University of Dublin

TRINITY COLLEGE

POInt: An Android Application for Tourists in an Unknown Area.

Anya Mangat (15314697)
B.A.(Mod.) Computer Science and Business
Final Year Project, April 2019
Supervisor: Fergal Shevlin

School of Computer Science and Statistics

O'Reilly Institute, Trinity College, Dublin 2, Ireland

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

Name_____ Date_____

Acknowledgements

Firstly, I would like to thank my supervisor Fergal Shevlin for his guidance and complete support, aid and motivation through the planning, implementation, testing and completion of this project.

I extend my heartfelt gratitude to my family and friends who have provided unconditional support to me throughout this project, and my degree in its entirety. They have helped me to meet deadlines, stay focused and finally to finish this project to the best of my ability.

Lastly, I would like to thank all the participants who took part in this project for testing purposes. Their large contribution to this project was exceptionally helpful.

Abstract

This project aims to research, design, implement and test an Android application that allows tourists to Ireland to obtain information about an unknown location, without any prior knowledge or manual input.

The overall goal of this project is to develop an application that allows tourists in an unfamiliar area to be able to retrieve information about their surroundings, without having to manually input the name of the location into a search function, or without having to recognise their desired location on a 2D map. All they are required to do is match real-life imagery to what is shown on their screens. This ability is created in the hopes of making travel around Ireland easier and more efficient. It also aspires to enable tourists to expand their travels to more covert destinations.

This application was created around the availability of Google APIs, and incorporates numerous Google Services throughout development to carry out the various intended functions. The app avails of the extensive locational information provided by these Google Services.

This application is comprehensively analysed using a business canvas model, and future aspects are examined.

Table of Contents

Abbreviations

API – Application Programming Interface

USP – Unique Selling Proposition

AR – Augmented Reality

UML – Universal Modelling Language

GDPR – General Data Protection Regulation

HTTP – HyperText Transfer Protocol

UI – User Interface

URL – Universal Resource Locator

URI – Universal Resource Identifier

FAQ – Frequently Asked Questions

SUS – System Usability Scale

USP – Unique Selling Proposition

1. Introduction

Ireland is notorious for their rural wonders and hidden gems, but how many people are brave enough to travel outside the advertised, well-documented areas to search for them?

This application caters for such visitors who are so curious that conventional tourist attractions and recommended sites are not enough to satisfy them that they have explored and discovered the country thoroughly enough for their trip. Not only will this app enable tourists to concede to their curiosity, but it will be able to feed into this curiosity too. It will encourage visitors to find and discover new places with its ability to provide the user with extra information about each location. This application serves as an extended, technical solution to the age-old problem of physical maps.

This report details the steps taken to achieve this overall goal. The author undertook research to identify the gap in the market, which fed into the implementation of the app itself. The author created the app with this research but also utilized their Computer Science and Business modules to create an application that adhered to a business canvas as well. This ensured that the app was not only technically feasible, but would additionally be a sound business idea which could be progressed in the future. The author felt using both perspectives in the construction of the app allowed it to make the best of the available time and resources, and essentially to be ready for market.

The author tested the final product to ensure that it was at the required standard and would be of use to adventurous tourists. Testing was endured throughout this project, at all stages and decisions. User testing added a unique aspect to the project and allowed the author to gain beneficial observations.

As this report was documented after the creation of the final implementation of POInt, it is written in this context. Other avenues of design and implementation are mentioned, but all aims, requirements and testing are only in regard to this final application.

1.1 Aims

The aim of this project is to develop an Android application that aids tourists in identifying places of interest without any prior knowledge of this said place. Specifically, this application aims to:

- ✓ Provide a platform that is simple and efficient to navigate with minimal instructions needed.
- ✓ Accurately open onto the user's current location for ease of use in an unknown area.
- ✓ Allow users to match images of the landmark they are trying to identify with Google Street View images with one click.
- ✓ Enable users to select a specific location from a comprehensive list provided.
- ✓ Once a location is selected, facilitate users to access additional information about it, such as address, rating, phone number etc.
- ✓ Allow the user to call the location's number from their phone dialler with one click.
- ✓ Provide direct access to the location's website, without the user having to manually open another application and search for this website themselves.
- ✓ Ensure that if no information can be obtained, users are adequately informed.
- ✓ Present a FAQ section to guide the user with any issues they may encounter.
- ✓ Enable the user to gain information on as many locations as necessary.

The author also has numerous personal aims to be fulfilled by the completion of this project such as:

- ✓ To learn all the skills necessary to build a completely functional application using Android Studio.
- ✓ Research and implement the appropriate Google APIs that best carry out the applications core functions.
- ✓ Communicate accurately with the author's supervisor when there is a problem, or when help is required.
- ✓ Utilize skills learnt from both Computer Science and Business to develop an application that takes both perspectives into account.
- ✓ Engage in useful user testing of the application, and accurately analyse the results.

1.2 Motivation

The primary motivation for this project was the author's own passion for travel around Ireland. From multiple expeditions around the country, the author has noted that some places of interest are hard to access, or do not have information readily available for visitors. For example, Loftus Hall in Co. Wexford, the most haunted house in Ireland[1], is set back significantly from the road and is closed all winter. Therefore, when tourists to the area see the magnificent house in the distance while driving past, there is no way to gain information about it without doing a manual search – but how do they even know what to search for?

From personal experience, the author identified this gap in the market and decided that creating an app that allowed the user to select the landmark based on matching up the image on their phone with what they are looking at was a satisfactory solution. An application that would also read the user's current location was an additional aid, in precaution of buildings or landscapes that had been recently altered, and would show up differently. This way, in the future when driving past the eerie looking house in the distance, the user just has to open POInt and match up the imagery to get all the exciting details.

1.3 Reader's Guide

Chapter 1:

This chapter introduces the project. It contains the aims for the whole project and the personal motivation behind it.

Chapter 2:

This chapter highlights the background research into the application idea and the gap in the market it will be fulfilling. The chapter describes research on other similar existing apps and further insights into how the use of Google Services can be utilized to reach the previously mentioned aims.

Chapter 3:

This chapter contains the functional and non- functional requirements of the application. It consists of a detailed analysis of the app itself, including the front-end, back-end and usability design aspects. The chapter includes the final design of the application.

Chapter 4:

This chapter illustrates the implementation of the application. It shows how Android Studio was used to create basic apps and then it goes on to describe how Google APIs were utilized and combined to produce the finished product. The chapter focuses on implementing functions to allow better usability for the app's target market too.

Chapter 5:

This chapter outlines the testing process of the application. It contains details on how the app was tested iteratively throughout development as well as how the final design of the app was tested functionally, and for usability by external participants.

Chapter 6:

This chapter includes a business model for the application, detailing the main aspects covered in the chosen model. The chapter focuses on the future potential of the project in each of these sections.

Chapter 7:

This chapter concludes the project. It outlines all failures, successes and any future prospects of the app. It evaluates the project and provides final closing statements.

2. Background and Research

2.1 Background

In 2018, 11.2 million people visited Ireland as tourists, 6% more than 2017[2] figures and 13% more than 2016. The 2017 figures show that while 62% of tourists stay in Dublin for their trips, 29% go South West and 22% go West[3]. This is a significant amount of tourism into these areas, namely to the most popular sites of the Blarney Stone (420,000 people in 2017[4]), The Ring of Kerry, The Cliffs of Moher (1,427,166 people in 2017[5]), The Burren and Achill Island. While the Blarney Stone and the Cliffs of Moher are thoroughly signposted to keep visitors knowledgeable, places like The Burren and The Ring of Kerry are less informative. For this reason, an application like POInt would be greatly beneficial to tourists as a 'One-Stop-Shop[1]' when travelling to all these different places.

An application would fill this market gap better than a physical tourist booklet or map due to recent smartphone penetration. A study by Deloitte 'Mobile Consumer Survey 2018: The Irish cut'[6] says that 93% of Irish people have access to a smartphone. A further study on Americans (more applicable as tourists to Ireland) claims that Americans check their phones 80 times a day while on vacation, with 10% admitting to checking their screens over 300 times each day[7]. Additionally, recent trends are showing a domination of the smartphone market by Android, who surpassed iOS devices in the middle of 2014. In 2017, Android had an 85.1% share of the worlds shipping volume of smartphones, overshadowing Apple's share of 14.8%[8]. Due to this, the author concluded that creating an Android application to carry out the necessary functions was the optimal solution, as well as the fact that a large proportion of tourists were more likely to use an application than buy a physical map.

Due to the large numbers of tourists to Ireland each year, and the current reliance on smartphones today, the author decided that creating this application would fulfil an immense gap in the market, as well as reach a considerably large audience.

1 One-Stop-Shop: an establishment where many different services or products are available

2.2 Existing Applications

After extensive research, both on the Google Play Store[9] and through numerous searches on the internet, it can be concluded that there are currently no apps of this exact type on the Irish market. Most of the applications that have been examined for similarities did provided the core functionalities of showing nearby places and / or user current location, but lacked in other functions unique to POInt. The applications detailed below are the closest existing apps to POInt currently on the market.

2.2.1 Google Street View

Downloads on Android Play Store: 2,169,982 (as on 21/03/2019)

User rating: 4.2

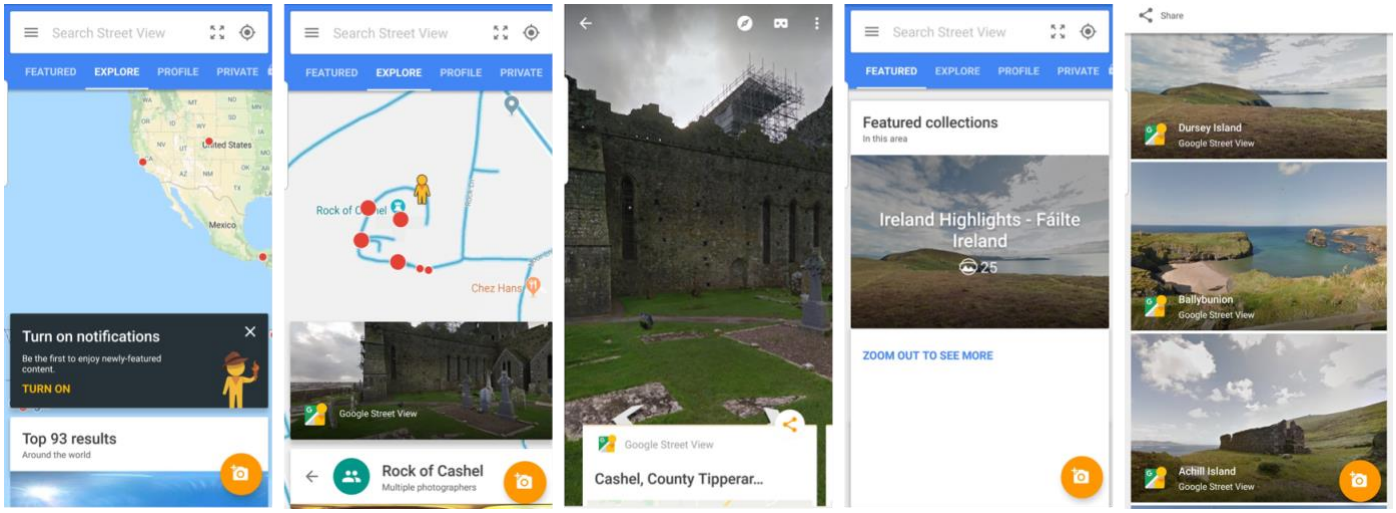


Figure 2.1 *– Google Street View Interfaces.*

The official Google Street View application is surprisingly difficult to use for a first-time user. The first noticeable difference to POInt is that the application does not automatically open on to the user's location. The user must manually select the location button to the right of the search bar as seen in Figure 2.1 to be brought to their current location.

If the user is at the Rock of Cashel for example, the app still does not open the Street View mode or provide any other information, other than the selected location and the immediate surrounding locations. These nearby locations have been added to the app by other users. The user again must manually select the Street View option to enable

this functionality. Once using the Street View mode, the usability is difficult to navigate, and no extra information is provided.

A positive difference between the Google Street View app and POInt is that the Google app goes on to recommend other similar places that might be of interest to the user based on the details of their current location. However, this addition again involves numerous extra clicks and interactions by the user.

2.2.2 Places

Downloads on Android Play Store: 100,000 + (as on 21/03/2019)

User rating: 4.1

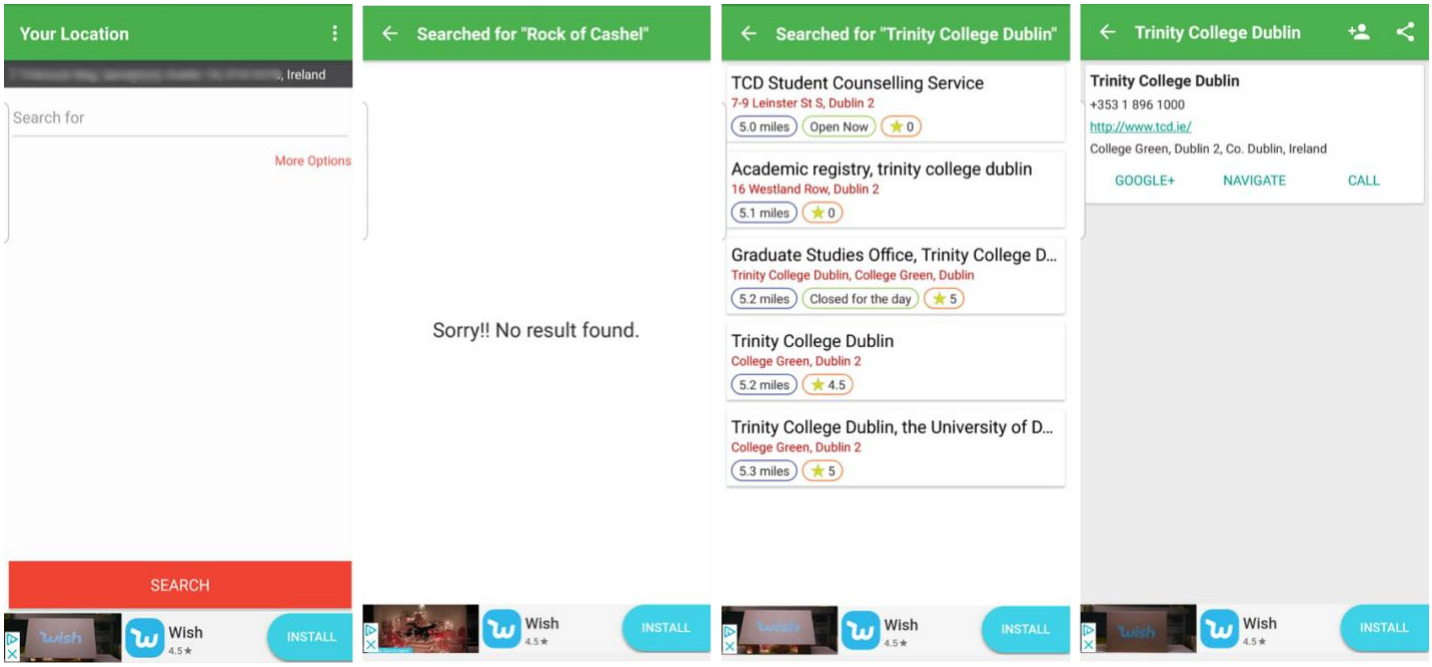


Figure 2.2 *– Places app Interfaces.*

From an aesthetic usability view, the Places app is exceptionally basic, although arguably simple and without distraction. The app automatically retrieves the user's current location when opened (as displayed blurred at the top Figure 2.2), yet does not take any action with it, such as opening a map of any sort.

The user must manually search for the landmark / building in which they are looking for more information on. This renders the app redundant if the user does not know what to search for initially. The user can enter the location displayed at the top (current

location) into the search bar manually, but as a result of numerous tests carried out by the author, this does not always bring up accurate results of the user's immediate surroundings.

As seen from the Figure 2.2, tourist destinations which are in more isolated locations, the exact target for apps like POInt, do not show up on the Places app. This again makes the app redundant in serving the immediate uses of POInt's target market. When the app does recognise a location, it is displayed without image confirmation. This does not provide enough information to the users to determine if they have selected the correct location corresponding to what they are looking at or where they are.

Like POInt, the Places app does provide extra information on the requested landmark / building, such as website or phone number. This still leaves a significantly large space in the market for an app that allows users with no knowledge of the area to find out this same information.

2.2.3 Instant Street View

Downloads on Android Play Store: 1,000,000 (as on 21/03/2019)

User rating: 4.1

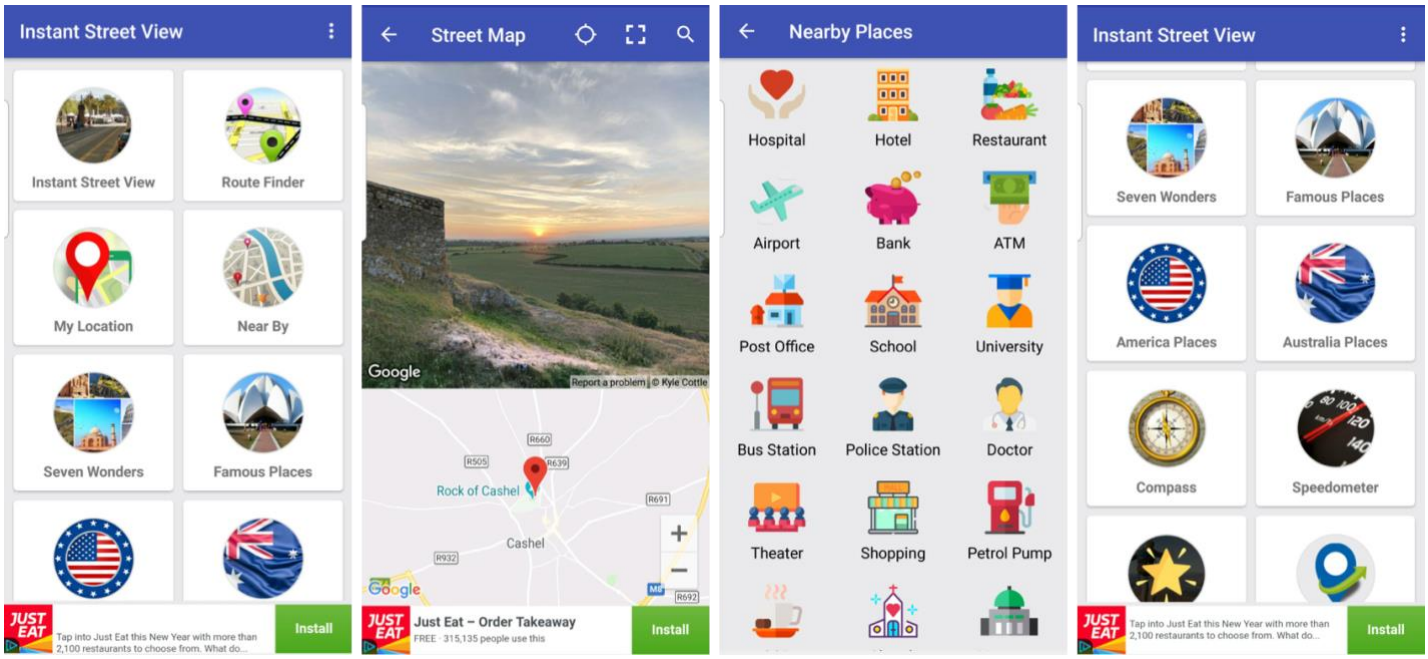


Figure 2.3 *– Instant Street View Interfaces.*

The Instant Street View app is similar to POInt in both functionality and design. When the “Instant Street View” option is selected, the application automatically opens on the user’s current location (altered here for privacy reasons). The app also provides a basic map function accompanying the Street View screen which provides the user with a better sense of where they are, as seen in Figure 2.3.

Adversely, the app does not provide any additional information about the specific location the user is currently at, apart from showing it on the map. This is somewhat combatted by the “Near By” screen which provides numerous options of types of places around the user. However, it does not contain any tourist specific types, and when a type is selected, this app simply opens the Google Maps app which then provides the functionality sought out by the user.

While this application does provide a significant amount of relevant information, it still does not solve the problem that POInt aims to, which is to provide additional information about an unknown location using Street View imagery.

2.2.4 TripBucket Ireland

Downloads on Android Play Store: 1,000+ (as on 21/03/2019)

User rating: 3.2

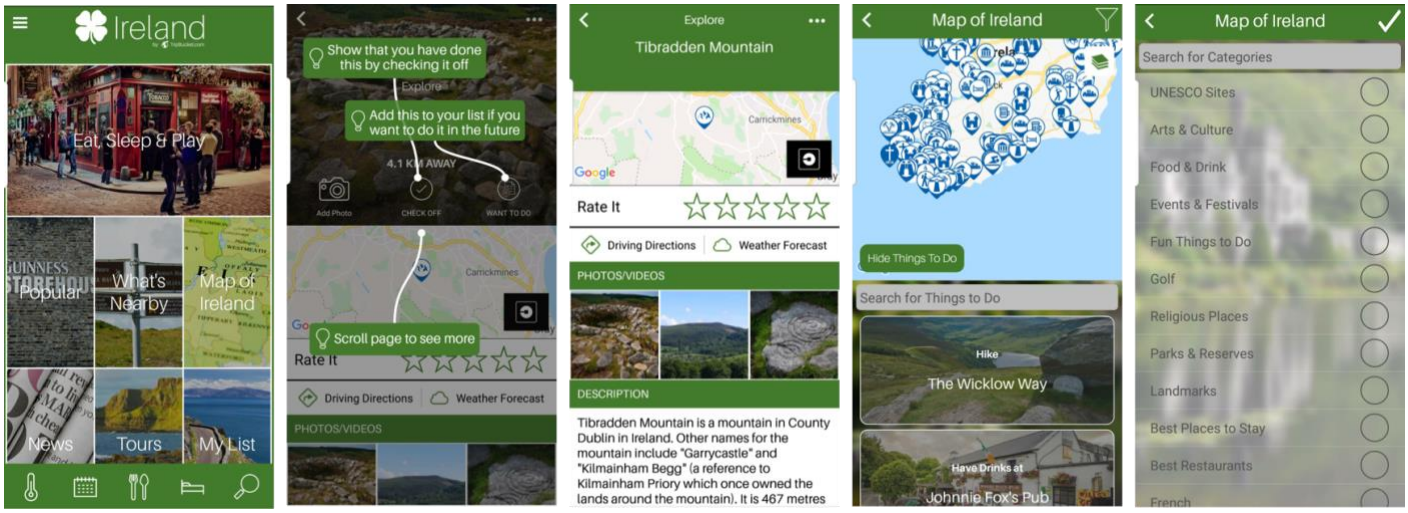


Figure 2.4– *TripBucket Ireland Interfaces.*

On the Irish market, TripBucket provides the largest competition for POInt, specifically through its focus on tourist landmarks and activities. While this app is comprehensive in the information it provides such as places to eat, places to hike and even places to drink, it does not allow for tourist use when they do not know where they are or what they are looking at.

The app does provide a 'What's Nearby' option as shown in Figure 2.4, but when the author investigated this tab further, they discovered that this took the user location and provided tourist activities within a county wide radius. It did not provide any closer, more location-specific options, and did not have any functionality for the user to discover the tourist attraction they were at, without knowing some information prior to using the application. Only after a location was selected did the app provide pictures which the user could match to their current view. This trial and error method does not provide for optimum efficiency. The app did give extra information about the site in question, but this was pulled directly from Wikipedia, an arguably unreliable source.

Lastly, this application has a 'Map of Ireland' section which allowed the user to select the type of tourist attraction they were looking for, for example Arts & Culture or Landmarks and this would be displayed on the country-wide map. While this section is directly aimed at tourists, it again does not take the user location into question, and a user with limited knowledge of Ireland would have difficulty locating the sites recommended on the app without external or further help.

2.2.5 Potential: Google Street View Augmented Reality

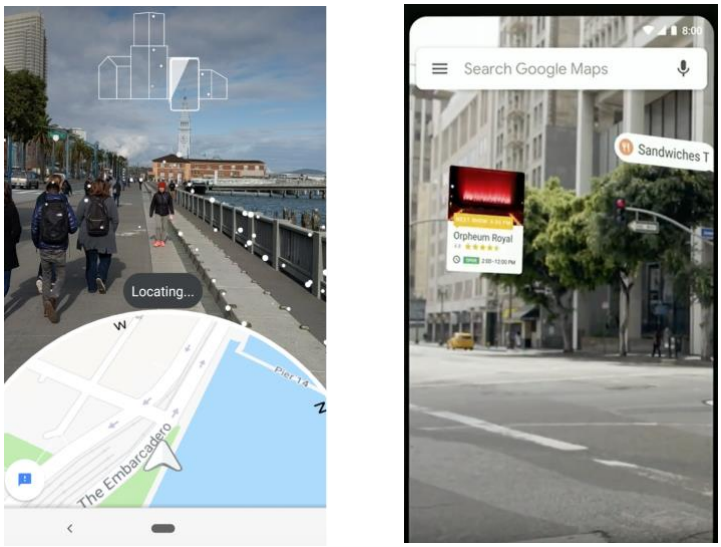


***Figure 2.5**- Google Augmented Reality Map Interfaces.*

Google are in the process of developing the addition to their Maps and Street View functionality that incorporates Augmented Reality (AR). This feature will provide an overlay to the user's phone camera that shows information on top of real world imagery. This can be seen in Figure 2.5. The idea stemmed from their Google Glass feature, and provides a more practical and interactive step from the traditional 2D Maps working alone[10]. The technology behind this idea is again comparable to that of POInt, in that the primary source of information needed is the user's location. Using this information, the image pool from the cloud that is used to analyse against images coming through the camera is narrowed down significantly[11].

The AR component will also include a 'Nearby Places' feature. This development in its entirety is a step up from POInt, and is arguably where POInt would like to be in the future. This new aspect however, is not focused on curious, adventurous tourists but rather lost tourists entirely, as Rachel Inman, the company's user-experience lead explains: "It's for those moments like, 'I'm getting off the subway, where do I go first?'"[12].

This feature has been available to a small subset of users from the beginning of February 2019[13] and while testing is expecting to be carried out with 'local guides' and a wider audience range soon, the feature is not due on the open market in the near future. This leaves the market gap open for POInt, as well as setting it up in a position as a transitional application towards this AR advanced functionality.

2.3 Google Services

The author carried out extensive research into the Google Services and APIs available to aid this application in the functions it aimed to carry out. The author created a Google Developer account to access all documentation and API's available on the Google Platform.

The author analysed the documentation accompanying the Google Maps APIs[14] to obtain a better idea of their respective functionalities and constraints. For the development of POInt, this included:

- ✓ API Best Practices
- ✓ Map Coverage Details
- ✓ Maps SDK for Android
- ✓ Maps Static API
- ✓ Street View for Mobile
- ✓ Street View Static API
- ✓ Places API
- ✓ Places SDK for Android

The author also completed further outside reading on the implementation of these Google APIs, which included articles, public forums and stack overflow answers. They watched numerous YouTube video tutorials on implementation and how the APIs worked to gauge if they would be appropriate for this project. These resources are given in Appendix D.

Once the author had refined the research to a small number of API's with potential to carry out the application's functions, they created several simple apps that embedded each API separately. This was done as a research exercise so the author had the opportunity to work on each API individually. After a few days of editing and 'playing' with the API, the author felt they had a comprehensive knowledge base of how it worked, how it was implemented and what other functionalities it contained or could aid. A detailed outline of the applications created for API practice is provided for in Section 3.1 from a 'Design Changes' aspect and Section 4.2 as an implementation research practice.

3. Design and Requirements

The initial brief for this project was broad and entirely open to interpretation. This worked to the authors advantage, as it gave them the chance not only to develop an application that they were passionate about, but also to develop new skills at their own pace, which when perfected, may prove to be useful in future projects.

The outline was to build an Android application with some form of image analysis. The author interpreted this in the form of Google Imagery, rather than user-taken content from their camera. Although the design for this project did touch on user imagery, Google Street View was always intended to be the core functionality of this project, whether as a background tool to compare images to, or as the interface for the app itself.

The design for this project was changed numerous times to enable the main functionalities to be carried out at optimal efficiency. As the design requirements and aims of this project were so expansive, numerous ideas and directions were explored before the final design of the application was decided upon. This was beneficial to the author in expanding their knowledge and usage of Android Studio and Google APIs in general.

This chapter outlines the design options that lead to the creation of the final design. This final project idea is the root of the subsequent functional and non-functional requirements that the author adhered to for the completion of this project. The design was deconstructed into front-end and back-end compositions. The completed final design is analysed at the conclusion of this Chapter.

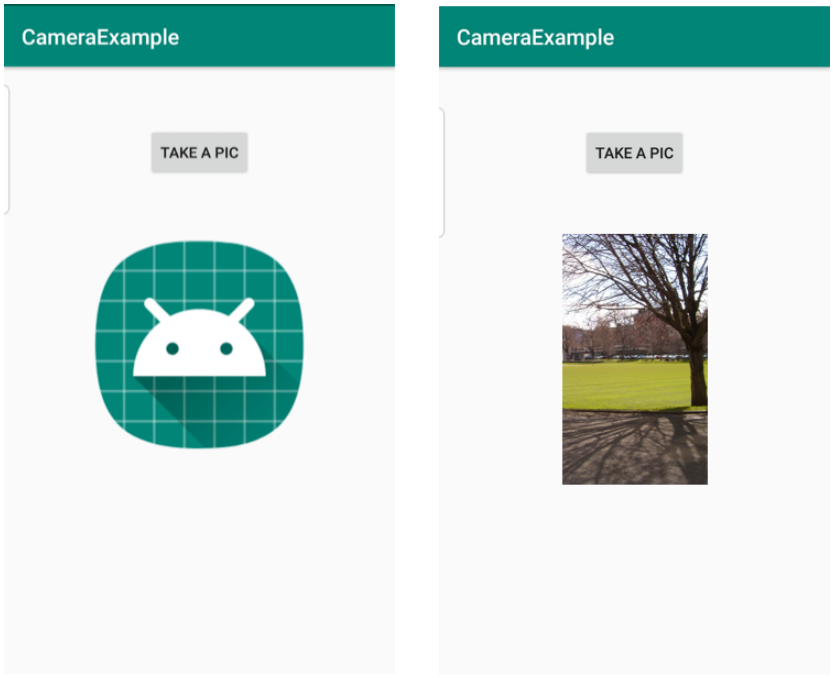
3.1 Design Changes

3.1.1 Take / Select Photo and Receive Landmark Information

The author's first idea to fulfil the aims of the project was to create an app where the user could take a picture, or select one from their phone gallery, which would be converted into a Bitmap[2] through the application. This would be displayed in the app's interface. In the background, Google Services would access the user location, and then compare the pixels of the Bitmap image to that of Google Street View images from around the determined location. When a match had been made, the app would return to the user the name of the place they had pictured, along with other generic information that may be required, such as address, phone number and user rating.

To assess the feasibility of an application of this kind, the author started by creating numerous applications that incorporated the taking or selecting of pictures and storing them as Bitmap images. These apps can be seen below:

Example 3.1 – Preliminary Camera App.

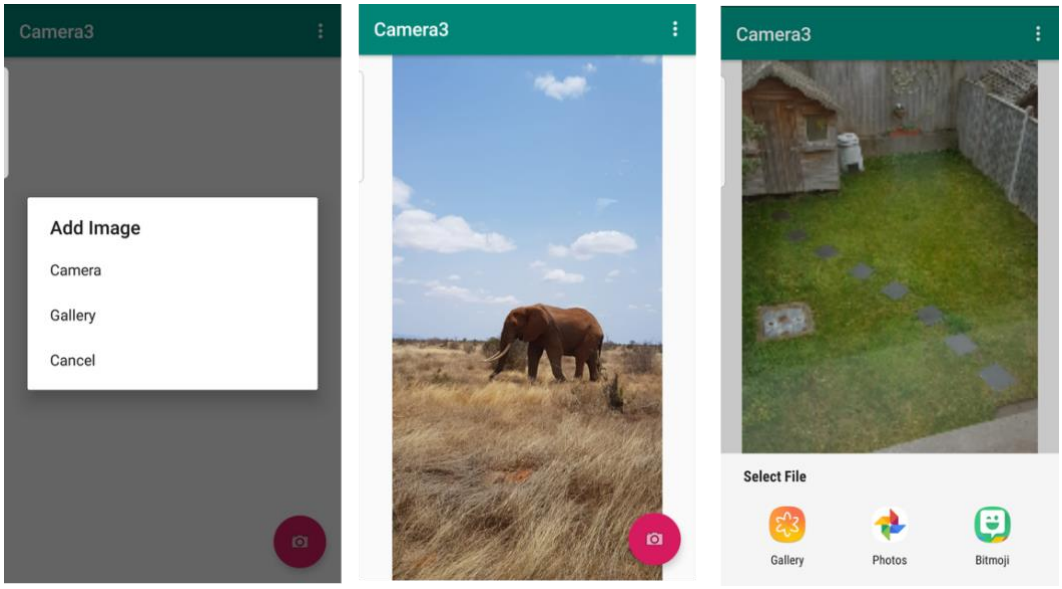


Example 3.1 demonstrates a simple application that requests access to the user's camera app to take a picture, and then save that in the application itself as a Bitmap.

[2]Bitmap images are stored as a serious of small pixels that are assigned a colour and then arranged into a pattern.

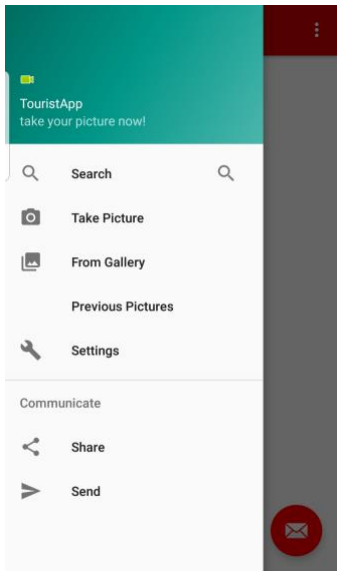
This application was the first stage in the author's build-up to analysing the pixels in this Bitmap and comparing them to Google Street View images.

Example 3.2 –Camera App with Camera or Gallery Function.



Example 3.2 was a logical step up, in that it allowed the user to select an image from their gallery as well as accessing their camera to take a current picture. The second picture shown is a picture from gallery saved as a Bitmap, and the third picture is the camera picture saved as a Bitmap. This again was used as practice for the author to build up to relating the image back to Google Services.

Example 3.3 – Menu Options.



Example 3.3 showcases that the author created numerous apps which carried out the same basic functions to familiarise themselves with the programming abilities required for this project. This experimentation method enabled them to choose the best options, whether layout styles, buttons or textViews or size of images, when moving forward. This example was only used to set up and manipulate a side-bar menu that would initiate the same actions from the previous example.

Reasons for change:

The author decided against this idea to carry out the project mainly due to the conceptual difficulty of comparing 2-Dimensional pictures to 3-Dimensional objects. For example, if the user takes a picture from a certain location, at a certain angle, at a certain hour and stores it as a Bitmap, comparing it to a rounded, pre-taken 360° Google Street View image would be difficult. Due to the vast number of different angles images can be taken from (both on a horizontal and vertical axis), there is too large of an image pool that would have to be created to account for each circumstance. Figure 3.1 shows that from one exact phone position there are a magnitude of different pictures that can be produced. Figure 3.2 analyses this further and provides a clear overview of how numerous pictures, taken from similar locations can vary greatly. All four pictures show the tower centred in the frame, but due to the distance, we can see that Exhibit 3.2.2 contains a tree in the foreground that will be further analysed, unlike in the other 3 pictures. Figure 3.2.2 also seems to be taken in the winter as the bush beside the tower has no leaves. Each figure shows the white building to the left of the tower as slightly different heights, widths and colours. Whether from a low angle or straight on view, all the images are slightly different but would require an immense amount of extra analysis. Thus, it was concluded that processing this amount of information would be too time and power consuming.

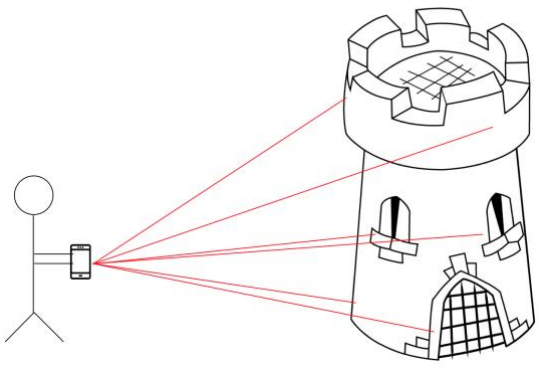


Figure 3.1 *– Potential Photos from the Same Position.*

Figure 3.2 *– Cloyne Tower, Co. Cork, from Various Angles.*

Exhibit 3.2.1 ***Exhibit 3.2.2*** ***Exhibit 3.2.3*** ***Exhibit 3.2.4***

Similarly, there were a few other smaller difficulties that too were almost impossible to overcome. One such example is the light hitting a specific building in a different way, creating shadows or illuminating areas that will not be picked up when a comparison of pixels is made.

Another fundamental problem would be related to images taken from the user's Gallery. If the image chosen did not contain a location stamp, the application would have to access and compare all images on the Google cloud. This would have been an exceptionally high power task, and would more than likely cause the application to crash before the comparison was completed.

The number of fundamental flaws with this idea rendered it unfeasible quite shortly after it was initiated. The author made no more progress on picture taking applications, and instead began to devise a new plan to complete this project.

3.1.2 Static Street View Image

The second idea that the author had to fulfil the aims of this project was to access the user location first, and then use this to open a static Google Street View image of the location. The user would then click on the area of the 360° panoramic image to indicate specifically the landmark they were looking for information on. From this, the app and the subsequent Google Services running in the background would be able to assess

what direction and what angle the specified landmark was at. Once the landmark was pinpointed, related information would be requested from the cloud which would be fed back to the user through an information interface.

The author carried out considerable research into accessing static Google Street View images, and discovered that this was only possible through a HTTP request[15]. To produce an image of the user's surroundings, a unique, encoded HTTP URL was created. The construction of this is shown in Figure 3.3.

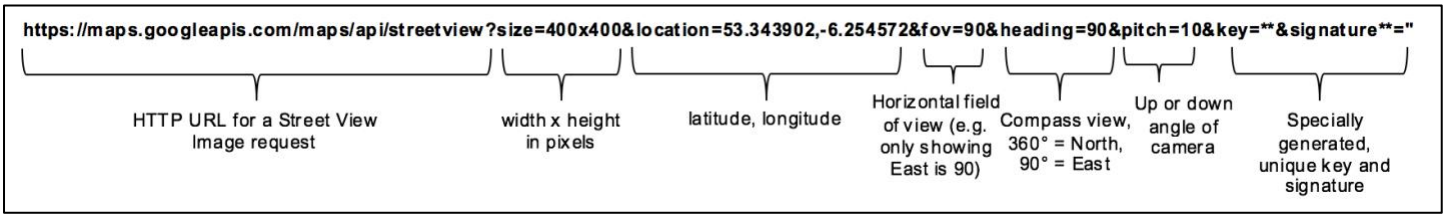
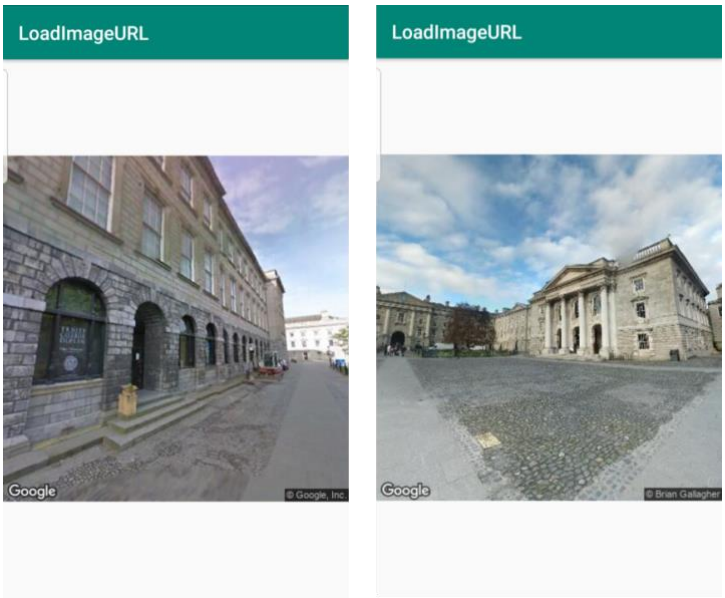


Figure 3.3 *– Deconstructed HTTP URL.*

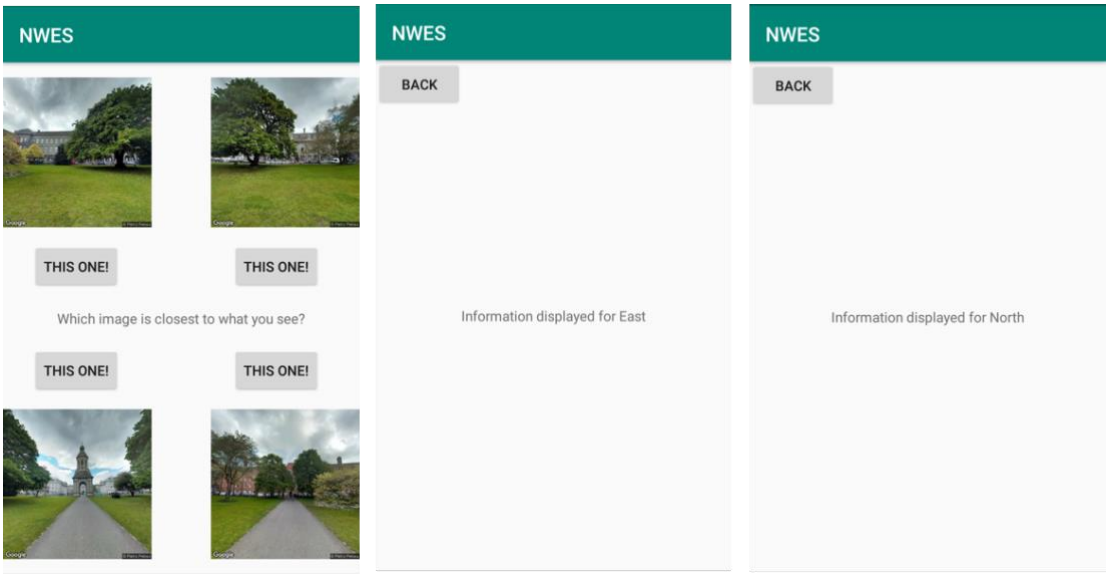
Example 3.4 – Static Street View Example.



Example 3.4 demonstrates an application created to manually take in the URL and display it as an image. Because the request automatically displays a picture, it was directly transferable into the application in the appropriate form. The picture displayed was static and the user could not interact with it. For this reason, the author would have had to split the image into detailed segments based on various factors such as; compass direction, camera angle, field of view and more. This information, coupled with variables extracted from the URL would be further analysed to zoom in on a segment containing the user selected landmark.

To assess feasibility, the author decided to first divide the image into four simple segments; North, South, East and West. This can be seen in Example 3.5 below.

Example 3.5 – Static Street View Image from Varying Compass Points.



The application shown in Example 3.5 allowed the user to pick the general direction that the exact landmark they were seeking information on was in. This was done by the author inputting four different URLS, each with the same location, but a different compass view to be displayed. Whilst this did ease the problem of determining what the author was pointing at by reducing the possibilities in each image, other larger problems evolved.

Reasons for change:

The author decided against this idea mainly due to the technical difficulties surrounding the dynamic URL. While Google did provide documentation on how to create a unique encoded signature within a program, the author had significant difficulty implementing it[16]. This was due in part to the private and unique signature and key that needed to be generated each time the user moved and a new location was detected.

Another reason why this idea was not followed through to completion was because of the usability. The application did not satisfy enough front-end requirements, and

overall the layout was too simplistic for what the author intended the app to be able to do. The application was ‘chunky’ and ‘rough-and-ready’ and did not fulfil the authors expectations for the outcome.

Furthermore, analysing a specific point on a picture to determine exact compass direction (e.g. North West), angle and height would have been an inefficient way to determine what landmark a user was looking to identify. There would not be enough information to make an accurate assumption about this, and so more user input and interaction would be required. The author could not determine an adequate way of obtaining this information from a user that was in an unknown location. This again hindered the usability, and so the author decided to stop implementation on this idea too.

3.1.3 Street View and Places (Final Design)

The final design of POInt came after many attempts and other unfeasible ideas. The final design was a direct by-product of the failure of these other ideas, or more accurately, it was a consolidation of all the solutions to those previous failures.

The final design stemmed from the high usability aspect Street View brought as an interface to an application. The previous apps constructed by the author did not have a ‘flow’ and came across as layers of code rather than a seamlessly integrated application. Street View as a centrepiece would allow this flow to be obtained and passed on to the user’s experience. Street View, as mentioned previously, allowed users to match real-life imagery to their screen. The author resolved that this matching had to be done dynamically, so the user could control where the camera was pointing at all times. Apart from allowing the user to feel more in control, this made it easier to pinpoint what they were looking at with less calculations and variables to consider.

As this is the final design for the application, a detailed analysis and breakdown is provided in the rest of the Sections of this Chapter.

3.2 Functional Requirements

The functional requirements for this project outline what the completed application should be capable of. This includes both front-end and back-end requirements. The requirements were devised by analysing the aims and motivation of this project, as well as utilising the functions that are readily available through various Google APIs.

The functional requirements of this project are:

- ✓ Display a Splash Screen to inform users how to use the application when it opens. Allow users to skip this if they deem necessary.
- ✓ Accurately detect the user's current location.
- ✓ Use the retrieved location to open a Google Street View interface.
- ✓ Allow users to navigate around their current location, in a manner like that of Google Street View, to find the landmark they are looking to identify.
- ✓ When a landmark is selected (clicked on) open the Google Places interface.
- ✓ Enable users to be able to choose the landmark they require efficiently and easily from the list provided by the Google Places API.
- ✓ Save the selected location and retrieve the additional data fields associated with it.
- ✓ Store this data and transport it to an information page.
- ✓ Display this information in an aesthetic way.
- ✓ Allow users to place a call to the number provided by the location with one click.
- ✓ Allow users to access the website of their chosen location by a direct link on the information page.
- ✓ Allow users to go back to the homepage to complete a new search.
- ✓ Enable users to easily accesses a help page.
- ✓ Incorporate a help page that provides a comprehensive overview of questions that are most commonly asked.
- ✓ Allow users to interact with the questions to receive their desired answers.

3.3 Non- functional Requirements

The non-functional requirements of this project are related to system quality. They include all additional aspects of the application. These requirements serve as constraints or restrictions of the interactions with the system. They also ensure that software reaches the usability level required by users, testers and even some regulation[17].

The following are the general non-functional requirements:

- ✓ **Reliability -** the application should perform as expected and contain minimal errors. It should be relied on to carry out the functions it is intended to seamlessly.
- ✓ **Usability -** the application should be easy to use, with clear instructions and navigation mechanisms. This should be achieved through a well-structured interface, logical flows and clear labels.
- ✓ **Testability –** the application should be easy to test throughout development by the author but also by external resources. The testing should provide clear outcomes.
- ✓ **Efficiency-** the application should be easy for users to learn and use efficiently. There should be no unnecessary features that slow down the applications' main functionalities.
- ✓ **Maintainability -** the application and the accompanying documentation should be clear and easy to follow. This will allow it to be easily maintained in the future and expanded on if needed.

The non-functional requirements specific to this project are:

- ✓ Aid users in an unknown location to find the landmark they are looking at and trying to identify through easy real-life matching facilitated by Street View.
- ✓ Allow users to minimise search time by selecting the location they want to identify from a small range of options.
- ✓ Offer users more information on the landmark they have selected than just the name. Display this information in a clear, easy to understand way.
- ✓ Allow users to access information on multiple landmarks of their choice by the simple use of the back button.

3.4 System Design

The author resolved that a UML Use Case Diagram would best describe the tasks of POInt. A Use Case Diagram outlines how the user interacts with external entities and, in this case, the system itself. The diagram tracks the requirements of a system and maps how they translate to tasks and design. The Use Case diagram in Figure 3.4 demonstrates the system design of POInt.

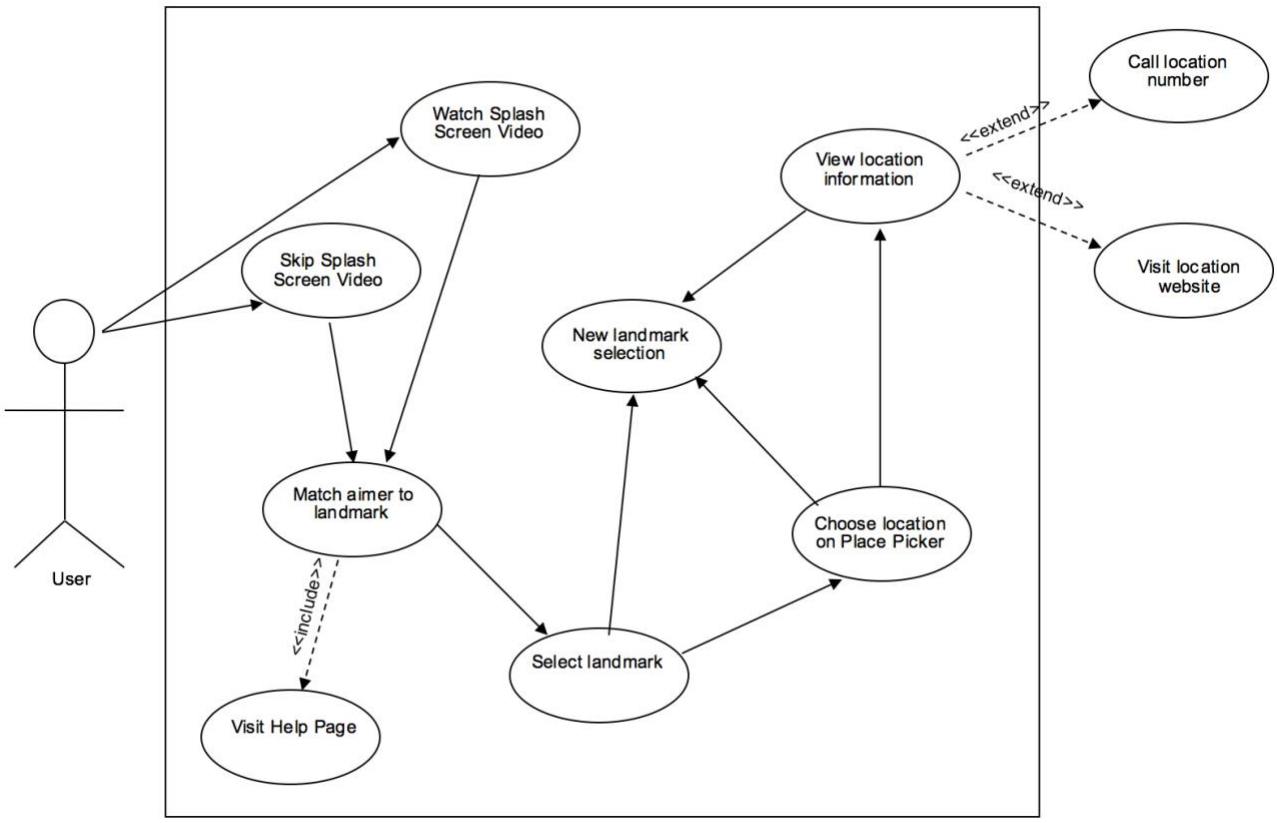


Figure 3.4 *– POInt UML Use Case Diagram.*

Figure 3.4 can be elaborated upon to illustrate the interactions between the actor (user) and the system:

1. The user opens the application and is presented with a splash screen video.
2. The user can watch this video or choose to skip it. Either way, the next page will open on a Google Street View interface, with an aimer in the centre.
3. The user matches the aimer to the landmark they are seeking more information on.
4. The user selects this landmark by clicking on the aimer.
    i. From the Street View interface, the user can also access the Help page.

5. This will launch the Google Place Picker, which will present the user with a list of places (business names mostly) within the area they selected with the aimer. The user chooses the location they desire to retrieve more information on.
6. This will open the information page containing additional details on the selected location.
    i. The user can select to call the given number from their phone dialler, and / or visit the location website in their default browser. For this reason, these two use cases are not directly inside the system (the application).

At numerous stages of interaction with the system the user can choose to make a new landmark selection. If they have clicked on the aimer to select a landmark, they may press the back button on their phone, and can immediately make a new selection. When they are in the Place Picker API and have selected a location from the list, they can select “Change Location” and again make a new selection. When the user has viewed and interacted with the information presented about their previous location selection, they may again press the back button to make a new one.

3.5 Back-End

The back-end of POInt is almost exclusively comprised of Google Services. This is due to the vast amount of information Google collects and allows access to through their APIs. The two main Google Services that are utilized by POInt are Google Street View and Nearby Places.

The initial, and arguably most crucial back-end process is however, accessing the user's current location. This is a design feature implemented to ensure ease of use for the user. If a tourist is lost, and the application opens on a random, predefined location, it would be no use to them as they would not know what to search for.

The user's location is fed into Street View, so that the application opens on an interface that has as many similarities as possible to the user's current real world view. Street View was chosen over maps for this precise reason. Through Street View, change listeners are implemented to determine if the user is moving, or has selected a location some distance from their current location, through navigation. This back-end design was utilized to keep track of the user's movements, and the new potential locations around them from these movements. This feature was also used as a testing mechanism, as outlined in Section 5.1.

Once the user selects their desired location, the Nearby Places API takes over. This provides a list of landmarks that are within the area the user selected. When the user chooses the appropriate location from the list, and confirms their choice, this API stores the location information in a 'Place' Object. The location details can be accessed by extracting the data from this Object. This was chosen as a design feature as the API itself facilitated the storage and extraction of the required data.

The stored information is then transferred into a new page and displayed for ease of interaction for the user. The final crucial back-end designs are the applications ability to open the user's phone dialler with the number of the location already input, and to open the user's default web browser directly onto the locations website. This design choice was made by the author to increase usability standards. The user does not have to manually input the phone number or website into these external apps. This

reduces the possibilities of errors when manual information is transferred, and allows the user to gain the extra details faster and more aptly.

These back-end designs were enforced by the author due to their user experience enhancing abilities. The features mentioned allow the application to fulfil the functional and non-functional requirements efficiently for the user. The back-end processes ensure that all the information presented is correct and errors are minimal. Figure 3.5 demonstrates the interactions of the back-end services within the system as a whole.

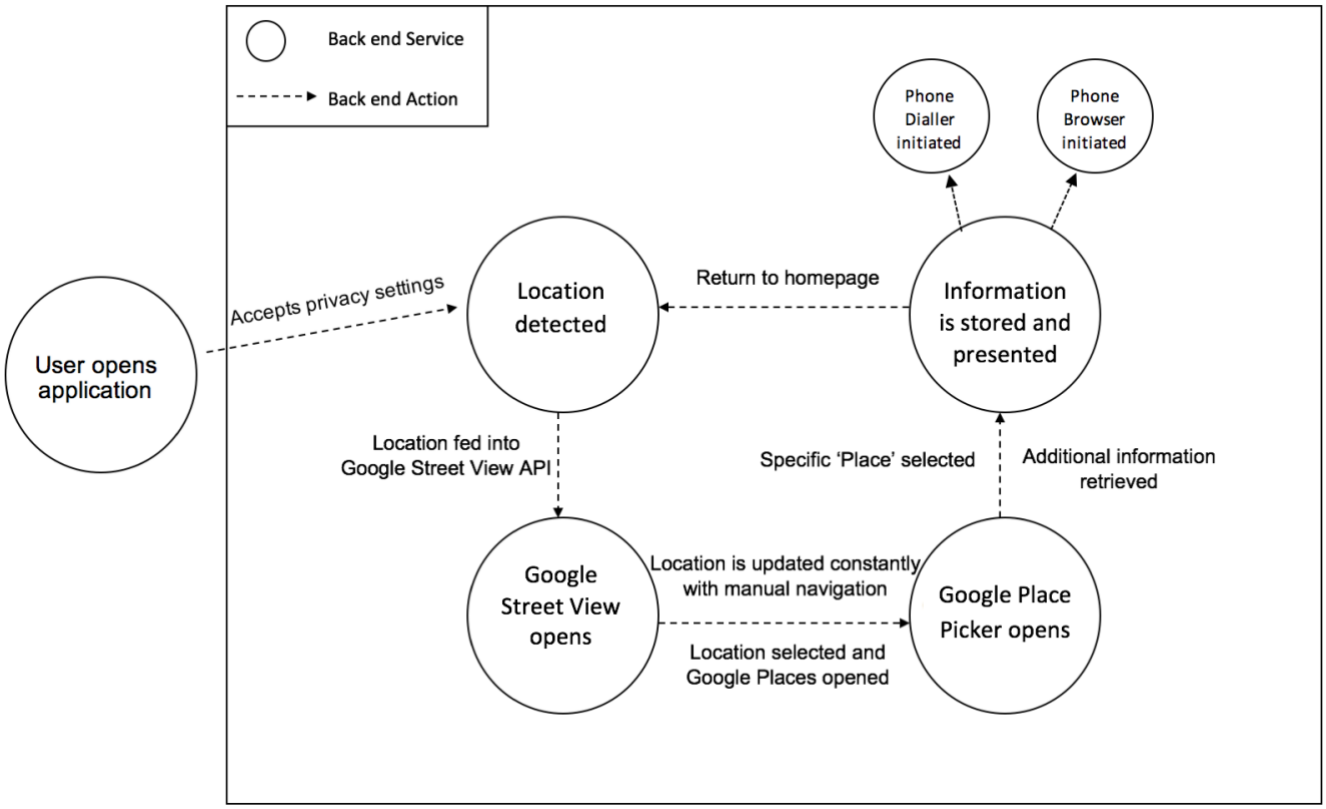


Figure 3.5 *– POInt back-end deconstruction.*

3.6 Front-End

Following comprehensive research, the author decided that the best way to ensure optimal front-end design was to follow two sets of principles, using one as a definitive guideline, and the other as a finishing checklist. Both guidelines have similar and overlapping rules, allowing for all aspects of front-end design to be incorporated.

3.6.1 Nielsen and Molich's 10 User Interface Design Guidelines[18]

Principle	Principle Description	POInt Application
Visibility of system status	Users should always be informed of system operations. Easy to understand visual displays, reasonable time given.	The aimer provides a self-explanatory visual display of a required action (and is also demonstrated in the splash screen). System updates are given as to what location has been chosen. As POInt consists mainly of two pages navigated bi-laterally, vigorous system updates are not necessary.
Match between system and real world	Mirror language and concepts users find in the real world. Present information in a logical order and piggyback on user's real-life expectations.	Google Street View was chosen as it provides the closest link between system and real world. The icons beside the locations in the list mimic common real-world depictions. Selecting a location and getting back information is a logical flow.
User control and freedom	Offer users a digital space where backward steps are possible. Can undo previous actions.	Selecting a new location is possible at any stage in the process. Backwards steps are available throughout.
Consistency and standards	Graphic elements and terminology maintained across similar platforms.	This is implemented through a continuous theme of colour and styles. Icons to represent images ensure global understanding.
Error prevention	Design system so potential errors are kept to a minimum. Users do not want to detect / fix problems.	Google APIs are used as they limit issues. Permissions are requested at the app launch to eliminate null errors.
Recognition rather than recall	Task-relevant information within the display. Consider limitations in memory to enforce recognition rather than the recall of information.	The splash screen highlights the aimer from the outset. Recognition of common icons like help, phone number and website make the app easily navigable. Information is always given in the same coloured text.
Flexibility and efficiency of use	Less interaction to allow faster navigation. Using function keys, hidden commands, etc. Users should	The aimer provides this, as all the user must do is point to a location, and select the name.

	be able to tailor the interface to what they use most frequently.	The rest of the details and information is provided to them automatically.
Aesthetic and minimalist design	Keep clutter to a minimum. Display must be only necessary components; unnecessary information competes for user's attention.	The main screen is devoid of any distracting buttons, as is the information page. No items are placed on any screen that are not of use.
Help users recognise, diagnose and recover from errors	Assume users are unable to understand technical terminology, messages should be displayed in plain language.	The FAQ help page provides answers to common issues or questions in non-technical terms. If the user selects a location that does not have information stored on it, they are informed of this, again in layman's terms.
Help and documentation	Users should be able to navigate the system without documentation. If it is needed, documentation should be easy to follow.	This report can be submitted in some part as a user manual.

3.6.2 Bokardo – Principles of User Interface Design[19]

This table was used by the author as a final checklist of front-end requirements.

Principle Outline	POInt?	Principle Outline	POInt?
Clarity: recognise what the app is trying to do, understand interactions	✓	Strong visual hierarchies: clear viewing order, first thing seen	✓
Interfaces exist to enable interactions: real-world comparability	✓	Smart organization reduces cognitive load: show relationships	X
Conserve attention: no unnecessary materials for distractions	✓	Highlight with colour: guides attention, avoid vibrant colours	✓
Keep users in control: no surprises or confusing pathways	✓	Progressive disclosure: show only what is necessary on the screen	✓
Direct manipulation: directly interact with objects, small footprint interface	✓	Help people inline: built in help	✓
One primary action per screen	✓	Zero state: crucial first interaction with the interface, provide guidance	✓
Keep secondary activities secondary: smaller & lighter	✓	Great design is invisible: users can focus on goals not interface issues	✓
Natural next step: anticipate next interaction and support it	✓	Build on other disciplines: typing, visuals, human factors	X
Appearance follows behaviour: if it looks like a button, should act like it	✓	Interfaces exist to be used: only useful if usable and used	✓
Consistency matters: group like items, show appropriate differences	✓		

3.7 Completed Final Design

3.7.1 Interface Analysis

The complete set of screenshots for the final design of POInt can be viewed in Appendix A. Figure 3.6, 3.7 and 3.8 are used to display the final design choices.

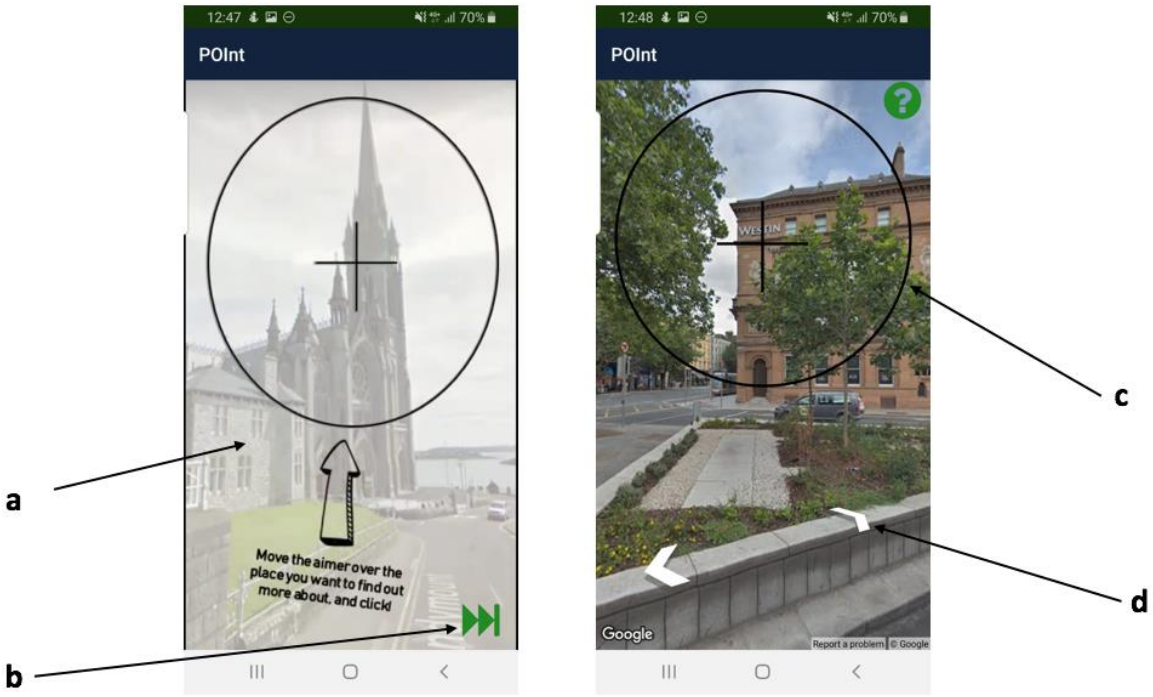


Figure 3.6 *– Splash Screen and Homepage.*

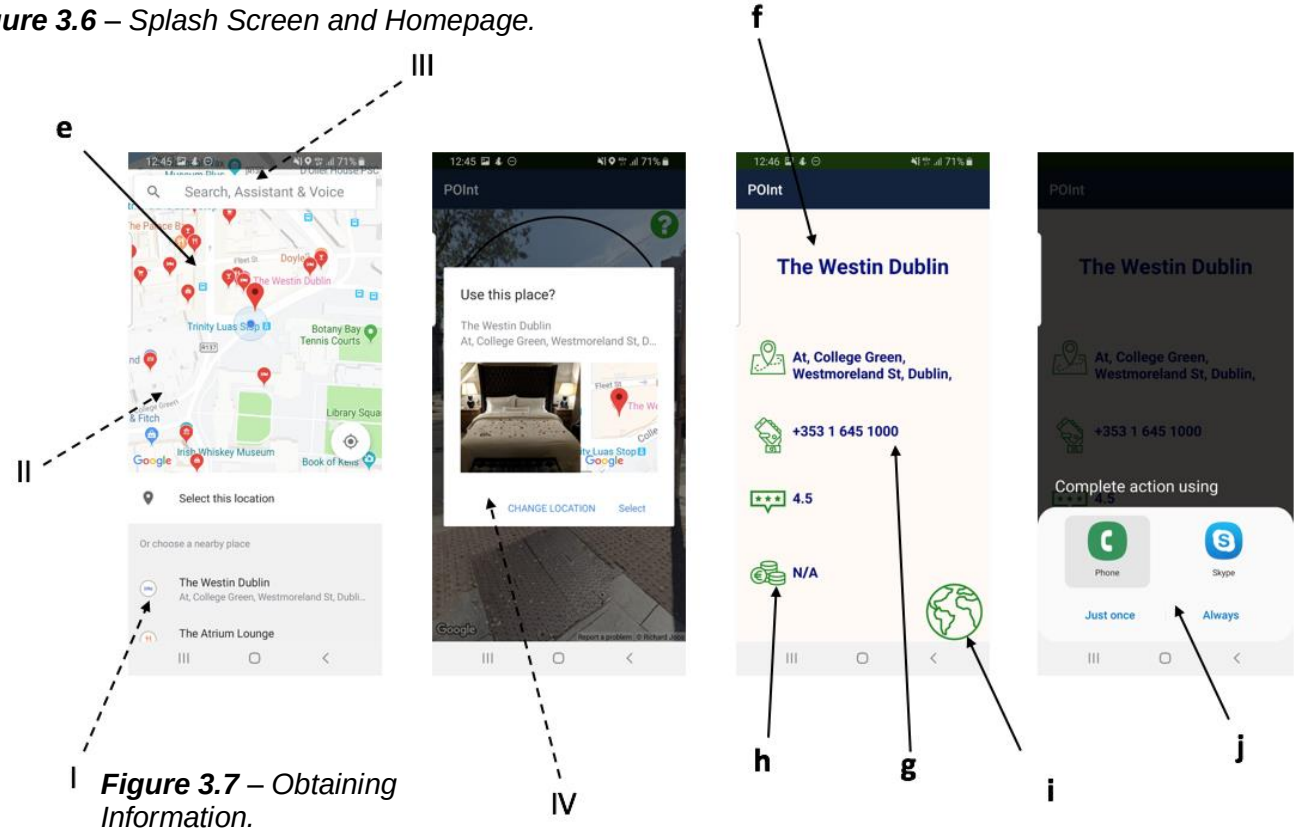


Figure 3.7 *– Obtaining Information.*

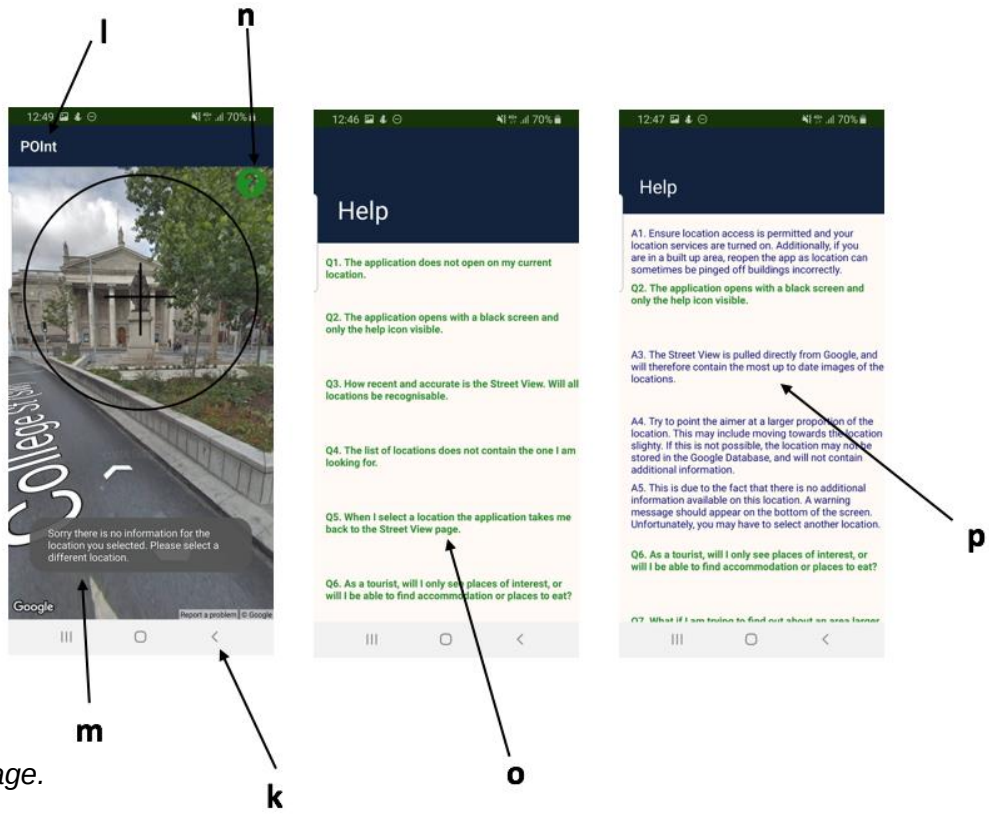


Figure 3.8 *- Help Page.*

a) **Splash Screen Video -** This video was created as a short demonstration of how to use the application. The video highlights the aimer multiple times, and instructs the user to click on this to receive information. It contains minimal icons and text so as not to cause confusion. A video animation is used instead of textual instructions to entice the user's attention. This video is 10 seconds long to not be boring and to not hinder usage.

b) **Skip Video Button –** The author added a button that will skip the video and open the Home Page when clicked on. This button is a design feature for repeat users of POInt who already know how to use the application. It is the same green as the other icons and text inside the application. It can easily be seen from the splash screen page when needed.

c) **Crosshair Aimer -** After numerous different design experiments, as can be seen in Figure 3.9, the author decided the design in Figure 3.6 was best suited to the application. The final design has a clear outline of the area in which the landmark should be placed in. The outline is not too thin, like in Exhibit 3.9.1, so it can be easily seen regardless of the background colours. Nor is it too thick, like Exhibit

3.9.2, where it takes away focus from the background and becomes an eyesore. The small cross in the centre of the aimer allows users to more easily compare it to a gun crosshair and therefore act appropriately. This was also altered numerous times, and varied between creations by the author and images found through icon providing websites (Appendix D). The absence of a centre piece could have caused confusion as evidenced in Exhibit 3.9.3, but too large and thick of a centrepiece again takes away from the landmark in the background itself, as in Exhibit 3.9.4.

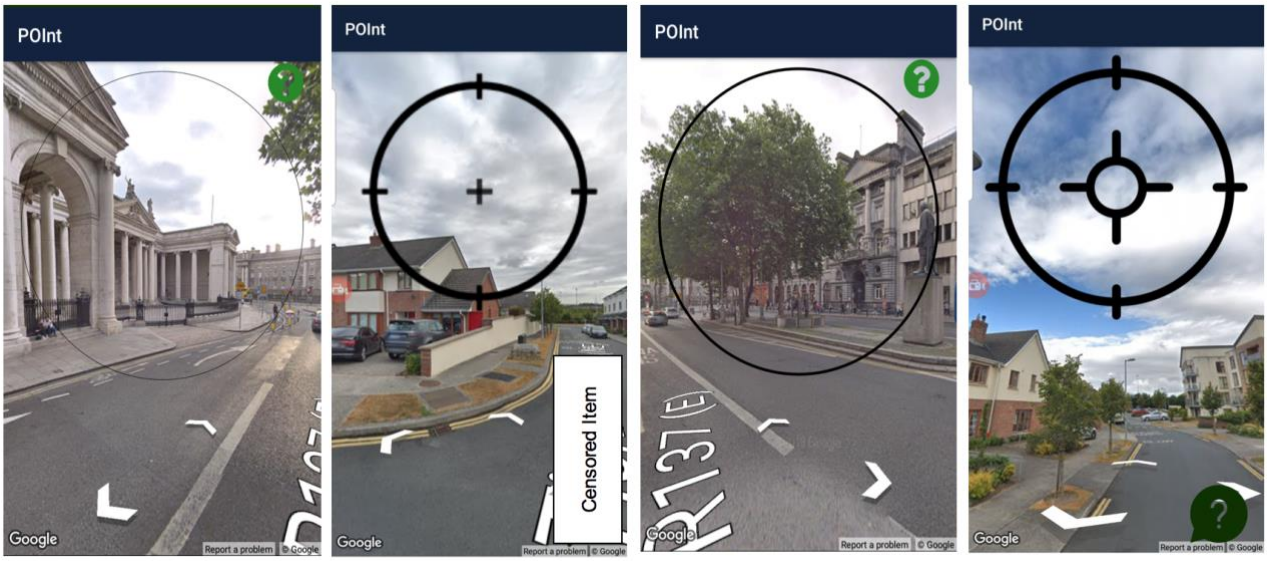


Figure 3.9 *– Aimer Research Choices.*

Exhibit 3.9.1 ***Exhibit 3.9.2*** ***Exhibit 3.9.3*** ***Exhibit 3.9.4***

d) Google Street View Navigation – The decision to keep the Street View navigation arrows and place names has remained consistent throughout the design process. The author felt that it was important for the user to immediately recognise the similarity to Google Street View, in order from them to be able to mimic its usability. The arrows ensure that users do not have to work out a new way of navigating the area to find what they are looking for. It also shows the user that when the application opens it is not a static picture, but a dynamic screen that can be manipulated using these arrows.

e) Google Nearby Places – Place Picker is a comprehensive API that is almost exclusively un-customisable. For this reason, it was integrated as a secondary action in POInt (after selecting the location through Street View).

I. **Place Type Icons-** Internationally recognised icons are beside location names to inform the user of the type of each location. Icons such as a tea cup to indicate a café allow users to narrow down their search for the location they require.

II. **Map of Surrounding Area -** For a user unfamiliar with the surrounding area, this map remains mainly an aesthetic function. It does however provide a wider image of the area, in case the user wants to search for another nearby location. If a user is looking for information on a Museum through POInt, it may be useful that they can see there is a restaurant next door, using this map.

III. **Search Bar-** The search bar allows users to search for a location and obtain information on a place that they may not be in view of. An example of this would be; a tourist in their hotel in Cork looking for information on the Cork Gaol that they plan to visit later, and want call to make bookings for.

IV. **Confirmation Dialog Box-** the last un-customisable feature of this API allows users to confirm that what they have selected is what they meant to gain information on. If a user was trying to find out about a Cathedral, but by mistake selected the park beside it, this would allow them to see their selection, with a defining image obvious to the type of location it is, and enable them to change it. Once this dialog box is confirmed, the information page opens automatically.

f) Name of Location Selected - This section of writing is at the top of the information page as this is where the user's eye will glance at first[20]. The significance of the name being shown first is to ensure that the user knows exactly what location they selected. The colour, size and font of this variable is consistent with the overall theme and has a definite but gentle contrast to the background shade.

g) Informational Text – The additional information displays are the same colour and font as the name of the location selected. They are slightly smaller, again adhering to the visual hierarchy.

h) **Informational Icons -** The icons shown were selected from 'Flat Icon' and their colours were matched to the overall green theme colour of the application. These icons were chosen as simple but easy depictions of the information they are coupled with. This is to assure that the information presented was globally recognised and understood.

i) **Website Icon –** This globe icon is the same colour as the icons mentioned above. The icon is not centred on the screen like the other informational icons, but placed in the bottom right-hand corner. This is because this icon should be seen after the other icons, as the user may have already found the information they were seeking (phone number or price range) and would have no need to access the website to manually retrieve this same information. The icon is still large enough to act as a secondary function for the page if the user does want to find out more.

j) **Phone Call Capabilities-** POInt allows users to directly call the number given of a location by clicking on it. This will automatically copy it into the user's dialler and begin a call. If an Android phone has multiple phone call applications on it, the user will be given an option of how the call will be placed.

k) **Back Button –** There is no inbuilt back button on the application as the author concluded from research carried out above (Front-end Design, Section 3.6) that numerous unnecessary buttons only seek to confuse users. The application is designed for Android systems, which always contain a back button on the screen or console. This makes sure that the user has the option to undo decisions they have made, without the distraction of an additional button, at any time.

l) **Name of the App -** The author resolved to always keep the name of the application visible at the top. This is mainly to keep within the theme of the application. It is also to inform users when they choose to be navigated to the selected location's website that this has been opened in their default browser and they will need to re-enter POInt to identify another landmark.

m) **Error Message-** The message to inform user's that no information is available in the Google databases on their chosen location is given as a Toast message[21]. This

message pops up at the bottom of the screen, grabbing the user's attention. It disappears after a few seconds to allow the user to continue using the app without distraction, once they have been informed and updated.

n) **Help Button -** This button has been integrated into the app using Visual Hierarchy Mechanisms[22]. The button has been manipulated in terms of size, colour, contrast, alignment and proximity to ensure that it is only spotted when it is being actively looked for. Users will interact with the application without using the help button, and only if they run into errors and need help, will they begin to search for it.
This button is significantly smaller than the aimer to ensure that it does not catch the users eye before they are looking for it. The colour is 'forest green', to match the theme but also to somewhat blend into the surroundings of Street View more than the stark black of the crosshair. A contrast to the colour of roads and buildings is still present, to ensure it is not too strenuous to find.

o) **Help Page Questions-** The help questions are all in bold and in the applications theme colours. The author has paid special attention to negative space on this page, and ensured that regardless of how long each question is, there is the same space between the start of the question and the start of the questions above and below it. This focus on negative space increases navigability of the page as a whole[23]. Equal spacing between the start of the questions ensures that regardless of how long each question is, they all take up the same space, hence no one question is seen as more important than another. This was also done to facilitate the varying lengths of the answers hidden underneath them.

p) **Help Page Answers –** The answers to the questions above are not in bold but are the same colour as the location detail extracts given on the information page. This is to create a link in the user's mind between the information given on the location, and the information given on the help page. Therefore, this colour will be associated with obtaining information. The author designed the help page where users must click on a question to reveal its answer to ensure there was no information overload opportunities. The user will be tempted to read about other problems and issues if there are answers given, whereas if they must reveal the answers manually, they are more likely to just look for what they need. The author

concluded that the manual labour that goes into the revealing of one or two answers offsets the labour intensity of reading through all the questions and answers if they were provided. Stemming from that, this method presents less information, so the user can find their question quicker and easier. All the FAQs and their answers can be viewed with the application screenshots in Appendix A.

3.7.2 Name

The name 'POInt' was derived from the common abbreviation for **P**oint **O**f **I**nterest associated with maps. The rest of the word was added on because the user must '**point**' to this Place of Interest to gain more information. Therefore, the 'POI' is in capital letters, and the word in its entirety makes up 'POInt'.

3.7.3 Logo

Figure 3.10 *– POInt Logo.*

Figure 3.10 shows the logo that was designed for POInt. The logo of an application is an important feature of its design. It is the first insight the user will get on what the software inside might be like[24]. The logo can be analysed and broken down into its main components and features as follows:

- **Background colour and shading** – The author chose dark blue and green as the colour for the logo primarily as these are earthy colours associated with land and sea. The colours are natural, non-intrusive, and relate directly to POInt's functionality of exploring new, unknown places. The shading allows for both colours to be equally present without a stark defined line. These two colours continue throughout the application itself, and so the author decided it was only fitting that they first appeared in the background of the logo.
- **Writing** – The white writing was chosen mainly to stand out against the darker colours in the background. When a user is searching through a sea of apps, the word 'POInt' will jump out from the dark background and allow for no ambiguity as to what the app is, or what it presents.

- **“I” location marker and globe** – The first tip for “Designing Logos That Don’t Suck”, according to The Design Shack is to use a ‘Visual Double Entre’[25]. The definition given of this is; two pictures wrapped into one through clever interpretation of a concept or idea. This can be seen with the globe inside the marker as the ‘i’ dot. The globe represents discovery and venturing into new, unknown places, while the marker represents location, and POInt’s focus on using the user’s location to produce information.

3.7.4 Theme / Style / Colours

As mentioned in Section 3.7.3, the colours of POInt were chosen because they are earthy colours associated with nature, and by extension, exploration. The colours blue and green run throughout the app in different shades, such as the darker green for the banner against the forest green for the app icons.

The cream background chosen for the information and help page provides a gentler contrast to the overlaying colours. The author felt as though a white background was too harsh, and when using the application in direct sunlight was quite reflective, which hinders usability.

Overall the style of POInt was designed for ease of use for the users. Research was carried out into colours and styles of applications that best suited certain markets and certain app types[26]. The author feels as though the theme selected for POInt wholly encompasses its aim and what service it intends to carry out.

4. Implementation

The author implemented this application using the Agile Software Development Methodology[27]. The cycle of this method is depicted in Figure 4.1[28]. All stages of the below technique can be repeated at each interaction, as many times as needed. This allowed the author to respond quickly and efficiently to changes throughout the development of this project[29]. The Agile Methodology set the author up to easily deal with numerous adjustments in the planning, design, implementation and testing stages. The use of this methodology also enabled the author to test, isolate and address issues when they arose, rather than waiting until the end of development when numerous problems from each stage had culminated.

Figure 4.1 *– Agile Software Development Methodology.*

4.1 Android Studio

To implement this project, the author had to familiarise themselves with Android Studio as this was their first time using the software. To do this quickly and methodically, the author participated in numerous online tutorials and classes. The author concentrated on Android Developer Tutorials, Google Developers Training and TutorialsPoint classes. They also watched multiple YouTube videos on how to set up the software, how to customise it accordingly, and how to develop simple apps. (Appendix D).

The author then created several applications to test different functionalities and constraints of Android Studio. They experimented with the software for a considerable amount of time before proceeding to the implementation of POInt itself. With the immense amount of resources available relating to Android Studio, as well as the specific guidance tools for the entirety of this project, the author moved on to the creation of relevant applications considerably quickly.

The below sections detail the implementation of POInt from the beginning, including the applications and various different routes taken to reach the final product.

4.2 Google API's

4.2.1 Simple Map API

Once the final decision had been made to incorporate Google Map APIs into POInt, the author began practical research and implementation. This was done through various stages of assimilation of the Maps API; its constraints, abilities, and what it could be implemented alongside.

To develop a Simple Map Application, the author proceeded with the following steps:

1. Examined the Google Console Documentation on how to create a Google Developer Account and hence, create a restricted API key[30]. Guidelines on how to create this API key, as well as a direct link to the website, was given in the 'google_maps_api.xml' file in Android Studio when the 'Maps Activity' was chosen.

2. Consulted further documentation[31] and YouTube tutorials (See Appendix D) on how to integrate this API key into the program to produce a working map app. Below is the incorporation of the API key into the 'google_maps_api.xml' file:

```
<resources>

    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyBHfJUOw5KvrCzWWU6pMITzn0xL</string>
</resources>
```

   This is a restricted key that has been created on the authors private account. It is created when the author inputs the unique Package name and SHA-1 certificate fingerprint of the project into the key creator.

3. Next, the author added numerous permissions to the AndroidManifest.xml file. This file contains these permissions and essential application information. Android Studio feeds all this information into the Build tools and the Operating System[32] to run an app. The permission additions[33] are shown below:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

The 'ACCESS_FINE_LOCATION' permission is already included on creation of a new project. The other permissions shown above are implemented so that the app can access the approximate location of the device, open network sockets, write to external storage and read Google Services. These were input to ensure the functionalities of the Maps API could access and manipulate the application as needed.

4. Changes were also made to the grade scripts to enable Google Services to be built into the application. After these alterations, the application needed to resync.

5. The Simple Map App is comprehensively set up and when the app is run, it shows the interface depicted in the first figure of Figure 4.2. The 'MapsActivity' selected on creation of this project is implemented with dummy code that places a marker in Sydney, Australia.

6. In continuation of the research relating to the Maps API, the author manipulated the location details to instead place a marker on Trinity College Dublin. This can be seen in the second image of Figure 4.2. This was done simply by editing the coordinates given in the 'LatLng' variable in the 'onMapReady' function, as shown:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    LatLng sydney = new LatLng( v: 53.343951,  v1: -6.254521);
    mMap.addMarker(new MarkerOptions().position(sydney).title("Marker at Trinity"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

This application was created as a base for the rest of the Maps API practical research. The key research notes achieved from the development of this initial application were; the creation and integration of the restricted API key and the alteration of permissions and grade scripts to produce a functioning map application.

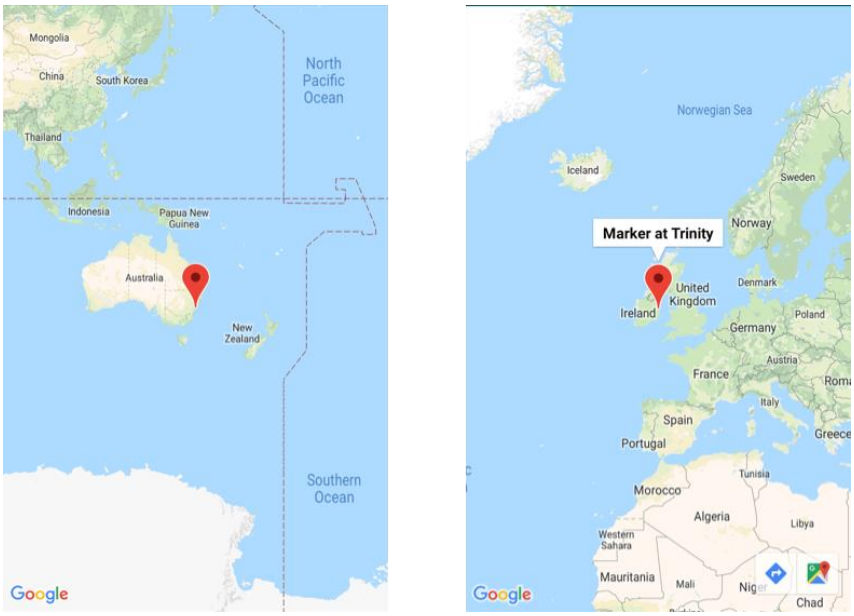


Figure 4.2 *– Simple Map App (Sydney & Trinity).*

4.2.2 Current Location on Simple Map

Throughout all stages of design, the user's current location had always been a core aspect of POInt. Due to this, the ability to access and display a map of the user's location was a critical feature, and the author felt that this should be integrated and perfected as a top priority in the research and implementation process.

Again, after referring to documentation[34] and utilizing YouTube tutorials (Appendix D) on the topic, the author began to add the user location feature into the Simple App shown above.

Before the user location could be accessed by the application, numerous functions were created to ensure that the user granted permission for this to happen. Only after these permissions were manually accepted by the user did the application obtain this information. This aligns directly with the General Data Protection Regulation (GDPR) requirements.

The functions to obtain and display the user's current location are shown below:

```
protected synchronized void buildGoogleApiClient (){
    googleApiClient = new GoogleApiClient.Builder( context: this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build();

    googleApiClient.connect();
}
```

This function creates an instance of GoogleApiClient which provides a common point of entry for all Google services running in the app. This is also used to manage the network connection between these services and the user's device. The Location Service API is explicitly specified for use in this function.

```
@Override
public void onLocationChanged(Location location) {
    LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
    mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
    mMap.animateCamera(CameraUpdateFactory.zoomBy( v: 12));
    if(googleApiClient != null)
    {
        LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,  locationListener: this);
    }
}
```

The above function creates a 'LatLng' (Latitude and Longitude) variable that stores the user's current location. The '.getLatitude' and '.getLongitude' actions each retrieve a double from the Location variable passed through the function as a parameter. This is the location of the user's device that is accessed directly through the Google Location Services.

The 'CameraUpdateFactory' is a class that contains methods to modify the map's camera[35]. In this instance the '.newLatLng' method is used to move the camera so that the centre of the screen is this new location, which is the user's current coordinates. The 'zoomBy' method changes the zoom level of the current camera to a new specified level (as a float).

```
@Override
public void onConnected(@Nullable Bundle bundle) {
  locationRequest = new LocationRequest();
  locationRequest.setInterval(10000);
  locationRequest.setFastestInterval(1000);
  locationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
  if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)== PackageManager.PERMISSION_GRANTED)
  {
      LocationServices.FusedLocationApi.requestLocationUpdates(googleApiClient, locationRequest,  locationListener: this);
  }
}
```

The ‘onConnected’ function displayed above is invoked when Google Play Services are connected and the user’s last known location is sought out. The function sets periodic location updates to be retrieved, in case the user moves and a new current location is determined. The author felt this was an important aspect for POInt as the user would likely be walking, driving or navigating around while using the application. Figure 4.3 shows the result of the finished application.

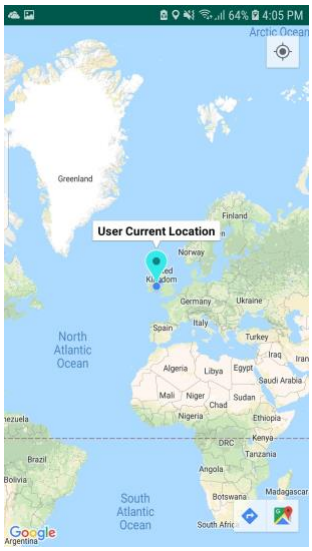


Figure 4.3 *– User Current Location Displayed on Map.*

4.2.3 Hybrid Map

Once user location was integrated into the application, the author felt comfortable enough to expand on the Map API and add a search bar feature. This was to facilitate simple user input, a valuable aspect of any app to perfect. In parallel with this the author experimented further with the different kinds of Maps available and the interactions each of them facilitated.

A significantly common application in tutorial classes and YouTube videos was the Hybrid Map. This allowed normal or satellite (or a mixture of both) maps to be displayed when the user input their desired location.

The implementation of this application utilized knowledge gained from a combination of Computer Science modules over the last four years, as well as some earlier research the author carried out for this project. The application was relatively

straightforward to construct, and fundamentally only included two more main functions from the applications above. These are:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

    getMenuInflater().inflate(R.menu.menu_main, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()){
        case R.id.none:
            mMap.setMapType(GoogleMap.MAP_TYPE_NONE);
            break;
        case R.id.normal:
            mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
            break;
        case R.id.satellite:
            mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
            break;
        case R.id.hybrid:
            mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

This function made use of a simple switch statement (studied in CS1010 –Introduction to Programming) to define and initiate what type of map was chosen to be displayed. The cases were chosen when the user selected a type from a menu which contained the four items. The '.setMapType' is a void method that calls an 'int' constant which is previously defined in the Google Maps API. Each constant initiates a different Map type to be opened[36].

```
public void findOnMap (View v){
    Geocoder geocoder = new Geocoder( context: this);
    try{
        List<Address> mylist =  geocoder.getFromLocationName(e1.getText().toString(),  maxResults: 1);
        Address address = mylist.get(0);
        String locality = address.getLocality();
        Toast.makeText(getApplicationContext(), locality, Toast.LENGTH_SHORT).show();
        double lat = address.getLatitude();
        double lon = address.getLongitude();
        goToLocation(lat, lon,  zoom: 15);
    }
    catch (IOException e){
        e.printStackTrace();
    }
}
```

This function utilizes the Geocoder class. This class transforms a description of a location into a latitude longitude coordinate. The geocoder is constructed to localize responses from the user. User input text is taken in from the 'editText' search box and converted to a String. The '.getFromLocationName' method returns this as an array of addresses describing a location[37].

The first item in the list (the searched for location) is converted into an 'Address', another class that contains a set of Strings representing a location[38]. The longitude and latitude figures were extracted from this address using similar methods described in Section 4.2.2. These coordinates are then displayed on the map.

The functioning Hybrid Map app is exhibited in Figure 4.4. While this app has no direct relation to POInt, it served to enhance the author's confidence in creating Map apps, as well as showcasing the large range of functions the Maps API could facilitate.

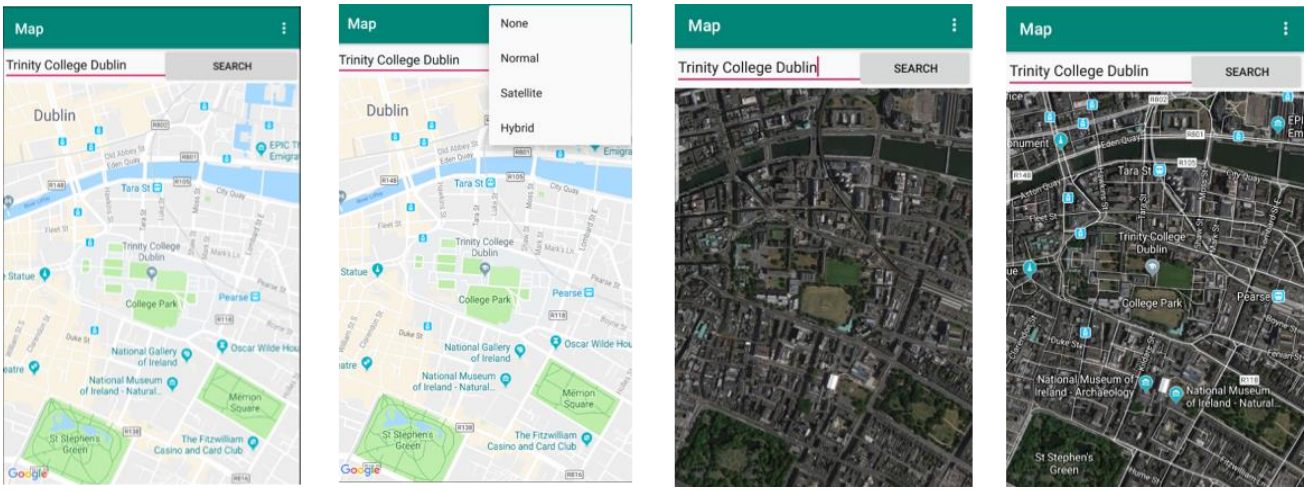


Figure 4.4 *– Hybrid Map Interface.*

4.2.4 Google Street View at Current Location

When the author completed the rigorous design phase of the application showcased in Section 3, it became evident that Google Street View was an essential part of POInt. The integration of Street View, although utilizing many features from the applications above, consisted of a significantly different implementation. The author started by creating a simple Street View App which opened at a predefined location.

While the permissions and grade scripts for Maps and Street View were the same, the main Java file was completely distinctive from the outset. The MapActivity class for the previous Map apps implemented OnMapReadyCallback whereas Street View apps implement OnStreetViewPanoramaReadyCallback, as shown:

```
public class MapsActivity extends FragmentActivity implements OnStreetViewPanoramaReadyCallback{
```

Both callback mechanisms invoke different methods and functionalities, and thus two completely unique application interfaces are displayed.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    StreetViewPanoramaFragment streetViewPanoramaFragment = (StreetViewPanoramaFragment)
            getFragmentManager().findFragmentById(R.id.map);
    streetViewPanoramaFragment.getStreetViewPanoramaAsync( onStreetViewPanoramaReadyCallback: this);
}
```

Subsequently, the onCreate method for Street View also contains differences. This function creates a Street View fragment[39], which is a wraparound panorama view that must also be added to the corresponding XML file manually. This method sets a callback option to be triggered when the Street View instance is used[40].

```
@Override
public void onStreetViewPanoramaReady(StreetViewPanorama panorama) {
    panorama.setPosition(new LatLng( v: 53.344104,  v1: -6.254510));
    panorama.setStreetNamesEnabled(true);
    panorama.setUserNavigationEnabled(true);
    panorama.setZoomGesturesEnabled(true);
    panorama.setPanningGesturesEnabled(true);
}
```

Exhibited above is the main class of the Street View feature and the entry point for all related methods. When a StreetViewPanorama Object is created, it can be manipulated in numerous ways. The author experimented slightly with this class, by enabling various methods such as street names and zoom gestures.

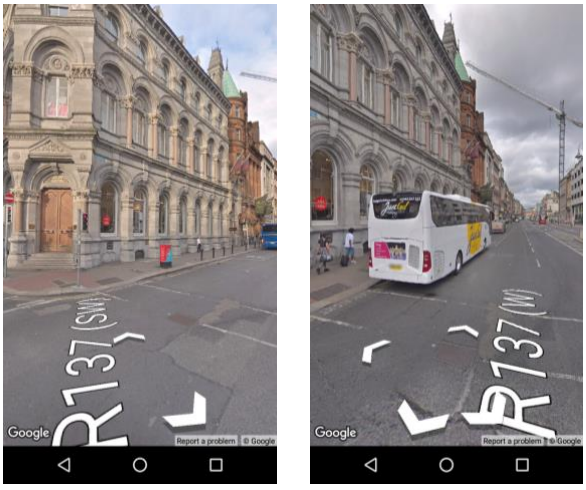


Figure 4.5 *- Street View Interface.*

Once this application was functioning with the necessary features at a pre-set location, as seen in Figure 4.5, the author advanced to integrate this Street View with the user's current location.

```
LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
Log.e(TAG,  msg: "UPDATED LAT LONG" + latLng.latitude + " " + latLng.longitude);
streetViewPanorama.setPosition(latLng);
fusedLocationProviderClient.removeLocationUpdates(locationCallback);
```

This code snippet shows that techniques previously accomplished in Section 4.2.2 (Current Location) to obtain the user's current location are also used in this application. The coordinates found are not used to move the map, but this time they are used to open the Street View interface at this location.
The author printed these coordinates in the terminal to ensure that the correct location doubles were being operated on.

```
@Override
public void onStreetViewPanoramaReady(StreetViewPanorama streetViewPanorama) {
    this.streetViewPanorama = streetViewPanorama;
    this.streetViewPanorama.setOnStreetViewPanoramaChangeListener(streetViewPanoramaChangeListener);
    this.streetViewPanorama.setOnStreetViewPanoramaClickListener(streetViewPanoramaClickListener);
}
```

This function was also altered from the first Street View application above. The 'ChangeListener' is a method called when a new Street View panorama is loaded from a new location. This also prints out this new user location in the terminal each time it changes. The ChangeListener will later be examined in Section 5.1 as a testing method. The 'ClickListener' is called when the user taps on the panorama, usually to alter their position again. Both methods handle movements from the user, and are essential to POInt as the user will be interacting with the Street View interface in abundance to find the exact location they are looking for.

The interface for Street View with the user's current location is the same as Figure 4.5, and can also be manipulated in similar ways.

4.2.5 Nearby Places – Place Picker

Once the author had sufficiently implemented enough Map and Street View applications to progress the development of POInt, the logical step forward was the integration of the Nearby Places API. The author began investigation into this API and its capabilities[41].

The Place Picker is a User Interface (UI) widget in the Places API that provides an interactive map and a list of nearby places. The integration of this is shown below:

```
    getPlace.setOnClickListener((v) → {
            PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();
            Intent intent;
            try {
                intent = builder.build( activity: MainActivity.this);
                startActivityForResult(intent, PLACE_PICKER_REQUEST);
            } catch (GooglePlayServicesRepairableException e) {
                e.printStackTrace();
            } catch (GooglePlayServicesNotAvailableException e) {
                e.printStackTrace();
            }
    });
}
```

To initiate the facility, an intent is first constructed. This is done when the user clicks on a certain element (Button, TextView, etc.) created to invoke the interface. The 'startActivityForResult' method ensures that the activity is started, but also that a result is obtained[42]. In this instance, the result is the Place Picker functionality, created below. The latter part of the function above utilizes a try catch block to ensure Google Play Services are accessible as this API is reliant on them.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_PICKER_REQUEST) {
        if (resultCode == RESULT_OK) {
            Place place = PlacePicker.getPlace( context: this, data);
            String toastMsg = String.format("Place: %s", place.getName());
            Toast.makeText( context: this, toastMsg, Toast.LENGTH_LONG).show();
        }
    }
}
```

This activity creates and returns the result that is requested in the first method. The Place Picker interface is opened and the user is prompted to select a location from the list provided[43]. The list is compiled of various locations around the user's location, or

another specified location. The list includes restaurants, shops, tourist attractions, parks, waterways and other more inconspicuous places. The location the user selects from the given list is retrieved by calling '.getPlace'. Details on this place are then stored in a 'Place' variable to be extracted when necessary. In the above code, the name of the selected place is displayed as a Toast message.

The Place Picker interface in use can be seen in Figure 4.6. The author also experimented with colour and style manipulation for this application as preliminary usability research.

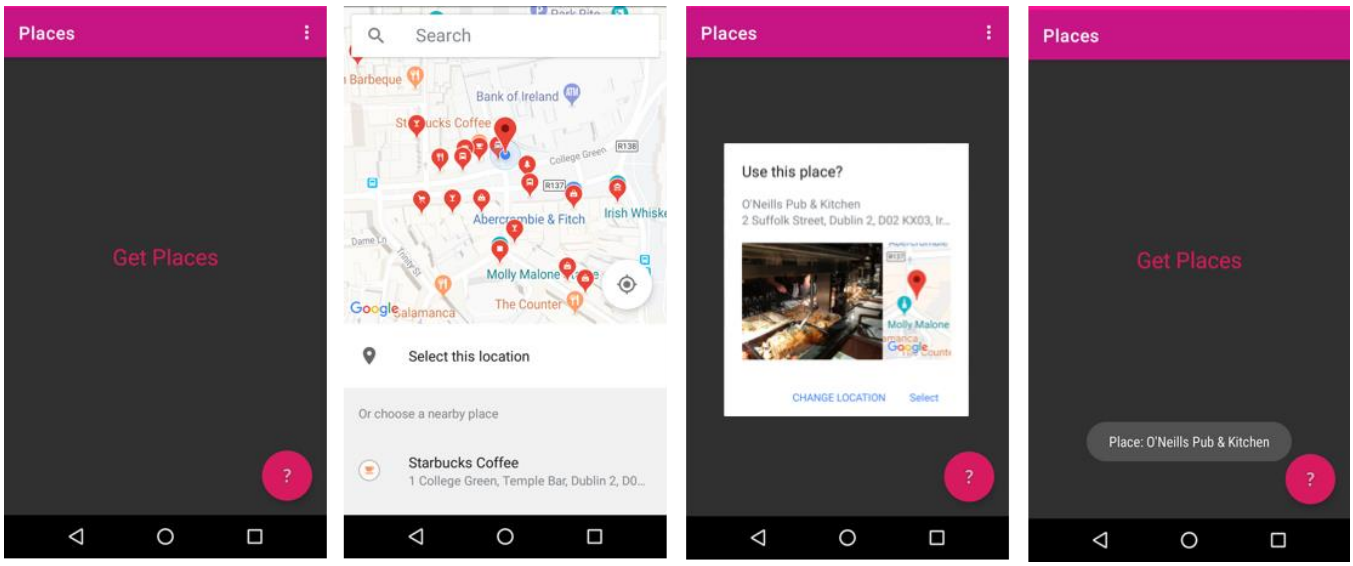


Figure 4.6 *– Nearby Places Interface.*

During the implementation of this API the author carried out considerable research into narrowing down the types of places given in the list, so that only places of interest to tourists would appear. The author soon discovered that this was a bug within the API that has not been amended by Google yet, rendering Place Picker considerably un- customisable[44].

After overcoming this error and implementing Nearby Places as a crucial part of POInt, the author identified that the Google Place Services version of Places SDK was being depreciated and eventually will be turned off on July 29th 2019. As this is the only API of its kind, it was irreplaceable as a centrepiece of POInt.

4.3 Combination of API's

To combine the relevant API's into one comprehensive interface (Google API's were not directly implemented for the information page or the FAQ help page), the author had to make slight changes to some of the files. The 'MapsActivity.java' file was implemented with Street View launching on the user's current location as the primary function. This was implemented identically to the latter part of Section 4.2.4. Therefore, the main changes took place in the xml files, as various layouts had to be amalgamated. In the 'activity_maps.xml' file, buttons and images had to be implemented as overlays to the Street View Fragment. As this fragment was created inside a Frame Layout, this overlaying method was possible, and each new child created was integrated on top of the previous one[45]. This worked to the authors advantage, and created the unique interface that can be seen in Figure 3.6 in Section 3.7.

As the Place Picker API contains its own external interface, this did not have to be compensated for, and the API was implemented almost identically to Section 4.2.5 above. The only difference being the button in POInt was created as a rounded, transparent, drawable shape in a separate xml file.

The implementation practices outlined in Section 4.2 greatly aided in the construction of POInt, as code snippets and in some cases, whole functions could be copied from the practice applications and put straight into POInt. Each time this methodology was invoked, the new feature and the app in its entirety was tested vigorously to ensure that the copied code worked correctly and carried out the intended actions. This is explained in detail in Section 5.1.

The addition of the information page and help page were executed by creating new Activities in Android Studio, which automatically create and format the accompanying xml files. The implementation of these Activities is described in Section 4.4 and 4.5 below.

4.4 Obtaining and Presenting Information

4.4.1 Location Details

The information presented to the user is accessed through the Places API using a Place ID. This is a unique ID tied to the location that the user chooses from the Place Picker interface. Using this ID, the application can extract certain datatypes from the place. The details are stored and accessed from the Google cloud, and so only locations which have provided Google with these details are available.

In POInt, the additional data is obtained from the specified location like so:

```
Place place = PlacePicker.getPlace( context: this, data);
if (place.getWebsiteUri() != null) {
    placeUri = place.getWebsiteUri();
    placeAddress = place.getAddress();
    placeNum = place.getPhoneNumber();
    placeRate = place.getRating();
    placePrice = place.getPriceLevel();
    placeName = place.getName();
    openPlacesInfo();
```

The Place Picker intent is created as seen in Section 4.2.5. (Nearby Places – Place Picker), with the same function parameters and conditions. The method in POInt however, extracts the various details of the place and stores them in the appropriate variables.

Website URI –This is a direct link to the website provided to Google from the location itself (www.oneillspubdublin.com for O'Neills Pub & Kitchen).

Address –This is the physical address stored in the Google cloud. (e.g. 2 Suffolk Street, Dublin 2, D02 KX03, Ireland for O'Neills Pub & Kitchen) This is not to be mistakenly interchanged with the location.

Phone Number – This number is always presented with a country code, to make it fully accessible for tourists, the target market of POInt (e.g. +353 1 679 3656 for O'Neills Pub & Kitchen).

Rating –This is a number that is from 1.0 to 5.0 and is presented as a cumulative average from Google user's reviews (e.g. 4.2 for O'Neills Pub & Kitchen).

Price Level –This can only be a whole number from 1 to 5. Anything outside of this is presented as a null. This information again is obtained from the Google cloud (e.g. 2 (converted to €€) for O'Neills Pub & Kitchen).

Name –Although locations can be found without entering this exact name, this is what will appear in the list on the Place Picker interface, and this is the official name provided for the location by Google.

This information is presented openly by Google in their Map website, as in Figure 4.7.1. The same information as presented in POInt is shown in Figure 4.7.2, for comparison.

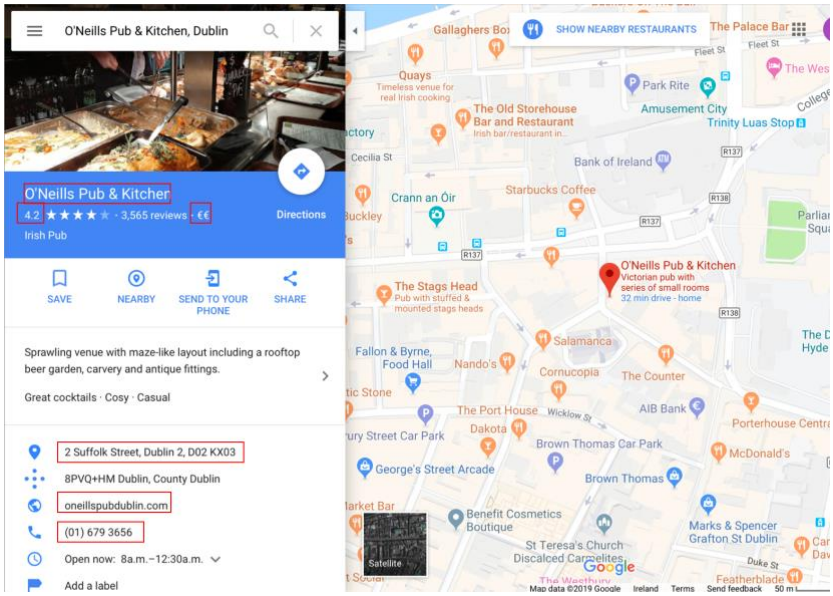


Figure 4.7.1 *– Google Maps Information Screen.*

Figure 4.7.2 *– POInt Information Screen*

To present this information inside a new page (Activity) in the application, it had to be stored in a specific way.

```
public void openPlacesInfo(){
    Intent intent = new Intent ( packageContext: MapsActivity.this, placesInfo.class);
    intent.putExtra( name: "URL", placeUri.toString());
    intent.putExtra( name: "Address", placeAddress.toString());
```

This function is called when the user confirms their location selection through the Place Picker Dialog Box. The new intent that is created facilitates the change from the MapsActivity (default) to the Information page. Shown above are two examples of how the variables are stored to enable them to be transferred to the information page.
The putExtra method stores this information in a way that can be accessed from any activity within the application[46]. The method consists of a String name, which is how to identify the variable, and the value, which is byte data value.

Shown below is an extract of how this same data is accessed from the placesInfo.java file. The intent and its extended methods are called again. The byte values stored are saved as new variables in this Activity to then be displayed.

```
address.setText(""+getIntent().getStringExtra( name: "Address"));
number.setText(""+getIntent().getStringExtra( name: "Number"));
```

As the price data is given as a number, the author decided to translated this into Number of Euro Signs to make it more understandable for users. This was done through simple 'If' statements. The price ranges from € to €€€€€, and anything else would be presented as 'N/A'[3]. Name, Address, Rating and Phone Number are all displayed the same way, as can be seen from Figure 3.7 in Section 3.7. (Final Design) and Appendix A.

4.4.2 Website and Phone Call

The website data was handled differently as the author intended users to be able to access this website through POInt directly, but in their default browsers. The phone number while displayed similarly to the other information points, was also handled differently, to enable the user to call the number with one click through POInt, instead of having to manually copy it into their dialler.
The code shown below demonstrates how this was done in the placesInfo.java file.

```
public void browser1(View view) {
    String placeWebsite = getIntent().
            getStringExtra( name: "URL");
    Intent browser = new Intent(Intent.ACTION_VIEW,
            Uri.parse(placeWebsite));
    startActivity(browser);
}
```

```
private void dialPhone(final String phoneNumber) {
    startActivity(new Intent(Intent.ACTION_DIAL,
            Uri.fromParts( scheme: "tel", phoneNumber,
                    fragment: null)));
}
```

A new intent was created to parse the String containing the location website or the phone number. ACTION_VIEW and ACTION_DIAL were used to initiate the respective actions required[47]. The Strings were passed through a Uri Object, which accesses the necessary content (through one-to-one mapping) to operate on, to bring up the website in the browser or the phone number in the dialler.

[3] N/A is a common abbreviation for Not Applicable. It can also be taken to mean Not Available, or No Answer.

4.5 Usability

4.5.1 Splash Screen

The splash screen was implemented by adding a videoView to a new Activity, which was specified in the AndroidManifest.xml file to open first when the app was launched. The video takes up the entirety of the screen and provides a short demonstration of how to use POInt.

The author added a straightforward button at the bottom of the page to allow users to skip the video if they already knew how to use the application from previous usages. The button opens MapsActivity.java as soon as it is clicked on.

The splash screen can be viewed in Section 3.7, Final Design and Appendix A.

4.5.2 Aimer

The aimer was implemented as a button. The button had to be created in a separate xml shape drawable file. The outline and transparency of this button were also stated in this file. The button was implemented to open the Place Picker API, and the code similar to which this button initialises can be seen in Section 4.2.5. The crosshairs in the centre of the button are also a drawable file. The aimer was implemented to aid the user in pinpointing their exact location, and for aesthetic effect.

The aimer in the centre of the main MapsActivity screen can be viewed in Section 3.7, Final Design and Appendix A.

4.5.3 Help Page

The Help page was created using a scrolling Activity. After careful research into the implementation of this activity type, the author discovered that the scrolling layout only allowed for one child, which in this case was one TextView[48]. To combat this, the author created a separate Content xml file which contained the relevant FAQs in a Relative Layout, surrounded by a Nested Scroll Layout. This allowed multiple FAQs to be entered and interacted with separately. In the Help.java file, each FAQ was made clickable, so that users could interact directly with the FAQ they needed.

```
public void q1Clicked(View v){
    Q1.setVisibility(View.INVISIBLE);
    A1.setVisibility(View.VISIBLE);
}
public void a1Clicked(View v){
    Q1.setVisibility(View.VISIBLE);
    A1.setVisibility(View.INVISIBLE);
}
```

This method (q1Clicked) ensures that when question 1 is clicked, the relevant answer appears. This method (a1Clicked) makes answer 1 disappear and the question appear again, to limit confusion and information overload.

5. Testing

5.1 Incremental Testing

The author partook in incremental testing at each stage of the implementation process, and sometimes at numerous points throughout the design stage too. This was to ensure that small errors were not carried forward to engage in the 'Snowball Effect' and become more harmful. Testing is a critical feature of Figure 4.1 (Section 4) and could not be overlooked. Testing took place in 5 main ways.

1) **Per Feature-** For each new feature added to POInt, testing was carried out to ensure this feature functioned correctly, and carried out the required capabilities. It also ensured that the feature adhered to front-end requirements and did not hinder the usability of the application. Each feature was tested before the implementation of a new feature could begin. This decreased the possibility of errors piling up, as well as ensuring that errors which were present could be easily identified and fixed, before being buried in more code.
2) **Back-end Console –** Back-end testing was carried out by printing all variables into the console. This method was also carried out at each stage of implementation, or after a new feature was added. The author printed system changes into the console, to ensure that data points were being accessed and stored correctly. This was to determine that when certain buttons were pressed, only the required actions took place, correctly and in a specified order. The author implemented the 'ClickListener' as mentioned in Section 4.2.4, which printed all location changes into the console, like in Figure 5.1. This allowed for assuring that the correct location was being operated on, as well as providing an understanding into where the location was being accessed from (cell phone towers, satellite, etc.) and what factors affected the accuracy of it. Testing in this way ensured that faulty actions or variables were identified before they could be carried through other stages of the application.

Figure 5.1 *– Terminal, Evidencing Back-end Testing.*

3) **Front-end–** Interface testing was essential to POInt, as it is a user-friendly application that relies heavily on the correct functionality of this. Interface testing was also carried out incrementally, with vigorous evaluations carried out for each new interface integration. Continuous testing of front-end requirements ensured that each different interface of the application looked the way it should to present to the user, but also confirmed that the interaction with the interface was properly implemented. To prevent buttons from carrying out the wrong functions, or scrolling features to be integrated backwards, this testing had to be carried out again with each new feature or interface. Certain parts of the front-end assessment were carried out through user testing, as is outlined in Section 5.2 below. Both techniques were used in parallel throughout implementation to ensure that usability was always at the core of the application.
4) **Using Section 3 (Design and Requirements) –** As mentioned in point 2 and 3 above, recurring testing was executed in close examination of the Design and Requirements laid out in Section 3. When each new feature or function was added, the author referred to the original specifications to ensure that the new addition was in line with this. If the feature did not adhere to the entirety of this section, it was modified or removed. This not only ensured that features did not stray outside the scope, but determined that all requirements were met.
5) **Per variable –** In line with point 2, back-end testing; each new variable or data point was tested vigorously. This was done through back-end and interface testing. The back-end testing confirmed that each variable was the correct one, and the interface testing established if the variable was being relayed to the users accurately. This was done for each information variable, as well as each button, and feature. Iterative testing in this way ensured that wrong information did not carry through and affect other information further down the implementation line. An example of this would be; the wrong website being transferred into the information page and then further into the users default browser.

5.2 Usability Testing

User testing allowed the author to evaluate how users interacted with the application on numerous different levels. This method was used to detect issues that required a different perspective to the developer's perspective. To carry out this form of testing, the author sought ethical approval from the Research Ethics Committee. The documents submitted for this process can be viewed in Appendix B. Once the author was granted approval, they began the testing process.

The author recruited 16 participants to undergo testing for POInt, with all participants remaining anonymous. Each participant that was enlisted was given the documents from Appendix B as a briefing method. When the participant had signed the consent form and was willing to take part in testing, they were shown where to find the app on the test phone, and began interaction with it. Once they were satisfied with their usage of POInt (approximately 10 minutes) they were given the testing survey to complete. The results of this have been analysed below.

Demographic Information

1. Most common age group was **18-25**.
2. **93.75%** of participants were Irish residents.
3. **87.5%** of participants were Dublin residents.
4. **75%** of participants said they prefer using apps to obtain information rather than physically obtaining the information.

Due to time and resource constraints, the author had access to limited diversity in participants for testing. This can be seen from the above results, as most contributors were Irish residents, and in particular, Dublin residents. As the test was conducted in Dublin, Ireland, it meant that their perspective of the application was different to someone in an unknown place. For this reason, the author has considered the various bias factors, and subsequently analysed the results accordingly.

Question 4 provides evidence of the growth in apps and app usage as a preference to the alterative manual work giving the same result. Question 4 must be considered in

relation to Question 1, and the author acknowledges that age may have slightly skewed this answer, but does not make it any less relevant.

Technical Information

5. **93.75%** of participants could open the application without difficulty.
6. **100%** of participants said the application fit the screen correctly with all buttons fully visible.
7. **31.25%** of participants experienced lag using the application.
8. **87.5%** of participants said the app opened on their current location
9. **100%** of participants could navigate using the arrows to their required destination
10. **100%** of participants said the application opened the location website correctly.

From a technical aspect, POInt performed well at the user testing stage, with the main concern being lag. Five participants specified that they experienced lag at some time during their use of the application. No questions were asked or put on record to further enquire if this was due to their internet connection, or the application itself. While the author conducted other testing, which provided further evidence that this was most likely caused by connection failure, they still attempted to fix any problems with the implementation that might have contributed to this, such as additional lines of code executed without reason. These adjustments were made after the user testing phase, and so were only incrementally tested.

The other minor issue was concerning the application failing to open on the user's current location. This would not have been due to location services being turned off on the device, as the test device was configured prior to testing, and location services were confirmed to always be on. The author has considered this issue in depth and concluded that this may have been due to the user being indoors, where Street View is unavailable, or in a built-up area, where location data pings off buildings and can be obscured[49]. The relevant code was also investigated, but was ruled out, as the location function requested "PRIORITY_HIGH_ACCURACY" to gain the most accurate user location details[50].

Usability Information

The author utilized the System Usability Scale[51] (SUS) created by John Brooke in 1986 to evaluate POInt further. As can be seen in Appendix B, this section of the questionnaire contained 10 standardised statements, both positively and negatively poised, to be rated from 1 – Strongly Disagree, to 5 – Strongly Agree.

Interpreting the scores was done by converting each answer to a new number. This was carried out by subtracting one from the user response for all odd numbers, and subtracting the user response from 5 for all even numbers. This gave scores from 0-4, which were then added together and multiplied by 2.5. The overall score was out of 100, which was not to be interpreted as a percentage, but instead as a percentile ranking. A score of 68 is considered average, with everything above providing evidence for exceptionally good software, and anything below being inadequate.

The author implemented this method, with the aid of an online tool[52], and arrived at an average score of **82** which can be classified as an 'A' application[53]. This classification falls under the bracket of apps that 'are more likely to be recommended to a friend'. The author was satisfied with this SUS score, and decided not to make any significant changes to the usability aspects of the app, for risk of hindering this.

In conclusion, the user testing exercise provided invaluable insights from a different perspective into POInt. The author could identify issues and constraints that did not appear obvious to them during development, but which did affect the usability of the app. The user testing allowed individuals other than the author to experience the application, which proved to be a successful study overall.

6. Business Model

The Business Model Canvas was proposed by Alexander Osterwalder, a Swiss business theorist, consultant, and entrepreneur. The canvas was based on his earlier book: Business Model Ontology[54]. The canvas provides an easy step-by-step guide for entrepreneurs when developing the initial plans for their business. It enables them to lay out the key aspects of the business in logical steps, in a way that allows for easy and successful expansion.

The author decided that they would lay out the business strategy for POInt using this canvas, as it breaks the business down into its most important components, and does not allow for exaggeration or unrealistic assumptions[55]. This business model was also used as a tool for the design and implementation of the application as a whole. The model was considered and consulted when all major decisions were made in the earlier stages, as it was an app originally created for use and potential expansion. An outline of the canvas can be seen in Figure 6.1, and a more detailed, completed canvas can be viewed in Appendix C.

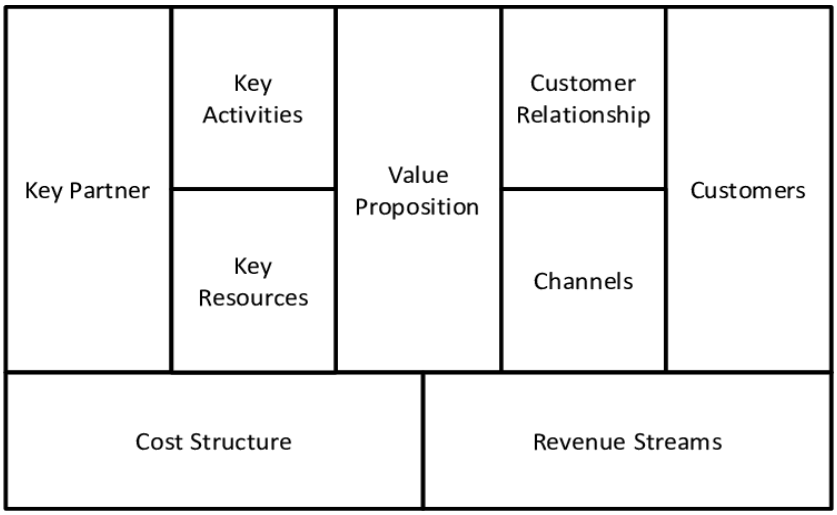


Figure 6.1 *– Business Model Canvas.*

6.1 Value Proposition

POInt delivers value by providing tourists with an application that they can rely on with zero knowledge of their surroundings. This improves the quality of their trip to the area, and allows them to discover places of interest that do not appear on conventional travel guides. POInt solves the consumer's problem of needing to recognise buildings

and streets on a 2D Map, by allowing them to observe these same locations through real-life imagery.

As an elaboration of the research in Section 2.2, it can be concluded that POInt is the only application of its kind on the market currently, which again adds value. The application can be said to 'get the job done', in which it does not include any unnecessary functions to depreciate its value to the customer. POInt is accessible and easy to understand for all users within the target market.

6.2 Key Partners

Key partners refer to external partners needed to develop a product or service. This partnership may be temporary (e.g. during set-up) or may last the lifetime of the product or service.

Currently, POInt has no key partners, bar the lead developer of the application and temporarily, the ethics council to approve user testing. To expand reach and notoriety POInt would engage in partnerships with tourist bodies such as Tourism Ireland[56] and Fáilte Ireland[57]. For more formal features, POInt would establish connections to the Department of Transport, Tourism and Sport (DTTAS)[58]. Forming alliances with these organisations would allow POInt to increase their informational reach, as well as gaining advertising privileges. These partnership aspects will be examined further in various sections of this business plan.

Further down the line, POInt would also aim to establish closer ties to Google to avail of restricted information and databases as a further Unique Selling Proposition (USP)[59]. This USP would be access to information that other applications of this same type would not have. The Google partnership would allow for the acquisition of the scarce resources of knowledge and technology.

A bonus for POInt would be joint ventures with airlines (e.g. Ryanair) and hotel chains (e.g. Clayton or Maldron). This could encourage deals and discounts to be created, as well as further advertising strategies. For example, when you buy a Ryanair ticket[60], not only will you avail of car hiring services and hotels, but POInt will also be displayed as part of the complete travel package.

6.3 Key Activities

This can be defined as activities that are critical to the business creating value. The most prominent activity for POInt is the successful implementation of the application.

This creates value by allowing the application to be useful to its users. Stemming from this, key activities include maintaining and updating the app regularly to ensure no bugs hinder usage. Updating the application also provides value as the users are getting the most relevant and recent information, which is more likely to be accurate, for example, when businesses change location.

A key activity in line with revenue streams would be to market to POInt's target customers efficiently. This could be done by utilizing the partnerships discussed above and availing of advertising space on tourism, airline or hotel websites. This would aid POInt in gaining recognition and would add value for tourists, as they would be able to gather all the necessary information from one source.

6.4 Key Resources

Key resources are the assets fundamental to providing value to customers. POInt's primary asset is its uniqueness on the market. Providing customers with a way of finding out large amounts of information with no prior knowledge is a considerable USP. Another asset POInt possess is access to Google data and immense location specific details. While this is not unique to the app, it is still an indispensable resource in providing value to the consumers.

Again, employing the above key partnerships will provide POInt with unique resources in the form of distribution channels and revenue streams. Overall, the main resource POInt exhibits to provide value is its access to information and access to the target market.

6.5 Customer Segments

There are five main customer segments outlined as part of the business model. The author investigated all five divisions to derive which one POInt could be best mapped to. The analysis can be seen below:

- Mass Market – Wide pool of potential customers as the product is relevant among the general public.
- **Niche – Customer segment based on highly specific needs and unique client traits.**
- Segmented – Create further segments in the main customer segment based on slight variations in the customer's demographics and needs.

- Diversify – Flexible in iterations of the product. Tweaking product or service to suit needs of segments with different needs.
- Multi-sided Platform – Customers who have a relationship with each other, create pull with both sides.

The author concluded that the Niche market best suited POInt. While POInt can be utilized by a wide range of users, it was specifically designed to cater to the needs of tourists in unknown surroundings. The customers highly specific needs are to gain information about their surrounding area, without having to manually input the name of a location, or find it on a 2D map. The unique client traits are, as mentioned, their lack of knowledge of the area and their ability to obtain the desired information through other means. POInt caters best for this niche market and by focusing on one small target market, can address their precise needs thoroughly, rather than trying to focus on multiple needs in a 'mass market' scenario.

6.6 Customer Relationships

Again, the author investigated the various customer relationship categories, to arrive at a conclusion that best matched POInt's interaction with its customer base. The analysis was as follows:

- Personal Assistance – company interacts with the customer directly, by providing the human touch. Assists customer presale, at sale and post-sale.
- Dedicated Personal Assistance – very close interaction between customer and company through dedicated representative. Responsible for entire experience.
- Self-Service – places customer experience on tools the company provides for the customer to serve themselves.
- **Automated Services – customized self-service relationships where historical preference of the customer is taken into account to improve experience.**
- Communities – creating communities of clients allows organisations to communicate with them directly. Enhanced client experience through shared challenges and solutions.
- Co-creation – customer has direct hand in the form the product or service will take.

The author determined that POInt took on an ‘Automated Services’ angle. The customized self-service interaction is present within the help page containing FAQs in the application itself. This method uses historical data to better the customers’ experience. This historical data was obtained through user testing, and research into similar apps and FAQ sections, so the author could derive twelve questions that appeared most commonly throughout. These questions now serve as a self-service interaction with the application, and therefore with the ‘company’ itself. Any other, more personal customer relationships would be inappropriate, mostly because they are harder to facilitate through an application. This would also be unsuitable as the app is fundamentally designed to be quick and easy to use, so if a user did have a problem, solving it with human interaction (whether personal assistance or communities) would be time consuming and unnecessary. Automated services are easily facilitated and best suited to the target market of POInt.

6.7 Channels

Channels are the medium in which the organization reaches and provides value to its customers. Channels can be company owned or partner channels. In analysing both, the author has noted the exceptional benefits and growth of social media, for example, in Ireland social media advertising was up 37% in 2018 from the previous year[61]. However, after continued research, the author concluded that this was not the most appropriate way for POInt to reach tourists. This is because social media platform usage varies significantly from country to country, as can be seen in Figure 6.2[62]. Targeting tourists from the US would require a completely different strategy to Spanish tourists, and would therefore be unfeasible. Hence, the main avenue POInt will use to reach its customers will be through partner websites. When booking flights, hotels or tour buses, users will be notified of the existence and features of POInt, as an add-on to their travel essentials. POInt will reach consumers through non-intrusive ads on partner websites, so when a tourist is searching for ‘The best things to do in Galway’ a POInt advertisement will pop up notifying them that they can add to their checklist by wondering out of their comfort zone to experience unique locations without prior research.

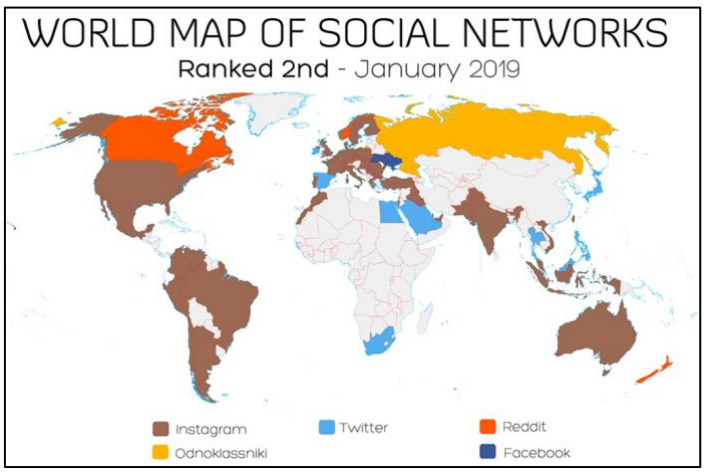


Figure 6.2 *– Popular Social Networks Globally.*

6.8 Cost Structure

POInt is a prime example of a value driven business. Therefore, the focus is to bring value to the customer, rather than focusing on minimizing investment into the business. The costs that POInt inevitably do incur can be broken down as follows:

- **Fixed Costs –** The main fixed cost of POInt is the Google Play Store registration fee. This is a once off fixed payment of £25.[63]

  Other fixed costs are predicted to be partnerships with tourism bodies and booking sites. POInt can pay a fixed subscription (monthly or annually) to partners to advertise on their sites, or offer discounts on their behalf.

- **Variable Costs –** The primary variable costs are associated with the distribution of the app. Once POInt has gained feasibility, there is potential to start to monetize the application. This could be an initial fee for download, or an upgrade fee to minimize ads, or unlock additional features. When users pay towards the application through Google Play Store, 30% of the revenue is kept by the store[64]. This 30% is a variable cost, as users may not choose to upgrade. If users have paid for partner applications, the initial download fee may be wavered, therefore this cost cannot be calculated from the outset, and remains variable.

  Another major variable cost stems from the authors Google Developer account. This account is billed on average $0.25 per 1000 API requests. Specifically, for Dynamic Street View (the basis of POInt), it costs $14 for up to 100,000 requests and for the Places API it costs $17 per 100,000 requests[65]. Therefore, this variable cost moves in tandem with the app usage.

- **Economies of Scale –** This is when the growth of a business results in a reduction of costs. This can be exhibited mainly regarding partnership advertising. When POInt grows in popularity, it can provide advertisement for less popular sites. For example, Tourism Ireland could benefit from the usage of POInt for tourists venturing into Co. Kildare and visiting Castletown House[66], a hidden gem left off many tourist lists. These ads could be in exchange for a reduced subscription cost. In this way, POInt would make use of being an economy of scale to lower certain costs as popularity increases.
- **Economies of Scope –** This is when the reach and partnerships of a business result in a reduction of costs. As mentioned in numerous previous sections, partnering with similar core businesses provides POInt with scope to reduce costs, as well as create value in many other ways.

6.9 Revenue Streams

As POInt is a new business that will initially be hitting the market solo and unrelated to other businesses, revenue is projected to come slowly, if at all. Revenue is crucial however, to offsetting costs and producing a profit to keep the application in service. As 98% of worldwide app revenue comes from free apps, according to Google Play[67], there is money to be made without charging per download. Upgrades and restricted unlocks may be considered when the application gains credibility, but will not be used as a primary source of revenue. The main source of revenue will be advertising. Advertising in POInt would not be intrusive or irrelevant. Ads from partner businesses as well as related items only will be authorised, so as not to decrease the value to the customer. Ad providers will be charged on a pay-per-click basis, and ads will mostly be in the form of banner ads and native ads[68]. This source of revenue will again increase with popularity, and add value to the overall consumer experience within the app.

Another source of revenue that is projected in the long-term future is subscriptions and in app purchases[69]. This means that subscriptions to tourism guides and related information would be provided as part of the application, while in app purchases could eventually facilitate purchasing partner discount tickets. Therefore, through POInt, users would be able to purchase 10% off tickets to a 2pm Guinness Storehouse tour as well as signing up for a subscription of Sheridan's Irish Cheese membership. The proceeds from aiding in these sales will come in as revenue to POInt.

7. Conclusion

This report has outlined the stages the author completed to arrive at the integration of a fully functional application, which adhered to all predetermined requirements and specifications. The author feels as though POInt was an accomplished production, and personally, has gained valuable insights from the process.

7.1 Evaluation

Overall, this project was successful and completion was attained with minimal complications. The author started production on this application with little to no Android app development skills, and feels as though their confidence in this area has been heightened. The author had the opportunity to showcase skills learnt in all aspects of Computer Science and Business to complete this project in its entirety.

The introductory chapter of this report outlined the aims of the project and the motivation behind it. The author successfully completed all the aims mentioned, while also fulfilling the ideas identified in the project motivation.

The author felt as though the background and research chapter enabled them to successfully isolate a gap in the market that could be exploited by an application of this calibre. The existing applications study not only informed the author of where mistakes could be made and functions could fall short, but provided them with ideas of what was useful and appealing to users. As Google Services were the basis of POInt, extensive research into this also proved invaluable for moving forward with implementation.

The design and requirements section contained the most crucial information, as the author habitually referred to this chapter throughout the production of POInt. The author felt as though the design changes mentioned at the start of this application were essential journeys to be made, and without them the final design may have been different, and much less comprehensive. Repeatedly referring to this information ensured that the final product did not steer out of scope, while also assuring that the app contained all the necessary features. The author confirms that each requirement, functional and non-functional, was adhered to. The application performs as intended

due to the author strictly following the front-end and back-end designs that were blueprinted prior to implementation. The final designs showcased in Section 3.7 is evidence of this.

The implementation chapter allowed the author to document their thought process throughout the creation of POInt in tandem with the relevant development progress. This encouraged the author to see more clearly and pinpoint what worked and what didn't work early in this procedure. This chapter was cumulated from draft notes kept by the author throughout the coding stage of this project, therefore, the author is certain that the chapter does not fall short in detailing all the steps, directly related or not, taken to reach the resulting application.

The testing of POInt proved beneficial both from an app improvement perspective as well as a personal one. The testing highlighted where previously unidentified issues occurred, which enabled the author to respond appropriately. It also allowed the author to develop new skills surrounding this method and the ethical analysis it required.

The business plan for POInt was devised using knowledge from previous business modules studied, coupled with online research. The author broke this plan down into the nine most critical aspects and analysed each one separately. The author concluded that this plan was a feasible strategy for POInt to employ in the future, and looking at the application from this different perspective also allowed the author to identify various other strengths and weaknesses of the project.

In conclusion, the author deems this a profitable project, both personally and technically. All requirements were adhered to and included in the application, and testing was carried out comprehensively to ensure this. The author also gained useful skills such as app development, time management, regular reporting to a senior and presentation practice.

7.2 Difficulties

The author encountered numerous difficulties throughout this project of various severities. However, the author felt as though no difficulty resolutely affected the

completion of this project, and overcoming each one became a valuable learning experience.

7.2.1 Design Changes

The first major difficulty the author was faced with was the design changes as outlined in Section 3.1. This was noted as a substantial issue due to its time-consuming nature. The author created each design as a functioning application to establish the feasibility. The author decided to move forward in this way, as they are a Computer Science and Business student, and had a significant amount of free time in the first semester of this project. This extra time enabled the author to test each design thoroughly before deeming it achievable or not. This was still a major difficulty, as each new idea had to be drafted from scratch, implemented and then assessed. The author surmounted this difficulty and utilized the research done for these previous designs to better the design and functionality of the final product.

7.2.2 Nearby Places API

The second, and most prominent dilemma the author faced was the extensive restrictions of the Nearby Places API. As previously touched on in Section 4.2.5, this API contains numerous constraints and bugs that severely hindered its functionality in POInt. The most critical problem was the inability to filter the types of places that were presented in the Place Picker list. The author estimated serious impacts on the usability of this application for the target market due to this. This constraint meant that tourists to an unknown area would have to sift through a list of irrelevant, and arguably confusing information to find what they were looking for. This would be exceptionally problematic when tourists are searching for information on more obscure places (such as church ruins or graveyards) that appear as the standard grey box icon in the API[70]. The author discovered that filtering the place types was only an option in a website setting, rather than an Android application[71]. Through research routes into other APIs as well as the attempted manipulation of this API the author concluded that regardless of the constrains, this was still the best method for the desired functionality. As this app is designed to be used in remote, under-documented areas, the author has argued that there will be limited places to be shown on the Place Picker list, and therefore, it will most likely be obvious what the user is looking for information on.

A recent addition to the complications involving the Place Picker API (again mentioned in Section 4.2.5) is that it has been depreciated as off January 29th 2019, and will be turned off completely on July 29th 2019. A new version of 'Places SDK for Android' will be released on the market soon, which must be accessed as a static client library, instead of through Google Play Services as it currently is[72]. This issue is dealt with in Section 7.3 below.

7.2.3 User Testing Demographics

The last major problem the author faced during this project was gathering participants for the user testing phase who fell into the target market. This was mentioned in Section 5.2 briefly as a bias factor in the results. User testing was carried out when the application was near completion, and for this reason, the author was limited in time and resources. The author recruited friends, family, colleagues and course mates to take part in this testing, most of which accessed POInt and subsequently completed the survey in an area they were familiar with. While this still provided useful technical and usability information, opinions on if the app comprehensively carried out its intended use were harder to gauge. Assessing if enough information was present, or if the information given was relevant from a user's point of view was even more difficult. Regardless, the author concluded that this constraint did not hinder the project, and important results were still obtained.

While the author did face numerous challenges, some larger than others, they believe that they overcame each challenge successfully and utilized the challenge as a learning point for the future. The author felt that some challenges even contributed to the successful completion of POInt and provided them with invaluable personal experience to take forward.

7.3 Future Work

7.3.1 Augmented Reality

The most ambitious development the author foresees for POInt would be the conversion from Street View to Augmented Reality (AR). The author noted from the outset of the project that while AR would have carried out the required functionality seamlessly and more efficiently, it was not feasible due to the authors skill set and

allotted time for this project. The prospect of converting POInt to AR in the future would allow users to enjoy a vast upgrade in the imagery they are presented with to match with their real-life views[73]. Implementing AR would allow layers of digital information to be added directly onto the user's camera view, which would make it more obvious to tourists trying to identify places what each place was.[74]. AR would upgrade POInt into a new class of navigation aid apps, enabling it to compete with larger, more popular applications and preform more in depth tasks[75].

7.3.2 Partnerships

As specified in depth in Section 6 (particularly Section 6.2) creating partnerships with tourist companies in Ireland would be a productive future move for POInt. Partnerships would allow POInt to implement related additional features as well as monetize on unlockable items. Partnerships provide potential for advertising opportunities directly to the target market, as well as subscriptions to offer discounts and deals on behalf of these partners through POInt itself. Partnerships with Google would also be a productive move to avail of more up-to-date information. Alliances which are beneficial for both sides involved would be a logical and favourable step forward.

7.3.3 New Place Picker API

Section 4.2.5 and Section 7.2.2 mention the depreciation of the Place Picker API. From the depreciation date, for POInt to continue operating the way it is (not considering AR or other methods), the new Place Picker API must be implemented. The author would characterise this development in a 'maintenance and upgrades' bracket, as this affects the running of the application itself, and so it is an essential feature.

7.3.4 Offline Mode

As with most applications on the market, having an offline mode provides competitive advantage. It is estimated that in the US at any given time, 15% of smartphone users are using offline applications[76]. This was put down to being in 'dead zones' where there are no online capabilities present. Ireland contains a large amount of these areas, especially in the countryside where signal is notoriously bad, which is arguably where tourists will be using POInt the most. An equally important reason again lies with the target market itself. Even though approximately 71.3% of tourists into Ireland

(2018) were from the EU and Britain, the other 28.7% were from outside this area[77]. This means that these tourists into Ireland would need to be using Wi-Fi, or would be charged for data roaming. As Wi-Fi is rarely available everywhere and data roaming is exceptionally expensive, offline mode would be a welcomed solution. Similarly, when tourists are travelling around the country, they may not have access to charging plugs. Offline mode saves battery life so that tourists can use the application as much as they need. An offline mode would add significant value for the users of POInt for a variety of reasons, and therefore should be implemented as a next step.

7.3.5 iOS Application

Although currently the Android market outweighs the iOS market, as discussed in Section 2.1, Apple still has an immense user base with approximately 218 million iPhones sold in 2018[78]. Therefore, penetrating this market as well as the Android market would be beneficial for POInt to expand their scale and scope. As the current application for Android has been developed in Java; Swift, Objective-C and Objective-C++ can easily be utilized to convert this app to an iOS one[79].

7.4 Closing Statements

In conclusion to this report, the author considers this project to be a triumph. From a technical aspect, the application functions as intended and provides all the necessary features. From a business point of view, the application fills a gap in the market, and provides value to the consumers with ease and efficiency.

The author feels as though this project has also been a large personal success in many ways. From starting this project with close to no experience of Android app development, the author is finishing this project with extensive knowledge of creating an app, as well as more complex skills such as integrating APIs. The author has also developed several soft skills as a result of this project. These include communication skills from weekly update meetings with their supervisor and presentation skills gained by the demonstration and user testing briefing.

The author had the opportunity to showcase knowledge and expertise gained over the last four years of studying BA. Computer Science and Business. This project has been a consolidation of insights from various modules taken by the author, as well as an expression of the skills they attained from them such as researching and report writing.

8. Appendix

Appendix A: Final POInt Screenshots

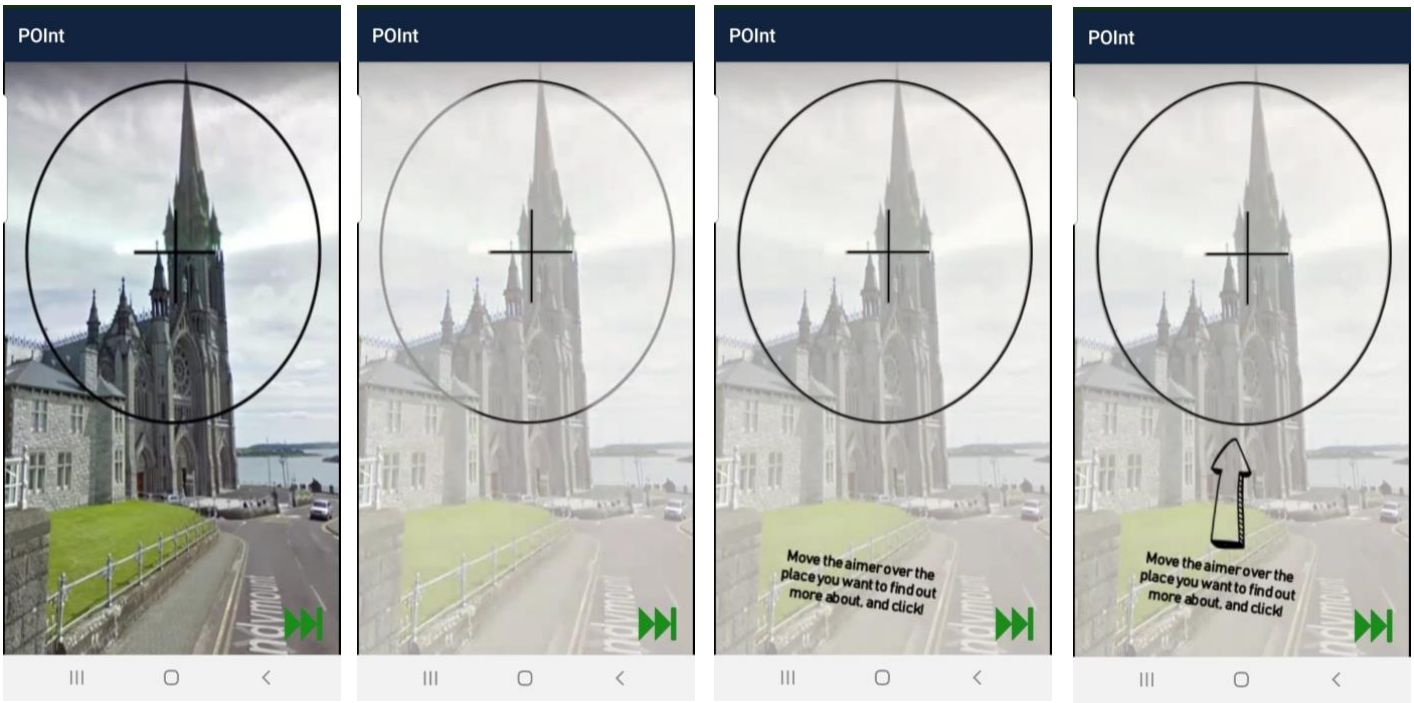


Image A-1 – Splash Screen

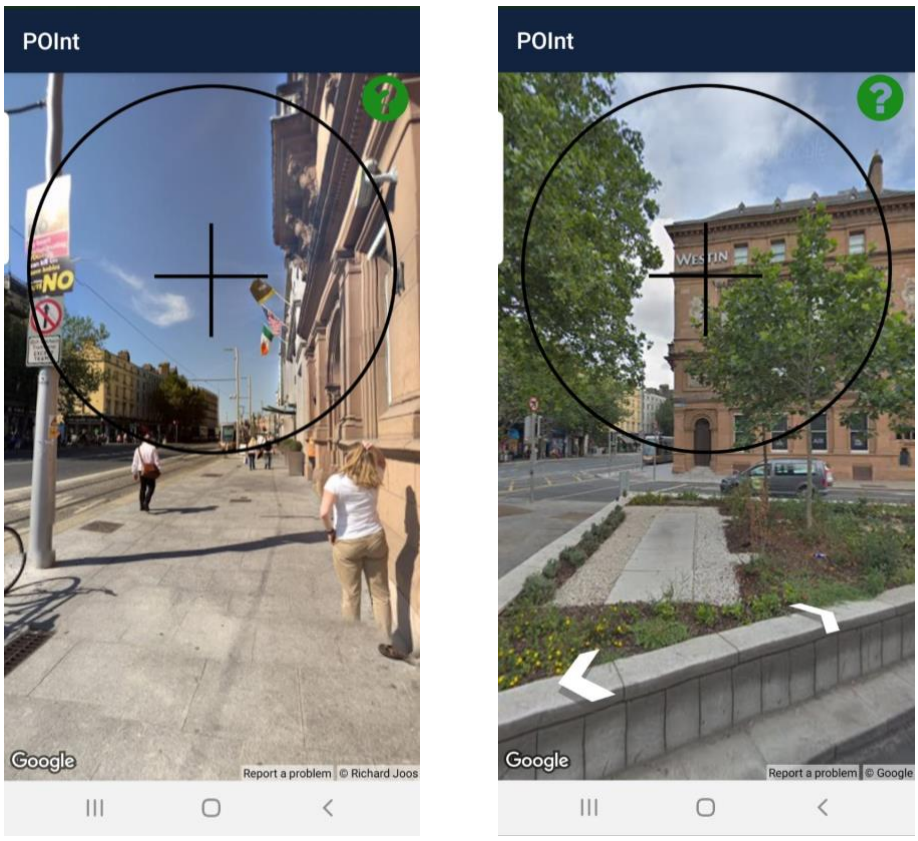


Image A-2 – Navigation Screen (pointed at selected location)

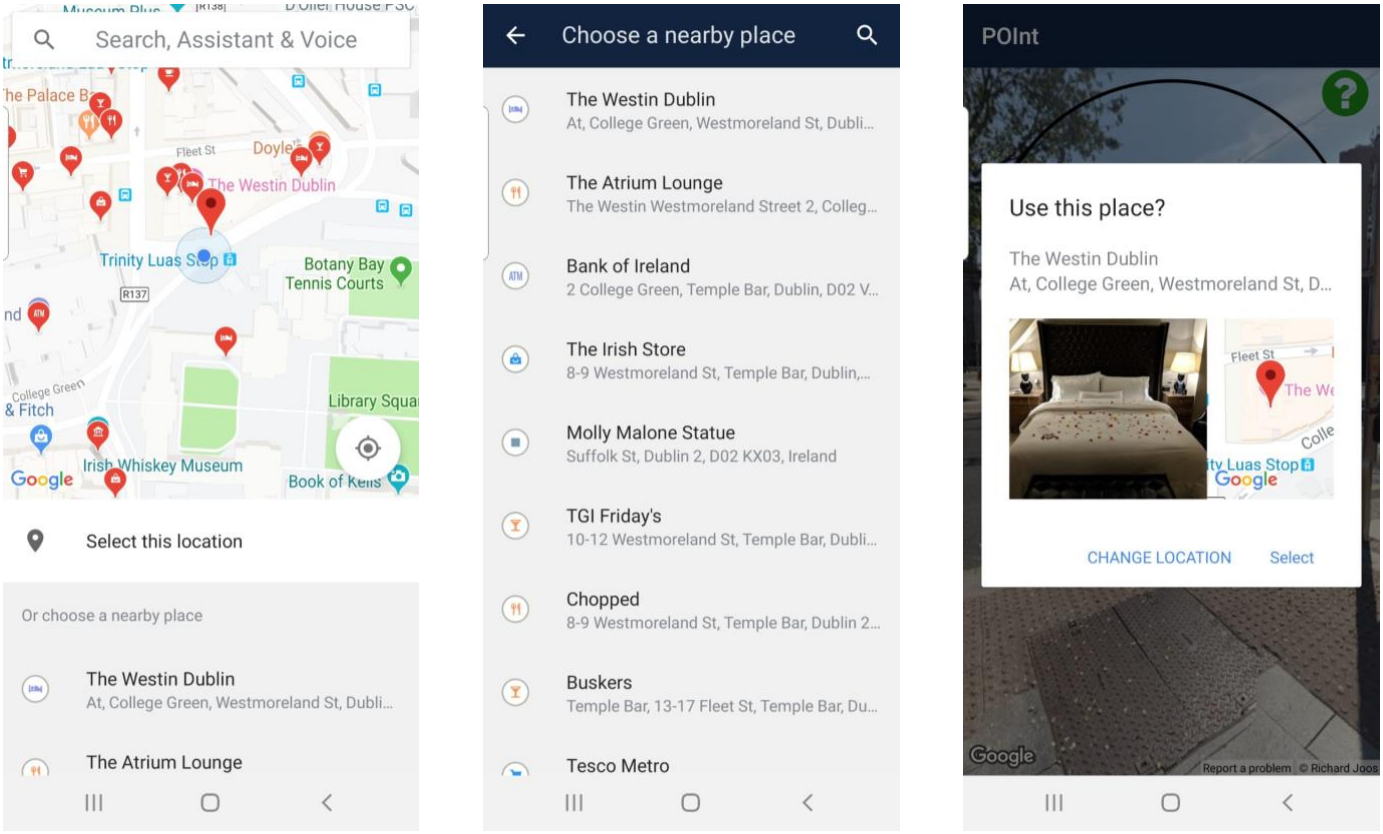


Image A-3 – Place Picker API

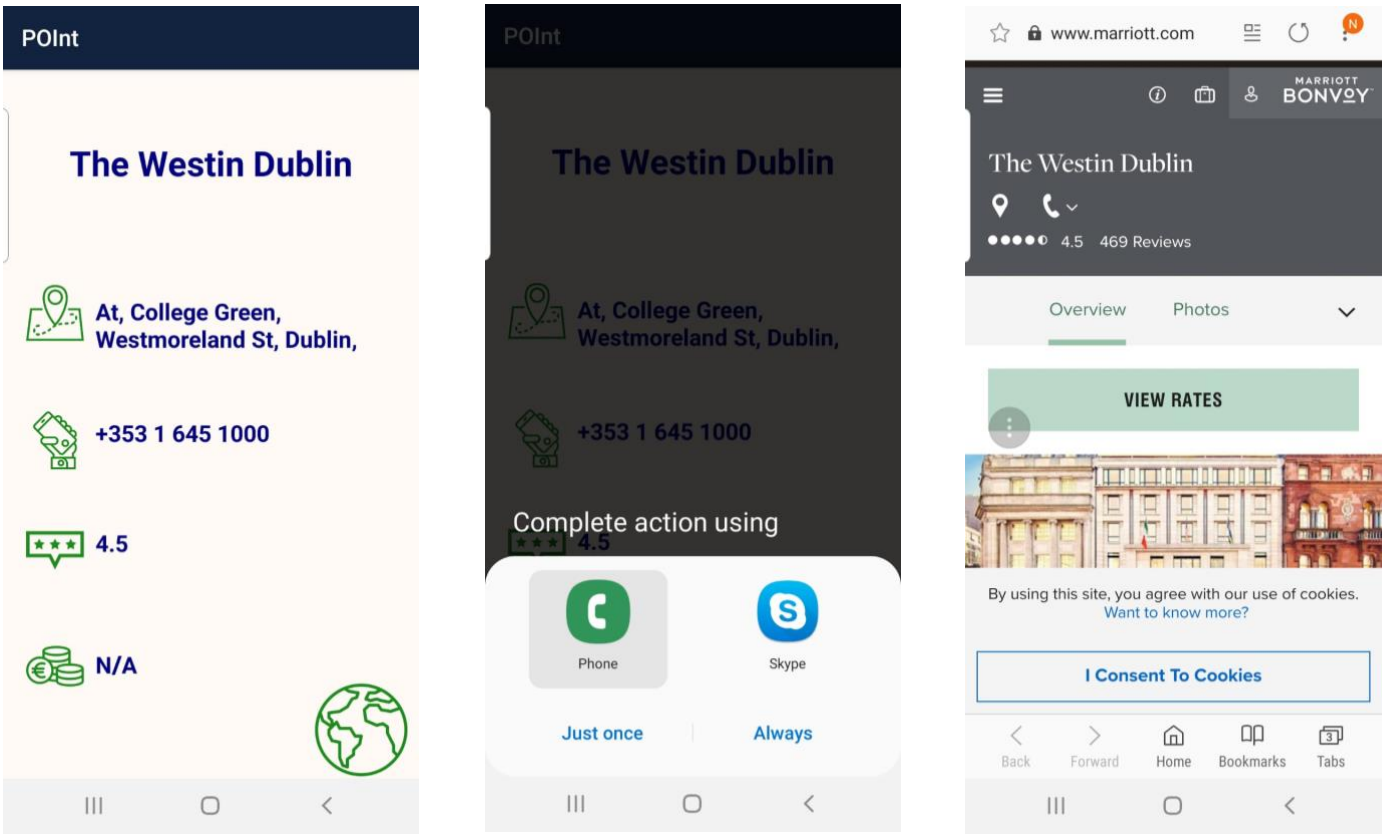


Image A-4 – Information Page.
Call & Website add-ons

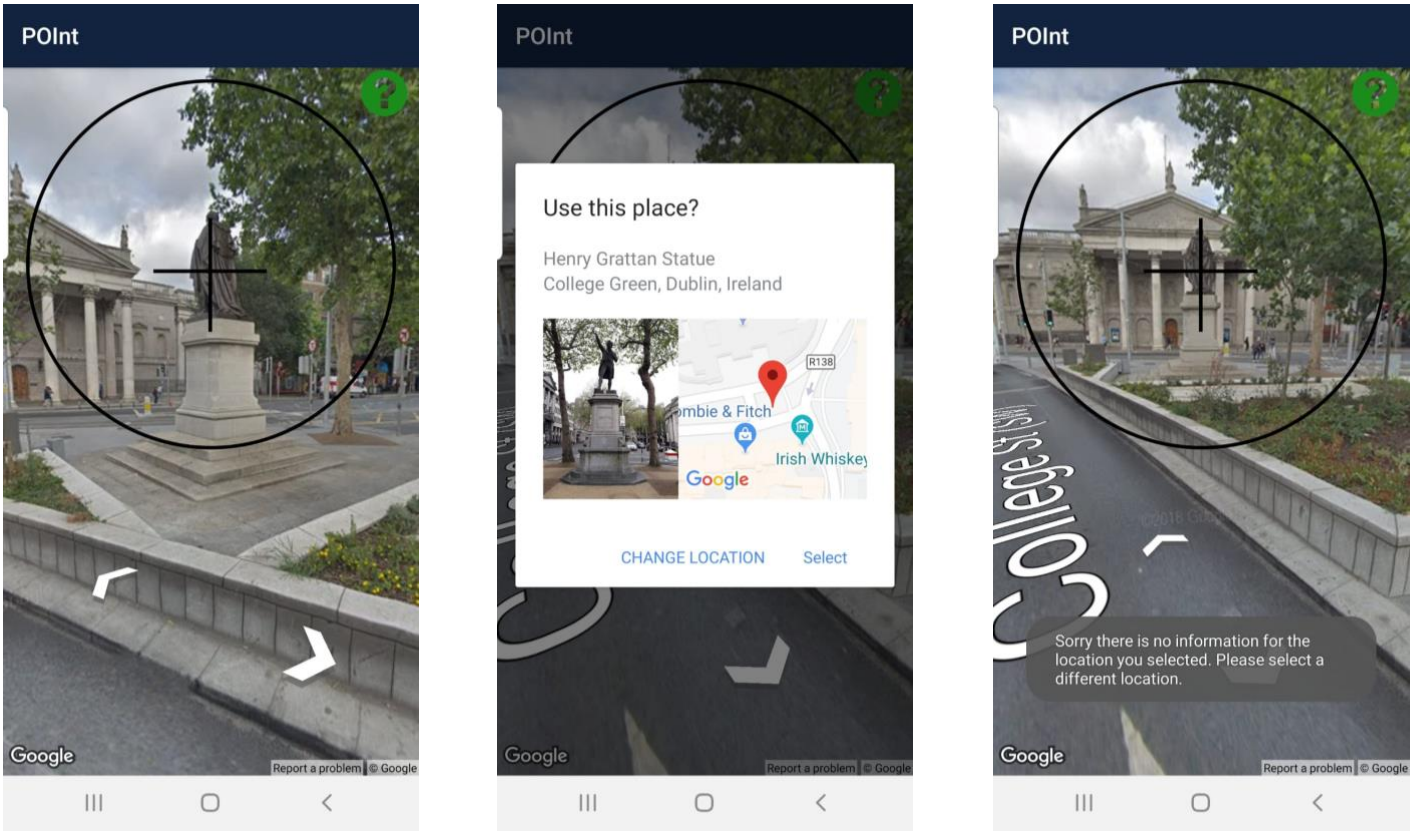


Image A-5 – No Information Found

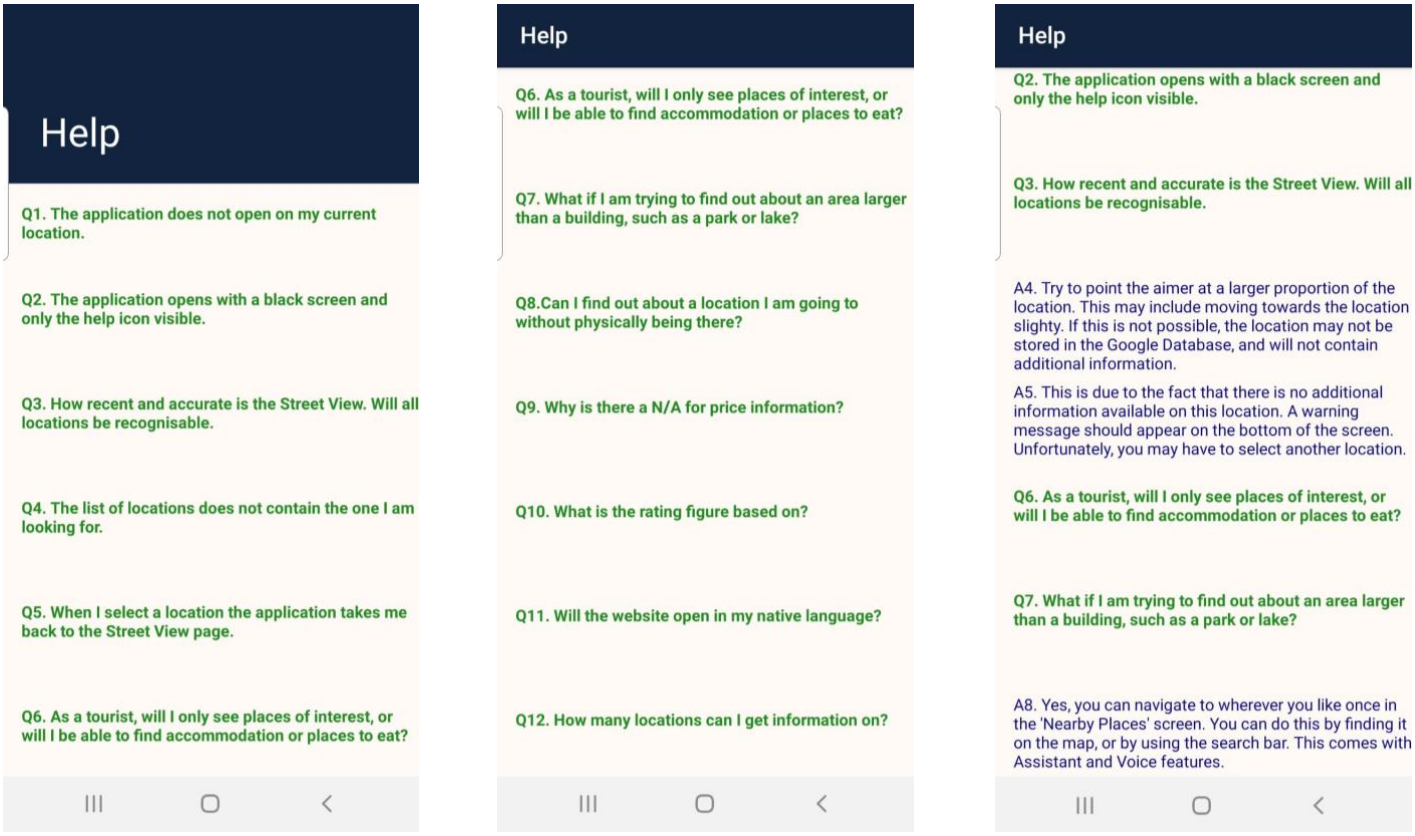


Image A-5 – Help Page – Questions & Mixed

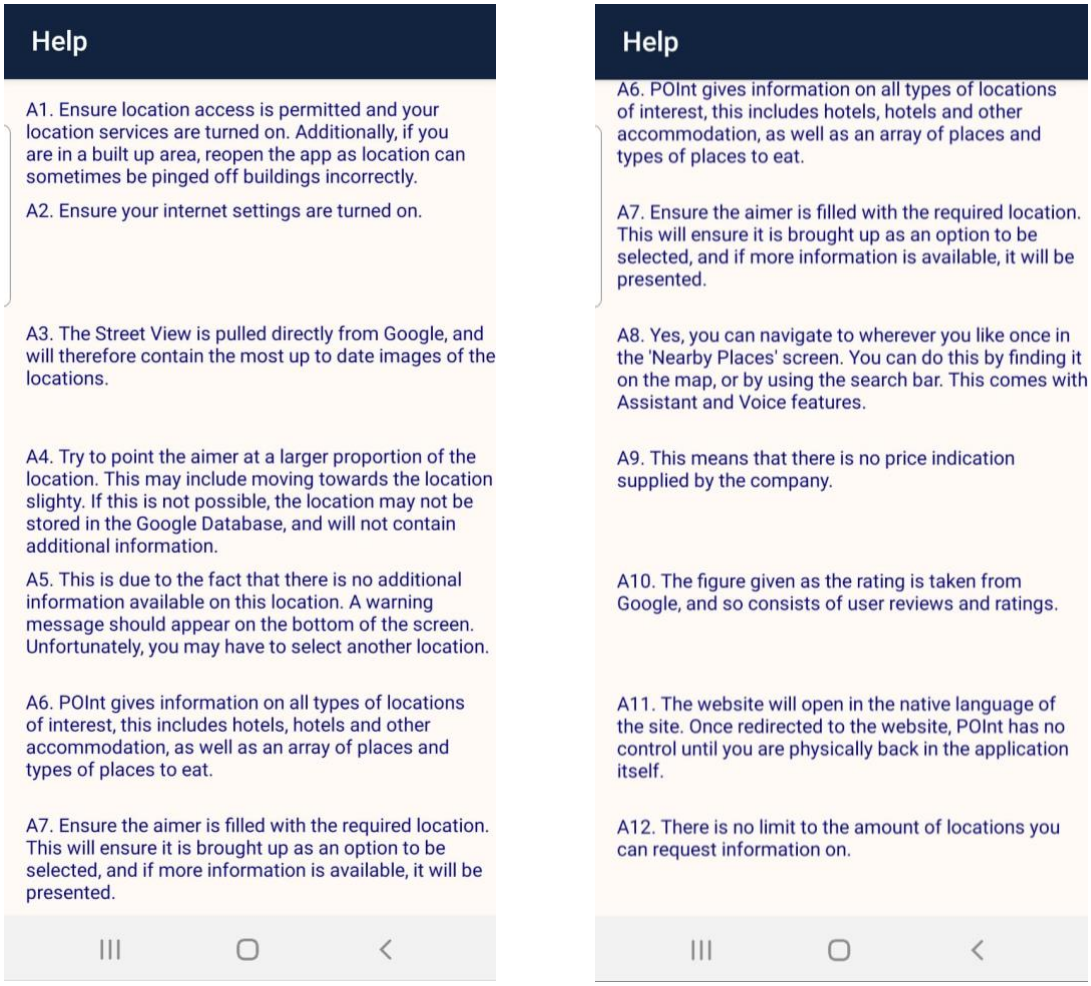


Image A-6 – Help Page – Answers

Appendix B: Ethics Application

RESEARCH PROJECT PROPOSAL

Project Title: POInt - an Android Application that aids tourists around Ireland

Purpose of Project:
This study is being carried out to test the usability of POInt, an Android Application that aids tourists around Ireland identify landmarks and places of interest through the lenses of Google Street View APIs. The study is aimed to gather information about the functionality of the app, as well as the ease of understanding for the user and the fulfilment of the initial requirements and aims of the project. This app is being made for a B.A Computer Science and Business Final Year Project.

Methods and Measurements to be used:
Users will first have the opportunity to test the application, ensuring that they have had enough time to test all functionalities and aspects of the app. When they are satisfied that they have interacted with the application for long enough they will be given the survey. The users will fill out the survey and submit it anonymously. The data in the surveys will be compiled into a secure, encrypted excel spreadsheet. This will then be further analysed for findings and conclusions that can be used in the report. The analysis will include various methods, for example, manipulating the scores of the System Usability Scale (SUS) to allow percentile ranking to take place. Once this report is submitted all data collected will be correctly discarded.

Intended Participants & Recruitment Process:
Participates will include classmates and students enrolled in Computer Science and Business and numerous other degrees at Trinity College Dublin. Participants may also include friends and family of the researcher.
All participants must be over the age of 18, without exception. The researcher will try to recruit participants from a wide range of demographics (gender, age, nationality etc.)
The researcher intends to obtain findings from approximately 20 participants. These participants will be recruited by telephone call or email. To ensure that 20 participants are found, the researcher intends to ask 30+ potential participants, in the understanding that some will decline.

Debriefing Arrangements:
Before starting this study, all participants will be briefed. Each participant will be provided with an information sheet, a consent form, and the survey. They will then be asked to test the app and answer the survey questions. After completion of the desired task in this study, participants will have the opportunity to ask any questions that may arise, or have arisen during the study.

Ethical Considerations:
A potential conflict of interest may arise by making use of personal relationships. This will be eradicated by anonymising the data and encouraging honest answers to the survey.

Legal Considerations:

The researcher has familiarised themselves with the Data Protection Act (even though no personal data will be collected at any time during this study) and College Good Research Practice Guidelines.

INFORMATION SHEET FOR PROSPECTIVE PARTICIPANTS

This study is being carried out in order to test the usability of POInt, an Android Application that aids tourists around Ireland identify landmarks and places of interest through the lenses of Google Street View APIs. The study is aimed to gather information about the functionality of the app, as well as the ease of understanding for the user and the fulfilment of the initial requirements and aims of the project.

I confirm that I will (where relevant):

- Familiarise myself with the Data Protection Act and the College Good Research Practice guidelines.
- Not take any extra information such as audio recordings / pictures / videos without informing participants and obtaining the relevant permissions.
- Provide participants with an information sheet that describes the main procedures.
- Obtain informed consent for participation.
- Ask participants for their consent to be observed.
- Tell participants that their participation is voluntary.
- Tell participants that they may withdraw at any time and for any reason without penalty.
- Give participants the option of omitting questions they do not wish to answer on the survey.
- Tell participants that their data will be treated with full confidentiality and that, when published, it will not be identified as theirs.
- Debrief participants at the end of their participation.
- Verify that participants are 18 years or older and competent to supply consent.
- Declare any potential conflict of interest to participants.
- Inform participants that in the extremely unlikely event that illicit activity is reported to me during the study I will be obliged to report it to appropriate authorities.
- Act in accordance with the information provided (i.e. if I tell participants I will not do something, then I will not do it).

Who is carrying this study out:
This study is being carried out as part of the requirements for a B.A. in Computer Science and Business at Trinity College Dublin. The main researcher is Anya Mangat, and the supervisor for this project is Fergal Shevlin.

Who we are looking for:
We are seeking participants who have no extensive prior knowledge of computer science or Android application development. Ideally, we are seeking participants with little to no knowledge of Dublin and the surrounding tourist attractions, this will enable us to achieve the most accurate results of the functionality of the app. Some of the participants will be friends or family members of the researcher, but will be urged to answer all questions honestly.

The Procedure and Duration:

1. During this study, you will first be asked to use the app in question. This application does not involve entering any personal data, and will only access the user location after the participant has opted in to allow this. The participants will be encouraged to use the app for approximately 10 minutes to allow them to explore all the functions the app provides.
2. Following the testing of the application itself, the participants will be given a printed survey to complete. The survey consists of 20 questions and is estimated to take approximately 15 minutes.

Overall this procedure is estimated to take between 25 – 35 minutes in total.

Risks:
The main risk is associated with this study is the ability of the app to access user location information. This information is not permanently stored on the app and is overwritten when the user moves / choses a different location. No details of user location will be provided on the survey, and therefore the researcher will have no record of the location information given.
For some cases of this study, the app will be tested through an emulator and so user location will be hard coded into the testing script and will not be required.

Another risk linked to this study and the subsequent results is the use of family and friends for the testing. There is potential for bias, which will be eliminated by anonymising the data and strongly urging participants to answer as honestly as possible.

Anonymity and Data Preservation:
Your participation is entirely voluntary, and you can withdraw at any time up until the survey is submitted. As there is no way for us (or anyone else) to identify an individual participant once they submit their data, therefore it will no longer be possible to withdraw once you have submitted the survey. Anonymity will be held to the highest regard in the collecting, analysis and publication of the results of this study. There will be no audio / image / video recordings of this study taken. The surveys will be stored in a secure location and will be correctly discarded once this project has been submitted.

Questions:
Every question in this survey is optional. Feel free to omit a response to any question; however, the researcher would be grateful if all questions are responded to.

Debriefing after Participation:
At the end of the study, participants will have the opportunity to ask any questions that arose throughout the study. All concerns will be addressed, including those surrounding the processing and publication of the results. We will present all participants who request it with a copy of the final report by email.

Preservation of Participant and Third-party Anonymity:
The data will be analysed in order to assess the functionalities and usability of the app in question. We plan to publish the results of our research in a B.A Final Year Project Report. The researcher will do this in a way which does not identify you, or

any other individual participant. No other means of information (audio, imagery, video) are taken during this study.

Inadvertent Discovery of Illicit Activities:
While it is unlikely that illicit activities would be disclosed, if you do so, we would be obliged to report them to the appropriate authorities.

Declarations of Conflicts of Interest:
Due to the selection process of participants, conflict of interests might arise by making use of personal relationships. This will be eradicated by anonymising the data and encouraging honest answers to the survey.

Contact Details:
If you have any queries, feel free to contact Anya Mangat at mangata@tcd.ie or +353863995595.

INFORMED CONSENT FORM

This study is being carried out in order to test the usability of POInt, an Android Application that aids tourists around Ireland identify landmarks and places of interest through the lenses of Google Street View APIs. The study is aimed to gather information about the functionality of the app, as well as the ease of understanding for the user and the fulfilment of the initial requirements and aims of the project.

The Procedure and Duration:

1. You will first be asked to use the app in question. You will be encouraged to use the app for approximately 10 minutes.
2. You will be asked to complete a survey. The survey consists of 20 questions and is estimated to take approximately 15 minutes.

Overall this procedure is estimated to take between 25 – 35 minutes in total.

DECLARATION:

- I am 18 years or older and am competent to provide consent.
- I have read, or had read to me, a document providing information about this research and this consent form. I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and understand the description of the research that is being provided to me.
- I agree that my data is used for scientific purposes and I have no objection that my data is published in scientific publications in a way that does not reveal my identity.
- I understand that if I make illicit activities known, these will be reported to appropriate authorities.
- I understand that I may refuse to answer any question and that I may withdraw at any time before the survey is submitted, without penalty.
- I understand that if the results of the research have been published, or my data has been fully anonymised so that it can no longer be attributed to me, then it will no longer be possible to withdraw.

- I understand that no audio, imagery or video recordings will take place during this study.
- I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights.
- I understand that my participation is fully anonymous and that no personal details about me will be recorded.
- I understand that if I or anyone in my family has a history of epilepsy then I am proceeding at my own risk.
- I have received a copy of this agreement.

By signing this document, I consent to participate in this study, and consent to the data processing necessary to enable my participation and to achieve the research goals of this study.

Participant Name: _____

Participant Signature: _____

Date: _____

Statement of investigator's responsibility:
I have explained the nature and purpose of this research study, the procedures to be undertaken and any risks that may be involved. I have offered to answer any questions and fully answered such questions. I believe that the participant understands my explanation and has freely given informed consent.

Researcher's Contact Details:
Anya Mangat
mangata@tcd.ie
+353863995595

Researcher's Signature: _____

Date: _____

USER TESTING SURVEY

Each of the following questions are optional. Feel free to omit a response to any question; however, the researcher would be grateful all questions were responded to.

All answers on this survey are anonymised.

The researcher would like to thank you for your participation.

Demographic Information:

1. Please tick the box that contains your age:

18-25	26-30	31- 40	41-50	51-60	61+

2. Are you an Irish resident?
Yes No

3. Are you a Dublin resident?
Yes No

4. Do you prefer using apps to get information or obtaining physical information?
Apps Physical

Technical Information:

5. Were you able to open this application without difficulty?
Yes No

6. Did the application fit the screen correctly, with all buttons fully visible?
Yes No

7. Did you experience lag at any time using this application?
Yes No

8. Did the app correctly open on your current location?
Yes No

9. Were you able to navigate using the white arrows to a required destination?
Yes No

10. Did the application successfully open the webpage of the location you selected in your default browser?
Yes No

Usability Information:
Based on the System Usability Scale (SUS).

<u>The scale used for the following questions is as follows:</u>
1 – Strongly Disagree
2 – Disagree
3 – Neither Agree nor Disagree
4 – Agree
5 – Strongly Agree

11. I think that I would like to use this system frequently.

1 2 3 4 5

12. I found the system unnecessarily complex.

1 2 3 4 5

13. I thought the system was easy to use.

1 2 3 4 5

14. I think that I would need the support of a technical person to be able to use this system.

1 2 3 4 5

15. I found the various functions in the system were well integrated.

1 2 3 4 5

16. I thought there was too much inconsistency in this system.

1 2 3 4 5

17. I would imagine that most people would learn to use this system very quickly.

1 2 3 4 5

18. I found the system very cumbersome to use.

1 2 3 4 5

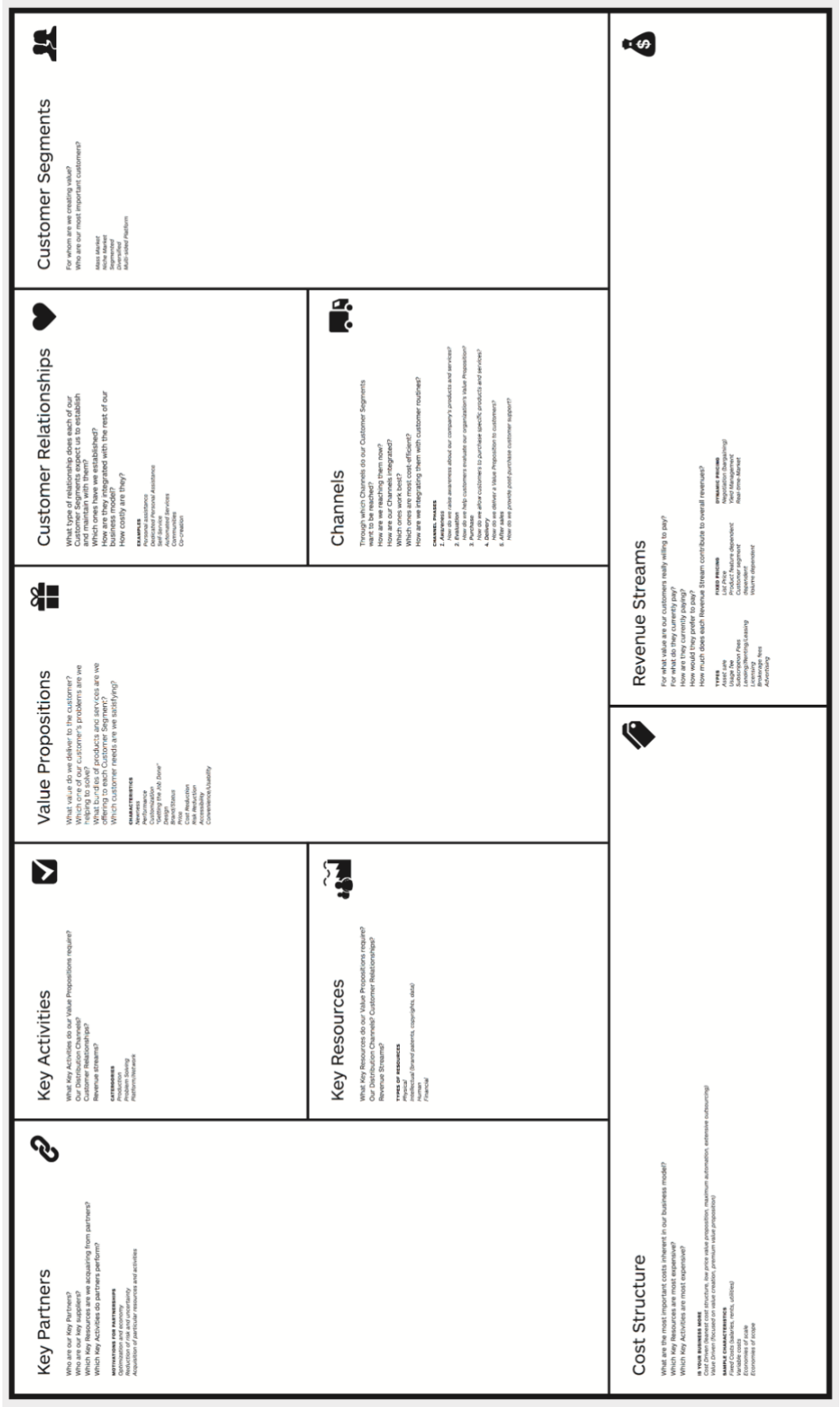
19. I felt confident using the system.

1 2 3 4 5

20. I needed to learn a lot of things before I could get going with this system.

1 2 3 4 5

Appendix C: Business Model Canvas



Appendix D: Resources

YouTube Video Title: Google I/O 2018 keynote in 14 minutes.
https://www.youtube.com/watch?time_continue=592&v=BRUvbiWLwFI

Android Developer Tutorials.
https://developer.android.com/guide

Google Developers Training
https://developers.google.com/training/

TutorialsPoint Training
https://www.tutorialspoint.com/android/android_studio.htm

Flat Icons – Website to create icons.
https://www.flaticon.com

YouTube Video Title: How to Install Android Studio on Mac + Build Your First App in Android Studio
https://www.youtube.com/watch?v=br-1Vuwljd8

YouTube Video Title: Android Studio Tutorials - 45 : Google Maps Android API V2 in Android
https://www.youtube.com/watch?v=sJBlQv6IptQ

YouTube Video Title: How to Implement Google Map in Android Studio | GoogleMap | Android Coding
https://www.youtube.com/watch?v=eiexkzCI8m8

YouTube Video Title: Android capture image from Camera and select image from Gallery
https://www.youtube.com/watch?v=i5UcFAdKe5M&t=194s

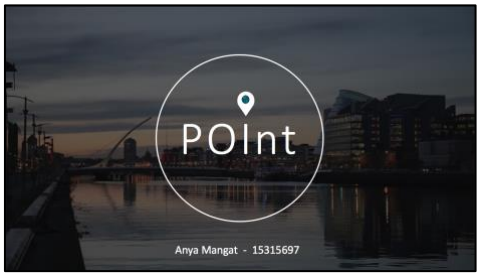
YouTube Video Title: Google Maps Current Location in Android Studio using Google Map - Get Current Location of user
https://www.youtube.com/watch?v=4kk-dYWVNsc&t=14s

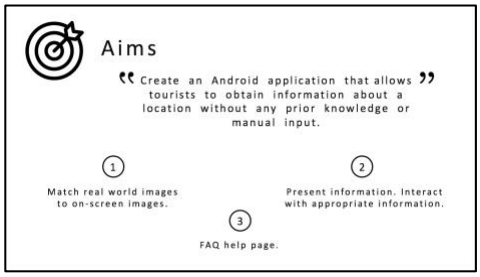
YouTube Video Title: android google map streetview
https://www.youtube.com/watch?v=A3xsGNldYQA

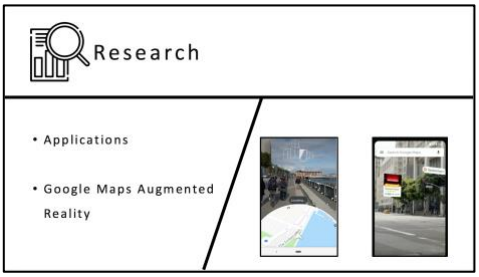
YouTube Video Title: Android Tutorial | Video as a Splash screen | Android Studio
https://www.youtube.com/watch?v=dL9HRcqBJ-s

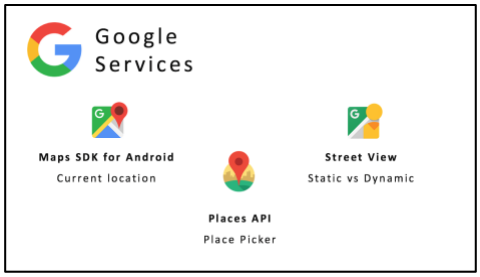
Appendix E: Demonstration Slides

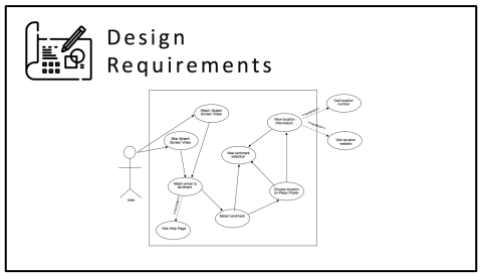
The background colour of these slides has been removed for better quality printing. The content remains the same.

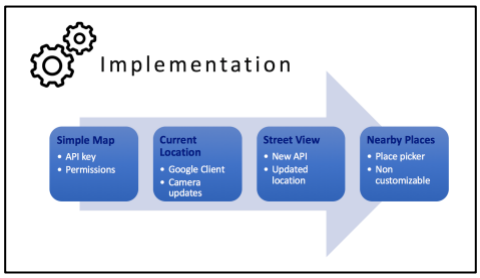


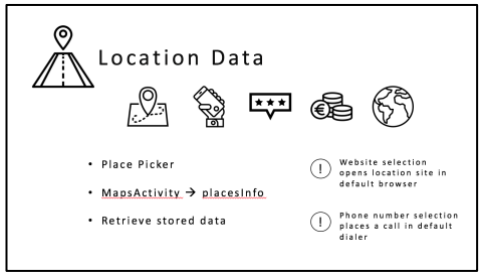


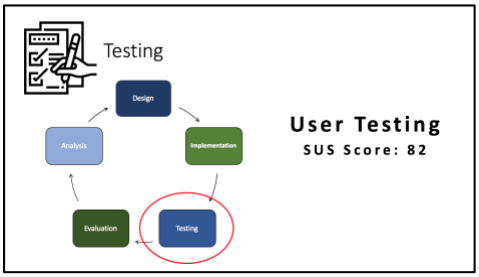


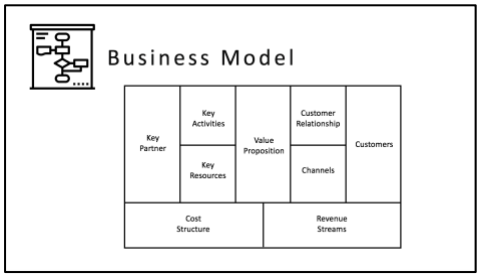


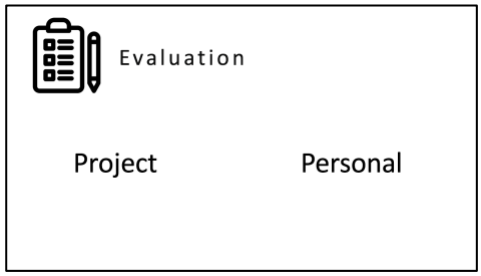


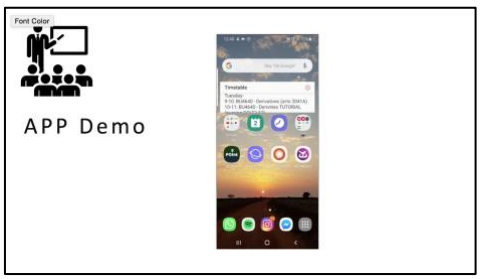


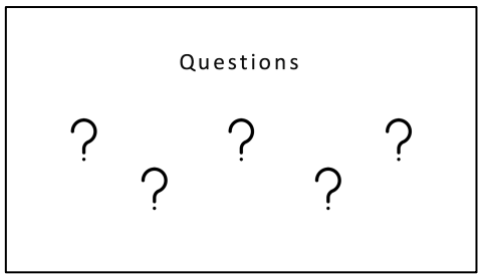












9. References

[1] Loftus Hall. *Homepage.* [online] Available at: https://www.loftushall.ie [Accessed 12 Feb 2019]

[2] Breaking News.ie. *Record year for Tourism Ireland with 11.2m visitors in 2018.* [online] Available at: https://www.breakingnews.ie/business/record-year-for-tourism-ireland-with-112m-visitors-in-2018-892771.html [Accessed 12 Jan 2019]

[3] Tourism Ireland. *Island of Ireland, Overseas Tourism Performance, Facts & Figures.* [online] Available at: https://www.tourismireland.com/TourismIreland/media/Tourism-Ireland/Press%20Releases/TI_2017_Facts-Figures.pdf?ext=.pdf [Accessed 12 Jan 2019]

[4] O'Riordan, S (2017). *Visitor attractions entice record numbers*. [online] Available at: https://www.irishexaminer.com/ireland/visitor-attractions-entice-record-numbers-451800.html [Accessed 12 Jan 2019]

[5] O'Riordan, S (2017). *Visitor attractions entice record numbers.*

[6] Deloitte. *Mobile Consumer Survey 2018: The Irish cut.* [online] Available at: https://www2.deloitte.com/ie/en/pages/technology-media-and-telecommunications/articles/global-mobile-consumer-survey0.html [Accessed 12 Jan 2019]

[7] Slade, J. (2018). Study Finds That *Americans Check Their Phones 80 Times A Day While On Vacation.* [online] Available at: https://wixx.com/blogs/chronicles-of-slade/1740/study-finds-that-americans-check-their-phones-80-times-a-day-on-vacation [Accessed 12 Jan 2019]

[8] Jones, C. (2018). *Apple's iOS Loyalty Rate Is Lower Than Google's Android, But Apple May Steal More Users Each Year.* [online] Available at: https://www.forbes.com/sites/chuckjones/2018/03/10/apples-ios-loyalty-rate-is-lower-than-googles-android-but-apple-may-steal-more-users-each-year/#4c05e61b68a8 [Accessed 12 Jan 2019]

[9] Google Play Store (2018), *Google Services*. [online] Available at: https://play.google.com/store [Accessed 16 Jan 2019]

[10] Gartenberg, C. (2018), *Google Maps is getting augmented reality directions and recommendation feature*. [online] Available at: https://www.theverge.com/2018/5/8/17332480/google-maps-augmented-reality-directions-walking-ar-street-view-personalized-recommendations-voting [Accessed 22 Feb 2019]

[11] Kumparak, G. (2019), *Hands-on with an Alpha build of Google Maps' Augmented Reality mode.* [online] Available at: https://techcrunch.com/2019/02/11/hands-on-with-an-alpha-build-of-google-maps-augmented-reality-mode/?guccounter=1&guce_referrer_us=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_cs=nCtZEHfv9Y4NaciR75QbBA [Accessed 4 March 2019]

[12] Liptak, A. (2019), *Google is letting some users test its AR navigation feature for Google Maps.* [online] Available at: https://www.theverge.com/2019/2/10/18219325/google-maps-augmented-reality-ar-feature-app-prototype-test [Accessed 4 March 2019]

[13] Schroeder, S. (2019) *Google's augmented reality Maps are live for some users.* [online] Available at: https://mashable.com/article/google-ar-maps/?europe=true [Accessed 4 March 2019]

[14] Google Maps Platform. *Google Maps Platform Documentation.* [online] Available at: https://developers.google.com/maps/documentation/ [Accessed 16 Oct 2018]

[15] Street View Static API*. Developer Guide* [online] Available at: https://developers.google.com/maps/documentation/streetview/intro [Accessed 15 Dec 2018]

[16] Street View Static API. *Get API Key and Signature Guide.* [online] Available at: https://developers.google.com/maps/documentation/streetview/get-api-key [Accessed 15 Dec 2018]

[17] Scaled Agile. *Non-functional Requirements.* [online] Available at: https://www.scaledagileframework.com/nonfunctional-requirements/ [Accessed 21 Dec 2018]

[18] Neilson and Molich. (1990). *User Interface Design Guidelines.* [online] Available at: https://www.interaction-design.org/literature/article/user-interface-design-guidelines-10-rules-of-thumb [Accessed 4 Dec 2018]

[19] Porter, J. (2019). *Principles of User Interface Design.* [online] Available at: http://bokardo.com/principles-of-user-interface-design/ [Accessed 7 Jan 2019]

[20] Soegaard, M. (2019). *Visual Hierarchy: Organizing content to follow natural eye movement patterns.* [online] Available at: https://www.interaction-design.org/literature/article/visual-hierarchy-organizing-content-to-follow-natural-eye-movement-patterns [Accessed 31 March 2019]

[21] Developers. *Toasts overview.* [online] Available at: https://developer.android.com/guide/topics/ui/notifiers/toasts [Accessed 12 Dec 2018]

[22] Soegaard, M. (2019). *Visual Hierarchy: Organizing content to follow natural eye movement patterns.*

[23] Tubik Studio, (2018). *Negative Space in UI Design: Tips and Best Practices.* [online] Available at: https://uxplanet.org/negative-space-in-ui-design-tips-and-best-practices-98311cb2ad16 [Accessed 7 Dec 2018]

[24] Cimet, L. (2016). *The Overlooked Importance of App Icons.* [online] Available at: https://medium.com/hatch-feed/the-overlooked-importance-of-app-icons-c87271cdaf5f [Accessed 22 Dec 2018]

[25] Johnson, J. (2018). *10 Tips for Designing Logos That Don't Suck.* [online] Available at: https://designshack.net/articles/inspiration/10-tips-for-designing-logos-that-dont-suck/ [Accessed 5 Dec 2018]

[26] Nelson, N. (2015). *The Power of Colour in App Design.* [online] Available at: https://medium.com/@nicknelo/why-use-colour-branding-in-apps-a95deba49da [Accessed 8 Dec 2018]

[27] Agile Alliance. *Agile 101.* [online] Available at: https://www.agilealliance.org/agile101/ [Accessed 12 Dec 2018]

[28] Half, R. (2017). *6 Basic SDLC Methodologies: Which One is Best? [*online] Available at: https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best [Accessed 7 Dec 2018]

[29] Sacolick, I. (2018). *What is agile methodology? Modern software development explained.* [online] Available at: https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html [Accessed 13 Dec 2018]

[30] Directions API. *Get API Key*. [online] Available at: https://developers.google.com/maps/documentation/directions/get-api-key [Accessed 23 Oct 2018]

[31] Maps SDK for Android. *Get Started.* [online] Available at: https://developers.google.com/maps/documentation/android-sdk/start [Accessed 25 Oct 2018]

[32] Android Developers. *App Manifest Overview.* [online] Available at: https://developer.android.com/guide/topics/manifest/manifest-intro [Accessed 25 Oct 2018]

[33] Android Developers. *Manifest.permission.* [online] Available at: https://developer.android.com/reference/android/Manifest.permission [Accessed 25 Oct 2018]

[34] Maps SDK for Android. *Location Data.* [online] Available at: https://developers.google.com/maps/documentation/android-sdk/location [Accessed 30 Oct 2018]

[35] Google APIs for Android. *CameraUpdateFactory.* [online] Available at: https://developers.google.com/android/reference/com/google/android/gms/maps/CameraUpdateFactory [Accessed 1 Nov 2018]

[36] Google APIs for Android. *GoogleMap.* [online] Available at: https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap [Accessed 6 Nov 2018]

[37] Android Developers. *Geocoder.* [online] Available at: https://developer.android.com/reference/android/location/Geocoder [Accessed 6 Nov 2018]

[38] Android Developers. *Address.* [online] Available at: https://developer.android.com/reference/android/location/Address.html [Accessed 6 Nov 2018]

[39] Android Developers. *StreetViewPanoramaFragment.* [online] Available at: ftp://117.239.68.101/Android/sdk/extras/google/google_play_services/docs/reference/com/google/android/gms/maps/StreetViewPanoramaFragment.html#getStreetViewPanoramaAsync [Accessed 24 Nov 2018]

[40] Google APIs for Android. *OnStreetViewViewPanoramaReadyCallback.* [online] Available at: https://developers.google.com/android/reference/com/google/android/gms/maps/OnStreetViewPanoramaReadyCallback [Accessed 24 Nov 2018]

[41] Places SDK for Android. *Overview.* [online] Available at: https://developers.google.com/places/android-sdk/intro [Accessed 27 Nov 2018]

[42] Android Developers. *Getting a Result from an Activity.* [online] Available at: https://developer.android.com/training/basics/intents/result [Accessed 12 Dec 2018]

[43] Places SDK for Android. *Place Picker.* [online] Available at: https://developers.google.com/places/android-sdk/placepicker [Accessed 14 Dec 2018]

[44] Google Issue Tracker (2015). *Filter Place Picker by PlaceType.* [online] Available at: https://issuetracker.google.com/issues/35826944 [Accessed 19 Dec 2018]

[45] Developers. *FrameLayout.* [online] Available at: https://developer.android.com/reference/android/widget/FrameLayout [Accessed 21 Dec 2018]

[46] Developers. *Intent.* [online] Available at: https://developer.android.com/reference/android/content/Intent [Accessed 21 Dec 2018]

[47] Developers. *Intent.*

[48] Developers. *ScrollView.* [online] Available at: https://developer.android.com/reference/android/widget/ScrollView [Accessed 10 Feb 2019]

[49] Smith, C. (2008). *Cell Phone Triangulation Accuracy Is All Over The Map.* [online] Available at: https://searchengineland.com/cell-phone-triangulation-accuracy-is-all-over-the-map-14790 [Accessed 18 March 2019]

[50] Google APIs for Android. *LocationRequest.* [online] Available at: https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest [Accessed 13 Nov 2018]

[51] Usability.gov. *System Usability Scale.* [online] Available at: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html [Accessed 18 Dec 2018]

[52] MeasuringU. *Measuring Usability with the System Usability Scale (SUS).* [online] Available at: https://measuringu.com/sus/ [Accessed 17 March 2019]

[53] Bangor, Kortu, and Miller, (2009). *Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale.* Vol. 4, Issue 3, May 2009, pp. 114-123

[54] Greenwalk, T. (2012). *Business Model Canvas: A Simple Tool For Designing Innovative Business Models.* [online] Available at: https://www.forbes.com/sites/tedgreenwald/2012/01/31/business-model-canvas-a-simple-tool-for-designing-innovative-business-models/#722525b416a7 [Accessed 22 March 2019]

[55] Cleverism. *Business Model Canvas: A Complete Guide.* [online] Available at: https://www.cleverism.com/business-model-canvas-complete-guide/ [Accessed 15 March 2019]

[56] Tourism Ireland. *Homepage.* [online] Available at: https://www.tourismireland.com [Accessed 23 March 2019]

[57] Fáilte Ireland. *Homepage.* [online] Available at: http://www.failteireland.ie [Accessed 23 March 2019]

[58] Department of Transport, Tourism and Sport. *Background.* [online] Available at: http://www.dttas.ie/tourism/agencies/english/national-tourism-development-authority-fáilte-ireland [Accessed 23 March 2019]

[59] Entrepreneur Europe. *Unique Selling Proposition (USP).* [online] Available at: https://www.entrepreneur.com/encyclopedia/unique-selling-proposition-usp [Accessed 25 March 2019]

[60] Ryanair. *Homepage.* [online] Available at: https://www.ryanair.com/ie/en/ [Accessed 23 March 2019]

[61]AdWorld, (2018). *Online Advertising Grows to €491, in 2017 But Rate of Growth Slows).* [online] Available at: https://www.adworld.ie/2018/04/13/online-advertising-grows-e491m-2017-rate-growth-slows/ [Accessed 22 March 2019]

[62] Vincos Blog. *World Map of Social Media Networks.* [online] Available at: https://vincos.it/world-map-of-social-networks/ [Accessed 24 March 2019]

[63] Mackenzie, T. (2012). *App store fees, percentages, and payouts: What developers need to know.* [online] Available at: https://www.techrepublic.com/blog/software-engineer/app-store-fees-percentages-and-payouts-what-developers-need-to-know/ [Accessed 22 March 2019]

[64] Orlowski, A. (2018). *A decade on, Apple and Google's 30% app store cut looks pretty cheesy.* [online] Available at: https://www.theregister.co.uk/2018/08/29/app_store_duopoly_30_per_cent [Accessed 22 March 2019]

[65] Google Cloud. *Pricing for Maps, Routes and Places.* [online] Available at: https://cloud.google.com/maps-platform/pricing/sheet/ [Accessed 10 Dec 2018]

[66] Castletown. *Homepage.* [online] Available at: http://castletown.ie [Accessed 10 Dec 2018]

[67] SpaceO Technologies, (2018). *How Do Free Apps Make Money? 11 Proven & Popular App Monetization Methods.* [online] Available at: https://www.spaceotechnologies.com/how-do-free-apps-make-money/ [Accessed 24 March 2019]

[68] Aprofita. (2018). *6 Types of Mobile Ads That Will Rock Your App.* [online] Available at: https://medium.com/@aprofita_co/6-types-of-mobile-ads-that-will-rock-your-app-cda134a5f98c [Accessed 22 March 2019]

[69] Think Mobiles. *How Do Free Apps Make Money on Android and iOS in 2019.* [online] Available at: https://thinkmobiles.com/blog/how-do-free-apps-make-money/ [Accessed 24 March 2019]

[70] Birch, J. (2015) *Exploring Google Play Services: Place Picker & Autocomplete.* [online] Available at: https://medium.com/exploring-android/exploring-play-services-place-picker-autocomplete-150809f739fe [Accessed 20 Dec 2018]

[71] Google Maps Platform. *Place Types.* [online] Available at: https://developers.google.com/places/web-service/supported_types [Accessed 22 Dec 2018]

[72] Google Maps Platform. *Migrating to the New Places SDK Client.* [online] Available at: https://developers.google.com/places/android-sdk/client-migration [Accessed 3 March 2019]

[73] Medenica, Kun et al. (2018). *Augmented Reality vs. Street Views: A Driving Simulator Study Comparing Two Emerging Navigation Aids.* [online] Available at: http://www.oskar.easyjapanese.org/doc/fp495-medenica.pdf [Accessed 25 March 2019]

[74] Think Mobiles. *What is Augmented Reality (AR) and How does it work.* [online] Available at: https://thinkmobiles.com/blog/what-is-augmented-reality/ [Accessed 25 March 2019]

[75] Jansen, M. (2018). *The best augmented-reality apps for Android and iOS.* [online] Available at: https://www.digitaltrends.com/mobile/best-augmented-reality-apps/ [Accessed 25 March 2019]

[76] Shah, S. (2018). *Why you should develop an offline mobile app.* [online] Available at: https://www.digitaldoughnut.com/articles/2018/february/why-you-should-develop-an-offline-mobile-app [Accessed 25 March 2019]

[77] Central Statistics Office. (2018). *Overseas Travel.* [online] Available at: https://www.cso.ie/en/releasesandpublications/er/ot/overseastraveldecember2018/ [Accessed 25 March 2019]

[78] Statistica (2018). *Global Apple iPhones sales from 3rd quarter 2007 to 4th quarter 2018 (in million units).* [online] Available at: https://www.statista.com/statistics/263401/global-apple-iphone-sales-since-3rd-quarter-2007/ [Accessed 25 March 2019]

[79] Bohun, O. (2018). *How to convert android app to iOS with minimum loss and maximum profit).* [online] Available at: https://brainbeanapps.com/best-practices/how-to-convert-android-app-to-ios-with-minimum-loss-and-maximum-profit/ [Accessed 25 March 2019]