



# DYNAMIC GENERATION OF PUBLIC TRANSPORT INFORMATION

FINAL REPORT

**John William Olegario**

C1429814 – CM3203 One Semester Individual Project - 40  
School of Computer Science & Informatics

*Supervised by:* Dr. Martin Caminada

*Moderated by:* Dr. Federico Cerutti

11 May 2018



# Abstract

TARA is a travel companion mobile application one that aims to provide a dynamic generation of public transport information to travelling tourists. This report details the essential functionalities of TARA and the approach taken to provide an end-user solution and future development plans. TARA's features were explored based on the existing technologies which provided a learning platform during TARA's development on how different technologies can be utilised to provide an effective approach.

Considering the main issues, independent tourists face when travelling on different countries when finding immediate public transport information, TARA's development evolved by closely identifying how the application behaves to its end-users. Looking at existing available technology such as the GPS (Global Positioning System), which can be used to return the actual location of the user through their mobile device. Additionally, TARA utilises Google's powerful query API's (Application Programming Interface) such as Google *places*, Google *maps*, and Google *transit* to identify local transport information.

An overall view of the following current technologies and best practise approach have lead to finding practical solutions to TARA's functionalities. TARA is a travel companion application one that provides directions, and reliable transit information to its users when travelling to their desired location.

## **Acknowledgements**

Firstly, I would like to thank Dr Martin Caminada for sharing a moment of his time to provide support, guidance and feedback throughout this project. It was an absolute pleasure to work with someone who believed in my abilities when completing this project. My gratitude also goes to the School of Computer Science and Informatics at Cardiff University for helping me to bolster the skills that I need to get through my career life afterwards.

I would like to extend my deepest gratitude to my family, especially to my parents Leonora Asibal Olegario and William Buenaventura Olegario, who provided me with all their love and support throughout my years of study as I would have not made it without them. I am also extending my personal gratitude to my special friend Ana Landicho throughout my degree studies who became my companion and inspiration to succeed.

I was never the smartest person throughout my studies, and disappointments have happened. Through perseverance, willingness to succeed and because I have never given up I was able to get throughout my studies. I am ready to take more challenges in the next chapter of my life.

# Table of Contents

Acknowledgements .....	2
Table of Figures .....	5
[1] Introduction .....	6
[2] Background.....	6
2.1 Google Maps and React Native-maps library .....	7
2.2 Google Maps and Google places search API.....	7
2.3 Google Directions API and transit timetable .....	7
2.4 Google Webservices API Justification.....	8
[3] Specification, Design, and Approach .....	8
3.1 Back-end development.....	8
3.1.1 User accounts and PostgreSQL database integration .....	9
User’s schema .....	9
3.1.2 GPS and User location via Google Geolocation API .....	9
3.1.3 User Location and Google Places API (location information) .....	10
3.1.4 Google Directions and Transit information integration.....	10
3.2 Front-end development.....	11
3.2.2 Main Page and the Current location state.....	12
3.2.3 Main Page and nearest bus station state .....	13
3.2.4 Location Page and directions state.....	13
3.3 Changes to the initial approach .....	14
3.3.1 NoSQL MongoDB or Relational Database (PostgreSQL).....	14
3.3.2 Searching Directions.....	15
3.3.3 Agile vs Waterfall development cycle .....	15
3.3.4 React native or native android development .....	16
[4] Deliverables and approach implementation .....	17
4.1 User accounts and authentication development implementation .....	17
4.2 User current location functionality .....	19
4.3 Nearest bus station markers .....	20
4.4 Finding directions and timetable .....	22
4.4.1 Finding Directions.....	22
4.4.2 Bus stop location.....	23
4.4.3 Bus station Timetable .....	24
[5] Initial Results, Testing and Evaluation .....	26
5.1 TARA Agile Test Result Record 1 .....	26
5.2 Google Webservices integration TARA Agile Test Result Record 2 .....	28
5.3 Overall Evaluation of the Testing Process .....	32
5.3.1 Sprint no-1 User log in .....	32
5.3.2 Sprint no -2 Google API call and display result into UI.....	32
[6] Future Work .....	33
6.1 Smart suggestions and machine learning.....	33
6.2 Accelerometer and Vehicle positions .....	34
6.2.1 Google Transit API and Vehicle Positions feed update functionality .....	34
6.3 User login interface change .....	34
6.3.1 User Registration page.....	35
6.3.2 Users persistent login .....	35
6.4 TARA iOS deployment.....	35
6.5 Several modes of possible public transport travel in Map View .....	35
6.6 Nearest restaurants, toilets, hotels? .....	35
6.7 User location automation update .....	35
6.8 Nearest bus station tap and reveals schedule .....	35
6.9 Google Direction API unavailable transit coverage.....	36
6.9.1 Locations Collection Database .....	37
User Connectivity.....	38

[7] Conclusions..... 39

[8] Reflections on Learning ..... 39

[8] Bibliography/References..... 41

# Table of Figures

Figure 1: Initial TARA Flow Chart .....	8
Figure 2: UML User Login .....	9
Figure 3: Latitude and Longitude .....	10
Figure 4: Nearest bus stop search.....	10
Figure 5: User Location, transit and directions integration.....	10
Figure 6: Basic Flow chart of TARA.....	11
Figure 7: Login State Transition diagram.....	12
Figure 8: Main and current location State Transition Diagram .....	12
Figure 9: Main Page and nearest stop state transition diagram .....	13
Figure 10: Locations stage and directions state transition diagram .....	13
Figure 12: Single and Double loop diagram.....	14
Figure 13: Django User model.....	17
Figure 14: User serialiser code to API HTML view.....	18
Figure 15: React native fetch call and login UI (User Interface) .....	18
Figure 16: User location fetch call and UI result .....	19
Figure 17: Map view of user location and UI .....	20
Figure 18: Nearest bus stations fetch call and UI.....	21
Figure 19: Polyline fetch call and UI view.....	22
Figure 20: Bus station fetch call and UI .....	23
Figure 21: Fetch timetable and arrival stop and UI.....	24
Figure 22: Users post state transition diagram .....	36
Figure 23: Location database integration .....	37

## [1] Introduction

*Where do I catch the bus? Nearest terminal? Where do I stop to reach my destination?* Travelling to a location for many tourists or travelling enthusiast requires immediate access to reliable public transport information. TARA is a travel companion application for android mobile devices one that provides transport information to its users that ensures correct and reliable information. Existing transit mobile application is now in existence throughout the world providing bus timetable, and travel routes to users, enabling them to know transit information within their vicinity. This report details the distinctions of why TARA is different from other applications that are currently out in the market.

TARA 's target audiences are tourist and travelling enthusiast. Typically, users download transit applications which are specific to their local hometown. TARA's core functionality provides transit information anywhere and where ever in the world. Taking advantage of the enormous knowledge database of Google's web services API, TARA renders the response and deliver the relevant information in aesthetically pleasing view and one that avoids information overload.

In Cardiff Wales, an application called Traveline Cymru exists which covers specific locations within the South-Wales area. In Derbyshire, two independent companies run services on particular regions such as the Arriva Bus which comprises the main Derby city centre and the Trent Barton bus company which includes most of the outskirts outside the city. When tourists arrive on a new location, to obtain crucial transit information, most companies will have a distinct mobile application which only reveals the services specific to the area that is covered by their transit services. According to Barents Tourism (2017), as the internet and social media play an essential role in our society, tourism destinations need to work harder to enable travellers to access reliable and accurate information by using information technology. TARA provides an immediate solution to this, and that is to have a single application that will work anywhere and everywhere, by Google's webservices API coverage. Users will be saving their mobile memory as they would often require downloading many transit applications due to many different travel applications on specific locations.

TARA's user interaction design has a distinctive design from other applications, as the majority requires two inputs from the user. Uniquely, only one input from the users is necessary for TARA when finding transit information to their desired destination. As soon a user logs in, TARA already knows the user location which is used as a parameter to fetch the vast knowledge of Google's webservices API to provide the necessary information. The main aim when designing TARA's user interaction is to decrease the amount of user-input required.

Since TARA fetches information from Google, this means that all this information is also available in Google Maps. Google maps require additional inputs from the user to get to the destination that they need to be. For instance, as soon as the user uses Google Maps, other information will also be revealed such as nearest restaurants, bars or shops and this causes an information overload to travellers that only requires transit information.

In addition, Google map users will have to input the information themselves or apply extra arguments in their query, for example, *nearest bus stop*. In contrast to TARA which is already filtered and hardcoded in the back-end to bring relevant transit information, one that returns the nearest bus stations to its users. Hence, TARA focuses on transit information which includes bus travel times offering a more convenient user interaction to its users.

## [2] Background

Many western countries including the United Kingdom, United States, Netherlands or Luxembourg have state of the art and modern transport information systems. Transportation information is available to the public as transit companies released transit data to many third-party technology companies. Companies like Google aggregate this data and store them on their web servers. Petr Gazarov (2016) states at some point or another; most large companies have built APIs for their customers, or internal use. TARA will be taking advantage of the benefits of this technology and will be utilising Google's powerful APIs to attain valuable information and to be made viewable to its end users. TARA is built to provide transport information, eliminating other irrelevant details.

TARA will make use of existing technologies such as GPS which are available on modern mobile phones that determine the device location. By combining the use of various Google APIs, this provides essential information which will be rendered by the application; discussed further in the approach and implementation section of this report. Furthermore, coverage issues are one of the leading concerns that have been indentified when planning TARA's functionalities as Google's API may not provide full coverage about a country or a particular location to date. Nevertheless, Google has

already collected vast information about transportation information data in many places. More information about the utilisation of various Google API is in sections 3 and 4.

TARA required a considerable amount of human computer interaction planning; essentially from the beginning to know how the application will behave once it is released. It was important for the application to be fast and responsive one that dynamically reveals information to the user when tapping a button within the application. TARA also required an effective back-end system that holds user-accounts, which enables users to use credentials to login into the application. In general, TARA's development has several phases; one is to provide a full operational back end system, another is to provide aesthetically view to users in the front end. TARA's initial deployment will be for android and integrated to several back end databases.

The target audience for TARA are travelling tourists, and often tourists will have to disturb the locals in the country asking for information, although this is not usually a massive issue, one would prefer to obtain information on self directly. TARA is a filipino word for *let us go*, and it aims to provide information to travellers, to ensure they know where to go.

## 2.1 Google Maps and React Native-maps library

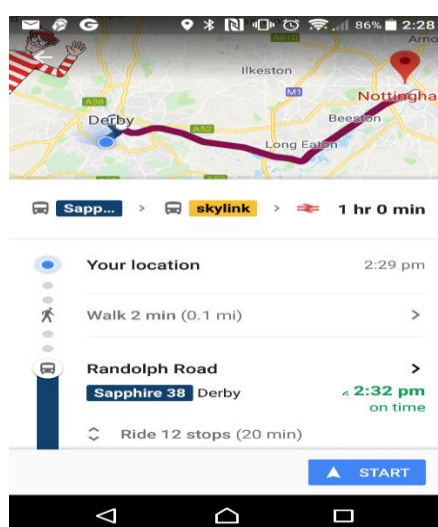
React-Native is a popular JavaScript framework, a cousin of a well-known framework called React which deals with website development. React-native is specific to the development of mobile applications and chosen as the front-end base of TARA. React Native maps is a library from React-native and integrated into Google Maps API. The whole purpose of integrating Google Maps API along with the Google Places API is to provide information about a specific location. A notable feature of Google places called Location Awareness as stated by Google Developers (2018) that the power of mobile devices can be utilised to give users contextual information whenever or wherever they are.

## 2.2 Google Maps and Google places search API

Every android mobile device has a GPS Global Positioning System which returns the location of a specific individual. TARA works by utilising this technology, and it finds users location via the latitude and longitude, this is used as a query that triggers the functions within TARA. The purpose is to return information about a particular location as TARA is a travel companion application. Information of interest is nearest bus stations or terminal; TARA will fetch the data about nearest bus stations from API and renders this information to the user using the map view from the application. TARA will benefit with this functionality as an essential requirement as Google Places already have the massive collection of transit data on specific locations including the nearest bus stations.

## 2.3 Google Directions API and transit timetable

Following the deliverance of nearby bus stations regarding around the user's location, TARA will also benefit from React native maps directions which will provide directions to the users on how to get from one direction to the other. Integrating React native maps direction with Google Directions API gives complete and detailed direction information.



Google Maps Directions API

Apart from providing transport information about the area such as the nearest bus stop, by using the extensive coverage of Google Maps Directions API, TARA will able to provide directions between two locations based on the query search of the user including bus timetables. Setting the default location of the user from its current location, are handled by the geocoder library, the user is then free to use to query the application of where to go. Many public transit agencies publish their transport information to Google's webservices, and is available in Google Directions; it contains worldwide coverage of transport information including timetables.



On the other hand, not all countries are covered by the Google’s webservices yet. According to Google Developers (2018) some data such as public transit routes, are available at a city level but do not appear in the list of countries that are covered by Google Directions.

2.4 Google Webservices API Justification

There are currently a couple of Transit API in existence which is alternative to Google’s Directions and Transit API. One of which is called the Transport API which only available in the UK, it also provides live departures and arrivals, timetables. According to Transport API (2018), it is the most comprehensive transport data platform in the UK. However, this does not fully suffice the intended requirements for TARA, as TARA is required to work in many countries around the world as possible. Further research leads to the discovery of HERE. HERE is a robust set of developer tools that enables applications to make maps and locations technology as stated in Here (2018). HERE became a formidable contender with coverage to many cities worldwide. However, it does not currently have the broad coverage in comparison to the enormous collection of Google’s Transit data which covers many cities, counties, and provinces around the world. TARA’s core functionality is to work in many areas including provinces and regions within a country and Google’s Transit coverage has the scope for TARA’s requirements. Google Transit does not have overall coverage worldwide. However, it is enough to satisfy most users until Google has expanded its coverage.

[3] Specification, Design, and Approach

TARA’s development consists of two stages, the front end, and back end development. Full stack development of TARA required the use of modern full-stack programming tools for developing the back-end webserver and the latest tools for improving the best and responsive user interface in the front end. This stage of the report will focus on the individual technologies, which are essential throughout the development process, as well as the approaches that were taken to achieve the behaviour and interaction that are intended for TARA’s end-users. Not all of the planned functionalities of TARA will be achievable on the first iterations of its development, and this is segregated in to sprints to identify the deliverables over the last two months and months to come.

3.1 Back-end development

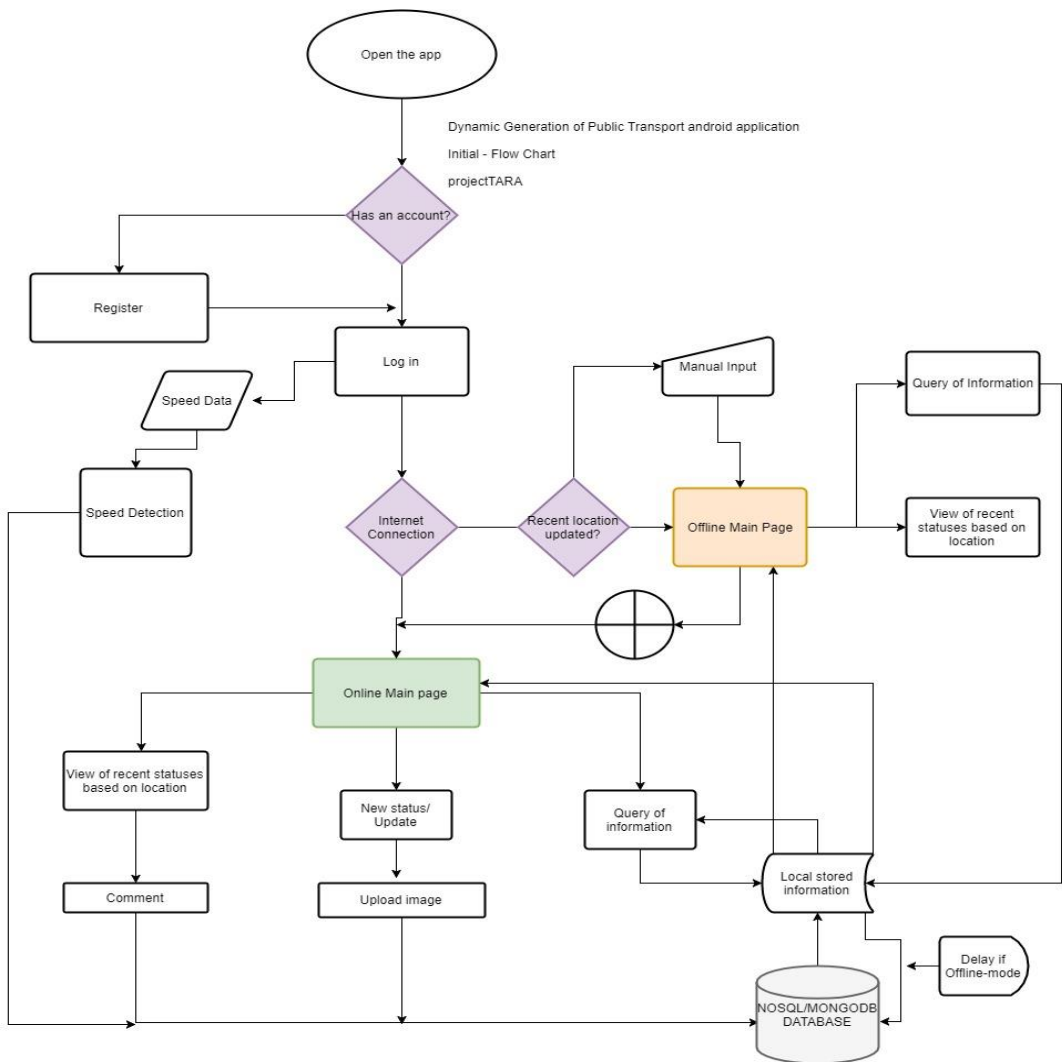


Figure 1: Initial TARA Flow Chart

This flow chart was the initial plan for TARA’s back and front-end interaction for data retrieval. Significant changes have occurred since the initial planning which will be discussed further in section 3.3. This section focuses on the retained interactions which will be in detail in this section.

Django is the chosen back end technology, and it is a web framework for the Python programming language. TARA required fast-paced development goals as it was needed to bring initial concepts into completion as quickly as possible and Django has a reputation for this. TARA requires users to register for access, therefore user account databases that will hold email address, name and password was essential. The justification for having user accounts is part of a future development that is mentioned in section 6. The following sections will explain the approaches that were taken to build the back-end functionalities of TARA.

### 3.1.1 User accounts and PostgreSQL database integration

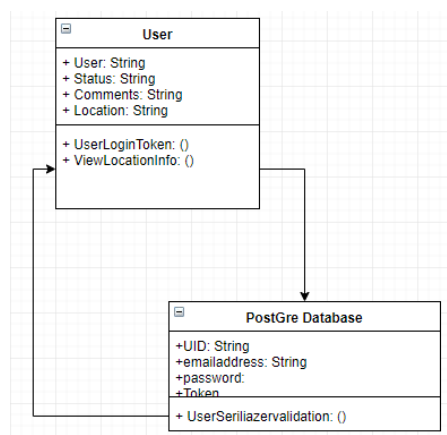
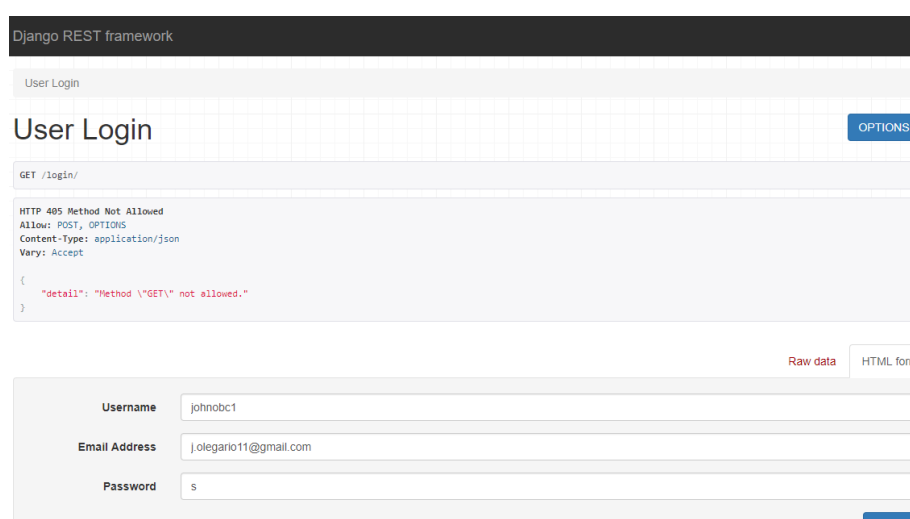


Figure 2: UML User Login

Database user model schema consists of User ID, Email Address, Password, and Token. For a user to be able to login, a token is generated from the back end. This token will be the argument required in the front end to verify if the user is a valid user of TARA. A library called Django rest framework is being utilised to automatically serialise the schema of the database which in return provides a REST API in the form of a JSON format file.



Django REST Framework, API in action

The image shows the serialiser function from Django in action. This API is open and accessible by its domain name; more information on how it was accomplished is in the implementation stage. TARA will only require the username, email address and password of any user to login.

### User's schema

The user schema consisted of the following attributes:

- Email, username, and password – this holds the user's credentials to access and use the application.
- Status – each new post of the user, will be stored in the status attribute.
- Location – the location of the user is stored here, when a user arrives at a specific location, the new location will be appended to this table. The location attribute stores all the previous places that the user has visited.
- Speed Data – Speed data is taken from the user for specific future functionality that is intended for TARA as mentioned in section 6.

### 3.1.2 GPS and User location via Google Geolocation API

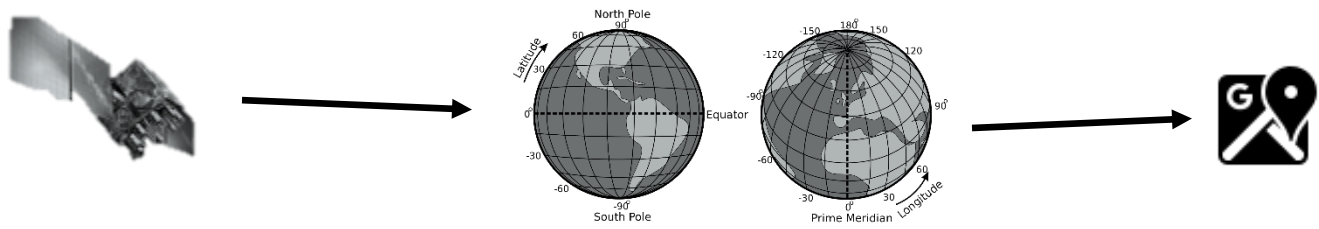


Figure 3: Latitude and Longitude

Every android mobile device has an embedded GPS technology, and determining the user location is by detection of the device's latitude and longitude value in the surface of the earth. Capturing the latitude and longitude values is the main argument that was needed in the Geolocation API to identify the user's location. Further information about the implementation is in section 4 of this report.

### 3.1.3 User Location and Google Places API (location information)

Following the return of the user's location via Google Geolocation API this became an essential requirement and the argument that is needed to query the Google Places API which was filtered to only return specific locations within the particular area such as the nearest bus stations.

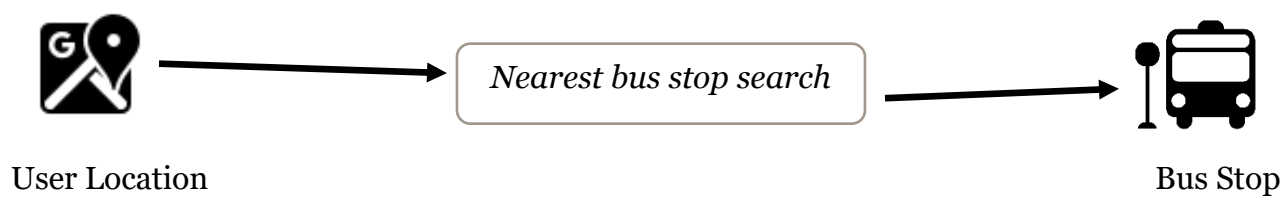


Figure 4: Nearest bus stop search

Users latitude and longitude values are not constant, and it is embedded within TARA's functionalities to continue to mount and detect the exact location of the user. Nearest API from Google will return a list of JSON objects revealing the information response of the area. A sample of this response is in the implementation section. Acquiring the latitude and longitude values from the response which in return used as a parameter to identify the location of the markers in the Map View. Map View is a React native integration of Google maps which displays the current location of the user along with the nearest bus stations. A sample of this will be in section 5.2

### 3.1.4 Google Directions and Transit information integration

Following the return of the nearest bus stations within the user's location, Google Directions API concerns the transportation information. Google Directions has comprehensive coverage of transport information, and the majority will contain bus timetables.

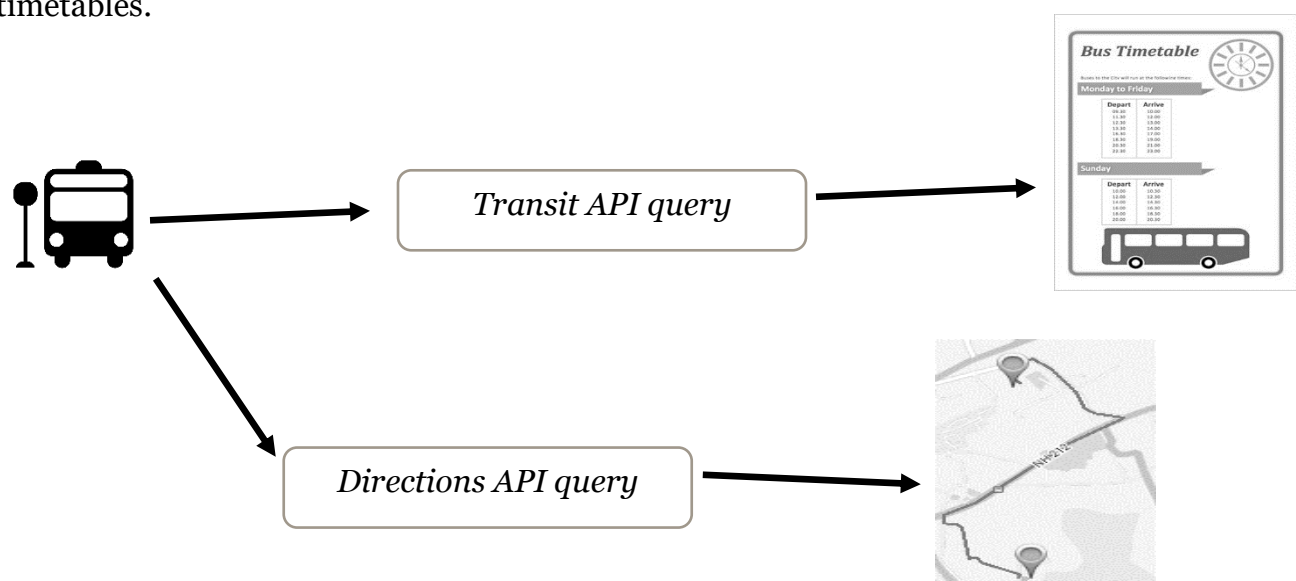


Figure 5: User Location, transit and directions integration

The timetable is displayed along with the marker in map view, revealing the user's location and line that leads to the desired destination of the user.

### 3.2 Front-end development

TARA's user-interface is powered by React-native as mentioned in section 2.1. This section of the report will cover the design and various states of the user interaction of TARA and in conjunction with the basic flow in figure 6 as it follows the intention on how the user will view the data from the Google API. TARA's user interface was an essential part of initial planning as the goal is to provide an aesthetically pleasing view and one that offers immediate information to the user.

#### UI (User Interface Interaction) TARA

Data is coming from various APIs from Google and rendered to bring an aesthetically pleasing view to its users while providing critical data to its users immediately.

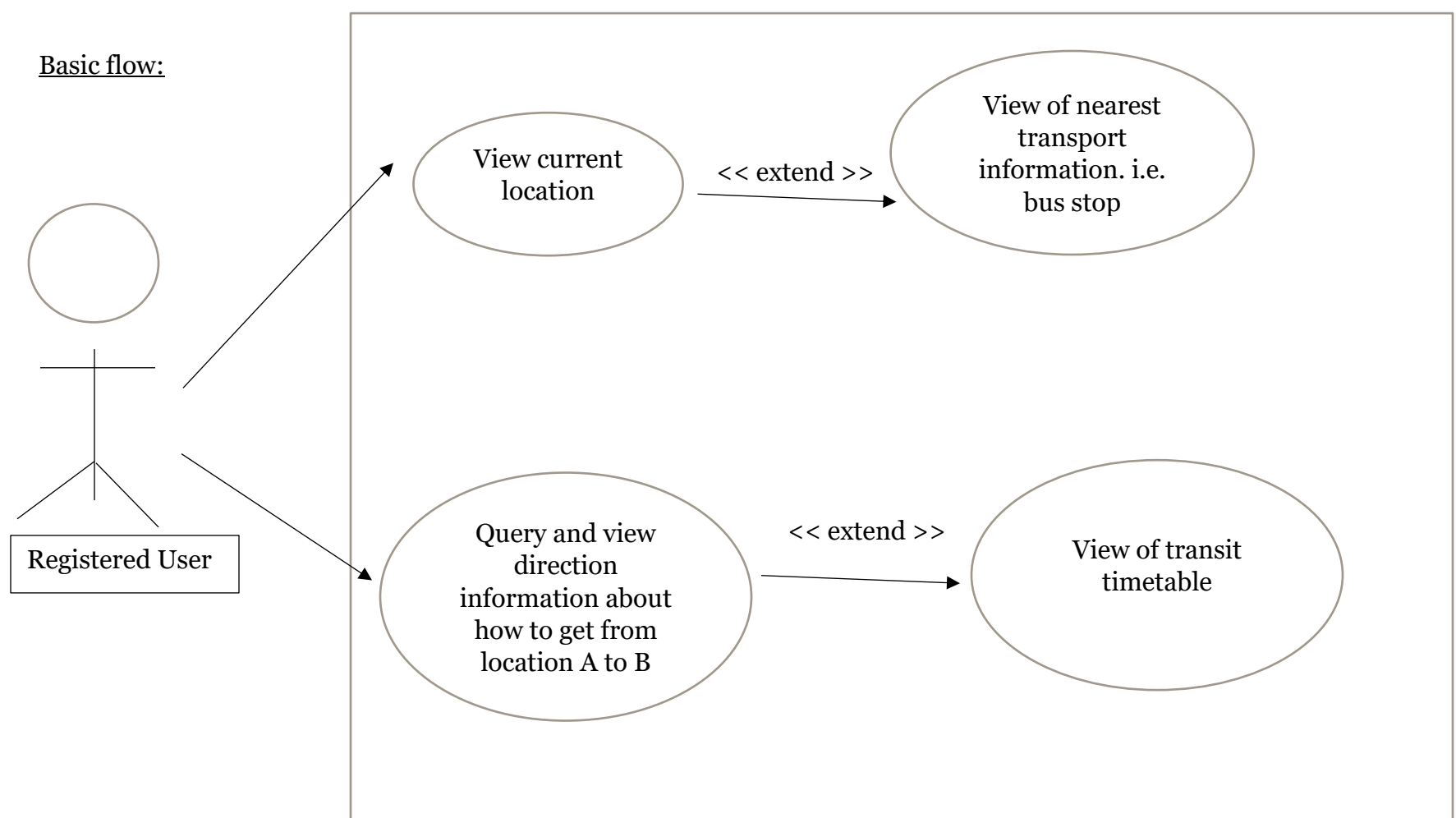


Figure 6: Basic Flow chart of TARA

#### Basic Flow Outline:

1. Registered users will be able to view their current location.

**Extension:** When users view their current location, the application triggers an event as part of its functionality that queries about travel information about Google and the location of the nearest bus terminal or bus stop.

2. Registered users will be able to query about directions on how to get to their desired destination. TARA renders information and returns direction information to the user.

**Extension:** When users query about directions, the application will supply transit timetable to the user as soon as a search query is triggered.

The basic flow outline above reveals the intended flow for users as they are using TARA, it also briefly outlines how the application responds and the views that are available to the users. More information on this in sections 3 and 4.

3.2.1 User Login state

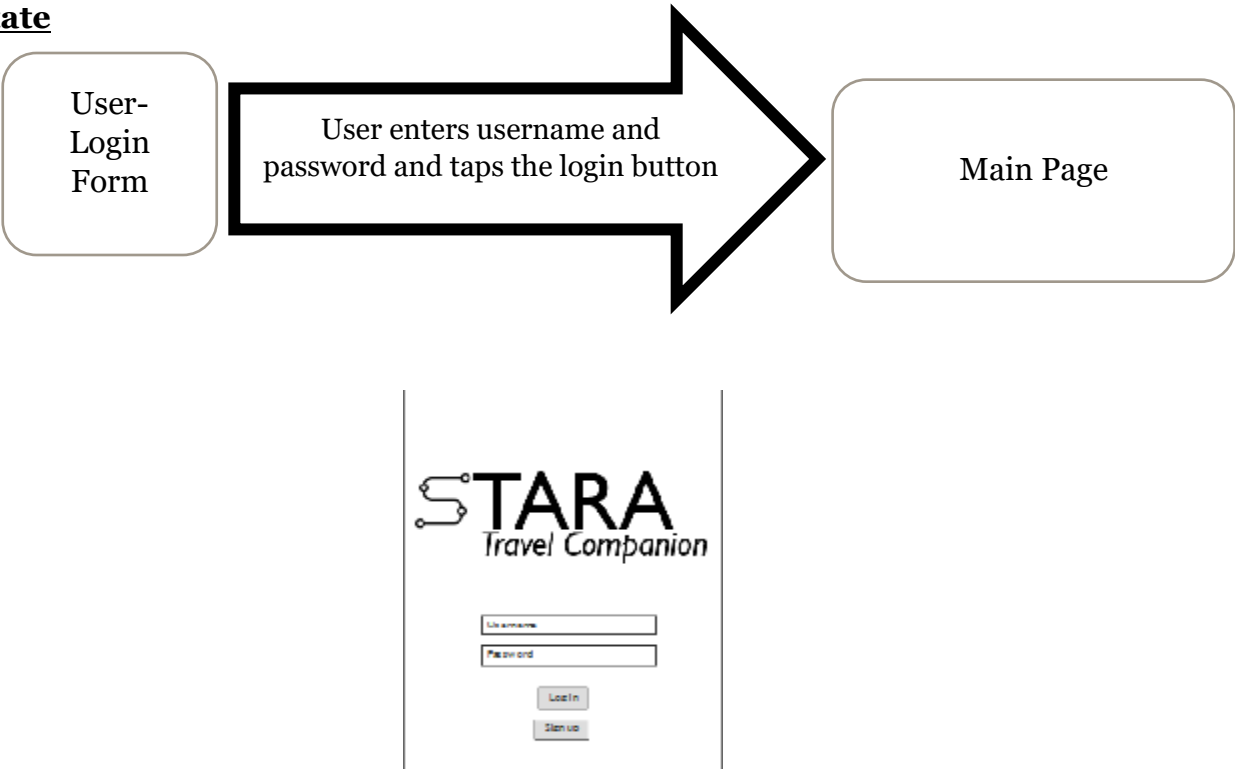


Figure 7: Login State Transition diagram

Users will be greeted with a login screen prompting them with their username and password to log in. When a user login into TARA, a validation occurs to verify a token during login ensuring a user does not exist in the back-end database. Login will not progress to the next screen if credentials are incorrect or to unregistered users. At this stage, user registration functionality in TARA is not functional; user administrator must be informed to register new members using the API. Section 6.3.1 mentions the future work regarding user registration.

3.2.2 Main Page and the Current location state

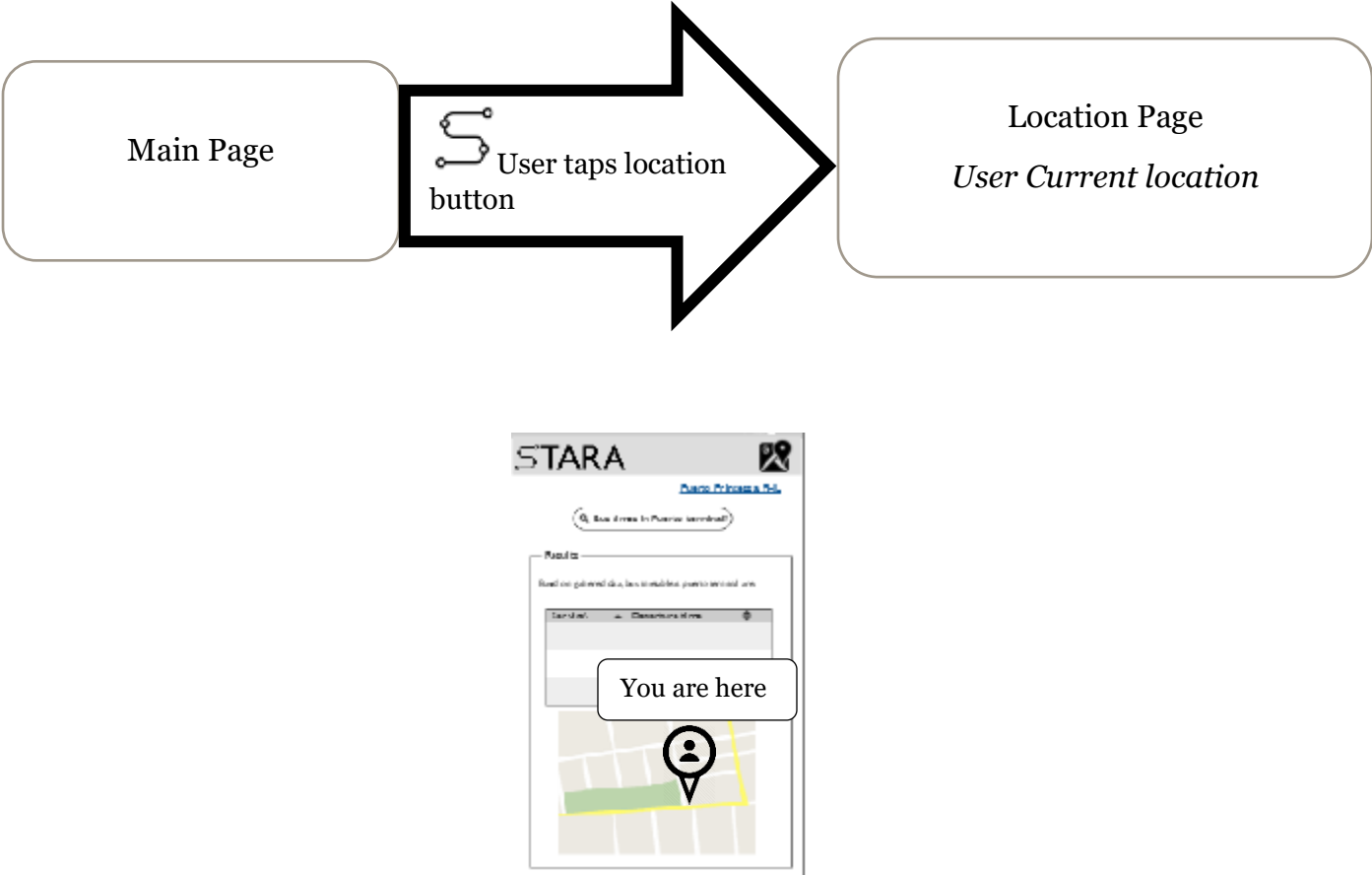



Figure 8: Main and current location State Transition Diagram

A core functionality that informs the user of their current location in map view occurs when the user taps this icon . Subsequently, triggers an event from TARA that fetches the latitude and longitude values which in return processed by Google’s geolocation API. The latitude and longitude values are already determined after a successful login by the user as it is used to set up markers inside a map view which pinpoints the actual location of the user.

**3.2.3 Main Page and nearest bus station state**

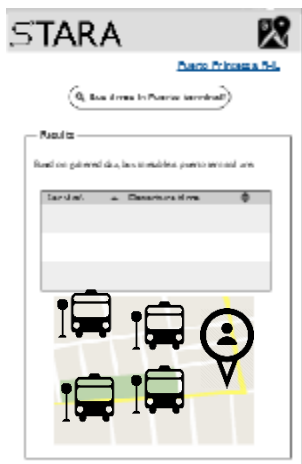
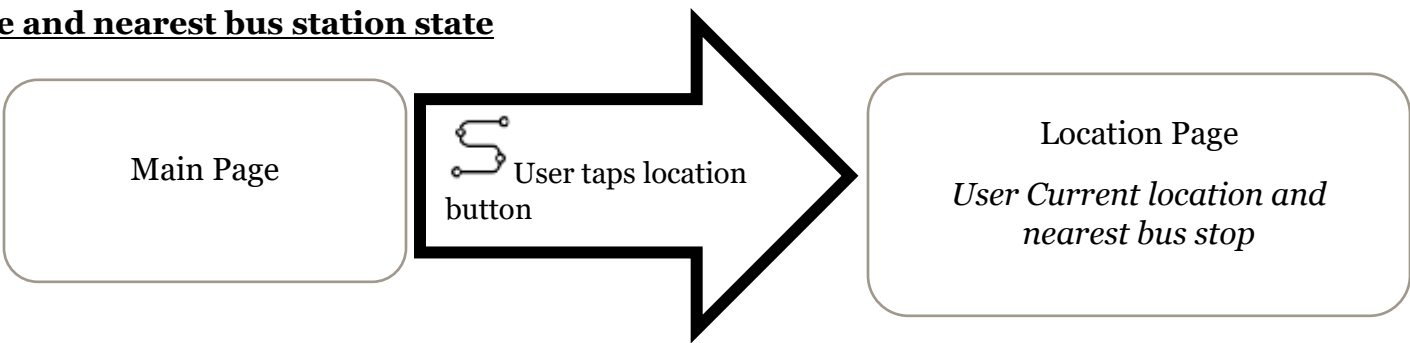


Figure 9: Main Page and nearest stop state transition diagram

In addition to the location of the user in map view, it is also a functionality of TARA to reveal the nearest bus stations in the area based on the user's location. Additional calls to Google's places nearest API that uses the current latitude and longitude values of the user as an argument to the API. The request is filtered to search the bus stations in the area, and additional markers are visible in the map view to reveal the whereabouts of the nearest station.

**3.2.4 Location Page and directions state**

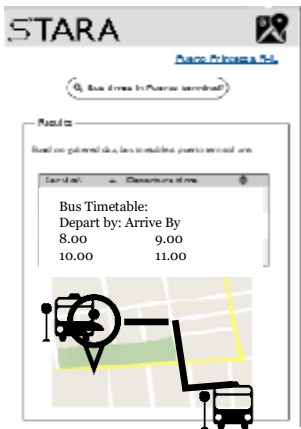
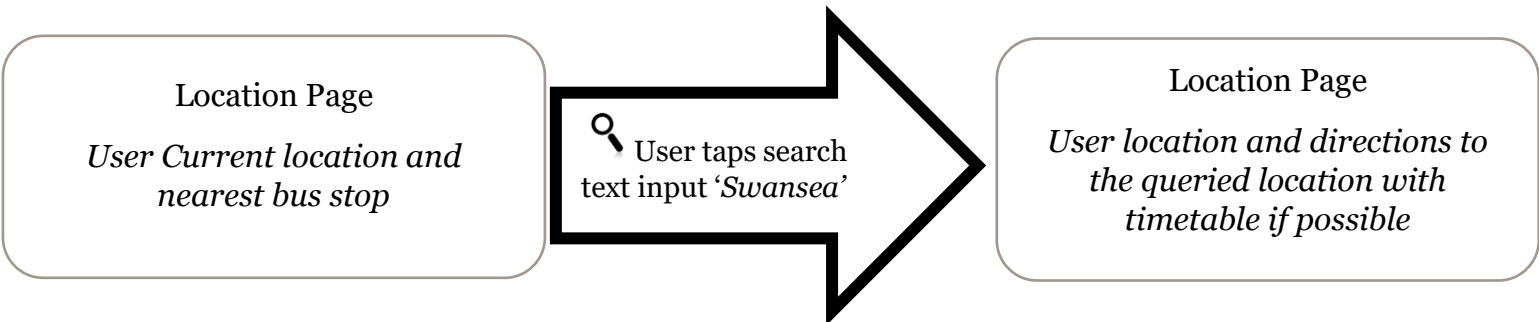


Figure 10: Locations stage and directions state transition diagram

Following on the locations page, a search text input is visible to query the point of destination showing route directions on how to get between locations, using a line. Users are no longer in need to input their starting point location as the application is programmatically filtered to take in the user location via their latitude and longitude values as their starting point location. A distinctive feature of TARA, which aids in reducing the number of user inputs that are needed for the user to get information.

In addition to providing route directions, when a user queries their desired destination, this also reveals the actual bus timetable of the particular bus station that leads to their destination. Distinctive bus icons will appear on the map which will be in the implementation section of this report.



### 3.3 Changes to the initial approach

This section of the report discusses all the relevant changes since the original planning of TARA. As development progress, necessary decisions were taken place to question the initial solutions that were planned for TARA and identify whether they are the correct solutions. These changes were essential to address the issues of the initial approach to produce the right outcome.

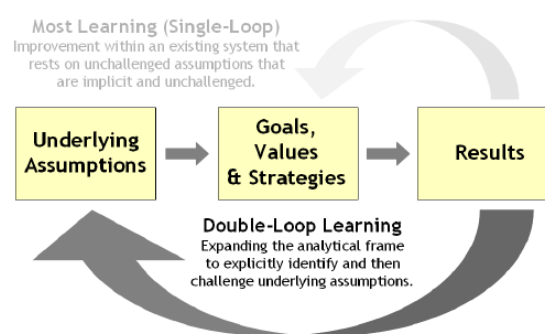


Figure 12: Single and Double loop diagram

Nevertheless, not all outcome is the finalised outcome, and future solutions of TARA are open for review whether an improvement from the current system is needed or a change from the original plan. Figure 12, reveals the iteration process of the decision making process, should a change from the original plan is required.

#### 3.3.1 NoSQL MongoDB or Relational Database (PostgreSQL)

##### Original Plan:

Initially, TARA's back end architecture consisted of a NoSQL MongoDB database as it justifies that data can be inconsistent for a short period as connectivity to Google API and MongoDB may be disconnected. According to the BASE (Basically, Availability, Soft-State, Eventual Consistency) approach. A BASE data store values availability since that is the important scale, but it does not guarantee consistency of replicated data but will eventually get updated in the future (Sasaki, 2015). About TARA, considering that connectivity may not be available at all times, it was found that inconsistent data could occur when connectivity is disconnected. Subsequently, it is not a major issue as long as there are a considerable amount of available data in the cache enough to provide transportation information to the user. Connectivity will eventually resume when a user nears a Wi-Fi area only then will the database be updated.

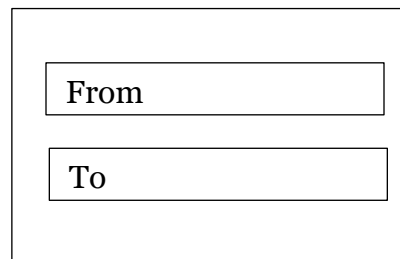
##### Updated Approach:

On this particular case, it was decided to take a single loop approach. Improvement to the current architecture was needed, and the decision was to continue using to the current BASE approach at the same time the system cannot entirely store data in MongoDB. TARA's user accounts are separated in to a relational database with the possibility to integrate two databases in Django, as referenced from the Django documentation which follows, a detailed overview in the implementation section. This change was necessary as user authentication and No-SQL approach for Django is a newly launched feature and it may take time to integrate, therefore it is chosen at this time that the user accounts will be in a relational database called PostgreSQL. In addition, TARA needed some of the features of a relational database which follows an ACID (Atomic, Consistent, Isolated, Durable) approach. User accounts required some consistency especially when users decided to update post status or update sensitive details such as passwords, ACID properties mean that once a transaction is complete, the data is consistent Sasaki (2015). No-SQL MongoDB is part of a future work for TARA mentioned in section 6, containing supplementary information that is triggered when users arrived at a location and a perfect choice for storing large collection of location data.

### 3.3.2 Searching Directions

#### Original Plan:

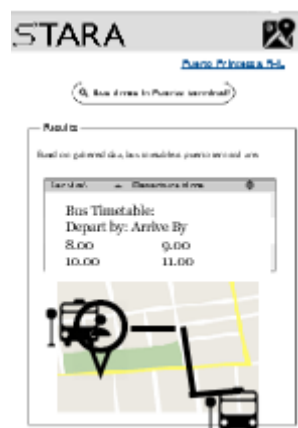
TARA's original design when a user searches for directions was initially to have a *from* and *to* text input form that asks users of their original location and destination.



A simple form with two text input fields. The top field is labeled 'From' and the bottom field is labeled 'To'. Both fields are empty and have a standard rectangular border.

A common functionality to many travel applications is when asking two inputs as users are more familiar with this interface design. The goal is to have fewer taps and input from the user interface, and this design does not conform to the required functionality which lessens the user input.

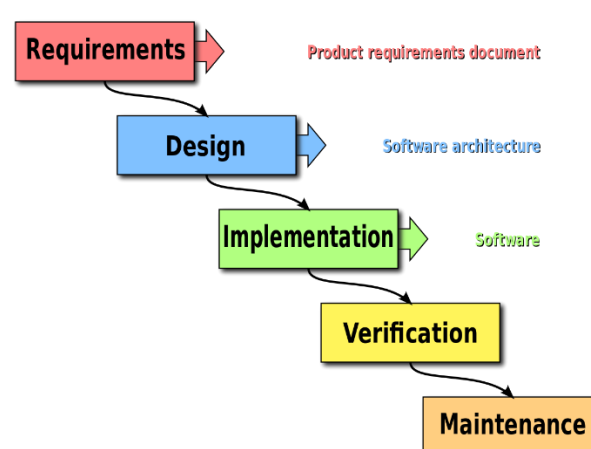
#### Updated Approach:



As mentioned in section 3.2.4, finding a destination when using TARA will now only require one input from the user, and that is the point of destination. TARA uses latitude and longitude data to return the user's precise location when considering the starting point location. Additionally, critical information is available right away to the users as the map view will contain their current location nearest bus stop in the area.

### 3.3.3 Agile vs Waterfall development cycle

#### Original plan

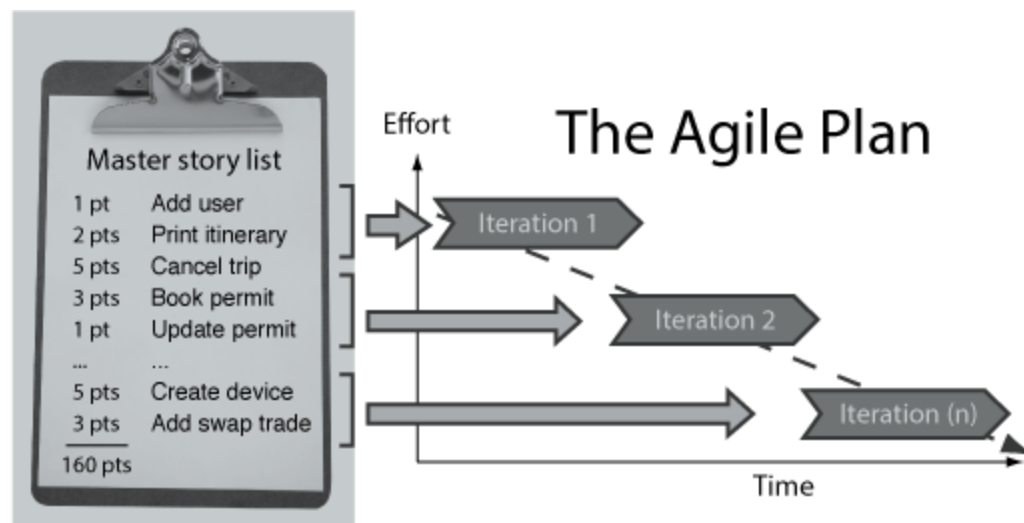


Initially, TARA will be following a waterfall model approach during development. Waterfall model involved requirements gathering, analysis, design, build, test and deployment, and it suits perfectly well to TARA's development processes as each stage is documented and the entire process is considered as the whole development lifecycle. The issue with TARA is that consisted of many requirements, and development time was limited to three months and changes are expected to occur during analysis and development. Using waterfall model would have meant, more documentation and if there are any changes at the end of development, it would mean going back to the first stage.

#### Updated approach:

Agile development and testing is the chosen development lifecycle for TARA as this ensures implementation of the core requirements within two months. The introduction of sprints for each implementation aggregates the specific development task within an iteration space.





These iterations consisted of development time of at least 2 to 4 weeks. During this time the development of a requirement must be completed before attempting to implement other requirements, and any unfinished task is reserved for the next iteration sprint. The implementations are deployed in stages rather than deploying all the application in one go.

Furthermore, it was also easier to test small chunks of the functionality during development rather than having large documentation of test cases when using waterfall. Subsequently, this gives the developer and the project's supervisor enough time to challenge and revisit assumptions that could be further improved as waterfall leads to further documentations and rewriting requirements. Agile was a necessary approach during TARA's development as it was not about making development time faster, it was about ensuring that development can begin with less documentation and initial requirements implemented and tested along with the development as well as deployed in stages.

### 3.3.4 React native or native android development

#### **Original Plan:**

TARA's original development platform was to develop it natively using Android Studio, a platform built by Google to build an application for many android devices. TARA's future goals involve deployment to both Apple and Android devices and a native development in Android studio requires programming in Java, this isolates TARA from deployment to another platform which would eventually require to re-build it on a different language. Installation processes are slow, and there are often quite a lot of resources needed if development continued in Android studio. Leading to further installation of dependencies and connection of the actual physical mobile device into the computer to run or use a simulator which requires sufficient memory capacity. A new technology called React native is in existence in a faster and efficient way to build, test and deploy.

#### **Updated approach**

React native only requires one code base one that guarantees that it would work on both mobile platforms such as iOS and Android. Current development state of TARA is just for android devices, but future deployments will also involve deployment in iOS devices. React native is fast and lightweight, build by Facebook to bring more rapid mobile development. As TARA's developer, with in depth experience in web development, it was great to be able to use web application elements by using HTML like features in React native. Enabling me to build the user interfaces and design fetch calls to API using React native in a short time. There are no massive installations required, and development was immediately performed using React native.

Furthermore, mobile application development was more convenient due to the existence of Expo. Wern Anchetta (2018) suggests developers can create React Native apps without all the frustrations that come with installing and configuring software dependencies with the likes of Android Studio. Using Expo enabled TARA's development in a mobile device to be effortless, and it was possible to build, test and deploy without the need of a simulator which required a tremendous amount of memory capacity.

## [4] Deliverables and approach implementation

Continuing from the mentioned approaches in section 3, this section of the report highlights the application of core functionalities of TARA.

### 4.1 User accounts and authentication development implementation

TARA's back end system is using Python's powerful web development framework called Django. Django was designed to aid developers to take applications from concept to completion as quickly as possible Django (2018). The first step was to develop the schema that will accommodate the user's username, email, and password, using a library called *AbstractBaseUser*. Rather than creating and form a new user account model, it was decided to use the pre-defined base user model of Django which are accessible by the *AbstractBaseUser* library. Django partners with a lightweight webserver and it was designed to access and manage user groups of any database that are integrated to Django.

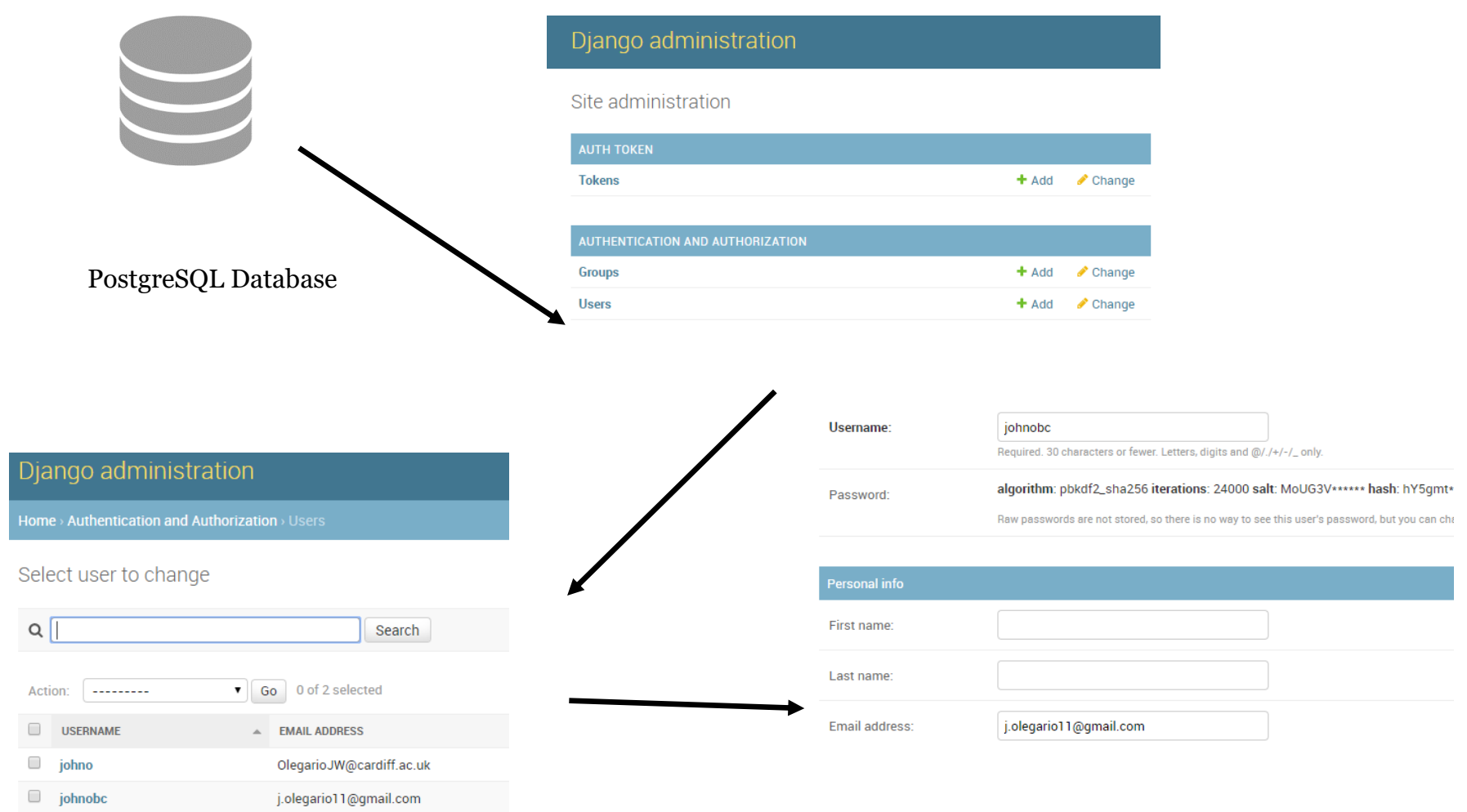


Figure 13: Django User model

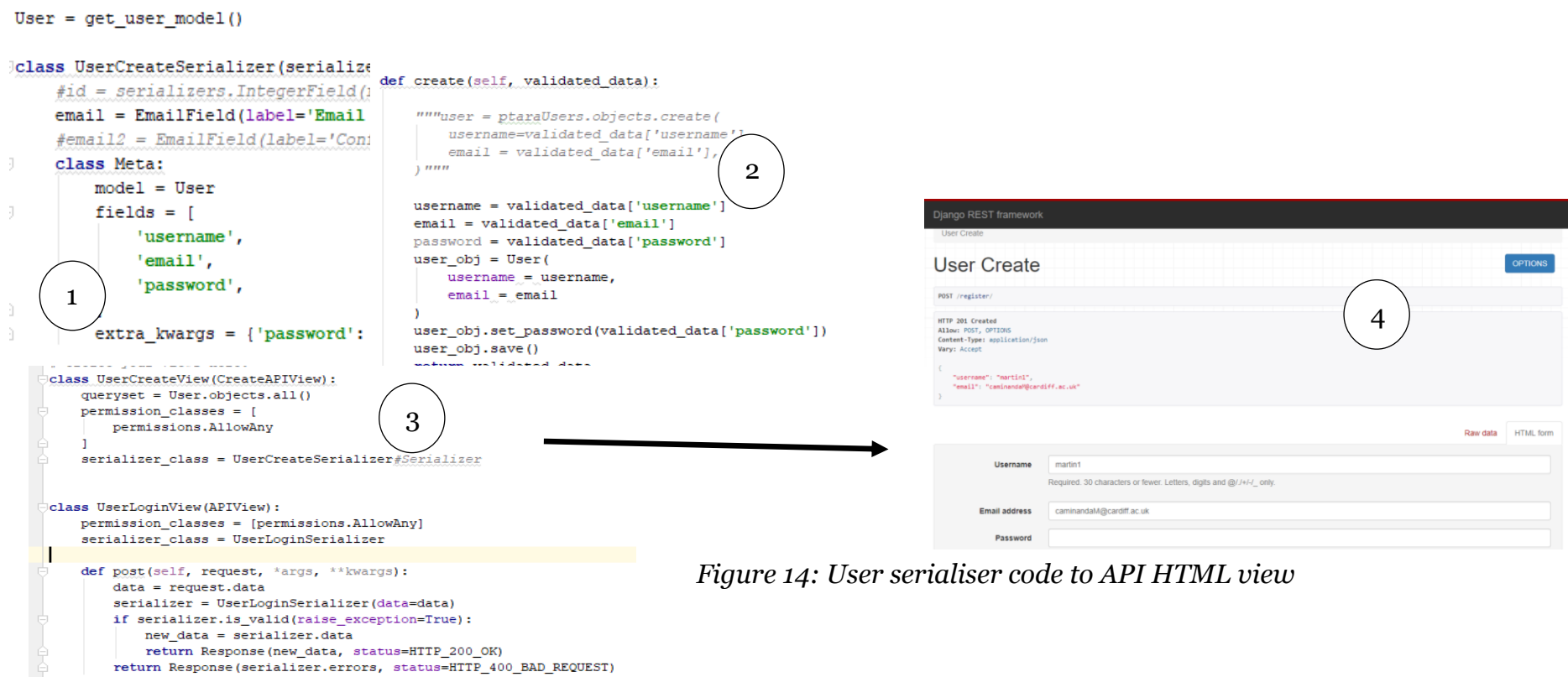
Figure 13 reveals how user accounts are accessed and modified. The model contains the username, password, first name, last name and email address. TARA users will only require an email, username and password during sign-up and just the email and password when logging in. All passwords under the *AbstractUserBase* library is protected with an encryption algorithm, ensuring a secure entry when users are logging in to TARA's system.

Each user has a dedicated token to authenticate a user login; this token is a validation requirement when a user logs in to the system as each call to the back-end web server to access the user database will validate the user using this token for access.

```
# token is: 4bd97c6a3da72d83cee684617f43718811db4d88
@receiver(post_save, sender=settings.AUTH_USER_MODEL)
def create_auth_token(sender, instance=None, created=False, **kwargs):
    if created:
        Token.objects.create(user=instance)
```

This function enables the creation of a token for individual users. Also, the contents of the database are delivered in the form of a REST (Representational state transfer) API with default domain name for the front-end client to access its contents. A REST API defines a set of functions which developers can perform the request and receive responses via HTTP protocol such as GET and POST, Sam Derring (2012). To make this possible a library called DRF (*Django rest*

framework) is a library in Django which enables to serialise the contents of the database, as user login and registration can be done using this API.



DRF is an excellent tool for building web API the RESTful way: allowing you to interact with your database, Richard Tier (2014). The code in figure 14 reveals how the user model of username, email and password are serialised. According to DjangoRESTframework2 (2018), serializers allow complex data such as query sets and model instances to be converted to native Python datatypes that can be then easily rendered into JSON, XML or other content types.

- 1) *UserCreateSerializer* code is the implementation of the serializer for user registration; there is also a separate serializer for user login.
- 2) The following is a section of the code is the functionality of how the user registration saves the new user information in to the database.
- 3) *UserLoginView* code handles how the serializer is viewed as an HTML page, and it includes of setting the permission to allow any changes which could either be a read or write and conditional handle to verify if the serializer data is valid as a post request.

The purpose of having the contents of the database and serialise in the form of a REST API is to enable GET or POST request from the front-end client in React Native, which will be calling the webserver from its domain. React native has a fetch function in the background, and to access the contents of the database, authorisation headers are present in the code as well as the concatenated token to authenticate access.



The login user interface has two pages as one will require users to login if they are already a member, two input boxes are for their username and password, successful authentication will lead to the main page of TARA. Figure 15 reveals the function within the React native code with the use of the fetch functionality since all the data stored in a database and requested via an API. Embedded within this fetch code is the call to the API domain as well as the authorisation headers along with the token and content type which is in a JSON format as DRF reveals the content of data in a JSON format.

This call is a POST method and is a destined approach when supplying request via username and password as details will be sent in the HTTP messages rather than the URL as stated in the article GET vs POST, Diffen (2013). The initial design of the login-in and sign-up user interface which required users the need to login or register before using the application and is subject to change in future development which will be in section 6. User registration is functional in the API; it is not yet functional on the React native user interface.

## 4.2 User current location functionality

User current location as mentioned from previous sections as part of TARA's feature is to determine the exact location of the user as this section provides the detail of how this implemented. TARA is taking advantage of the embedded GPS technology to many android devices which in return gives the latitude and longitude value which are rendered using React native and by using Google's geocode API would determine the actual location of the user based on the latitude and longitude values. The figures below will show a visual representation of how this was coded and the user interface view.

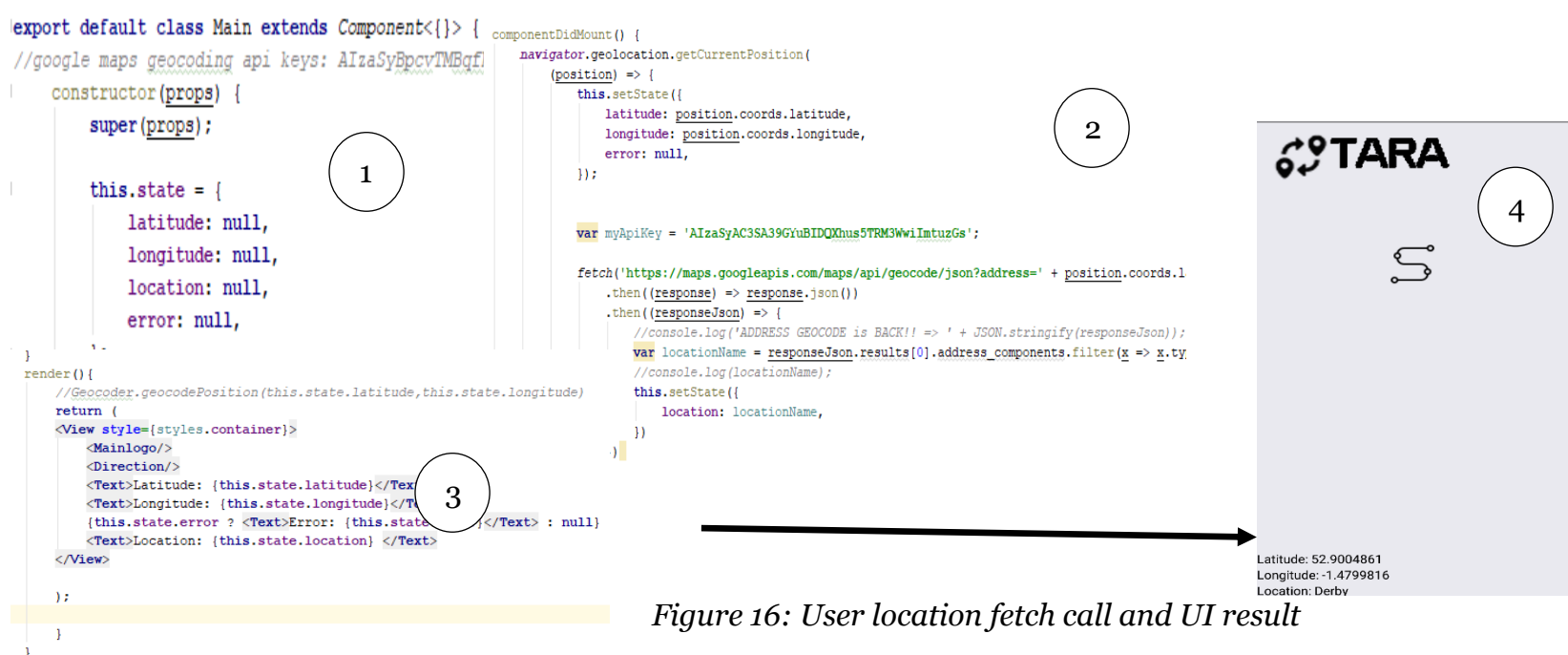


Figure 16: User location fetch call and UI result

- 1) This section of the react native code holds the states that are set to null by default, this is a global state where it will keep the latitude, longitude, and location values once the geocoder API has determined the actual value.
- 2) React enables to create components by invoking the `React.createClass()` method which expects a render method and triggers a lifecycle that can be hooked into via several so-called lifecycle methods, A. Sharif (2015). `ComponentDidMount()` functionality is an essential feature of React regarding the component lifecycle, and it holds the fetch function which makes the calls to the Google Geocoder API to identify the user's location.
- 3) The render section holds mark-up section of the React-native code which in return handles the actual view of the user interface. Similar to HTML elements that can be styled and positioned, React-native elements are used and designed to display information as revealed in figure 16, showing the latitude and longitude values and the exact user location.


The next stage of finding the user location is having this displayed in an actual map view. A map view is perfect for informing the user about their exact geo-spatial location rather than showing latitude and longitude values in a text view. The latitude and longitude will become arguments to the geocoder library and integrated to Google maps in react native, one that pinpoints the actual location of the user as mentioned in section 3.2.2, the user must tap this icon  that leads to the *Where-to* page, where a map exists.



Figure 17: Map view of user location and UI

- 1) This code section reveals the latitude and longitude states and is recalled in the render section using the map view tag. The code shows how the latitude and longitude values revealed a marker in map view of the geospatial location of the user. Zoomed towards the actual region of the user in the earth, a marker is responsible for pinpointing the exact location of the user.
- 2) The actual user interface which reveals the exact geo-spatial location of the user in Google Maps.

### 4.3 Nearest bus station markers

Adding to TARA's essential features is to show nearest bus stations around the user's location and must be marked in the map view. As previously mentioned in the previous sections, TARA is utilising several of Google API webservices including the use of Google places and Geolocator API to locate the user's location. The location of the user is determined via latitude and longitude values. Also, these values are used as further arguments to the request made to Google nearest location API which uses the current location of the users and is programmatically filtered to reveal the bus stations within the specified radius of the user's vicinity.







Figure 18: Nearest bus stations fetch call and UI


The code and user interface above reveals how the implementation in the React native code and the actual user-interface view.

- 1) The first section of the code shows how current width and height was used as constant values and halved, which is a style to fill only the half part of the user interface. Performed to give space for the other components, i.e. text-input box. The remaining area will be used to display additional relevant information to the users such as bus timetables.
- Also, dealing with multiple markers lead to declaring a markers array state which will house all the results from the nearest location API. Subsequently, rendered in the view section to reveal the nearest bus stations from the user's location, represented as station markers.
- 2) The code reveals the use of an additional fetch function from React native, directed to call from the Google's nearby search API. Latitude and longitude values determine current user location are arguments when calling the API and filtered to reveal the nearest bus stations from the user's vicinity. Google's nearest API has a wide range of coverage of transit information worldwide which makes it highly suitable for this purpose, the screenshot below shows the actual API results when the API returns data.

```
{
  "html_attributions": [],
  "results": [
    {
      "geometry": {
        "location": {
          "lat": 52.900504999999999,
          "lng": -1.478267
        },
        "viewport": {
          "northeast": {
            "lat": 52.90185398029149,
            "lng": -1.476918019708498
          },
          "southwest": {
            "lat": 52.89915601970849,
            "lng": -1.479615980291502
          }
        }
      },
      "icon": "https://maps.gstatic.com/mapfiles/place_api/icons/bus-71.png",
      "id": "4e992af207c8ff3503999574d25155ccf9b48f70",
      "name": "Randolph Road",
      "place_id": "ChIJM0syu93weUgRXSKhX6Flay8",
      "reference": "CmRRAAAAE7cMBH9gq0wIU18RftzX1GsEjTb7G2msTt4sL1hnbv5DnwTpf",
      "scope": "GOOGLE",
      "types": [
        "bus_station",
        "transit_station",
        "point_of_interest",
        "establishment"
      ],
      "vicinity": "United Kingdom"
    }
  ]
}
```

Example result of the Google nearby search API in JSON format

The results of the Google nearby API revealed an array list of the nearest bus stop locations in the area and set to a radius of 2000 meters. The results are placed under constant markers variables and map the result into a list of latitude and longitude values. The *latlng* object holds the response which contains that latitude and longitude values of the nearest bus stations. An initial markers state array was declared which is set to contain the latitude and longitude values from the *latlng* object.

- 3) The third section of the code reveals the rendering of the markers array state in map view. A finite array list of markers houses the latitude and longitude values of the nearest bus stations from the user's current location. The map view has the coordinate section which takes the marker and the latitude and longitude values that reside within the *latlng* object, became arguments for the react native maps library. The latitude and longitude coordinates are used to pinpoint the geospatial locations in map view.
- 4) Respresenting the user interface that was rendered from the view section of the React native code and zoomed to the region with a marker in place to represent the actual location of the user. Additional markers are present around the region of the user's location which identifies the exact location of the bus stations, represented with an icon image. 

This section of TARA's functionality reveals the proximity of transportation terminals around using Google's nearby search API. Assuming that most travellers will have modern android mobile devices connectivity is achieved most of the time. Caching information from Google maps is against Google's terms and conditions when using their API, therefore when connectivity is disconnected, the cached data will come from a different source. The source of this cached data comes from a locations database which is part of a future work for TARA mentioned in section 6.

### 4.4 Finding directions and timetable

Adding to TARA's functionality is the ability to query about the directions of their intended destinations. As specified in section 3.3.4, the application will provide direction information on how to get to a specific location via transit information, such as the location of the bus station and its timetable.

#### 4.4.1 Finding Directions



Figure 19: Polyline fetch call and UI view

- 1) The first part of the code is the *fetchDirections* () function which is only triggered once a user interaction is detected. The event happens by tapping the *Go* button as it contained fetch functions which directly calls to Google's Direction API. TARA's components are immediately invoked such as the text inputs, map views and will re-invoked once that fetch calls are triggered. The first part of fetch calls to are directed to acquire polyline data from the response.

```
"polyline": Object {
  "points": "a|~aIte_Hc@NC@gBj@MDKDaBdAq@^GHCFCFCNCFEHEF",
},
"start_location": Object {
```

#### Example result of polyline JSON response

The polyline points are specific points within the surface of a geo-spatial map that can be decoded and represented as a line from the starting location and final destination of the user. A unique library within React-native is used to decode the polyline response called Mapbox-polyline. A polyline is just a mash of many smaller segments, line segments that uses latitude and longitude values to plot parts of the path, Ravish Rawal (2017).

- 2) The second part of the code is the contents of the render function as it shows all the variables containing the individual states with unique values. In instance the variable *const actuallocaton = this.state. Latitude* holds the

latitude value from the initial state. Conveniently recalled in the render function, reveals the front-end code for the Text-Input is asking users to input their desired destination. A Touchable Opacity button called *Go* where it was bind to the *fetchDirections ()* button, which will only be triggered once the user taps this button. Once the user inputs their destination, this becomes a parameter to the fetch call along with the latitude and longitude values of the actual location of the user.

```
fetch('https://maps.googleapis.com/maps/api/directions/json?origin=' + latitude + ',' + longitude + '&destination=' + goingto + '&mode=transit&transit_mode=bus&key=' +
```

Example fetch call to acquire polyline data to Google Direction API

- 3) This section of the code is the continuation of the render and view section of the react native code which reveals the uses of the *polyline* class from the mapbox library. It contains the stored coordinates from the constant variable called *directioncoordinates* which includes the latitude and longitude coordinates of points to render the polyline within the map view.
- 4) This section reveals the actual user interface view when the user queries their desired destination; a line should appear which shows the starting location leading to their desired destination.

4.4.2 Bus stop location

TARA can inform users which bus station to get to in order need to get to their point of destination. It is an essential functionality that highlights within the map view of where users need to go to catch their public transport and at this stage TARA’s development it is initially achieved to reveal bus travel information.

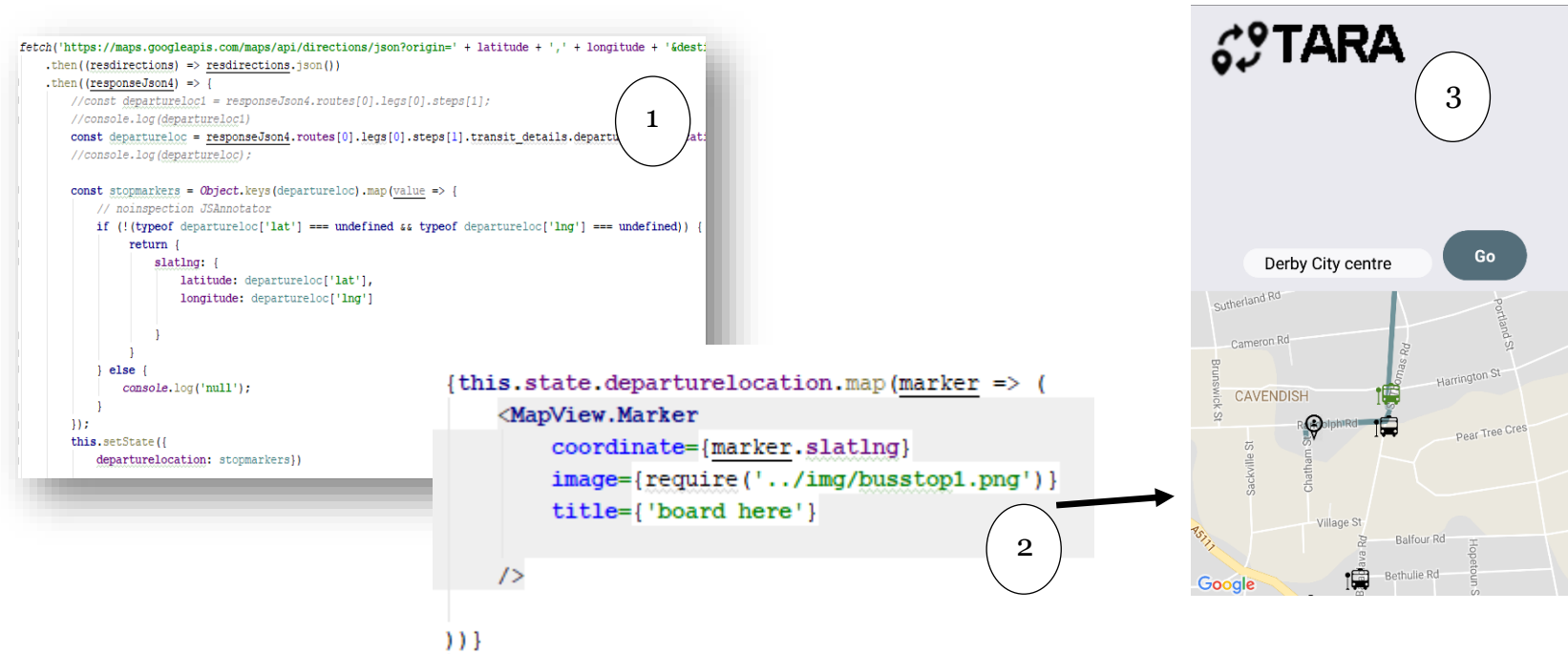


Figure 20: Bus station fetch call and UI

- 1) Within the *fetchDirections ()* function is another fetch call to the Google Directions API and filtered to reveal transit details. This response to this call gives an entire list of object response; unlike the previous responses a sample is revealed in section 4.3, a response ID accompanies all of the responses. Subsequently, this required extra steps that are hardcoded within the code that traverse the individual objects to get to the right object that contained transit information.

```
const departureLoc = responseJson4.routes[0].legs[0].steps[1].transit_details.departure_stop.location;
//console.log(departureLoc);
```

Sample code that traverses to the object responses of Google Directions

*Map ()* function creates a new array and as shown on previous fetch calls to set latitude and longitude values as markers and rendered in the map view. On the other hand, the response call from Google Directions API are all objects, and it was impossible to use the *map ()* function as it only works with arrays.



```
Object {
  "geocoded_waypoints": Array [
    Object {
      "geocoder_status": "OK",
      "place_id": "ChIJ2YDFT9zweUgRgt5Tws6n3hs",
      "types": Array [
        "premise",
      ],
    },
    Object {
      "geocoder_status": "OK",
      "place_id": "ChIJW2PCihXxeUgRRWgA9kOzpjY",
      "types": Array [
        "establishment",
        "food",
        "lodging",
        "point_of_interest",
        "restaurant",
      ],
    },
  ],
  "routes": Array [
    Object {
      "bounds": Object {
        "northeast": Object {
          "lat": 52.9242042,
          "lng": -1.4751044,
        },
        "southwest": Object {
          "lat": 52.900484,
          "lng": -1.4801554,
        },
      },
    },
  ],
}
```

An example response from Google Directions API

The only solution at this stage was to append the initial response from Google Directions into an array so the `map()` function will work. Made possible by using *the* `Object.keys()` function as it returns an array of the given object. As shown in figure 21, `Object.keys()` was used to return the entire response into an array and mapped, to be able to traverse through the objects. The object of interest is called *transit\_details*, which consisted of latitude and longitude values that will be stored to pinpoint the actual location of the station. Markers are in place and have been set to the latitude and longitude values from *transit\_details* and rendered as markers in the map view. Attempting to traverse and map without using `Object.keys()` function will result in the following error

[Unhandled promise rejection: TypeError: undefined is not a function]


- 2) This section of the code is the render section which reveals the front-end code which sets the markers according to the latitude and longitude values of the detected bus station.
- 3) The final UI results are revealed, when a user queries their desired destination, it will reveal the decoded line and the location of the bus station. Informing users where they need to be to catch the bus that will get them to their destination. The green bus station icon informs users that this is the station that they will need to be to get to their desired destination.

4.4.3 Bus station Timetable

An essential functionality of TARA when users are queries their desired point of destination is to reveal the transport timetable schedule, showing the departure and arrival time of particular transit service. At this stage, TARA will be able to reveal bus timetable services depending on Google webservices coverage. Accompanied by the line points and the bus station location is the actual timetable and final destination stop, this section of the report reveals this functionality in detail.



Figure 21: Fetch timetable and arrival stop and UI

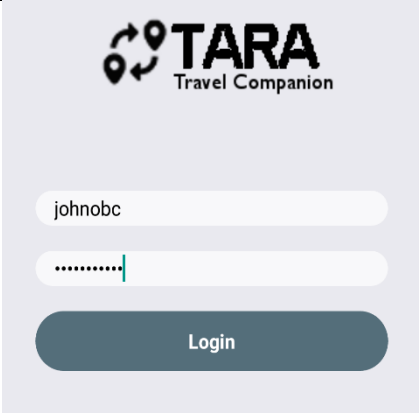
- 1) This section of the code is fetching to acquire the timetable information response JSON result from Google Directions API, which is now filtered to directly take the departure stop name, departure time, arrival stop name, and arrival time.
- 2) Another function call that should reveal the arrival stop icon to users and displayed in the Map view, informing the user that this is their destination stop. This function is similar to the code revealed in figure 21, only, in this case, shows the arrival station in a red bus station icon. 
- 3) This section is under the render and view of the react native code which shows how bus timetables are called from the states and rendered in the view section. Constant variables hold the states about the related transit information of the particular station.
  - Startingstop (this.state.busstopname) – holds the actual departure station name
  - Starttime (this.state.departbustime) – holds the latest departure time of the specified destination
  - Finalstop (this.state.arrvstopname) – holds the arrival station name
  - Finaltime (this.state.arrivbustime) – holds the actual arrival station name
- 4) Revealing the actual user-interface, it reveals the combination of the overall mentioned functionality when a user queried their desired destination. A line appears to show the routes to the destination; it also reveals the station location which is represented by a green bus icon. Subsequently, this also informs users which station to get to travel to their desired destination as well as the timetable of the particular service. Lastly, the red bus station icon appears indicating that this is where a user should stop as this is the final destination.

One of the primary requirements that were achieved and that is for the user's ability to query transit information and reveal direction information, notably the departure and arrival point. The users will be able to query transit information worldwide depending on Google maps API coverage, one reason that it differs to many other applications as most are local applications based on their location, e.g. Traveline Cymru for wales, MTR Mobile in Hong-Kong.

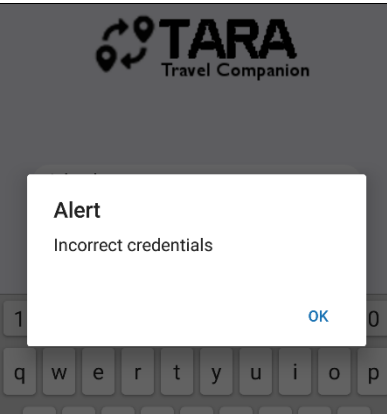
# [5] Initial Results, Testing and Evaluation

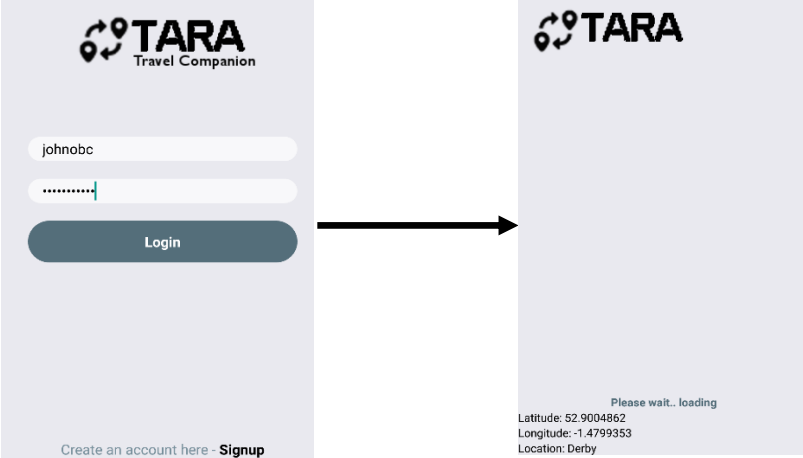
Implemented TARA’s core functionalities initial testing are revealed in this section. Additionally, this section also of the report provides an overview of the results in detail and evaluation of the testing process after completing the initial sprints of development.

## 5.1 TARA Agile Test Result Record 1

<b>Sprint Title</b>	User Log in	<b>Development Period</b>	28/02/2018 – 19/03/2018
<b>Functionality Test Title</b>	User Login Password secure entry	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	1	<b>Date of Test</b>	20/03/2018
<b>Test No</b>	1	<b>Time of Test</b>	13.00
<b>Expected Result</b>	Functioning text input box and typing the password should be encrypted.		
<b>Test Result</b>	Functioning text input box and password is encrypted, displayed results to the users are normalised as bullet points.	<b>Decision</b>	Deployment ready
<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO

<b>Sprint Title</b>	User Log in	<b>Development Period</b>	28/02/2018 – 19/03/2018
<b>Functionality Test Title</b>	Unregistered users denied login	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	1	<b>Date of Test</b>	20/03/2018
<b>Test No</b>	2	<b>Time of Test</b>	
<b>Expected Result</b>	Incorrect credentials or Non-members of TARA should not be able to login the application.		
<b>Test Result</b>	Incorrect credentials and non-registered users of TARA will be refused entry to the main application of TARA. A message will appear informing users that they have given incorrect credentials	<b>Decision</b>	Deployment ready

<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO

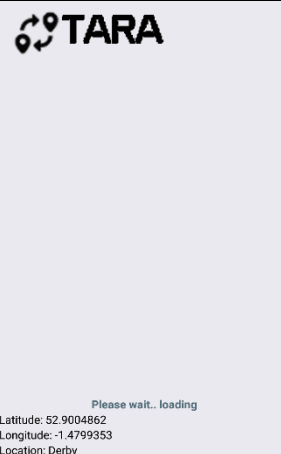
<b>Sprint Title</b>	User Log in	<b>Development Period</b>	28/02/2018 – 19/03/2018
<b>Functionality Test Title</b>	Authentication access	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	1	<b>Date of Test</b>	20/03/2018
<b>Test No</b>	3	<b>Time of Test</b>	15.00
<b>Expected Result</b>	Valid username and password will be granted access to the main page of TARA, authentication involved validation of token from the users in the back end, and a wait of 5 seconds should occur before access to the <i>where to</i> button after successful navigation into the main page.		
<b>Test Result</b>	User authentication successful results to successful navigation to the main page and upon initial login into the main screen, <i>‘loading, please wait’</i> message will appear, for at least 5 seconds to ensure all the fetch calls and components have been loaded.	<b>Decision</b>	Deployment ready
<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO




<b>Sprint Title</b>	User Log in	<b>Development Period</b>	28/02/2018 – 19/03/2018
<b>Functionality Test Title</b>	Successful User registration leads to the main page	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	1	<b>Date of Test</b>	20/03/2018
<b>Test No</b>	4	<b>Time of Test</b>	15.00
<b>Expected Result</b>	Unregistered users will register by tapping the sign up page, it requires their username, email and password and doing so will lead to the main page. New credentials will be stored in the user’s database.		


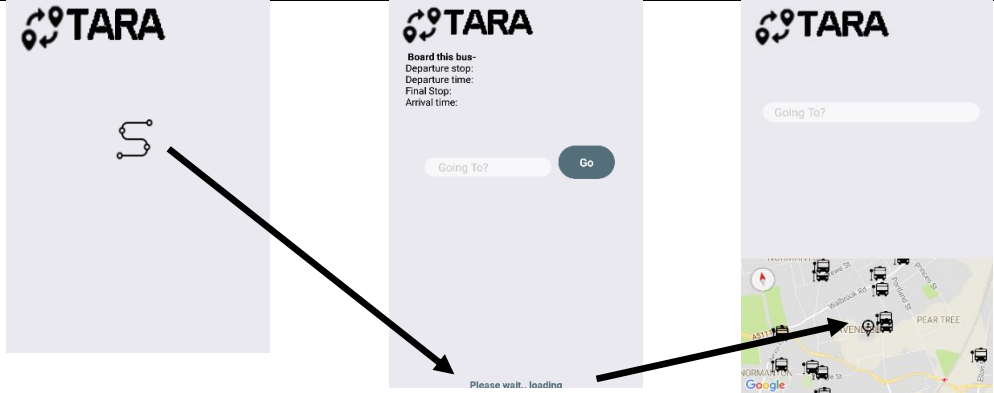
<b>Test Result</b>	Functionality is currently not working, tapping sign up does not lead to the main page.	<b>Decision</b>	Reserved for the next development sprint
<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO



### 5.2 Google Webservices integration TARA Agile Test Result Record 2

TARA’s use of Google’s Webservices API transtlates into several functionalities of different iterative sprints. One call provides information about the actual user location, nearest bus stations, and directions to the point of destination. This section reveals the results after initial testing.

<b>Sprint Title</b>	Google API call and display result into UI	<b>Development Period</b>	21/03/2018 – 19/04/2018
<b>Functionality Test Title</b>	User latitude, longitude and location display	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	2	<b>Date of Test</b>	21/04/2018
<b>Test No</b>	1	<b>Time of Test</b>	10.00
<b>Expected Result</b>	As soon as the user was granted access to the main page, the user's location should have already been determined and displayed on the main page after initial login		
<b>Test Result</b>	Latitude and Longitude values are revealed to the user after successful log in at the bottom left corner of the screen.	<b>Decision</b>	Deployment ready
<b>Screenshot Result</b>	 <p>The screenshot shows the TARA application interface. At the top, there is a header with the TARA logo. Below the header, there is a map view. At the bottom of the screen, there is a text overlay that reads: "Please wait... loading", "Latitude: 52.9004862", "Longitude: -1.4799353", and "Location: Derby".</p>		
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO

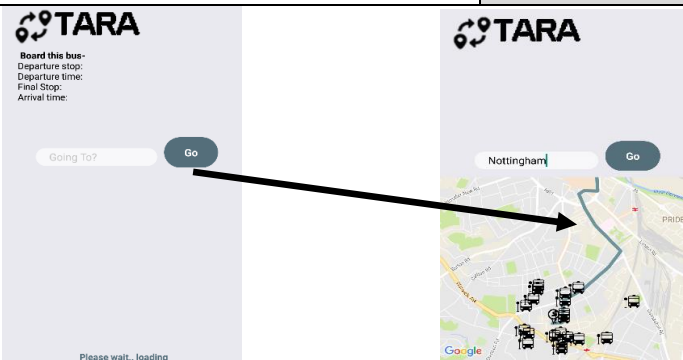
<b>Sprint Title</b>	Google API call and display result into UI	<b>Development Period</b>	21/03/2018 – 19/04/2018
<b>Functionality Test Title</b>	User location revealed in map view	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	2	<b>Date of Test</b>	21/04/2018
<b>Test No</b>	2	<b>Time of Test</b>	11.30
<b>Expected Result</b>	After initial log in the application will load for 5 seconds and a touchable component should appear  . Tapping  will navigate to <i>Where to</i> page and expect a five wait load to let the application fetch all the data and only then the map view will be visible with the user’s current location.		
<b>Test Result</b>	Once  is tapped, there is a 5 second wait before the actual user location is displayed.	<b>Decision</b>	Deployment ready

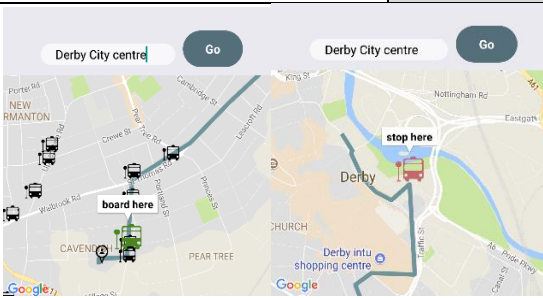
	User current location is represented in map view by this icon 		
<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO

<b>Sprint Title</b>	Google API call and display result into UI	<b>Development Period</b>	21/03/2018 – 19/04/2018
<b>Functionality Test Title</b>	Nearest bus station location	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	2	<b>Date of Test</b>	21/04/2018
<b>Test No</b>	3	<b>Time of Test</b>	14.00
<b>Expected Result</b>	Tapping this icon  will proceed to the <i>Whereto</i> page. In addition to the appearance to user's location, the nearest bus stations will appear based on the user location. A five second load will occur, and then the map view will appear revealing the user's location and the nearest bus station within the user's vicinity and within a radius of 2000 meters.		
<b>Test Result</b>	The application reveals the current location and the nearest bus station around the user's vicinity with a radius of 2000 meters.	<b>Decision</b>	Deployment ready
<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO

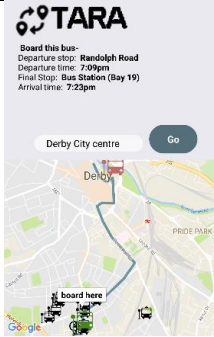
<b>Sprint Title</b>	Google API call and display result into UI	<b>Development Period</b>	21/03/2018 – 19/04/2018
<b>Functionality Test Title</b>	Direction line in map view	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	2	<b>Date of Test</b>	21/04/2018
<b>Test No</b>	4	<b>Time of Test</b>	16.00
<b>Expected Result</b>	<i>Whereto</i> section only shows one text input from the user, and it is asking users their desired destination. Subsequently, this should reveal a line which is known as a polyline in the geo-spatial map view showing the way from the current location of the user to their final destination.		
<b>Test Result</b>	When the desired destination was entered from the text input in the <i>Where to</i> page, and the <i>Go</i> button was tapped. It reveals a line	<b>Decision</b>	Deployment ready



	between the user location and destination.  Occurs when a user inputs their desired destination, i.e. ‘Bridgend’ or ‘CF23 5EF.’		
<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO

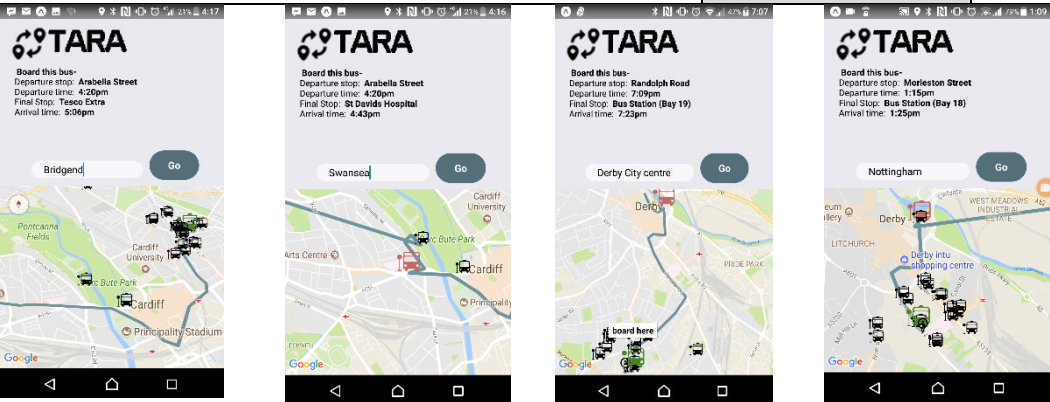
<b>Sprint Title</b>	Google API call and display result into UI	<b>Development Period</b>	21/03/2018 – 19/04/2018
<b>Functionality Test Title</b>	Bus stop location to board and to stop in map view	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	2	<b>Date of Test</b>	22/04/2018
<b>Test No</b>	5	<b>Time of Test</b>	10.00
<b>Expected Result</b>	<p>When a user is in a specific location and used TARA to query a destination, not only it would reveal the line of travel, the application will also show the location of the bus station in the geo-spatial map view. As soon as the user queries a destination, this would return the line and the location of the bus station that the user needs to board and to stop. Each time the user moves to a different position, the application will update the user of the new station location and indicates the user of the whereabouts of the new station. This icon will appear informing the user of the site of the station for boarding 🚌 and this icon should appear notifying the user where to stop in the map view 🚏. Also, it is expected that this functionality will work in many locations and is dependent on Google’s direction API coverage.</p>		
<b>Test Result</b>	<p>This functionality is fully operational. If a stop or link is not available or if there are interruptions in the bus service then Google webservices tries to provide the data by giving users the alternative route. Subsequently this is reflective in map view, and the geo-spatial line will be visible on the map. If there are no bus services at all TARA, no data will be returned.</p>	<b>Decision</b>	Deployment ready
<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO

<b>Sprint Title</b>	Google API call and display result into UI	<b>Development Period</b>	21/03/2018 – 19/04/2018
<b>Functionality Test Title</b>	Bus timetable view	<b>FUT- Version (Functionality under test)</b>	1

<b>Sprint No</b>	2	<b>Date of Test</b>	22/04/2018
<b>Test No</b>	6	<b>Time of Test</b>	13.00
<b>Expected Result</b>	Successfully fetching timetable data from Google Directions API and displayed in the <i>Whereto</i> page of TARA. When a user queries a destination, it reveals the direction line, the location of the stop, as well as the station name, departure time, arrival stop name and the arrival time. Bus transit schedule should be displayed when the user queries a destination which is related to the station that was identified as the boarding station.		
<b>Test Result</b>	The application reveals the timetable of the determined location of the station that the user will need to board. Tapping the icons will show a short message in the map view, the <i>board here</i> and <i>stop here</i> . If it is connected travel, the application will display the timetable information about the first station first. When the user arrives at the station to transfer to a different bus, TARA should disclose the timetable information about that individual station then	<b>Decision</b>	Deployment ready
<b>Screenshot Result</b>			
<b>Tester Name</b>	John Olegario	<b>Tester Initials</b>	JO

<b>Sprint Title</b>	Google API call and display result into UI	<b>Development Period</b>	21/03/2018 – 19/04/2018
<b>Functionality Test Title</b>	TARA usable on multiple locations	<b>FUT- Version (Functionality under test)</b>	1
<b>Sprint No</b>	2	<b>Date of Test</b>	22/04/2018
<b>Test No</b>	7	<b>Time of Test</b>	15.00
<b>Expected Result</b>	TARA is expected to function in many different locations and is dependent on the coverage of Google’s webservices API. Core functionalities are expected to work including the identification of the user’s location, nearest bus stations, directions, view of the bus station to board and timetable.		



Test Result	Initial testing was performed in the UK in two locations, one was in Cardiff Wales, and the other was in Derby England. Core functionalities are fully functional, yet more testing will be required to be granted for deployment to ensure that it is functional and provides the right information on any location.	Decision	Subject to further testing
Screenshot Result			
Tester Name	John Olegario	Tester Initials	JO

### 5.3 Overall Evaluation of the Testing Process

This section of the report provides an overview of the significant sprints that was documented from the Agile Test Records mentioned in sections 5.1 and 5.2. The section includes achievements that were agreed on the initial report and explanations in areas that were not successfully implemented as initially agreed, which was either moved to future sprints or subject to further testing.

#### 5.3.1 Sprint no-1 User log in

The development of user log in began on the 28<sup>th</sup> of February, and this was one of the major requirements that were agreed on the initial plan to bring forward user accounts to use TARA. The testing was conducted right after the implementation was successful in ensuring that users can log in. The testing device was in an Xperia Z5 smartphone, with 3Gb of memory and like most android phones with 3G and 4G enabled network capability as registered users of TARA, should be able to log in to the application whether using Wi-Fi or network data.

**Test No.1** refers to the assurance of protecting users credentials when logging in, and password entry should always be encrypted when logging in.

**Test No.2** refers to the user interaction of the application to non-registered users with a message appearing to the user that credentials are incorrect. It is assumed that it is within user’s knowledge whether they are registered to TARA or not.

**Test No.3** refers to the user interaction of the application to registered users, granting access to the *main* page of TARA.

**Test No.4** refers to un-registered users when attempting to register using TARA’s during registration.

What the test cases regarding users log in functionality was conducted regardless of whether an implementation was completed or not. It was done to provide some documentation of the next step for the feature. Test 1, 2, and three are all cleared for deployment. Test no.4 was not successfully implemented during this sprint, and it was decided to diverge from the agreed plan from the initial report that user registration will release along with the user login.

Implementation of user’s login functionality took longer than expected due to complications when setting up the back-end system it was decided to no longer use MongoDB as the back-end database and use PostgreSQL instead, the justification for this was mentioned in section 3.4.1. It was not possible to fit the other sprints within the user’s login sprint timetable that was originally agreed on the initial report. As such, the integration of MongoDB, and speed detection functionality were postponed due to the unexpected more extended development of the user’s login, and it was decided to move all this functionality for future development mentioned in section 6.

#### 5.3.2 Sprint no -2 Google API call and display result into UI

The development and integration of Google API began on 21/03/2018. It was decided to completely alter the set of core functional requirements that was mentioned in the initial report. The research was on-going up until the end of week 4, upon the initial release of the user login functionality and the initial core requirements that was mentioned in the initial report. The initial core requirements included data taken from users post, comments as social media elements which at

that time was a great idea. However, it was simply not sufficient enough to justify the actual use of TARA as it is branded as a travel companion app. TARA required immediate access to a reliable and vast collection of transportation data, and social media data would not have sufficed as TARA does not have any users yet to acquire sufficient data that would be enough to inform users of relevant information.

Therefore, it was decided that Google's API webservies will take over as the main data hub for TARA, which is integrated to the application to call for transportation information and to render the response for display within the application. Testing began from the 21/04/2018 to 22/04/2018, which are several tests that relate to several fetch calls to the Google API of which are relevant to the core functionalities of TARA. Testing was performed in a Sony Xperia Z5 smartphone, fitted with GPS technology which is highly important in determining the user's location from the responses acquired from Google's webservies API. The justification for the use of Google's Webservices was mentioned in section 2.4.

**Test No.1** refers to the determinacy of the user's location as soon as registered users were successful on logging into the main page.

**Test No.2** refers to the rendered response of the user's location which should be viewable in the *Whereto* page where a map is revealed to the users revealing their actual location.

**Test No.3** refers to the rendered response of the location of the nearest bus station around the user's location, which is viewable in the map view in the *Whereto* page.

**Test No.4** refers to the polyline response from Google's Directions API, which is triggered when a user entered the desired destination. Subsequently, this equates to a geo-spatial line that is visible in map view showing the directions on getting to the desired destination.

**Test No 5** refers to the appearance of bus station locations within the map view, informing users that they must board this bus on this location to get to their desired destination.

**Test No 6** refers to the appearance of the bus timetable after a user has queried their desired location. In conjunction with Test No 5, the bus timetable information will appear as soon as the location of the boarding station was revealed.

**Test No 7** refers to the ability of TARA to be able to use in many locations by its users by the coverage of Google's webservies API.

All test-cases were tested thoroughly apart from Test No.7, as the test was only performed in two locations. Further testing is required to ensure that it is functional when used in many places. Testing in multiple locations or different countries was not possible at this stage due to time constraints. Overall it is with high confidence that Google's webservies API was correctly integrated and should provide constant information to most users of TARA, concerning Google's transit coverage.

## [6] Future Work

This section of the report provides a detailed overview of the next development plans for TARA which will be for future development sprints. TARA is far from completion, and there are still many on-going issues that need to be addressed which are reserved for future sprints of development for TARA.

Here are some of these functionalities:

### 6.1 Smart suggestions and machine learning

An essential feature of TARA which is heavily favoured for development is the introduction of the user posts and comments which are all stored in a locations database to provide supplementary information should Google's webservies API failed to supply all the necessary information. There is another reason for the creation of user accounts and the ability to post data, and that is because it is open for data analysis. All of the data that are saved in the database could be analysed which in return open for machine learning algorithms that identify patterns within the user posts or comments, and other sources of attributes from the users post such as location, vehicle speed, station locations and many more.

A particular algorithm that is of high interest is called the Naïve Bayes Classifier. Naïve Bayes is a collection of classification algorithms based on Bayes Theorem; A family of an algorithm that shares a universal principle that every feature that is classified is independent of the value of any other feature, Mike Waldron (2015)

The diagram illustrates the components of the Naïve Bayes formula. At the top, 'Likelihood' points to  $P(x|c)$  and 'Class Prior Probability' points to  $P(c)$  in the numerator of the equation  $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ . Below the equation, 'Posterior Probability' points to  $P(c|x)$  and 'Predictor Prior Probability' points to  $P(x)$ . Below the main equation, the joint probability formula is shown:  $P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$ .

User posts analysis could be categorised and judged accordingly, and sentiments that are to be labelled as good and bad reviews about specific locations. It may even use as a tool that analyses the locations database regarding interpreting the best possible routes. Subsequently, provides a smarter recommendation system to the users about public transport, regarding the best mode of transit on getting from location A to B., i.e. *It is better to take the ferry than the bus.*

## 6.2 Accelerometer and Vehicle positions

GPS locator was not entirely phased out in TARA, in fact, there are two planned future implementations for this as briefly mentioned in section 3.1.4, with the use of Google's Transit API and was the vehicle positions functionality. Vehicle positions are used to provide automatically generated information on the location of the vehicle such as from a GPS on board on a mobile device, Google Transit API (2018). The data from this will be stored in the locations database and analysed that identifies which areas in the location or within the geo-spatial map are a user travelling fast using accelerometer data. Reaching 25 mph could be a sign the user is inside a slow-moving vehicle.

Monitored and analysed on when or where exactly the user is currently travelling and where the car stops regularly and travelling which could, in turn, give essential transport information to users of when and where exactly to catch a bus. This functionality is particularly useful especially in countries that have less coverage from Google's webservices API. In instance busses or jeepneys in the Philippines could depart or stop at a random location without a station or a proper schedule. The functionality acts as additional supplementary information apart from the user's posts, and comments, as TARA, will be taking on-board this relevant technology that could be used to provide indefinite aid travelling tourist on their journey. Users post, and comments data may be unreliable or not available in specific locations, and vehicle positions data could bring valuable information in finding where precisely all forms of public transport are stopping and leaving.

### 6.2.1 Google Transit API and Vehicle Positions feed update functionality

In addition to the Google Transit API which returns the timetable and bus services based on the user's location, one another feature that TARA will contain is the knowing of when or where vehicles will arrive and depart. This functionality from Google Transit real-time will return the actual location of the vehicle via latitude and longitude coordinates and speed. Monitoring and storing the speed of a vehicle is an essential functionality of TARA as this will determine when or where exactly the bus departs as users can be informed of these locations. In the instance when travelling from Koronadal City (Mindanao, Philippines), the application will detect whether the vehicle has left the terminal. Furthermore, a typical low-speed vehicle such as a bus can run up to 25 mph (40 km/h). Liam Garrahan and Tom Herbert (2017) states buses shorter than 12 meters in length are limited to 50 mph on a dual carriage way, and up to 70 mph on motorways, buses longer than 12 meters are limited 50 mph. Any higher figures would mean that the vehicle is travelling to its destination. TARA analyses data from linear accelerometers and vehicle positions functionality to identify whether a user is inside a moving vehicle if it is around 0 – 9 mph then the user is considered walking or running. An optional functionality that will be incorporated in the future is the addition of trip updates, which is functionality from Google's Transit API called Vehicle Positions, and it identifies whether there are imminent changes to transport information such as trip cancellations.

## 6.3 User login interface change

Users requiring to login to use TARA may soon change, as the current design original assumption is questioned as to why the user expects to log-in to use TARA. User accounts are a highly considered functionality of TARA as this will provide the mechanism to enable users to post their statuses as each of these posts associates with the user account. However, this is not enough reason as to why the user has to login before use. It decided that future versions of TARA will no longer require users to login before application of the finding destinations functionality. The user will be provided with an option to login if they wish to use the social media elements functionality of TARA, this ensures that TARA is immediately usable as soon as the application it is tapped.

### 6.3.1 User Registration page

Non-registered users of TARA cannot sign-up for an account at this moment using the application, anyone that wants access will to inform the administrator who will create an account by accessing the user's login database API. User registration is reserved for development in the next sprint as the implementation of the registraton page was postponed to begin development of other major requirements of TARA.

Registered users will soon be able to create an account using the sign-up page. The user interface for this functionality was already built and is accessible via the sign-up link as soon as the user requires to login to use TARA. To registers, new users will require to input their username, email and password. Users will be identified by other users via their username when posting statuses and adding comments to other posts.

### 6.3.2 Users persistent login

Users logging into TARA will soon have a remember me functionality, this ensures that logging in for most members will only have to be done once. Eliminating the requirement of having to restart the application, which brings additional convenience when users are using the application.

## 6.4 TARA iOS deployment

As previously mentioned, the reason that the fundamental reason why TARA was built in React native as it is planned to be deployed in iOS in the future. React native allows to have one base code which will be rendered in both iOS and Android. Travelling tourists with iOS devices will soon be able to use the benefits of TARA.

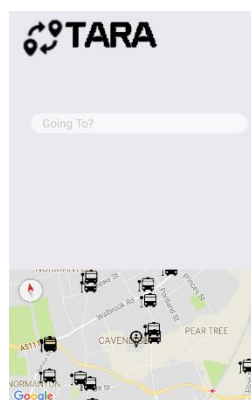
## 6.5 Several modes of possible public transport travel in Map View

At this stage, TARA only reveals bus station information and publishes bus schedules in any location. Future development plans include the consideration of other public transport travel, such as taxi, airplane, and rail travel. It is considered there are also other forms of public transport in various countries. For example, in the Philippines, the existence of tricycles and jeepneys are a common form of public transportation. TARA's future development will soon accommodate this particular mode of transportation and viewed in the map view.

## 6.6 Nearest restaurants, toilets, hotels?

As previously mentioned that this information can be supplemented by the user posts, which could include information about other details on the specific location that they have visited before. TARA will soon have an additional map view functionality that would reveal the nearest restaurants, toilets or hotels within the user's location. This information is also vital to travelling tourist apart from public transport information.

## 6.7 User location automation update



When a user views its location in map view, user's position will not update unless the application has been restarted. This user interaction design was decided due to ensuring that the user will only require viewing the application once to get the information needed. However, this poses an issue as the user expects to keep restarting the application to identify their exact location, this design is planned to be changed in the future development sprints of TARA, so the user's correct location can be viewed in map view. Using the *watchPosition ()* function which is part of the Geolocation library from React native that identifies the user location when the user location changes from their initial site. This functionality is currently not in use in TARA and is planned for deployment with other future updates.

## 6.8 Nearest bus station tap and reveals schedule

At this stage the only icons the only time a bus timetable would appear is when a user queries their desired destination. Future functionality includes the reveal of bus timetables of other bus stations within the user's area. The planned interaction is done by tapping the icon of other bus station in the area which in return would show its specific bus



schedule. Reserved in the next development sprint, as at this stage it is not possible to get travel schedule from Google API without using Google Directions.

6.9 Google Direction API unavailable transit coverage

The locations database will supply TARA with supplementary information if the Google Directions API fails to provide this information. TARA will attempt to look into the location database based on the location of the user. If the there are no timetables available from the Google API or locations database, the view will return a message *the timetable for this bus stop is not yet covered*.

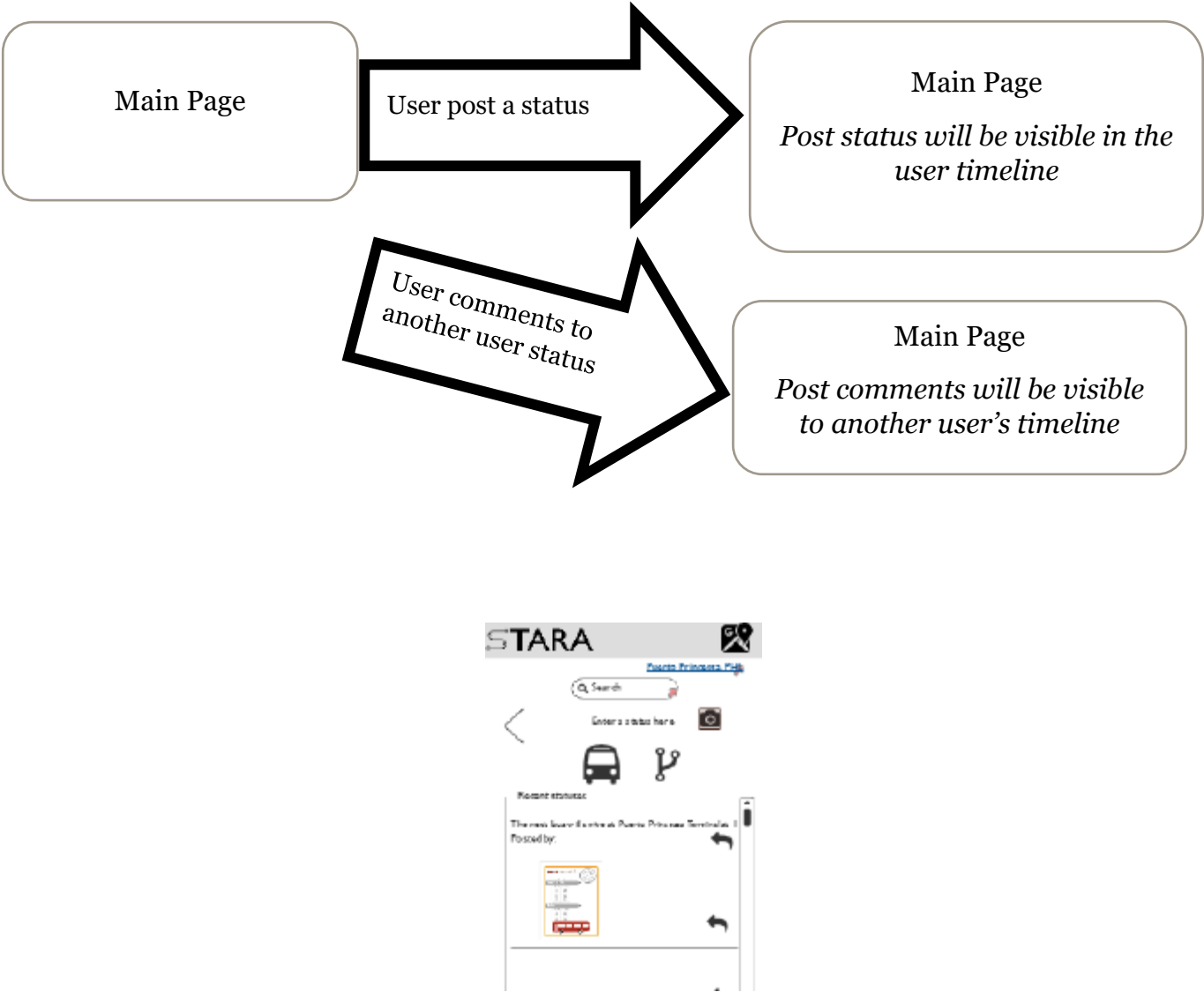


Figure 22: Users post state transition diagram

Registered users of TARA will be able to post status and post comments to another user’s status. There are no link or connections between users, but this may change in the future. The purpose is that TARA is using some social communication between users through the user of statuses and post. Users will be able to view additional information about the location through other users who are in the same location. Additionally, users will only ever see another user’s status of those who are in the same location. Subsequently, TARA does save all the status and comments in to the locations database which later becomes useful information should Google’s API failed to supply the information. TARA’s end-users will be able to provide supplementary information through post or other forms of data that was collected. Also, users should be able to attach or take a photo when uploading a status or when commenting on another user’s status as images of timetables; the vehicle could provide valuable information when travelling to a foreign country.

At this stage users post, and comments functionality was postponed until Google maps integration and location database integration tasks are complete. An essential feature as this is the supplementary information required to fill the attributes of the locations database and TARA has no users at this stage as the application is not live yet. This functionality requires an independent sprint development as it involves designing and analysis of how the users post statuses and view comments. It is planned to include users post functionality in the *Main* page along with a view of recent posts of other users, which only returns information depending on the current location of the user. Nevertheless, without this functionality, it is expected that Google has a vast coverage of transit detail location worldwide, which ensures that most users will be still able to use the benefits of TARA.

**User statuses** – The same as Twitter’s timeline or user’s post in social media sites like Facebook or Twitter, users will soon be able to post their travel status on their profile on the main page. Informing fellow users of TARA who are in the same location of the current affairs that are happening on their location. Statuses could be in any context about their location and information could come from the users status especially transit information where users can post statuses or

take pictures of a timetable, in the event Google webservices have no coverage about transit details on a specific location yet. Posts can also be questions as which will be viewable to other users and users, i.e. *Which station do I need to go to get to El-Nido Palawan Philippines from Puerto Princessa Philippines?*

**User Comments-** The purpose of the user comments is to socialise to other users posts and leave comments, which could lead to answering some of the questions or adding additional information about the users transit related posts. Conversation context could be of anything from transit or other information about the location such as *nearest restaurants in Koronadal City Philippines*.

The existence of a locations database will be an asset to TARA, as all of the user's post is unique to the particular location that they have visited, which will be saved stored in the locations database. Valuable supplyof additional information to the users should a specific location is not covered by Google's webservices API. Classed as a unique feature which uses actual data from the real users who have been on the location and knows a specific location better which could, in turn, provide valuable information.

### 6.9.1 Locations Collection Database

TARA has taken advantage of several Google's API, and as long as it is already a covered location by Google, the application will be able to access transport information about their current location. On the other hand, as mentioned previously mentioned, not all locations have reached the coverage of Google API. Therefore TARA contains a location information storage which also returns information about a specific location and triggered when the location of the user is known. The database utilises a non-SQL database called MongoDB, and the data is delivered to the front end via an API version of this database due to Django's rest framework capability. There have been many changes that were applied in to TARA's architecture since the original planning; these changes will be mentioned in section 3.3.

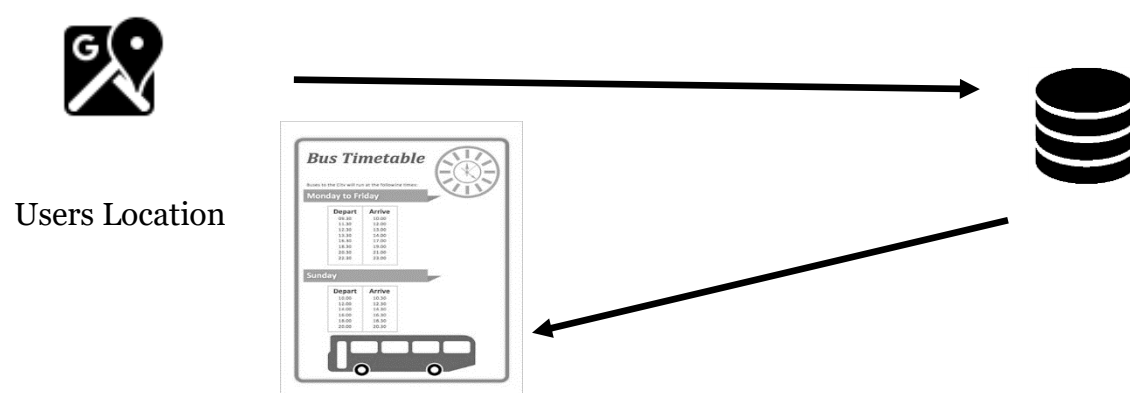
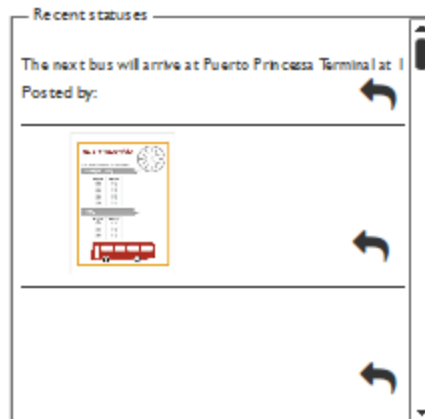


Figure 23: Location database integration

This database contains the user locations schema, a collection of TARA's location information. Django is capable of integration of more than one database, and in addition to the PostgreSQL that have been integrated, MongoDB is also integrated. This database holds all the supplementary information that has been collected by the application based on the locations that many tourists have visited. Apart from calling from Google's API, the application is also calling from this database, should Google's webservices failed to supply all the information. As previously mentioned, TARA will continuously be storing information from users such as their location, speed, and statuses. Data from the locations database could contain a timetable; this fulfils a known issue about Google's API location coverage of transit information as a timetable may exist but not yet covered by Google.

Non-existent data it is rather a case of undocumented data that is available that is simply not available on Google Maps yet, as Google looks to gather more data from transit companies worldwide. On the other hand, bus timetables may still existent on the location of the user or the worst-case scenario is that no timetable may exist, and it is unclear when exactly the bus will leave the terminal. It is a case in many South-east Asian countries like Malaysia or the Philippines, like many transportation locations such as bus terminals, ferries or jeepneys many not have a timetable as it leaves or arrives unpredictably. TARA's development has considered this issue and applies notable solutions to the following:

#### I. The existence of timetables and not-documented:



As previously mentioned TARA will have social media elements on board, the ability of users to supply supplementary information using user statuses and post functionality. Users of TARA can provide the much-needed information and share this information, so other users will be able to view their post, giving transport information of when and where to catch the bus or jeepneys. TARA aggregates and analyses this data into a central database and will later provide detailed information to users to aid them further when travelling. The use of user post and statuses should give extra confidence to the users as this information came from a fellow tourist who has been in the same location and shared transport information.

## **II. Non-existent timetables:**

Dave, (2011) States I can't think of anywhere else I've been with such diversity of public transport options as South East Asia, if you got wheels, you would probably find someone offering to give you a ride in it. The unpredictability of when or where to catch the bus or jeepney due to the unavailability of a timetable is one of the main issues. Apart from the users who will be providing user status, TARA will make use of existing technologies already mentioned in section 2.2.1. TARA will also aggregate and analyse this data into a central database of vehicle speed data, location and time to better identify the mode of transportation and identifying of when it left a certain location and the time of arrival by identifying the differences in speed. The information taken from this calculation will provide details to users of the most likely locations to catch the public transport.

### **User Connectivity**

TARA benefits from ensuring getting information from Google Maps API along with other of its API functionalities such as Google Maps, Places and Transit as well as supplementary data from user post and statuses if there is an internet connection. Google API services may not be entirely functional without internet connectivity as TARA's functionality requires to call Google's webserver at least once to provide data. This issue is then address separated into three separate cases.

#### **Internet Connectivity in countries that has better internet infrastructure:**

TARA is an application for travelling tourists it is considered that users will be connected at all times whichever or whenever travellers choose to travel. Due to cheaper data roaming costs from many mobile carriers, and many now do offer free roaming data around the world. Therefore most users will be able to acquire internet connectivity coverage wherever they are in the world. TARA users are then required to be at least on a Wi-Fi secured location at least once to synchronise data from the webserver, unless mobile data coverage is available. It is highly likely that tourists will be in a Wi-Fi area at some point, and this data will typically be locally stored on the mobile's cache memory, providing enough data for the user to query information.

#### **Internet Connectivity in countries with improving internet infrastructure**

TARA users will be required to be in a Wi-Fi secured area to supplement enough data to render for its users to view. Also, Google does not allow to cache any of its data as this violates their terms and conditions until recently. The new version of Google Maps API V2 will continue to work even if there are less connectivity compared to previous versions. It is then highly likely the connectivity will be lost for some users but not at all times and considered that tourists at some point connectivity would be resumed. TARA will ensure to provide information at all times even with less connectivity as enough data may be stored in cache memory to provide information to users.

Users of TARA will aid in collecting this information by posting a status, as supplementary information. TARA's integration to Google Maps webservices is successful, and it is expected that most tourist will have modern mobile devices to access the service which in return TARA renders and give the best view of this information to its users. Unfortunately, it

was not possible to fit the development of this on the current sprint of development for TARA. The locations database is currently not live; this specific task requires another development sprint which is planned to be next goal after the Google webservice integration is complete.

## [7] Conclusions

Overall TARA has managed to address the initial problems that the tourist is facing when travelling to a new location. The first question any tourist would ask when they arrived from an airport is public transport as TARA is designed and built to work on any location based on the coverage of Google's webservice API.

The following functionalities were achieved during the first three months of development sprint that addressed the following issues that are facing the tourist:

*Where am i?*

Upon login users, the application already identifies the actual location of the user via the geo-location functionality that returns latitude and longitude values.

*The nearest bus terminal or station?*

The nearest terminal, stop, or station is immediately available to the user upon initial access on the *Whereto* page of the application, the map view reveals the current location of the user and nearest stops in the area within the user's vicinity.

*Travelling to Swansea or at any location?*

Typing the desired destination in the text input within the *Whereto* page of TARA will provide line directions to the user between their locations and their desired destination. It also includes information of which station the users' needs to be in order to catch the bus the leads to their destination as well as the timetable.

The following achieved functionality have addressed the main essential features of TARA which is to provide immediate transit information, and enough to guide users where they will need to go on their journey. Google webservice API provides vast transit information worldwide. Therefore it is guaranteed that most tourist will be able to use the benefits of TARA as long as there is connectivity. Rendered that ensured that users are given enough information without overloading information about transit information.

Also, TARA is far from completion, and future development plans as mentioned will provide supplementary information to the users when travelling between different locations including the introduction of the social media elements into travelling. Tourist will no longer bother anyone from the countries they are visiting to ask for information, the application and its users of it who have been on a specific location before will provide the information. Furthermore, the application is planned to become smarter in the future one that offers suggestions and recommendations of which mode of public transit is best to take such as *the journey to El-Nido Palawan will take -1hr using the airplane*.

All in all, TARA is a perfect travelling companion solution for most tourist and occasional travellers, as it provides all the travel information wherever and whenever eliminating the need to download many travel applications. Many places have distinct travel application and saving on phone memory space. TARA aims to be the only travel companion application that will work in many locations.

## [8] Reflections on Learning

Throughout this project, continuous research was carried out, and it was the key to learning new technologies and integration of various API for developing modern mobile applications. I have also learned to justify the different technologies that were available and made critical considerations on my initial assumptions whether they are the correct solutions or if an imminent change was needed. This section of the report reflects on the mastery of new skills and application of existing skills that I have approached that lead to the current solutions when developing TARA. Four weeks was spent on research to understand the basic concepts of React native, differences between transportation API's and apply the newly learnt skills by implementing the core requirements of TARA. As mentioned in section 3.4, various decisions were decided to alter a different path from my original plan in the initial report, discussions with my supervisor included my deliverables over a 4-week timeframe of the current development task that I am undertaking.

Time constraints were a principal benefactor on my inability to implement all of the discussed requirements from the original plan and TARA has to be a functioning application within three months. Through the realms of best practise and



professionalism by knowing Agile development and testing as mentioned in section 3.4.4, it ensured development was not overwhelming as different tasks were aggregated into different sprints which lead to the successful deployment and testing of the core functionalities. The goal was not to guarantee faster development time rather than it was used to ensure that the initial core requirements are implemented without any dependency on large chunks of documentation; the implementation can begin immediately, followed by an initial deployment of the application. The plan for the future is to ensure that each new functionality that was implemented will be released on the next consecutive sprints. Subsequently, this assured that I had a manageable workload over the last three months with core functionalities deployed, if not entirely completed can be moved in to the next development sprint.

In addition, I have also made full justifications on the existing technologies that I have used to implement TARA and made a critical justification for the choices that I have made. In instance using react-native have ensured that no software dependencies have to be installed and saving computer memory on the need to install a simulator on my PC to deploy and test my code. React native and through Expo have ensured that my deployment was performed on the actual device, rather go through further dependency installations with Android Studio. I have also learned new concepts in developing new mobile applications as React-native is a new technology and is indistinguishable from an application built in Objective-C for Apple and Java in android devices. Developing in React-native ensures that my application is deployable on both platforms which is a future development plan for TARA, without the need for re-coding the application in any other languages.

Furthermore, I have entirely justified the diversion of TARA from the original initial plan as mentioned in section 5.3.2, that it was recently decided not to deploy social media elements as a core functional requirement due to the lack of data if this functionality was implemented. TARA needed vast and reliable transportation information which justifies the use of Google's webservices API as mentioned in section in section 2.4. This decision resulted in ensuring the use of an already open public data that are available, which is rendered in a way, that it is an aesthetically pleasing view to its users and avoiding information overload. Also, it ensured that TARA would have access to a vast collection of data from around the world providing the application will work in many locations within the UK and outside with dependency to Google's API coverage. I assured less development time in the future as Google is already doing the job on expanding their coverage as TARA's coverage will also extend to these locations allowing more users to use the application.

Finally, although the initial sprints did not guarantee the completion of TARA, I have managed to implement the core requirements which is sufficient enough to identify the purpose of the application, which is a travel companion application. TARA provides the essential functionality to the travelling enthusiast on giving directions to their desired destination with complete public transit information including timetables of bus terminals. I firmly believed I created a solid starting point for the application which ensures future development task is clear about what the application does and what it can do next.

## [8] Bibliography/References

- Barents Tourism (2017) *Accessibility and Transportation Infrastructure*. Available from: <http://www.barentsinfo.org/barentstourism/Accessibility-and-Transportation-Infrastructure>, accessed 1 March 2018
- Dave (2011) *Taking it Slow: Public Transport in South East Asia*. Available from: <https://whatsdavedoing.com/taking-it-slow-public-transport-in-south-east-asia/>, accessed 30 March 2018
- Diffen (2013) *Get vs POST*. Available from: <https://www.diffen.com/difference/GET-vs-POST-HTTP-Requests>, accessed from 15 March 2018
- Django (2018) *The webframework for a perfectionist with deadlines*. Available from: <https://www.djangoproject.com/>, accessed from 14 March 2018
- Djangorestframework2 (n.d.) *Serializers*. Available from: <http://www.tomchristie.com/rest-framework-2-docs/api-guide/serializers>, accessed 13 March 2018
- Google Maps Platform (2018) *Map Coverage Details*. Available from: <https://developers.google.com/maps/coverage>, accessed 5 March 2018
- Google Maps Platform (2018) *Welcome to Google Maps Platform*. Available from: <https://developers.google.com/places/>, accessed 3 March 2018
- Google Transit API (2018) *Vehicle Positions*. Available from: <https://developers.google.com/transit/gtfs-realtime/guides/vehicle-positions>, accessed 26 March 2018
- Here (2018) *Location for Developers APIs and SDKs for maps and location-aware web and mobile apps*. Available from: <https://developer.here.com/>, accessed 8 March 2018
- Mike Waldron (2015) *Naïve Bayes for Dummies; A Simple Explanation*. Available from: <http://blog.aylien.com/naive-bayes-for-dummies-a-simple-explanation/>, accessed 24 March 2018
- Peter Gazarov (2016) *What is an API? In English, please*. Available from: <https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82>, accessed 3 March 2018
- Ravish Rawal (2017) *Polyline Sorcery with React Native*. Available from: <https://medium.com/@ravishrawal/polyline-sorcery-with-react-native-8e194bd993f1>, accessed 22 March 2018
- Richard Tier (2014) *Django Rest Framework User Endpoint*. Available from: <https://richardtier.com/2014/02/25/django-rest-framework-user-endpoint/>, accessed 4 March 2018
- Sam Derring (2012) *Do you know what a REST API is?* Available from: <https://www.sitepoint.com/developers-rest-api/>, accessed 12 March 2018
- Sasaki (2015) *Graph Databases for Beginners: ACID vs BASE explained*. Available from: <https://neo4j.com/blog/acid-vs-base-consistency-models-explained/>, accessed 8 March 2018
- Sharif (2015) *Understanding the React Component Lifecycle*. Available from: <http://busypeoples.github.io/post/react-component-lifecycle/>, accessed 22 March 2018
- Tom Herbert (2017) *What does the law say on when slow-moving vehicles, caravans and tractors should pull over to let you pass?* Available from: <https://www.getsurrey.co.uk/news/surrey-news/what-law-say-slow-moving-13344965>, accessed 26 March 2018
- Transport API (2018) *The digital platform for transport* Available from <https://www.transportapi.com/>, accessed 5 March 2018
- Wern Anchetta (2018) *Easier React Native Development with Expo*. Available from: <https://code.tutsplus.com/tutorials/easier-react-native-development-with-expo--cms-30546>, accessed 12 March 2018