# SIT302 Project / Analysis and Design

Table of Contents

[SIT302 Project / Analysis and Design 1](#_Toc419047520)

[2 Introduction 2](#_Toc419047521)

[2.1 Problem defined  2](#_Toc419047522)

[2.2 Solutions 3](#_Toc419047523)

[2.2.1 Native vs Web Comparison 3](#_Toc419047524)

[2.2.2 Mapping API’s Comparison 3](#_Toc419047525)

[3 Software Design 4](#_Toc419047526)

[3.1 Application functionality 4](#_Toc419047527)

[3.2 Database / Data model design 7](#_Toc419047528)

[3.2.1 Data Storage 7](#_Toc419047529)

[3.3 User Interface / Style guide 8](#_Toc419047530)

[3.3.1 User Interface 8](#_Toc419047531)

[3.3.2 Style Guide 8](#_Toc419047532)

[3.3.3 One Page Style Guide 9](#_Toc419047533)

[3.4 Data Flow / Use Case 10](#_Toc419047534)

[3.5 Planning 11](#_Toc419047535)

[3.5.1 Team Member Roles Defined 11](#_Toc419047536)

[3.6 Implementation Plan 12](#_Toc419047537)

[3.7 Testing plan  12](#_Toc419047538)

[3.8 Deployment Plans 12](#_Toc419047539)

[3.9 Maintenance plan 13](#_Toc419047540)

[4 References 13](#_Toc419047541)

# Introduction

This document will cover the analysis and design elements of the Tourist Tour Planner Application (henceforth referred to as “The Application”). The group has been tasked with developing a tourist tour planner Application that enables the user to enter a start and end destination, pick a mode of transportation and add intermediary locations as well as set the duration in days of the tour.

## Problem defined

To accurately define the problem there are two components that must be addressed the first being including all of the specified major features of the Application, the second being how to include all of these major features in a useful way to the user that is also accurately scaled to the projects deadline and the skill levels of the group members.

The Application as specified should allow the user to:

* Plan their route from point a to point b
* Selected a preferred travel mode
* Select days of a tour (referring to the duration of the tour/ trip)
* Select attractions and points of interest (referring to the user not only being able to select locations via an entered address i.e.: “16 Somerville lane”, but by an attraction name such as the “sunny lodge”.

Listed above are the specified basic mandatory features of the Application that must be included to achieve the overall conceptual vision for the Application, and thus be considered a complete application. The second component however arises in trying to include all of these application features while still being a practical task for the development team.

Some of the features listed are open to a certain degree of interpretation, and thus addressing mandatory features was one component but rather creating a design based on these features according to a scaled interpretation is also a key component in defining this task and problem.

## Solutions

As specified within the project requirements it was idea that the software be developed as an application on either the Android or IOS platform. Therefore a combination of different platforms and tools were considered to develop the software solution.

When designing a solution to this problem there were many considerations to be taken into account from required functionality, which in turn determined the platform and API’s used in developing the application.

### Native vs Web Comparison

A big influencing factor while deciding between developing a native mobile application and a web application was the abilities and skill levels of the development team.

#### Native

A native application could have only been developed on either an Android or Apple mobile device, developing for both was not an option because of the time constraints on the project. Both Android and Apple offer stable IDE’s, of the two Apple being the better option as the Android IDE is far younger and is a lot less user friendly. Both offer the ability to use API’s that would provide the needed functionality for the applications requirements. That being Apple allows use of the Google Maps API as well as the native Apple Map Kit.

#### Web Application

A web application was chosen for its ability to be accessed on both the Android and IOS platforms by making the application HTML5 compatible. This also allowed for use of any available maps API.

### Mapping API’s Comparison

The two main mapping solutions considered when designing the “Tourist Tour Planner” application were the Google Maps API which was ultimately chosen and the Microsoft Bing Framework was an option earlier on but was ruled out due to platform decisions.

#### IOS Map Kit Framework

While Apple’s Map Kit Framework is extensive and provides a wide array of features, it was ultimately ruled out as a result of the platform chosen.

The IOS Map Kit would have only been a possibility if the application was developed for the IOS platform.

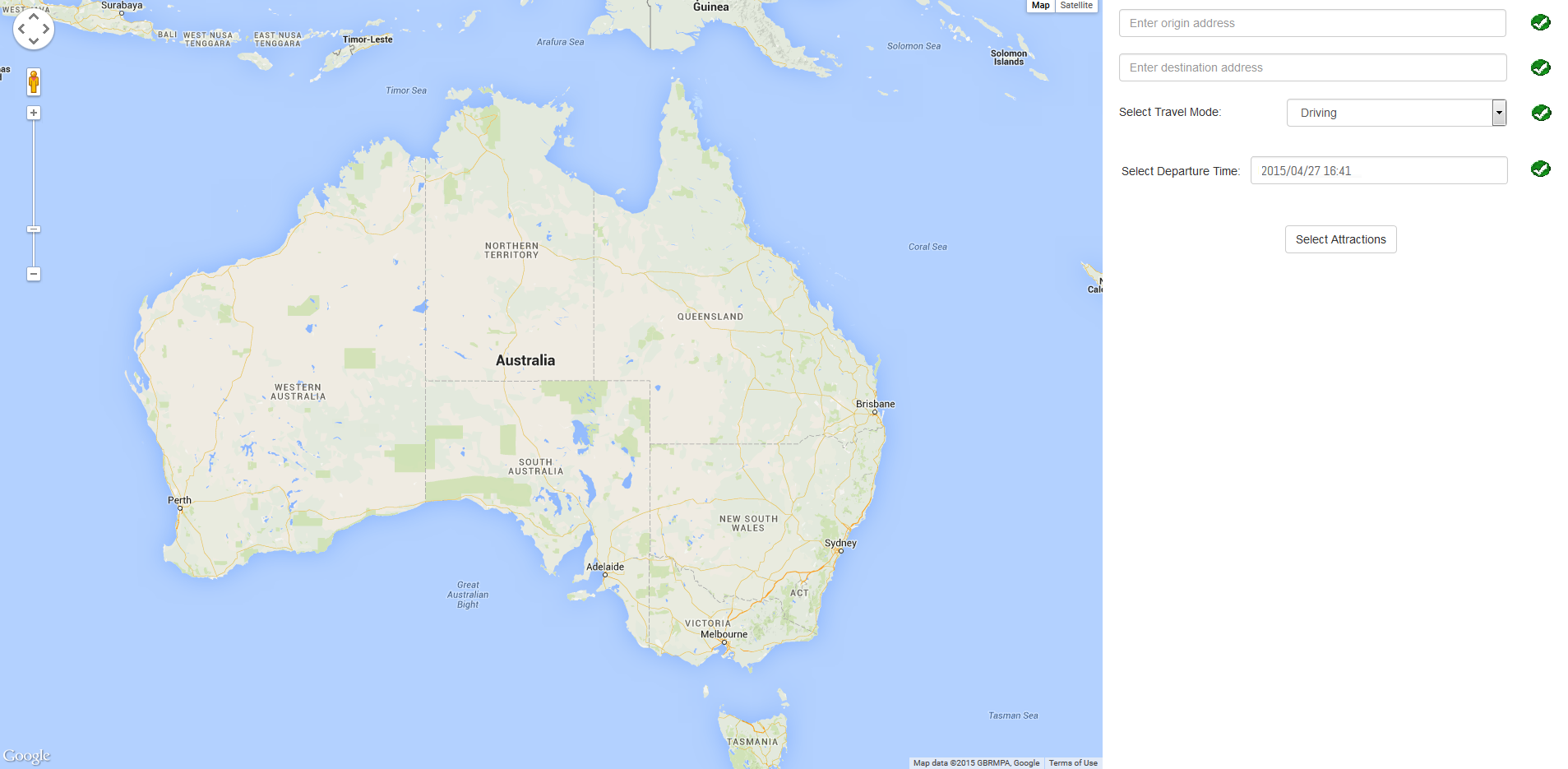
#### Microsoft Bing

Microsoft’s Bing Maps is a strong competitor and alternative to Google Maps, Bing like Google maps provides and extensive route and distance calculation functionality, as well as a bird’s eye view (a feature not offered by Google Maps. However street view as well as the additional bird’s eye view are not available to free users and a paid account must be registered to access these features.

# Software Design

## Application functionality

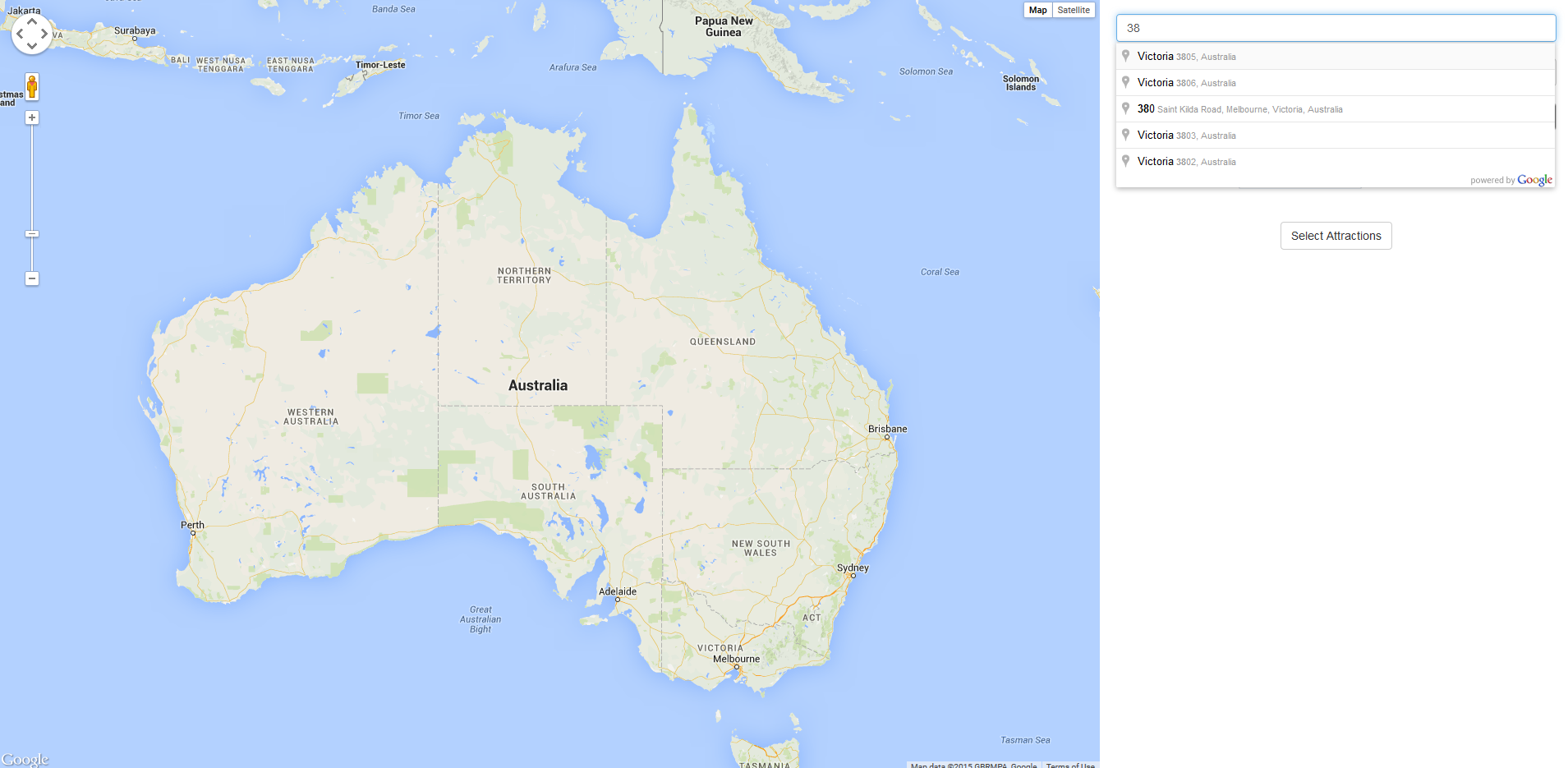
The Application’s functionality is broken up into pop-up windows, as a means to not clutter the screen with too many buttons and features. When the user first opens the Application on a mobile device or a web browser the main screen is displayed.



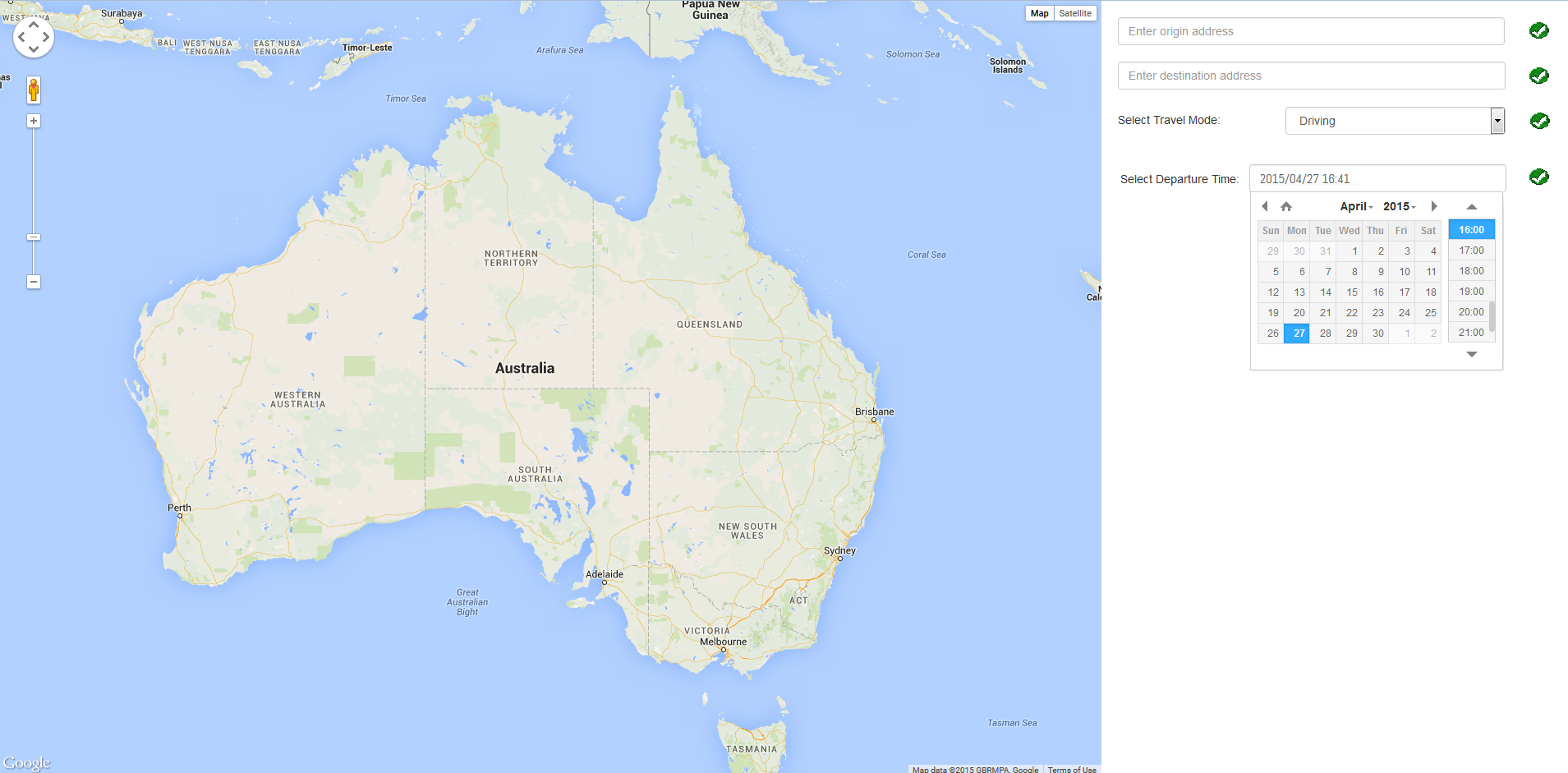
The Application’s main screen contains two address boxes, in which the user can enter their trips starting location and their destination location. The screen also contains a drop down box which the user select there mode of transportation as well as an additional text box that allows the user to select their trips departure time. Once input is made into one of the mentioned fields it is immediately validated on the client side and a validated symbol is displayed to the right of the input field.

The main screen also contains a select attractions button which when clicked displays the select attractions screen. The select attractions button only becomes functional after all of the necessary inputs have been made.

When a user begins entering an address the Google Maps Autocomplete feature displays a drop down box which is populated with suggestions for the location the user might be trying to enter.



When the user selects the departure time input box they are immediately presented with a pop-in drop down date picker.



After all the necessary input have been made and the select attractions button has been clicked the user is presented with the following window which pops-over the main screen.

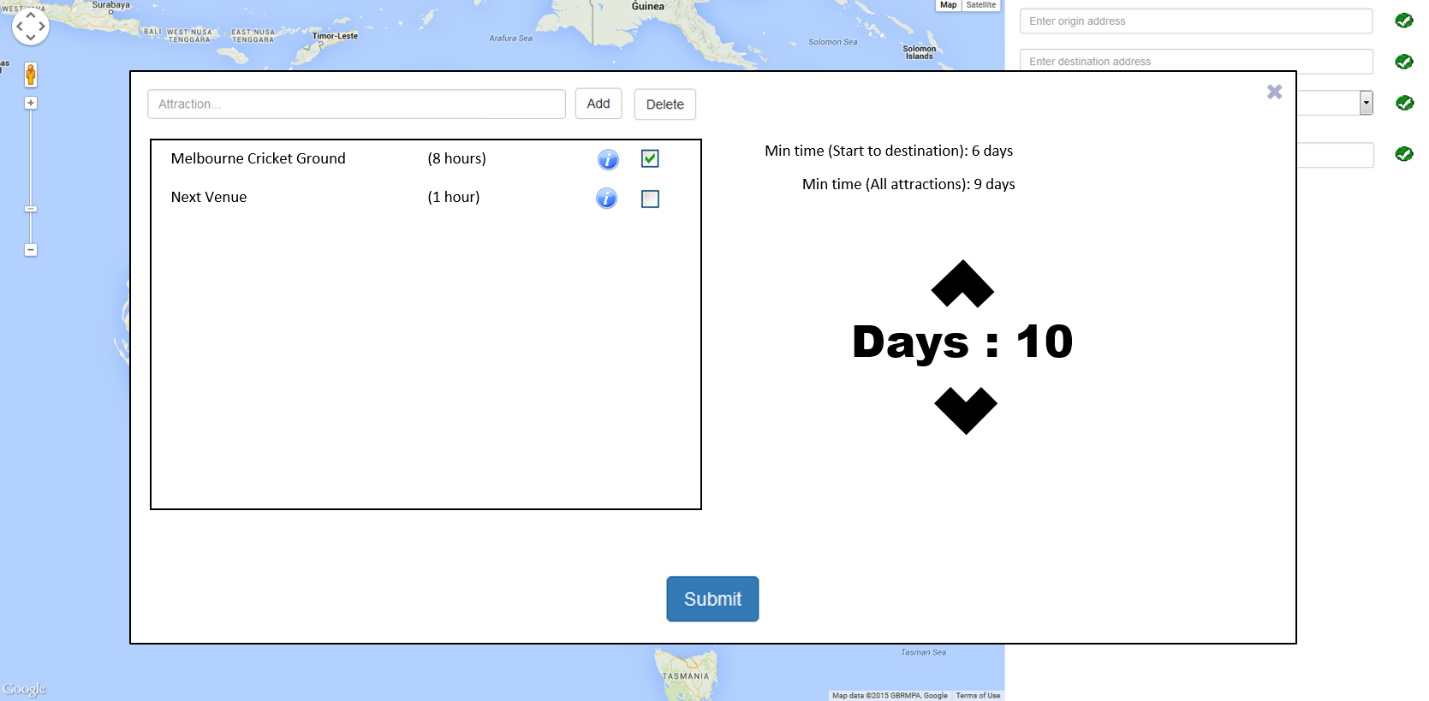
The screen contains an attractions input box which like the input and destination input boxes has a Google Maps Autocomplete drop down box which attempts to guess and suggest what the user is entering, the suggestions however are filtered to only suggest attractions.

Once an attraction has been selected it can be added to the list of attractions which is displayed in a list format below the attractions search box. Attractions are specific to the order they are entered in and can only be altered by being deleted and re-added.

Once an attraction has been added it is displayed in the list by its name, the amount of time it adds to the intended trip, and information button and a checkbox.

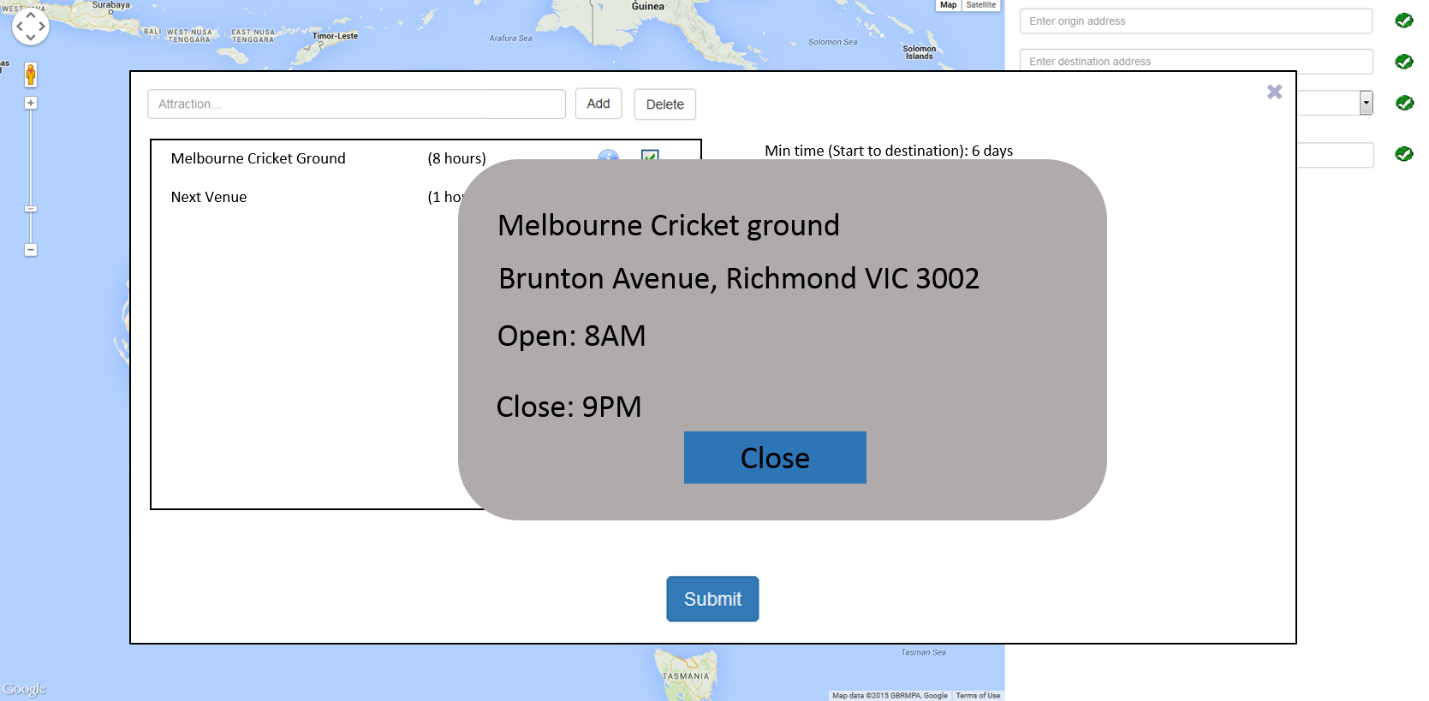
If the information button is clicked a popup window is displayed that contains information about that specific attraction.

An attractions check box is a means for the user to delete an attraction, checking an attractions checkbox and clicking the delete button removes the attraction from the list.



The window all

If the information button is clicked a popup window is displayed over the two existing windows which specifies the name of the location, the address and the opening and closing times.



## Database / Data model design

### Data Storage

Data for the Application will be stored in cookies within the web browsers storage framework, which gives us around 4KB of space initially[[1]](#footnote-1). The modern browsers that we support also provide extra storage to applications, a minimum of 10MB is provided by the browser for data storage.

Our storage implementation will be as follows:

* Locations
  + Origin
  + Destination
  + Points of Interest + Opening times
  + Location order
* Times
  + Trip time
  + Segment time
* Travel type
  + Driving
  + Cycling
  + Walking
  + Public Transport

This data may also be stored in the URL, allowing users to share travel information just by copying the URL and sending it to another device.

## User Interface / Style guide

### User Interface

Our Application presents a single unified user interface to all devices, automatically resizing to almost all device sizes. Supported device widths include from 320 pixels to 1366 pixels. Wider screens will work yet won’t be supported, and narrower screens will not be supported.

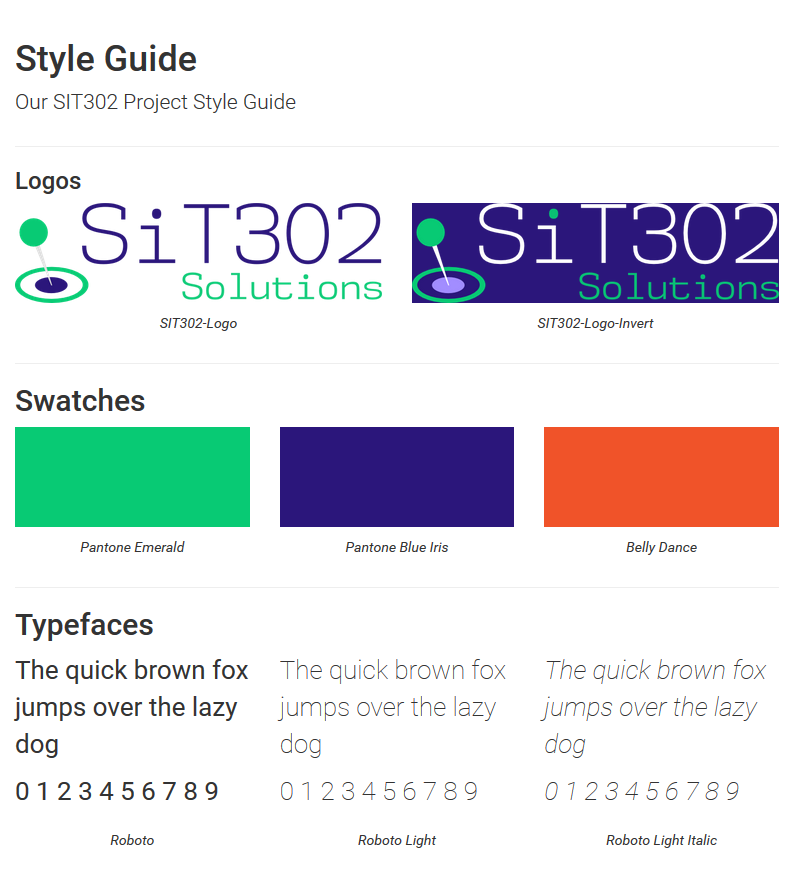
On desktop devices we will be taking advantage of the wider screens by showing a multiple column design to users, and if the device is mobile, or the browser is resized in the case of users wishing to multitask, the Application will present a single column layout with a more focused set of possible interactions.

### Style Guide

We have developed a consistent styling for the Application, which should enhance usability and improve our brand. A consistent visual direction helps to create a continuous and professional Application, with great cohesion within Application segments.[[2]](#footnote-2)

Our styling begins with a choice for a typeface or typefaces to be used throughout the Application, as we are already using Googles mapping API, we have decided to look at typefaces that they have developed. Droid Sans is cited as a popular and mobile friendly humanist typeface.[[3]](#footnote-3) The successor to Droid, Roboto, looks great on many devices and can be very easily incorporated into the project through Google’s Web Font repository and HTML5’s online font support. A sample of Roboto is included in our one page style guide.

### One Page Style Guide



## Data Flow / Use Case

The data flow for this application is fairly simple, the application loads assets from the internet, then starts accepting user input. After the selections have been made, the route is calculated, and the application returns to the main map to display the automatically planned route.



## Planning

### Team Member Roles



*Miley Zhang*  
Documentation, Record Keeping, Bug Testing

*Brian Ma*  
Documentation, Quality Assurance, Bug Testing

*Iana Bardash*  
Interface Programmer, Application Debugging

*Matt Sienkiewicz*  
API Programmer, Client Liaison, App Debugging

*George Paton*  
UI/UX Design, Shipping Leader, UX Debugging

## Implementation Plan

The Application will be implemented by further developing our prototype to cover all requirements. During this phase and after the Application is feature complete, the UI and UX will be developed to improve usability and the core experience. We have plotted our time plan for implementation and the team members involved in each process in the Gantt chart from our initial report.

## Testing plan

Our team will be testing the Application throughout development, to catch any errors that may pop up, whether by incorrect inputs or poor network conditions. Our testing conditions are as follows:

* Network conditions
  + Local and functioning network as control
  + Simulated poor connection (Bad Wi-Fi, Bad Cellular, Lost Connection)
* Unexpected inputs
  + Empty fields (should always attempt to suggest action)
  + Unexpected control characters or unsupported Unicode
* Differing browsers
  + Targeting iOS Safari and Google Chrome, and expanding to other popular browsers
  + Ensuring functionality is the same in all supported browsers

Our supported browsers include:

* iOS Safari 8+
* Mac Safari 6+
* Google Chrome Stable
* Internet Explorer 10+
* Mozilla Firefox 14+

## Deployment Plans

The Application can run standalone from a file, and requires no pre-processor. However, for the sake of ease of use, we will be serving the HTML through a simple HTTP webserver. GitHub will provide hosting of the files and can provide html straight to the user through a developer defined domain.

The license for the Application is the permissive MIT License.

The Application will be deployed at tour.arpeggi.at

## Maintenance plan

The Application will have a feedback link provided to the user, and any errors that are caught will be automatically sent to a collection email address, and sorted by error code.

Serving of application pages will be handled by the server application. Load balancing and other technologies related to the serving of web pages will be handled by the host, which in this case is Github.com

Any modifications to the code after release will be restricted to bug fixes and highly desired yet small changes requested by both users and the client. Desirability is at the discretion of the client. These modifications will be administered through the version control system to ensure any errors can be traced and fixed promptly.

At this point in time, a retirement timeframe has not been established, or yet identified as necessary.

# References

IETF.org, 'IETF Organisation - State Management Mechanism'. N.P., 1997. Web. 10 May 2015.

Neville, Kat. 'Designing Style Guidelines for Brands and Websites – Smashing Magazine'. *Smashing Magazine*. N.P., 2010. Web. 10 May 2015.

Tested, 'What's The Best Typeface For Smartphone Interfaces? - Tested.Com'. N.P., 2011. Web. 10 May 2015.

1. (IETF Organisation - State Management Mechanism', 1997) [↑](#footnote-ref-1)
2. ('Designing Style Guidelines For Brands And Websites’, 2010) [↑](#footnote-ref-2)
3. ('What's The Best Typeface For Smartphone Interfaces?’, 2011) [↑](#footnote-ref-3)