University of Hull

A Mobile Application for Open Day Tour Guides

By

Matthew J Ashworth

Dissertation

Submitted for the BSc in Computer Science

May 2018

Word Count: 8624

Table of Contents

[1. Introduction 4](#_Toc513577065)

[1.1 Project Understanding 4](#_Toc513577066)

[1.1.1 Smartphone Usage 4](#_Toc513577067)

[1.1.2 Technology in the Open Day Sector 5](#_Toc513577068)

[1.2 Project Goals 5](#_Toc513577069)

[1.3 Use Case 7](#_Toc513577070)

[1.3.1 Tour Guides 7](#_Toc513577071)

[1.3.2 Open Day Organisers 7](#_Toc513577072)

[2. Aims and Objectives 7](#_Toc513577073)

[2.1 Objective 1 – Canvas API 8](#_Toc513577074)

[2.2 Objective 2 – User account system 8](#_Toc513577075)

[2.3 Objective 3 - Backend Server 8](#_Toc513577076)

[2.4 Objective 4 – Timeline View 8](#_Toc513577077)

[2.5 Objective 5 – Real Time Location Updates and Google Maps API 9](#_Toc513577078)

[2.6 Objective 6 – Push Notifications 9](#_Toc513577079)

[2.7 Objective 7 - Organiser Counterpart/ Open Day Setup 9](#_Toc513577080)

[3. Background 9](#_Toc513577081)

[3.1 Problem Comprehension 9](#_Toc513577082)

[3.2 Similar apps and projects 9](#_Toc513577083)

[3.3 Authors prior use of technologies within the project 10](#_Toc513577084)

[3.3.1 Timeline View 10](#_Toc513577085)

[3.4 Programming Languages 10](#_Toc513577086)

[3.5 Integrated Development Environments 10](#_Toc513577087)

[3.6 Comparison of Technologies 11](#_Toc513577088)

[3.6.1 Android 11](#_Toc513577089)

[3.6.2 iOS 11](#_Toc513577090)

[3.6.3 Windows Phone 12](#_Toc513577091)

[3.5 Alternative Solutions 12](#_Toc513577092)

[3.5.1 HTML5 12](#_Toc513577093)

[3.5.2 Xamarin 13](#_Toc513577094)

[3.5.3 Universal Windows Platform (UWP) 13](#_Toc513577095)

[3.6 Processes and Methodology 13](#_Toc513577096)

[3.6.1 Google Maps API 13](#_Toc513577097)

[3.6.2 Custom Backend Server Deployment 13](#_Toc513577098)

[3.6.3 Algorithms 14](#_Toc513577099)

[3.6.4 Firebase 14](#_Toc513577100)

[3.6.5 Parse Server Platform 15](#_Toc513577101)

[3.6.6 Canvas API 15](#_Toc513577102)

[3.7 Potential Problems 15](#_Toc513577103)

[3.7.1 Canvas 15](#_Toc513577104)

[4. Technical Development 16](#_Toc513577105)

[4.1 System Design 16](#_Toc513577106)

[4.1.1 Class Design 16](#_Toc513577107)

[4.1.2 Use Case Diagram 16](#_Toc513577108)

[4.1.3 Implementation Layout 18](#_Toc513577109)

[4.1.4 API Design 19](#_Toc513577110)

[4.1.5 Database Design 19](#_Toc513577111)

[4.1.6 Coding Convention 19](#_Toc513577112)

[4.2 UI Design 19](#_Toc513577113)

[4.3 Testing Design 19](#_Toc513577114)

[4.4 Implementation 19](#_Toc513577115)

[5. Evaluation 19](#_Toc513577116)

[5.1 Potential Risks 19](#_Toc513577117)

[5.2 Ethical Concerns 19](#_Toc513577118)

[5.2.3 User Location 19](#_Toc513577119)

[5.2.4 User Data 19](#_Toc513577120)

[5.3 Risk Analysis 19](#_Toc513577121)

[5.4 Project Achievements 20](#_Toc513577122)

[5.5 Objective Evaluation 20](#_Toc513577123)

[5.6 Further Work 21](#_Toc513577124)

[6. Conclusion 21](#_Toc513577125)

[6.1 Learning Outcomes 21](#_Toc513577126)

[Appendices 21](#_Toc513577127)

[References 21](#_Toc513577128)

# Introduction

This dissertation has six sections in total that will aim to provide detail on all phases of the project. From conception, realisation and development through to project completion. The six phases of this report include:

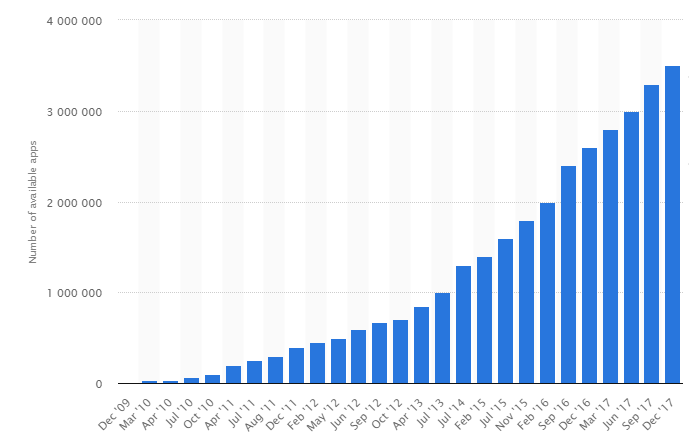
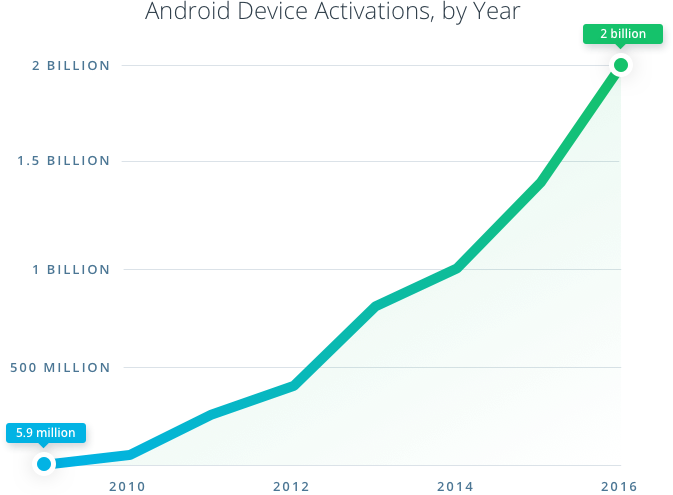
* **Aims & Objectives:** Detail what the project aspires to achieve by splitting up the project into task based objectives.
* **Background:** Description of the problem context and the necessary background information and comparisons of the technologies chosen for this application.
* **Technical Development:** UI Design, algorithm design and feature showcase with an exhaustive comprehension on implementation and testing.
* **Evaluation:** Summary of the project progress and achievements, with personal reflection and any further improvements that would be made.
* **Conclusion:** Overall review of the project and reflection on the aims and objectives set.

## Project Understanding

### Smartphone Usage

The use of smartphones in our daily lives is on the rise with 85% of adults in the UK owning a smartphone, up from 55% in 2012 (Consultancy.uk, 2017). Smartphones over the years have become increasingly more powerful and productive to the point where smartphones are replacing laptop computers. While smartphone adoption has seen a year on year rise, laptops have fluctuated over the years peaking in 2015 with 79% adoption amongst adults in the UK against 78% in 2017 (Consultancy.uk, 2017). More adults in the UK own a smartphone than a laptop computer, and is it any wonder? Productivity tools such as Microsoft Office were once exclusive to more traditional desktop and laptop computers but now are available on mobile platforms such as iOS and Android (Microsoft, 2013). With the combined functionality of productivity, entertainment and communication smartphones are shaping the world we live in. Android is the leading mobile operating system in terms of market share with more than 2 billion active Android devices in December 2016 (*Figure 2).*

To enable such growth in the smartphone market, hardware and software advances have gone hand in hand to make smartphones what they are today. Smartphone processor architecture advanced from single core devices to octa-core devices in a matter of years and flagship devices containing as much if not more memory (RAM) than modern day laptops. Mobile software has always pushed the boundaries of available hardware limits and in turn has given us some revolutionary new ways of working, playing and interacting.

*Figure 1: Apps available on Google Play* (Statista, 2017) *Figure 2: Global active Android devices* (Udacity, 2017)

### Technology in the Open Day Sector

This project will utilise this ever changing smartphone technology to achieve something that hasn’t really been accomplished in the market yet. An application that will assist open day tour guides with the day’s proceedings. With today’s market of technology in the work place and Universities being at the forefront of new technologies an application like this is more relevant now than ever. The traditional open day de facto of having a notebook with the day’s schedule and notes about the university or memorising it all to mind and hoping nothing goes wrong is over. This project aims to replace all that with a modern, more efficient tool that brings new, exciting and useful features to open day tour guides.

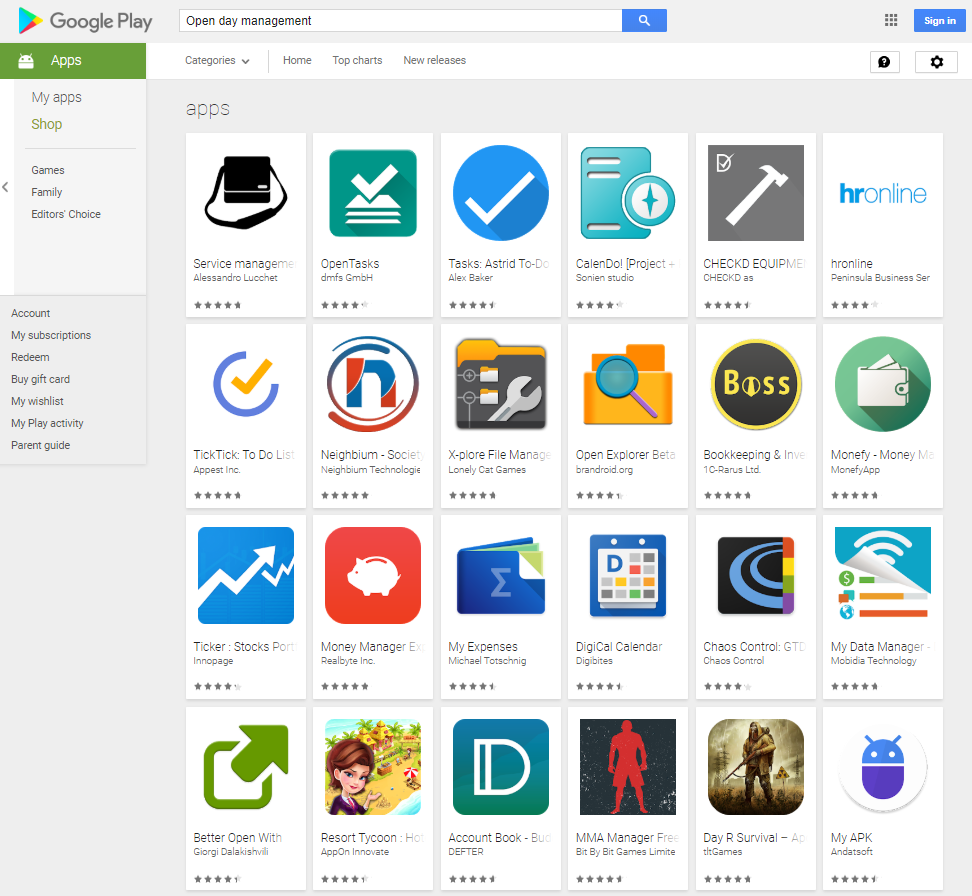
## Project Goals

The intended goals of the application are to present a professional open day tour guide management client that the open day organisers can use to manage their open day tour guides and that tour guides can use to assist them in the day’s proceedings. The application will facilitate many different features, some exclusive to the type of user.

A tour guide should be presented with a map of the campus showing all the spots he/she should cover in the tour. A schedule of events should be given showing when to arrive at each spot. Information about each location should be provided to the tour guide upon arriving at a location so that should the tour guide need any assistance in talking about a specific topic he/she can look to the application for help. Data about the tour guides location should be relayed back to the organisers for greater control and management of the open day.

The open day organisers will be presented with the same application but with different functionality allowing them to create and tailor an open day tour guide experience to their liking. Such functionality will include creating and assigning tour guide accounts, creating route, locations to visit and the ability to key in information about said locations for the tour guide to use. Organisers will be able to view the location of all tour guides on a real time map and have the ability to push notifications to a specific or all tour guides.

In order to achieve this, research was conducted on the availability of similar open day tour guide management solutions across various platforms and marketplaces.



In conclusion to the research conducted, there was not one publically available application on any platform that provided the solutions intended by this project. It is the understanding of the author that while no such solution was found to be publically available it is more than likely that Universities and other such organisations that host open days have their own private or in house solution. The closest publically available applications found we’re that of university open day companion apps designed for prospective students. While these apps did feature some technologies that will be present in this project such as mapping with on campus locations. None offered anything close to the solution this project intends to provide.

The intention of this project is not to be publically available but initially start out at the University of Hull and potentially branch to other universities and organisations for a fee.

## Use Case

For this stage of the report two use cases will be explained and demonstrated as there are two types of end user, the tour guide themselves and the open day organisers.

### Tour Guides

A tour guide arrives at the university or organisation ready for the open day. They should be able to open the application on their own device or a university/ organisation issued device and login to their tour guide account. Login information should have been provided to them beforehand. Once logged in the tour guide will see that they have been assigned one or more tour groups. From here they can see what route around campus they should take, what locations to stop by and talk about as well as any necessary talking points about said locations. The tour guide will see there is a timeline view that updates automatically as the tour progresses. From here everything is relatively seamless and automatic. The schedule will start once the time has been reached and the timeline will progress automatically. The tour guides location will be tracked and they will be told via a notification if they are behind or ahead of schedule. Should the tour guide need to they are able to contact the organisers through the application.

### Open Day Organisers

An organiser should be able to open the application on their own device or a university/organisation issued device and login with their organiser account. From here they will be presented with a menu of options ranging from creating and assigning new tour guides, modifying or creating a new schedule, viewing the location of all or a specific tour guide and issuing messages to all or a specific tour guide.

# Aims and Objectives

The following aims and objectives are for the development of a mobile application for open day tour guides, an Android app to assist open day planning and navigation.

The app for use by university tour guides will dynamically give a unique route to each tour guide around the university set by the organisers. It contains a timeline schedule for tour guides to stick to and be able to tell the guide how long to stay at each location as well as giving them any necessary facts or interesting information about each location to use should they be unsure of the script.

The application allows organisers to create an open day event and assign tour guides to a schedule of their creation. The application will be able to determine the location of each individual open day group at any time and also be able to send urgent messages to a tour guide in the case of changes in arrangements.

The app will also feature mapping that is able to route the guide around the university to each of their assigned locations at the assigned time and allow organisers to view the locations of all tour groups.

The aim specified above will be met by the following objectives. These objectives are cross sections of the app showing how its inner workings will all come together to create the end product discussed in the goals of the project.

## 2.1 Objective 1 – Canvas API

A decision was made very early in development to authenticate each user of the app with Canvas before they are allowed to use the app fully. In order to register as a tour guide the user will first have to sign in through Canvas. This decision was made first and foremost to prevent any unauthorized access to the open day schedule and to protect the end schedule server from any unnecessary requests. In addition to this the application receives information about the user from Canvas such as session cookies, usernames and full names. This allows the application to feel more personal and tailor the experience to the individual user.

## 2.2 Objective 2 – User account system

Due to the nature of this project, data will have to be stored externally to create a usable dynamic system. Thus each tour guide will be assigned an account which stores their information such as name and email as well as their schedule, and location data such as details about each specific location on their schedule. This data will be stored on a server. For this project the server is external to the University but in a real world scenario could be on campus. The app then pulls this required data from the server when they log into their account upon opening the app. Each user account will be tied to a specific open day at a specific university. For example, user 510407 could be a tour guide at the University of Hull on a Saturday, whereas user 408183 could be a tour guide at the University of Manchester on a Friday. Each username stored on the server is inherited from the Canvas API thus no two users will be the same and each University will have their own university prefix such as UoH510407 for UoM408183. Organisers will be able to create accounts, tie an account to a specific schedule and day and remove accounts if necessary. Organiser accounts can be set up by the developer in agreement with the University.

## 2.3 Objective 3 - Backend Server

As mentioned in the previous objective, data will need to be stored externally to allow for reusability and robustness of the app. If the app is to be used on more than one occasion at the University or at several other Universities then information about the time schedule, location data and other unique University related items shouldn’t be presented to the developer to develop a new version of the application for each new open day for each university. Instead the app should have a counterpart where the organisers can manage the open day schedule. For this reason, a backend server should be implemented. For user friendliness the organisers shouldn’t have to manually enter data about the open day into the server database instead the organisers will have their own counterpart within the base application to enter specific open day information into the database.

## 2.4 Objective 4 – Timeline View

At the core of this project is the app’s ability to display a schedule to the tour guide. For this project a timeline will be implemented. A timeline allows the tour guide to quickly and easily see where they need to be and when. The timeline would be dynamic updating automatically as the tour goes on. It would show the tour guide what location is next and at what time. A little bit of information will also be provided about each location within the timeline fulfilling the aim to be able to provide any necessary facts or suitable words to use should the guide be unsure of the script.

## 2.5 Objective 5 – Real Time Location Updates and Google Maps API

Real time location data allows the app to alert the tour guide when it’s time to leave a location. This can be done with a push notification or by vibrating the tour guides device.

The Google Maps API will be used in two ways. The first being that it can show the tour guide the route around campus to take and also show them where any locations are should they be unsure. Secondly it will be used to show the organisers where each tour group is at any given time.

## 2.6 Objective 6 – Push Notifications

Push notifications are implemented across the application for numerous purposes. The main function is to allow tour guides to receive messages from the organisers should there be a change of events. However in addition to this push notifications will alert the tour guide if they are behind or ahead of schedule. This helps keep the open day flowing in a functional and controllable manor.

## 2.7 Objective 7 - Organiser Counterpart/ Open Day Setup

In order to ensure the end product is easily reusable, part of the Open Day Guidance app will be locked off from tour guides and will be for organiser accounts only. This part of the app is for university officials and organisers to set up and control their open day guides. The organiser would select the day(s) of the open day and specify how many tour guides there needs to be. The app would then go ahead and create the specified amount of tour guide users and assign them each a blank time schedule. The organiser would then specify a location and a time for each group to be at. Finally, the organiser would input any key words or facts for each location. After the open day has finished user accounts can be retired or retained for future use. Within this section of the app the organisers will be able to modify user data, location data or timetables and view a real time map of where each tour group is at any point during the session. Additionally to this the organisers will be able to push notifications to each tour guide from this section of the app. They will be able to select a tour guide and write a message which will be pushed to the tour guides app for them to read.

# Background

## Problem Comprehension

Prior research was conducted into the formation of the aims and objectives of this project. In order to reach these goals certain concepts, technologies and principles need to be explained and understood before implementation of these technologies began in development.

## Similar apps and projects

The current public market for open day applications is sparse. Some universities have their own open day applications such as Liverpool’s Virtual Open Day app (Revolution Viewing, 2017) and Loughborough’s University Open Day App (Loughborough University, 2017). However these apps are tailored to prospective students that will be visiting an open day, not tour guides operating an open day. A current solution that aims to achieve similar things to this project is Tour Buddy by Tour Buddy Apps (Tour Buddy Apps, 2018). Tour Buddy incorporates many of the technologies this project aims to implement such as push notifications, geo location triggers, interactive GPS maps and location detail screens. However Tour Buddy is a solution for companies to present to visitors. Tour Buddy removes the need for a tour guide and the solution effectively becomes the tour guide. This project does not aim to replace the tour guide, rather assist them. Despite the different target audiences, Tour Buddy shares a lot with what this project aims to achieve and thus will be a source of inspiration.

## 3.3 Authors prior use of technologies within the project

Many of the technologies and algorithms used within this project are home-grown and have been used in publically available apps and projects before this. The author has an extensive knowledge and background in Android development and some of the features that will be discussed have used purpose written technologies and algorithms that will be applied to this project.

### 3.3.1 Timeline View

A large part of this project relies on the use of a timeline to show the tour guide a chronological list of locations they should be at and when. See Appendix C for details. The timeline algorithm was developed by the author for a project published in May 2017 (Ashworth, 2017).The use of this timeline has been modified and applied to this project.

## 3.4 Programming Languages

Before development on the project began a platform had to be selected. Each platform has its own benefits and drawbacks along with its own programming language in most cases. An application of this nature would require a rather great understanding of the required programming language(s). Below is a table showing the breakdown of platforms and their corresponding programming languages as well as the developers experience and the availability of resources that may aid development.

|  |  |  |  |
| --- | --- | --- | --- |
| Platform | Language(s) | Experience | Available Resources |
| Android | Java, Kotlin, C# | Experienced | Many |
| iOS | Swift, Objective-C | Little | Many |
| Windows Phone | C# | Some | Some |

Ultimately this project is for the creation of an Android application so Android will be the chosen platform however objectively speaking given the opportunity of releasing on any desired platform Android would be the chosen initial platform anyhow. The combination of the developers experience and the availability of resources to aid development makes Android the perfect platform. The developer is competent in both Java and Kotlin, the native Android development languages. However C# could be used via Xamarin should the developer wish.

In order to achieve a working presentable application in the limited time period, work was started before the full conception of this project was realised. Small projects were created such as the Canvas API, Backend server integration and Google Maps feature in order to fully understand how these could be implemented into the final application. After the deliverables were set it was the developer’s prerogative to use past home grown algorithms and libraries used in other projects in order to advance the development of this project given the limited timescale. Libraries such as the Timeline View are used in other publically available applications by this developer but made sense to implement within this project.

## 3.5 Integrated Development Environments

Like most projects a specific Integrated Development Environment (IDE) will be used to develop this project dependant on the language and platform. Since Android was chosen as the platform there are four main IDEs available to develop android applications in. Android Studio, Xamarin, Eclipse and Visual Studio (**Figure X)**. With Visual Studio being limited in the development of Android applications.

Insert Figure X: IDE visualisations

Due to the competency of the developer in Android Studio, Java, Kotlin, Android Studio was chosen as the development IDE of choice.

## Comparison of Technologies

### Android

Android is currently at the time of writing the market leader of mobile operating systems by a long way. As of quarter 2 2017 Android holds 87.7% of the global mobile OS market share (Statista, 2017). This puts Android head of all competing mobile operating systems such as iOS and Windows Mobile, the latter being scheduled to discontinue in December 2019. As a result of this software released on the Android OS has the widest possible audience.

Android’s success can be largely accredited to the platform being Open Source (Google, 2018). Due to the open source nature Android can be found on smartphones from hundreds of manufacturers across the globe, this allows these companies and manufacturers to use an operating system that is already built and fully realised and thus reduce development costs as they do not need to develop their own operating system. Similarly, being Open Source users are able to tweak and modify their device without fear of voiding warranty in most cases.

Developing for the Android platform had a rocky start. Initially apps were created in the Eclipse IDE which was ill received due to its stale developing style and the countless amount of bugs. Eclipse is a Java IDE that was repurposed to allow development of Android applications. However as of December 2014 Google have released stable versions of a new IDE, Android Studio. Android Studio is based off the community version of JetBrains’ IntelliJ IDEA and is a purpose built IDE designed for Android development (Google, 2013). Android Studio is a much more complete mobile IDE than Eclipse and boasts features such as Gradle, Maven, Improved GUI Designer, Auto Completion and Unit Testing built in.

Android is also a much more accessible platform to develop for. Android applications can be developed on Mac, Windows or Linux for free and can be published in the Google Play store for a one time developer fee of $25 (Google, 2018). Google’s developer account fee of $25 is the second cheapest mobile developer fee available, behind Windows Mobiles $19 for individuals, however corporate organisations are required to pay $99 (PROTALINSKI, 2014).

### iOS

The iOS platform market share is currently on the decline at 12.1% in Q2 2017 against 17.9% in Q4 2016 (Statista, 2017). That’s a 75.6% difference compared to Android.

Being on the majority of smartphones globally you would think Android applications are more successful than their iOS counterparts however this is not always the case. In Q4 2017 iOS App Store users spent a total of $11.5 Billion dollars against Google Play users spending only $5 Billion (Dilger, 2017). That’s a 95% lead for the iOS App Store over Android’s Google Play. It has been widely known for years that while Android has a greater market share, Apple’s users tend to spend more money.

There’s no doubt that even though iOS market share is falling it is still the most popular platform to develop for. This is in part due to the defragmentation of iOS versions across all devices. The majority of all active iOS devices are running the latest version of iOS making applications much easier to develop and much more accessible. 85.84% of all active iOS devices are running the latest version of iOS at the time of writing (Apteligent, 2018) (**Appendix 1)**. Android on the other hand is much more fragmented with the majority of its active devices spread across 8 versions of its operating system with the latest version having only a 0.03% adoption rate 3 months after its public release (Hollander, 2017).

iOS is a closed source platform unlike Android’s Open Source. This presents a few development limitations such as the inability to draw over other applications and Apple’s strict regulations on the use of NFC within applications. Additionally iOS applications have a longer approval time for them to be published on the store. It is not unlikely that an application could go days even weeks waiting for approval, however Apple recently took steps to improve this and now typically apps are approved or denied within a day (DORMEHL, 2016). Android uses a semi-automatic process that usually processes the approval or rejection in a matter of hours. Additionally the development of iOS applications requires the Mac OS operating system and cannot be developed in a Windows or Linux environment. This limits the potential developers for the platform as Mac OS machines are typically £1000 plus whereas a capable Linux or Windows development machine can be had for less than half that price. Furthermore Apple require an annual $99 enrolment fee in order to publish apps on the App Store. This further reduces the number of potential developers such as students as $99 annually is a fair amount of money compared to Google’s onetime fee of $25.

### Windows Phone

Windows phone not only has a small market share at 0.3% in Q4 2016 and negligible as recent as Q2 2017 (Statista, 2017), but also the smallest app marketplace compared to Android and iOS, Venture Beat’s Emil Protalinski estimates that the Windows Store has 850,000 apps on the store (Protalinski, 2016). More recent data is estimated and inaccurate due to the low market share of Windows 10 Mobile. With such a small app store, apps that are published to the Windows store are more likely to gain attention and traction as it’s a less crowded market.

Windows 10 Mobile is scheduled to be discontinued in December 2019 due to low market share, user adoption and the lack of applications for the platform. This no doubt has occurred because Windows Mobile came so late to the smartphone operating system market and by that point competing platforms already had large user bases and a good range of apps available. Windows Mobile fundamentally didn’t bring anything new to the table and users ultimately decided not to give up the current eco system they were apart of.

Windows Phone applications are also limited in development as they require the Visual Studio IDE which at the time of writing is only available on Windows (Microsoft, 2018).

## 3.5 Alternative Solutions

### 3.5.1 HTML5

HTML5 can be used to develop apps for all three major platforms, Android, iOS and Windows 10 Mobile. Its main advantages include; cross platform portability, being able to build and release on all platform instantaneously. It can also utilize hybrid apps whereby an app is created and then wrapped for each platform, making updates quick and easy. This also reduces costs when it comes to development as only one language is required.

However, HTML5 has its drawbacks in that the speed and efficiency of apps are much slower than native applications as they are not specifically tailored or optimised for that specific platform. Another drawback is the inability to access most of each platforms independent specific hardware, meaning functionality is decreased too.

### 3.5.2 Xamarin

Xamarin is a middleware solution to developing apps. It uses a C# shared codebase to write Android, iOS and Windows apps in their native format keeping the performance and optimisation that you would see on a natively written application.

It is available to Windows and Mac OS through the use of Visual studio and Xamarin Studio respectively. It’s C# codebase allows applications developed in Xamarin to run on the billions of devices across the three main platforms. It can share on average 75% of app code(Lanaguage, APIs, Data Structures) and nearly 100% of all user interfaces making it extremely versatile with the portability to multiple platforms seamlessly without having to learn each platforms native languages.

The only drawback is the price as Xamarin studio is particularly expensive compared to all other solutions. The lowest package being $300 per year for access to deploy to App Stores. Higher tier packages range up to $1899 per year for access to all Xamarin features available.

### 3.5.3 Universal Windows Platform (UWP)

## 3.6 Processes and Methodology

Having chosen the platform, development language and IDE, research was conducted in order to achieve some of the more technical features of the project.

### 3.6.1 Google Maps API

Google Maps is used in thousands of applications worldwide and is an industry leading navigation service. Started in October 2004 (Google LLC, 2004) Google Maps is said to be the world’s most popular app for smartphones with 54% of global smartphone users having used the app at least once (Smith, 2013). The Google Maps API will be used within this project for its well documented public API and for the reliability of the service. Many other apps not limited to open day apps make use of the Google Maps API to provide location data and navigation to its users. For example, Uber uses the Google Maps API for its drivers and customers to provide maps and location data to its customers and similarly maps, location and navigation data to its drivers. This project will be taking a similar approach to how Uber uses the Google Maps API in that it will provide mapping and location data to tour guides and organisers and additionally navigation data to tour guides.

### 3.6.2 Custom Backend Server Deployment

The last option considered was creating a backend from scratch which would be managed and developed by hosting self-created code. The benefits primarily would be having complete control of the data that is passed to the server and back as well as the amount that can be used. There are no limitations that have to be considered unlike using an API service like Parse/Firebase.

The advantages of a custom backend server are:

* Fully customisable
* Developers choice of backend language (most likely Python or Ruby on Rails)
* No limits on number of requests or features available (other than hardware limitations)

The disadvantages of a custom backend server are:

* Developer must be knowledgeable in server side scripting and deploying servers.
* Server platform will take time to develop whereas solutions like Firebase and a Parse Server are ready to go.
* Additional costs for buying and running own servers

### 3.6.3 Algorithms

Many algorithms are used within this project and a key algorithm that needed research and extensive deliberation is the use of a sorting algorithm. A sorting algorithm is needed within this project on many occasions. For example the timeline needs to be sorted into chronological order so that the first location isn’t at 14:00 and the second location at 13:30 for example. Another area of the project that needs sorting is user data so that organisers and officials can easily access and change a record without having to search through the whole database because it’s unsorted.

While researching sorting algorithms a few key areas needed to be evaluated. These included; algorithm efficiency, complexity and the ability to be used on wide sets of data from numbers to string and arrays. From the research gathered it looked initially like a bubble sort would be a suitable algorithm. While bubble sorts are relatively slow this project wouldn’t be handling large sets of data and a bubble sort would be more than efficient enough. Coupled with its simple complexity the bubble sort could easily be applied to the project. However a change was made shortly into development as sorting alphabetical characters via a bubble sort is highly inefficient compared to simply sorting integers and real numbers. The sorting algorithm the project will use upon delivery is the quick sort. A quick sort is much more complex but also much more efficient and can relatively easily sort alphanumeric data.

The quick sort is a divide and conquer algorithm and is recursive unlike the bubble sort. It works by selecting a random element from the array, this becomes the pivot. It then rearranges the array based on whether each element value is less than or greater than the pivot. Elements less than the pivot go before it and elements greater than the pivot go after it. After each element has been separated the pivot is then in its correct position. The algorithm then recursively repeats these steps until the array is sorted. See appendix D for visualization and a sample quick sort algorithm.

### Firebase

Firebase offers similar features to Parse and is currently owned by Google. It offers the same benefits of giving developer’s access to an Online and Local database without any backend setup required. However, the main difference is its use of JSON to format data back and forth with the app allowing real-time cloud data service (Firebase, 2016).

The advantages of a Firebase solution would be:

* Free
* Google owned and supported so implementation with Android is good
* Push Notifications
* Good Documentation
* Support for mobile platforms (Android & iOS)

The disadvantages are:

* Non SQL database, storage is in JSON (JavaScript Object Notation)

Firebase seems to be a complete solution for the Android platform this project will be developed for. Firebase being from Google has many tie ins with Android and even Android Studio has the option to automatically configure the Firebase libraries for a project.

### Parse Server Platform

Parse was a company owned by Facebook which provided mobile developers with an API capable of handling local & online databases without any backend setup required. It handled data securely and efficiently in the cloud making server-side logic simpler for any platform. However in 2017 Parse was shut down by Facebook however the platform was Open Sourced meaning that anyone could deploy a Parse Server on their own hardware. Over the past year many companies have sprung up offering a parse server on their hardware for a tiered based pricing platform.

The advantages of a parse server are:

* Free for small projects
* Exhaustive API
* Well Documented
* Support for many platforms (Android, iOS, Mac OS, Windows, etc)

The disadvantages are:

* Limited customisation
* Free tier limits number of requests

Parse allows developers to handle multiple apps each with multiple tables for either data or User Accounts. It’s also possible to store Files in the database directly unlike other databases such as SQLite, meaning user recipe images easier to handle. It also supports ‘relation’ between tables which would mean easier handling of multiple tables connected to one user, for example, a tour guides time schedule and the specific location data such as facts and figures.

### Canvas API

The publicly available Canvas API as made available by Infrastructure is designed for individuals and companies that wish to admin their own Canvas VLE. For an existing VLE such as the one operated by the University of Hull in which this project intends to use for authentication the API is of no use as the developer does not have administrative rights over the current Canvas VLE implementation.

The Canvas API referred to in this report is a private API developed and maintained by the developer. Fundamentally it uses HTTP POST and GET requests to send and receive authentication data.

## Potential Problems

### Canvas

A potential issue may occur with the Canvas API written by the developer specifically for this project. While the application does not breach any terms and conditions the API does access a private endpoint.

# Technical Development

## System Design

### 4.1.1 Class Design

The Class Diagram seen in Figure X shows an overall basic design of the project before major development implementation took place. It shows how each class, activity and fragment will relate to each other in the application. Figure X shows how two different account types present two different designs. Above from account login shows the organiser application design whereas below account login shows the tour guide application design.

Insert figure X: Class Design

Both sides of the application, the Tour Guide and Organiser counterpart are built using fragments. The main class in each related design is FragmentExperience. Essentially FragmentExperience is an adapter which manages the transaction of fragments. Classes such as timelineView and locationMapping are not traditional activities rather fragments that are inflated within FragmentExperience. This allowed for much greater design fluidity and a better user experience. As the organiser has a greater number of options and features available to them it made sense to implement a navigation drawer on that part of the application. Whereas within the tour guide section a pager adapter is used alongside a bottom navigation bar for simplicity as there are only a few fragments accessible by the tour guide.

### 4.1.2 Use Case Diagram

The following is two use case diagrams (Figure X & X) which shows the navigation of the application for both types of users and how each user would reach a specific goal within the application. For each use case diagram there are four actors in the system. The first actor being the end user who is interacting with the application, the second being the Canvas API in order to authenticate user logins. The third being the Parse database that deals with user data and accounts and finally the fourth which is the Google Maps API to give location data and realtime location updates.

Insert Figure X

The triggers for this diagram are:

* “Sign in with Canvas” button is pressed
* Sign in is completed
* “Maps” navigation is pressed
* “Updates” navigation is pressed
* “Send Message” button is pressed

The goals for this diagram are respectively:

* Pass login data to Canvas API and authenticate securely, pass login data to Parse to validate user
* Schedule data is downloaded if not stored on the device locally
* Google Maps API is invoked and returns user location and schedule locations
* Auth token is passed to Parse and update messages are received from server
* Auth token is passed to Parse and message data is send to parse and on to organisers.

Exceptions that may arise in this diagram are:

* A user is logged in elsewhere > Display message saying they are logged in elsewhere and to log out there first. This prevents location issues.
* Data entered into the message field is in the incorrect format > Display message showing the acceptable formats of data.

This diagram presumes that:

* The user has a canvas account
* The user has details registered on the Parse server by the organisers
* The user has a stable connection to the internet
* The correct login details have been entered
* The user has a stable GPS lock

Insert Figure X

The triggers for this diagram are:

* “Sign in with Canvas” button is pressed
* Sign in is completed
* “Maps” navigation is pressed
* “Updates” navigation is pressed
* “Send Message” button is pressed
* “Create Tour Guide” button is pressed
* “Create Schedule” button is pressed
* “Edit Users” button is pressed
* “Edit Schedule” button is pressed

The goals for this diagram are respectively:

* Pass login data to Canvas API and authenticate securely, pass login data to Parse to validate user
* Existing open day user information and schedule information is downloaded if not stored locally already
* Google Maps API is invoked and returns location of all tour guides
* Auth token is passed to Parse and update messages are received
* Auth token is passed to Parse and data is sent to the correct tour guide
* Auth token is passed to Parse and data about new tour guide is created on Parse server
* Auth token is passed to Parse and new schedule data is created on Parse server
* Auth token is passed to Parse server, differences in data is checked and changes are made on Parse server
* Auth token is passed to Parse server, differences in data is checked and changes are made on Parse server

Exceptions that may arise in this diagram are:

* A user is logged in elsewhere > Display message saying they are logged in elsewhere and to log out there first. This clashing versions of schedules and user data.
* Data entered into the message field is in the incorrect format > Display message showing the acceptable formats of data.
* A tour guide user already with the requested username already exists in the system
* Data entered into the message field is in the incorrect format > Display message showing the acceptable formats of data.
* Data entered into create a tour guide fields is in the incorrect format > Display message showing the acceptable formats of data.
* Data entered into create a schedule fields is in the incorrect format > Display message showing the acceptable formats of data.

These Use Case diagram are useful for understanding how the 4 main components which cannot be fully controlled will interact with each other and what would need to be thought about during implementation in order to deal with as many situations as possible.

### 4.1.3 Implementation Layout

The structure of the application was wholly designed to feature two Activities only. The login activity where by either an organiser or tour guide can log into the application and the FragmentExperience class. FragmentExperience is a fragment holder in which the layout is empty but fragments are inflated within that activity. By implementing a fragment transaction manager, users can switch between different pages within the application. In the case of the Organiser this is achieved by a navigation drawer, and with the Tour Guide users it is achieved using a bottom navigation view. Fragments are used as they are much more reusable and less memory intensive than activities.

Figure X shows the overall structure of the application, encompassing both organiser and tour guide accounts.

Insert figure X: Project Stucture

While this implementation allows for a back stack to be constructed. Whereby fragments can be opened and be placed in the stack in chronological order from when the user accessed them and then referred back to in order when the user presses the back key. The developer found this to be counter intuitive and hinder navigation. Fragments should not be disposed of but should also not be accessible via a hardware back key press. The navigation built for this project is simple and very intuitive, if the user wishes to reaccess a page then a simple press or swipe and click is needed. Every page within the application for both organisers and tour guides is reachable within two clicks, most times being only one click. To better explain this Figure X shows a design where a back stack should be implemented and Figure X shows the design of this application where a back stack would be counter intuitive.

Insert Figure X: Back stack implementation design

Insert Figure X: Application implementation design

### 4.1.4 API Design

#### 4.1.4.1 Canvas API

The design of the Canvas API is particularly special

#### 4.1.4.2 Parse API

#### 4.1.4.3 Firebase API

### 4.1.5 Database Design

### 4.1.6 Coding Convention

## UI Design

## Testing Design

## Implementation

# Evaluation

## Potential Risks

## Ethical Concerns

### 5.2.3 User Location

A concern that was identified before development began was that of real time location tracking of a user’s whereabouts. People in the modern age of technology are even more privacy orientated than ever and thus this project should reflect those values. Tracking a user’s location for the purpose of the application should only occur under a few circumstances. Firstly the open day is in session, for the hours before or after the open day GPS tracking of users should be disabled. Secondly the ability to track a user should be confound to a specific location. In this case GPS tracking will only be enabled while on campus. The moment a user steps off campus tracking is disabled.

### 5.2.4 User Data

Another concern that arose before development began was the security of user data stored on the Parse and Firebase Servers. There are a few conditions in which data should be accessible. It should belong to the user that is currently logged in or the organisers of the open day event. To achieve this some measures were taken during database and server design to ensure that user data is attached to a specific user and only accessible by said user or a user with authority. Users are given a unique ID which follows them throughout the application. Data such as schedules, user data and GPS locations all require this user ID along with a secure authentication token to access. The Parse Server already has measures built in on top of what the developer has implemented. Parse handles user data effectively by utilising Access Control Lists, where each Object in the database is assigned an ACL with the same authentication as the user who creates it.

## Risk Analysis

Additionally to the risks predicted at the conception of this project, which are seen in Appendix X. There are some additional risks realised during the development of this project that were initially unforeseen. These risks can be seen below.

Insert table of risks

## Project Achievements

After evaluating the original burn down chart for this project in comparison to the actual development and implementation time it is evident that the project was a success. However this wasn’t without its problems and setbacks. The initial burn down chart suggested that the final stages of development would take place in week 28 (Week commencing 5th March 2018). In actuality the development ran much later than this finishing in week 37.

The 9 week setback can be attributed to the over ambitious goals of the project. Features like the Canvas API, dynamic timeline updates and the integration of both Parse and Firebase solutions took much longer to implement than initially predicted. This when comparing actual development time against the time plans given in the initial report it was wholly inaccurate. The initial time plan which can be seen in appendix X showed a list of tasks and when they were estimated to be completed in chronological order. The actual time plan would show that some tasks ran across the whole development time, some tasks aimed to start development later on actually started rather early on. It is fair to say that the developer did not adhere to this time plan at all.

In terms of development consistency, time spent on implementation and development was in bursts throughout the entire period with the majority of work taking place at the beginning and at the end of the allocated development time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task Number | Priority | Task Name | Status | Duration (Days) |
| 1 | 3 | UI Design | Completed | 5 |
| 2 | 4 | Login | Completed | 2 |
| 3 | 1 | Timeline View (2017 Development) | Completed | 30 |
| 4 | 5 | Interim Report | Completed | 4 |
| 5 | 6 | Integration of Parse Server | Started | 5 |
| 6 | 7 | Tie Parse Users and Parse Data together | Started | 6 |
| 7 | 2 | Google Maps | Started | 5 |
| 8 | 11 | Dynamically update timeline | Not Started | 10 |
| 9 | 9 | Firebase and push notifications | Not Started | 5 |
| 10 | 8 | Organiser Counterpart | Not Started | 15 |
| 11 | 10 | Canvas API | Started | 10 |
| 12 | 12 | Testing | Not Started | 5 |
| 13 | 13 | Final Development | Not Started | 3 |
| 14 | 14 | Final Report | Completed | 20 |

As the developer had never taken on a task as ambitious as this project, in combination with limited knowledge of some of the technologies used within this project, greater time management would have greatly eased the development of this project. Even though the project was a success the developer believes with greater time management and workload prioritisation the project could have been a much more complete solution than it currently is.

## Objective Evaluation

### Canvas API

With the implementation of a custom made private Canvas API, the project is able to successfully validate a user as a member of the university. Whether that be a tour guide or an organiser. This prevented any unauthorized requests to the backend server in order to keep costs down. Additionally to this the Canvas API returns data about the logged in user such as their full name. Being able to use a user’s full name within the application instead of their username gives the application a more user friendly appeal.

**Status: Complete**

### User Account System

The utilisation of the Canvas API and the backend Parse Server in order to create a user account system allows the project to successfully have multiple users with different rolls and abilities within the application. Multiple tour guides with multiple open day tour schedules is possible because of the account system. The user account system also allowed the project to have two types of users with the correct authentication. Because of this the application supports the creation of new tour guides, new open days and new schedules.

**Status: Complete**

### Backend Server

The backend server ties in closely with the Canvas API and the user account system.

### Timeline View

### Real Time Location Updates and Google Maps API

### Push Notifications

### Organiser Counterpart

## Further Work

# Conclusion

## Learning Outcomes

# Appendices

# References

Apteligent, 2018. *iOS Distribution and iOS Market Share.* [Online]   
Available at: https://data.apteligent.com/ios/  
[Accessed 06 May 2018].

Ashworth, M. J., 2017. *Zombies Story Timline.* [Online]   
Available at: https://play.google.com/store/apps/details?id=com.mattjamesashworth.android.zombiestimeline  
[Accessed 06 May 2018].

Consultancy.uk, 2017. *UK smartphone penetration.* [Online]   
Available at: https://www.consultancy.uk/news/14113/uk-smartphone-penetration-continues-to-rise-to-85-of-adult-population  
[Accessed 06 May 2018].

Dilger, D. E., 2017. *Apple's iOS App Store users spent $11.5 billion in Q4, 95% more than Google Play.* [Online]   
Available at: https://appleinsider.com/articles/18/01/26/apples-ios-app-store-users-spent-115-billion-in-q4-95-more-than-google-play  
[Accessed 06 May 2018].

DORMEHL, L., 2016. *Apple cuts App Store approval time to just one day.* [Online]   
Available at: https://www.cultofmac.com/428101/apple-cuts-app-store-approval-time-to-just-one-day/  
[Accessed 06 May 2018].

Google, 2013. *Android Studio Release Notes.* [Online]   
Available at: https://developer.android.com/studio/releases/  
[Accessed 06 May 2018].

Google, 2018. *Android unites the world! Use the open source Android operating system to power your device..* [Online]   
Available at: https://source.android.com/  
[Accessed 06 May 2018].

Google, 2018. *Google Play Developer Account.* [Online]   
Available at: https://support.google.com/googleplay/android-developer/answer/6112435?hl=en-GB  
[Accessed 06 May 2018].

Hollander, R., 2017. *Android fragmentation is worsening.* [Online]   
Available at: http://uk.businessinsider.com/android-fragmentation-is-worsening-2017-11  
[Accessed 06 May 2018].

Loughborough University, 2017. *Loughborogh Open Day 2017.* [Online]   
Available at: https://play.google.com/store/apps/details?id=mobilessence.loughborough.university.openday  
[Accessed 06 May 2018].

Microsoft, 2013. [Online]   
Available at: https://products.office.com/en-gb/mobile/office  
[Accessed 06 May 2018].

Microsoft, 2018. *Visual Studio.* [Online]   
Available at: https://www.visualstudio.com/  
[Accessed 07 May 2018].

PROTALINSKI, E., 2014. *Microsoft reduces Windows Dev Center fee from annual to one-time, details Benefits program for developers.* [Online]   
Available at: https://thenextweb.com/microsoft/2014/09/17/microsoft-reduces-windows-dev-center-fee-annual-one-time-details-benefits-program-developers/  
[Accessed 06 May 2018].

Protalinski, E., 2016. *Hey Microsoft, how many apps are in the Windows Store?.* [Online]   
Available at: https://venturebeat.com/2016/03/30/hey-microsoft-how-many-apps-are-in-the-windows-store/  
[Accessed 07 May 2018].

Revolution Viewing, 2017. *University of Liverpool VOD.* [Online]   
Available at: https://play.google.com/store/apps/details?id=com.revolutionviewing.uniliverpool  
[Accessed 06 May 2018].

Statista, 2017. *Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2017.* [Online]   
Available at: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/  
[Accessed 06 May 2018].

Statista, 2017. *Number of available applications in the Google Play Store from December 2009 to December 2017.* [Online]   
Available at: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/  
[Accessed 06 May 2018].

Tour Buddy Apps, 2018. *Build a better visitor experience.* [Online]   
Available at: http://www.tourappbuilder.com/  
[Accessed 06 May 2018].

Udacity, 2017. *Android Development by Google.* [Online]   
Available at: https://eu.udacity.com/course/android-developer-nanodegree-by-google--nd801  
[Accessed 06 May 2018].