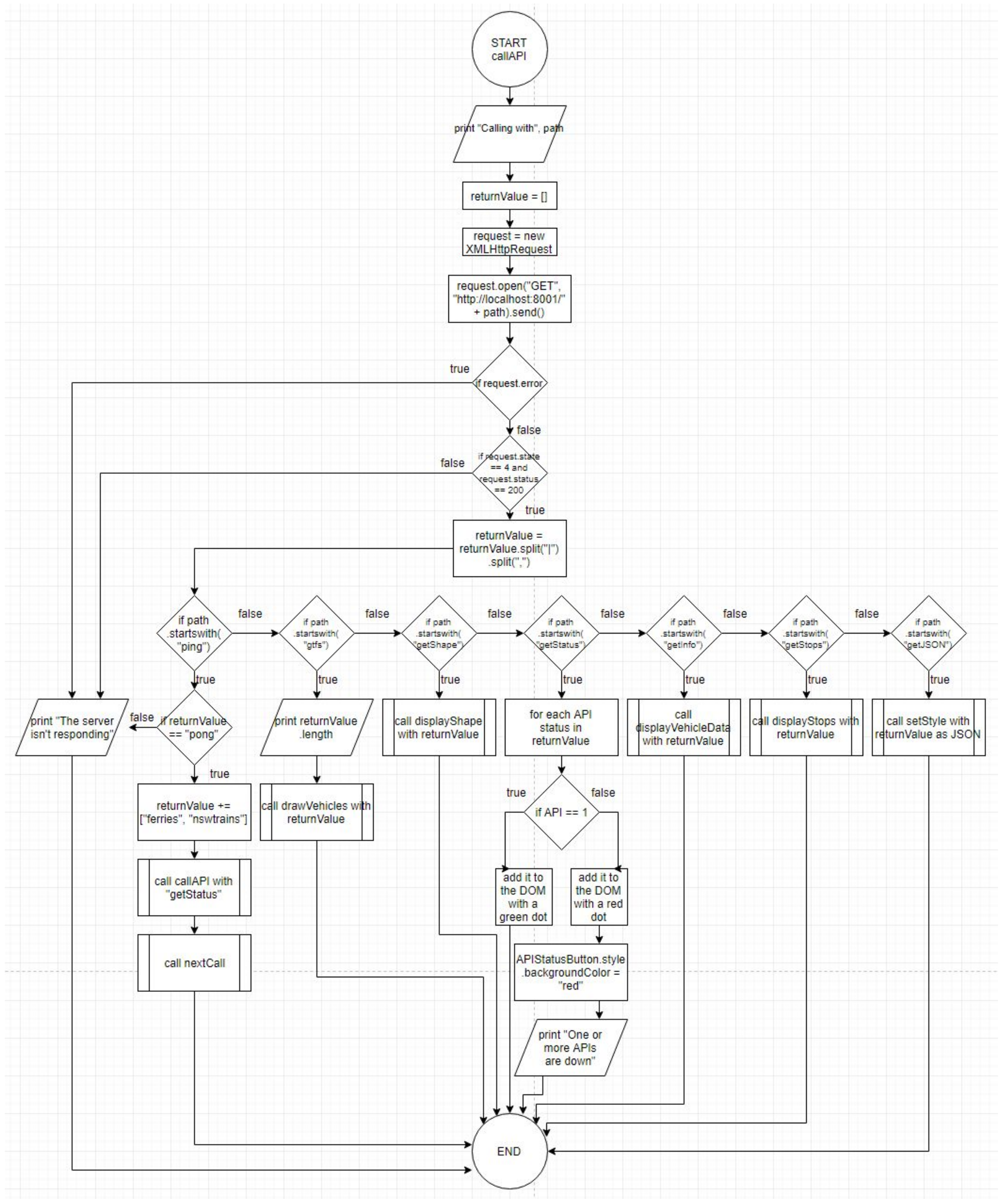
Also preventing this application from running on a phone is the amount of options within the application. This could clutter a phone/screen menu very quickly and would leave little room for the map to be displayed. The heavy data transfers that take place might be an issue with phones as they generally have lower processing power.

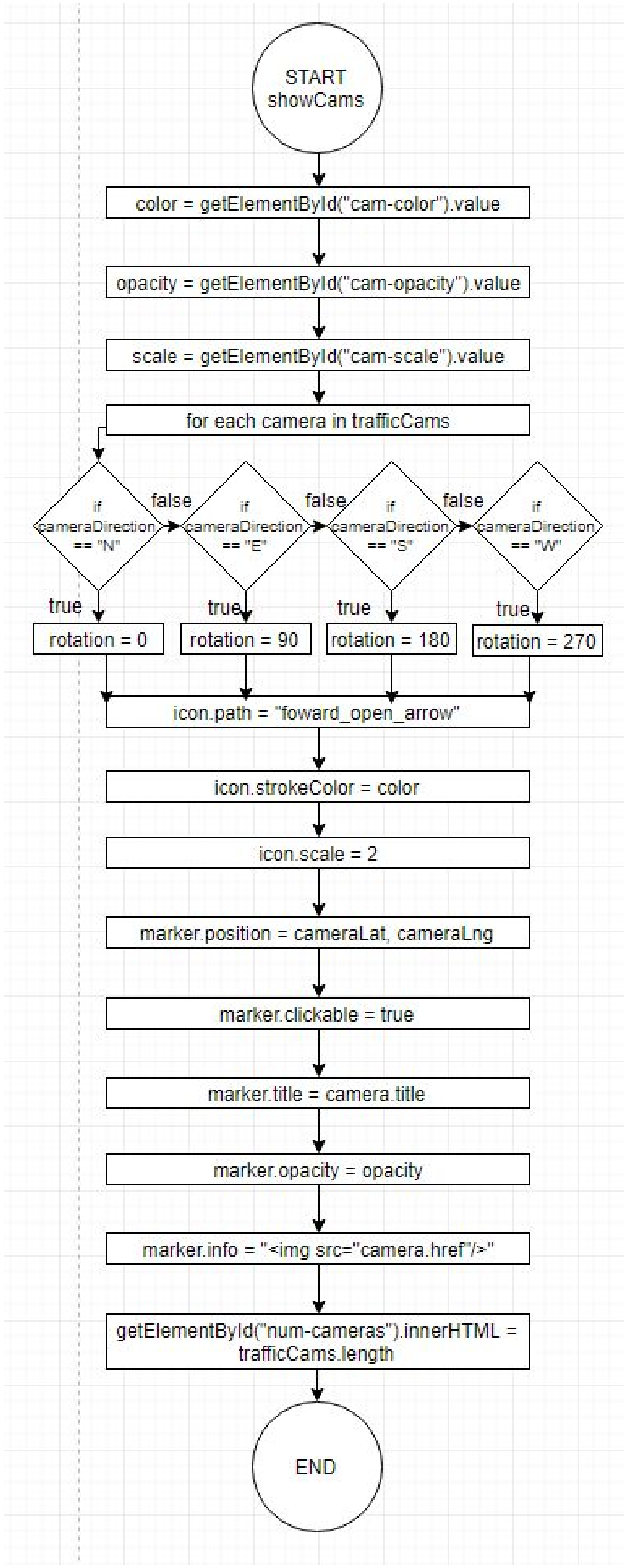
Tripview chose to serve on mobile phones as their service is a more informative, on- demand application. My program is more of an overview and less of an on demand information platform. Testing would takes longer since IOS and Android apps need to be compiled since they use Objective C and Java respectively.

Google Maps is more of an overview too, but more for trip planning and not a tool for niche information. Seeing as there are large amounts of information available, Google has chosen to serve on desktops whilst compromising as they make a mobile app with slightly less available information.

Algorithm flowcharts


START nextCall
callNum += 1
if callNum > vehicleTypes .length
false
true
call callApi with vehicleTypes[callNum - 1]
callNum = 0
call nextCall in 20 / vehicleTypes.length seconds
END


START callAPI
print "Calling with", path
returnValue = []
request = new XMLHttpRequest
request.open("GET", "http://localhost:8001/" + path).send()
if request.error
true
false
if request.state == 4 and request.status == 200
false
true
returnValue = returnValue.split("|").split(",")
if path .startswith("ping")
if path .startswith("gtfs")
if path .startswith("getShape")
if path .startswith("getStatus")
if path .startswith("getInfo")
if path .startswith("getStops")
if path .startswith("getJSON")
false
true
print "The server isn't responding"
if returnValue == "pong"
print returnValue .length
call displayShape with returnValue
for each API status in returnValue
call displayVehicleData with returnValue
call displayStops with returnValue
call setStyle with returnValue as JSON
returnValue += ["ferries", "nswtrains"]
call drawVehicles with returnValue
if API == 1
call callAPI with "getStatus"
add it to the DOM with a green dot
add it to the DOM with a red dot
call nextCall
APIStatusButton.style .backgroundColor = "red"
print "One or more APIs are down"
END


START
showCams
color = getElementById("cam-color").value
opacity = getElementById("cam-opacity").value
scale = getElementById("cam-scale").value
for each camera in trafficCams
if cameraDirection == "N"
false
if cameraDirection == "E"
false
if cameraDirection == "S"
false
if cameraDirection == "W"
true
rotation = 0
true
rotation = 90
true
rotation = 180
true
rotation = 270
icon.path = "foward_open_arrow"
icon.strokeColor = color
icon.scale = 2
marker.position = cameraLat, cameraLng
marker.clickable = true
marker.title = camera.title
marker.opacity = opacity
marker.info = "<img src="camera.href"/>"
getElementById("num-cameras").innerHTML = trafficCams.length
END

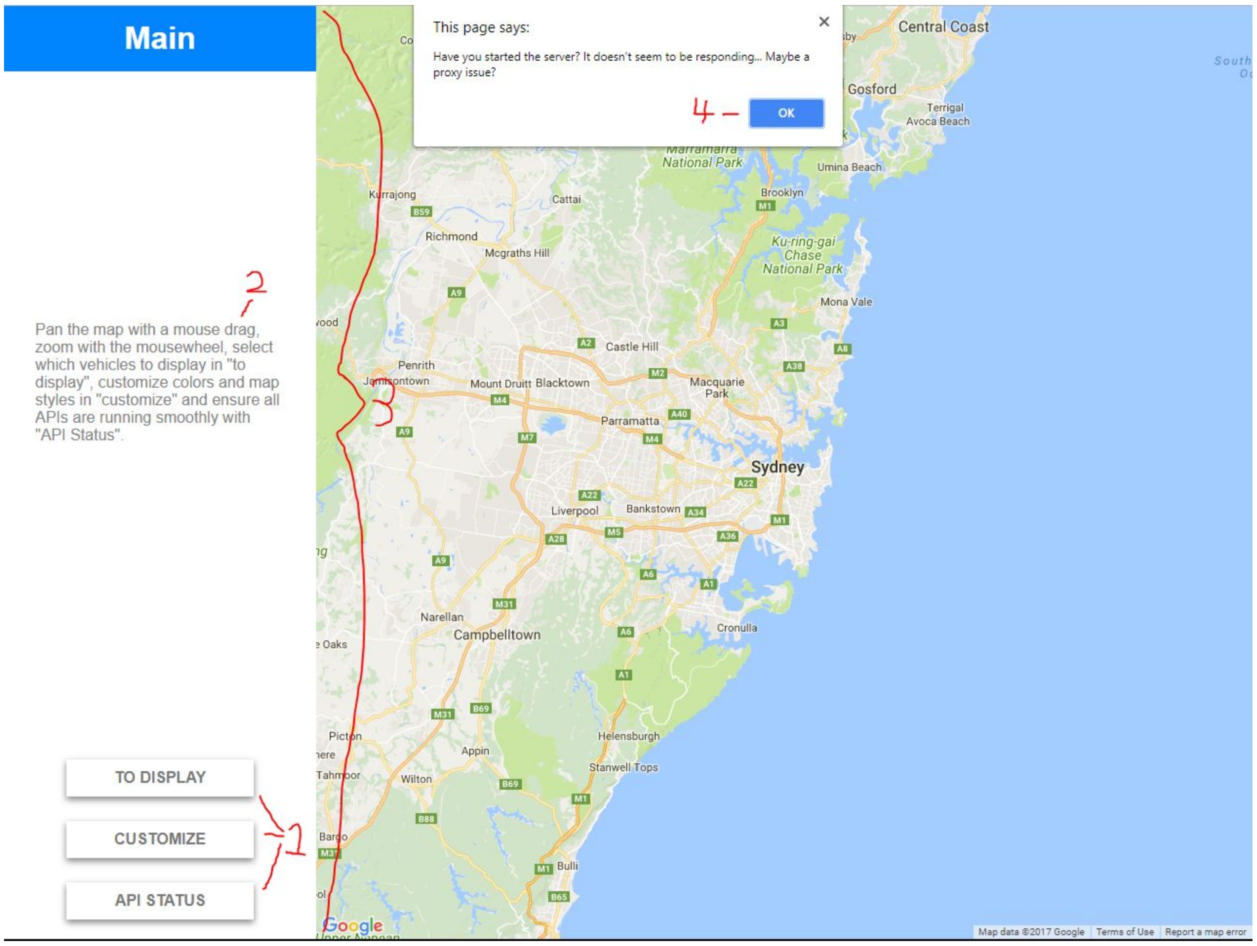
Pseudocode

```
START nextCall (callNum, vehicleTypes)
    add one to callNum
    IF callNum is greater than the length of vehicleTypes
        set callNum to 0
    ELSE
        callApi(vehicleTypes(callNum minus one))
    run nextCall in 20 secs divided by the length of vehicleTypes
END

START callAPI (path)
    print "Calling with" path
    set returnValue to an empty array
    set request to a new XMLHttpRequest
    open a GET request at "http://localhost:8001/" + path
    send request
    If request has an error
        print "The server isn't working"
    when the request changes states
    IF the state is 4 and the status is 200
        returnValue = responseText split at the pipes
        returnValue = each splice sliced again at the commas
        IF path starts with "ping"
            IF returnValue equals "pong"
                add "nswtrains" and "ferries" to vehicleTypes
                run callAPI with "getStatus"
                run nextCall
            ELSE
                print "Server isn't responding"
        IF path starts with "gtfs"
            print returnValue length
            run drawVehicles with each vehicle in returnValue
        IF path starts with "getShape"
            run displayShape with returnValue
        IF path starts with "getStatus"
            FOR EACH API status
                IF API is up
                    add it to DOM with green dot
                ELSE
                    add it to DOM with red dot
                    make API Status button red in DOM
                    print "One or more APIs are down"
        IF path starts with "getInfo"
            run displayVehicleData with returnValue
        IF path starts with "getStops"
            run displayStops with returnValue
        IF path starts with "getJSON"
            run setStyle with returnValue as JSON
END

START showCams (trafficCams)
    color = get value of element by id of "cams-color"
    opacity = get value of element by id of "cams-opacity"
    scale = get value of element by id of "cams-scale"
    FOR EACH camera in trafficCams
        if camera direction equals north
            rotation equals 0
        if camera direction equals east
            rotation equals 90
        if camera direction equals south
            rotation equals 180
        if camera direction equals west
            rotation equals 270
        icon.path = forward_open_arrow
        icon.strokeColor = color
        icon.scale = 2
        marker.position = cameraLat, cameraLng
        marker.clickable = true
        marker.title = camera.title
        marker.opacity = opacity
        marker.info = HTML image with SRC of camera.href
        set value of element by id of "num-cameras" to length of trafficCams
END
```

User interface design


This page says:

Have you started the server? It doesn't seem to be responding... Maybe a proxy issue?

Pan the map with a mouse drag, zoom with the mousewheel, select which vehicles to display in "to display", customize colors and map styles in "customize" and ensure all APIs are running smoothly with "API Status".

The map is large so detail can be seen easier helping ergonomics and accessibility. It's also large as it's the main focus of the program and it's what my audience (moderately tech literate individuals of all ages that have an interest in the transport of NSW) are the most interested in. Right there on the main page is online help explaining what there is to the application which isn't much as it's mostly backend.

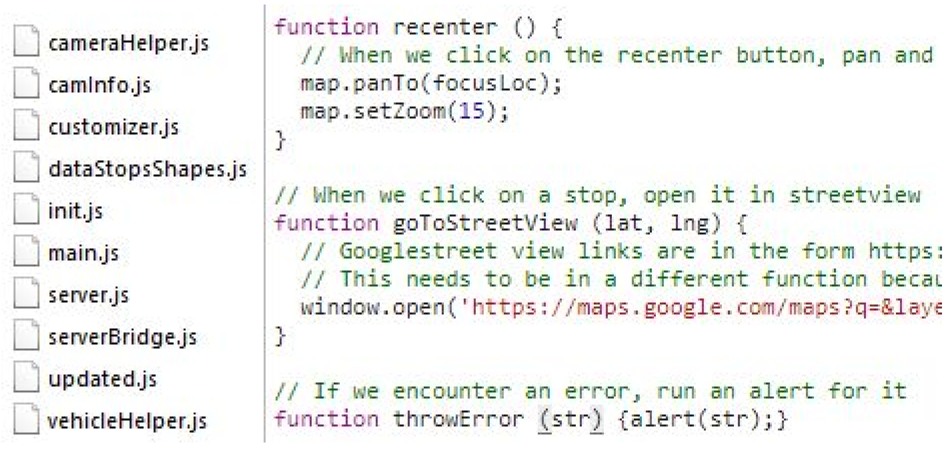
Social and ethical issues

Ergonomically, the buttons are large [1] removing precision needed with the mouse and text is contrasted [2] to aid readability. Most mouse clicks occur on the left [3] saving the user mouse movement. Also reducing this movement is part-keyboard navigation. Prompts and input elements can be navigated with the keyboard [4].

Implementing

Creating and Organisation of Code

My code is split up into multiple files to find modules/code easier. Within each file, the code is then further split up into functions that can be reused as seen by, for example, the "throwError" function.



```
cameraHelper.js
camInfo.js
customizer.js
dataStopsShapes.js
init.js
main.js
server.js
serverBridge.js
updated.js
vehicleHelper.js
```

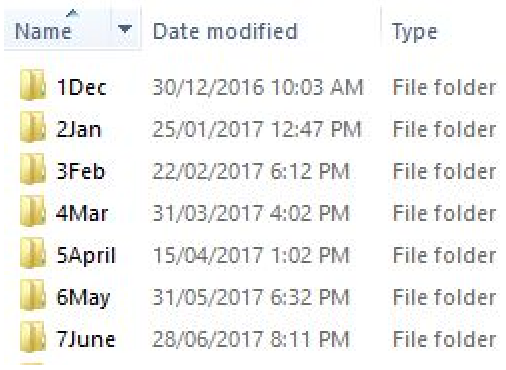
```
function recenter () {
  // When we click on the recenter button, pan and
  map.panTo(focusLoc);
  map.setZoom(15);
}

// When we click on a stop, open it in streetview
function goToStreetView (lat, lng) {
  // Googlestreet view links are in the form https:
  // This needs to be in a different function becau
  window.open('https://maps.google.com/maps?q=&laye
}

// If we encounter an error, run an alert for it
function throwError (str) {alert(str);}
```

Version Control

With the project being spread out over a considerable amount of time, a monthly version backup was appropriate. It wasn't too often that it took too much time and it wasn't too far apart were I could lose months of work if the current version were to fail.

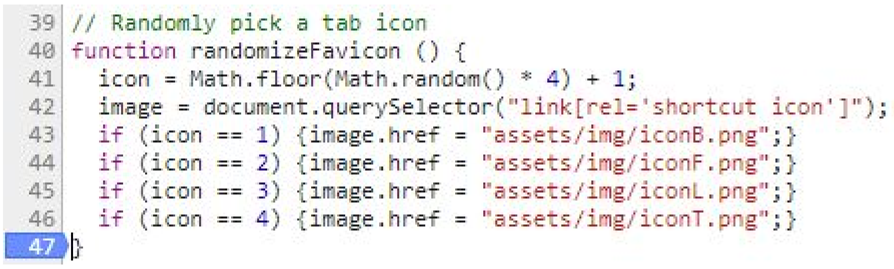


Name	Date modified	Type
1Dec	30/12/2016 10:03 AM	File folder
2Jan	25/01/2017 12:47 PM	File folder
3Feb	22/02/2017 6:12 PM	File folder
4Mar	31/03/2017 4:02 PM	File folder
5April	15/04/2017 1:02 PM	File folder
6May	31/05/2017 6:32 PM	File folder
7June	28/06/2017 8:11 PM	File folder

Testing of Code

Breakpoints

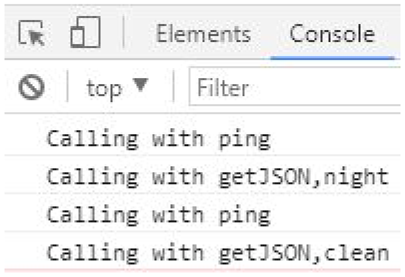
I added my randomize favicon function into the code rather late and I wanted to test just that function. So to prevent wasted resources by loading and running the rest of my code, I set up a breakpoint after the randomize favicon function so only it would run. This was to stop the code immediately after the function had been run so its output could be examined.



```
// Randomly pick a tab icon
function randomizeFavicon () {
  icon = Math.floor(Math.random() * 4) + 1;
  image = document.querySelector("link[rel='shortcut icon']");
  if (icon == 1) {image.href = "assets/img/iconB.png";}
  if (icon == 2) {image.href = "assets/img/iconF.png";}
  if (icon == 3) {image.href = "assets/img/iconL.png";}
  if (icon == 4) {image.href = "assets/img/iconT.png";}
}
```

Debugging Output Statements

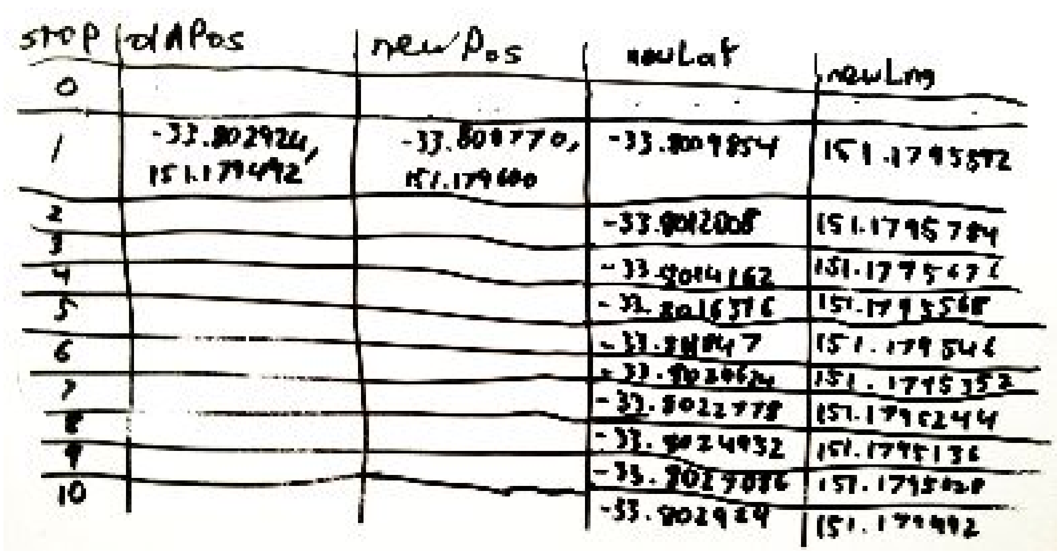
To ensure my server wasn't getting called continuously and was getting called with the right values, I set up a debug statement that would fire every time the server was called to see if the server was getting called with the right value.



```
Calling with ping
Calling with getJSON,night
Calling with ping
Calling with getJSON,clean
```

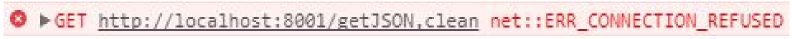
Desk Checks

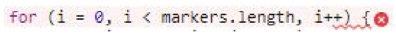
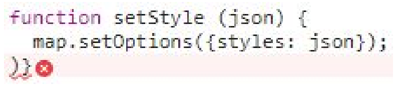
To ensure that my interpolation algorithm was outputting the correct steps, I ran a desk check and did the math manually. I picked two possible locations and ran through the function once. The desk check was to confirmed that the values returned by the function was correct, which it was.



stop	oldPos	newPos	newLat	newLng
0				
1	-33.802924, 151.179492	-33.801770, 151.179600	-33.8009854	151.1795572
2			-33.8012008	151.1795784
3			-33.8014162	151.1775476
4			-33.8016376	151.1795568
5			-33.8018[illegible]7	151.179546
6			-33.8020624	151.1795352
7			-33.8022778	151.1792244
8			-33.8024932	151.1795136
9			-33.8027086	151.1795028
10			-33.802924	151.179492

Table of errors found

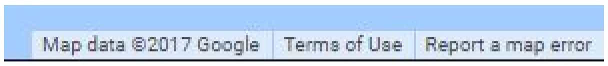
Type	Explanation	Fix
Runtime	 `GET http://localhost:8001/getJSON,clean net::ERR_CONNECTION_REFUSED` One day I opened the website without starting the server first and the page wouldn't load, this was because the application needed to contact my server to initialize. Chrome showed it as a "ERR_CONNECTION_REFUSED" error.	When loading the website, first make a handshake with server to ensure the rest of the application can be loaded otherwise alert the user to start the server.
Syntax	Once I went to disable NSW trains only to find an error that "vehicleTypesChangd" is not defined. The onclick event for the checkbox had the incorrect spelling for the function.	Add "e" to fix the spelling.
Logic	When running my interpolation algorithm for the first time, vehicles disappeared from the map. The algorithm had produced bad positions. I came to find out my order of operations was incorrect.	Added some brackets to specify calculation order.
Runtime	Occasionally the page would fail to load, this was because my randomize favicon function picked numbers from 0-3 rather than 1-4 which sometimes gave the browser a bad image thus crashing it.	Added one to the random number

Syntax	 `for (i = 0, i < markers.length, i++) {` In my change vehicle marker color function, I had used commas instead of semicolons when constructing a for loop.	Changed commas to semicolons.
Syntax	 `function setStyle (json) {` `map.setOptions({styles: json});` `}}` There was a rogue closing bracket in my set style function.	Removed rouge bracket.

Social and Ethical Considerations

Copyright

To not infringe on Google Maps' copyrighted data, I have left a watermark and link to Google on the bottom of the map. This is also where the "terms of use" link is found. I have studied this document and adhered to it (e.g not charging a fee and not creating a competing service) as to not break the terms of use. Seeing as how the vehicle data is public, no reference was needed towards it.


Map data ©2017 Google | Terms of Use | Report a map error

Ergonomics / Ease of Use

To make the application easier to use, I have ensured that all mouse clicks generally occur in the leftmost 30% of the screen. This prevents repetitive mouse movements across the screen. I've used input elements found in HTML5 since they have keyboard navigation built in letting the user mostly navigate my interface without having to use the mouse if they so desire. This extends to the warning prompts that can be dismissed by simply pressing enter.

Accessibility

Again, because HTML5 elements were used, a magnitude of accessibility options are built in. Text colour is also high in contrast to aid in reading for vision impaired users. Buttons are large for ease of use and for users with minor motor control issues. I've also used alt-text for some images so users with poor eyesight can learn what the image portrays.

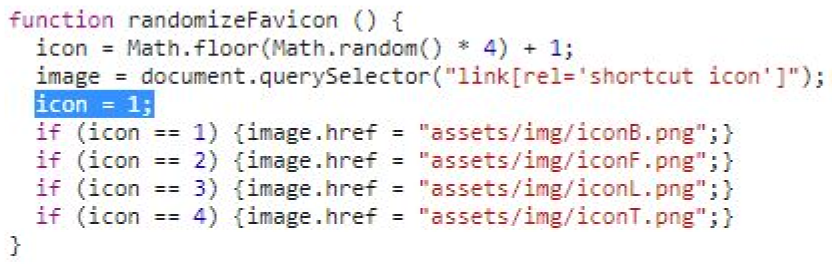
Testing, Evaluating and Maintaining

Test Data

Test Data	Justification
randomizeFavicon(): [1,2,3,4]	I needed to see if every value produced a different favicon in my randomizeFavicon function, so I used every possible value and observed that indeed, a different image was produced for each value.
goToStreetView(): [[-33.799854,151.175491], [-33.798285,151.178452], [27.175015,78.042155]]	I wanted to check that my goToStreetView function succeeded whether the coordinates were: not on a road but in the middle of a block, on a road or in another country. I came to find that it succeeded all three times.
throwError(): ["","text",<10000 character string>, 12.12, True]	I wanted to ensure that no matter the length or type of error passed, that it would still be shown to the user. The function passed all tests.
displayShape(): [Null, <one point>, <10000 points>]	I wanted to see if a line was drawn even if there were 0, 1 or 10000 points. The 0 and 1 points obviously didn't draw, but ran error free and the load test of 10000 worked flawlessly.
<server call>: [1GB of data, 200MB of data]	Load test to see if the application can send large amounts of data. 1GB failed however, this will never be seen in real use so it's not an issue. 200MB struggled but succeeded. The struggle wasn't an issue since only ever <1kb of data is sent to the server.
<server send>: [1GB of data, 200MB of data]	Load test to see if the server can send large amounts of data. 1GB struggled but surprisingly succeed. However, this struggle will never be seen in real use so it's not an issue. 200MB succeeded albeit slowly. This slowness isn't an issue since the most data that could be sent back in a real world scenario is ~2MB.

Module Testing

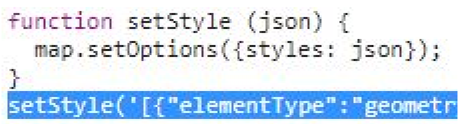
init.js - randomizeFavicon()



```
function randomizeFavicon () {
  icon = Math.floor(Math.random() * 4) + 1;
  image = document.querySelector("link[rel='shortcut icon']");
  icon = 1;
  if (icon == 1) {image.href = "assets/img/iconB.png";}
  if (icon == 2) {image.href = "assets/img/iconF.png";}
  if (icon == 3) {image.href = "assets/img/iconL.png";}
  if (icon == 4) {image.href = "assets/img/iconT.png";}
}
```

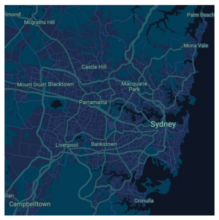
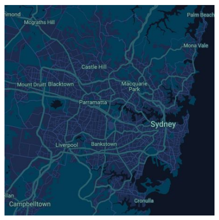
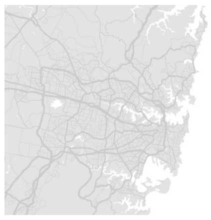
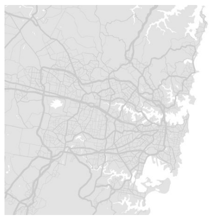
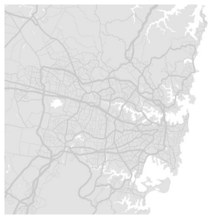
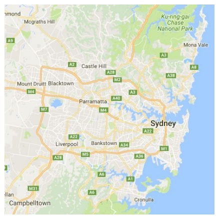
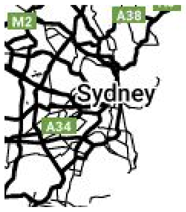
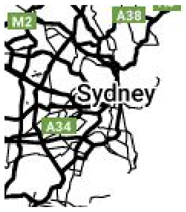
Input	Expected output	Actual output
1	iconB.png: Ⓑ	iconB.png: Ⓑ
2	iconF.png: Ⓕ	iconF.png: Ⓕ
3	iconL.png: Ⓛ	iconL.png: Ⓛ
4	iconB.png: Ⓣ	iconT.png: Ⓣ

customizer.js - setStyle()

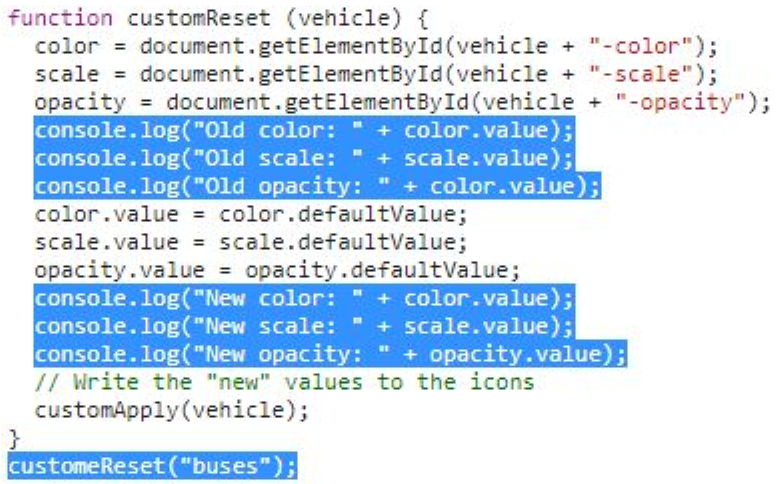


```
function setStyle (json) {
  map.setOptions({styles: json});
}
setStyle('[{"elementType":"geometr
```

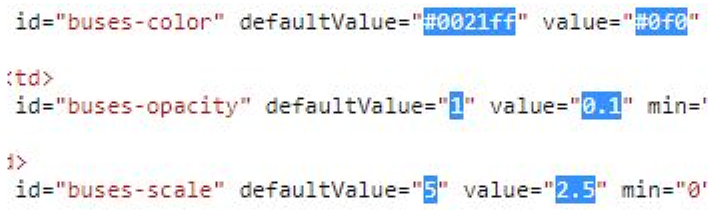
Input	Expected output	Actual output

<aubergine style JSON>		
<clean style JSON>		
<original JSON> (empty string)		
<external JSON> (one that doesn't come from the program)		

customizer.js - customReset()



```
function customReset (vehicle) {
  color = document.getElementById(vehicle + "-color");
  scale = document.getElementById(vehicle + "-scale");
  opacity = document.getElementById(vehicle + "-opacity");
  console.log("Old color: " + color.value);
  console.log("Old scale: " + scale.value);
  console.log("Old opacity: " + color.value);
  color.value = color.defaultValue;
  scale.value = scale.defaultValue;
  opacity.value = opacity.defaultValue;
  console.log("New color: " + color.value);
  console.log("New scale: " + scale.value);
  console.log("New opacity: " + opacity.value);
  // Write the "new" values to the icons
  customApply(vehicle);
}
customeReset("buses");
```

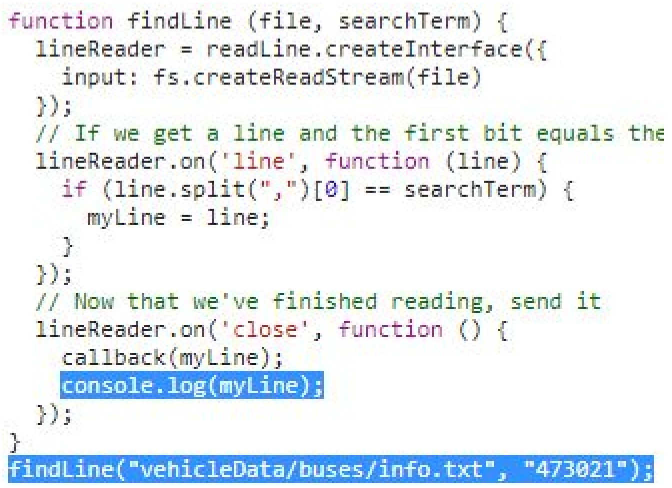

```
 id="buses-color" defaultValue="#0021ff" value="#0f0"

:td>
 id="buses-opacity" defaultValue="1" value="0.1" min='

i>
 id="buses-scale" defaultValue="5" value="2.5" min="0'
```

Input	Expected output	Actual output
[color: #0f0, opacity: 0.1, scale: 5]	Old color: #0f0 Old opacity: 0.1 Old scale: 5 New color: #0021ff New opacity: 1 New scale: 2.5	Old color: #0f0 Old opacity: 0.1 Old scale: 5 New color: #0021ff New opacity: 1 New scale: 2.5
[color: #0021ff, opacity: 1, scale: 2.5]	Old color: #0021ff Old opacity: 1 Old scale: 2.5 New color: #0021ff New opacity: 1 New scale: 2.5	Old color: #0021ff Old opacity: 1 Old scale: 2.5 New color: #0021ff New opacity: 1 New scale: 2.5

server.js - findLine()

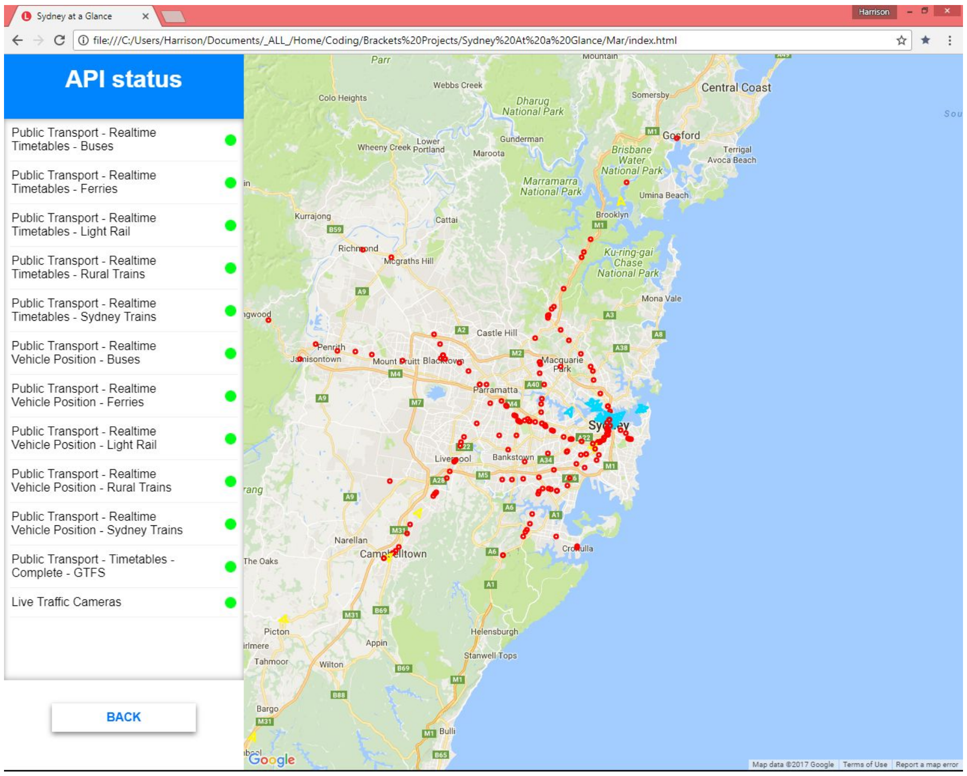

```
function findLine (file, searchTerm) {
  lineReader = readLine.createInterface({
    input: fs.createReadStream(file)
  });
  // If we get a line and the first bit equals the
  lineReader.on('line', function (line) {
    if (line.split(",")[0] == searchTerm) {
      myLine = line;
    }
  });
  // Now that we've finished reading, send it
  lineReader.on('close', function () {
    callback(myLine);
    console.log(myLine);
  });
}
findLine("vehicleData/buses/info.txt", "473021");
```

Input	Expected output	Actual output
["vehicleData/buses/info.txt", "473021"]	473021,2444_907,18148,0,1,Parramatta to Bankstown	473021,2444_907,18148,0,1,Parramatta to Bankstown

	via Bass Hill...	via Bass Hill...
[“vehicleData/lightrail/stops.txt”, “203782”]	203782,Glebe Light Rail,-33.87723,151.187415	203782,Glebe Light Rail,-33.87723,151.187415

System Testing

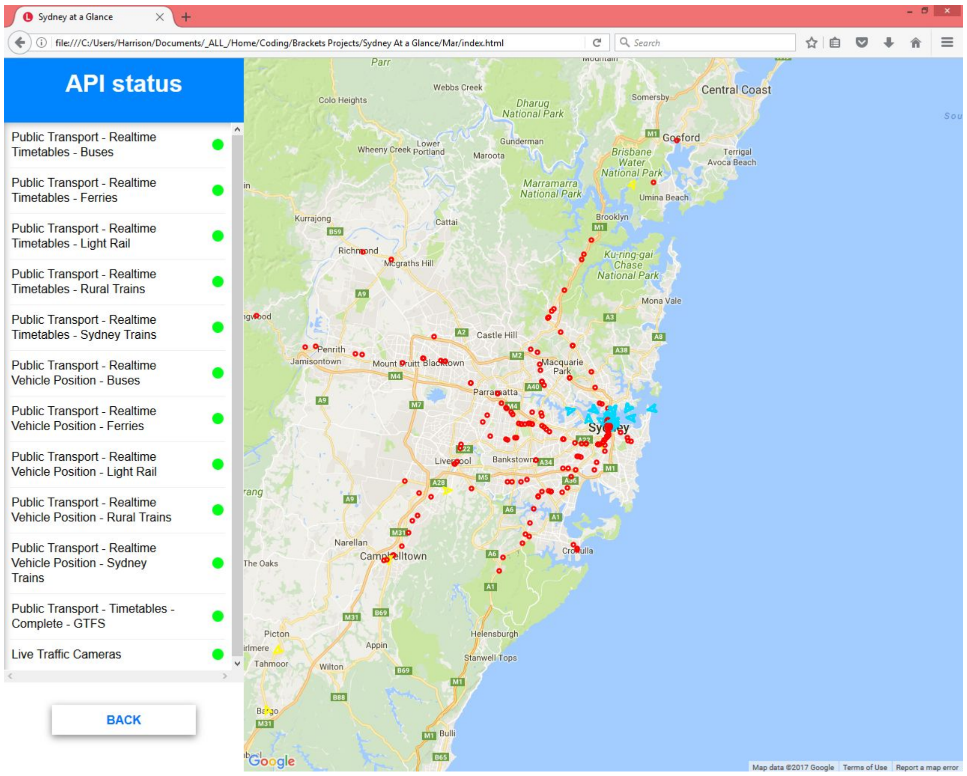
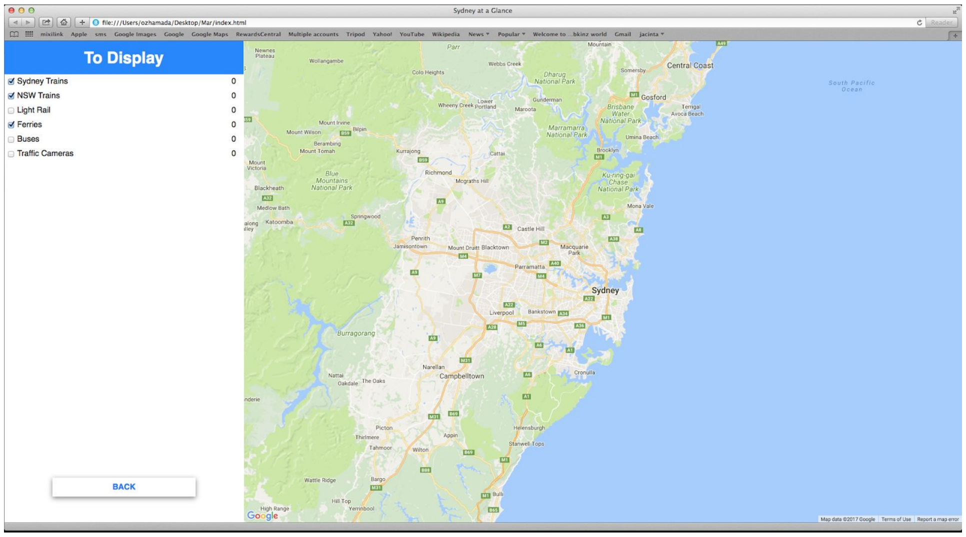
Browser Testing

Browser	Screenshot
Chrome Worked flawlessly. As it was designed and developed for it.	

Sydney at a Glance
file:///C:/Users/Harrison/Documents/_ALL_/Home/Coding/Brackets%20Projects/Sydney%20At%20a%20Glance/Mar/index.html
BACK

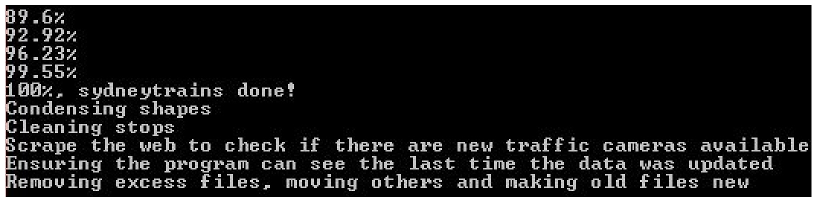
API status

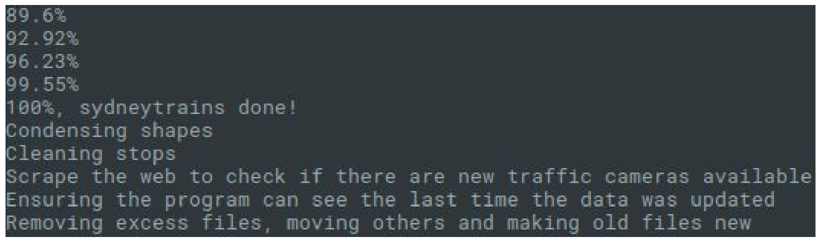
- Public Transport - Realtime Timetables - Buses ●
- Public Transport - Realtime Timetables - Ferries ●
- Public Transport - Realtime Timetables - Light Rail ●
- Public Transport - Realtime Timetables - Rural Trains ●
- Public Transport - Realtime Timetables - Sydney Trains ●
- Public Transport - Realtime Vehicle Position - Buses ●
- Public Transport - Realtime Vehicle Position - Ferries ●
- Public Transport - Realtime Vehicle Position - Light Rail ●
- Public Transport - Realtime Vehicle Position - Rural Trains ●
- Public Transport - Realtime Vehicle Position - Sydney Trains ●
- Public Transport - Timetables - Complete - GTFS ●
- Live Traffic Cameras ●

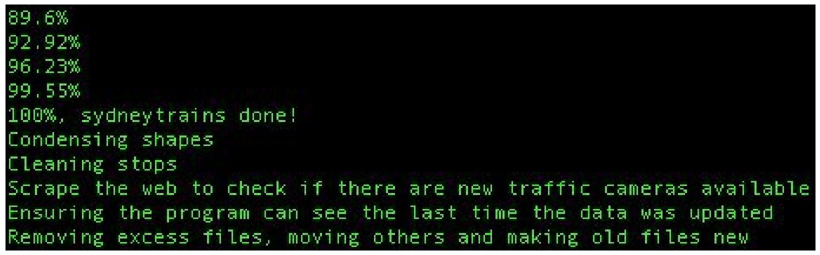
Firefox Ran very slowly with many vehicles, otherwise application ran well.	 **API status** Public Transport - Realtime Timetables - Buses ● Public Transport - Realtime Timetables - Ferries ● Public Transport - Realtime Timetables - Light Rail ● Public Transport - Realtime Timetables - Rural Trains ● Public Transport - Realtime Timetables - Sydney Trains ● Public Transport - Realtime Vehicle Position - Buses ● Public Transport - Realtime Vehicle Position - Ferries ● Public Transport - Realtime Vehicle Position - Light Rail ● Public Transport - Realtime Vehicle Position - Rural Trains ● Public Transport - Realtime Vehicle Position - Sydney Trains ● Public Transport - Timetables - Complete - GTFS ● Live Traffic Cameras ● BACK
Safari Worked but had speed issues with load.	 **To Display** ☑ Sydney Trains 0 ☑ NSW Trains 0 ☐ Light Rail 0 ☑ Ferries 0 ☐ Buses 0 ☐ Traffic Cameras 0 BACK

OS Testing (Update tool)

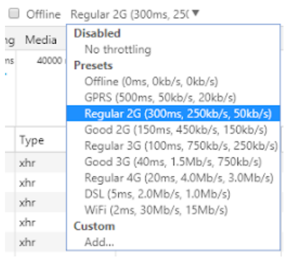
OS	Screenshot

Windows	
	`89.6%`
	`92.92%`
	`96.23%`
	`99.55%`
	`100%, sydneytrains done!`
	`Condensing shapes`
	`Cleaning stops`
	`Scrape the web to check if there are new traffic cameras available`
	`Ensuring the program can see the last time the data was updated`
	`Removing excess files, moving others and making old files new`

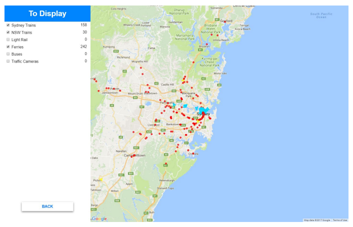
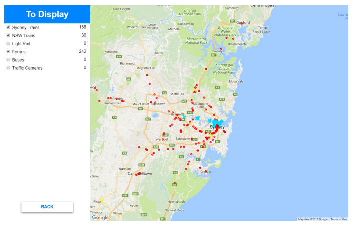
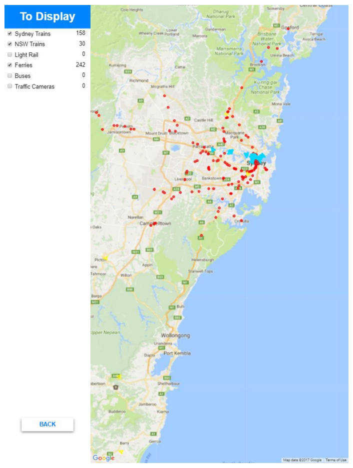
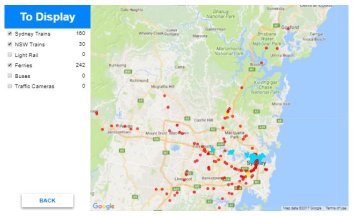
Linux	
	`89.6%`
	`92.92%`
	`96.23%`
	`99.55%`
	`100%, sydneytrains done!`
	`Condensing shapes`
	`Cleaning stops`
	`Scrape the web to check if there are new traffic cameras available`
	`Ensuring the program can see the last time the data was updated`
	`Removing excess files, moving others and making old files new`
	It then proceeded to error out right at the very end. 99% of the code ran fine but when it came to the intricacies of manipulating files in linux filesystems it broke.

Mac	
	`89.6%`
	`92.92%`
	`96.23%`
	`99.55%`
	`100%, sydneytrains done!`
	`Condensing shapes`
	`Cleaning stops`
	`Scrape the web to check if there are new traffic cameras available`
	`Ensuring the program can see the last time the data was updated`
	`Removing excess files, moving others and making old files new`

Internet Speed Testing

Internet Speed	Notes
 Offline Regular 2G (300ms, 25(Disabled No throttling Presets Offline (0ms, 0kb/s, 0kb/s) GPRS (500ms, 50kb/s, 20kb/s) Regular 2G (300ms, 250kb/s, 50kb/s) Good 2G (150ms, 450kb/s, 150kb/s) Regular 3G (100ms, 750kb/s, 250kb/s) Good 3G (40ms, 1.5Mb/s, 750kb/s) Regular 4G (20ms, 4.0Mb/s, 3.0Mb/s) DSL (5ms, 2.0Mb/s, 1.0Mb/s) WiFi (2ms, 30Mb/s, 15Mb/s) Custom Add...	Map loads very slowly but vehicle's load fine
Regular 3G - 750Kb/s	Map loads almost instantly, vehicles are instant
Regular 4G - 4Mb/s	Map and vehicles load instantly

Screen resolutions

Resolution	Screenshot
1440x900	
1280x800	
768x1024	
640x384	

To Display
- ☑ Sydney Trains 158
- ☑ NSW Trains 30
- ☐ Light Rail 0
- ☑ Ferries 242
- ☐ Buses 0
- ☐ Traffic Cameras 0

BACK

Modifications in Response to Testing

Issue	Consideration	Fix
When I wanted my map to revert to its regular styling, I sent an empty string to be set as the JSON to wipe changes. My code recognized that an empty string wasn't valid JSON and thus the map styling remained as it was.	This is a moderate issue since clearing the map styling is a prominent option.	Add an if statement to check if the styling string sent is empty and if so, check my JSON validator.
Linux failed to run the update tool completely.	Linux isn't an officially supported platform for my program so I didn't fix. Regardless, the rest of the program works, you just won't see specific vehicle data.	
With a slow 2G internet connection, the map loads incredibly slowly.	This isn't an issue since the program was designed to run on a computer. Also, internet speeds are much faster than 2G these days. Finally, it was the map loading slowly, that relies on Google's API, so I couldn't fix it if I wanted too. It's on them.	

Evaluation

What I am Satisfied With

I'm satisfied with the fact that I had this idea in my head and was able to fully realise every aspect I desired. I'm happy that most computers will be able to process the sheer amount of data I'm pushing. I'm also very pleased that I could visualize so many data sets. Real time positions, routes, stops, traffic cameras, road congestion, API statuses... Finally, I'm glad that I was able to access and use all this data for free and that the data is (more or less) consistent. I'm also glad that the Google Maps API and the real time vehicle position API were very easy to communicate with. Oh and I'm happy with the design of the webpage.

What I am Unsatisfied With

It took a very very long time to get this project off the ground. The documentation for the transport data is poor in places and took a long time to understand. I started off with the real time vehicle position API but kept failing to read the data. After much deliberation I discovered that I needed a library from Google to decode the data. What also took a lot of time was figuring out how to access network data such as routes, stop sequences, congestion of vehicles… The documentation continually referred to this API that provided all this data. I searched for a long time to figure out how to use this API. I came to find out that it was no API at all but more or less a download link for very large zipped CSVs that needs to be redownloaded daily. This created the need for a separate downloading and parsing tool. The unzipped data could be 500MB! I tried creating databases and call requests to turn these CSVs into my own API but there was too much data. The update tool prunes all the fat, trims the files and makes them small enough to be searched line by line to find specific data. I just very much wish that the Open Data website had implemented a search API themselves rather than getting every client to download such large amounts of data. DAILY. The data between each vehicle is very different and the Ferry data is in a world of its own. The shape files for the routes they take still baffles me which is why it isn't included in my application because last I checked, boats can't walk over the CBD... I am also unhappy with how the sheer amount of buses can slow down the browser very much.

What I could Improve

The interpolation algorithm could include a prediction service that would guess more or less the path that a vehicle might take in the close future to reduce jolts in vehicle movement. The traffic cameras could automatically update their image every minute rather than having the user do that themselves every minute. Data finding could be sped up by implementing a database. Sydney train bearings could be calculating by finding the vector between the last and current point. Finally, I could lighten each vehicle's processing power so that many many buses wouldn't slow down the browser.

If I were to do it Again…

I'd use a friendlier data set that has all the searching features built into the API and use this extra time to implement a routing algorithm that helped a user find the fastest way to get from one place to another. I'd also have a current vehicle search so that, for example, a mother could search the model number of a train her son is on to see his progress on his journey and to calculate when she should leave to pick him up. However, overall, I'm very happy.