CYCLEPLAN

Cloud Computing Project

cycleplan-s3542413.appspot.com

Joseph Johnson
S3542413

Table of Contents

Objectives

The main objective of the CyclePlan project was to create a web application that was integrated with Google and Google Maps, that would allow users to create and save routes so they could plan cycling trips. I wanted the application to integrate more data into the route than Google Maps or other similar services currently does, to give a user a better picture of what routes are safe and what type of weather to expect along the way.

Introduction

Motivation

The motivation for creating this project was to allow cyclists to choose routes that were safer by integrating VicRoads collision data with the website, as well as integrating weather details for start and finish points using the Dark Sky API. This would allow users to get a full picture of what areas they should avoid on a bike, or if they want to go on a ride that day depending on the weather. Another major motivation was to allow users to save routes, which is not always possible on other services that are similar.

What the App does, why is it required, and what new things does it offer?

As mentioned previously, the app is required because it adds features that other similar apps or websites do not offer. CyclePlan integrates with Google Maps, giving users the most up-to-date maps available and also giving users access to mapped cycleways. Several Google API's were used in development to add certain features; such as the map itself, the ability to autocomplete when searching for places, and Google directions services to allow routing for bicycles. Once a user plans a route they are then given information about the start and destination. The App uses the Dark Sky API provided by forecast.io to get the latest weather data for the ride, helping s rider make a judgement call on if today is a good day for a ride. The user will also be shown the average number of accidents involving cyclists in those areas, which allows them to make an informed decision on if those areas are safe for riding or not.

The features mentioned above are not available in other cycle-planning type applications and web services. I wanted to cyclists the ability to make informed decisions on if today is a good day for riding, and if the route they chose is a safe one. Many times, whilst planning a ride, cyclists will be going to areas they are not familiar with and roads they have not been on before, so they have no way of knowing if they are going to be safe or if the roads are narrow and likely to lead to accidents. CyclePlan helps cyclists know which areas might be unsafe, and also automatically routes users onto dedicated cycle paths where possible.

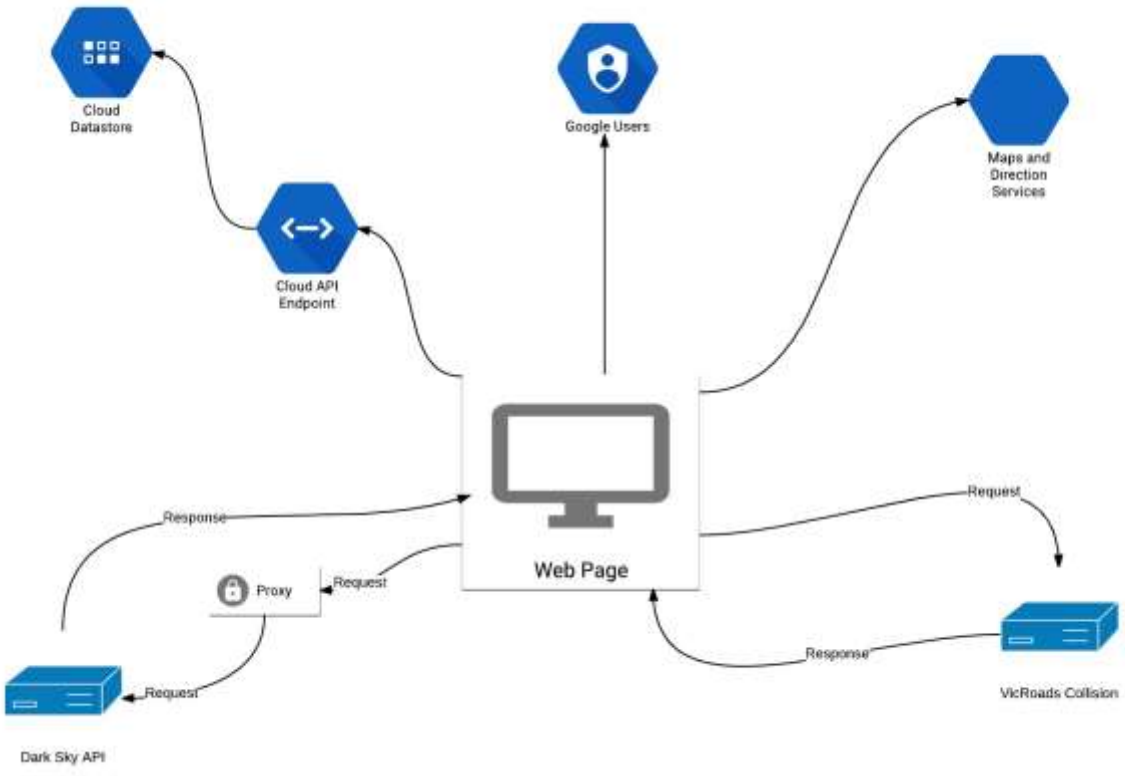
Another major benefit that CyclePlan offers is the ability to save rides for future reference – a feature that not all other mapping services offer. Specifically, Google Maps does not offer a simple way for you to save a route for future use. CyclePlan lets users save their favourite routes so that they can come back to them later, either to check the directions again, or to find the latest weather information.

Related Work

There are other applications and websites that offer some of the services that are offered by CyclePlan. The first one would be Google Maps itself, which would allow you to create routes designed for bicycles.

However, Google Maps does not offer any of the other information such as collision data or weather information. Another very popular application for mapping routes is Strava (www.strava.com). This application is more tailored towards recording rides rather than planning rides however. Planning is possible with Strava but it can become quite tedious as you have to manually plan the route step-by-step. Another similar app to Strava is Garmin (www.connect.garmin.com) – however this is very similar to Strava in its features.

Software Design



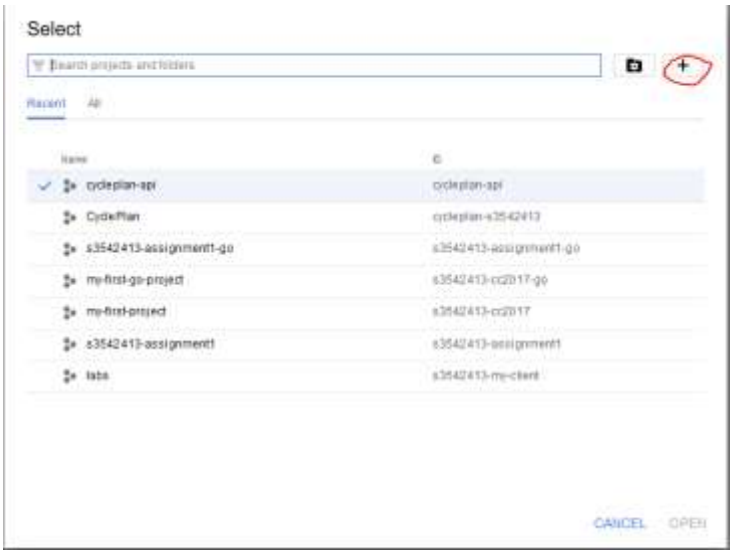
Above is a high-level diagram of how the website works and interacts with API's and Google's Datastore service. In short, a user can login using their Google account – this is not required for searching rides but is required for saving rides. A user can then search for places on the map and get directions using Googles Maps and Directions services. Once a route has been searched for, the website will make a request to VicRoads collisions data, and another request – via a proxy – to the Dark Sky API. Once the responses are received the information is displayed on the web page. Finally, if a user logs in and has saved rides, the Google cloud Datastore is accessed via an API endpoint I wrote in Java and deployed on Google's Cloud – all of a users existing saved rides will be displayed, and they will also be able to save the ride they are currently viewing which will be saved to the datastore via the endpoint API.

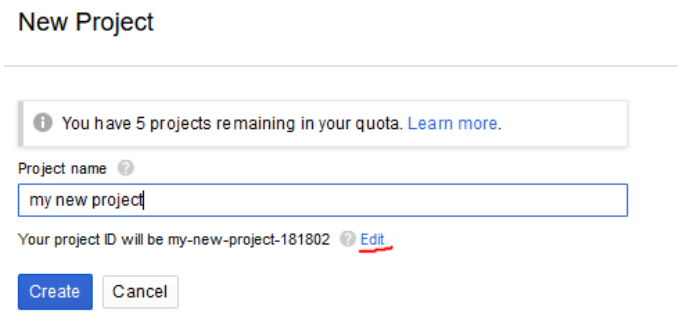
The web page and associated scripts are hosted on Google's Cloud (cycleplan-s3542413.appspot.com).

Implementation

1. Create a project using Google Cloud

Go to console.developers.google.com and create a new project to host the web page.





You may want to change the project ID to something you will remember as this is what will be used as a URL to access the website.

2. Deploy the project using the Google Cloud Console

First set the deault project in the console to your new project id, using the command:

```
gcloud config set project [YOUR_PROJECT_ID]
```

Once you have a project created and ready to be deployed (including an app.yaml file), you can deploy your app using the command (make sure the current directory is the project directory):

```
gcloud app deploy
```

If this is the first time deploying, you will have to set some default options, otherwise the command will run as normal and update the files on the cloud with your current directories files.

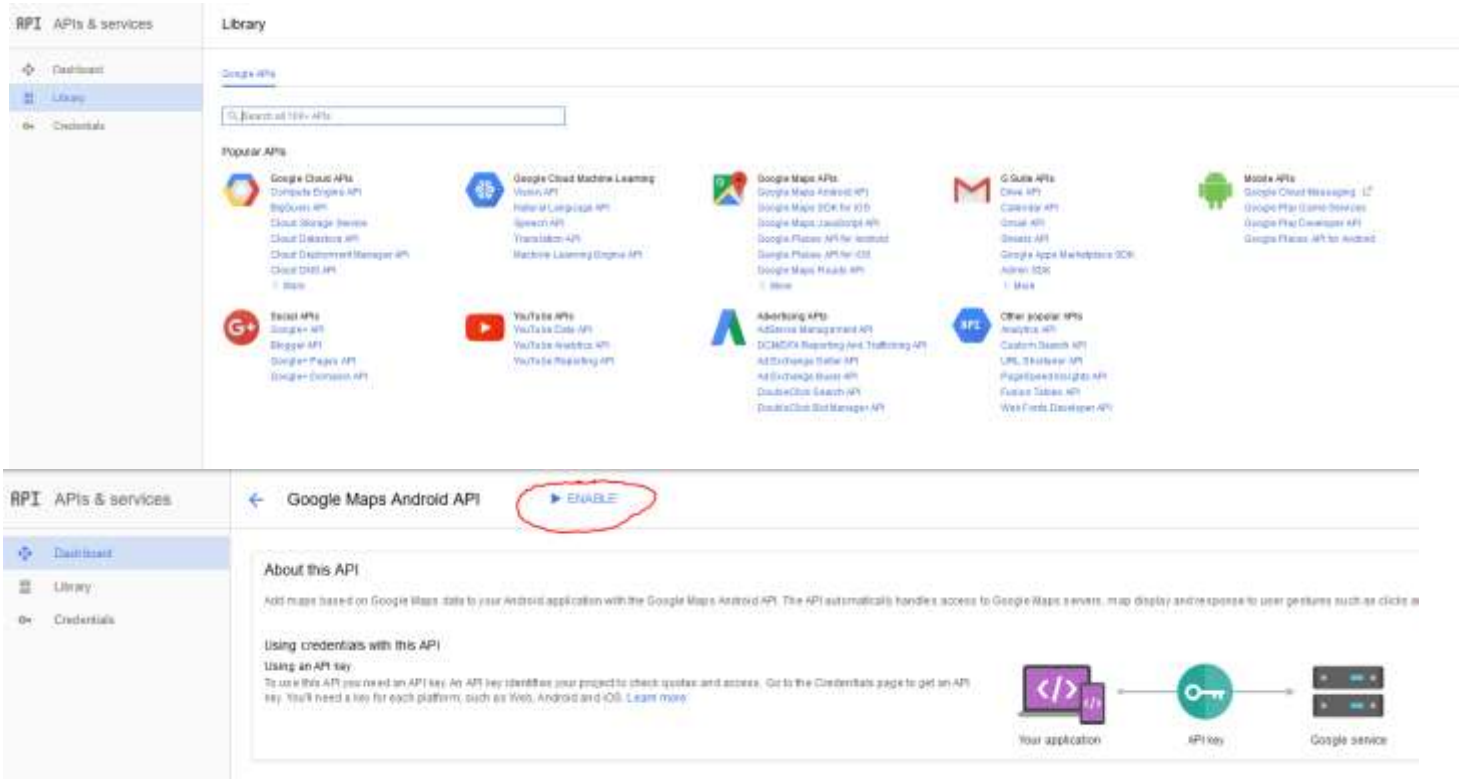
3. Enable the relevant API's

Google Cloud offers many different API's to interact with Google's services, however most of them need to be enabled first. The API's I enabled are:

- Google Maps JavaScript
- Google Maps Distance Matrix
- Google Maps Directions
- Google Maps Geocoding
- Google Cloud Datastore

Some of these API's require an API key to be used, so make sure to note them down once they are enabled – they will also be saved in the credentials section of your API dashboard.

To enable an API, go to console.developers.google.com/apis. You can then click 'Library', search for the relevant API's, then click 'Enable'.



4. Create Endpoint API for accessing Google Datastore

Create an Endpoint API that will access Google Datastore – similar to how to was done in **Lab/Tute #3** on the course blackboard. However, there are some important changes - we will be saving a Ride object rather than a Quote object. This means the variable names and types need to be different (as well as the class name).

```
@PersistenceCapable(identityType = IdentityType.APPLICATION)
public class Ride {
    @PrimaryKey
    @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
    Long record_id;
    @Persistent
    String user_mail;
    @Persistent
    String start_lat;
    @Persistent
    String start_lng;
    @Persistent
    String end_lat;
    @Persistent
    String end_lng;
```

In the datastore I saved the above attributes. ‘record_id’ was used as a unique identifier for that entry in the datastore, ‘user_mail’ was used to identify which user owned that ride. The latitude and longitude are used to find a location on Google Maps.

5. Write API in JavaScript for interacting with your endpoint and Google Datastore

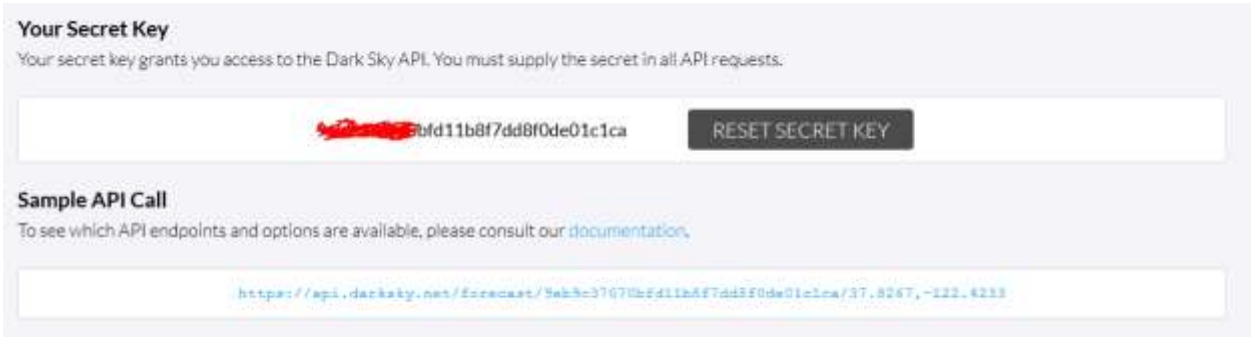
Normally, Datastore will have four general interactions (insert, delete, edit, view). In my application, only insert and view were used as I saw no use for a user to edit a ride. First you need to initialise/load the API, and then have functions that call the endpoint functions.

```
function initApi() {
    gapi.client.load('rideendpoint', 'v1', null,
        'https://cycleplan-api.appspot.com/_ah/api');

    document.getElementById('save_route').onclick = function() {
        saveRide();
    }
}

function saveRide() {
    if (checkSignIn()) {
        var email = document.getElementById('user_id').value;
        var s_lat = document.getElementById('s_lat').value;
        var s_lng = document.getElementById('s_lng').value;
        var d_lat = document.getElementById('d_lat').value;
        var d_lng = document.getElementById('d_lng').value;

        var request =
        {
            userId: email,
            startlat: s_lat,
            startlng: s_lng,
            endlat: d_lat,
            endlng: d_lng
        }

        gapi.client.rideendpoint.insertRide(request).execute(function(resp) {
            if (!resp.code) {
                alert('Ride saved');
            }
        });
    }
    else {
        alert("You must be signed into Google to save rides");
    }
}
```

```
function viewRides() {
    if (checkSignIn()) {
        gapi.client.rideendpoint.listRide().execute(function(resp) {
            var user_email = $("#user_id").val();
            for (var i = 0; i < resp.items.length; i++) {
                if (resp.items[i].userId == user_email) {
                    createList(resp.items[i]);
                }
            }
        });
    }
    else {
        alert("You must be signed in to view your rides");
    }
}
```

6. Get API Key for Dark Sky

Go to www.forecast.io and click on ‘Developers’, then click ‘Try For Free’. You can register for the API using an email address. Once registered and you have confirmed your email address via a link you can login and obtain your API key.



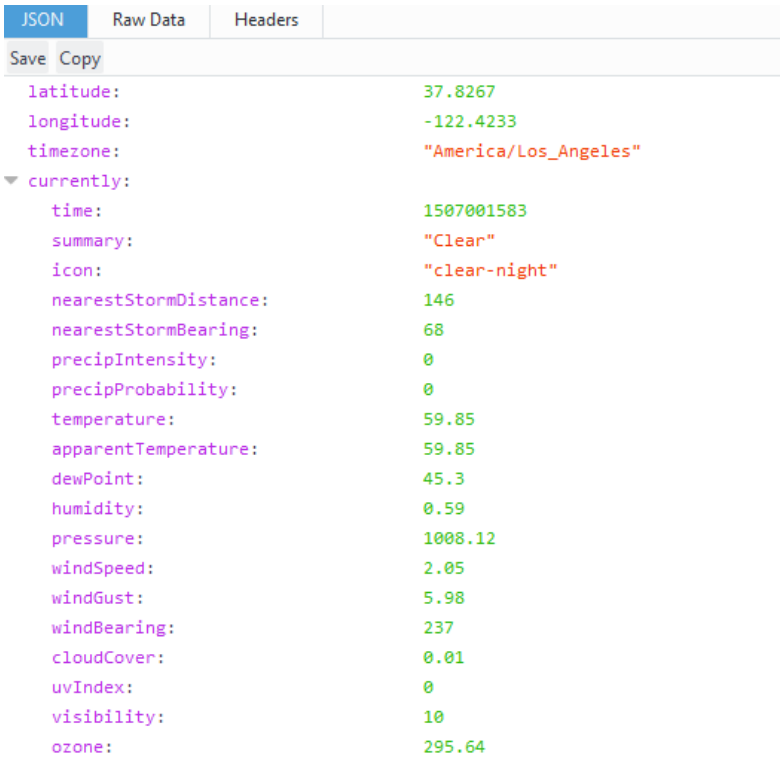
You can also see a sample API call on this page which takes the form:

```
https://api.darksky.net/forecast/[YOUR_KEY]/[LAT],[LNG]?units=si
```

This request cannot be made on a client-side script without a proxy – so that your key cannot be compromised, I used cors-anywhere to make my request which is a free proxy provided online. Using cors-anywhere, the request now takes the form:

```
https://cors-anywhere.herokuapp.com/[DARK_SKY_REQUEST]
```

A request response will be in the form of JSON, which contains lots of information about the weather for that location. I selected certain parts of the weather to display that I felt was relevant to cycling. I parsed the JSON using an ajax request in JavaScript/jQuery.



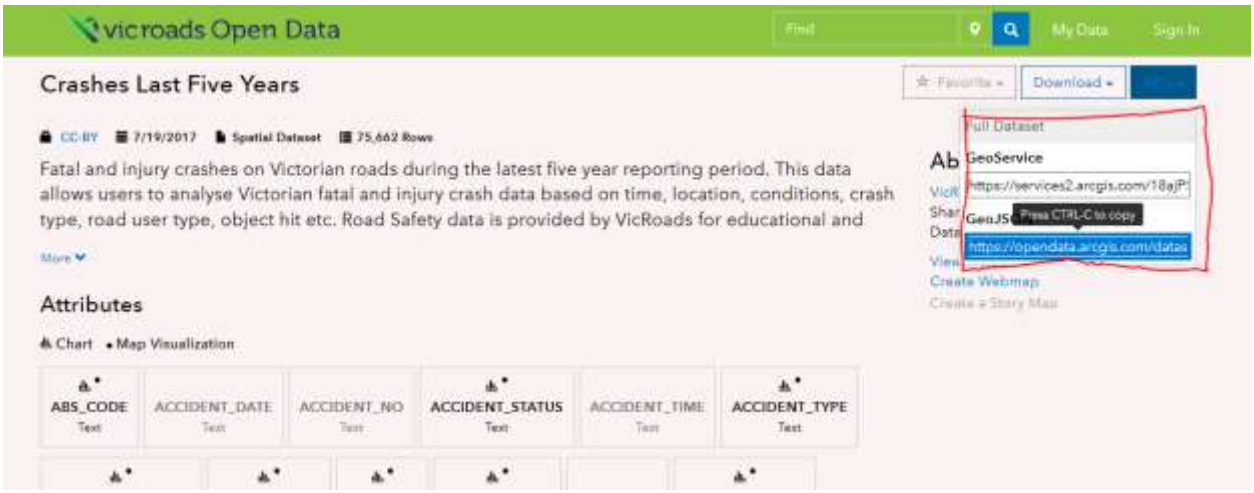
7. Make VicRoads data request

As with Dark Sky, I made the request to VicRoads using ajax which returned a JSON object. VicRoads requests contain a lot of data and have many options, I chose options specific to bicyclists and my request URL was:

```
https://services2.arcgis.com/18ajPSI0b3ppsmMt/arcgis/rest/services/Crashes_Last_Five_Years/FeatureServer/0/query?where=1%3D1&outFields=INJ_OR_FATAL,BICYCLIST,LGA_NAME_ALL&outSR=4326&f=json
```

You can browse the data and options at http://vicroadsopendata-vicroadsmaps.opendata.arcgis.com/datasets?t=Road%20Safety%20Open%20Data. One minor problem I ran into whilst searching this data is that VicRoads do not classify crashes into suburbs and towns, they use government townships. For example, no data would appear for 'Hawthorn', but it would for

'Booroondara' as that is the name of the government that Hawthorn belongs to. This means that some suburbs may show slightly inaccurate data.



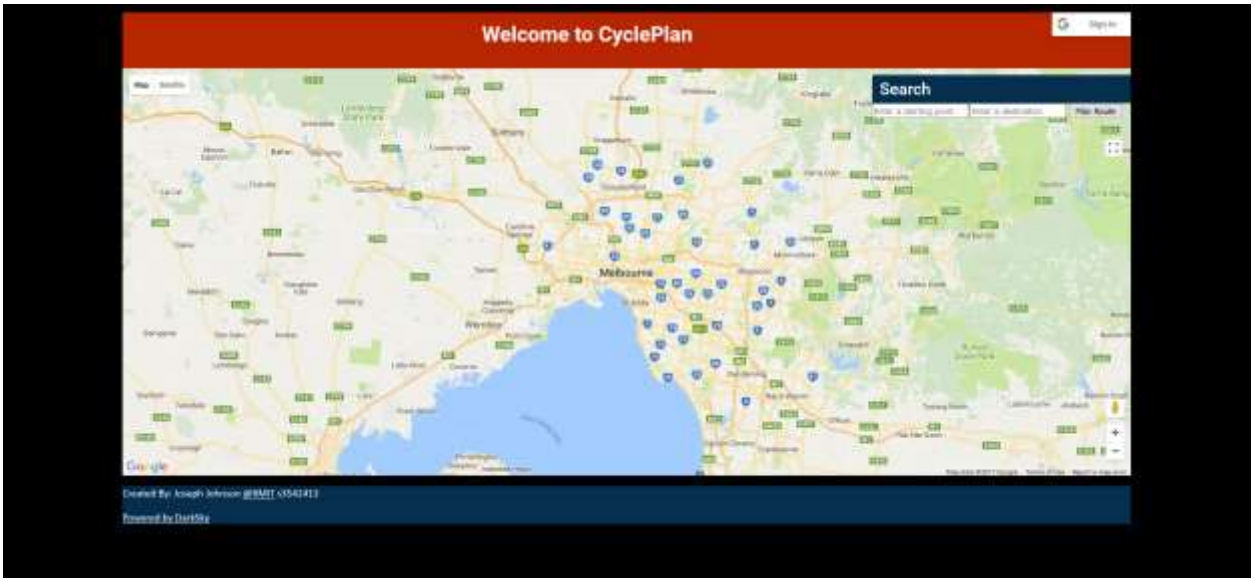
Summary

The above information provides the basic steps to recreate my project, however it does not include all the JavaScript scripting to make the web page interactive and to parse and display the correct data which is where most of my time was spent. All JavaScript code is available in the code folder in the index.js file.

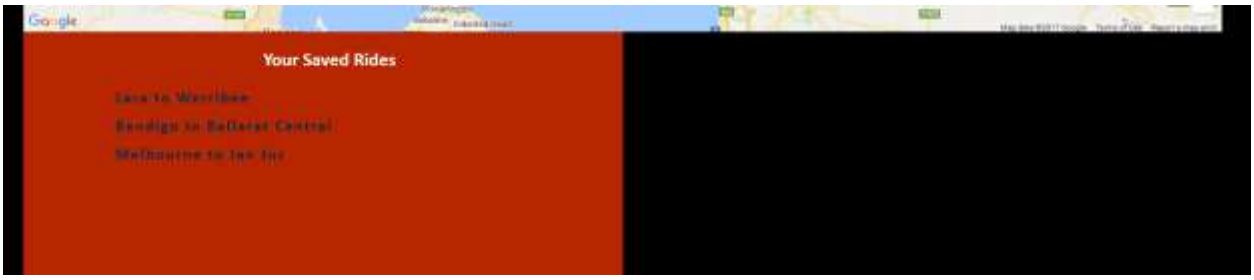
User Manual

Go to cycleplan-s3542413.appspot.com to get to the welcome page.

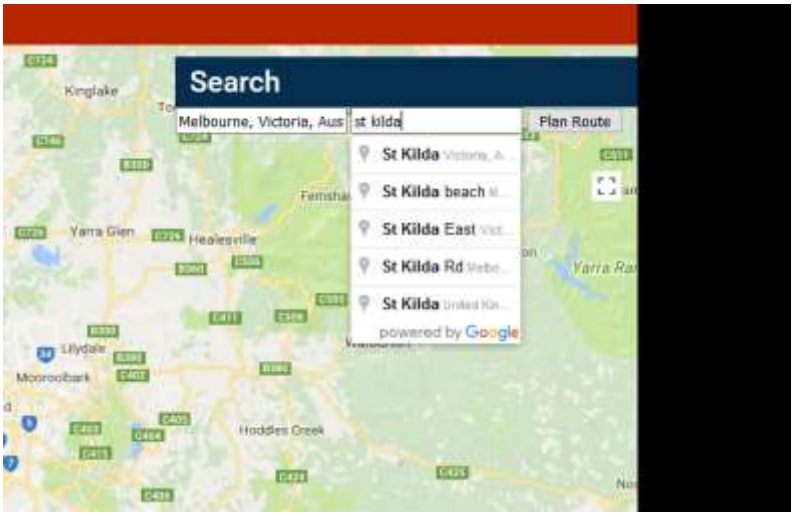
Click on the Google Sign In button in the top right corner to sign in using your Google account.



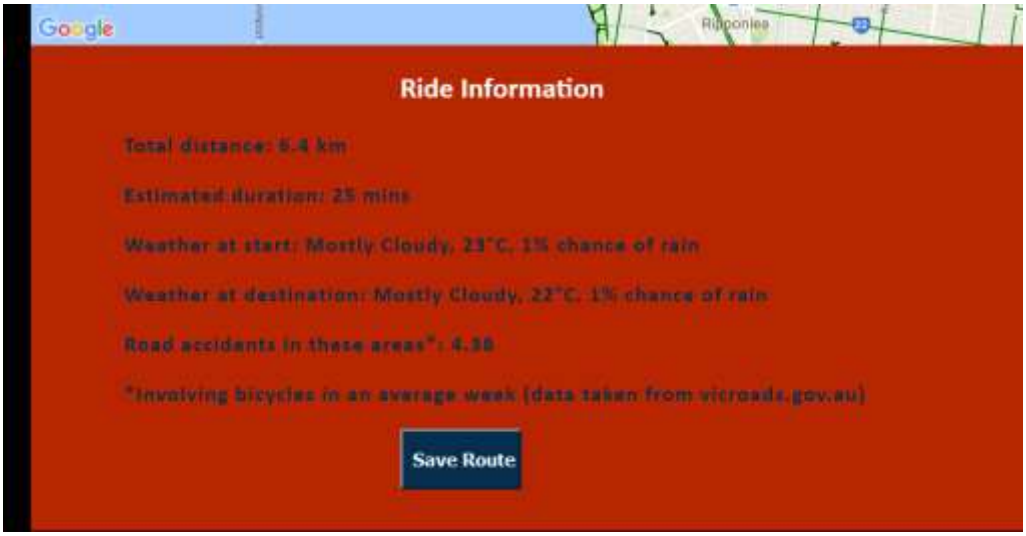
Once you are signed in, you will be shown a list of your saved rides – if any exist.



If you click on a saved ride it will load the route on the map and display the route information. This can also be done by manually searching for locations. On the map there is a search bar that will allow you to search for locations, it will aid you by using autocomplete – focussing on areas that are currently in the maps viewport.



Once you click ‘Plan Route’ the route will be displayed on the map. You will also be presented with route information such as distance, time, weather, and collision data.



This screen also gives you the option to save a route. If a route save was successful you will be presented with a dialog box informing you the route was saved.