



**DUBLIN INSTITUTE
of TECHNOLOGY**
Institiúid Teicneolaíochta Bhaile Átha Cliath

Geotagged Photo Sharing Mobile Application

Final Year Project Report

**DT228
BSc in Computer Science**

Patrick Cull

Damian Bourke

School of Computing

Dublin Institute of Technology

26th March 2015

Abstract

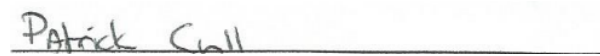
Social media usage and photo sharing has increased drastically in recent years. A lot of young adults and teenagers use social media applications almost every day, and it has become a major part of their lives. With this in mind, there is an opportunity for social media applications to become widely-used and successful.

The goal of this project was to develop a cross-platform social media application which allows users to browse and upload photos to a map. The application will allow users to upload photos on to the map, based on where the photo was taken. This will create a new an interactive way for users to share photos with their friends, and see different areas of the world through photos taken by other users.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

A handwritten signature in dark ink, appearing to read "Patrick Cull", is written over a horizontal line.

Patrick Cull

26th March 2015

Acknowledgements

I'd like to thank my project supervisor, Damian Bourke, for all the help he has provided throughout the duration of this project. He gave me plenty of ideas and encouragement, which gave me the drive to keep improving.

I also want to thank my friends, who helped with the testing and implementation of the project.

Finally, I want to thank my parents, who have worked hard to put me through college for the last four years. I wouldn't have been able to do it without them.

Table of Contents

Abstract.....	2
Declaration.....	3
Acknowledgements.....	4
Table of Figures.....	8
1. Introduction, Project Aims, and Background.	9
1.1 Introduction.....	9
1.2 Project Aim.....	9
1.3 Background Research.....	9
1.4 Aims and Objectives.....	14
1.4.1 Priority Tasks.....	14
1.4.2 Application Functionality.....	14
1.5 Timeline.....	15
1.6 Chapter Walkthrough.....	17
1.7 Conclusion.....	18
2. Research and Evaluation	19
2.1 Introduction.....	19
2.2 Heuristics.....	19
2.3 Existing Solutions.....	19
2.2.1 Snapchat	20
2.2.2 Whisper.....	20
2.2.3 PhotoMap	21
2.4 Heuristic Review of Existing Systems.	22
2.5 Summary of Findings.....	25
2.6 User Requirements.....	26
2.7 Survey Results	27
2.8 Conclusion	27
3. Technology Used	27
3.1 Introduction.....	27
3.2 Presentation Tier.....	28
3.2.1 Core Technology.....	28
3.2.2 HTML5 Technologies	30
3.2.3 Mapping Technologies.....	32

3.3 Logic Tier	34
3.3.1 PHP.....	34
3.3.2 Ruby on Rails	34
3.4 Data Tier	34
3.4.1 MySQL.....	34
3.4.2 PostGIS and PostgreSQL	35
3.5 Summary	36
3.6 Conclusion	37
4. Design	38
4.1 Introduction.....	38
4.2 Methodology	38
4.2.1 Rapid Application Development.....	38
4.2.2 The Spiral Development Model.....	39
4.2.3 Chosen Methodology	39
4.3 System Architecture	40
4.4 Diagrams.....	41
4.4.1 Entity Relationship Diagram (ERD)	41
4.4.2 Use Case Diagram	42
5. Implementation	44
5.1 Introduction.....	44
5.2 Development Environment	44
5.3 Developing the Application	46
5.3.1 Dashboard – The Map Screen	47
5.3.2 Camera – Adding Upload Functionality.....	48
5.3.3 Friends and Groups Screen.....	51
5.3.4 User Profile Screen	55
5.4 Visual Design	55
5.5 Project Structure	56
5.5.1 Core Application Structure	56
5.5.2 Leaflet Map Code.....	57
6. Testing and Evaluation	59
5.1 Introduction.....	59
5.2 White Box Testing	59

5.1.1 Database Entries.....	59
5.1.2 Correct Information Returned from Database	61
5.1.3 Ensure Images are Uploaded Correctly	61
5.3 Black Box Testing.....	63
5.2.1 Fundamental Functionality.....	63
5.2.2 Other Functionality.....	64
5.4 Nielsen’s Heuristics	65
5.5 Unit Testing	66
7. Conclusion	67
7.1 Intro.....	67
7.2 Summary of Findings.....	67
7.3 Aims and Objectives	70
7.4 Future Work	70
7.5 Personal Statement.....	72
8. References	74
9. Appendix.....	77
Appendix A: Survey Monkey Results.....	77

Table of Figures

Figure 1- Usage of social media sites [1].	10
Figure 2- Increase in photo sharing online [3]	11
Figure 3 - Age of Image Sharing online. [3]	11
Figure 4 - Smartphone Ownership amongst adults [5]	12
Figure 5 - Smartphone Ownership by Age [5]	12
Figure 6- Mobile Platform Usage [5]	13
Figure 7 - Project Timeline	15
Figure 8 - Snapchat Screen	20
Figure 9 - Whisper Screen	21
Figure 10 - PhotoMap Screen	22
Figure 11 - System Architecture	28
Figure 12 - How Cordova Works [12]	31
Figure 13 - Ionic Template	31
Figure 14 - Architecture with Technologies Included	36
Figure 15- Rapid Application Development [20]	38
Figure 16 - Spiral Development [28]	39
Figure 17 - System Architecture	40
Figure 18 - Entity Relationship Diagram	41
Figure 19 - Use Case Diagrams	42
Figure 20- Ionic CLI	44
Figure 21 - Ionic Browser Page	45
Figure 22 - Ionic Lab	45
Figure 23 - phpMyAdmin	46
Figure 24 - Cordova FileTransfer Documentation [31]	48
Figure 25- Choose Image Screen	50
Figure 26 - Upload Screen	51
Figure 27 - Modal Window	53
Figure 28 - Friend Photos	54
Figure 29 - Design Change	55
Figure 30 - Core File Structure	56
Figure 31 - www folder structure	56
Figure 32 - Map Folder Structure	57
Figure 33 - Image Directory	62
Figure 34 - Pins Table	63

1. Introduction, Project Aims, and Background.

1.1 Introduction

As social media and mobile technology becomes more and more used in an average person's daily life, so too does the demand for social media applications. Instagram, Facebook, Snapchat and Twitter are becoming the primary means of communication for many people, and so it is clear that social media is a major part of the modern world.

With this in mind, there is an opportunity for a social media application to become widely used and successful. As more people use social media as a means to keep up to date with what their friends are doing, having an application which lets users see pictures of places their friends have been displayed on a map could be a new way to keep up to date with friends.

This kind of application also provides a means for users to explore different places of the world. It would allow people to view photos from different countries, and see places through the eyes of a local.

In this chapter, the aims and objectives of the project are discussed in detail. This chapter also contains the extensive background research which went into the project, as well as the project's timeline. The following section describes the aim of the project.

1.2 Project Aim

The goal is to develop a cross platform social media application which allows users to browse and upload photos to a map. The application will allow users to upload photos on to the map, based on where the photo was taken. The photos are attached to pins, which a user can click on to view the image associated with that pin.

Other users can then browse these photos on the map, and rate the photo. The map will display the top photos in the area, based on the ratings from users.

Users will also be able to add friends in the application, and create groups. Members of the group can then view photos from that group on the map, and add photos to the group. The user can decide to make a photo publically available, or just viewable by friends and/or a specific group. Pins created by friends will be displayed in a different colour, as will group photos. Users can also filter the pins, and choose to display only friend's photos on the map.

With this idea in mind, research was conducted into how social media and mobile applications have evolved over recent years. This background research is discussed in detail in the following section.

1.3 Background Research

In order to understand the likelihood of the proposed application usage, it is important to conduct background research. This will give a clearer understanding of why the proposed application was chosen, and will provide an estimate of the size of the target market.

In this section, we will look at the background aspects of the proposed application, such as the increase social media usage and the popularity of smartphones in recent years.

1.3.1 Social Media

The surge in popularity of social media has changed the way people interact with each other immensely in recent years. It makes it easier for friends to contact each other, and keep up to date what one another are doing. Research conducted by Pew Research shows that in 2013 73% of adults in the US who use the internet use social media. [1] Social media is especially popular among teenagers and people in their twenties, with another report done by Pew Research in 2012 indicating 94% of teenagers use social media. [2]

Figure 1 is a graph which shows how the usage of social media sites has increased between 2012 and 2013, and demonstrates the percentage of users which use each social media site.

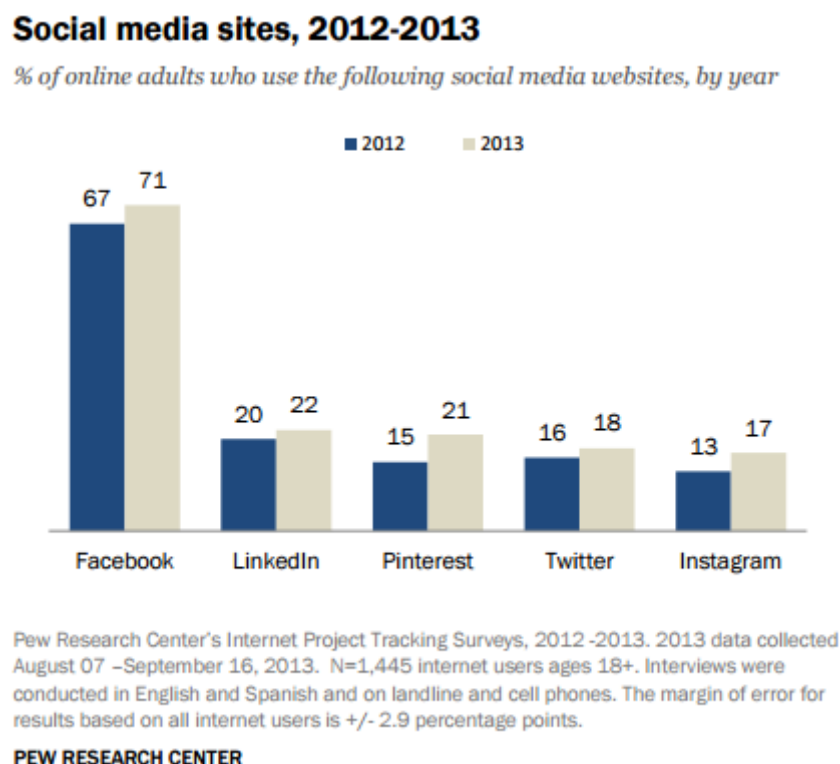


Figure 1- Usage of social media sites [1].

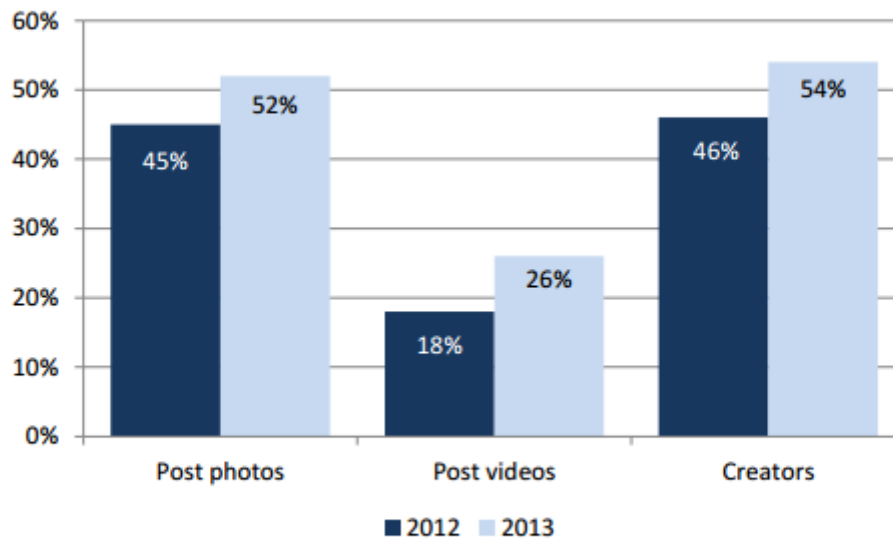
This also indicates that the usage of image sharing sites such as Pinterest and Instagram have increased the most in recent years, which indicates that people are more likely to share images online now than they were years ago.

1.3.2 Photo Sharing

As social media grows in popularity, so too does the sharing of photos and videos. A study done by Pew Research in 2013 shows that 62% of adult internet users have posted original photos or videos online, which increased from a study done in 2012, which showed 56% of users posting images online. [3] This increase in photo sharing could be attributed to the rise of smartphones, which allow users to easily take high quality photos, and then use a photo sharing application to share it easily. Figure 2 shows the increase in the usage of photos online between 2012 and 2013.

Online Creators: Posting original photos and videos, 2012-2013

Among internet users, the % who post photos and videos and the % who post either



Source: Pew Research October Omnibus Survey, October 3-6, 2013. n=852 internet users ages 18+. Interviews were conducted in English on landline and cell phones. The margin of error for results based on internet users is +/- 4.0 percentage points.

Figure 2- Increase in photo sharing online [3]

This study indicates that more and more people are sharing original photos online, which is most likely due to the rise in popularity of social media, which was discussed in the previous section. The same study demonstrates how young people are also much more likely to post images online, as shown in the table in Figure 3.

Do you ever post PHOTOS that you, yourself, have taken to any kind of website?

Among all internet users, the % who share photos they have taken themselves

All internet users (n=852)		52%
a	Men (n=414)	48
b	Women (n=438)	56 ^a
Age		
a	18-29 (n=131)	79 ^{bcd}
b	30-49 (n=222)	56 ^{cd}
c	50-64 (n=265)	37 ^d
d	65+ (n=208)	19

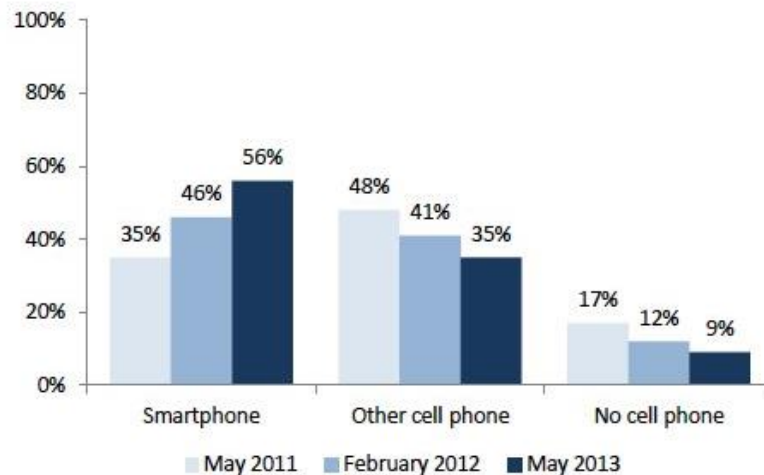
Figure 3 - Age of Image Sharing online. [3]

1.3.3 Smartphone Usage

Smartphone usage has also drastically increased in recent years, particularly amongst teenagers. According to a study done in 2013, 47% of teenagers own a smartphone, up from 23% in 2011. [4] Smartphone usage amongst adults shows a similar trend to that of teenagers, with 56% saying they own a smartphone, up from 35% in 2011. [5] Figure 4 shows how these numbers have increased recently.

Changes in smartphone ownership, 2011–2013

% of all U.S. adults who own...



Source: Pew Research Center's Internet & American Life Project April 26-May 22, 2011, January 20-February 19, 2012, and April 17-May 19, 2013 tracking surveys. For 2013 data, n=2,252 adults and survey includes 1,127 cell phone interviews. All surveys include Spanish-language interviews.

Figure 4 - Smartphone Ownership amongst adults [5]

These percentages are much higher amongst young adults in the 18-24 age bracket, with 79% of people this age saying they own a smartphone, as shown in Figure 5.

Own a smartphone		
All adults (n=2,252)		56%
Gender		
a	Men (n=1,029)	59 ^b
b	Women (n=1,223)	53
Age		
a	18-24 (n=243)	79 ^{cdef}
b	25-34 (n=284)	81 ^{cdef}
c	35-44 (n=292)	69 ^{def}
d	45-54 (n=377)	55 ^{ef}
e	55-64 (n=426)	39 ^f
f	65+ (n=570)	18

Figure 5 - Smartphone Ownership by Age [5]

1.3.4 Mobile Applications

A social media mobile application which is massively popular at the moment is Snapchat. It allows users to send photos and videos to one another. Based on a study done by Pew Research, around 26% of people aged 18-29 in the US use Snapchat on a daily basis [3]. Another application that has become popular is Instagram, with the same research report indicating 43% of people aged 18-29 said they use Instagram regularly.

With these statistics in mind, it is possible for social media applications which use photos and videos to become extremely popular, especially among young people. The proposed application is based on this functionality. Displaying the photos on an interactive map creates a social way for users to see different areas of the world, through photos taken by other users. As more and more people use their mobile device abroad, the proposed application could become extremely popular for users on holiday. According to a survey by TripAdvisor [6], 61% of people use social media while they are travelling.

As smartphones have become the most commonly used type of phone, it is important to consider the different platforms that exist. The most popular platform is Android, with iOS a close second. Figure 6 shows the percentage of phones that run each type of operating system.

		% who say their phone is an iPhone	% who say their phone is an Android
All cell owners (n=2,076)		25%	28%
Gender			
a	Men (n=967)	24	31 ^b
b	Women (n=1,109)	26	26
Age			
a	18-24 (n=238)	31 ^{ef}	43 ^{cdef}
b	25-34 (n=279)	34 ^{def}	40 ^{def}
c	35-44 (n=283)	29 ^{ef}	33 ^{ef}
d	45-54 (n=354)	25 ^f	27 ^{ef}
e	55-64 (n=392)	19 ^f	17 ^f
f	65+ (n=478)	11	7

Figure 6- Mobile Platform Usage [5]

As smartphone operating system usage is so evenly divided between these two platforms, it is an important choice to make when developing a mobile application. Creating a cross platform device could entail a lot of extra work, as iOS applications are written in Objective-C [7], while Android applications are written in Java [8].

1.4 Aims and Objectives

In this section, the planned functionality of the application is discussed in detail. This section also covers what work will be required for the application to be completed.

The aim of this project is to develop a cross-platform mobile application which allows users to share and browse photos on a map.

It is vital that the application is easy to use, and that the user interface is clean and intuitive. To ensure the application is social, users will also be able to add friends within the application, and become a part of groups. To ensure the application is delivered to a high standard, there are a number of tasks that must be completed.

1.4.1 Priority Tasks

- Research existing solutions similar to the proposed application.
- Decide on the technologies to use.
- Ensure the application is likely to be used by target market.
- Ensure the application is easy to use, and has a good user interface.
- Design the database structure.
- Determine use cases and user requirements.
- Ensure core features are fully functional.
- Test the application with a target group.
-

1.4.2 Application Functionality

The application itself has core functionality which must be included in the delivered project, and there are also secondary features which would make the application more usable.

Core Functionality

- Allow users to create an account.
- Let photos be uploaded to the map.
- Let users browse the photos on a map.
- Let users add friends and create groups.

After the core functionality has been completed, secondary features will be added to the application.

Secondary Functionality

- Let users upload pre-existing photos to where they were taken.
- Allow photos to be made private.
- Let users delete their photos from the map.
- Add tags to photos.
- Allow users to rate photos.
- Implement a filtering system for the pins.

After outlining the aims and objectives of the application, and what functionality will be included in the application, we can now look at the timeline for the project.

1.5 Timeline

This section provides a rough outline of how the project was developed, and what was done in each month leading up to submission. Figure 7 is a visual representation of the timeline of the project. The following sections discuss each month in more detail.

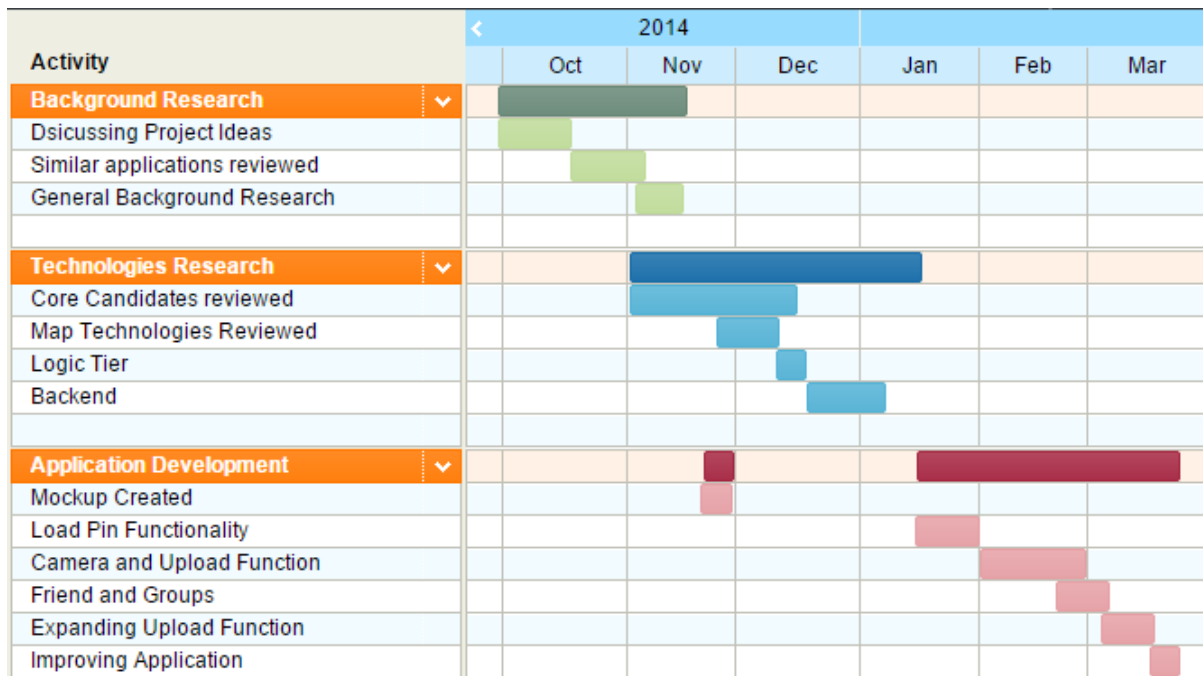


Figure 7 - Project Timeline

October 2014

Work began on the project at the start October 2014. Applications that were submitted as projects for the previous year's final year projects were reviewed. This gave a clearer understanding of what would be required for the project, and what format the thesis would require.

In mid-October, the idea of the proposed application was discussed in detail, and was considered more seriously as a project. Brief background research was conducted on similar applications, and ideas for the application were brainstormed. It was a concern that the proposed application may be too simplistic, but as the application could be modularised, complexity could be added later.

Once the project had been decided upon, other applications were reviewed in more detail. Heuristics were applied to three applications similar to the proposed application, which gave a better understanding of the user requirements.

November 2014

In November, Use Case diagrams were designed for the proposed application. This gave a clear visual representation of the user requirements. Once the diagrams had been completed, possible technologies for the proposed application were researched in detail, and HTML5 was chosen for the front end.

During this time, a meeting with Mr. Mark Foley - a lecturer who teaches Geographical Information Systems in DIT, was conducted. Mr. Foley gave recommendations for the back end of the application. This recommendations included the

In knowing the technologies that were going to be used, a rough mock-up of the application was developed. The mock up at this stage was extremely primitive, and only contained some menu options and a map screen. By the end of the month, the map was loading pins from a local database onto the map.

December 2014

In December, little work was completed on the proposed application, as college exams were before and after the Christmas break, and time was used to study. More research went into the advantages and disadvantages of the chosen technologies were conducted, and some tutorials were completed online to gain a better understanding of the chosen technologies like AngularJS and PHP.

January 2015

In mid-January, work on the project recommenced. Now that the systems technologies had been decided upon and background research had been conducted, the implementation phase of the application could be started. At the end of this month the map was successfully loading pins from a database, as well as displaying images from URL's that were hardcoded into the database.

February 2015

This month was used to get the application camera upload function working. This phase took longer than expected, as discussed in detail in chapter 5 of this report. By mid-February however the upload function as working, and the users location was added to pins that were upload. At this stage the application was semi-functional, and could be used to upload images onto the map.

Once the image upload function was working, functionality which allowed users to add friends within the application was developed.

March 2015

Once friends were working properly, a filter was created to allow friends photos to show up as different coloured pins on the map.

Login and Create Account screens were created, as well as creating sessions so a user would remain logged in after closing the application. This allowed users to be created within the application, whereas before users were entered manually into the database.

Groups were created, and users could create and add groups within the application. Group pins also show up in different colours on the map. At this stage the application had all of the core functionalities developed.

Functionality which allowed users to upload photos from anywhere was added. This meant extracting a photos EXIF data, which provided a location for where the image was originally

taken. This meant that users could upload photos they had taken somewhere else, and it would still be uploaded to the original location, as opposed to where the user currently was.

After that, the image uploading was given additional functionality. Users could now add a tag to a photo, or add photos to a group within the application.

A feature which allowed to place their photo manually on the map was then added, in case a photo had no geolocation data.

The following section will give a brief explanation of each chapter contained in this report.

1.6 Chapter Walkthrough

This section is a brief summary of what each of the chapters contained in this dissertation discusses.

Chapter 2 - Research and Evaluation

In this chapter of the report existing solutions similar to the proposed application will be reviewed in detail. A general review of each application will be done, as well as a description of the system architecture for each application. Nielsen's Heuristics will be applied to each of the applications. The results will be used to refine the user requirements of the proposed application. It will also provide a clearer understanding of what type of system architecture will be used in the proposed system.

Chapter 3 - Technology Used

For each layer in the architecture of the proposed system, a variety of technologies will be reviewed and compared. This will allow an informed decision to be made on what technologies and programming languages will be used to develop the proposed system.

Chapter 4 - Design

In this chapter, the methodology that was used for the proposed system will be discussed in detail. It will also be compared to other methodologies, to give a clearer understanding of why this methodology will be used. This chapter will also cover the system architecture of the system, and show how each piece of technology used will interact with each other. Use Case diagrams, Entity Relationship Diagrams and Unified Modelling Language diagrams are also included in this chapter.

Chapter 5 - Testing and Evaluation

This chapter includes a critical evaluation of how well the system works, and outlines the shortfalls of the project. This chapter also covers any testing of the system which was conducted, and the results will be critically evaluated and discussed in detail.

Chapter - 6Implementation

This chapter covers how the system was actually implemented. Implemented features which were outlined in the Aims and Objectives section of this report will be evaluated, and any features which could not be implemented will be outlined and discussed.

Chapter 7 - Conclusion

This chapter will provide a review of how the entire project went, the learning outcomes, and will also include a personal reflection on how the project went.

1.7 Conclusion

In this chapter the aims and objectives of the project were discussed, and vital features were outlined. These vital features will take priority moving forward. Extensive background research was also conducted in several areas which relate to the project. These areas include social media, smartphone usage, photo sharing and the use of mobile applications. Background research helped determine the existing environment, and helped determine how likely the proposed application would be used. At this stage, there is a general idea of what the application will do, but the application must be compared against existing solutions that are similar to the proposed application. We cover this in the following chapter.

2. Research and Evaluation

2.1 Introduction

There is already a general idea of the user requirements and functionalities of the proposed application, but further research is required to truly define the user requirements. This research will include critically evaluating three applications that are similar to the proposed application, and applying a set of heuristics to each. The results of this evaluation will then be examined, and will be used to gain additional user requirements for the application.

In the next section, we decide on an appropriate set of Heuristics which will be applied to the proposed application.

2.2 Heuristics

In order to further examine the aspects of these applications, a heuristic evaluation was conducted. It was decided that Nielsen's Heuristic's would be suited for the evaluation of these applications. [23] Nielsen's Heuristics consist of ten usability principles which help determine the issues of a user interface. As the proposed system is a mobile application, usability will be a major factor in the success of the application. Here are the ten usability heuristics according to Nielsen. Each heuristic is explained in more detail in section 2.4 of this report.

1. Visibility of System Status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

To further refine the requirements of the proposed applications, we need to look at pre-existing solutions that are similar to our application. This will help define the user requirements, as well as giving an idea as to how the user interface should look. In the next section, three systems are studied.

2.3 Existing Solutions

There are applications that already exist which provide some similar functionalities to what the proposed application will do. Some of these applications are hugely popular, and should be used for reference when designing the application. The applications that were chosen to be reviewed are "Snapchat", "Whisper" and "PhotoMap". In order to further define the user requirements of the proposed application these applications are reviewed in detail in the following sections.

2.2.1 Snapchat

Snapchat is an application which allows the user to take photos and send them to their friends. [35] Friends can be added using their username or their phone number. Sent photos have timers, and when the timer is finished, the photo is automatically deleted from the recipient's device. Users are also able to create a 'story' with a photo. This means that the photo is able to be viewed by the user's friends as many times as they want, within a twenty-four hour period from when the story was created. After the twenty-four hours are up, the story is automatically removed. Figure 8 is a screenshot of Snapchat.

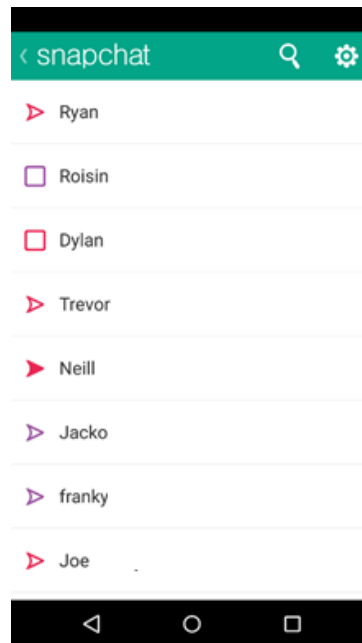


Figure 8 - Snapchat Screen

As Snapchat is a mobile application, there is very little information available about the architecture of the system. Based on research and use of the application, it can be assumed it uses a three tier architecture. The user would send the photos to a server, which then stores the photos in a database, and notifies the recipient. Once the recipient has viewed the photo and the timer runs out, the photo is assumedly delete from the server, and it cannot be viewed again.

2.2.2 Whisper

Whisper is an application which allows users to write a sentence, and then, using this sentence as baseline, it automatically suggests an image to put as the background of this sentence. [36] The photo is then uploaded, and the public can view it. Photos are known as 'Whispers' in the application. Whispers can be hearted, indicating that the user likes the photo. Users can also message other users, or reply to a Whisper with a Whisper of their own. There are four different categories to browse within the application; popular, new, school, and nearby. Figure 9 demonstrates what whisper looks like.

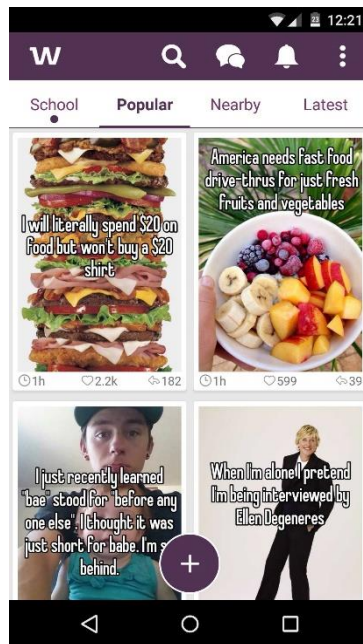


Figure 9 - Whisper Screen

The popular section displays photos which are currently deemed popular, and is based on the amount of hearts a photo has. Nearby photos can be browsed, but they are not displayed on a map. The school section allows users to join their school in the application. Once a user has joined their school, they can view whispers shared by fellow students. The new section simply displays the photos which have been recently uploaded.

This application is also assumed to have a three tier architecture, with the user uploading a photo to the server, and it is then stored in a database. On this application, the photos are stored permanently, unless the uploader decides to delete the photo.

2.2.3 PhotoMap

PhotoMap allows users to view all of their devices photos on a map. [37] The application reads the geolocation information of pre-existing photos on the user's device, and uses this information to display it on a map. Photos are displayed in folders, and choosing a photo displays the photo on a Google Map. All photos are ones which are stored on the user's device, so the application assumedly uses a single tier architecture. The only interaction with a server is the Google Maps server, which allows the photos to be viewed on to the map. Figure 10 is a screenshot of the application.

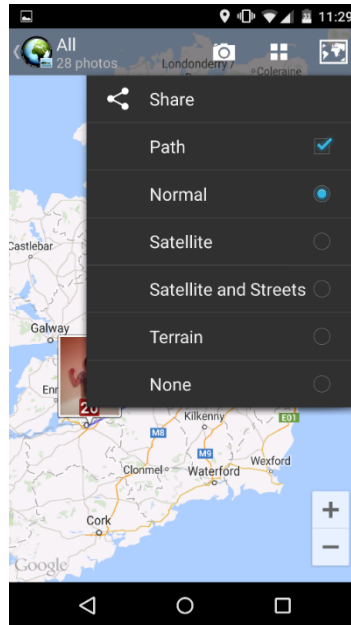


Figure 10 - PhotoMap Screen

After the above applications were reviewed, a Heuristic Evaluation was done, as shown in the following section.

2.4 Heuristic Review of Existing Systems.

In section 2.2 a heuristic evaluation was chosen, Nielsen's Heuristics. In section 2.3, existing solutions that are similar to the proposed application were reviewed and tested. It is now important to apply these heuristics to the existing solutions, which will help refine the user requirements of the proposed application. Each heuristic will have a score, and a brief description of why the score was given.

Visibility of System Status

This heuristic is to measure how well the system keeps the user informed of what is happening in the application, and where they currently are within the application.

Application	Summary	Score
<i>Snapchat</i>	Snapchat is very easy to understand, as there are only 4 screens. While there is no heading on each screen to say where you are, the content is easily distinguishable from other screens.	9
<i>Whisper</i>	Whisper has four clear headings at the top of the screen, which represent the different sections. It is very easy for users to understand where they are in the application.	10
<i>PhotoMap</i>	Very hard to understand. The icons used are open to interpretation, and it is difficult to know where one is when using the application.	4

Winner: Whisper.

Match between system and the real world

This heuristic checks that the words used in the application are understandable to a lay person. It should not contain any technological terms that some users may not understand.

Application	Summary	Score
<i>Snapchat</i>	Snapchat uses very little language, and consist mainly of the word 'Snapchat', as well as friend's usernames. Because of this, it is able to be used easily by anyone, even without knowing the language.	10
<i>Whisper</i>	Whisper also uses very little words, and the main sections are the only real text within the application. It is available in different languages, and the English is perfect.	10
<i>PhotoMap</i>	The application uses icons, with very little words. The words used are things like 'Map' and 'Folders', to indicate what the user is looking at.	8

Winner: Snapchat and Whisper

User control and freedom

Users should be able to easily go back within the application, and have a clear undo button.

Application	Summary	Score
<i>Snapchat</i>	All of Snapchats screens can be exited by simply hitting the back button, as well as an 'X' on the top right of the screen. There is an undo button, but there is no redo button.	8
<i>Whisper</i>	Whisper makes the user use the devices back button to exit screens, and it works well. There are no graphical back buttons.	7
<i>PhotoMap</i>	Very confusing. Pressing the back button differs depending on where the user is in the application. It will sometimes go back to the previous screen, and sometimes it will exit the application entirely.	4

Winner: Snapchat

Consistency and standards

It should be clear to a user what each button or action will do. The application should also follow a platforms standards, such as enabling the use of physical hardware buttons.

Application	Summary	Score
<i>Snapchat</i>	Snapchat allows users to use the physical back button on their device, as well as the camera button, if there is one. The icons used are relatively easy to understand, but users need to learn two of the notification icons.	7
<i>Whisper</i>	Users must use their devices physical back button to exit screens. The icons used are simple, easy to understand, and effective.	10
<i>PhotoMap</i>	Users must use their devices physical back button to exit screens. The icons are poorly chosen, as some are open to interpretation.	5

Winner: Whisper

Error prevention

Applications should have a confirmation dialog before an important action is done. This help prevent simple mistakes where a user might click an option by accident.

Application	Summary	Score
<i>Snapchat</i>	Snapchat has an undo and cancel button, but there is no confirmation text when sending pictures, which may lead to pictures being sent by mistake.	8
<i>Whisper</i>	It is easy to exit any area of the application by hitting the back button. There is no confirmation box when posting a whisper, but it is easy to delete once posted.	9
<i>PhotoMap</i>	The back button changes its behaviour depending on where the user is in the app, and there is no confirmation dialog when the user presses back to exit the application.	2

Winner: Whisper

Recognition rather than recall

Applications should be easy to understand for a first time user who has no prior knowledge of the system. Methods to implement this include the use of icons instead of words. Each option should be clearly marked.

Application	Summary	Score
<i>Snapchat</i>	Snapchats simple four screens makes it very easy to understand, but users do need to learn where each of these screens are, as there are no clear indicators on each screen. The notification icons are just coloured squares, which also need to be learned.	8
<i>Whisper</i>	The icons used are very easy to understand, and users will be able to recognise them easily.	10
<i>PhotoMap</i>	Icons are open to interpretation, and the application is difficult to traverse. It takes quite a while to get used to using the application.	2

Winner: Whisper

Flexibility and efficiency of use.

Advanced users should be allowed to create a shortcut if they want to, so as to increase ease of use. This speeds up the workflow for advanced users, and does not affect novice users.

Application	Summary	Score
<i>Snapchat</i>	There are no real customisation options for Snapchat, and users cannot create shortcuts.	0
<i>Whisper</i>	There are no customisation options available.	0
<i>PhotoMap</i>	Users can change the view of the folders to display in folders, weeks, months or years. The user can change the colour scheme of the application, and can also view the map as normal, satellite, or terrain.	8

Winner: PhotoMap

Aesthetic and minimalist design

The information displayed to the user should be as minimal as possible, without any unneeded information displayed to the user. This reduces the amount of clutter on the screen, and makes it easier for users pay attention to important information.

Application	Summary	Score
<i>Snapchat</i>	Snapchat is excellently designed, and conforms to the new minimalistic, flat design trend.	10
<i>Whisper</i>	Whisper is very well designed, and utilises flat design.	9
<i>PhotoMap</i>	The application design is quite outdated, and does not use modern design.	6

Winner: Snapchat

Help users recognize, diagnose, and recover from errors

Error messages explaining what went wrong should appear anytime there is an error. This makes the user more likely to try and solve the problem, rather than getting frustrated and stop using the application.

Application	Summary	Score
<i>Snapchat</i>	Snapchat displays an error when it cannot connect to the internet, and asks to user to check their internet connection. There were no other errors that could be found or tested.	9
<i>Whisper</i>	Whisper lets the user know when there is no internet connection. There are no other errors that could be tested.	9
<i>PhotoMap</i>	When there is no internet connection, the application still tries to display the map, but is only able to display a grey background. There should be an error message explaining the problem.	0

Winner: Whisper and Snapchat

Help and documentation

While it is important the application can be used without much help, users should still have access to help and documentation if they want to view it.

Application	Summary	Score
<i>Snapchat</i>	Snapchat has visual help steps when a user initially opens the application. There is also a help guide which is in the options menu.	10
<i>Whisper</i>	Whisper does not offer any help for users, even when the application is initially launched. It does provide a user guide in the settings though.	8
<i>PhotoMap</i>	There is no tutorial or help documentation for this app, and it is hard to understand as it is.	0

Winner: Snapchat

After giving each application and heuristic score, we review these results mean in the next section.

2.5 Summary of Findings

Application	Final Score
<i>Snapchat</i>	79/100
<i>Whisper</i>	81/100
<i>PhotoMap</i>	39/100

The application with the highest overall score was Whisper, with Snapchat a close second. PhotoMap scored the lowest out of the three reviewed applications.

Snapchat and Whispers design makes it easy for new users to understand, which is vital for social media applications. The proposed application needs to be easy to use, and Whispers screen layout is similar to what the proposed application is planned to do. Swipe actions to traverse between screens will not be possible with a map screen, as swiping will just move the map, so buttons will be used instead.

Whisper has headings across the top of the screen is a layout which may be applied to the proposed application, as it is very easy to understand and use.

While PhotoMap has a lot of pitfalls, the way it displays the individual pins and photos is a feature which may be implemented in the proposed application. It displays photos as folders when zoomed out, then spreads the album out into individual photos when the user zooms in. If a user clicks on a folder pin, the photos are displayed in a grid view.

Another feature which was used by each of the apps studied is the use of icons rather than text. Using icons makes the function of the icon clear, and does not rely on being the same as the user's language. For this reason, icons will be used in the proposed application.

In the customization section of the Heuristics, both Snapchat and Whisper got a zero, as there are no customization options available. This is not necessarily a downside to using the applications, as there is no need for customisation, PhotoMap has options which I may be used in the proposed application. Changing the look of the map may be something a user would like to do, and may be integrated in the proposed application.

The error prevention Heuristic showed Whisper and Snapchat both told the user when there was an error connecting to the internet. Photomap did not display any error messages, and scored a 0/10. The proposed application will need to tell the user when there is a problem, either with their internet connection or when any other action fails, such as uploading photos.

Now that the applications have been reviewed and critically evaluated, it is important that these results are used to refine the proposed applications user requirements.

2.6 User Requirements

After studying three similar systems, it helped define user requirements. It also helped give more ideas as to what features that could be used in the proposed application. Here is a list of ideas and requirements that have been gained from evaluating the three other applications, followed by the application(s) that inspired the requirement.

- Screen Layout should be easy to understand and navigate.– Whisper
- Application should use icons as much as possible - Snapchat.
- Application should use as little words as possible – Snapchat and Whisper
- Adding friends should be extremely simple – Snapchat

- Users do not necessarily need to be able to login with a social network - Snapchat
- Users should be presented with error messages – Snapchat and Whisper
- User should be able to customise the map – PhotoMap
- Application should provide user with error messages – Snapchat and Whisper
- Application should have confirmation dialogs where applicable – Whisper.
- Application should handle grouped pins in an attractive way – PhotoMap

2.7 Survey Results

When the idea for the proposed application was still in development, it was discussed with several peers currently studying Computer Science in DIT. These people were all in the 18-30 year old age group. Verbal feedback was taken into account, and it was largely positive, with most suggesting it was a good idea.

When the idea was better formed, an online survey with SurveyMonkey was created. [39] The results were extremely positive, and everyone who took the survey said they would be interested in such an application. It also confirmed the studies which were covered in the background research in Chapter One of this report – Android and iPhone usage dominate smartphone usage, and the usage between the two is extremely similar. The full results of this survey can be found in the appendix of this report.

A meeting was also conducted with Mr. Mark Foley, a lecturer here in DIT whom specialises in Geographical Info Systems. Mr. Foley recommended using PostGIS [26] if the proposed application needed a spatial database, and he also recommended Leaflet.js [18] for displaying the pins on a map. These technologies are both reviewed in detail in chapter 3 of this report.

2.8 Conclusion

This chapter described the research that went into existing solutions similar to the proposed application. Nielsen's Heuristics were applied to three applications, Snapchat, Whisper, and PhotoMap. For each heuristic, each application was given a rating from one to ten, and the results were examined. After evaluating these results, user requirements were deduced, which will help when implementing the proposed application.

After gathering the user requirements and getting an idea of how similar applications work, there is now a clearer idea of what the application functionalities will be. We now need to determine what technological options there are for each layer of the system architecture, and define the advantages and disadvantages of each. This review is covered in the next chapter.

3. Technology Used

3.1 Introduction

In Chapter 2, similar systems were evaluated, which helped determine that a three tier system architecture will be used for the proposed system. Figure 11 is a basic visualisation of what this system looks like logically. For each of the layers in this architecture, there are a variety of technologies and programming languages which could be used.

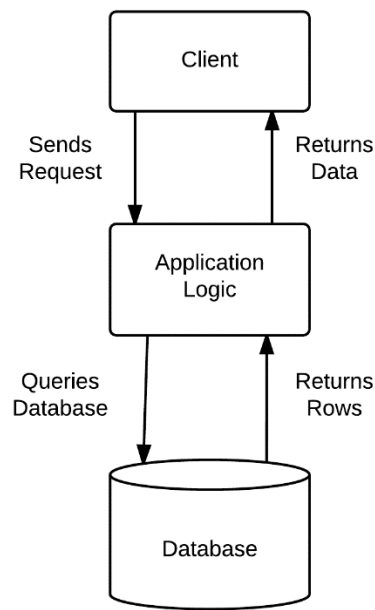


Figure 11 - System Architecture

In the previous chapter, the user requirements were defined, which will allow for a more informed choice when choosing the technology to use for the proposed application. It is important that the chosen technology is capable of the functionalities required by the proposed application.

This chapter reviews several of these technologies, and outlines the reasons why each technology was eventually decided on. The following section reviews the presentation tier of the application.

3.2 Presentation Tier

The presentation tier is everything that the user can see and interact with. It contains the user interface, as well as client-side processing. In the proposed application, client side processing will be minimal, as most of the processing can be done on the server and then sent to the client. In this section, candidate technologies are examined.

3.2.1 Core Technology

For the core code base of the presentation tier, two technologies were reviewed; HTML5 and Java. The following sub-sections look at the advantages and disadvantages of each, in order to determine which language will be best for the proposed application, keeping in mind the user requirements.

3.2.1.1 HTML5

HTML5 is a relatively new way to develop mobile applications, but according to a study done by Telerik [10], 41% of developers are now developing in HTML5, and a further 32% are developing hybrid apps, which utilise native functionality as well as HTML5 elements.

HTML5 uses tags to define elements web page. These elements can be assigned to a class, and then CSS can be applied to these classes. With the advent of HTML5, more semantic

tags have become available, such as the canvas tag and the video tag. While the video tag will not be used in the proposed application, the canvas tag can be used to generate the map.

Advantages

- Personal familiarity with HTML, CSS and JavaScript.
- Cross platform – If the application is developed in HTML5 it can run on any platform, and it will also be possible create a website easily from the same code.

Disadvantages

- Not as powerful as Java. [38]
- Newer - less documentation for application development online.
- Can have slower performance. [38]
- Cannot access device hardware on its own. [38]

The biggest shortfall of HTML5 application development is the lack of access to the devices hardware. However, this issue is addressed using the Cordova. [11] Cordova is discussed in detail in section 3.2.2.

HTML5 is also not as powerful as Java, which can lead to the application being slow. [38] This should not be a problem for the proposed application, as most of the resource intensive processing can be done on the server before it is sent to the client to be displayed.

3.2.1.2 Java

Java is the default language if one uses Googles Android Software Development Kit (SDK). [8] Java is one of the most commonly used programming languages, and has extensive online documentation.

Advantages

- Faster performance. [38]
- More powerful than HTML5. [38]
- More documentation and tutorials online than HTML5.
- Better gesture support. [9]

Disadvantages

- No personal experience in developing Java Android applications
- Java applications are not cross-platform. A lot more work will be required to port the application to other platforms. [34]

Java development allows the application to be more powerful, as it is native on android and allows for full access to the devices processing power. [9] As discussed in the previous section, performance should not be a problem for the proposed application, as most of the processing can be done on the server. Java also has more extensive documentation online, which would help when developing the application. The big downside of developing in Java is the lack of cross platform support.

Chosen Technology for the Presentation Tier

After researching the benefits and downfalls of both Java and HTML5, HTML5 was decided upon to develop the application front end. The downfalls of HTML5, namely performance issues, should not be a problem for the proposed application, as most of the work can be done on the server before it is sent to client. HTML5's other main downfall, the inability to access a devices native functionalities, can be addressed using Apache's Cordova. Cordova creates a web view which sits above each platforms native SDK, and allows for access to the devices functionality, such as the camera and geolocation. Cordova is discussed in detail in the following section.

The development of HTML5 applications is becoming more and more common. [10] As the research in Chapter 2 of this report showed, Android and iOS ownership is almost the same, so developing an application for both platforms effectively doubles the target market.

Now that the core technology has been decided, the technologies that are available when developing HTML5 mobile applications will now be examined in detail.

3.2.2 HTML5 Technologies

There are several useful technologies that exist which will be used when developing the proposed application. These tools assist with developing the user interface, accessing the devices native features. The native features which will be required by the proposed application will be;

- Access to Device Camera
- Devices Geolocation
- Access to device file system

This section discusses each of these technologies, and provides reasoning as to why they will be used.

3.2.2.1 Cordova

Apache's Cordova is a tool which helps develop cross platform hybrid HTML5 applications. A hybrid application uses HTML5 for the user interface, and also allows the application to access the devices hardware.

Cordova comes with built in functions for building cross-platform applications. It uses a command line interface. Adding android or iOS support can be done using one line of code, namely "cordova add platform iOS/android." It also makes it easy to add native functionality, such as accessing the devices camera, geolocation or file system. Figure 12 illustrates how Cordova works.

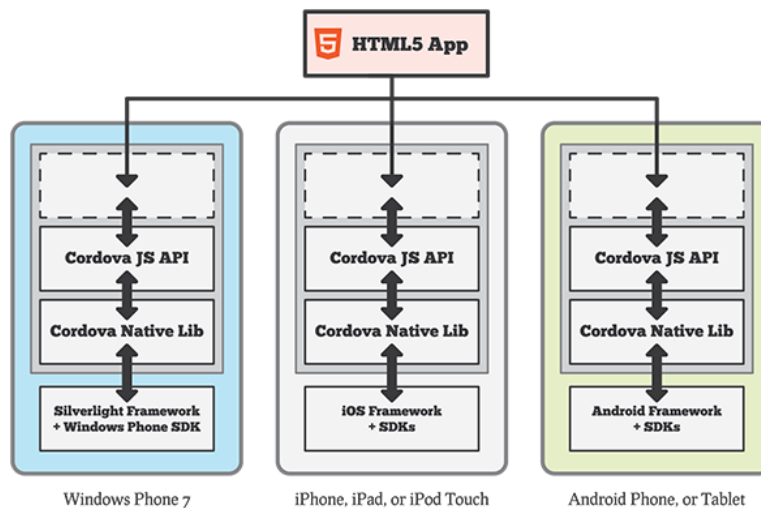


Figure 12 - How Cordova Works [12]

The HTML5 application, which is basically a web-view, is created using HTML5, CSS and JavaScript. Cordova then enables this web view to interface with each platforms API. This allows the devices native features to be accessed by the application. The downside of using Cordova is that the online documentation is quite poor, as it is quite a new technology.

To become familiar with how Cordova works, several sample application were created, and the online documentation was read.

To help design the user interface, the Ionic Framework will be used in the application, which is discussed in the following section.

3.2.2.2 The Ionic Framework

Ionic is a library which sits on top of Cordova, and uses Cordova to compile the application for use on mobile. [13] It comes with application templates, and makes it easy to get a template of an application working. It uses AngularJS and HTML5 to create views within the app, and makes transitions between screens smooth. Ionic also comes with JavaScript functions for displaying popups or menus. Figure 13 is an example of an Ionic Template.

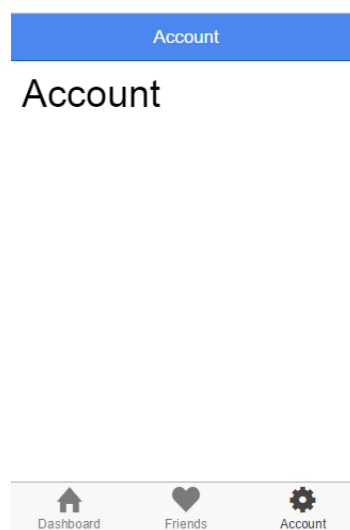


Figure 13 - Ionic Template

Ionic uses AngularJS to navigate between the different screens in the application. AngularJS is reviewed in more detail in the next section. To become familiar with Ionic, as with Cordova in the previous section, sample applications were created, and online documentation was read.

3.2.2.3 AngularJS

AngularJS is a JavaScript library which allows developers to create single page web applications. [14] It uses custom tags to extend HTML5 functionality. It allows for data manipulation, and variables can be used to store information.

In the proposed application, it will allow users to navigate between screens, or 'views' as they are known in AngularJS, without having to reload the page. By using AngularJS to define where the content section of the screen is, it ensures only the content gets changed between screens, and not the navigation bar. Clicking on a navigation button calls a function which will then load the content of the respective page into the content area. This provides a much more fluid experience for the user. The AngularJS website has tutorials to help understand how AngularJS works. Several of these tutorials were completed to become familiar with the library.

How each of these technologies are used within the application is explained in detail in the following section.

How these technologies work together.

The way these technologies work together is as follows. Cordova creates a web-view, which sits on top of each devices respective SDK. This allows the application to access all of the devices native features, and the user interface is developed in HTML5. Developing in HTML5 alone does not allow for access to the devices native features, such as the camera or geolocation. [9] Ionic is used to create a template, which sits inside this web-view. Ionic also uses AngularJS to switch between the screens of the application, and allows the user to traverse screens without reloading the entire page.

3.2.3 Mapping Technologies

For the map screen of the application, there are important requisites for the technology to be used. These requisites are;

- It must work well on mobile.
- It must be free.
- It must allow custom pins and markers to be displayed on the map.

With these requisites in mind, two technologies were researched, Leaflet.js, and Google Maps. These are discussed in detail in the following section.

3.2.3.1 Leaflet.js

Leaflet is an open source JavaScript library which loads and displays pins on a map. [18] Here we will discuss some of the advantages of this technology.

Advantages

- Good documentation and tutorials online.
- Open source
- Well suited for mobile, is small and lightweight, only 24kb.
- Allows for custom map markers.
- Library of useful plugins [24]
- Uses CSV files to load pins, which can be easily extracted from a database.

This library provides a lot of functionality which is required by the proposed application when displaying the pins on the map. It also works with both mobile and desktop browsers, so it will be easy to make a functional website using this library. Leaflet comes with a lot of useful functions and plugins, which will help when displaying the pins in an attractive way.

3.2.3.2 Google Maps

Google Maps is a free mapping service provided by Google. [25] It offers a free JavaScript library which utilises the Google Map API. It also has built in functionalities such as Directions to locations, and loading pins on the map. There is also fantastic documentation online. It does have a limit on map views, 25 000 map loads each day, for more than 90 consecutive days, after which the service becomes paid.

Advantages

- Free up to a point.
- Good Documentation Online
- Feature rich.
- Uses CSV file to load pins from the database.

Disadvantages

- Larger file size than leaflet.
- It costs money if the map views goes over the limit.
- Built in features may not be needed.

Google Maps, like Leaflet, allows for custom pins to be loaded on to the map, which is required for the proposed application. The downsides of using it include the usage limit which is 25,000 views per day, and the size of the JavaScript file compared to leaflet.

Chosen Technology for Mapping Technology.

After researching these two solutions, Leaflet.js was picked as the mapping technology to be used in the application. Leaflet is extremely lightweight, and is more suitable for mobile usage. It also has a huge library of plugins, which could be added to the proposed application. Google Maps comes with a lot of features, but these are built in, and may not be required by the proposed application.

3.3 Logic Tier

The middle end of the system will contain the application logic. It will take requests from the user's device, interpret the request, and query the database with this request. It then sends the returned dataset to the user's device.

As discussed in the previous section, one of the problems of developing in HTML5 is that it is not as powerful as Java, and can be slow when processing large amounts of data. Because of this, the vast majority of the data processing will be done on the server, and the results will then be sent to the user's device. This will minimise the amount of processing required on the client side, and will improve the applications performance.

3.3.1 PHP

PHP is the most commonly used backend programming language. According to W3Techs, 82% of all websites use PHP as their backend language, which is up from 77.8% in 2012. [15] In the proposed application, it will be used to query the database. When the result set is returned, the PHP will then format it for use in the application. PHP will allow all of this processing to be done on the server before it is sent to the user's device.

3.3.2 Ruby on Rails

Ruby on rails is also a backend programming language that allows for interfacing with a database. [40] The downsides of ruby are;

- Less widely use than PHP, so less help and documentation online. [15]
- According to online benchmarks, it is slower than PHP in most operations [41]

Chosen Technology for the Logic Tier

As pointed out in the previous section, ruby on rails has less online documentation, and is slower than PHP. Given these two extremely important factors, PHP was chosen for the logic tier of the application.

3.4 Data Tier

After deciding on the middleware to be used in the application, we now need to determine what type of database will be used. The back end of the application will deal with storing the application data, and will be a MySQL database.

3.4.1 MySQL

MySQL is an open-source relational database owned by Oracle. According to Oracle, it is the world's most widely used Database [16]. The main advantages of a MySQL database are;

- Free to use.
- Relatively fast. [17]
- Extensive documentation.
- Personal familiarity with SQL Queries.

The fact MySQL is free, and works with useful Database Administration tools such as myPHPAdmin, makes it ideal for this project. Familiarity with doing Oracle SQL queries is also an advantage, and MySQL has almost the exact same syntax. If further research is required, there is extensive online documentation, as it is such a widely used database.

Should a spatial database be required in the future, PostGIS with PostgreSQL will be used instead, which is reviewed below.

3.4.2 PostGIS and PostgreSQL

PostGIS is an open source spatial database extender for PostgreSQL. PostgreSQL is a relational database similar to MySQL. Using PostGIS with PostgreSQL allows for spatial objects and queries to be performed on the database. [26] Spatial objects can include an area on a map, polygons, or a line between points. The main advantages of using this combination of database would be;

- It's free.
- It is easy to transfer data from a MySQL database to a PostgreSQL database
- Good documentation online.

The current requirements of the proposed application are covered by using MySQL. The reason MySQL was chosen over PostgreSQL was for the amount of documentation and help available online as MySQL is more widely used. [16] If the requirements change, and the database needs to be transferred from MySQL to PostgreSQL so that PostGIS spatial queries can be used, there is a tool called PGLoader [27] which helps with this. It is an open source tool which copies all the data from MySQL into a PostgreSQL database, using a simple command line interface.

Chosen Technology for the Data Tier

For the data tier of the application, MySQL was chosen. It was chosen for its extensive online documentation and the fact it is the most widely used in the world. If the requirements for the application change so that it needs a spatial database, PostgreSQL will be used, along with the PostGIS plugin. As the requirements are now, MySQL is sufficient, so it will be used.

3.5 Summary

After looking at the technologies in more detail, we now have a better idea of the application will look logically. Figure 14 shows the system architecture with the chosen technologies included.

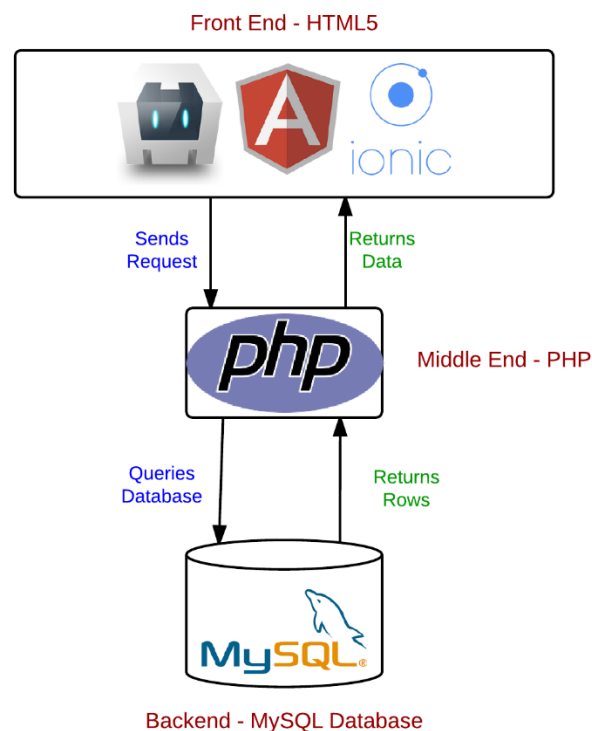


Figure 14 - Architecture with Technologies Included

On the front end, HTML5 will be used to develop the user interface. Using Cordova, the proposed application will have access to the devices hardware, which was a downfall of HTML5. Ionic will be used to design the application, and will allow for quick development of an appealing user interface. Ionic will be using AngularJS to provide a fluid user experience, and minimise page loads.

The middle tier will use PHP, and will allow users to interact with the database. When the user uses the application to access data or upload new data, the application will call a PHP script. The PHP script will take the users request and query the database. It will then format the returned dataset and send it back to the client's device.

The database that will be used is a MySQL database. MySQL is a relational database, and will consist of tables that are linked with keys and foreign keys.

3.6 Conclusion

In this chapter, the different types of technologies which will be used in the application were discussed. Each technology was critically evaluated, and the advantages for each were outlined. The reasoning for choosing each technology was also discussed. It was vital that each piece of technology to be used in this project was free to use, as there is no budget for the project. With this in mind, the technology chosen for each layer is still well suited for its purpose, and no functionality was conceded.

In the following section, the design of the system will be discussed, and several diagrams will be presented and explained.

4. Design

4.1 Introduction

In Computer Science, it is important to have a clear logical understanding of how different aspects of the system will work. To do this, several diagrams must be created.

In this chapter we will discuss the development methodology that was used for the project, as well as another methodology to compare the advantages and disadvantages of each. This chapter also contains the system architecture diagram and use case diagrams.

4.2 Methodology

When developing any system, it is beneficial to follow a design methodology. Different methodologies suit some systems better than others, so it is important to choose one that is best suited to the system in mind. In this section, we will review two different methodologies that could apply to the proposed system.

4.2.1 Rapid Application Development

The Rapid Application Development (RAD) methodology was developed by James Martin in 1991 [19].

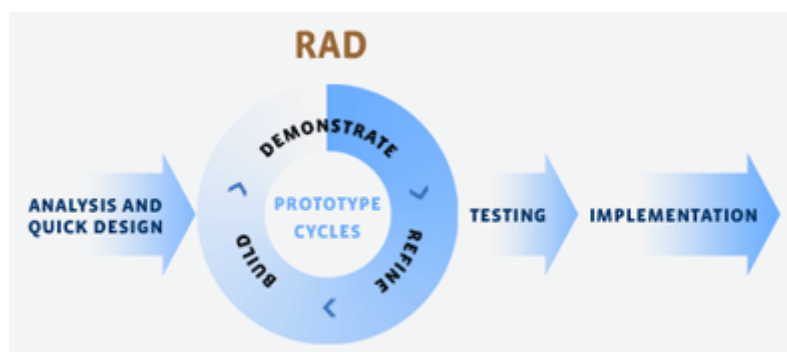


Figure 15- Rapid Application Development [20]

The objectives of this approach, as described by James Martin are;

- Fast development of high quality system at low investment cost
- Reduce projects risk by breaking it up into smaller parts
- Uses iterative prototyping throughout the development phase
- Involves the User heavily throughout the development phase

This development method involves developing prototypes quickly, and adjusting the requirements as the project is developed. It involves modularising the project functionalities. It helps to reduce the risk, as each functionality can be added iteratively. This approach makes it easier to add and remove functionality as the application is being developed, and will be particularly useful once the core functionality is complete. A way in which the proposed application development will differ from the RAD approach is the User involvement. As the application will be defined and coded by a single person, the User requirements are known, and can be integrated into the application iteratively.

The application will be developed incrementally, and testing will be conducted at each stage. Each of the main functionalities will be coded separately, and then interactivity with the other functionalities will be coded. The functionalities will be developed in order of importance. Functionalities of higher importance will be coded first, and then the next most important, and so on until the application is completed. This reduces the risk of having an unusable application before the deadline, as the application can be delivered in a working condition without the least important functionalities added.

The spiral development model was developed by Barry Boehm in 1998. [28] It is a modified version of the waterfall development process, and addresses the issues associated with it. The main issues of the waterfall process are that all the requirements need to be known up front, and integration is delivered all at once at the end of the process. This does not follow the problem solving nature of software development, as requirements can change as the application is developed. Figure 16 is a representation of how the spiral development process works.

Figure 16 - Spiral Development [28]

4.2.3 Chosen Methodology

involvement can invoke delays, but for the proposed application it will not apply. The fact that the application can be modularised means it is suited for RAD, and it will reduce the risk of delivering a system which is not functional. This is due to the fact prototypes are developed rapidly and improved upon, and each prototype becomes more and more functional. This is demonstrated in Chapter 6 of this report.

The way in which RAD differs from the spiral development model, is there is less risk analysis involved in RAD. RAD uses constant feedback to improve the application, and goes through prototypes extremely quickly.

4.3 System Architecture

The system will have a three tier architecture, as seen in Figure 17.

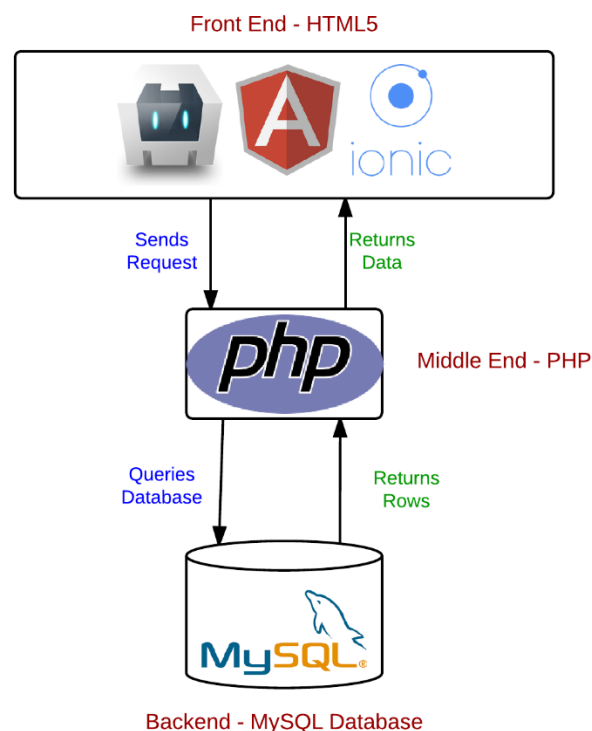


Figure 17 - System Architecture

The front end will consist of the HTML5 application, which uses Cordova and AngularJS technologies to function. This front end then uses PHP scripts as a means of communicating with the MySQL database.

4.4 Other Diagrams

4.4.1 Entity Relationship Diagram (ERD)

The ERD in Figure 18 represents how the tables are stored in the database, and the type of relationship that they have with each other.

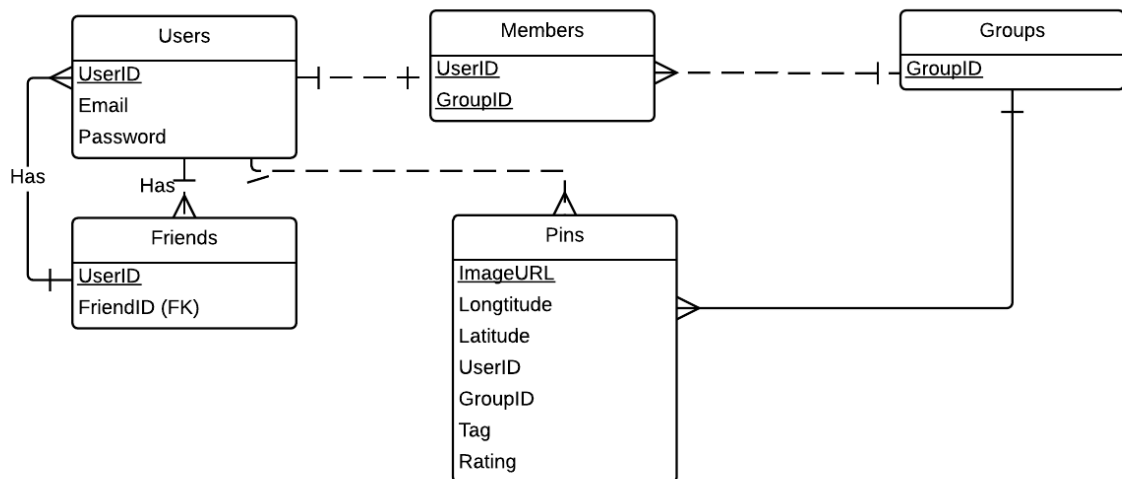


Figure 18 - Entity Relationship Diagram

A user can be a group member, which is to be part of a group. Groups are identified by unique GroupID. A user can also have friendships with other users. Users can create many pins, with all the information about the pin stored in this table. Several Pins can be associated with an Activity, and an activity is uniquely identified with an Activity ID.

4.4.2 Use Case Diagram

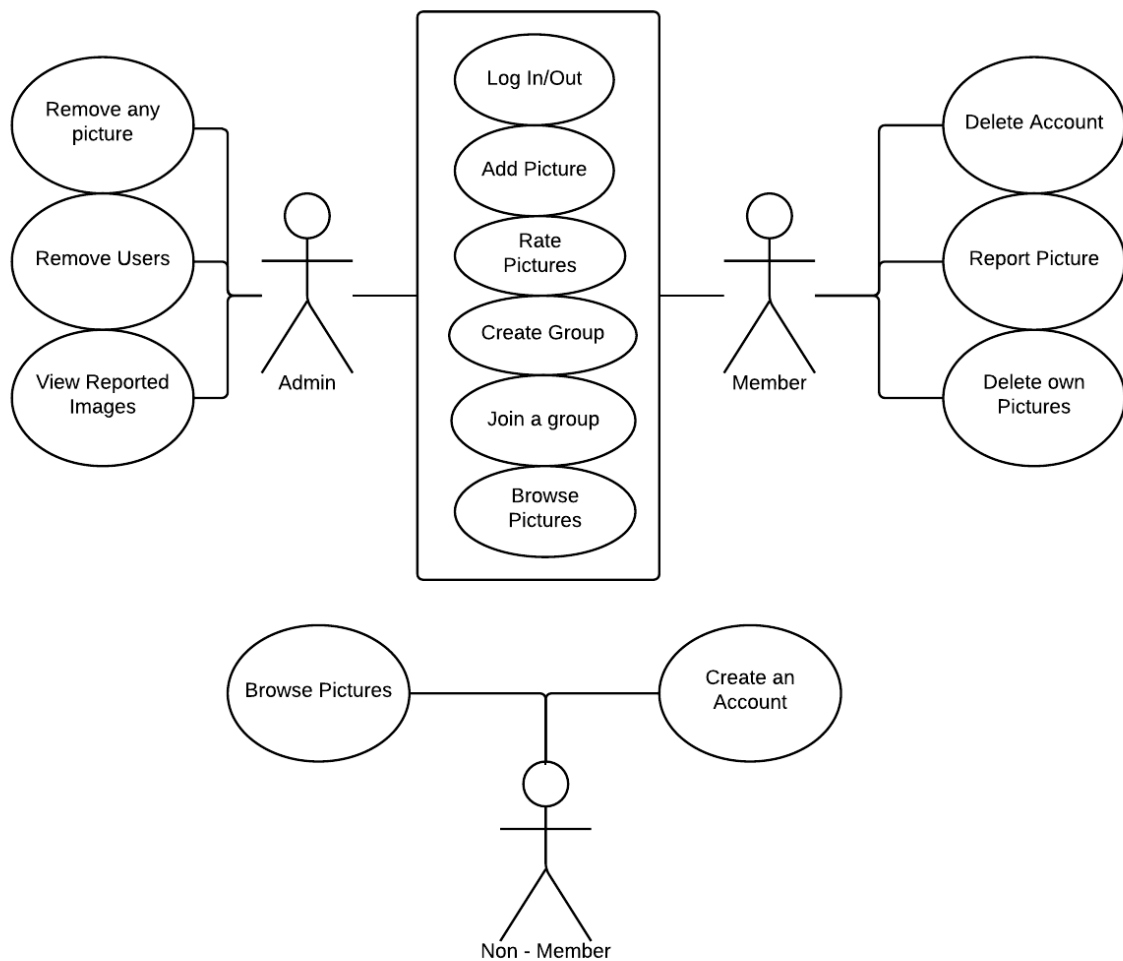


Figure 19 - Use Case Diagrams

4.4.2.1 Member

- Log in and out
- Delete account
- Add a picture
- Browse pictures
- Report pictures
- Rate pictures
- Delete own pictures
- Create a group/session
- Join a group/session

A member will need to be able to log in to the application, and access all the core functions of the application. These include browsing the map and public photos, or deleting photos they have uploaded. A member should also be able to rate the photos they view. Members will also be able to create a group session, or join a group created by someone else.

When the user opens the application, they will first see a map centred on their current location. They will be able to see all pins in their current view of the map. Clicking into these pins will bring up the corresponding image, and the user will then be able to rate or report this picture. When the user browses the map, different pins will be loaded. The way these pins will be displayed when zoomed out is still to be decided.

4.4.2.2 Administrator

- Log in and out
- Browse pictures
- Rate pictures
- Remove any picture
- Remove users
- Add picture
- Report picture
- View reported pictures
- Create a group/session
- Join a group/session

An administrator will need to be able to do all the things a member can do. They will also need to be able to browse reported photos, delete any photo, and review users who have been reported.

4.4.2.3 Non – Member

- Create an account
- Browse pictures

When a user is not logged in or is a non-member, they will only be able to browse the public photos on the map. They will not be able to rate or report pictures, and they will not be able to join or create a group. There will need to be a clear 'Create Account' option.

5. Implementation

5.1 Introduction

In the previous chapter, the logical design of the application was developed. The development methodology was also decided upon. This builds on previous chapters where background research was done on the application, and the technologies that will be used were chosen. This chapter discusses how the application was developed and implemented.

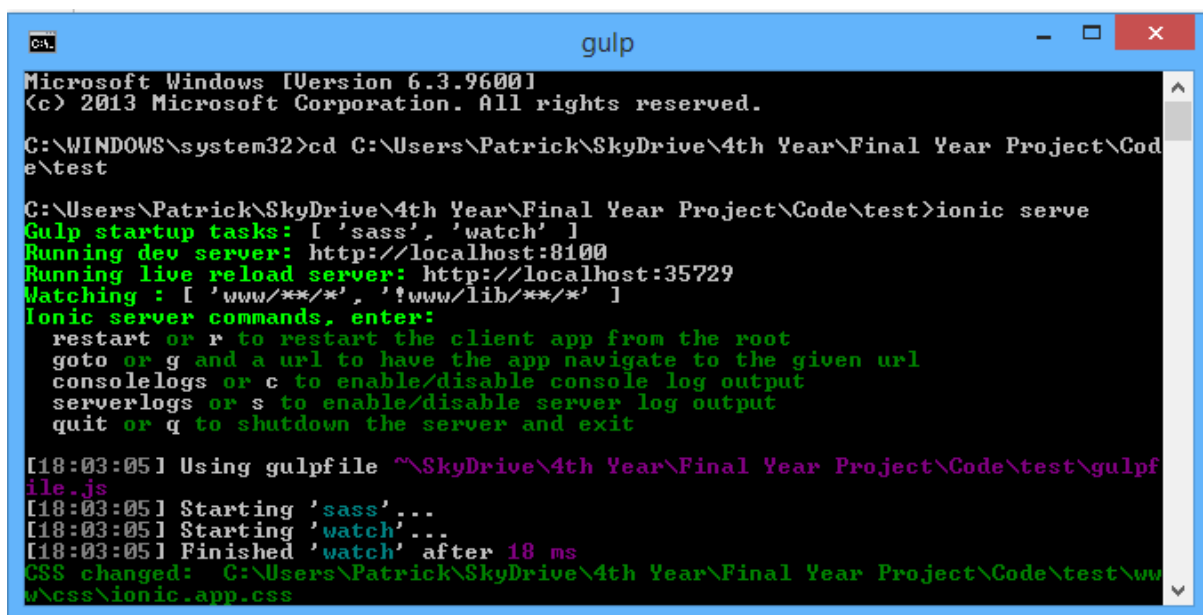
In this chapter, the development environment used to develop the application is discussed, and the differences between different prototypes of the application are examined in detail. As the application followed the Rapid Application Development approach, the development of the application was modularised, and each screen of the application was developed separately and then integrated with the rest of the application.

The application screens were not developed entirely separately. Each screen was developed to a functional standard, and added into the application. Once this was done, additional functionality was added to each screen where necessary.

In the following section, we discuss the environment used to develop the application.

5.2 Development Environment

The client side development involved developing the application in HTML5, CSS, JavaScript and AngularJS. This involved using the Ionic Command Line Interface (CLI) to build the application, and the text editor used was Sublime Text 3. No IDE was used, as using the 'ionic serve' command with its CLI generated a web page which automatically refreshed with any changes to the source code. This provided immediate feedback on any code changes, and allowed for quick development. Figure 20 demonstrates the CLI after the command 'ionic serve' has been executed.



```
gulp
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\Patrick\SkyDrive\4th Year\Final Year Project\Code\test

C:\Users\Patrick\SkyDrive\4th Year\Final Year Project\Code\test>ionic serve
Gulp startup tasks: [ 'sass', 'watch' ]
Running dev server: http://localhost:8100
Running live reload server: http://localhost:35729
Watching : [ 'www/**/*', '!www/lib/**/*' ]
Ionic server commands, enter:
  restart or r to restart the client app from the root
  goto or g and a url to have the app navigate to the given url
  consolelogs or c to enable/disable console log output
  serverlogs or s to enable/disable server log output
  quit or q to shutdown the server and exit

[18:03:05] Using gulpfile ~\SkyDrive\4th Year\Final Year Project\Code\test\gulpfile.js
[18:03:05] Starting 'sass'...
[18:03:05] Starting 'watch'...
[18:03:05] Finished 'watch' after 18 ms
CSS changed: C:\Users\Patrick\SkyDrive\4th Year\Final Year Project\Code\test\www\css\ionic.app.css
```

Figure 20- Ionic CLI

This command automatically opens a web browser with localhost: 8100, and gives a representation of how the application will look. Figure 21 represents how this page looks. Any changes to the code will cause it to automatically reload.

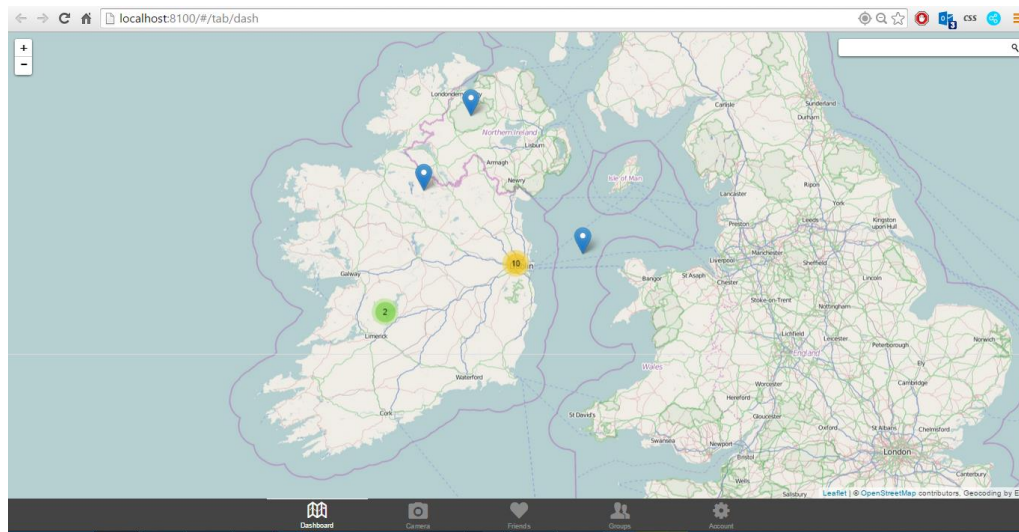


Figure 21 - Ionic Browser Page

Another useful feature built into Ionic is the “ionic serve –lab” command. This loads a page that contains a view of how the application works on both Android and iOS. This is an extremely useful feature when developing a cross platform application. Figure 22 demonstrates how this looks.

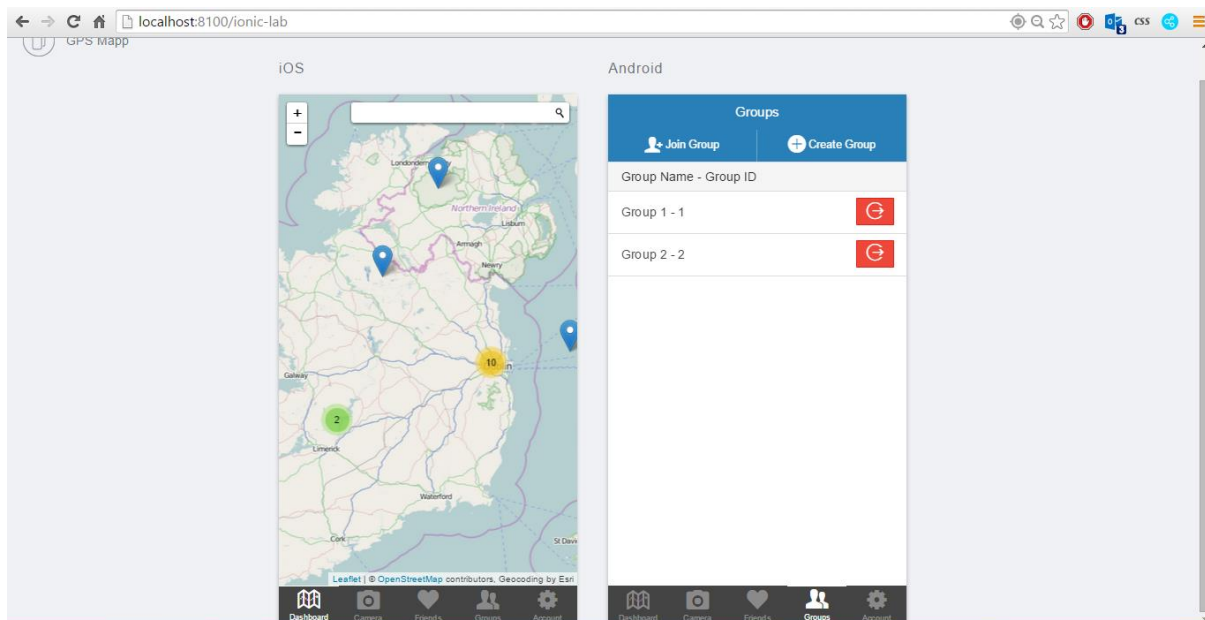


Figure 22 - Ionic Lab

Using this environment made it possible to develop the application in a fast and reliable way, and it gave constant feedback on how the application would look on both target platforms.

For the middle tier of the system, which consists of PHP, Sublime Text was also used. The PHP was then tested on a server by loading the page after uploading the file.

For the backend, which is a MySQL database, software called phpMyAdmin was used. [42] phpMyAdmin provides a Graphical User Interface which allows users to perform database operations easily. It also provides an easier way to edit existing data compared to using SQL commands. Figure 23 represents a screenshot of phpMyAdmin's dashboard.

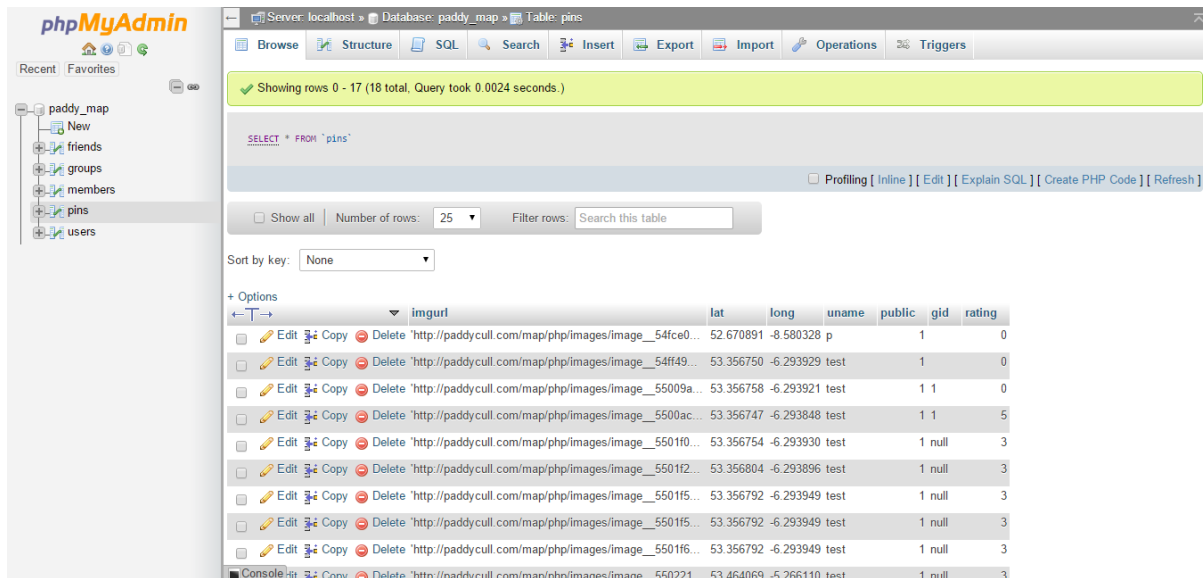


Figure 23 - phpMyAdmin

In the next section, we discuss the order in which the application was developed, and how it followed the chosen development process.

5.3 Developing the Application

As discussed in Chapter 4, the methodology that will be used is the Rapid Application Development (RAD) methodology. This methodology allowed the application to be modularised into separate parts. Following this, each part was developed separately and then integrated with the application.

The core part of the application is its ability to switch between screens fluidly. This is done using AngularJS to define the content area of the screen, and when a navigation button is clicked, only the content area gets reloaded so that the navigation bar remains constant. Here is how the nav view is defined within index.html. The rest of index.html has been removed for clarity.

```
<ion-nav-view></ion-nav-view>
```

And then views can be defined in separate HTML pages as follows;

```
<ion-view title="Account">
  <!-- Page content goes here -->
</ion-view>
```

The navigation bar is kept constant by defining it as 'abstract'. As explained by the AngularJS documentation, abstract pages are not called by normal means, they are called when one of their child pages are called. [29] The child states in this case are any of the pages within the application that require the navigation bar. The Angular code for this is;

// setup an abstract state for the tabs directive

```
.state('tab', {  
  url: "/tab",  
  abstract: true,  
  templateUrl: "templates/tabs.html"  
})
```

Once the page transitions were understood, the development of the proposed applications pages began. The application development has been modularised based on the different screens. These are discussed in the following sections. In each of the following sections, there are several prototypes for each screen. Each screen has a different functionality, and was modularised which is in-keeping with Rapid Application Development.

5.3.1 Dashboard – The Map Screen

Following the modularisation approach as part of Rapid Application Development, the first part of the application that was developed was to be the map screen. The rest of the application revolves around this screen, so it was given priority. The map screen uses an iframe to display the map. This iframe takes up the entire view.

Prototype One

In one of the first prototypes, the iframe was set simply using the "src" attribute of an iframe. This is how it was set;

```
<iframe style="width:100%;height:100%" src="http://www.paddycull.com/map"></iframe>
```

This is simple code, and works well. As the application was further developed, the map required the user username, so it could load friend's pins in a different colour. This is reflected in the current setup.

Prototype Two

In the current version, the iframe makes use of the "ng-src" attribute. This is an angular attribute, and allows the iframe source to be set dynamically. In the proposed application, the URL of the source is used to POST the users username. This username is then read using the PHP script, and loads the user's pins in a different colour. This is how the iframe is currently setup;

```
<iframe style="width:100%; height:100%" ng-src="{ {mapurl} }"></iframe>
```

The "mapurl" variable is set within the map screens Angular controller;

```
$scope.url = "http://paddycull.com/map/index.php?uname=" + $scope.username;  
$scope.mapurl = $sce.trustAsResourceUrl($scope.url);
```

The `trustAsResourceUrl` is required for security reasons. [30] This appends the user's username to the end of the URL, and the PHP loads this username as a POST variable, and can be used using a GET command in the PHP script.

```
$uname = $_GET["uname"];
```

And this variable is then used to query the database, and get both the friends pins and the public pins;

```
$myquery = 'select `imgurl`, `lat`, `long`, `uname` from pins where uname IN ( SELECT  
fname from friends WHERE uname = "' . $uname . '" )';
```

5.3.2 Camera – Adding Upload Functionality.

The next part of the application to be developed was the camera screen. It was decided that this was the second most important part of the application, as the application would become semi-functional once this part of the application was implemented. This part of the application took the longest time to develop out of any of the screens, as it involved a lot of different technologies interfacing with each other, and took a lot of fine tuning before the upload function was working.

Prototype One

The early prototypes were to simply allow a user to either take a photo or choose one from their photo library, and upload it to the server. This would also create an entry in the 'pins' table of the database so that it could be loaded onto the map.

For this functionality, the application needed to use the Cordova library to access the devices camera and gallery. This functionality was simple enough to implement, and did not take long. Problems arose when attempting to upload the photo.

As mentioned in Chapter 3, the Cordova documentation is quite poor, and using the `FileTransfer` method of the Cordova library caused problems. The code that was stopping the upload of the file is as follows.

```
var ft = new FileTransfer();  
ft.upload(imageURI, "http://paddycull.com/map/php/upload.php", win, fail, options);
```

This was following the `FileTransfer` documentation exactly, as can be seen in this screenshot in Figure 24 taken directly from the Cordova website. [31]

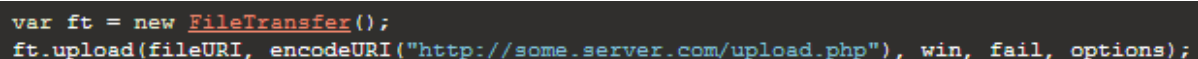
A screenshot of a code editor showing two lines of JavaScript code. The first line is `var ft = new FileTransfer();` and the second line is `ft.upload(fileURI, encodeURI("http://some.server.com/upload.php"), win, fail, options);`. The text is in a monospaced font with some color coding (e.g., `FileTransfer` is in red).

Figure 24 - Cordova FileTransfer Documentation [31]

This line of code went unnoticed for quite some time, and time was wasted checking server side scripts and configuration, as well as other JavaScript functions. The problem turned out to be an optional parameter missing from the upload function. At the end of the function a

parameter must be passed which allows uploads from all sources, which is required for this application. The new line of code is;

```
ft.upload(imageURI, "http://paddycull.com/map/php/upload.php", win, fail, options, true);
```

This small change unfortunately took a lot of development time to figure out. Once it was discovered though, the image was uploading correctly.

Prototype Two

After the image was uploading properly, the application made use of another of Cordova's functions, which allowed access to the devices geolocation. This would be used to insert a latitude and longitude into the database, and would be able to be loaded onto the map. This did not take long to implement, and the image was soon being successfully uploaded along with the devices co-ordinates. This made the application semi-functional, as it allowed for the uploading of photos onto the map.

At this point, the camera screen functionality was integrated with the map screen.

Prototype Three

After the Friends and Groups Functionality had been added, which are discussed in section 6.3.3, functionality that allowed a user to upload a photo from where it was originally taken from was added. This meant that a user could upload photos they have in their gallery to the location they were originally taken, while on holiday for example.

This involved reading an images EXIF data. EXIF data is data that is automatically stored with an image when it is taken. This applies to digital cameras as well as mobile phones. EXIF data contains lots of information, such as the device type, resolution, device orientation and even lens type. The information we are interested in is the geolocation information.

To extract the EXIF information from the images, a third party library called exif.js. [32] This allowed the required geolocation data from the image. There was a problem with this, however. The Latitude and Longitude are stored in degree format in EXIF data, and the proposed applications database stores the latitude and longitude using decimal notation. This required the extracted data to be converted before it was inserted into the database. This was done using the following function.

```
//This function converts the EXIF co-ordinates into decimal co-ordinates.
var toDecimal = function (number, direction) {
    //Convert it.
    var dec = number[0] + (number[1] / 60) + (number[2]/3600);
    //Round it
    dec = dec.toFixed(7);
    //Check if the decimal value should be minus or not.
    if(direction == 'S' || direction == 'W'){
        dec = dec*-1;
    }
    return dec;
};
```

This worked well, and the extracted data was able to be used to upload the image with the correct co-ordinates. Figure 25 shows how the upload screen looked during this prototype.

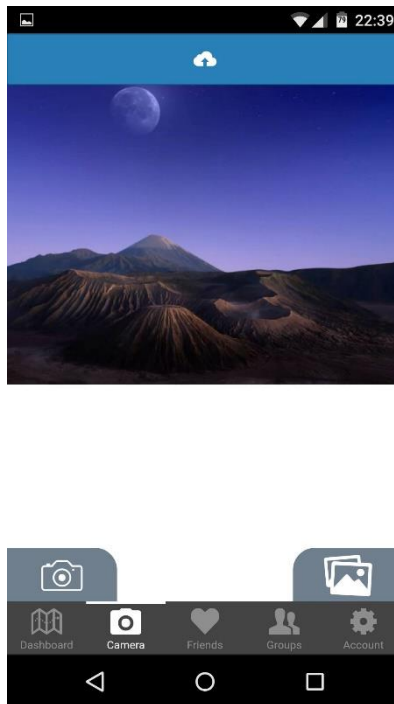


Figure 25- Choose Image Screen

The upload function could now upload photos from the users library from where the photo was originally taken, but a check still needed to be implemented that warns the user if there is no EXIF data available, and allow the user to place a pin on the map. This was delivered in the following prototype.

Prototype Four

In prototype four, the upload operation was expanded, so that the user could attach extra tags to the photo they want to upload. These options included adding a group, adding a tag to the photo, rating the photo and making the photo public or private.

This screen also contains a miniature map, which displays where the photo was taken if the photos EXIF geolocation data is available. If the data is not available, the map centres on where the user currently is. When the user clicks upload, and there is no EXIF data available, they are presented with a warning message, and they are asked to place a pin on the map, so the photo can be uploaded to that location. Figure 26 is how the extra upload screen looks.

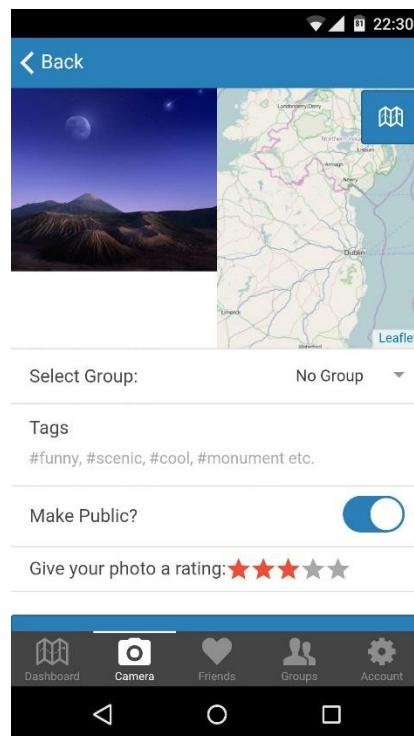


Figure 26 - Upload Screen

At this stage of the design, it was decided to solely use the EXIF data for image upload location. Preceding this, any photo that was taken from within the application would use geolocation information using the Cordova geolocation functionality. It now extracts the EXIF data, as photos taken from within the application have EXIF data as well. This made the upload faster, as the application did not need to wait for the user to be located, which took two or three seconds on average.

This prototype is a release candidate, as it is fully functional and has an aesthetically pleasing design. The only extra functionality that may be added is a progress indicator of the image upload, which would be beneficial for users with a slow internet connection.

5.3.3 Friends and Groups Screen

The friends and groups screens are similar in functionality, so they will both be covered in this section.

Prototype One

The first prototype for both groups and friends simply displayed the user's friends or groups, with no additional functionality. It simply queried a PHP script, which took the users username and ran it against the database. The dataset that was returned from the database was formatted into JSON format, so that the AngularJS code could load the data into the screen. The following code snippet is from the friend's page, before additional functionality was added to the code.

```

<ion-list>
  <ion-item ng-repeat="pal in pals | filter:searchFilter | orderBy:'fname'" item="pals">
    {{pal.fname}}
  </ion-item>
</ion-list>

```

This code simply displayed the user's friend's usernames. The code for the groups screen is essentially the same in this prototype. The AngularJS code that runs the PHP script is as follows;

```

//Get friends of current user, and return query data on success.
var request = $http({
  method: "post",
  url: "http://paddycull.com/map/php/getfriends.php",
  crossDomain : true,
  data: {
    'username': $scope.username,
  },
  headers: { 'Content-Type': 'application/x-www-form-urlencoded' }
});
//On success, assign the data to the variable 'pals'
request.success(function(data) {
  $scope.pals = data;
});

```

Assigning the data to a scope variable makes it visible to the html page with the corresponding Angular controller.

Prototype Two

Prototype Two involved adding the functionality which allowed a user to add a friend, join a group, and create a group. Clicking any of these buttons brings up a modal window, which asks the user to enter the required details. This then checks if the action is possible, for example if the friend the user trying to add does not exist, or if the user is already friends with the user, or if the user tries to add themselves as a friend an error is thrown,.

It was important that these functionalities were unobtrusive, but clear. The use of a modal window prevents the need of a new page being loaded, and keeps the user experience fluid. Figure 27 shows how these modal windows look.

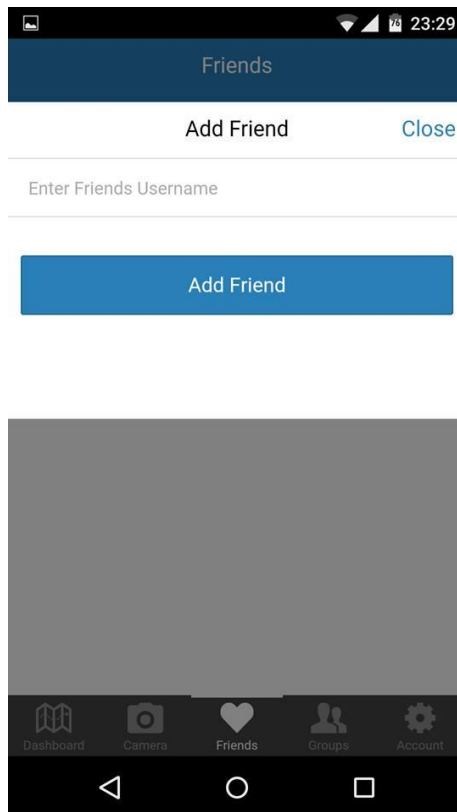


Figure 27 - Modal Window

This modal window also provides appropriate feedback to the use if there is an error with the attempted operation.

Prototype Three

After the data was being successfully inserted and loaded from the database, the friend and group names needed to link to a page that displayed the respective photos. This involved changing the existing page, developing another page to display the photos, creating a new Angular controller, and a new PHP script to load the photos.

The new HTML page(s) are simple, and look like this;

```
<ion-view title="{{ gname }} Photos">
  <ion-content has-header="true">
    <ion-list>
      <ion-item class="imageitem" ng-repeat="photo in photos">
        
      </ion-item>
    </ion-list>
  </ion-content>
</ion-view>
```

The new Angular controller calls a PHP script that returns the respective image URLs in JSON format, so they can be displayed on the screen.

Once these parts of the application were coded, clicking a friend or groups name brings up a page as shown in Figure 28.

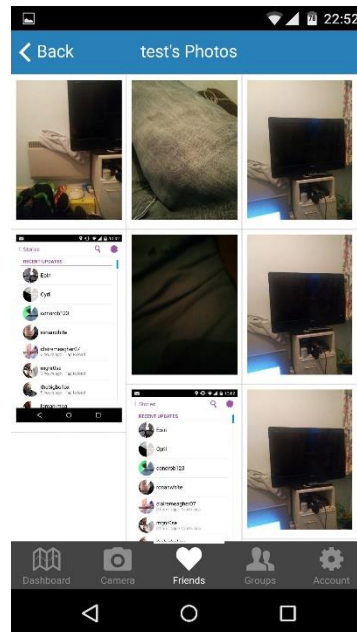


Figure 28 - Friend Photos

The friends screen was also changed, so that clicking a friends name was brings the user to the respective page. Friends pages are created dynamically, so that every friends page does not need to be stored, it is simply a different request to the database. The code for the friends screen now looks like this.

```
<ion-list>
  <ion-item href="#/tab/friend/{ {pal.fname} }" ng-repeat="pal in pals |
filter:searchFilter | orderBy:'fname'" item="pals">
    <div class="item-button-right">
      {{pal.fname}}
      <button class="button button-assertive" ng-
click="deleteFriend(pal.fname)">
        <i class="icon ion-close"></i>
      </button>
    </div>
  </ion-item>
</ion-list>
```

The “href” of each item list reroutes the user to the friend’s page. The friends name is passed into the Friends Detail controller. That controller then uses the name to look up the users images in the database, and returns the image URLs.

5.3.4 User Profile Screen

Prototype One

The users profile screen does not have much functionality. Because of this, only one prototype was needed. The page displays the users username, a button to view the users own photos, and a log-out button. The logout button clears the devices local storage, and redirects the user to the login screen. The local storage is used to store the user's username. This is also used in a check when the application is opened. If a username exists in local storage, then the user is considered logged in, and they do not have to go through the login process again. If the local storage is empty, then the login screen is opened when the application is booted.

5.4 Visual Design

The visual design of the application was kept in mind during the entire development process. As outlined in Chapter 2 when other applications were examined, it is important to have a clean, aesthetically pleasing user interface. Ionic helped with this, and different parts have been modified to create a more appealing design. The default colours of Ionic were changed. Figure 29 is an example of how the interface was modified to make it more appealing.

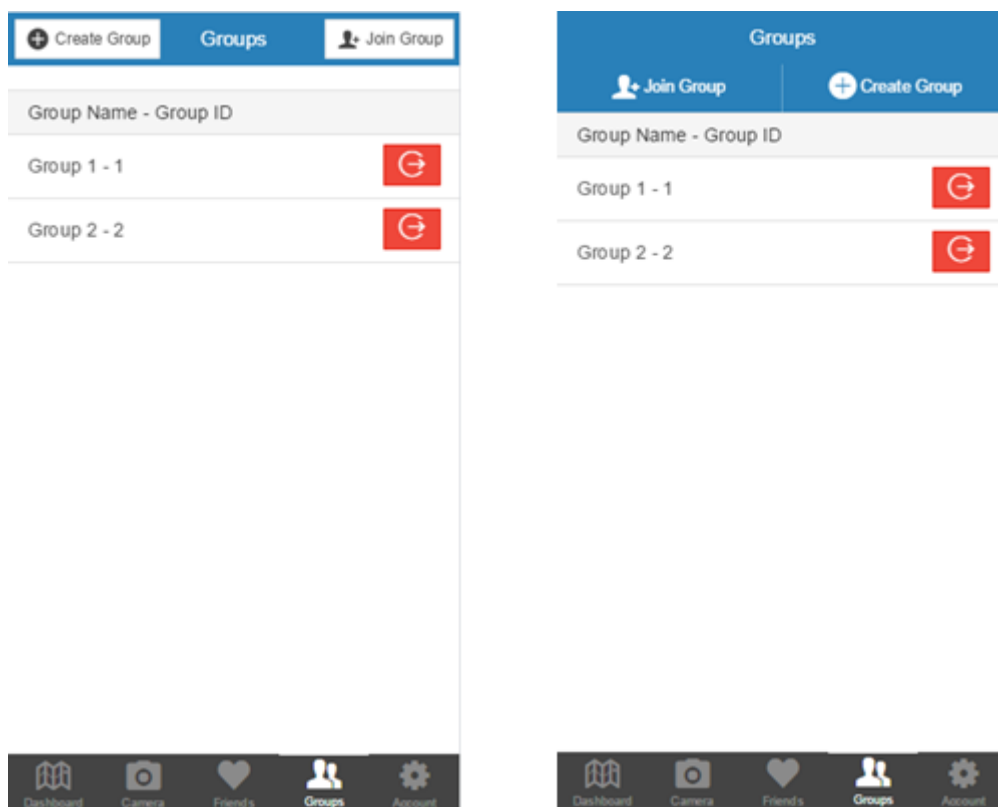


Figure 29 - Design Change

This design is more similar to the new trend 'Material Design', which is used by Google in its new operating system, Android Lollipop. [35] The application also uses smooth animations to navigate between screens.

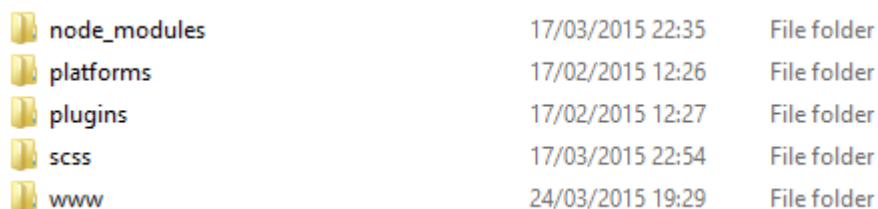
In the next section, the project file structure is discussed, for both the application code and the leaflet map code.

5.5 Project Structure

This section will discuss how the applications file structure is laid out. Any third party code will be clearly outlined below. There are two separate code structures that will be examined. The application itself, and the leaflet map which is hosted and displayed within an iframe in the application. The application code will be looked at first.

5.5.1 Core Application Structure

In the root directory of the application, there are several folders, as can be seen in Figure 30.




node_modules	17/03/2015 22:35	File folder
platforms	17/02/2015 12:26	File folder
plugins	17/02/2015 12:27	File folder
scss	17/03/2015 22:54	File folder
www	24/03/2015 19:29	File folder

Figure 30 - Core File Structure

Most of these folders are created by the software that was used to create the proposed application. Node_modules contains node modules that are required by the application. Platforms contains the different platforms that the application has been built for. This folder is created by Cordova. Plugins is also created by Cordova, and contains plugins that allow access to the devices native functionality, such as the camera and the geolocation. The scss folder contains SASS files that are used with the project. This folder is created by Ionic, and contains slightly modified files that effect the applications visual style. The folder which contains the custom code is the www folder. The next section looks at this folder in more detail.

The www folder

The www folder contains all of the custom code that was created for the proposed application. Figure 31 represents how the folder is laid out.



css	24/03/2015 19:50	File folder
img	26/01/2015 18:00	File folder
js	24/03/2015 19:53	File folder
lib	24/03/2015 19:06	File folder
templates	22/03/2015 21:11	File folder
index.html	24/03/2015 19:55	File folder

Figure 31 - www folder structure

Each of the following sub-sections will discuss what each folder contains.

CSS Folder

This folder contains all of the applications CSS files. It includes both custom CSS, and CSS which comes included with Ionic.

JS Folder

The JavaScript folder contains JavaScript files which were added or created for the proposed application. The files in this folder are “app.js”, “controllers.js” and “camera.js”.

The files “app.js” and “controllers.js” contain most of the applications JavaScript code. Camera.js is used to open the devices camera.

This folder also contains a folder called “thirdparty”, which contains the third party JavaScript file which was used in the development of the project. The third party file that it contains is “exif.js”, which is used to extract an images EXIF data.

Lib Folder

This folder contains files which came built into Ionic. They are third party, and were not modified.

Templates Folder

The templates folder contains all of the application screens. The files are HTML that were created for the application, and use custom code.

index.html

This file is used for the applications navigation bar. This file contains the navigation bar of the application. There is also a “nav-view” tag within this file, and the files from the “templates” folder are loaded within this tag when they are called.

Now that the core application file structure has been determined, the map code will be discussed in the following section.

5.5.2 Leaflet Map Code

The applications map screen contains an iframe which displays a map with image pins. This map uses a separate file structure which will be discussed in this section.

The folder structure is shown in Figure 32.



css	23/03/2015 14:09
images	22/03/2015 16:55
js	24/03/2015 19:05
php	22/03/2015 19:31
createMap.php	12/03/2015 21:26
index.php	24/03/2015 19:30
showPinMap.php	22/03/2015 18:51

Figure 32 - Map Folder Structure

In the following sections, each individual folder will be discussed.

CSS Folder

The CSS folder contains CSS files which were used for the map screen. The only file within this folder which was custom coded is the 'style.css', each of the other files are used by third-party libraries.

Images Folder

This folder contains images that are used for the map pins.

JS Folder

This folder contains all JavaScript files which are used by the application. This folder contains only two custom made JavaScript files; "load-markers.js" and "add-marker.js", all other files are third party JavaScript files and libraries.

PHP folder

This folder contains a multitude of custom made PHP files. These files are used by the application to interact with the MySQL database. These files are the middle-tier of the application.

Root PHP files

The root PHP files are used to display maps in different situations. The index.php file is the main map which is used for the applications map screen. The createMap.php is used when user wants to create a pin on the map to attach an image to. The showPinMap.php is used to display the user's image location when they are uploading it.

6. Testing and Evaluation

5.1 Introduction

In order to evaluate the functionality of the application, it is important to perform critical, unbiased testing on the application functions. In order to fully evaluate the system, both White box and Black box testing will be carried out. White box testing tests the code and inner workings of the application, while black box testing tests the user interaction with the application.

The same heuristics which were applied to the existing applications in Chapter 2 of this report will also be applied to the application, to provide a comparison.

5.2 White Box Testing

White box testing involves testing the internal workings and code of an application. [20] The tester must have an intimate knowledge of the code mechanics of the application, and full access to the database.

The white box testing for the proposed application involves several different checks. Which include;

- Check that data is inserted correctly to the database.
- Ensure the correct data is returned from the database.
- Ensure images are uploaded correctly, and the correct URL is inserted into the database.

We will discuss each of these checks in detail in the following sections.

5.1.1 Database Entries

When a user uses the application, there are several actions which entail entering data into the database tables. Creating an account, joining a group, uploading a photo and adding a friend are examples of when entries need to be made. The expected result for each of these actions is that the data is correctly entered into the database. Database entries are a vital aspect of the application, so it is important that each entry pass the test.

User Action	SQL Code	Result
Create Account.	INSERT INTO users (uname,password,email) VALUES ("JohnDoe","pwd","johndoe@email.com")	PASS
Join Group.	INSERT INTO members (gid,uname) VALUES ("GroupID","JohnDoe")	PASS
Create Group	INSERT INTO groups (gid,gname) VALUES ("GroupOneID", "Group Name")	
Add Friend.	INSERT INTO friends (uname,fname) VALUES ("JohnDoe","JaneDoe")	PASS
Upload photo.	INSERT INTO pins (imgurl, lat, lon, uname, public, tag, gid, rating) VALUES ('http://paddy.cull.com/images/imageurl.JPG', '6.534563', '54.456432', 'JohnDoe', '1', 'scenic', 'GroupID', '4')	PASS

For some of the above entries, an additional check is required to determine if the data should be entered or not. We look at these checks in the following section.

5.1.1.1 Additional Checks for Database Entries

Creating an account or adding a friend.

When creating an account, an additional check is required to see if the username is already in use. We also need to perform the same check to ensure a user exists when a user tries to add a friend within the application. The PHP code for this check is;

```
$qry_em = 'select count(*) as cnt from users where uname = ' . $uname . ' ';;  
$qry_res = mysqli_query($con, $qry_em);  
$res = mysqli_fetch_assoc($qry_res);
```

This query will return 1 if the user already exists, and 0 if no user exists with that username. To test this, we try to re-enter a user with the same username as John Doe, above. The test should fail, as a user with these credentials already exists in the table. We will also check if a user is able to add a friend with a username that does not exist.

Joining or creating a group.

When a user decides to join or create a group, we must check if the group exists or not. When joining a group, the group must exist before the user can join that group. When creating a group, we must check that a group with that Group ID does not already exist. The PHP code to check if a group exists is.

```
$qry_em = 'select count(*) as cnt from groups where gid = ' . $gid . ' ';;  
$qry_res = mysqli_query($con, $qry_em);  
$res = mysqli_fetch_assoc($qry_res);
```

To test this, we will try to create a group with a GroupID that already exists, and we will also try add a user to a group which does not exist.

Additional Checks Results.

Action	SQL Code	Result
Create User.	INSERT INTO users (uname,password,email) VALUES ("JohnDoe", "pwd", "johndoe@email.com")	FAIL – A user with this username already exists
Add Friend.	INSERT INTO friends (uname,fname) VALUES ("JohnDoe","NonUserID")	FAIL – No user with this username exists.
Create Group.	INSERT INTO groups (gid, gname) VALUES ("GroupOneID", "This Group")	FAIL – a group with this ID already exists
Join Group.	INSERT INTO members (gid,uname) VALUES ("NonGroup","JohnDoe")	FAIL – No group with this ID exists.

The results confirm that the additional checks are working, as each test has failed as expected.

5.1.2 Correct Information Returned from Database

In the application, there are multiple cases where the user needs to access data from the database. It is extremely important that a user only has access to data that they should have access to. Examples of what data a user needs access to are the user friends, friends pins, public pins, groups a user belongs to, group photos, users own photos. The following table shows each of these tests and their results.

Action	SQL Code	Result
Get Friends List	SELECT fname FROM friends WHERE uname ="JohnDoe"	PASS
Get Groups List	SELECT GID, gname FROM groups WHERE GID IN (SELECT gid from members WHERE uname ="JohnDoe")	PASS
Get User Photos	SELECT imgurl FROM pins WHERE uname ="JohnDoe"	PASS
Get Group Photos	select imgurl from pins where gid = "GroupOneID"	PASS
Get Friends Pins	SELECT `imgurl`, `lat`, `long`, `uname` from pins where uname IN (SELECT fname from friends WHERE uname ="JohnDoe")	PASS
Get Public Photos	select `imgurl`, `lat`, `long`, `uname` from pins WHERE uname NOT IN (SELECT fname from friends WHERE uname ="JohnDoe") AND public=1'	PASS
Group nearby Pins when zoomed in	Not Implemented Yet.	FAIL

5.1.3 Ensure Images are Uploaded Correctly

When a user uses the application to upload the photo, it not only a database table entry which needs to be completed. The image also needs to be stored on the server, and then this image URL can be inserted into the database table.

In the server side script, the uploaded image is given a random name, and is prefixed with the string "image_". The PHP code is as follows.

```

if ($_FILES["file"]["error"] > 0)
{
    echo "Return Code: " . $_FILES["file"]["error"] . "";
}
else
{
    $new_image_name = 'image_' . '_' . uniqid() . '.jpg';
    move_uploaded_file($_FILES["file"]["tmp_name"], "images/" . $new_image_name);
}

```

The “if” statement checks if the file has been sent correctly. If the file has been received, the script creates a new, unique image name, and inserts it into the images directory. This image URL can then be inserted into the database.

To test that the upload is working, an image was uploaded from the application, and the server’s image directory was checked. Figure 33 shows the format of the image file names, and the location of the directory.

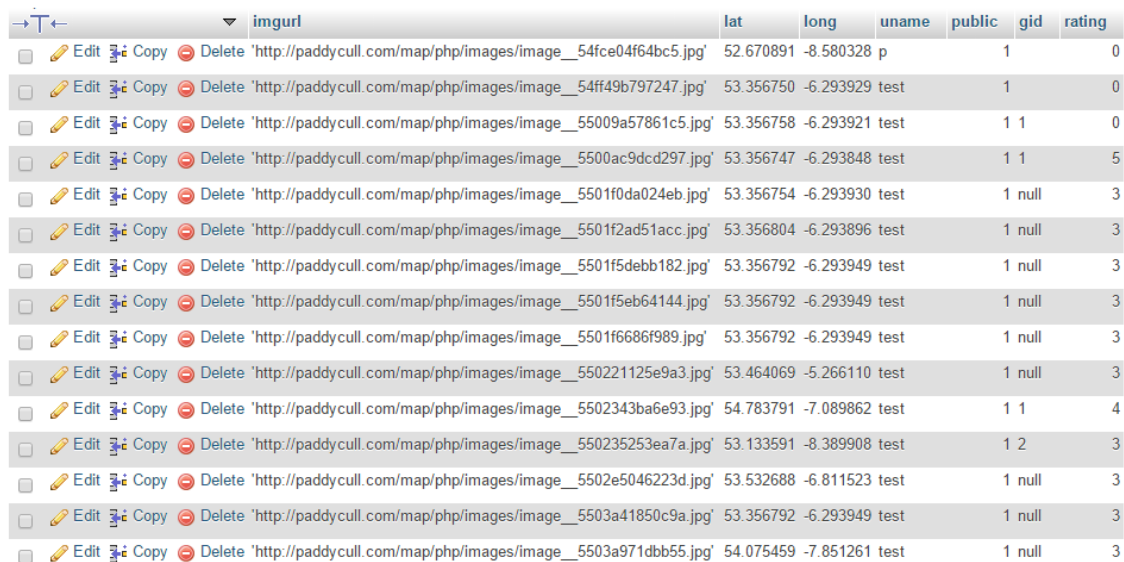
Index of /map/php/images

- [Parent Directory](#)
- [image_54fce04f64bc5.jpg](#)
- [image_54ff49b797247.jpg](#)
- [image_55009a57861c5.jpg](#)
- [image_5500ac9dcd297.jpg](#)
- [image_5501edfaf3345.jpg](#)
- [image_5501f0da024eb.jpg](#)
- [image_5501f2ad51acc.jpg](#)
- [image_5501f5debb182.jpg](#)
- [image_5501f5eb64144.jpg](#)
- [image_5501f6686f989.jpg](#)
- [image_550221125e9a3.jpg](#)
- [image_5502343ba6e93.jpg](#)
- [image_550235253ea7a.jpg](#)
- [image_5502e5046223d.jpg](#)
- [image_5503a41850c9a.jpg](#)
- [image_5503a971dbb55.jpg](#)

Apache Server at paddycull.com Port 80

Figure 33 - Image Directory

The insertion to the database is also working correctly, as demonstrated when the pins tables is checked, as shown in Figure 34. Each of the inserted images correspond to an image in the image directory.



		id	imgurl	lat	long	uname	public	gid	rating
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__54fce04f64bc5.jpg'	52.670891	-8.580328	p	1		0
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__54ff49b797247.jpg'	53.356750	-6.293929	test	1		0
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__55009a57861c5.jpg'	53.356758	-6.293921	test	1	1	0
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5500ac9dcd297.jpg'	53.356747	-6.293848	test	1	1	5
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5501f0da024eb.jpg'	53.356754	-6.293930	test	1	null	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5501f2ad51acc.jpg'	53.356804	-6.293896	test	1	null	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5501f5debb182.jpg'	53.356792	-6.293949	test	1	null	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5501f5eb64144.jpg'	53.356792	-6.293949	test	1	null	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5501f6686f989.jpg'	53.356792	-6.293949	test	1	null	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__550221125e9a3.jpg'	53.464069	-5.266110	test	1	null	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5502343ba6e93.jpg'	54.783791	-7.089862	test	1	1	4
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__550235253ea7a.jpg'	53.133591	-8.389908	test	1	2	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5502e5046223d.jpg'	53.532688	-6.811523	test	1	null	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5503a41850c9a.jpg'	53.356792	-6.293949	test	1	null	3
<input type="checkbox"/>	Edit Copy Delete		'http://paddycul.com/map/php/images/image__5503a971dbb55.jpg'	54.075459	-7.851261	test	1	null	3

Figure 34 - Pins Table

5.3 Black Box Testing

Black Box Testing tests the functionalities of the application, without considering the internal workings, or the coding behind the functionality. [22] The tester knows what the application is expected to do, but it is not necessary for the tester to understand the code behind it. There are several different areas to cover with black box testing, which we will go through in the following sections.

5.2.1 Fundamental Functionality

The fundamental functionality includes things like opening the application, traversing the screens, and ensuring correct data is displayed to the user.

Action	Expected Result	Actual Result
Click application icon	Application Opens	PASS
Click navigation icon	Respective page loads	PASS
View map screen	Map with pins loads	PASS
Click Friends Screen	Users friends displayed	PASS
Click friends profile	Friends photos displayed	PASS
Click Groups Screen	User groups displayed	PASS
Click group	Group photos displayed	PASS
Click Account	Account page displayed	PASS
Click View Photos	User photos displayed	PASS

After the fundamental functionality of the application has been determined to be working properly, we now need to evaluate the more complicated functionalities of the application.

5.2.2 Main Functionality

In this section, we test the main functionalities of the application. This functionality includes logging in, adding a friend, adding a photo, creating groups and many others. The following table shows the results of the testing.

Action	Expected Result	Actual Result
Log in	Map screen displayed	PASS
Login with incorrect credentials	Error message displayed	FAIL – Password functionality has not been implemented at time of testing.
Click Create Account	Create account screen displayed	PASS
Log Out	Redirect to login screen	PASS
Click on a pin	Image and username displayed	PASS
Click on pin image	Image displayed in full screen	PASS
View map	Friends and group pins appear in different colour	PASS
Fully zoom into map	Closely situated pins are grouped into one pin.	FAIL – Pins appear where they are uploaded, and don't group together.
Click 'Add a friend' button	"Add friend" modal window appears.	PASS
Attempting to add friend	User is given appropriate feedback when adding a friend, i.e. success or given the reason of a failure.	PASS
Deleting a friend	Friend is removed after a confirmation dialog.	PASS
Clicking 'Create Group' button	"Create Group" modal window appears	PASS
Attempting to create group	User is given appropriate feedback when creating a group, i.e. success or given the reason of a failure.	PASS
Clicking 'Join Group' button	"Join Group" modal window appears	PASS
Attempting to join group	User is given appropriate feedback when joining a group, i.e. success or given the reason of a failure.	PASS
Click leave group button	User is removed from group after a confirmation dialog.	PASS
Choose photo to upload	Image taken by camera, or taken from gallery is displayed in full screen.	PASS

Action	Expected Result	Actual Result
Click "Upload" on image selection screen	Redirect to the upload page. Upload page should contain the image thumbnail and available options for the user.	PASS
Click "Upload" on upload screen	Upload image, while providing feedback on progress. Redirect to map when done.	PARTIAL PASS – Feedback on upload progress is not fully complete. Image uploads correctly and alerts with "Done" when complete.

5.4 Nielsen's Heuristics

As Nielsen's Heuristics were applied to the similar applications in Chapter 2 of this report, it is important that the same heuristics are applied to the proposed application.

	Test	Score
1	Visibility of System Status	10
2	Match between system and the real world	10
3	User control and freedom	9
4	Consistency and standards	9
5	Error prevention	10
6	Recognition rather than recall	8
7	Flexibility and efficiency of use	0
8	Aesthetic and minimalist design	10
9	Help users recognize, diagnose, and recover from errors	6
10	Help and Documentation	0
	Total	72/100

The final score of the application is 72/100. This is mainly brought down by the fact two sections scored zero.

Results Evaluation

- 1) Every screen is clearly marked with the navigation bar, and most pages also have headings, so the user always know where they are within the application.
- 2) The application uses intuitive icons, as well as simple words, so it is extremely easy for the user to understand.
- 3) Every sub-level screen has a clear back button on the top of the screen.
- 4) Each icon is clearly marked, and usually paired with text.
- 5) Every time a user tries to commit an important action, they are presented with a confirmation dialog. That is why the application scored 10/10
- 6) The application has icons which are easy to understand, but there are a few options within the application which may cause a few users to take some getting used to, such as adding tags or groups to a photo.

- 7) There are no customisation options within the application, so this heuristic has been given a score of zero. Customisation options are planned though, such as changing the look of the map.
- 8) The design of the application is clean and minimalistic, and very little words are used.
- 9) There are no checks in place to tell the user if they are not connected to the internet. There is, however, an error check when a user tries to upload a photo and it fails. There is also error checking in place when a user tries to add a friend, or join or create a group.
- 10) No help or documentation has been created for the application yet, but this is easy to create, and will be implemented at a future date

5.5 Unit Testing

As the proposed application is a mobile application, it is important that the application works well across a wide variety of devices. To confirm this to be true, the device was installed and tested on the following devices.

Device	Operating System	Result
LG Nexus 5 (Smartphone)	Android Lollipop 5.0	Fully functional
Asus Nexus 7 (Tablet)	Android KitKat 4.4	Fully Functional
Amazon Kindle Fire HD (Tablet)	Amazon modified Android 4.04	Fully functional
Samsung Galaxy S5 (Smartphone)	Android KitKat 4.4	Fully functional
Samsung Galaxy Trend Mini (Smartphone)	Android JellyBean 4.2.2	Fully Functional

For each of the above, the application worked and displayed exactly as intended. In contrast to the research in chapter three, in regards to HTML5 versus Java performance, the application ran extremely smoothly on all devices. This testing included old devices which were slow to boot the majority of applications, but this application was extremely quick to boot in comparison, which was an encouraging result. This is due to the fact most of the processing is done on the server, and the client device only needs to load simple HTML pages.

7. Conclusion

7.1 Intro

This chapter covers a summary of the entire project, and whether or not the objectives of the project have been met. This chapter also discusses any future work that is planned for the application, and a personal reflection on the project.

The next section gives a summary of each of the previous chapters, and gives key points that arose from each chapter.

7.2 Summary of Findings

Developing this application was a long process, and involved many different stages to be completed. Some steps needed to be completed before others could be started, and some needed to be completed concurrently.

Chapter One - Introduction

In the first chapter of this report, the aim of the project was briefly discussed, and the core aims and objectives for the application were given. Chapter One also covered background research for the proposed application, and from that several key points arose. These key points included;

- Social media is being used more regularly.
- Smartphone usage is increasing.
- Teenagers and young adults use social media and smartphones the most.
- Photo sharing is becoming more popular.

This gave a clearer understanding of the target market for the proposed application, and let an informed decision be made on whether or not the application was likely to be used. In knowing that the proposed application had the potential to be widely used, more specific research on similar applications was conducted in chapter two.

Chapter Two – Research and Evaluation

In chapter two, three applications that are similar to the proposed application were chosen, and a set of heuristics were applied to each. The three applications were Snapchat, Whisper and PhotoMap. The heuristics which were deemed most appropriate were Nielsen's Heuristics. To ensure the review was critical and unbiased, each application was rated from one to ten for each heuristic, and the results were studied. These results allowed the existing user requirements of the application to be refined, and gave rise to new ones. The most important findings were;

- The application must be easy to use.
- The user interface must be aesthetically pleasing.
- Users must be informed of errors.
- Users must be presented with confirmation dialogs for certain actions.

These requirements were not immediately apparent from the research done in chapter one, so it was vital that these three other applications were examined.

Once the requirements were refined enough, the technologies which could potentially be used in the application were reviewed and compared in detail in chapter three.

Chapter Three – Technology Used

Choosing the correct technologies to use for the proposed application is perhaps the most important part of the decision process, so it was vital that the technology best suited for the application was chosen. Due to the proposed applications nature, it was clear that the majority of the processing could be done on the server side, and then sent to the client device. This helped in the decision making for the front end, as the downside of HTML5 when compared to Java is that it can be slower at processing. Once HTML5 was decided on for the front end, development tools and languages which would be used with HTML5 were reviewed and explained.

After deciding on the front end technology, the middle tier of the application was chosen. The middle tier consists of PHP scripts which interface with the client and database, and transfers data between the two. The middle tier was extremely important for the proposed application, as most of the processing would be done here. PHP was chosen over Ruby on Rails as PHP is faster, and there is more documentation online.

The final piece technology to choose was the backend. The backend would store user data and pin locations, and MySQL was chosen. PostgreSQL was also reviewed, and would have been used if the database required geospatial queries, which it did not.

Chapter Four – Design

In this chapter, the Rapid Application Development model was decided on to be used for the proposed application. To ensure the decision was informed, another methodology, the Spiral Development Model, was also reviewed. After comparing the advantages and disadvantages of each, Rapid Application Development was decided to be most suited for the proposed application.

This chapter also included the system architecture. The architecture of the proposed system is a three tier architecture. The diagram includes the technology that is used at each layer.

It was important to get an idea of how the system would work logically. For the database, and Entity Relationship Diagram was created. This provided a much clearer idea of how to implement the database, and the relationships each table would have with one another. Doing this saved implementation time, as there was already a clear idea of how the database would work. Use Case Diagrams were also developed for the application. These provide a visualisation of what the user requirements are, and allow the requirements to be understood at a glance.

Chapter Five

After the systems technology had been decided on in chapter three, and the logical structure had been studied in chapter four, it was now time to implement the system. This chapter covered all the aspects of the development, and covered areas such as;

- The development environment.
- Developing each screen of the application
- The visual design of the application.

Sublime Text was used in conjunction with Ionic and Cordova to develop the front end and middle tier of the application, and phpMyAdmin was used to manage the MySQL database at the backend.

Chapter five also discussed the problems with developing the application, and how these problems were overcome. The biggest challenge was developing the upload screen of the application, as many different technologies needed to interface for it to function properly.

Chapter Six

After the proposed system was implemented, testing on the system was required. This testing included;

- White Box testing
- Black Box testing
- Nielsen's Heuristics applied to proposed application
- Unit Testing

The white box testing involved running database queries, and ensure the correct result set was returned. All of the white box tests passed successfully. The black box testing was not so successful, as there were a few tests which failed as some functionality has not been implemented yet.

Nielsen's Heuristics were also applied, and the results were mostly positive. A few sections scored 0/10 though, as they have not been implemented yet.

The Unit Testing involved testing the application on several different devices. The application worked as intended on each of the tested devices.

7.3 Aims and Objectives

In chapter one, the aims and objectives were outlined, as follows;

Feature	Implemented?
Core Functionality	
Allow users to create an account.	Yes
Let photos be uploaded to the map.	Yes
Let users browse the photos on a map.	Yes
Let users add friends and create groups.	Yes
Secondary Functionality	
Let users upload pre-existing photos to where they were taken.	Yes
Allow photos to be made private.	Yes
Let users delete their photos from the map.	No
Add tags to photos.	Yes
Allow users to rate photos.	Partially
Implement a filtering/grouping system for the pins.	No

While the majority of the functionality was implemented, there were some that were not completed on time. All of the core functionality did get completed on time however, and the application is fully functional.

One of the functionalities that was not implemented was the ability to delete user's photos. This functionality was not overly complicated, but other more important features took priority, and there was not time to implement it.

Allowing users to rate photos is partially implemented. A user can rate a photo of their own when they are uploading it, but not other user photos. Again, this is due to other features taking priority, and not enough time was left to implement this feature.

The final feature that was not implemented was grouping the photos together and filtering some pins out. The reason this was not implemented is due to the fact all other core functionalities needed to be fully functional before this feature was implemented. Time was dedicated to developing a grouping system, but it was not developed before the delivery date unfortunately.

7.4 Future Work

As so much development and research has already gone into the application, it will be completed after the project submission. While the application is fully functional, there is still some work to be done before it can be considered complete.

Grouping System.

First and foremost is the grouping system for nearby pins. Some time has already been spent developing this feature, and it will be the next feature that is implemented in the proposed application. The grouping system will make pins that are in close proximity appear as one pin, so that the user can easily view each photo, and the map is not overloaded with pins.

An approach which may provide this functionality is to use a spatial database, which would allow for spatial queries.

There would be a JavaScript check which checks what zoom level the user is currently at on the map screen. Once the user is zoomed in a certain amount, the map would send a query with the current view's map boundaries to the database. The query would take all pins within the current view of the user's screen, and then loop through the pins, and group together pins that are close together. Then the query would return an array of images URLs that are included in the grouped pins, and the grouped pin would be represented as a folder icon. Clicking on the pin would result in the images being displayed in an image gallery.

Improve Rating System.

When the pins are being grouped together sufficiently, a rating system will be implemented, so users can rate any photo on the map. This will make the application feel more interactive. This rating system can then be used to improve a filtering system for the aforementioned pin-grouping feature.

This functionality could be implemented by bringing up a rating system as well as the full-screen image when a user clicks on an image. It would be implemented using JavaScript and PHP. Clicking on a rating would send the rating to a PHP file, which would insert the rating into the corresponding photo's average.

There would be problems with dealing with a user who has already rated a photo, and may require a separate table called "ratings" for each user. The table would contain their username, the rated images URL, and the rating they gave. This would prevent a user from creating several ratings for the same photo.

Password functionality.

Before the application is released, it is vital that a secure password system is implemented. The application in its current state does not store the password which is entered, but simply checks if the username exists when logging in, and the user gets logged in as that user. This is a major security flaw, and needs to be implemented as soon as other functionality is completed.

Password functionality will involve implementing a hashing system so passwords are never sent over the network as plaintext.

Login with other social media sites.

Another aspect of the application that should be developed further is the ability to login with other social media sites, such as Facebook or Twitter. This will streamline the login process, and using the Facebook friends API could make it easier to find friends using the application. This could also lead to user profiles being improved upon, with things such as user country, a profile picture and other details being stored about the user. This will make the application more involving and personal.

Developing a website.

Creating a website is also a feature which would be interesting to implement. As the application was developed in HTML5, it should be easy to convert it to a functional website.

The user interface would need to be changed, and the upload function will need to be recoded for use on a desktop, but the core functionality is already in place. Some users may find it better to browse the map with a computer screen, and it would be easier to upload images in bulk from a computer as opposed to a smartphone.

7.5 Personal Statement

When the college year started in September, I was concerned that I was not skilled enough to be able to develop a project to a good standard. I believed that whatever I ended up submitting would be of low quality, and that I might be able to just scrape a pass if I tried my best. I was happy with the project idea that I chose though, and pitched the idea to friends both in my Computer Science course and outside it. I received positive feedback, and that helped motivate me to try and develop the application to a good standard.

As time went on, and I researched the different technologies that were available to develop application with, my confidence in my ability grew. I discovered tools like Cordova and Ionic, which allowed me to develop the application in HTML5. HTML5 is one of my favourite programming languages, and after I put more research into it, I found out that it was not only possible to make my application in HTML5, but it had advantages over developing the application in Java as well, so my decision was justified. This was a massive confidence boost, and I began to realise that I would be able to develop a good project if I worked hard at it. My coding skills have never been the best, but I studied the chosen technologies, completed online tutorials, and became more and more comfortable with each of them.

Once development started on the project, and I was able to finally see the results in an actual interactive mobile application, it gave me even more encouragement. I have always been more interested in the visual result of code, which is why I love HTML5 so much. That's not to say I do not appreciate the importance of the efficiency of the code behind the functionality, I just enjoy seeing the results of my work. The Ionic live-reload feature, which creates a web page that automatically reloads when any of the source files are changed. This allowed me to implement features quickly and see the results immediately, the code did not even need to be compiled. The only time I needed to actually compile the code was when I wanted to see the application running on a mobile device, which I done at different stages to ensure the application looked the same as it did in my laptop browser.

As I had no prior experience with some of the technologies, namely AngularJS, Ionic and Cordova, a lot of my development time was actually spent looking up how to do simple things using those technologies, and trying to learn how they worked. As time went on though, my skills improved, and I spent less time looking up simple things, and more time implementing important features.

As the deadline for the application came closer and closer, I began stepping up my devotion to the project. I spent almost all of my free time working on the application and trying to implement the extra features before the deadline. I began to regret not starting the project earlier, as I had left it quite late before I actually started to code the application. It was too late to change that at this stage, so I knuckled down and just worked harder than I had ever done before.

Once the application began to take shape, and the core functionalities were implemented so that the application was semi-functional, I began to feel more proud of what I'd achieved. This gave me another motivational boost, and I was determined to implement as much functionalities within the timeframe as I possibly could.

Within the last month, I've never worked harder. It has made me appreciate how easy my previous college years were in comparison to the sheer workload of this year. I genuinely believe that it has made me a harder worker in general, as it has become almost normal for me to spend every spare minute working on the project. I am sure that this will benefit me in the long run, as I look forward to working after I am finished college.

If I was to change anything about how the project went, it would be to start much earlier. I did not really appreciate the amount of work that would be required, and before the Christmas break I did not put as much effort as I should of into the application. If I had started earlier, I believe I would have been able to complete all the features that are included in the 'Future Work' section of this chapter.

Although not every feature of the application was implemented in time, I am still extremely proud of what I have achieved. As I said, at the start of the year I was worried if my application was going to work at all, but now I have delivered an application which is fully functional. This is a massive personal achievement for me, as coding has never been my best attribute. I am determined to finish the application off, as I truly believe it is a good idea for a social media application, and I'd love to see an application of mine being used.

Working on the project has been invaluable to me. The huge workload has fundamentally changed my work ethic, and I know it will help me keep the hard-working mentality going forward in both college and work life. I have developed new skills and learned new languages, such as AngularJS which is huge at the moment. I have also become much more familiar with other languages which I knew, HTML5, PHP and SQL. I learned how to each of these technologies interface together, and I know I will be able to apply the same type of thinking to different technologies.

8. References

- [1] Duggan, M., & Smith, A. (2013). Social media update 2013. *Pew Internet and American Life Project*.
- [2] Madden, M., Lenhart, A., Cortesi, S., Gasser, U., Duggan, M., Smith, A., & Beaton, M. (2013). Teens, social media, and privacy. *Pew Internet & American Life Project*.
- [3] Duggan, M. (2013). Photo and video sharing grow online. *Pew Internet*.
- [4] Madden, M., Lenhart, A., Duggan, M., Cortesi, S., & Gasser, U. (2014). Teens and Technology 2013 (Washington, DC: Pew Research Center, 2013).
- [5] Smith, A. (2014). Smartphone ownership 2013. Pew Internet & American Life Project, 5 June 2013.
- [6] TripAdvisor.org, (2013) <http://ir.tripadvisor.com/releasedetail.cfm?releaseid=808058>
Date Visited: December 2013
- [7] Kochan, S. G. (2011). *Programming in Objective-C*. Addison-Wesley Professional.
- [8] Android Software Development Kit. (n.d.). Retrieved March 16, 2015, from <http://developer.android.com/sdk/index.html>
- [9] Meier, R. (2012). *Professional Android 4 application development*. John Wiley & Sons.
- [10] Telerik (2013). The HTML5 vs. Native Debate is Over and the Winner is... *HTML5 Global Developer survey*
- [11] Apache Cordova. (n.d.). Retrieved March 16, 2015, from <http://cordova.apache.org/>
- [12] "Apache Cordova :The Framework Formerly Known as PhoneGap" (n.d.). Retrieved March 16, 2015, from <http://www.risingj.com/archives/267>
- [13] Ionic Framework (n.d.). Retrieved March 16, 2015, from <http://ionicframework.com/>
- [14] HTML enhanced for web apps! (n.d.). Retrieved March 16, 2015, from <https://angularjs.org/>
- [15] Usage statistics and market share of PHP for websites. (n.d.). Retrieved March 16, 2015, from <http://w3techs.com/technologies/details/pl-php/all/all>
- [16] MySQL (n.d.). March 6, 2015 from <http://www.mysql.com>
- [17] Valade, J. (2007). *PHP and MySQL for Dummies*. John Wiley & Sons.
- [18] Leaflet - an open-source JavaScript library for interactive maps. (n.d.). Retrieved December 9, 2014, from <http://leafletjs.com/>
- [19] Martin, J. (1991). *Rapid application development*. Macmillan Publishing Co., Inc.
- [20] Chrysalis Solutions, Inc. | Delivery. (n.d.). Retrieved March 16, 2015, from <http://www.chrysalis-solutions.com/delivery.html>
- [21] Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

- [22] Beizer, B. (1995). *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc..
- [23] Nielsen, J., & Molich, R. (1990, March). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 249-256). ACM.
- [24] Leaflet Plugins. (n.d.). Retrieved March 18, 2015, from <http://leafletjs.com/plugins.html>
- [25] Google Maps JavaScript API (n.d.). Retrieved December 18th, 2015 from <https://developers.google.com/maps/web/>
- [26] Baas, B. L. P. (2012). NoSQL spatial–Neo4j versus PostGIS.
- [27] PGLoader (n.d.). Retrieved March 18, 2015, from <http://pgloader.io>
- [28] Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), 61-72.
- [29] Angular-ui/ui-router. (n.d.). Retrieved March 18, 2015, from <https://github.com/angular-ui/ui-router/wiki/Nested-States-&-Nested-Views>
- [30] AngularJS \$sce. (n.d.). Retrieved March 18, 2015, from [https://docs.angularjs.org/api/ng/service/\\$sce](https://docs.angularjs.org/api/ng/service/$sce)
- [31] PhoneGap Documentation. (n.d.). Retrieved March 18, 2015, from http://docs.phonegap.com/en/edge/cordova_file_file.md.html#FileTransfer
- [32] exif-js (n.d.). Retrieved March 18, 2015 from <https://github.com/jseidelin/exif-js/>
- [33] Introduction – Material Design – Google Design Guidelines (n.d.). Retrieved March 19, 2015 from <http://www.google.com/design/spec/material-design/introduction.html#introduction-principles>
- [34] Goadrich, M. H., & Rogers, M. P. (2011, March). Smart smartphone development: iOS versus Android. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 607-612). ACM.
- [35] Snapchat. (n.d.). Retrieved March 20, 2015, from <https://play.google.com/store/apps/details?id=com.snapchat.android&hl=en>
- [36] Whisper. (n.d.). Retrieved March 20, 2015, from <https://play.google.com/store/apps/details?id=sh.whisper&hl=en>
- [37] PhotoMap. (n.d.). Retrieved March 20, 2015, from <https://play.google.com/store/apps/details?id=eu.bischofs.photomap&hl=en>
- [38] Smart Phones Application development using HTML5 and related technologies: A tradeoff between cost and quality
- [39] SurveyMonkey (n.d.). Retrieved December 9th, 2014, from <https://www.surveymonkey.com/>
- [40] Ruby on Rails. (n.d.). Retrieved March 21, 2015, from <http://rubyonrails.org/>

- [41] Ubuntu : Intel® Q6600® one coreComputer Language Benchmarks Game (Ruby vs Python) (n.d). Retrieved March 21 from <http://benchmarksgame.alioth.debian.org/u32/compare.php?lang=yarv&lang2=php>
- [42] PhpMyAdmin. (n.d.). Retrieved March 21, 2015, from <http://www.phpmyadmin.net/>

9. Appendix

Appendix A: Survey Monkey Results

Question 1: How old are you?

Answer Choices	Responses
12 to 17	0.00% 0
18 to 24	100.00% 18
25 to 34	0.00% 0
35 to 44	0.00% 0
45 or older	0.00% 0
Total	18

Question 2: How often do you use social media applications?

Answer Choices	Responses
Every day	100.00% 18
A few times a week	0.00% 0
Once a week	0.00% 0
Not often	0.00% 0
Never	0.00% 0
Total	18

Question 3: How likely would you be to use an app that allows users to share and browse photos on a map?

Answer Choices	Responses
Very Likely	66.67% 12
Probably	22.22% 4
Probably Not	11.11% 2
I would not use it	0.00% 0
Total	18

Question 4: What type of phone do you have?

Answer Choices	Responses
Android	55.56% 10
iPhone	44.44% 8
Other	0.00% 0
Total	18

