INB356 – ASSIGNMENT 1 REPORT

PETTER HARSEM – 8683417

INTRODUCTION

OVERALL DESCRIPTION

My mashup project, Eventfinder, is a web application based on APIs from Google Maps, Eventful and OpenWeatherMap. The application provides a service that places events on a map together with a weather report for the venue. The purpose of this service is for users to look up events on a map by location, dates and categories, and see what the weather is at the specific location.

The interface of the application is based on Google Maps, and each time the map is changed, the application request event info from the Eventful API based on the current location. When the event information is received, a marker for each event is placed on the map. When clicking on the markers, the users can see information about the event, together with weather information provided by the OpenWeatherMap API.

The application is a great way to find information about events worldwide, and whether you want to see what is going on in your own neighbourhood or you want something to do while you are on vacation in a foreign city, Eventfinder can help you. The weather information provided for each event saves you from having to look up the weather on an external site, and it helps you plan for the current conditions.

SERVICES AND APIs USED

Google Maps JavaScript API v3 (https://developers.google.com/maps/web/)
The Google Maps API lets developers build highly customisable maps, and use these maps to create rich web applications. Google provides several libraries and services, such as Geocoding, Directions and Street View, that the developers can utilise in their applications.

Eventful API (http://api.eventful.com/)
Eventful is the world's largest collection of events around the world. Their API lets developers access and use all of the Eventful data and functionality in their applications. Their search options include event categories, venues, locations, dates and much more.

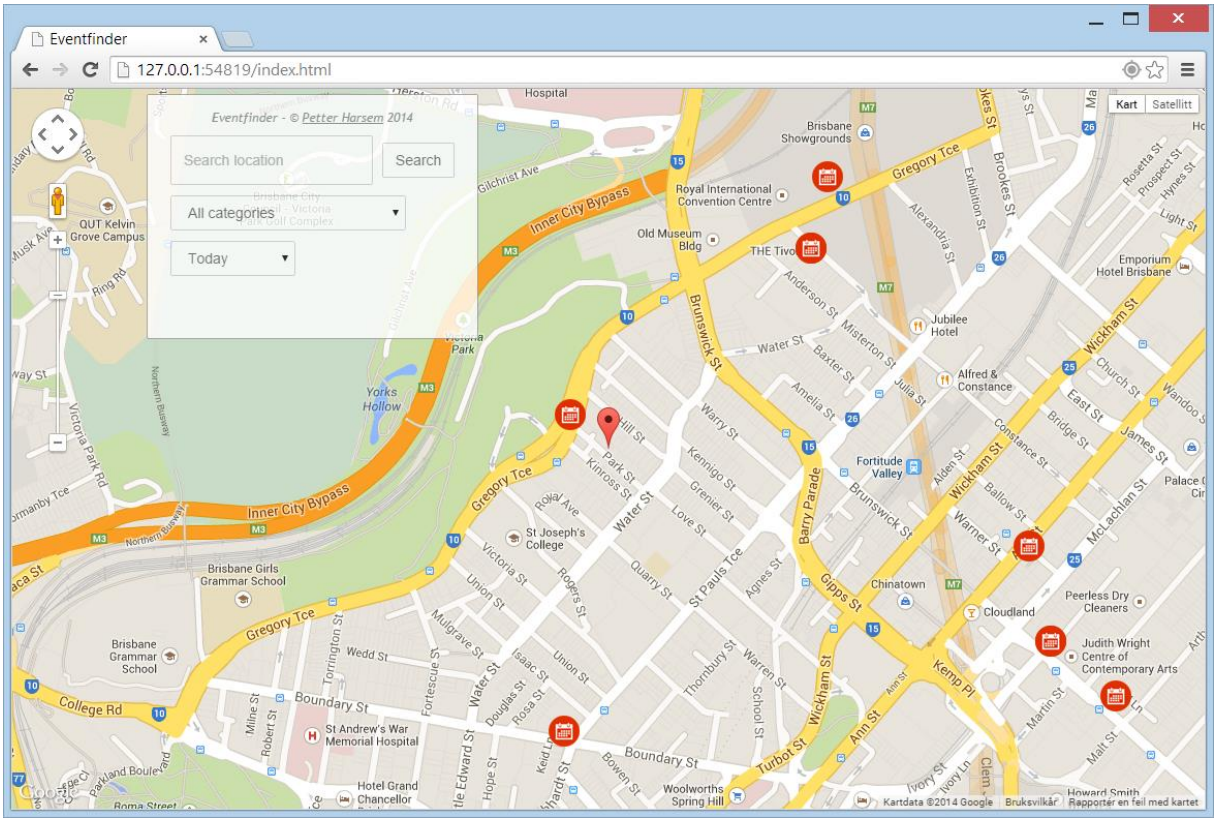
OpenWeatherMap API (http://openweathermap.org/api)
OpenWeatherMap provides an API with access to weather data such as current weather, weather forecasts and weather history. It uses data from official meteorological broadcast services, radars and official weather stations, and provides this information for developers.

USE CASES

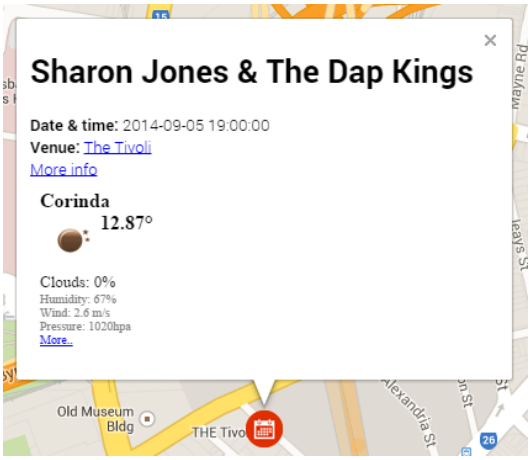
USE CASE 1

As a local resident, I want the system to help me find out what events are on in my neighbourhood so that I don't have to sit at home being bored.

When first loading the Eventfinder, it collects your position, centres the map on you and load today's events that are within the radius of the viewable area.



A calendar icon on the map represents each event, while a normal map marker represents the user's location. After loading the map, the user can check out the nearby events by clicking on the icons.



In the info window that opens up, the user can see what the event is, and if interested, more info can be found by clicking the link.

The API call to get the map for this specific example is as follows: https://maps.gstatic.com/cat_js/maps-api-v3/api/js/18/o/intl/no_ALL/%7Bmain,places%7D.js

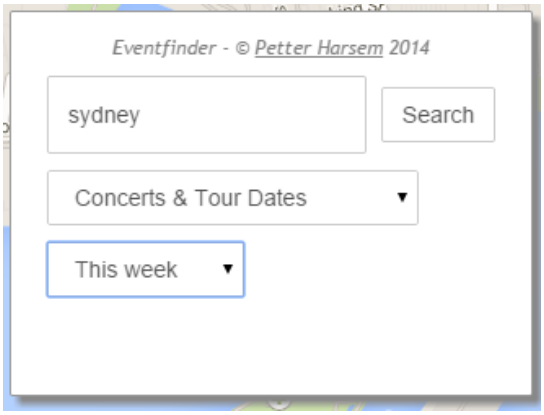
The API call to get the event info from Eventful uses the centre of the map as the "where" parameter, the distance to the north-western corner of the viewable are as search radius and finds all events for today:

http://api.eventful.com/json/events/search?app_key=Tt3kFgcQxK8gR45v&where=-27.45423524329223,153.03555428842165&within=1.3114810762491058&page_size=100&sort_order=date&date=today&callback=processJSONP

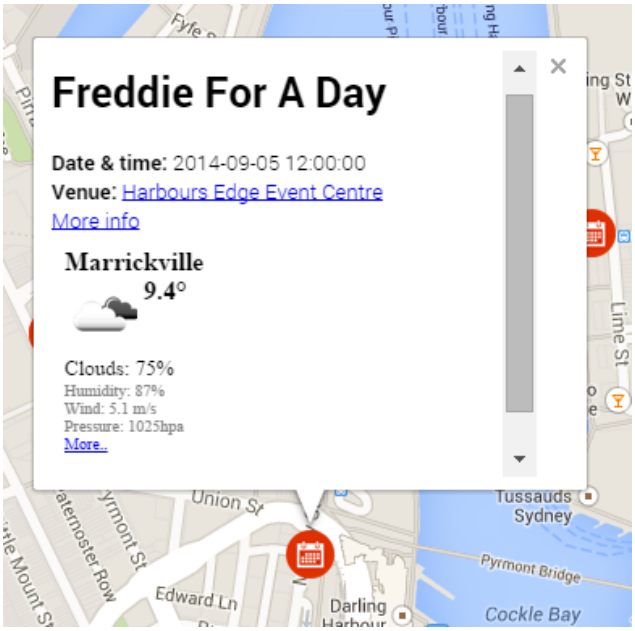
USE CASE 2

As an event goer, I want the system to show me the weather for the event I'm going to, so that I can plan and dress accordingly.

For this user story, the user can specify criteria to find the specific event on the map:



When the event is found, the user clicks the marker and the info window appears. A call to the OpenWeatherMap API is made, and the weather for the venue is loaded into the info window:



If the user want's more info about the weather, for example to see a forecast for the rest of the day, this is available by clicking "More..."

Google Maps API request: https://maps.gstatic.com/cat_js/maps-api-v3/api/js/18/o/intl/no_ALL/%7Bmain,places%7D.js

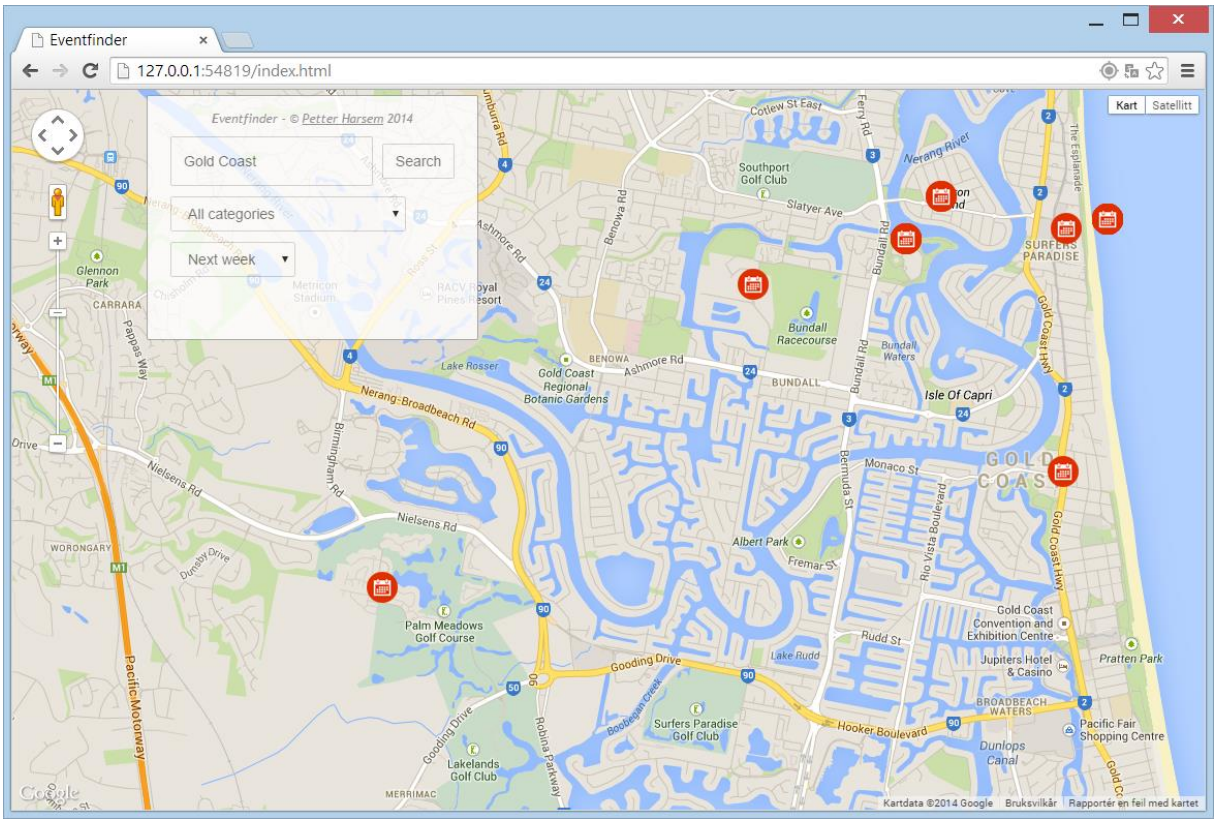
Eventful API request: http://api.eventful.com/json/events/search?app_key=Tt3kFgcQxK8gR45v&where=-33.86367399399354,151.20703311534433&within=1.2272246499537 53&category=music&page_size=100&sort_order=date&date=this%20week&callback=processJSONP

OpenWeatherMap API request, using the location of the clicked marker: http://api.openweathermap.org/data/2.5/weather?lat=-33.8706999&lon=151.19781&appid=49ff81ec495df27daf16057634e4aa03&mode=html

USE CASE 3

As a journalist writing for a local newspaper, I want the system to tell me what events are on in the local area so that I can plan what events I can cover.

If a journalist wants to plan the coming week, they can find their local area and get the events for next week:



The journalist can then investigate each of the events, find out if it is worth covering, and come up with a plan for the week.

Google Maps API request: https://maps.gstatic.com/cat_js/maps-api-v3/api/js/18/0/intl/no_ALL/%7Bmain,places%7D.js

Eventful API request: http://api.eventful.com/json/events/search?app_key=Tt3kFgcQxK8gR45v&where=-27.997813791670968,153.39080852513 43&within=3.016459096696077 6&page_size=100&sort_order=date&date=next%20week&callback=processJSONP

TECHNICAL DESCRIPTION

The user interface of the application is very simple and is based on a map provided by the Google Maps API that fills the whole window. There is an overlay options panel, and dynamic content on the map. The HTML and CSS is therefore simple and sparse, and all the action is happening in the JavaScript.

In the JavaScript file, there is one main function and a helper function that is executed as soon as the document is ready. The helper function is just collecting event categories from the Eventful API to populate the drop down list in the options panel. The other function that starts when the document is ready initialises the map and geolocation feature, as well as adding three event listeners for when the map is fully loaded the first time, for each time the zoom level changes and for when the user finishes a dragging operation.

These three event listeners all execute the same function, the function to start the loading of event info. This function determines the search parameters to be used in the Eventful API request by looking at the latitude and longitude of the current centre of the map, the radius determined by the distance from the centre to the corner of the viewable area and the user specified options. It then sends the JSON request by use of a helper method, and the data that is returned from this request is processed in another function.

This other function is what places the event markers on the map. It iterates over the data returned from the API request, and picks out the relevant information. This information is used to create markers, and generate content for the info windows that are assigned to each marker through the use of an event listener that detects clicks on the markers. Within this event listener, a function to collect weather data is called.

Having the weather data collection in a separate functions means that the info is only collected if the user click on the specific marker, and this lightens the load and lets the application work smoother. There is a lot of data requests that needs to be made to get the events each time the viewable area of the map is changed, so it is necessary to try to lighten the load wherever possible.

The location search in the options panel is handled by a geocode function that centres the map on the search result. This function also calls the loadEvents() function, making the application load an updated set of events as soon as the map jumps to the search result.

BUGS AND SHORTCOMINGS IN THE CURRENT APPLICATION

The application is fairly stable, but there is a few bugs that occasionally appears, without affecting the user experience a lot. The console reports an uncaught TypeError now and then; this comes from the part of the code that is supposed to get the latitude and longitude from the collected event data. I haven't had time to investigate the issue much, but I suspect that it is because of the use of AJAX, and that the application tries to get the info from an object that is not fully loaded at the time. Sometimes this causes the process of placing event markers on the map to stop, and the user have to change the viewable area of the map so that the event loading can start over again.

The error handling in the application should have been improved. If the Eventful API is unavailable, the application will not work, but the user is not notified about this. If the OpenWeatherMap API is unavailable the weather info will not show, and again there is no way for the user to know why it fails.

The weather locations that are pulled from the OpenWeatherMap API is often pretty far off, but this is not a fault with my application. The request contains the exact latitude and longitude that I want the weather for, but sometimes the places returned are just wrong, like some locations in Fortitude Valley that OpenWeatherMap seems to think is in St. Lucia. However, this should not have much, if any at all, impact on the correctness of the weather info, so it is really a triviality.

Another shortcoming with the application, at least from the developer's point of view, is the fact that the JavaScript code could have used some cleaning up and some more commenting. Because of time restrictions, I did not get to do this before submission. The order of the functions is random, and there is several functions without any comments. While the code itself should be fairly readable and understandable for people with technical skills, some more comments would have been good. The code also contains some very inconsistent use of jQuery, and this should have been fixed.

COMPROMISES & ISSUES ENCOUNTERED

The only big compromise I had to make was about the weather data. Originally, I wanted to have a weather forecast for the exact start time of the event, but after trying for hours, I had to give up on this idea and instead use the current weather. The problem I had with this was about scope. Whatever method I tried, and it was probably at least five different methods, I could not get the response from the weather API request and the clicked marker/infoWindow object accessible within the same scope. If trying to assign the data returned from the JSON request to a variable, it was impossible to get the right info from the data (it was always *undefined*) because the asynchronous loading meant that the object was not fully loaded at the time it was assigned to the variable.

This meant that I had to process the data in a callback function, and getting the right infoWindow object into that same function proved impossible for me. I also tried to store the data in an HTML object and then retrieve it again within a function where I had the infoWindow object, but I could not get that to work either.

This was just a few of the methods I tried before having to use the current weather instead. The reason that I got the current weather to work is that the OpenWeatherMap API provides a possibility of retrieving this as a HTML page, and then I could just put the response into an iFrame without having to process the data any more.

Another issue I encountered was the fact that the Eventful API had a maximum page size of 100, meaning that I could only retrieve 100 events at a time. I could have chosen to iterate over the pages and retrieving all the events that way, but instead I chose to include a warning message in the interface whenever it is more than 100 results. If the application had to process several hundreds, maybe thousands, of events and place markers for all of them at once, it would probably have caused long delays and a bad user experience, so the solution I chose was good.

POSSIBLE EXTENSIONS

A possible extension for the application could be an optional side panel where the event info also appeared, instead of just placing the markers on the map. This would make it possible for users to browse events without having to click all the markers on the map.

APPENDIX

To get geolocation to work properly on all browsers, the application needs to be launched on a web server like Apache. Other than that, it should be pretty straightforward to get it working. A working copy of the application can also be found at http://norskstudent.net/eventfinder