Individual Project
Final Report
London Loop: A mobile guide through the London Outer Orbital Path

Emma Hulme
Under the supervision of Professor Michael Huth

June 18, 2015

Abstract

The London Loop is a 150 mile signed walk along public footpaths, through parks, woods and fields around the edge of Outer London. Currently, there exists a guide book, a section on the Transport for London website, and some information from the walker of the London Loop themselves. However, we are in an age today where society breathes technology, where society searches online for an answer to a problem before their brain can even respond. So how is our technology driven society to find such beautiful scenery as can be found on a walk like the London Loop? My solution is therefore to create a smart phone app to effectively guide a walker through the London Loop walk, interactively taking the user step-by-step from their front door to the walk, through the walk, and back.

With careful research into the current methods used to guide a walker through the London Loop, and research into how I could practically make use of these methods in a smart phone app through the use of online resources, I was able to develop a smart phone app which could solve these problems, and provide motivation for more users to begin the London Loop walk.

Acknowledgements

I take this opportunity to express gratitude to my project supervisor, Professor Michael Huth, for his guidance and support. I also thank my parents and my partner for their constant support, encouragement and time throughout this project.

Contents

1 Introduction

1.1 Motivation

People are becoming more reliant on technology and less active these days. Our eyes are glued to static screens, mobile phones an extension to our hands. We are more content to sit around on social networking sites, playing video games, watching television than to go out and enjoy our surroundings. As a result of the change in lifestyles, the UK has seen a rise in obesity. The risk of type-2 diabetes and heart disease is increasing in children, with a third of children overweight by the end of primary school education [1]. Additionally, "more than a third of obese adults perceive themselves as being just overweight; a fifth of overweight adults think that they are healthy" [2], suggesting that people are unknowingly putting on weight.

Having seen this, my motivation for this project is clear: to encourage people to become more active. Despite London reaching a population of 8.6 million [3], 47% of the 1572 square kilometre city is green space [4], giving Londoners little excuse not to go out and exercise, and yet research shows that Londoners spend less time walking than anywhere else in the UK [5].

This is where I introduce the London Outer Orbital Path. Also known as the "London LOOP", it is a 150-mile walk along the edge of Outer London. It is divided into 24 sections in 3 groups - "blue" for South London, "green" for North-West London, and "yellow" for North-East London. Proposed in 1990 in a meeting between ramblers and the Countryside Commission, the London LOOP opened its first section in 1996, becoming fully accessible in 2001. The walk is fully sign-posted with the London LOOP logo, a flying kestrel, as seen on the cover page, and there is a guidebook written by David Sharp [6] which details each section of the walk together with photographic illustrations.

However, the question lies: how can I encourage people whose lives revolve around their gadgets and devices to go on a walk like this? In order to best answer this question, I decided to go on a London Loop walk myself. To prepare for this walk, I had a look for what information I could find regarding the walk and found the following.

- The London Loop Book [6]

  Pro: Comprehensive guide on the surroundings during the walk, and on points of interest.

Pro: Mentions all points of transport near the walk and during the walk.

Pro: Has pictures of certain landmarks and points of interest.

Con: Maps are difficult to follow.

- The London Loop Wikipedia page [7]

  Pro: Contains the geographic coordinates of each section start point

  Con: No information regarding the actual walk route

- London Loop Transport for London page [8]

  Pro: Provides nearest station to start and end

  Pro: Provided printable PDF files for each walk which described the route with a clear line on different map segments

  Con: Has "Did you know?" boxes telling the walker about different places during the walk

- Saturday Walkers Club [9]

  Pro: GPX/KML file containing entire route

  Con: Difficult to load GPX/KML file onto phone or device without third party software.

Instantly, I began searching on my smart phone ways to get to the walk, using the geographic coordinates from the Wikipedia page, aided by the guidance of the Transport for London PDF file. However, once I arrived at the beginning of the walk it was much easier to navigate my way using a combination of the London Loop signposts and David Sharp's book. By reading his book, I noticed that it was not so much there to show walkers the way to go, but to inform on what places of interest there were on the walk. In particular, it would should the walker an interesting place such as a castle or woods to detour to during the walk.

Having completed a walk with barely any technology, I realised that the best way to make the London Loop both more accessible and more attractive would be to create a smart phone application that could incorporate all the current resources into one useful and interactive guide for London Loop walkers. This leads me to the Project Proposal, written as follows.

> The London Loop is, according to Wikipedia, "a 240-kilometre (150 mi) signed walk along public footpaths, and through parks, woods and fields around the edge of Outer London".

There exists a standard book - by David Sharp - that describes about equal length walks on that loop, with descriptions of how to get to the start of the walk, where to stop for drinks and food, etc.

But there does not seem to exist a suitable smart phone app that walkers of the London Loop could use to plan walks, navigate walks, find out information about points of interest, etc.

The project will produce a workable smartphone app for a standard platform of the student's choice that will meet these requirements. Time permitting, the app could explore interaction modes with walkers that would not be possible through conventional means such as guide books.

1.2 Objectives

It is clear from my motivation and project proposal that the main objective is to create a smart phone application for walkers of the London LOOP. However, in order to clarify this, and then expand on the functionality and usability of this smart phone application, I have put together a list of core objectives.

1.2.1 Core Objectives

Obj 1. The interface must have a good design and be clear without any ambiguity.

Obj 2. The interface must be easy to use regardless of technical skill.

Obj 3. The app must be flexible and work on a variety of smart phones with different screen sizes, operating system versions and processors.

Obj 4. The app must not require user authentication to function.

Obj 5. The app must navigate the user to the walk of the their choice from their current location.

Obj 6. The app must navigate the user through the London LOOP walks.

Obj 7. The app must allow users to discover points of interest during the walk and learn more about them.

Obj 8. The app must point out places where users can stop for a break (e.g. a shop or restaurant)

Obj 9. The app must retain statistical data about the user's walk, such as time walked, distances walked, etc. so that the user can track their progress.

Obj 10. The app must clearly show the user when they are beginning and ending a walk.

Objectives 1-4 are clearly core elements that should be achieved in any well designed smart phone application, whilst objectives 5-10 expand on the project proposal for the application's functionality. However, if I were to complete these early and have time to implement more functionality, the following list of extended objectives includes what I would like the smart phone application to be able to do in chronological order of implementation.

1.2.2 Extended Objectives

Ext 1. The app could allow users to navigate from the end of the walk to a location of their choice.

Ext 2. The app could allow users to pause the walk midway through and pick it up at a later date.

Ext 3. The app could keep a count of how many users are currently walking on a particular walk, and how many walks have been completed by the app's users as a whole.

Ext 4. The app could allow users to save their data through either an email address or social media user authentication so that the information can be retained.

Ext 5. The app could allow users to input and update points of interest, restaurants, and shops, etc. and have the ability to include images (taken from either the smart phone's gallery or camera) - Any user uploaded data should be moderated on the server by an administrator prior to being visible to all other users to ensure no offensive data is uploaded.

Ext 6. The app could allow users to share their photographs and experiences online with social media.

Ext 7. If a user were to use authentication via email address or social media, then the user could be given an option to allow other users to see their position via the navigation view and contact them either via social media, email or in messages within the app.

Ext 8. Each walk should contain a "chatroom" style view so that walkers can talk to other walkers.

2 Background

Before beginning implementation of my smart phone application, I identified that some initial background research must be done before I could make some essential decisions regarding the application. In particular, I needed to make the following choice in technologies, tools and app-specific aspects of the project:

B1 Mobile Operating System

B2 Mobile Framework

B3 Online Server

B4 Database

B5 APIs

B6 Development Environment

B7 Version Control

B8 Software Development Methods

I also identified that it was essential to do some research on smart phone apps already on the market before I could make any of these decisions, and I did this by looking at major smart phone stores such as Google Play, iOS App Store and the Windows Store.

2.1 Existing Applications

Looking back at my objectives, I notice that the app is made of two main parts - Obj 5 and 6. The app must be able to navigate the user to the London Loop, and also navigate the user through the London Loop, with Obj 7 to 10 subsidiaries to Obj 6.

Therefore, with these objectives in mind, I separated my research into these two parts. Clearly, an important part of any app guiding a user from point A to point B is that it must be easy to use (Obj 5). In particular, to customize the journey plan to the user's own needs, taking into account any delays, cancellations or closures to provide a selection of realistic options to choose from. This is key to getting the user to the destination in the fastest time possible and in the way they want. For example, users who own specific travel cards which limited them to only certain modes of transport may want to filter their journey, and so a method of customizing the journey would be important to those users.

In order to efficiently compare apps currently available, I formulated a table of comparison of these aspects in different smart phone apps (Table 1), rating each aspect of the app up to 5 stars.

App	Ease of use	Transport Mode Filters	Interactive Navigation	Realistic Durations
Google Maps	☆☆☆	☆☆	☆☆☆☆	☆☆☆
Transport for London	☆☆☆☆	☆☆☆☆☆	☆	☆☆☆☆
Citymapper	☆☆☆☆☆	☆☆☆☆	☆☆☆☆☆	☆☆☆☆☆

Table 1: Comparison of applications that plan routes

Similarly, Table 2 shows a comparison of smart phone apps which work as an interactive guidebook, again with a rating of up to 5 stars.

App	Ease of Use	Planned Route	Location Tracker
TripAdvisor	☆☆☆☆☆	☆☆☆☆☆	☆☆☆☆☆
London Tours	☆☆☆		☆☆☆
MapMyWalk	☆☆	☆☆☆	☆☆☆☆☆
Split Audio Tours	☆☆☆	☆☆☆☆	☆☆☆☆
New York Pass Guide	☆☆☆☆	☆☆☆☆	☆☆☆☆
Sheep Spotter	☆☆☆☆☆	☆☆☆	☆☆☆☆☆
	Surrounding Information	Personal Stats	Social Media
TripAdvisor	☆☆☆☆☆	☆☆	☆☆☆☆
London Tours	☆☆☆	☆☆	☆☆☆
MapMyWalk		☆☆☆☆☆	☆☆☆☆☆
Split Audio Tours	☆☆☆☆		
New York Pass Guide	☆☆☆		
Sheep Spotter	☆☆☆☆	☆☆☆☆☆	

Table 2: Comparison of different Mobile Apps in relation to my requirements

2.1.1 Google Maps

Google Maps [12] is one of the most well known apps for navigation. It is useful for looking up different places and allows users to update information presented on the map. However, in regards to transport, it does not allow specific customisation of the user's journey, allowing you to choose your preferred method of transport, but not restrict it. On the other hand, this app is flawless as a navigation aid, planning a route on a map that the user can follow, with the option of turn-by-turn navigation where a voice guides you on your way and takes traffic and travel updates to give the best estimated duration of the journey as possible.

2.1.2 Transport For London

The Transport for London app [13] is the best app in terms of transport choice, giving users the option to filter the modes of transport, travel via a different place, specify accessibility requirements, and also to choose the maximum time the user is willing to walk. I found this incredibly useful as having spent the day walking one of the sections of the London Loop, I was able to choose a shorter route when I was no longer willing to walk any longer. Transport for London is also very useful in terms of status updates, giving information regarding any disturbances to the journey at hand, so that the user avoids any trouble in the journey.

2.1.3 Citymapper

I found that Citymapper [14] had the cleanest and easiest to use UI to use, giving users a variety of options, and anticipating that users may only want to use buses or trains, and so gives suggestions of the journey as such. Additionally, it provides suggestions for how long it would take by foot, bicycle, and taxi, alongside the estimated costs of the taxi. In particular, I found this useful as it showed me what the actual distance of the journey was in a quantitative form that I could easily understand - the time it would take to walk is much easier to compare than a number of miles a particular journey was. On the other hand, it also saw the potential in the journey, for example a journey that would be sped up by the future Crossrail train lines currently being built.

2.1.4 TripAdvisor City Guide App

I first looked at the TripAdvisor City Guide App [15]. Its design was simple, but the main menu had so many buttons, meaning that I had to scroll down to see all of them which was too much information to start off with. I felt that they could have broken down the menu into sections and subsections rather than grouping everything together, as many of the sections searched for something nearby, be it a shop, food or attraction. Despite this, the user interface was simple. The suggested itineraries were important as they provided planned routes that I could go on which would, for example, not miss any of the most famous parts of London. I found the app particularly useful as it used GPS tracking to help guide me on the route. Additionally, if I were to want directions to a shop nearby, I could click on “Point Me There” and it would guide me with an arrow, much like a compass, and tell me how far away I am.

2.1.5 London Tours App

My first impressions of the London Tours App [16] was not quite the same. It requires an update before each start of the application, which is not appealing to any user. However, once the app is up and running, its design is much more elegant than TripAdvisor's app. However, whilst it provided itineraries similar to TripAdvisor, it does not guide you with a map, preferring simple descriptions instead. It has more weight on the social media aspect, by asking if the user wants to add a photo or 'like' each attraction. The City Map tab of the app takes you to a great map of London, which isn't explained except for some icons at the top and seems like an interactive key for the map. The load time is very slow, and it is not clear what each icon does.

2.1.6 MapMyWalk

The MapMyWalk App [17] seemed the most elegant so far. There was a a clear menu that described each section of the app and I was able to create my own routes or use those created by other users. I could add friends via Facebook, Twitter, the contacts on my phone or find friends already using the app. I could record my "workout", and see when friends were on a certain route. This was a useful feature to consider for users of my own app to find others walking the LOOP.

2.1.7 Split Audio Tours

Split Audio Tours [18] was a different type of app also. It did not guide me in the same way as the other apps, but used audio guides to explain attractions, whilst allowing the user to see on a map where all the attractions are by number. This is a way of guiding users on a route that I had not considered before and may be a useful extension for the app to include.

2.1.8 New York Pass Guide

The New York Pass Guide [19] has a very clean UI, with sample itineraries, as well as the ability to create a personal itinerary. Within each attraction, there is the option to go to it on a map, find out specific details such as opening times, phone number and address, as well as the ability to "like" it. However, this is only within the app and does not allow the user to interact with any social media.

2.1.9 Sheep Spotter

Sheep Spotter [20], a paid app which guides the user around London and Bristol to find all the Shaun the Sheep statues scattered around. When using the app, it would notify the user if they came within close proximity of a Shaun, and so the user would be able to press a button so as to say that they have found that Shaun. On the other hand, if the app were not able to attain a location, the user could enter a 4 digit code instead of pressing the button as before. Additionally, the app would be able to tell the user where the nearest Shaun was and how far it was. It also had an arrow to direct the user which way to walk. As an app aimed towards the younger generation, I found this very easy to use, with a clean and colourful UI. In regards to planned routes, it had a list of routes the user could take to discover Shauns, with an estimated duration of a route, and it contained a personalised page where the user could see which Shauns they had found, and achieve trophies, like rewards, for finding Shauns in a certain manner, such as by walking backwards or completing different routes.

2.1.10 Conclusion

Looking at different apps gave me a good idea of how to proceed with my own app. I discovered the importance of a clean and easy to use UI, as it often hindered my experience of the app when the UI was difficult to use. Additionally, when registration is required, I am almost immediately put off using a smart phone app. This brings me back to Obj. 1-4.

In terms of Obj 5, I have established the importance of allowing the user to customise their journey, and to guide the user along the journey - but not requiring a location signal as users will often lose a signal, when being underground for example, during the course of the journey.

I have discovered plentiful ways that I could navigate the user through a walk (Obj 6), including to use a rotating arrow which would guide the user along the path. Additionally, I could bring up notifications when the user comes within proximity of a certain sight, so that the user does not miss out on it.

In regards to Obj 7 and 8, I could use some kind of notification or marker to show where users can rest and where places of interest are.

Looking at Obj 9 and 10, I was able to reflect on my use of the Sheep

Spotter app and in particular their statistics screen. This led me to consider collecting certain data, such as how many times a walk has been completed, how many hours people have spent walking, how many hours or miles all app users have walked collectively, and how many people are using the app. In addition to this, it should be clear that these statistics are being collected at a point where the user knows the walk is beginning, and knows when it is ending.

Having done this research, I could now begin to develop my own designs and make informed decisions about my choice in technologies, tools and app-specific aspects of the project meaning that I could begin implementation.

2.2 Mobile Operating System

In order to make a choice on mobile operating system, I must first look at what my options are. Doing so, I have compiled a table of different mobile operating systems, with Wikipedia as my aid [10], from both a technical and personal point of view. This is visible in Table 3

OS	Free and Open Source License	No Development Cost	Common APIs for smart phones and tablets
Android	Yes	Yes	Yes
iOS	No	No	No
Windows Phone	No	Yes	8.1+
Firefox OS	Yes	Yes	Yes
Blackberry OS	No	Yes	No
Sailfish OS	Yes	Yes	Yes
Mer	Yes	Yes	Yes
Tizen	Yes	Yes	No
Ubuntu Touch OS	Yes	Yes	Yes
OS	Familiar with Interface	Familiar Programming Language	Easy access to Devices
Android	Yes	Yes	Yes
iOS	Yes	No	Yes
Windows Phone	Yes	Yes	Yes
Firefox OS	No	Yes	No
Blackberry OS	Yes	Some	No
Sailfish OS	No	Some	No
Mer	No	Yes	No
Tizen	No	Yes	No
Ubuntu Touch OS	No	Yes	No

Table 3: Comparison of available Mobile Operating Systems

Having seen how much each mobile operating system varies, I concluded that the best choice for me was to develop my smart phone app for the Android operating system. This was due to my familiarity with the Android programming language - Java - but also due to the development cost being free, making the process of beginning development much quicker than, for example, iOS, where a membership is required to develop any applications. Additionally, as an Android phone owner, it made it much easier for me to test any Android app as I would not have to rely on only an emulator.

2.3 Mobile Framework

Following section 2.2, I know that I will develop my app in the Android Development environment. Therefore, with some online help [11], I formulated a table, as can be seen in Table 4, which lists the main aspects of each framework that supports the Android operating system.

Framework	Languages	Tools
Android SDK	Java, XML	Eclipse IDE, Android Studio Android SDK, ADT
PhoneGap	HTML, CSS, JavaScript	PhoneGap Development Tools, Development Code
Sencha Touch	HTML5, CSS3, JavaScript	Sencha Touch
jQuery Mobile	HTML5, CSS3, JavaScript, jQuery	jQuery Mobile
RHOMobile	HTML, CSS, JavaScript, Ruby	RhoStudio
Titanium Mobile	HTML, CSS, Javascript	Titanium Mobile
Corona	Lua	Corona SDK, Lua editor of your choice
Adobe AIR	HTML, JavaScript, Action Script 3.0	Adobe AIR SDK
Adobe Flex	Action Script 3.0	Adobe Flex SDK (Eclipse based)
Unity3d	JavaScript, C#, Boo	Unity3d
Flixel	Action Script 3.0	Flixel

Table 4: Discussion of Mobile Development Frameworks

The most well known choice would be the Android SDK, which includes its own emulator, reducing development and test time. However, there is also the option of using a JavaScript framework - PhoneGap and Sencha Touch are two of these, which allow you to build applications using HTML, CSS and JavaScript. Sencha Touch is more designed for user interface, whereas PhoneGap is useful to access aspects of the phone itself such as GPS, compass, camera, accelerometer, etc. Additionally, JQuery Mobile is an HTML5-based user interface system used for creating basic layouts, navigations, and so on.

RHOMobile, best suited for Ruby developers is based on the Model View Controller architecture, and includes offline capabilities using Rhosync which includes sync capabilities. On the other hand, Titanium is best suited for those with a background in JavaScript and Java, as it builds a bridge between the two in Android.

Based on FLASH and HTML/JS technologies, Adobe Air runtime enables you to build applications with HTML, JavaScript, Action Script, Flex, Adobe Flash Professional, and Adobe Flash Builder, giving a lot of flexibility. Similarly, Flex is a cross-platform tool to create Rich Internet Applications(RIAs). It can be used to build applications which run using Flash player or Adobe AIR and enables server integration with PHP, JAVA, Ruby, .NET, SAP backend.

For games, there is Corona SDK, Unity3d, and Flixel. Corona SDK includes inbuilt libraries giving access to all the necessary physics engines that games require. Alternatively, Unity3d itself is a games engine, allowing the creation of 3 dimensional games, and Flixel is an open source game-making library, best used for the creation of FLASH games.

However, I was not aiming to make a smart phone game app, and it made the most sense to use a framework which would be as familiar as possible to what I am accustomed to. I also welcomed the many tutorials and information available online and in books, especially as this was the first app I was to make. I also anticipated some learning curves in my development process. Therefore, it seemed to me that my choices were between Android SDK, PhoneGap, Sensha Touch and Titanium. Having seen that the Android SDK uses tools such as Eclipse IDE and Android Studio, which is based on Intellij IDEA, both of which I have experience in, it seemed that Android SDK made the most sense.

2.4 Online Server

To be able to update my application from an online server, I had to choose the architectural style of the web service, where I should host the web server, and what languages I should write the web service in.

In choosing the architectural style, I was left to decide between REST (REpresentational State Transfer), a simple stateless architecture that generally runs over HTTP, and SOAP (Simple Object Access Protocol), a messaging protocol that allows programs that run on disparate operating systems (such as Windows and Linux) to communicate using Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML). In particular, for the Android development environment, it seems that SOAP is not as well supported, requiring third party libraries to make calling SOAP Web Servers easier. Additionally, REST does not require as much bandwidth as SOAP, making it better fit to use over the internet. Therefore, I concluded that it

was an easy decision to choose a RESTful web server for my application.

For whom to host my web server, I was inclined to choose the Department of Computing's Apache Cloudstack, which I had used previously during my group project. This made it much easier for me to host my web server online as I already knew how the interface would work. In addition, it would not restrict me to a specific amount of space, api calls, or require a subscription fee to be paid, unlike services like Microsoft Azure and Amazon EC2.

Therefore, all that was left to be decided were the languages on my web server. The obvious choices for this are Python, PHP or Ruby. Having not had much experience in any of these before, I discovered online that Python is the easiest to learn, whilst Ruby is the hardest - but yet Ruby has the best usability. On the other hand, PHP has the lowest usability but ranked in the middle for ease of learning [21]. However, having searched online for "RESTful API Android", the most common search results involve PHP as the language used in the web service. Therefore, I concluded to use PHP as my language of choice, as it may not be as easy to learn as Python, but it had the most information provided online to help me in the creation of this web service.

2.5 Database

In terms of a database, the choice for internal database was clear - SQLite, as it is already integrated into every Android App. I also needed to choose the database that would host on the web service I created. Indeed, the Department of Computing's Apache Cloudstack allowed me to create an Ubuntu or Windows virtual machine, meaning that I was free to install any database tools I require. Therefore, I found my choices were between using SQL Server on a Windows VM, or MySQL on an Ubuntu VM. As I was most familiar with the Linux environment and command line, I found that MySQL was the best option for my application, with plenty of tutorials to follow. In addition to that, I also found that I would easily be able to connect a MySQL database to a PHP web service through the use of a free software tool, PHPMyAdmin - which can be installed and configured on a Linux virtual machine.

2.6 APIs

Throughout my app, I required the use of several third party APIs to attain information about the local transport, weather and to be able to include a

map interface in my app. Therefore, I made a comparison of these APIs, as seen in Tables 5 to 7.

API	Price	Filter	Format
Google Maps	Free	No (Preferred only)	JSON
Transport for London	Free	Yes	JSON
Transport	Free	Yes	JSON
Citymapper	Free	Yes	Open in app

Table 5: Comparison of Map and Directions API

API	Limit/Cost	Accuracy	Format
Yahoo Weather	2000/day	Geographic Coordinates	JSON/XML
Accu Weather	Premium	Postcode	JSON
Met Office	2000/day	Geographic Coordinates	JSON/XML
Forecast.io	1000/day	Geographic Coordinates	JSON
OpenWeatherMap	1200/minute	Geographic Coordinates	JSON/HTML
WeatherBug	Premium		

Table 6: Comparison of Weather API

API	Limits	Active Users	Format
Facebook	Free	1.44 billion	JSON
Twitter	15-180/15 minutes	302 million	JSON
Instagram	5,000/hour	300 million	JSON
Tumblr	1000/minute	206 million	JSON
Google Plus	10,000/day	111 million	JSON

Table 7: Comparison of Social Media API

For the simple map API, it was clear that Google Maps would be the best option, as it was easy to integrate into an Android app as an activity already incorporated into the Android interface. However, as an API for transportation, it did not allow users to filter their journey. On the other hand, although Citymapper was the best app, as discussed in section 2.1.3, its API would only allow the user to open the route in their own app, and thus was not as useful. Therefore, I was left to decide between the Transport for London and Transport API. As the Transport API pulls data from both Transport for London and Citymapper, it made the most sense to use Transport API as it retrieves information from several sources.

For a weather API, I could see from table 2.6 that OpenWeatherMap has the best API for my app, allowing up to 1200 API requests per minute. Additionally, OpenWeatherMap has the clearest online documentation, which should make it much easier to set a request for the weather information using the geographic coordinates of each London LOOP section.

Similarly, from table 7, it was clear that Facebook had the highest number of active users out of any social media, followed by Twitter, Instagram, Tumblr and the least used social media site, Google Plus. Since the limits were all extremely high, and the return of the APIs are all JSON, it made sense to use Facebook, the most popular social media application in the world.

2.7 Development Environment

Despite being an Eclipse user since my first year studying, I decided that it was worth looking into other options. Android Studio, the main competitor to Eclipse in Android development tools, is based on the IntelliJ IDEA platform, and allows the user to create activities from a list available, and additionally integrates the Gradle set of build tools. This meant that, as someone who has never created an Android app before, I had a set of tools readily available which does not require installed plugins and add-ons to work. With a clear user interface, Android Studio also allowed me to easily run and debug my app directly onto my phone or through an emulator.

2.8 Version Control

As with any project, it was necessary to use a revision control system. As Android Studio can be integrated with git, I had chosen to work on GitLab [25]. This is a revision control system which I was accustomed to throughout use in my course, and allowed me to easily back up my work and keep different versions of it so that there was always a current working version. To begin my project, I have created a group on GitLab, "london-loop", and a project within, "londonloopapp" [26], where my project repository resides.

2.9 Software Development Methods

It is important before starting any project to work, it was necessary to have a strategy to planning all the workload, and to keep track of work progression.

I decided to use the Iterative Development Model [27], as can be seen in Figure 1, to help manage the project.

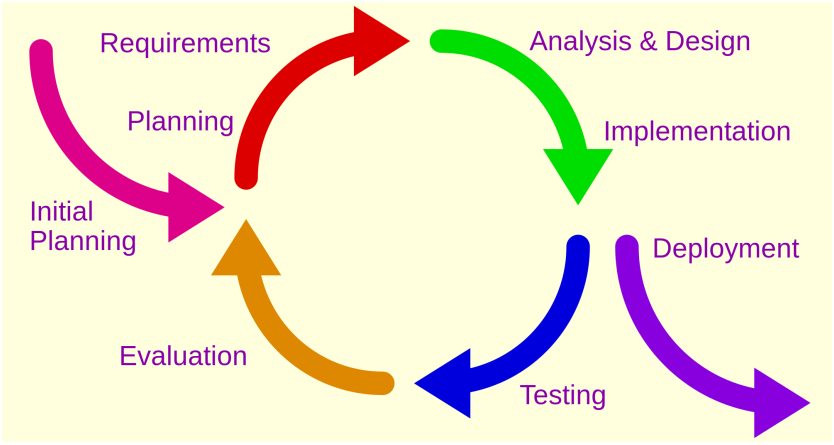


Figure 1: Iterative Development Model

Therefore, I completed my initial planning, where I examined the existing data and decided what my objectives are, as already seen in section 1.2. Following this, I created a full list of requirements, or features, which I will implement in my project. This includes a set of initial designs to be used in my app, and a simple timetable of how to complete my objectives. Then, having analysed and redesigned my ideas until I have a clear understanding of what I want my app to include, I was able to begin the implementation phase. In order to manage my progress, I used a web-based project management application, Trello [28], modelled by the kanban scheduling system, to manage tasks and personal deadlines so that I could stay on track with my project, and adapt my goals and project plan if required. Additionally, once I began implementation, I also began testing my application, and evaluation of any hurdles I come across.

2.10 Summary of Background Research

Having completed my initial background research, I come to the following conclusions for choice in technologies.

B1 Mobile Operating System - Android OS

B2 Mobile Framework - Android SDK

B3 Database - SQLite and mySQL

B4 Online Server - Apache Cloudstack

B5 APIs - Google Maps, TransportApi, OpenWeatherMap, Facebook

B6 Development Environment - Android Studio

B7 Version Control - GitLab

B8 Software Development Methods - Iterative Development Model

This left me in a position to begin creating the design and development process to my project.

3 Specification and Project Plan

3.1 Specification

To begin designing my app, I first settled on a main list of features for my app as follows.

1. List of London LOOP walks
   (a) Shows user which region of London the walk is in
   (b) Details the distance of the walk and the estimated time it would take to complete
2. Map of London LOOP walks
   (a) Marker showing walk locations
3. Detail of London LOOP walk
   (a) Information about the walk
   (b) Suggestions for attire for the current day's weather conditions
   (c) "Start" button which prompts you to be navigated to the walk's start point or to start the walk
4. Navigation to start point of a walk
   (a) Filters route as required by the user
   (b) Gives user different options to take
   (c) Guides user to the route, updating the guide as the user moves forward
   (d) Allows user to choose whether to start walk straight away once they have reached the start point for it.
5. Navigation through walk
   (a) Guides user to the route, updating the guide as the user moves forward
   (b) Shows user nearby points of interest, places of rest, shops, etc.
   (c) Allow user to choose whether to start next walk straight away or navigate somewhere else.
6. Navigation from end of walk

(a) Filters route as required by the user

(b) Allow user to choose type a postcode or station name as a destination.

(c) Guides user to the destination, updating the guide as the user moves forward

7. Statistics Page

(a) Give user information of their progress - walks completed, time taken, miles walked

(b) Gives user information of global progress - people currently walking, walks completed by everyone, time taken by everyone, miles walked by everyone

These can be summarised in figure 2.

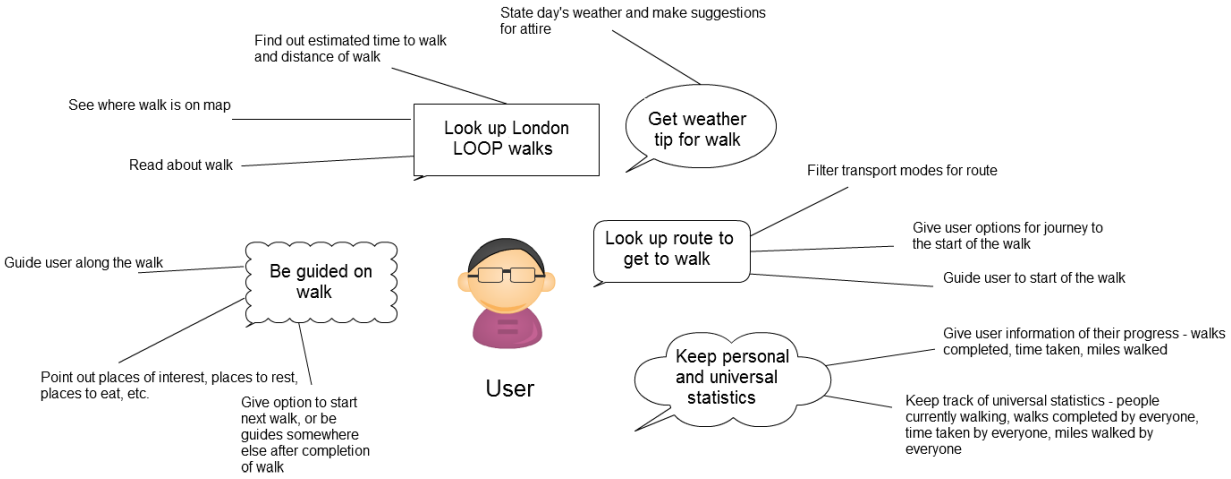


Figure 2: Functionality of App

Additionally, I created a list of extensions which expand on the core functionality of my app as follows.

1. See how many times each walk is completed

2. Allow the user to pause a walk and continue it at a later date without losing information.

3. Store user information online so that personal statistics can be retrieved on a different device.

4. See how many people are currently walking on a given walk
5. Progress bar so that user knows how far they are through the walk
6. Social Media Interaction
   (a) Check-In Facebook Button
   (b) Share photograph with Facebook
   (c) Find others using the app
7. Voice navigation

3.2 Project Plan

Following my background and design sections, I established a clear understanding of how I wanted my app to behave. Therefore, I created a timetable, as seen in figure 3 showing where I wanted to be in the time up to the dates I hand in my project.

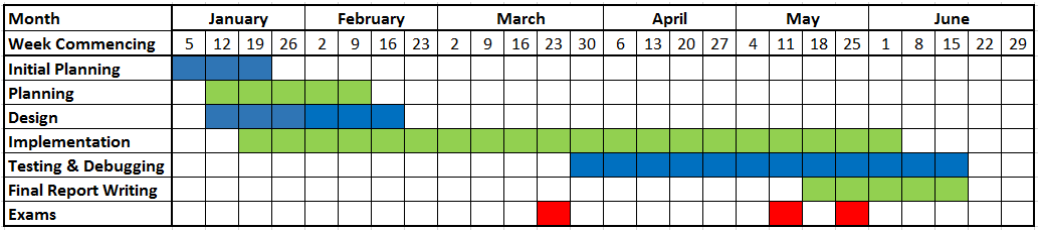


Figure 3: Project Plan

In my timetable, I gave myself 6 weeks to plan my project, 7 weeks to design my project, and 20 weeks to implement it. Following my first week of exams, I was to begin testing and debugging my app, whilst still implementing the project. In terms of my final report (this report), I planned to begin writing in the week of the 18th of May, the week between my two last exams. I chose this week, as I would be primarily revising for my final exam, and so writing would be the least taxing part of my project to do.

Having created an overall timetable, I also realised it would be important to create a detailed timetable for my implementation, as seen in figure 4. Here, I allowed myself a 2 week break prior to my computing exams on the week commencing the 23rd as I had 3 computing exams. On the other hand, I did not plan to stop working during my maths exams as I had only 2 maths exams on different weeks of May. I hoped to complete as much of the core

features during my Easter holidays, so that I don't have too much implementation to do following my exams and would have time to implement some extensions to my project.

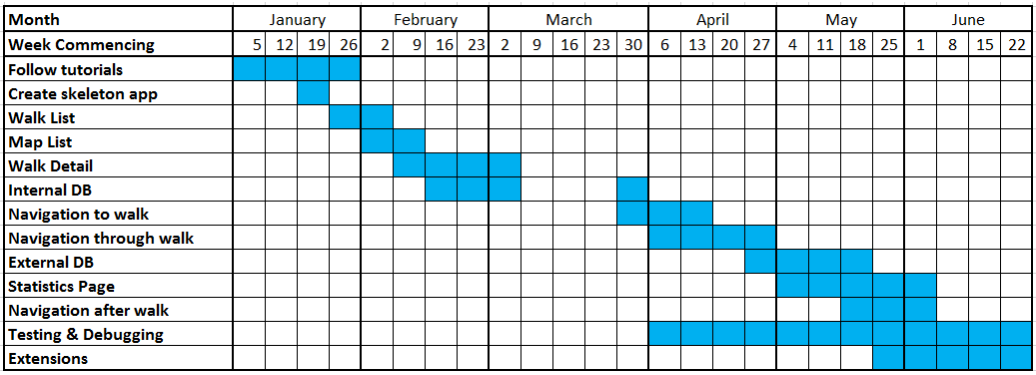


Figure 4: Detailed Implementation Timetable

Additional to my project timetables, I have also used a kanban scheduling system to manage my project - Trello. "Projects are represented by boards, which contain lists (corresponding to task lists). Lists contain cards (corresponding to tasks). Cards are supposed to progress from one list to the next (via drag-and-drop), for instance mirroring the flow of a feature from idea to implementation. Users can be assigned to cards. Users and boards can be grouped into organizations [29]." How I used this can be seen in figure 5.

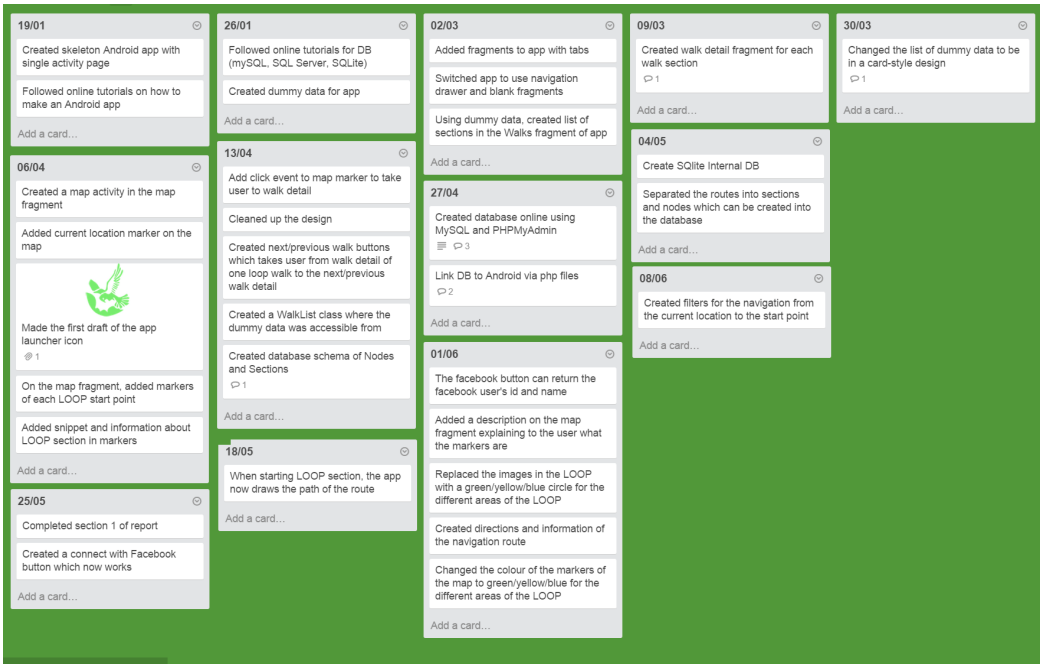


Figure 5: London Loop Project Board [28]

Following section 2.8 Version Control and 2.9 Software Development Methods, I intended to fully utilize Git's support for branching and merging in the development process. I created a new branch for a particular section or feature of my project, and merged into the 'master' branch when that feature or section was fully functional and working. In this way, I always have a working iteration of my project on the 'master' branch.

I also held weekly meetings with my supervisor, Professor Michael Huth. This helped me to aim to have a new iteration of my app complete before each meeting so that I could show how my project was progressing.

4 Design

4.1 Software Design

4.1.1 Web Service

For my web service, I wanted to create a CRUD RESTful PHP Web service. Therefore, I used the model in figure 6 to visualise how the client side (smart phone) interacts with the web service.

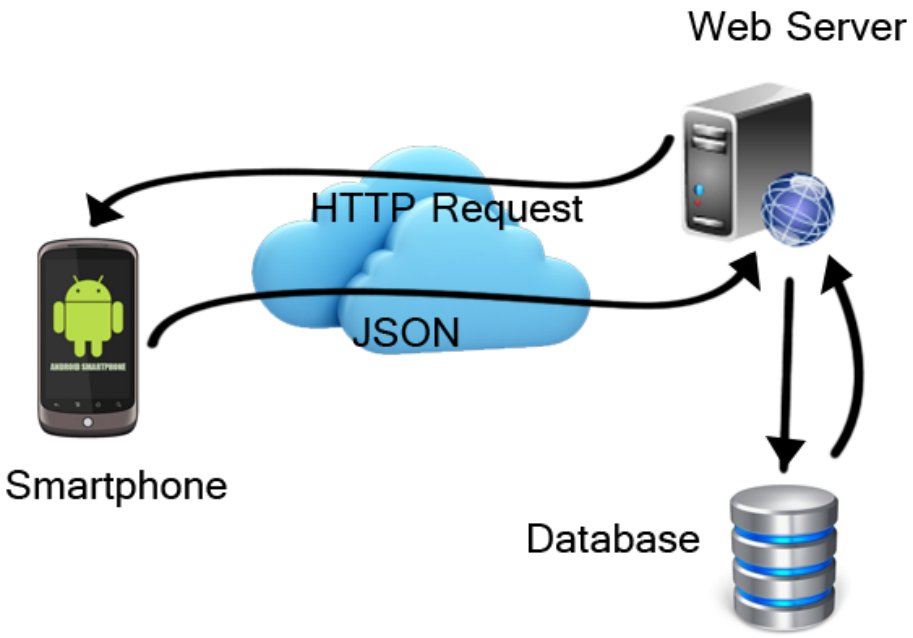


Figure 6: RESTful Web Service

Additionally, to ensure that every API request is successful, the JSON response will be issued with a status code as follows.

Status Code	Meaning
0	Success
1	Failed retrieving data
2	Failed updating data

4.1.2 Database Design

I designed a schema for each of the database tables as can be seen in table 8

Table	Name	Type	Definition
Node	NodeId	int	primary key
	Name	text	name of node (e.g. Erith, Old Bexley, ..)
	Latitude	real	latitude of node
	Longitude	real	longitude of node
Section	SectionId	int	primary key
	StartNode	int	id of start node
	EndNode	int	id of end node
	Description	text	description of walk section
	Length	real	length of walk in miles
Gps	GpsId	int	primary key
	GpsNo	int	incremental number of gps item within its section
	GpsLat	real	latitude of gps coordinate
	GpsLong	real	longitude of gps coordinate
	GpsSection	int	id of section that gps item corresponds to
	GpsNote	text	note (can be null) that tells user where to go next
Marker	MarkerId	int	primary key
	SectionId	int	id of section that marker corresponds to
	Latitude	real	latitude of marker
	Longitude	real	longitude of marker
	Type	int	0 for place, 1 for shop/restaurant, 2 for seating area
	Url	text	url which will open when info window is clicked
Statistics	WalkId	int	id of section the statistics corresponds to, 0 for all
	CurrentlyWalking	int	number of people currently walking section
	WalkTime	int	time spent walking in minutes
	MilesWalked	real	miles walked

Table 8: Database Schema

4.2 User Interface Design

Before embarking on designing the visual aspect of my app, I first had a look at current design strategies for smart phone apps, as well as the branding for the London LOOP itself. In particular, I have found that Google's Material Design is the most comprehensive guide as well as catered towards the Android system.

Google's Material Design states their goals as "Create a visual language that synthesizes classic principles of good design with the innovation and possibility of technology and science" and "Develop a single underlying system that allows for a unified experience across platforms and device sizes. Mobile precepts are fundamental, but touch, voice, mouse, and keyboard are all first-class input methods." In this way, Google has outlined key principles for style (e.g. colour, typography, imagery), layout (e.g. structure, measurements), components (e.g. buttons, dialogs, lists), and patterns (e.g. navigation drawer, scrolling techniques). In designing my smart phone app, I will bear all these guidelines in mind so that I can produce the best experience possible for the user.

Looking at the available material for the London LOOP online and in books, I find that the London LOOP is signposted with the logo as seen in figure 7.

Figure 7: London LOOP sign

The flying kestrel, the symbol of the London LOOP is also seen on Transport for London's London LOOP website, as seen in figure 8.

Figure 8: TFL London LOOP Logo

From figures 7 and 8, I designed my first draft of the logo I wanted to use for the app - this would be the launcher icon users seen on their smart phone's home screen or 'App' folder. This can be seen in Figure 9, where I have taken the kestrel from figures 7 and 8, and made that my logo.

Figure 9: First draft of my app's logo

However, due to the detail, I noticed that the icon may be quite ambiguous when overlayed on different backgrounds. In order to check this, I have taken the icon and placed it on top of a selection of backgrounds [31], as seen in figure 10. However, one can clearly see that the icon appears well on contrasting colours, but is not as easily visible on similar colours such as the yellow and blue backgrounds.

Figure 10: My first draft logo on different backgrounds

As a result, I turned to research alternative ideas for the logo, and came across a logo designed by a third party [32], as seen in figure 11.

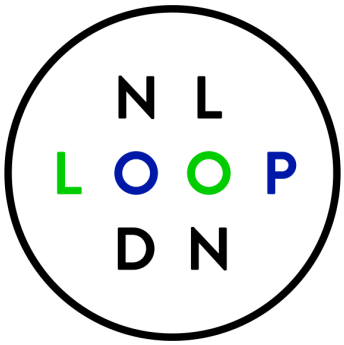


Figure 11: London LOOP Logo designed by Tudor Prisacariu

Therefore, I wanted to make a logo that was more contemporary and minimalistic, like figure 11, and so, taking the outline of the London LOOP (Figure 12), and taking the outline of the river Thames across London, I created a second draft of my logo (Figure 13).

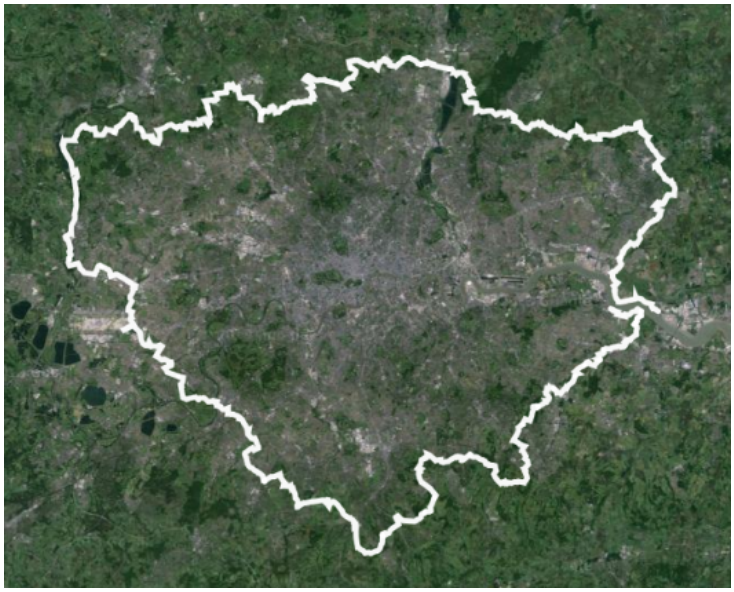


Figure 12: Outline of London LOOP

Figure 13: Second and final draft of my London Loop app's logo

Checking again if my new icon would be clearly visibly on various backgrounds (Figure 14), I could see that this new icon is much clearer and unambiguous.

Figure 14: My first draft logo on different backgrounds

Therefore, I concluded that this is the best icon to use as my app's logo.

The next step in creating the branding for my app was to create the name of my app. The maximum number of characters for an Android App is 12, and as "The London Loop" is too long at 15, I had to come up with a shorter title for my app. I went through names such as "LOOP", "London Loop", and "LOOP Walker". However, I settled for "London Loop" in the end, as it was clear and simple.

4.3 Mock Mobile Design

Following the list of features in section 3.1, I created a series of mock designs for my app, as seen in figures 15 and 16.

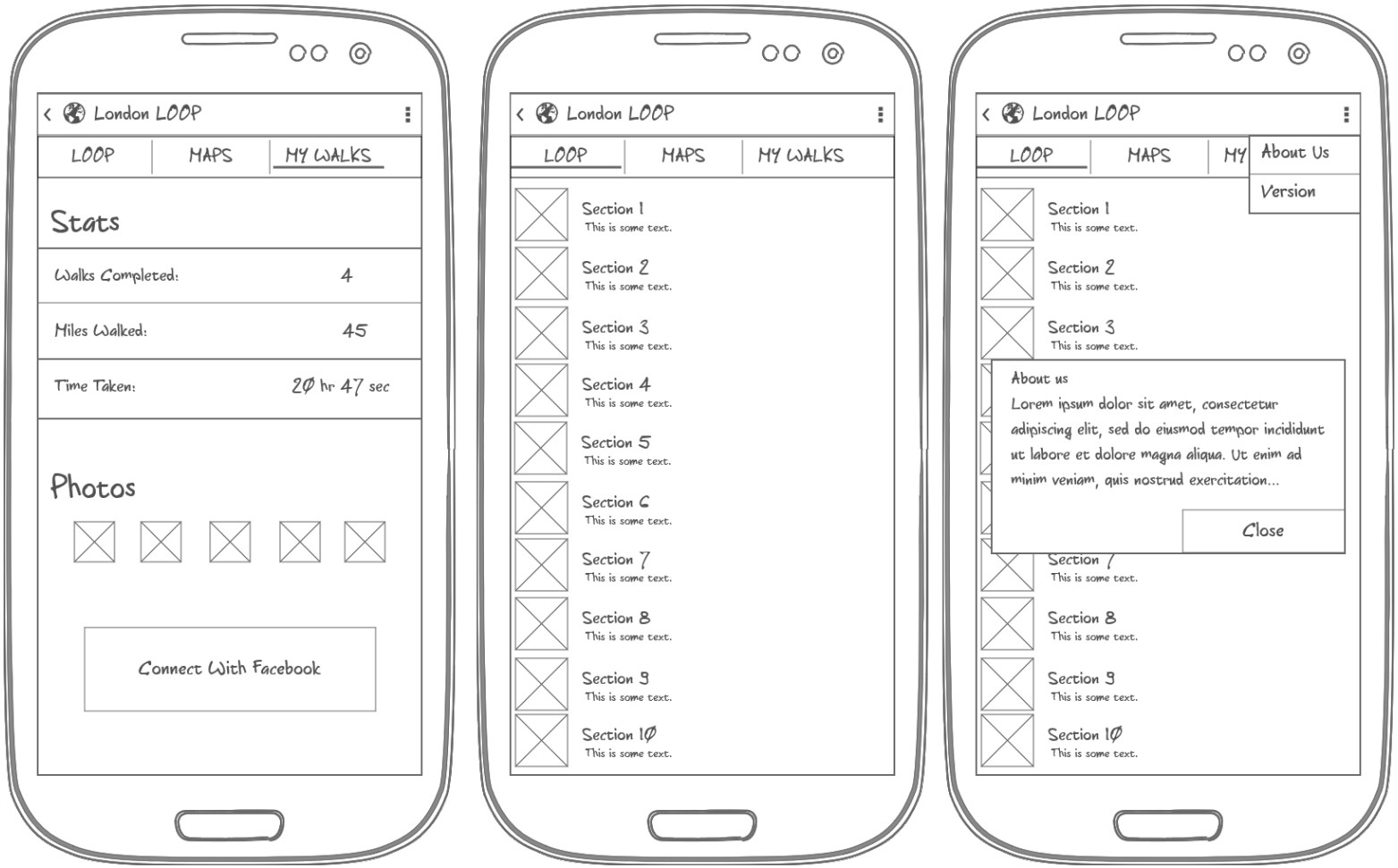


Figure 15: London LOOP Mockups

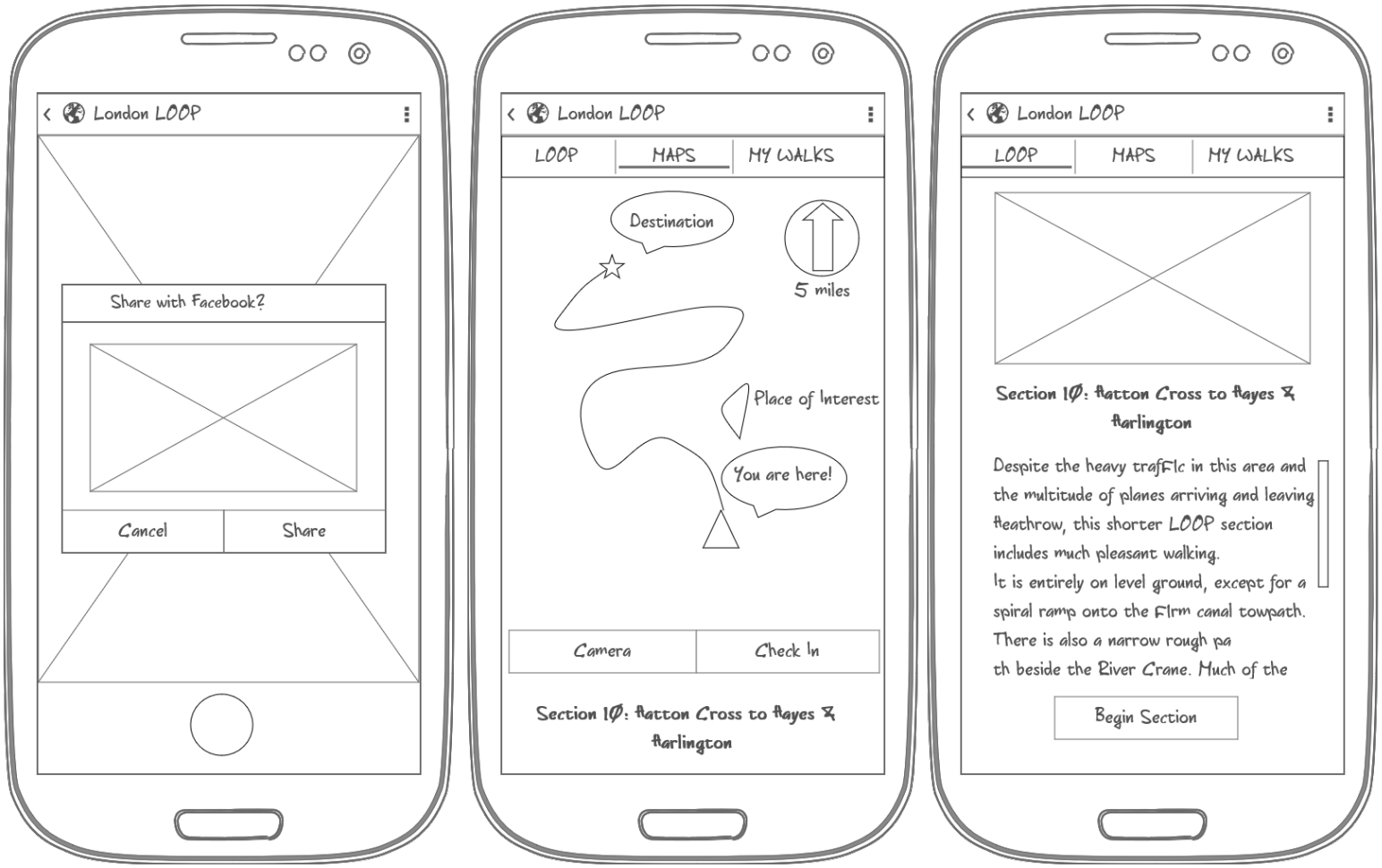


Figure 16: London LOOP Mockups cont.

However, as these are only my first designs, created prior to beginning implementation, I expected to make changes as I went along with building my app. Figures 17- 21 show the mockups I made later on in the project.

Figure 17 shows the navigation drawer I will have, which is a contrast to the tabbed buttons I had in my first mockups. However, I realised that it would be unnecessary to have tabs on the screen at all times because it is unlikely that the user would need to switch tabs once embarking on a walk.

For the list of walks, I changed the user interface to have a card style design, conforming to Google's Material Design, and added an icon letting the user know which part of London the walk is in - blue for South, green for North West and Yellow for North East. For the map of walks, I used different coloured pins, again with blue for South, green for North West and Yellow for North East, to visualise where the walks are in London.

The cards in the list of walks, and the pins in the map of walks would then take the user to see the detail of the walk, as seen in figure 18. Here, the user would be told a short description of the walk, as well as be given weather information about that area so that the user can be prepared for the walk if they choose to go on it. Once the user clicks start they are taken to the "What do you want to do?" page where they can either choose for the app to take them to the start of the walk, with filters of mode of transport, or to go straight into the walk if they are already there. If the user chooses to ask the app to plan a journey to the walk, then it would show the user the different options they can take. When the user clicks on one of these options, they are taken to the screen shown in figure 19. Here, the app has planned the journey on a map interface, and contains an information section below which not only tells the user where to go, but also updates this information as the user moves forward, and allows the user to skip forward or backward in the journey in the case that a connection has been lost, for example on the underground.

Once the user has reached the start of the walk, the app would prompt them to start the walk, allowing the user to choose yes, or no if they wish to have a break prior to starting the walk. In that case, they can come back to the app when they are ready and click on the button "I am at the start point".

When the user begins their walk (figure 20), they are guided in much the same way as the journey to the walk, with the added feature of markers on the map that show points of interest, and a checked flag showing the end so

that the user can see where the walk ends. Once they reach this end, they are given the option of what to do next. They are then given a similar option to before they start the walk where they can type a location for the app to guide them to, or, if they feel up to it, continue the loop and start the next walk.

The final page on the app is the statistics page (figure 21) where the user can see information about their progress on the London Loop, how many walks they have completed, how long they have spent walking, how many miles they have walked, plus additional information about how many other people are walking at the moment, how many walks people have completed, and so on. Additionally, by pressing the 3 dot button on the action bar, the user can get more information about the London Loop and the app.

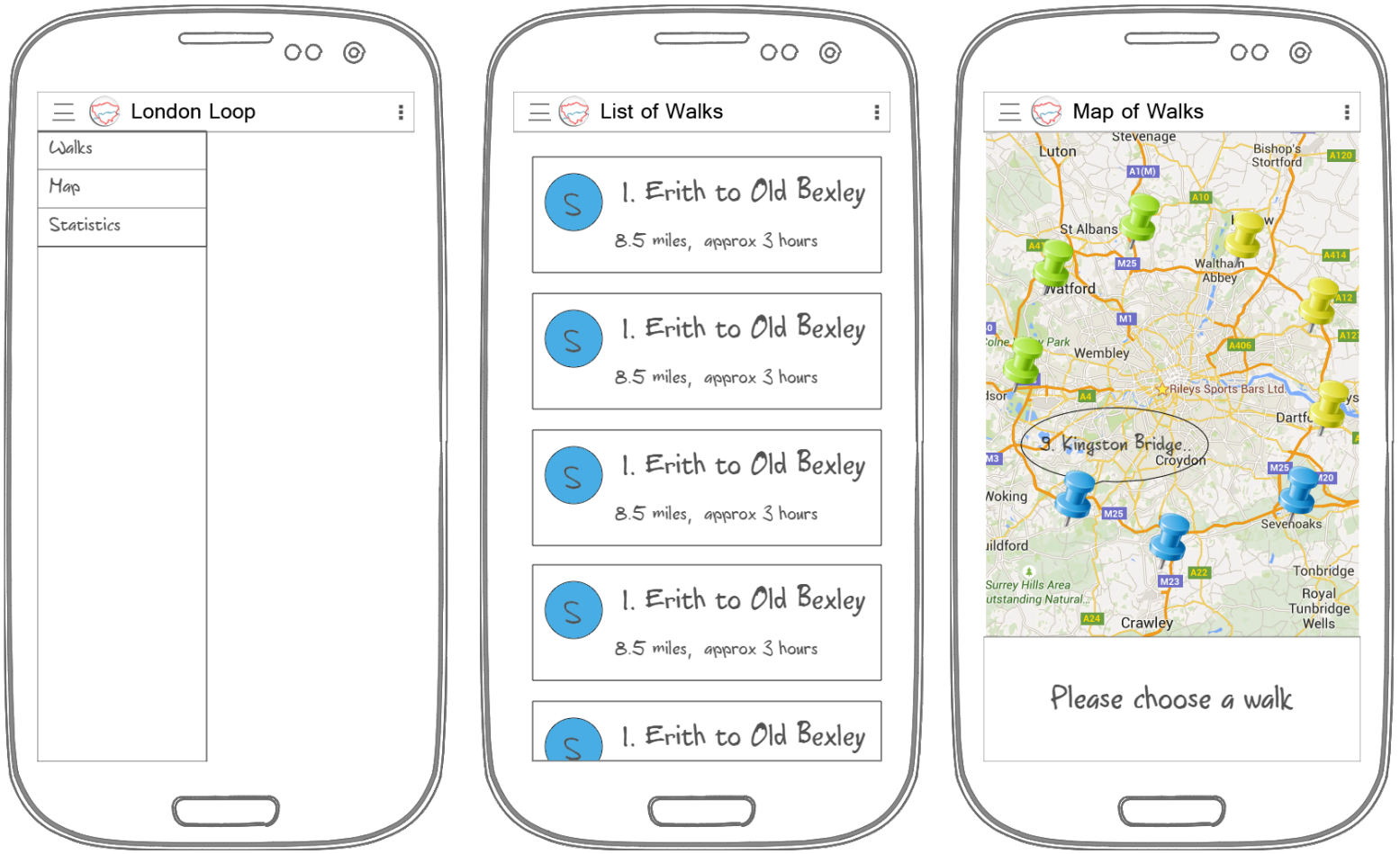


Figure 17: London LOOP Mockups

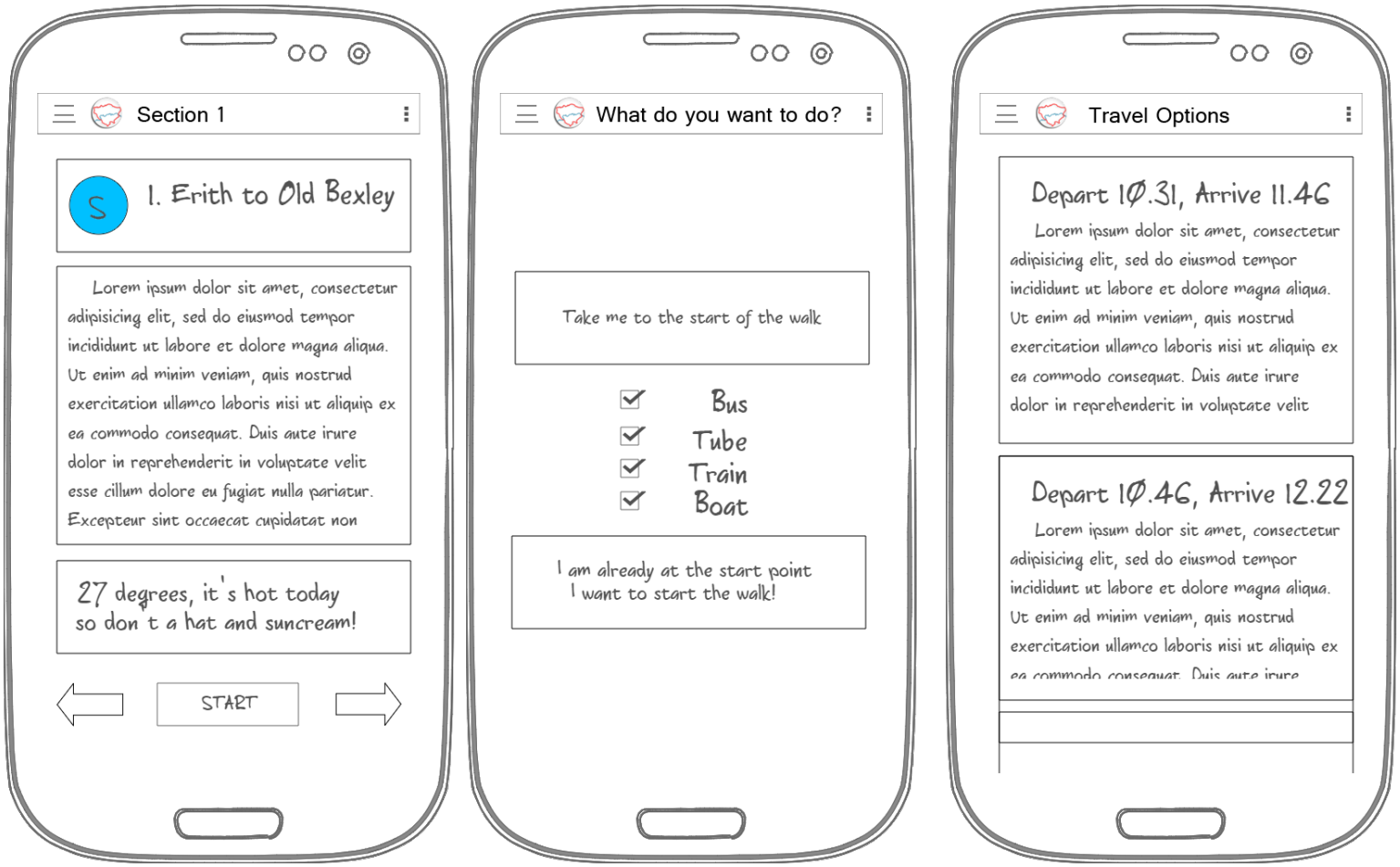


Figure 18: London LOOP Mockups cont.

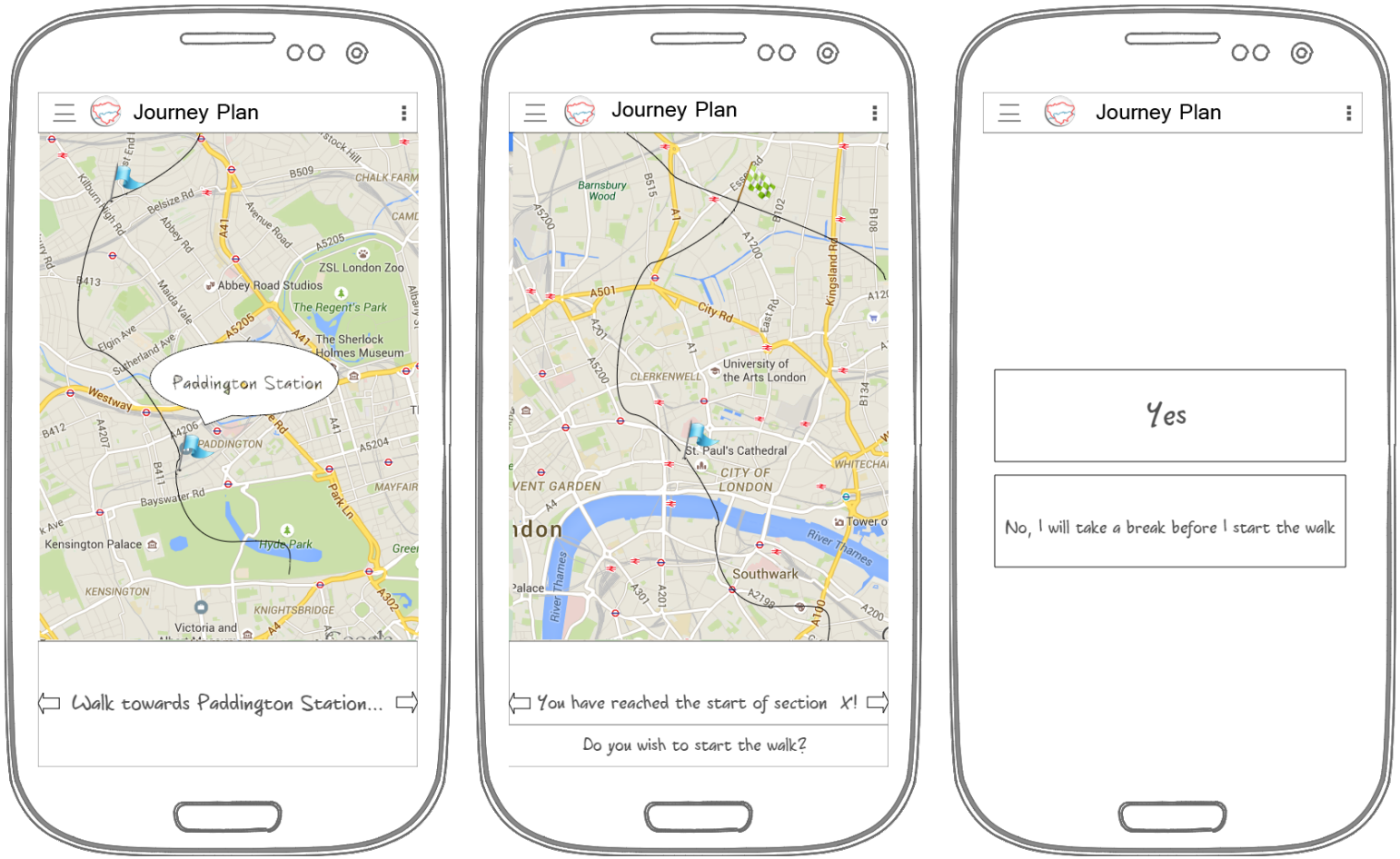


Figure 19: London Loop Mockups cont.

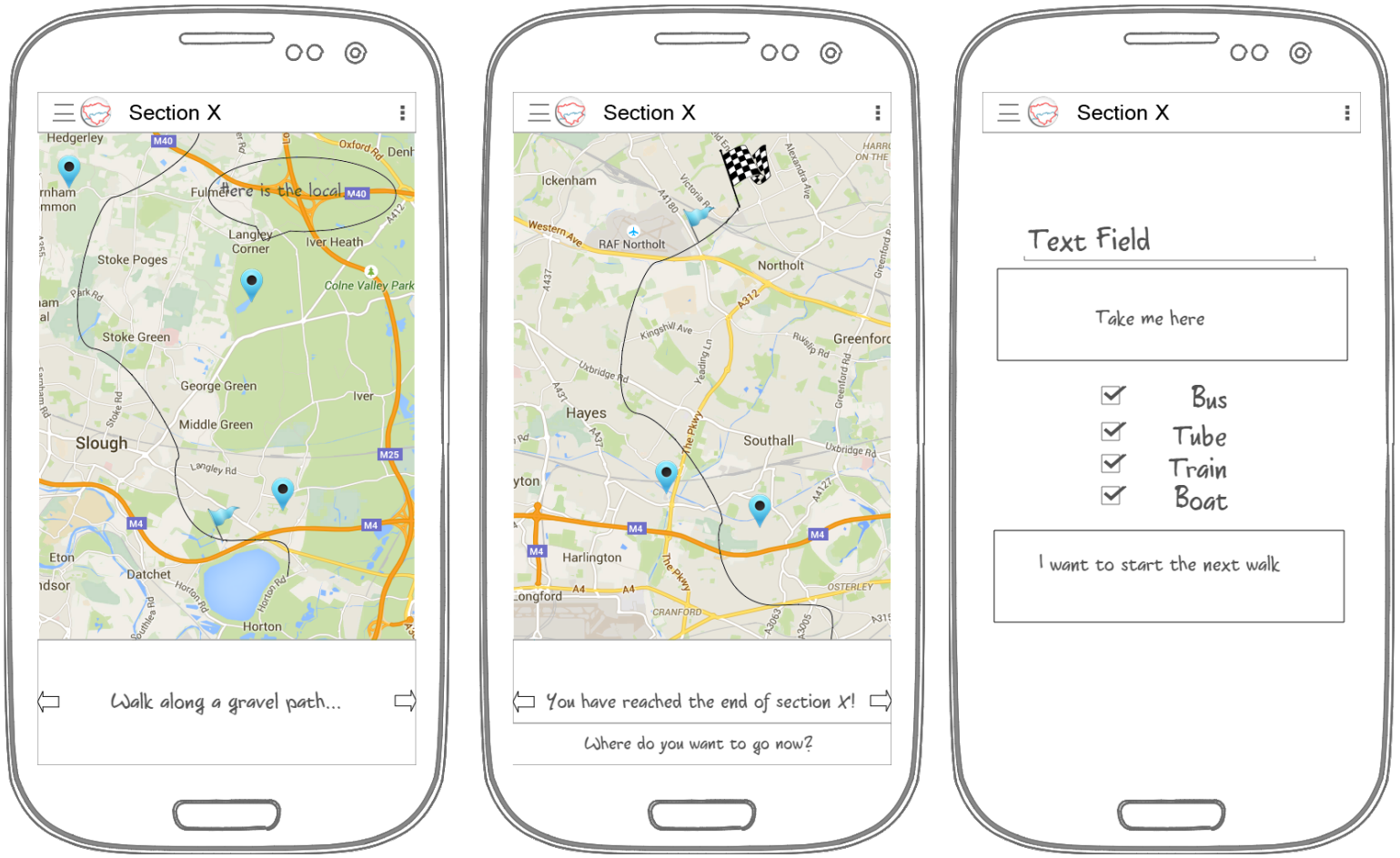


Figure 20: London Loop Mockups cont.

Figure 21: London Loop Mockups cont.

4.4 Testing and Feedback

An essential part to any development process is testing, and although it would be ideal to make the development of my project test-driven, I found that this was not as easy as it seemed. The main requirement for testing my app is the use of location services, and without physically taking the device to that location to test it, I could not know that the app would work. Therefore, with the help of friends and family, I was able to install my app onto their devices so that I was not the only person using my app, and so that they could provide feedback on any bugs they found, and the experience of using my app on site.

In order to quantify some of this feedback, I created a survey for them to complete so that I could see what aspects of my app I need to work on (figures 22 to 24). In particular, I asked users several questions relating to the design and functionality of the app.

Using feedback from users during the testing process, I was able to fix any bugs that I had not found myself in the app, and make the app easier to understand. Additionally, using the results produced by the surveys from not just users who tested the app, but users who had not tried the app before, I was able to achieve feedback regarding how useful users found each feature, as well as what they thought of the branding of the app.

1. Having used the app, how well do you feel you understand what the London Loop?

- Extremely well
- Very well
- Moderately well
- Not so well
- Not at all well

2. How easy was it to find what you were looking for on the app?

- Extremely easy
- Very easy
- Moderately easy
- Not so easy
- Not at all easy

3. How visually appealing is the app?

- Extremely appealing
- Very appealing
- Moderately appealing
- Not so appealing
- Not at all appealing

4. How well does the app..

	Extremely well	Very well	Moderately well	Not so well	Not well at all
navigate you to the London Loop?	○	○	○	○	○
navigate you during a London Loop walk?	○	○	○	○	○
show you places of interest during the walk?	○	○	○	○	○

Figure 22: My London Loop App Survey

5. How often does the app crash/freeze?

- ○ Extremely often
- ○ Very often
- ○ Moderately often
- ○ Not so often
- ○ Not at all

6. How much more useful is the app compared to the book to navigate the London Loop?

- ○ Extremely useful
- ○ Very much useful
- ○ Moderately useful
- ○ Not very useful
- ○ Not useful at all

7. How do you rate each of the following features in the app?

	Extremely useful	Very useful	Moderately useful	Not very useful	Not at all useful
Colour coded icons showing which walks are in South/NorthWest/NorthEast London	○	○	○	○	○
Colour coded map markers showing which walks iare n South/NorthWest/NorthEast London	○	○	○	○	○
The length of a walk	○	○	○	○	○
The estimated time it would take to complete the walk	○	○	○	○	○
Text describing the walk	○	○	○	○	○
Text stating the current day's weather forecast	○	○	○	○	○
Text suggesting attire due to the current day's weather forecast	○	○	○	○	○
Filters on modes of transport	○	○	○	○	○
Navigation notes that depend on your location on the walk	○	○	○	○	○
Statistics about the user's progress	○	○	○	○	○
Statistics about the global use of the app	○	○	○	○	○

Figure 23: My London Loop App Survey cont.

8. Above is an image of the logo I created for the app. How clearly does it describe the London Loop?

- Extremely clearly
- Very clearly
- Moderately clearly
- Not very clearly
- Not at all clearly

9. Please can you provide any additional feedback for the app?

Figure 24: My London Loop App Survey cont.

5 Implementation

As discussed in section 2, I use the Android Operating System (2.2) with the Android SDK (2.3), and implementing my app in the Android Studio IDE (2.7). I also use a virtual machine I created on Department of Computing's Apache Cloudstack for my web service (2.4) and database(2.5).

5.1 Creating the skeleton app

In order to begin my project, I began by creating a simple Navigation Drawer Activity from the window in figure 25.

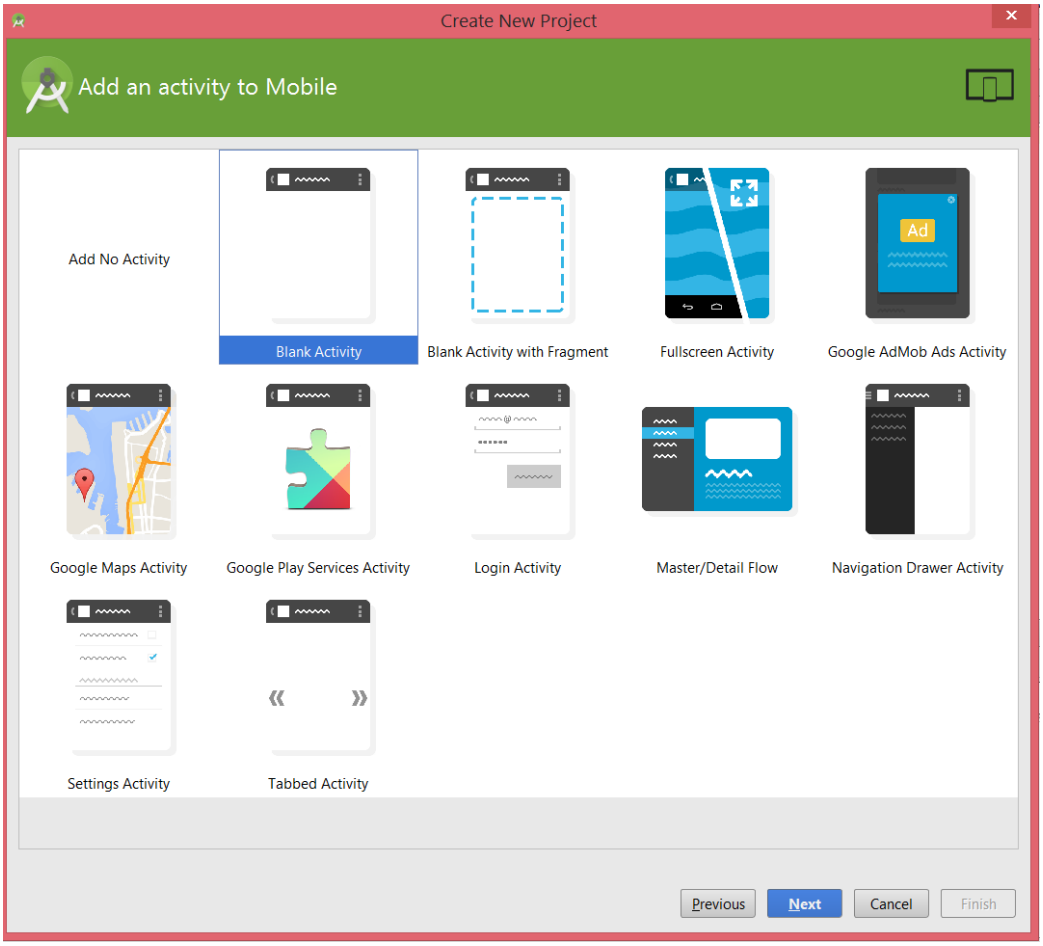


Figure 25: My London Loop App Survey cont.

Then, I added my 3 pages - 'Walks', 'Map', 'Statistics'. I wanted the user to see the walks in two different styles of lists - one in a conventional list, and one where the user clicks on a marker on a map to see what walk it is, whilst having a third page for the user's own statistics.

5.2 Creating the list of London Loop walks

Before creating my internal database, I set up my dummy data in an ArrayList holding a class WalkViewItem, whose constructor can be seen in the listing 1.

Listing 1: LocationListener

```
public WalkViewItem(Drawable icon, String title, String
    description) {
```

```
    this.icon = icon;
    this.title = title;
    this.description = description;
}
```

However, following redesigns, I removed the icon from the WalkViewItem, and have redevloped the class WalkViewItem into two separate classes, NodeItem and SectionItem, with their constructors seen in the listing 2.

Listing 2: NodeItem and SectionItem constructors

```
public NodeItem(long nodeId, String name, double latitude,
    double longitude) {
    this.nodeId = nodeId;
    this.name = name;
    this.latitude = latitude;
    this.longitude = longitude;
}

public SectionItem(long id, NodeItem start_node, NodeItem
    end_node, String description, double miles) {
    this.id = id;
    this.startNode = start_node;
    this.endNode = end_node;
    this.description = description;
    this.miles = miles;
}
```

I then created the new icons, as in my mockups (figure 17) which separate the walks into their sections of the map - North-West, North-East and South. This can be seen in figure 26.

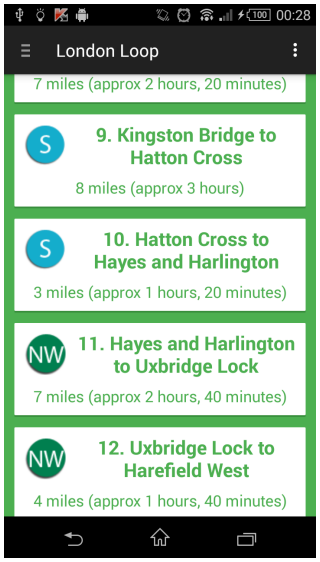


Figure 26: Walk List

5.3 Adding the map fragment

Following my mockups in section 4.3, I created my maps page, as seen in figure 27.

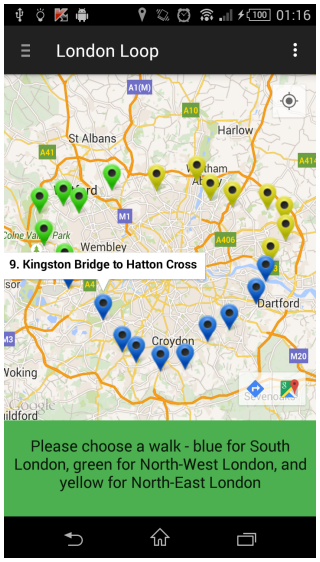


Figure 27: Map of Walks

Again, I have followed a similar pattern to that of the walks page, chang-

ing the colour of the marker to the colour of that section of the London Loop.

5.4 Linking the walks to its own page

Additionally, alongside both methods to visualise the walks, I also included the walk detail page. Both the marker's info window and the walk's card takes the user to the walk detail page. Here, I implemented a small piece of information about the walk, taken from Transport for London's London Loop page [8], as well as nearest train stations to the start and end point.

As can be seen in figure 28, I opted for the user to receive weather updates within the walk detail, as I was able to get weather information from my API (OpenWeatherMap as discussed in 2.6) to be specific to the location. Prior to this, I had considered making a separate page - but who would click on it?, as well as including it in the navigation bar - but what if the user never opens it? Therefore, it was clear to conclude to include weather information just above the "Start" button so that users see it before making a move.

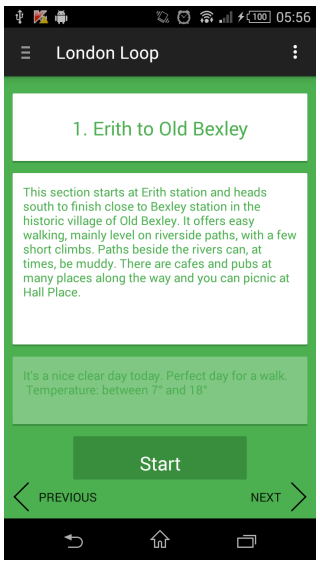


Figure 28: Walk Detail Screen

5.5 Switching to an internal database

As I discussed in section 5.2, I originally used dummy data for all the sections and nodes. However, in order to port this information into a database, I had to create a database helper class, MySQLiteHelper, extending SQLiteOpenHelper. There, I stated strings of the database, tables and columns, and

used them to help me form a “CREATE TABLE” statement for each of the tables, as well as methods to create, upgrade, drop and close the database. Additionally, each table has methods to get, get all and set elements in the database. Alongside these default methods, I set up new methods such as to select from a certain table where an element coincided with one from another table, and a method to check if a table existed so that the creation of the database occuring in MainActivity.java was not repeated.

The creation of this database occurs in MainActivity upon the first time the app is installed in a method known as ‘createWalks()’. This method takes text files held in the raw folder of the project’s resource directory, parses them, and inputs the data into the SQLite database. The implementation of this can be seen in more detail in Appendices A.

5.6 Navigating the user

Once the user clicks the ‘Start’ button, they are given two options - to be guided to the walk, or to start the walk immediately, as seen in figure 29.

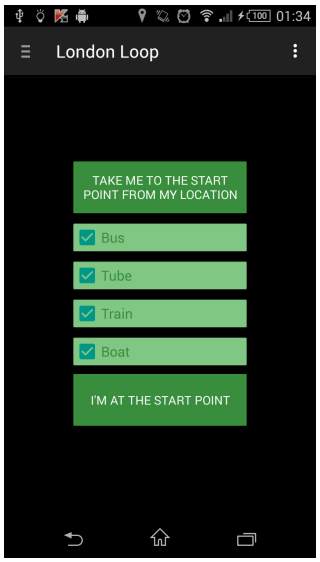


Figure 29: Start Walk Screen

If they want to be guided to the walk, I created 4 checkbox filters for the user to choose from which are defaulted to be checked. This means that the user can uncheck them if they decide that they do not want to use a particular mode of transport. I then create an API request, for example

http://transportapi.com/v3/uk/public/journey/from/lonlat:-0.1824585,51.515412/to/lonlat:-0.411109,51.470567.json?api_key=377843b343d1e052ac4d024fd9b7c93a&app_id=6109f899&modes=bus-tube-train-boat

which takes the 'from' location to be the user's current location's longitude and latitude, the longitude and latitude of the start point of the walk to be the 'to' location, the API key and id which is generated upon registering an account with transportapi.com, and a filter of modes (where each mode is removed upon unchecking a box).

Upon searching for a journey, the app first executes an AsyncTask which gets the user's current location, and then executes a second AsyncTask which sends the API request on a separate thread. Following a result from the API, the app either parses the returned JSON format into an ArrayList of RouteItems which are then displayed as shown in figure 30, or it produces an error message in a Toast which returns the user back to the search page. The implementation of this process is detailed in Appendices B.

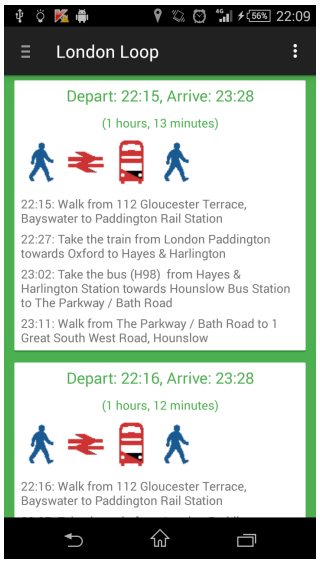


Figure 30: List of journey options

Then, once the user clicks on a route, the user is navigated as shown in figure 31. Here, I defined a location listener, as described in listing 3, to take a location changed event and change the text below as required.

Listing 3: LocationListener

```
// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
   public void onLocationChanged(Location location) {
      // Called when a new location is found by the network
          location provider.
      mLastLocation = location;
      hasLocation = true;
      setMapText();
   }

   public void onStatusChanged(String provider, int status,
       Bundle extras) {}

   public void onProviderEnabled(String provider) {}

   public void onProviderDisabled(String provider) {}
};
```

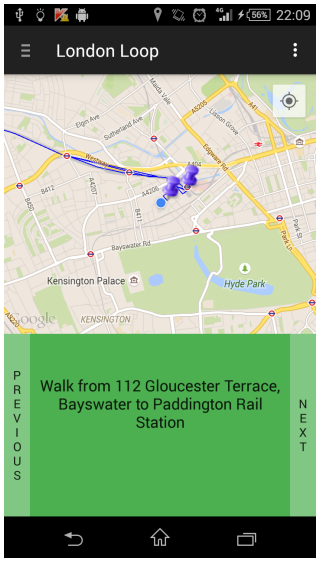


Figure 31: Screen showing guide from current location to London Loop

Additionally, if the user requires to see the text ahead themselves, then there are left and right arrow buttons to navigate through the journey text, as well as blue pins placed on the map to show milestones in the journey.

Once the user reaches the start of the walk, a button prompts the user to start the walk (figure 32), to which they respond with yes or no. If they

start the walk, then a new navigation map begins in much the same manner as with the navigation to the walk. However, the map pulls data from the Marker table in the database, therefore creating a new (pink) marker pin across the screen showing places of interest to the user (figure 33).

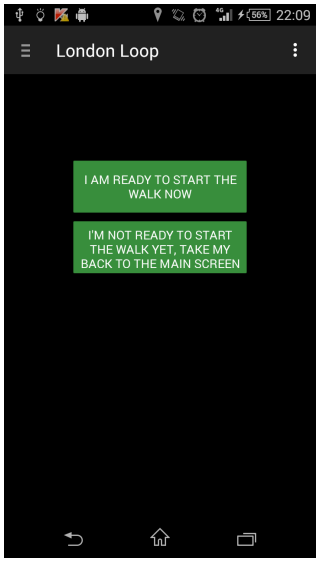


Figure 32: Prompt to start walk

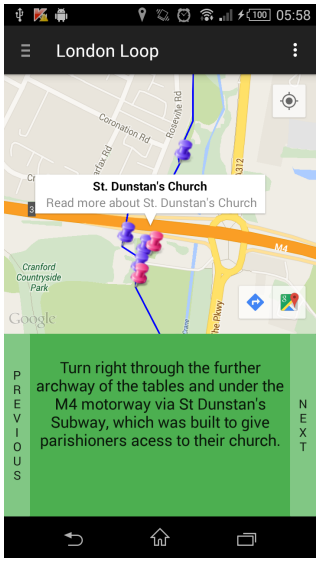


Figure 33: Screen showing route plan for London Loop walk

Once the user has completed the walk, they are given a similar option to when they began of what to do next. If they choose to start the next walk, the navigation continues to the next walk. If they choose to leave, then they can enter a location (postcode or station name) as a destination, and much in the same way as when planning the journey to the Loop, it plans a journey somewhere else.

5.7 Adding the weather service

In order to retrieve information regarding the weather, I followed the example as shown in figure 34. Therefore, taking the geographic coordinates of each walk, I send a JSON request to the OpenWeatherMap API, and then parse the returned JSON response to present the data required. In particular, I hold an if-else case on the response's main weather and display the weather note as seen in table 9.

Main Weather	Text
Clear Sky	It's a nice clear day today. Perfect day for a walk.
Few Clouds	There are only a few clouds in the sky today. Nothing to worry about.
Scattered Clouds	There's only a few clouds in the sky today. Nothing to worry about.
Broken Clouds	It's very cloudy today
Shower Rain	It's raining today, so don't forget your coat and umbrella.
Rain	It's raining today, so don't forget your coat and umbrella.
Thunderstorm	There's going to be thunderstorms today so take a coat and be careful
Snow	It's snowing today so don't forget some warm clothes and a carrot for your snowman.
Mist	It's misty today, so be careful when you're walking.

Table 9: Personalised weather notes

Additionally, I would also display the day's lowest and highest temperature in degrees Celcius so that the user also has some quantitative information about the day's weather.

- in JSON api.openweathermap.org/data/2.5/weather?lat=35&lon=139
- in HTML api.openweathermap.org/data/2.5/weather?lat=35&lon=139&mode=html


```
{"coord":{"lon":139,"lat":35},
"sys":{"country":"JP","sunrise":1369769524,"sunset":1369821049},
"weather":[{"id":804,"main":"clouds","description":"overcast clouds","icon":"04n"}],
"main":{"temp":289.5,"humidity":89,"pressure":1013,"temp_min":287.04,"temp_max":292.04},
"wind":{"speed":7.31,"deg":187.002},
"rain":{"3h":0},
"clouds":{"all":92},
"dt":1369824698,
"id":1851632,
"name":"Shuzenji",
"cod":200}
```


Figure 34: Example Weather Request and Response

5.8 Setting up the web service

In order to set up a web service, I first created an online virtual machine on the Department of Computing's Apache Cloudstack. This can be seen at the IP address 146.169.46.77, with port 55000. Initially, I was using port 80, however the Department of Computing locks the virtual machines so that their IP address is inaccessible from outside of the Department of Computing's network or VPN. Therefore, following a lengthy process requesting access outside the network, so that I could allow my friends and family to easily test my app, I was allowed access on May 30th 2015.

To create my web service, I installed PHP, MySql, as well as phpMyAdmin onto the virtual machine where I was able to create my server side database easily using the phpMyAdmin web interface. Following this, I proceeded to create CRUD(Create, Read, Update, Delete) functions in PHP, so that the app can perform these tasks on the database. Of course, I intended from the beginning not to require user authentication, so I do not have to worry about registration of user accounts and the security of the passwords. Instead, I use this to update information regarding the statistics about the users, and to hold a copy of all the information about the London Loop (e.g. a backup copy of the internal SQLite database) which can later be updated so that the app takes the information from the online server and stores it on its own database.

5.9 Creating the Statistics Screen

In my statistics screen, I wanted the app to show off what the user had accomplished as well as some other statistics they may find interesting. These include statistics about how many people are currently using the app to walk,

and how many walks have been completed by everyone collectively. I used the internal database to create a small Statistics table which I only use for the user of the app. When the user completes a walk, their statistics are changed to show that they have completed an extra walk, walked an extra so many miles, and spent so much time walking. Similarly, the online database will store a set of statistics for all the users - for example where any one user completing a certain walk would increment the WalksCompleted column.

5.10 Facebook API

Although the Facebook login no longer features in my app, I had tried to incorporate it into my app, creating a login button which was functional and returned information regarding the user when the login and password was input in the Facebook pop-up login screen. However, I soon ran into problems regarding what to do with the information, as I did not intend to have a table of user ids online, and therefore decided that it was more important to complete my core features than to move onto extensions too early in the project. I therefore left it in the additional specification for the possibility of the user being taken directly to the Facebook app from my London Loop app to post a photo onto their wall, for example.

5.11 Conclusion

In order to implement my app according to the specification (section 3.1) and objectives (section 1.2), I followed specification step-by-step, filling in its functionality as written in order. I also observed parts of the implementation which was unnecessary, such as the Facebook login, as it was extremely time consuming for the little functionality which it would add. However, I believe that I fulfilled my core requirement as necessary, leaving a small window to work on some of my extra specifications before the project hand in date.

6 Evaluation

Coming to the completion of my project, I reflect back on my methods to solve my problem, the issues faced, and the limitations of my solution. In particular, I begin by comparing my project plan (figure 4 from section 3.2) with the actual events that occurred throughout the course of the project. As seen in figure 35, the project plan is highlighted in blue and the actual events are highlighted in red below. In particular, I notice that I did not allow enough time in February to write my interim report, and so had less time to implement my app, as planned previously. However, I was aided by the time I had planned to leave for my March Computing exams and so slowly became back on track.

I also found during the process of implementation that some of the different features were easier to implement in a different order, for example implementing the walk detail before the map list as I only had to create a link from the map to the same walk detail from the map marker. Additionally, I found that the statistics page was not so easy to complete without the external database fully working, and so found it more difficult to implement until the 30th May 2015 when I was given access to my virtual machine without being connected to Department of Computing's network.

Therefore, I had become behind schedule by some weeks in terms of implementation, and began the core work in March. Fortunately, I had already allowed time during March for my exams at the end of that month. Therefore, in April, I was only behind in implementing the map list and internal database. I then began balancing my work with my own revision for the exams in May, but found that the majority of my work on this project was concentrated following my exams when I could fully focus on my app.

In retrospect, however, I would have preferred to have kept closer to my plan, and given more time to my project before January so that I had a head start before I had to focus on my revision.

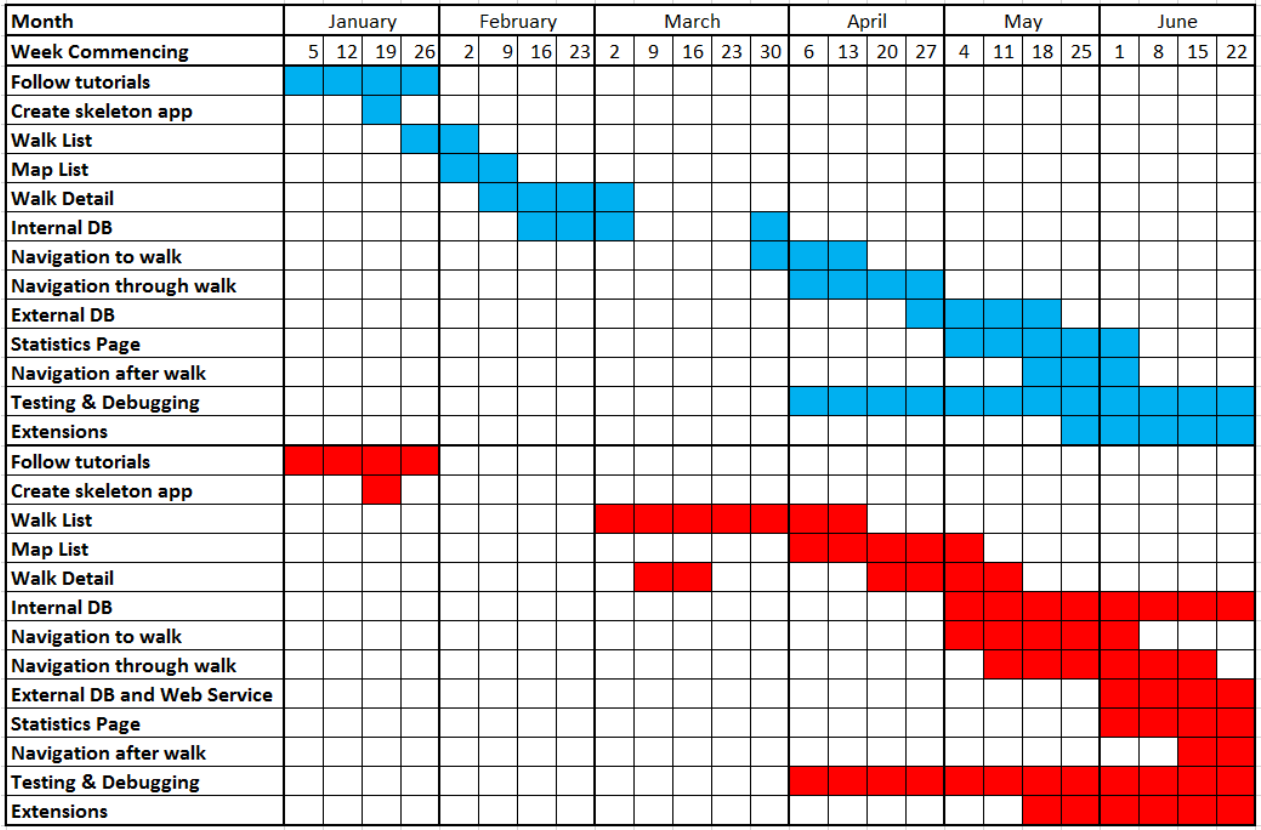


Figure 35: Comparison of Implementation plan (blue) with actual events (red)

6.1 Testing

During the course of the project, Android released a new version of their operating system - Android Lollipop. This brought in it new functionality which was not compatible with the previous versions. Therefore, in order to ensure that my app would work on both old devices and updated/new devices, I delayed upgrading my own smart phone to Android Lollipop and kept the original KitKat operating system installed. Particular new features, namely a new CardView and RecyclerView were included in the upgrade. This was perfectly designed to conform to Google's Material Design discussed in section 4.2 User Interface Design. However, I noticed that the use of this would push the minimum SDK version to be 21 (that of the new Lollipop operating system). Therefore, in my 'build.gradle' files, I set my target SDK to be 21, but kept my minimum SDK to be version 16, so that any Lollipop-only code would not be allowed to build. On the other hand, I would have also had to ensure that any code would not only work on my un-upgraded phone, but also on a phone which uses Lollipop. Therefore, I tested the London Loop app on both Lollipop and KitKat phones to ensure maximum compatibility.

In terms of testing out in the Loop itself, I was also able to discover new bugs in my code which I had not found earlier such as the following.

- When two checkpoints are too close to each other, the app updates the notes between them and so the user may not be getting the most accurate data regarding the walk.

  Solution: To decrease the distance the user has to be to a checkpoint for the correct note to be displayed. This requires heavy in field testing to ensure that all cases are accounted for.

- The app crashing when its orientation is landscape and the view is a map view.

  Short Term Solution: To fix orientation to portrait.

  Long Term Solution: To find out why the map causes a crash on landscape and solve the problem.

- Inaccuracy of the start point led users to cross a road which was unnecessary.

  Solution: To find out exactly where the walk starts so that the geographic coordinates are correct and would not lead walkers astray.

This allowed me to find solutions to problems which I may not have realised on my own or with my own device.

6.2 Human Feedback

In order to establish what other users thought of my app, I created a survey asking users of the app to provide feedback on their experience using the London Loop app, wherein the full results are visible in Appendices D. To summarize my results, I found that users have overall found the app easy to use, navigate, and visually appealing. This enforces the idea that design is a vital part in creating any product, and having completed a strong selection of mockups, I have been able to produce an app which is both aesthetically pleasing and clear to use.

The majority of users have also found the app easy to use both when navigating to the walk and when navigating during the walk, showing me that I have implemented the navigation in a way which is simple to use, and behaves correctly - particularly when the user's location changes. However, many of the users chose the option "Not so often" when describing how often the app crashed or froze, suggesting that there are still problems within the app that I have not yet realised every edge case when coding the app.

In asking users what they thought of the different features, the colour coding and global statistics seemed the least important feature of the app, whilst noting the use of filters on modes of transport, the navigation notes updating upon location changing, and information regarding the walk itself such as estimated time and length to be the most useful features in the app. Since these are the core features of the app, with the colour coding a touch on the appearance and the global statistics something that may not be useful, but may be interesting to the user, I found these statistics to conform with how I hoped the app would appear to users.

In terms of branding, users found a clear connection between my logo (figure 13) and the London Loop. This was very good news as it means that users will be able to remember the logo, and associate it with the app and walks.

From the written feedback from users, I learnt that the app had encouraged users to go out and complete the walks and thus bringing me back to my motivation in section 1.1 - "To encourage people to become more active." I am now confident that the app can be used as a tool to allow users to be more active and explore more of the city around them.

6.3 Choice of API

Having chosen my APIs since section 2.6, I created a survey to ask potential users what sorts of websites and apps they use for similar features. The questions are shown below in figure 36, and the results are visible in Appendix E.

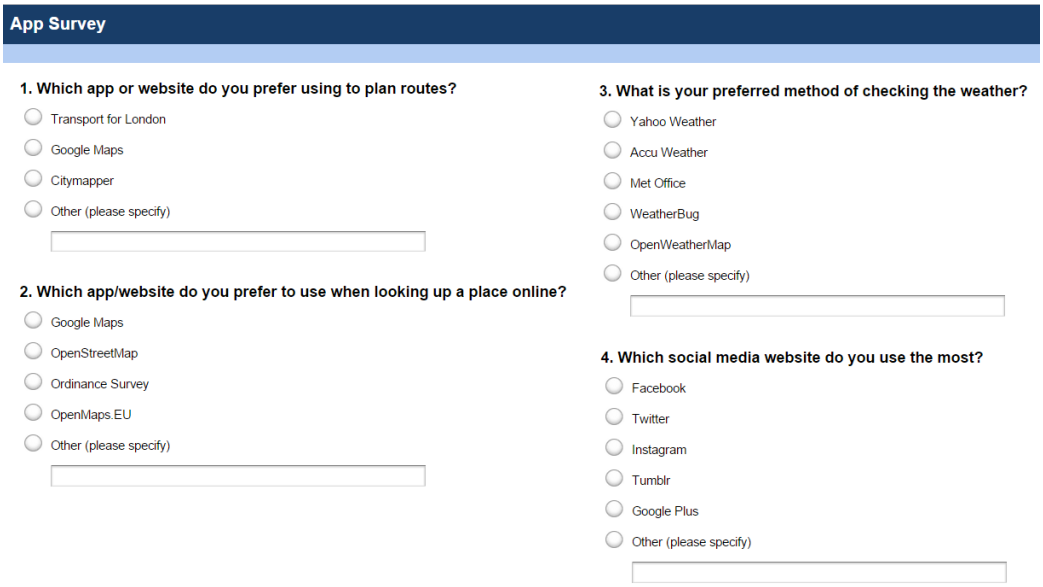


Figure 36: App Survey

To summarise these results, I found that contrary to what I had believed, more people used Google Maps than any other method to plan their journeys. I also found that the majority of people preferred Citymapper to Transport for London. However, due to Google Map's lack of filtering, I was swayed to choose a different API, as previously discussed in 2.6. This choice was the transportAPI, a resource which pulls data from both Transport for London and Citymapper. Therefore, with few results being from any other category other the three already mentioned, I am confident that my choice was still the correct one.

Google Maps also surpassed all other results in terms of looking up places, suggesting that people were accustomed to the Google Map interface and by using it in my app, brought something familiar and more appealing to users. However, for mapping out routes, Google Maps would only allow 8 waypoints per path, and so I instead had to consider an alternative. This was the use

of drawing a path or ellipse across the map interface, and updating the text guiding the user on location changed events. This meant that the app was guiding the user itself and not requiring a Google api to do so.

The majority of answers regarding the weather were in the Other section, with the majority using the BBC website or their inbuilt phone app. However, as BBC does not provide an API, I am satisfied that my decision was not wrong. The OpenWeatherMap allows me to pinpoint exactly where I want weather information about, not only just in a city but in a particular location. Additioanlly, the inbuilt app for Android, Accu Weather, is a premium API service and so out of my reach for this project. Therefore, I am left between Met Office, Yahoo Weather and OpenWeatherMap. MetOffice does not allow the user to choose a specific location but instead it must include a code for the location, making the request more difficult to form. Similarly, Yahoo Weather uses an id called 'woeid' to define locations. This makes the OpenWeatherMap the easiest to use and thus justifying my choice from earlier.
It is clear from the results that Facebook is the most popular Social Media interface that people use and so if I were to use any social media interaction in my app, I would use Facebook. However, as discussed earlier, I chose against including any interaction in my app unless time permits.

6.4 What I learnt

As this was the first time I had created a smart phone app, I expected there to be many learning curves during the implementation phase. In particular, what I found most difficult about how to start was where to start. I had all the tools required to make an app, but I did not know how to utilize them to something that would appear on a smart phone. Therefore, I completed several online tutorials, video tutorials, read books and viewed sample apps from Google's Developers website. In this way, I created and recreated several basic apps, allowing me to build a solid understanding of how Android apps work.
In particular, I found that Android has two UI components - fragments and activities. A Fragment represents a behaviour or a portion of user interface in an Activity, which means that a single activity can contain multiple fragments. This allows users with larger screens to be able to see two fragments at the same time, as well as for the app to retain the state of each activity so that the user can click the back button without losing the state of the previous fragment.
Additionally, the use of XML files for the visual aspect of implementation was

new to me, and I quickly learnt how to create layouts for different fragments, and create 'drawable' files for the layout files to contain. This allowed me to reuse the same layout or drawable multiple times in different situations - for example when using the same button style in different areas of the app.

The biggest learning curve I found was the use of AsyncTask to request data from an API, and also the creation of my own API. In order to send a requestion to an API, I must first create a class within the class where I want to use the data which extends AsyncTask. I can then add a ProgressDialog in the method 'onPreExecute' if required so that users know that the app is working on something. Then, overriding the method 'doInBackground', I perform the task of sending the HTTP request, retrieving the results, then do something with it in the method 'onPostExecute'. This became even more difficult when it came to my web service. I had successfully created my database online however when it came to reading and updating the database data from the app, I struggled. I was able to achieve the GET request from my database; however I was unable to send a POST request, but after a long process of following different tutorials online and reading up on the process of sending a request to a RESTful API, I was successfully able to update the database.

7 Conclusions and Future Work

Having now created a working smart phone app which, as my Project Proposal discusses, can be used by walkers of the London Loop to plan and navigate walks, as well as to find out more information about points of interest, I can see that my app fulfils all of its core objectives, and also fulfils the extended objective "The app should keep a count of how many users are currently walking on a particular walk" but also extends that to an abundance of statistics which the user can keep track of, as seen in the specification.

Through the project itself, I have learnt a great deal about London, and discovered places I had not seen before. From trying to walk the London Loop with only a book to hand, to testing out my app, I have been inspired not only to complete my app fully, but also to complete every walk on the London Loop and discover a London far different to that of the busy urban streets of Central London.

7.1 Future Work

As this is not a fully complete app, only containing data for one of the walks (Walk 10: Hatton Cross to Hayes and Harlington), the first piece of future work to be completed would of course be to flesh out the entire app, filling in all the details of each individual walk so that the app can be used through all of the London Loop.

I would then begin on the extended objectives I could not complete, such as holding user information online so that the user could retain their statistics if they were to switch to another device, and the ability for users to upload their own points of interests, restaurants, shops, etc. This would be incredibly useful as shops close down and new shops open. It would be impossible to keep track of all these occurrences without user input.

Although I had begun the process of including social media, I never completed it, and thus would like to finish this task, and most likely incorporate it into the user information stored online, so that users do not have to register with an email address but can instead use their Facebook account.

More options would be to allow users to upload images of their walk and share it with other users, giving new walkers the ability to see what others have seen on that walk. However, this would require a moderation system online where images posted would be reviewed and stored in a secure place

which could not be altered.

All in all, there are many ways I could take this app further, and with more time, I would like to be able to fulfil these ideas to build a fully fledged London Loop smart phone app.

8 References

[1] Kurt Wood (19 May 2015)
British children facing a rising tide of type 2 diabetes, says NHS chief
Available from:
http://www.diabetes.co.uk/news/2015/may/british-children-facing-a-rising-tide-of-type-2-diabetes,-says-nhs-chief-97493834.html [Accessed on 20th May 2015]

[2] Kurt Wood (13 May 2015)
British people unaware of their obesity, survey suggests
Available from: http://www.diabetes.co.uk/news/2015/may/British-people-unaware-of-their-obesity,-survey-suggests-96976829.html [Accessed on 20th May 2015]

[3] BBC News London (2 February 2015)
London's population hits 8.6m record high
Available from:
http://www.bbc.co.uk/news/uk-england-london-31082941 [Accessed on 5th February 2015]

[4] Simon Usborne (25 September 2014)
47 per cent of London is green space: Is it time for our capital to become a national park?
Available from: http://www.independent.co.uk/environment/47-per-cent-of-london-is-green-space-is-it-time-for-our-capital-to-become-a-national-park-9756470.html [Accessed on 5th February 2015]

[5] Robin Yapp (17 September 2014)
Londoners walk less and can't fit in time for exercise, research shows
Available from: http://www.standard.co.uk/news/london/londoners-walk-less-and-cant-fit-in-time-for-exercise-research-shows-9737043.html?origin=internalSearch [Accessed on 5th February 2015]

[6] Sharp, David, and Colin Saunders (2012)
The London Loop (Recreational Path Guides)
Published by: London: Aurum [2012, Print Edition]

[7] Wikipedia (9 May 2015)
London Outer Orbital Path

Available from:
http://en.wikipedia.org/wiki/London_Outer_Orbital_Path [Accessed on 21st May 2015]

[8] Transport for London (2015)
London LOOP
Available from: http://www.tfl.gov.uk/modes/walking/loop-walk [Accessed on 21th May 2015]

[9] Saturday Walkers Club (March 2014)
London Loop - Download GPS Route
Available from: http://www.walkingclub.org.uk/long-distance-path/london-loop/download-GPX-KML.shtml [Accessed on 10th February 2015]

[10] Wikipedia (15 June 2015)
Comparison of mobile operating systems
Available from:
http://en.wikipedia.org/wiki/Comparison_of_mobile_operating_systems [Accessed on 17th June 2015]

[11] Jude Aakjaer (1 February 2012)
Android Development: Do You Know Your Options?
Available from: http://www.sitepoint.com/android-development-do-you-know-your-options/ [Accessed on 10th June 2015]

[12] Google Inc. (15 Jube 2015)
Maps
Available from:
https://play.google.com/store/apps/details?id=com.google.android.apps.maps&hl=en [Accessed on 17th June 2015]

[13] LondonNut.com (4 November 2015)
London Transport Planner
Available from:
https://play.google.com/store/apps/details?id=com.london.londontransport&hl=en [Accessed on 4th June 2015]

[14] Citymapper Limited (16 June 2015)
Citymapper - Bus, Tube, Rail
Available from: https://citymapper.com/london/apps [Accessed on 17th June 2015]

[15] TripAdvisor LLC (2 June 2015)
London City Guide
Available from:
https://play.google.com/store/apps/details?id=com.tripadvisor.android.apps.cityguide.londo
[Accessed on 17th June 2015]

[16] GPSmyCity.com, Inc. (27 March 2015)
London Map and Walks
Available from:
https://play.google.com/store/apps/details?id=com.gpsmycity.android.u4
[Accessed on 17th June 2015]

[17] MapMyFitness, Inc. (15 June 2015)
Walk with Map My Walk
Available from:
https://play.google.com/store/apps/details?id=com.mapmywalk.android2
[Accessed on 17th June 2015]

[18] Mobitrips (7 November 2013)
Split City Walks Guided Tour
Available from:
https://play.google.com/store/apps/details?id=com.mytoursapp.android.app315
[Accessed on 11th February 2015]

[19] Mission Communications Ltd (26 March 2015)
New York Pass - Travel Guide
Available from: https://itunes.apple.com/gb/app/new-york-pass-travel-guide/id429167326?mt=8 [Accessed on 17th June 2015]

[20] Aardman Animations Ltd. (11 June 2015)
Sheep Spotter
Available from:
http://shaunthesheep.com/games/shaun-city-sheep-spotter [Accessed on 17th June 2015]

[21] Renee (16 November 2012)
Code Wars: Ruby vs Python vs PHP [Infographic]
Available from: https://blog.udemy.com/modern-language-wars/
[Accessed on 12th June 2015]

[22] Transport for London (2015)
OPEN DATA USERS: Our Feeds

Available from: https://www.tfl.gov.uk/info-for/open-data-users/our-feeds?intcmp=3671 [Accessed on 12th June 2015]

[23] Transport API (2015)
System Properties Comparison Microsoft SQL Server vs. MySQL vs. SQLite
Available from: http://www.transportapi.com [Accessed on 12th June 2015]

[24] Emma Hulme (15 June 2015)
My London Loop App Survey
Available from: https://www.surveymonkey.com/s/D2J8R8J [Accessed on 17th June 2015]

[25] GitLab (2015)
GitLab Version Control
Available from: https://about.gitlab.com/ [Accessed on 13th February 2015]

[26] Emma Hulme (17 June 2015)
London LOOP / LondonLoopApp
Available from:
https://gitlab.doc.ic.ac.uk/london-loop/londonloopapp.git [Accessed on 17th June 2015]

[27] Wikipedia (9 June 2015)
Iterative and incremental development
Available from:
http://en.wikipedia.org/wiki/Iterative_and_incremental_development [Accessed on 17th June 2015]

[28] Emma Hulme (17 June 2015)
London Loop Project Board
Available from: https://trello.com/b/kfXkxCS5/london-loop [Accessed on 17th June 2015]

[29] Wikipedia (17 June 2015)
Trello
Available from: http://en.wikipedia.org/wiki/Trello [Accessed on 17th June 2015]

[30] Google (2015)

Material Design
Available from:
http://www.google.com/design/spec/material-design/introduction.html
[Accessed on 12th June 2015]

[31] Mobile Themes (12 June 2013)
android-mobile-phone-system-smartphones-640x480-wallpaper-40_0
Available from: http://www.mobiwalls.net/android-wallpapers/android-640x480-wallpapers/12/android-mobile-phone-system-smartphones-640x480-wallpaper-40_0.html [Accessed on 13th June 2015]

[32] Tudor Prisacariu (2011)
LONDON LOOP
Available from: http://verde.io/work/london-loop/ [Accessed on 12th June 2015]

[33] Survey Monkey (12 June 2015)
App Survey
Available from: https://www.surveymonkey.com/s/XCH32NX
[Accessed on 17th June 2015]

Appendices

A. Method showing the creation of the SQLite Database

```
private void createWalks(){
   Resources resources = getResources();
   NodeItem[] nodeItems = setUpNodes();
   SectionItem[] sectionItems = setUpSections(nodeItems);
   setUpMarkers(sectionItems);
   if(db.hasTableCount(db.getReadableDatabase(), "Statistics")
       == false){
      long i = 0;
      while (i < 25){
         //index i-0 for all walks collectively, i=1-24 for
             individual walks
         db.createStatItem(new StatItem(i, 0, 0, 0));
         i++;
      }
   }

   if(db.hasTableCount(db.getReadableDatabase(), "Gps") ==
       false){
      InputStream inputStream =
          resources.openRawResource(R.raw.gps);
      BufferedReader br = null;
      String line = "";
      String cvsSplitBy = "\\+";
      try {
         br = new BufferedReader(new
             InputStreamReader(inputStream));
         while ((line = br.readLine()) != null) {
            // use comma as separator
            String[] gpsString = line.split(cvsSplitBy);
            if (gpsString.length == 5){
               GPSItem gpsItem = new
                   GPSItem(Long.parseLong(gpsString[0]),
               Integer.parseInt(gpsString[1]),
               new LatLng(Double.parseDouble(gpsString[2]),
               Double.parseDouble(gpsString[3])),
               sectionItems[Integer.parseInt(gpsString[4])-1],
                   "");
               db.createGPSItem(gpsItem);
            } else {
```

```
                GPSItem gpsItem = new
                    GPSItem(Long.parseLong(gpsString[0]),
                Integer.parseInt(gpsString[1]),
                new LatLng(Double.parseDouble(gpsString[2]),
                Double.parseDouble(gpsString[3])),
                sectionItems[Integer.parseInt(gpsString[4])-1],
                    gpsString[5]);
                db.createGPSItem(gpsItem);
             }
          }
       } catch (FileNotFoundException e) {
          e.printStackTrace();
       } catch (IOException e) {
          e.printStackTrace();
       } finally {
          if (br != null) {
             try {
                br.close();
             } catch (IOException e) {
                e.printStackTrace();
             }
          }
       }
    }
}

private NodeItem[] setUpNodes(){
    Resources resources = getResources();
    NodeItem nodeItems[] = new NodeItem[25];

    InputStream inputNodeStream =
        resources.openRawResource(R.raw.nodes);

    BufferedReader br = null;
    String line = "";
    String cvsSplitBy = ",";

    try {

       br = new BufferedReader(new
           InputStreamReader(inputNodeStream));
       int j = 0;
       long k = 1;
```

```
            while ((line = br.readLine()) != null) {

                // use comma as separator
                String[] nodeString = line.split(cvsSplitBy);

                nodeItems[j] = new NodeItem(k, nodeString[0],
                    Double.parseDouble(nodeString[1]),
                    Double.parseDouble(nodeString[2]));
                j++;
                k++;
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (br != null) {
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        if(db.hasTableCount(db.getReadableDatabase(), "Node") ==
            false){
            for (int i = 0; i < 25; i++){
                db.createNode(nodeItems[i]);
            }
        }

        return nodeItems;
    }

    private SectionItem[] setUpSections(NodeItem[] nodeItems){

        Resources resources = getResources();

        SectionItem[] sectionItems = new SectionItem[24];
```

```
InputStream inputSectionStream =
    resources.openRawResource(R.raw.sections);

BufferedReader br = null;
String line = "";
String cvsSplitBy = "\\+";

try {

   br = new BufferedReader(new
       InputStreamReader(inputSectionStream));

   int j = 0;
   long k = 1;

while ((line = br.readLine()) != null) {

   // use comma as separator
   String[] sectionString = line.split(cvsSplitBy);

   sectionItems[j] = new SectionItem(k, nodeItems[j],
       nodeItems[j+1], sectionString[1],
       Double.parseDouble(sectionString[0]));
   j++;
   k++;
}

} catch (FileNotFoundException e) {
   e.printStackTrace();
} catch (IOException e) {
   e.printStackTrace();
} finally {
   if (br != null) {
      try {
         br.close();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}

if (db.hasTableCount(db.getReadableDatabase(), "Section") ==
    false){
```

```
      for (int i = 0; i < 24; i++){
         db.createSection(sectionItems[i]);
      }
   }

   return sectionItems;
}

private void setUpMarkers(SectionItem[] sectionItems){

   Resources resources = getResources();
   InputStream inputSectionStream =
       resources.openRawResource(R.raw.markers);

   BufferedReader br = null;
   String line = "";
   String cvsSplitBy = ",";

   ArrayList<MarkerItem> markerItems = new
       ArrayList<MarkerItem>();

   try {

      br = new BufferedReader(new
          InputStreamReader(inputSectionStream));

      while ((line = br.readLine()) != null) {

         // use comma as separator
         String[] markerString = line.split(cvsSplitBy);

         int sectionNo = Integer.parseInt(markerString[1])-1;

         SectionItem s = sectionItems[sectionNo];
         LatLng l = new
             LatLng(Double.parseDouble(markerString[2]),Double.parseDouble(markerStri
         String name = markerString[4];
         Long id = Long.parseLong(markerString[0]);

         MarkerItem markerItem;

         if (markerString.length == 5){
```

```
            markerItem = new MarkerItem(id, s, l, name);
         } else {
            String text = markerString[5];
            String url = markerString[6];
            markerItem = new MarkerItem(id, s, l, name, text,
                url);
         }

         markerItems.add(markerItem);

      }

   } catch (FileNotFoundException e) {
      e.printStackTrace();
   } catch (IOException e) {
      e.printStackTrace();
   } finally {
      if (br != null) {
         try {
            br.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
   }
   if (db.hasTableCount(db.getReadableDatabase(), "Marker") ==
       false){

      for (MarkerItem m : markerItems){
         db.createMarkerItem(m);
      }
   }

}
```

B. Code showing the use of AsyncTask to retrieve the user's current location and a list of possible journeys for the user to take to the walk

```
public static String GET(String url){
   InputStream inputStream = null;
   String result = "";
   try {
      // create HttpClient
      HttpClient httpclient = new DefaultHttpClient();

      // make GET request to the given URL
      HttpResponse httpResponse = httpclient.execute(new
          HttpGet(url));

      // receive response as inputStream
      inputStream = httpResponse.getEntity().getContent();

      // convert inputstream to string
      if(inputStream != null)
      result = convertInputStreamToString(inputStream);
      else
      result = "Failed!";

   } catch (Exception e) {
      Log.d("InputStream", e.getLocalizedMessage());
   }
   return result;
}

private static String convertInputStreamToString(InputStream
    inputStream) throws IOException {
   BufferedReader bufferedReader = new BufferedReader( new
       InputStreamReader(inputStream));
   String line = "";
   String result = "";
   while((line = bufferedReader.readLine()) != null)
      result += line;

   inputStream.close();
   return result;
}
```

```
private ArrayList<Location> convertStringToCoord(String
    coordinates){
String delims = "[ \\[\\],]+";
String[] tokens = coordinates.split(delims);
ArrayList<Location> locations = new ArrayList<Location>();
for (int i = 1; i < tokens.length; i++){
if (i % 2 != 0){
Location l = new Location("TfL journey planning API");
l.setLongitude(Double.parseDouble(tokens[i]));
l.setLatitude(Double.parseDouble(tokens[i+1]));
locations.add(l);
}
}
return locations;
}

private ArrayList<RouteItem> getRouteItems(JSONObject
    jsonObject) throws JSONException {

   JSONArray routeArray = jsonObject.getJSONArray("routes");
   routeItems = new ArrayList<RouteItem>();

   for (int i = 0; i < routeArray.length(); i++){
      String duration =
          routeArray.getJSONObject(i).getString("duration");
      JSONArray partArray =
          routeArray.getJSONObject(i).getJSONArray("route_parts");

      RoutePart[] routeParts = new RoutePart[partArray.length()];

      for (int j = 0; j < partArray.length(); j++){
         String mode =
             partArray.getJSONObject(j).getString("mode");
         String from_point_name =
             partArray.getJSONObject(j).getString("from_point_name");
         String to_point_name =
             partArray.getJSONObject(j).getString("to_point_name");
         String destination =
             partArray.getJSONObject(j).getString("destination");
         String line_name =
             partArray.getJSONObject(j).getString("line_name");
         String part_duration =
             partArray.getJSONObject(j).getString("duration");
```

```
            String departure_time =
                partArray.getJSONObject(j).getString("departure_time");
            String arrival_time =
                partArray.getJSONObject(j).getString("arrival_time");
            String coordinates =
                partArray.getJSONObject(j).getString("coordinates");
            ArrayList<Location> locations =
                convertStringToCoord(coordinates);

            RoutePart rp = new RoutePart(mode, from_point_name,
                to_point_name, destination, line_name,
                part_duration, departure_time, arrival_time,
                locations);
            routeParts[j] = rp;
         }
         routeItems.add(new RouteItem(duration, routeParts));
      }
      return routeItems;
   }

   private class LocationControl extends AsyncTask<Context, Void,
       Void>
   {
      private final ProgressDialog dialog = new
          ProgressDialog(getActivity());

      protected void onPreExecute()
      {
         this.dialog.setMessage("Determining your location...");
         this.dialog.show();
      }

      protected Void doInBackground(Context... params)
      {
         Long t = Calendar.getInstance().getTimeInMillis();
         while (!hasLocation &&
             Calendar.getInstance().getTimeInMillis() - t < 30000) {
            try {
               Thread.sleep(Long.valueOf(1000));
            } catch (InterruptedException e) {
               e.printStackTrace();
            }
         };
```

```
        return null;
    }

    protected void onPostExecute(final Void unused)
    {
        if(this.dialog.isShowing())
        {
            this.dialog.dismiss();
        }

        //does the stuff that requires current location
        planJourney(mLastLocation, sectionItem);
    }

}

private class HttpAsyncTask extends AsyncTask<String, Void,
    String> {

    private final ProgressDialog dialog = new
        ProgressDialog(getActivity());

    protected void onPreExecute()
    {
    this.dialog.setMessage("Planning routes...");
    this.dialog.show();
    }

    @Override
    protected String doInBackground(String... urls) {
    return GET(urls[0]);
    }
    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
        JSONObject json = null;
        try {
            json = new JSONObject(result);
            routeItems = getRouteItems(json);
            setListAdapter(new RouteAdapterItem(getActivity(),
                routeItems, getActivity(), sectionItem.getId()));
            if(this.dialog.isShowing())
            {
```

```
            this.dialog.dismiss();
          }
        } catch (JSONException e) {
          Toast.makeText(getActivity(), "Unable to retrieve
              routes, please check internet connection and try
              again", Toast.LENGTH_SHORT).show();
          if(this.dialog.isShowing())
          {
            this.dialog.dismiss();
          }
          getActivity().getSupportFragmentManager().popBackStack();
        }
      }
    }
```

```
public View onCreateView(LayoutInflater inflater, ViewGroup
    container,
Bundle savedInstanceState) {

   View rootView = inflater.inflate(R.layout.fragment_gps_maps,
       container, false);
   //initialize layout
   walkNumber = getArguments().getLong("walkNumber", 0);

   db = new MySQLiteHelper(getActivity());

   dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
   startDate = new Date();

   statItem = db.getStatItem(walkNumber);
   globalStat = db.getStatItem(0);

   sectionItem = db.getSection(walkNumber + 1);

   gpsItemList = db.getGPSItemFromSection(sectionItem);
   markerItemList = db.getMarkerItemFromSection(sectionItem);

   //set current item to first gpsItem
   currentItem = gpsItemList.get(1);
   currentNo = currentItem.getIncr();

   //get buttons and textview from xml
   mapNavText = (TextView)
       rootView.findViewById(R.id.gpsMapText);

   gpsButton = (Button) rootView.findViewById(R.id.gpsButton);
   pButton = (Button) rootView.findViewById(R.id.preGPSButton);
   nButton = (Button) rootView.findViewById(R.id.nextGPSButton);

   //initialise text
   mapNavText.setText(currentItem.getNote());

   //set on click listeners for pre/next buttons
   pButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
```

```
      if (currentItem.getIncr()==1){
         //do nothing
      } else{
         setText(prevItemWithNote(gpsItemList.get(currentNo)));
      }

   }
});

nButton.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      if (currentItem.getIncr()==gpsItemList.size()){
         //do nothing
      } else{
         setText(nextItemWithNote(gpsItemList.get(currentNo)));
      }
   }
});


// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager)
    getActivity().getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
   public void onLocationChanged(Location location) {
      // Called when a new location is found by the network
          location provider.
      mLastLocation = location;
      hasLocation = true;
      setMapText();
   }
   public void onStatusChanged(String provider, int status,
       Bundle extras) {}
   public void onProviderEnabled(String provider) {}
   public void onProviderDisabled(String provider) {}
};

// Register the listener with the Location Manager to receive
    location updates
```

```
        locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
            0, 0, locationListener);
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
            0, 0, locationListener);

        setUpMapIfNeeded();

        drawPaths();

        drawMarkers();

        params = new ArrayList<NameValuePair>();

        return rootView;
    }

    private void setMapText(){

        mapNavText.setText(currentItem.getNote());

        for (int i = 2; i <= gpsItemList.size(); i++){
            Location l = new Location("gpsCoord");
            l.setLatitude(gpsItemList.get(i).getLatLng().latitude);
            l.setLongitude(gpsItemList.get(i).getLatLng().longitude);
            if (gpsItemList.get(i) != currentItem &&
                mLastLocation.distanceTo(l) < 50){
                setText(gpsItemList.get(i));
            }
        }
    }
    private void setText(GPSItem gpsItem){
        if (gpsItem.getNote().equals("")){
            //do nothing
        } else {
            mapNavText.setText(gpsItem.getNote());
            currentItem = gpsItem;
            currentNo = gpsItem.getIncr();
            //if final gpsItem
            if (gpsItem.getIncr() == gpsItemList.size()){

                gpsButton.setText("What do you want to do now?");
                gpsButton.setVisibility(View.VISIBLE);
                gpsButton.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {

            Date date = new Date();
            long diff = date.getTime() - startDate.getTime();
            long minutes = diff / (60 * 1000) % 60;

            //save stats internally
            statItem.setMiles(statItem.getMiles() +
                sectionItem.getMiles());
            statItem.setTime(statItem.getTime() + minutes);
            statItem.setCompleted(statItem.getCompleted() + 1);
            globalStat.setMiles(globalStat.getMiles() +
                sectionItem.getMiles());
            globalStat.setTime(globalStat.getTime() + minutes);
            globalStat.setCompleted(globalStat.getCompleted() +
                1);
            db.updateStatItem(statItem);
            db.updateStatItem(globalStat);

            //save stats online
            new putDataTask().execute();

            FragmentManager fragmentManager =
                getActivity().getSupportFragmentManager();

            EndWalkFragment wdf =
                EndWalkFragment.newInstance(walkNumber);

            fragmentManager.beginTransaction()
               .add(R.id.container, wdf)
               // Add this transaction to the back stack
               .addToBackStack("gpsFrag")
               .commit();
            }
        });
      }
   }
}


private GPSItem nextItemWithNote(GPSItem gpsItem){
   GPSItem curr = gpsItemList.get(gpsItem.getIncr()+1);
```

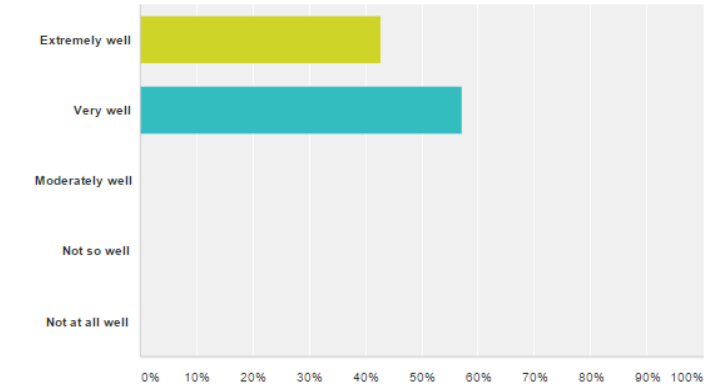
```
    while (curr.getNote().equals("")){
      //last item will always have a note
      curr = gpsItemList.get(curr.getIncr()+1);
    }
    currentNo = curr.getIncr();
    return curr;
  }

  private GPSItem prevItemWithNote(GPSItem gpsItem){
    GPSItem curr = gpsItemList.get(gpsItem.getIncr()-1);
    while (curr.getNote().equals("")){
      //first item will always have a note
      curr = gpsItemList.get(curr.getIncr()-1);
    }
    currentNo = curr.getIncr();
    return curr;
  }
```

D. Results from London Loop App Survey

Q1 Customize Export

Having used the app, how well do you feel you understand what the London Loop?

Answered: 7 Skipped: 0

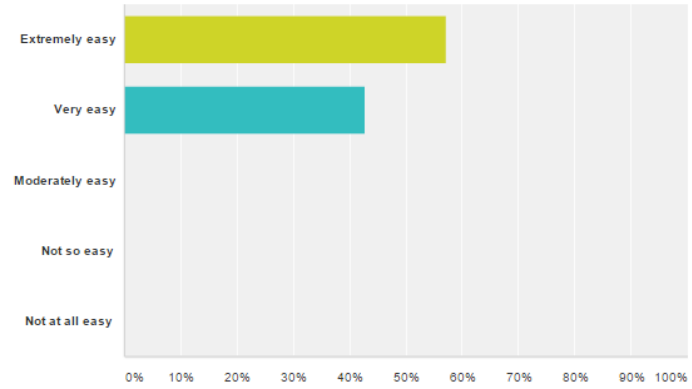


Answer Choices	Responses	
Extremely well	42.86%	3
Very well	57.14%	4
Moderately well	0.00%	0
Not so well	0.00%	0
Not at all well	0.00%	0
Total		7

Q2 Customize Export

How easy was it to find what you were looking for on the app?

Answered: 7 Skipped: 0

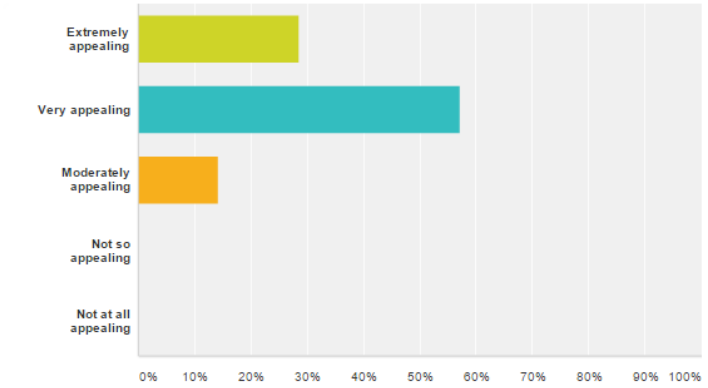


Answer Choices	Responses	
Extremely easy	57.14%	4
Very easy	42.86%	3
Moderately easy	0.00%	0
Not so easy	0.00%	0
Not at all easy	0.00%	0
Total		7

Q3 Customize Export

How visually appealing is the app?

Answered: 7 Skipped: 0

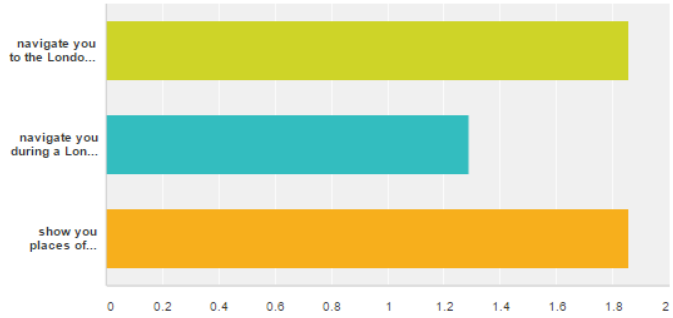


Answer Choices	Responses	
Extremely appealing	28.57%	2
Very appealing	57.14%	4
Moderately appealing	14.29%	1
Not so appealing	0.00%	0
Not at all appealing	0.00%	0
Total		7

Q4 Customize Export

How well does the app..

Answered: 7 Skipped: 0

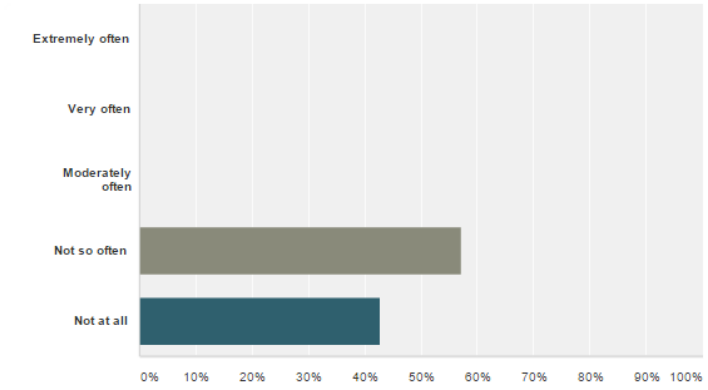


	Extremely well	Very well	Moderately well	Not so well	Not well at all	Total	Weighted Average
navigate you to the London Loop?	28.57% 2	57.14% 4	14.29% 1	0.00% 0	0.00% 0	7	1.86
navigate you during a London Loop walk?	71.43% 5	28.57% 2	0.00% 0	0.00% 0	0.00% 0	7	1.29
show you places of interest during the walk?	28.57% 2	57.14% 4	14.29% 1	0.00% 0	0.00% 0	7	1.86

Q5 Customize Export

How often does the app crash/freeze?

Answered: 7 Skipped: 0

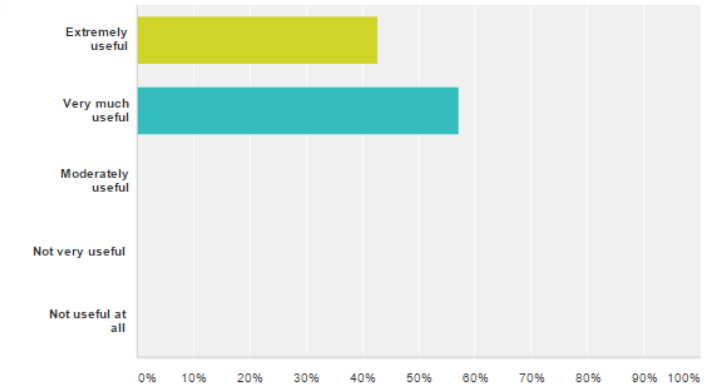


Answer Choices	Responses	
Extremely often	0.00%	0
Very often	0.00%	0
Moderately often	0.00%	0
Not so often	57.14%	4
Not at all	42.86%	3
Total		7

Q6 Customize Export

How much more useful is the app compared to the book to navigate the London Loop?

Answered: 7 Skipped: 0



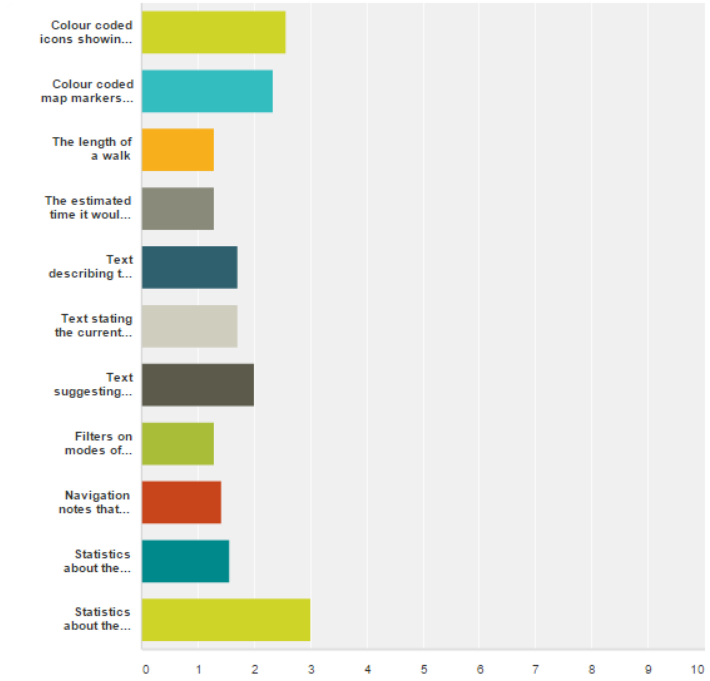
Answer Choices	Responses	
Extremely useful	42.86%	3
Very much useful	57.14%	4
Moderately useful	0.00%	0
Not very useful	0.00%	0
Not useful at all	0.00%	0
Total		7

Q7

Customize Export

How do you rate each of the following features in the app?

Answered: 7 Skipped: 0

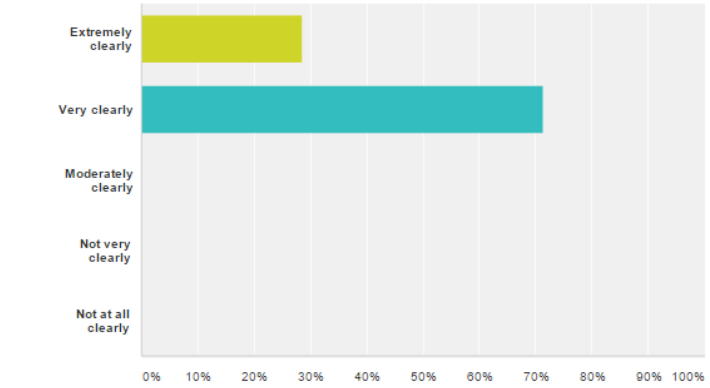


	Extremely useful	Very useful	Moderately useful	Not very useful	Not at all useful	Total	Weighted Average
Colour coded icons showing which walks are in South/NorthWest/NorthEast London	28.57% 2	14.29% 1	28.57% 2	28.57% 2	0.00% 0	7	2.57
Colour coded map markers showing which walks are in South/NorthWest/NorthEast London	33.33% 2	0.00% 0	66.67% 4	0.00% 0	0.00% 0	6	2.33
The length of a walk	71.43% 5	28.57% 2	0.00% 0	0.00% 0	0.00% 0	7	1.29
The estimated time it would take to complete the walk	71.43% 5	28.57% 2	0.00% 0	0.00% 0	0.00% 0	7	1.29
Text describing the walk	42.86% 3	42.86% 3	14.29% 1	0.00% 0	0.00% 0	7	1.71
Text stating the current day's weather forecast	42.86% 3	42.86% 3	14.29% 1	0.00% 0	0.00% 0	7	1.71
Text suggesting attire due to the current day's weather forecast	42.86% 3	28.57% 2	14.29% 1	14.29% 1	0.00% 0	7	2.00
Filters on modes of transport	85.71% 6	0.00% 0	14.29% 1	0.00% 0	0.00% 0	7	1.29
Navigation notes that depend on your location on the walk	57.14% 4	42.86% 3	0.00% 0	0.00% 0	0.00% 0	7	1.43
Statistics about the user's progress	42.86% 3	57.14% 4	0.00% 0	0.00% 0	0.00% 0	7	1.57
Statistics about the global use of the app	14.29% 1	0.00% 0	57.14% 4	28.57% 2	0.00% 0	7	3.00

Q8 Customize Export

Above is an image of the logo I created for the app. How clearly does it describe the London Loop?

Answered: 7 Skipped: 0



Answer Choices	Responses	
Extremely clearly	28.57%	2
Very clearly	71.43%	5
Moderately clearly	0.00%	0
Not very clearly	0.00%	0
Not at all clearly	0.00%	0
Total		7

Q9 Export

Please can you provide any additional feedback for the app?

Answered: 5 Skipped: 2

Responses (5) | Text Analysis | My Categories

PRO FEATURE
Use text analysis to search and categorize responses; see frequently-used words and phrases. To use Text Analysis, upgrade to a GOLD or PLATINUM plan.

Upgrade Learn more »

Categorize as... Filter by Category Search responses

Showing 5 responses

This app effectively guided me along the London Loop walk from Hatton Cross to Hayes and Harlington and showed me interesting places on the way.
6/17/2015 5:59 PM View respondent's answers

Using the app made me want to go on more walks to discover the London Loop
6/17/2015 3:25 AM View respondent's answers

It was fun to use this app.
6/16/2015 9:25 AM View respondent's answers

Very nice app to use. Functionality did everything I needed to efficiently reach the starting point by public transport, and then help me successfully navigate the section from Hatton Cross to Hayes & Harlington, providing useful information on route.
6/15/2015 12:48 PM View respondent's answers

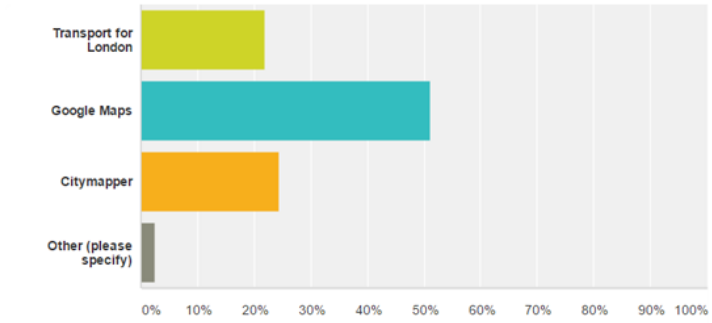
The app for the London Loop does what it sets out to do very well. I now feel well versed in the London Loop even though i had no prior knowledge of it beforehand. I feel confident that I could rely on the app through any of the walks it includes and much prefer it to the guide books. The app definitely encourages me to go out and complete all of the walks with the seemingly effortless way that it navigates me from start to finish.
6/15/2015 11:02 AM View respondent's answers

E. Results from App Survey

Q1 Customize Export

Which app or website do you prefer using to plan routes?

Answered: 41 Skipped: 1

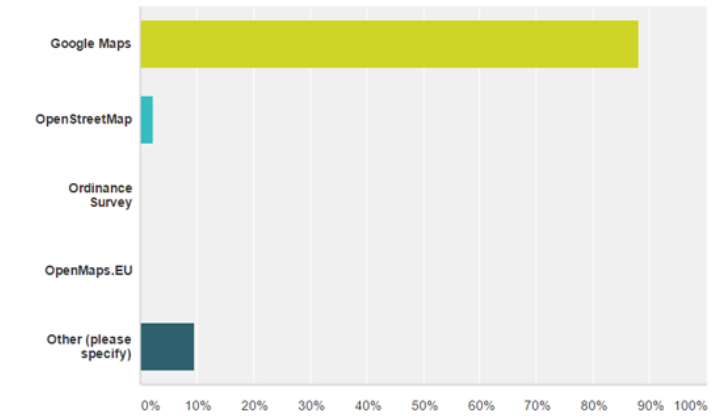


Answer Choices	Responses	
Transport for London	21.95%	9
Google Maps	51.22%	21
Citymapper	24.39%	10
Other (please specify) Responses	2.44%	1
Total		41

Q2 Customize Export

Which app/website do you prefer to use when looking up a place online?

Answered: 42 Skipped: 0

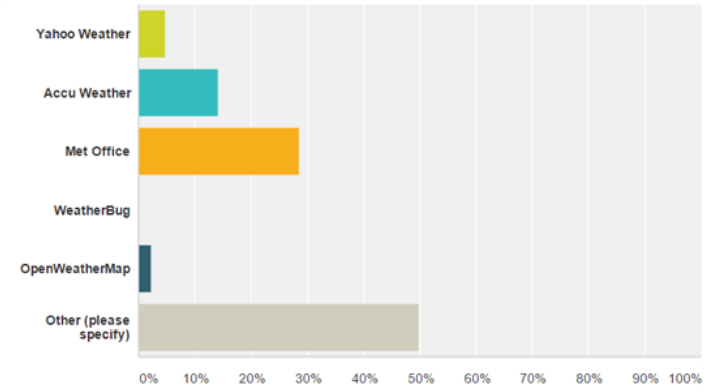


Answer Choices	Responses	
Google Maps	88.10%	37
OpenStreetMap	2.38%	1
Ordinance Survey	0.00%	0
OpenMaps.EU	0.00%	0
Other (please specify) Responses	9.52%	4
Total		42

Q3 Customize Export

What is your preferred method of checking the weather?

Answered: 42 Skipped: 0

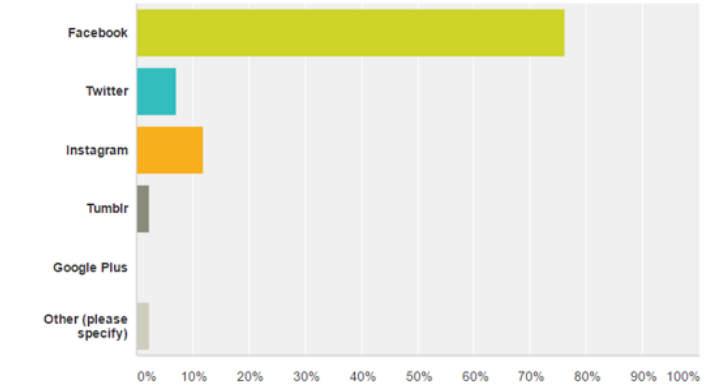


Answer Choices	Responses	
Yahoo Weather	4.76%	2
Accu Weather	14.29%	6
Met Office	28.57%	12
WeatherBug	0.00%	0
OpenWeatherMap	2.38%	1
Other (please specify) Responses	50.00%	21
Total		42

Q4 Customize Export

Which social media website do you use the most?

Answered: 42 Skipped: 0



Answer Choices	Responses	
Facebook	76.19%	32
Twitter	7.14%	3
Instagram	11.90%	5
Tumblr	2.38%	1
Google Plus	0.00%	0
Other (please specify) Responses	2.38%	1
Total		42