# Is Hybrid Mobile App Development a Feasible Alternative to Native App Development? An Exploration through building a Music Social Network App

Final Report for CS39440 Major Project

*Author:* Sean Anderson (sea6@aber.ac.uk)
*Supervisor:* Dr.Chuan Lu (cul@aber.ac.uk)

10th February 2017
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

# Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Sean Anderson

Date .....

# Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Sean Anderson

Date .....

# Acknowledgements

I am grateful to my parents for there constant help and support (along with the rest of my family), Josie and all the Jordans for always making me laugh.

Thanks to Lauren for keeping me sane at points during my final year.

# Abstract

The issue of the best way to develop mobile applications is a complex one. There are three different techniques for developing apps: native development, web based app development and hybrid apps. Hybrid apps combine techniques from both native and web based app development.

Through developing a social media app I have examined hybrid mobile app development whilst pondering whether they actually are plausible to developing apps in a native manner.

I found that hybrid apps are feasible alternatives to developing apps in a native manner for things rather trivial such as data input and output and storage, however from the research I carried out it appears that developing advanced graphics and games in hybrid apps is not currently a viable thing to do.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

# Background & Objectives

The purpose of this project is to establish whether hybrid mobile applications are a feasible alternative to native mobile applications. This will involve exploring, examining and developing a mobile application in a hybrid manner. For this project, a music social network app was developed. The app has various features including the ability to follow artists and see what they post as well as looking up events which take place in venues. A hybrid mobile application framework will be selected which will form the front end of this project. The back end of this project was developed using PHP and a MySQL database.

## 1.1 Background

There are three different types of mobile applications: native apps, web apps and hybrid apps [22] [15] [33]. A native app is an app which is developed in a platform specific language and is usually downloaded from a device's app store. As native apps are developed in a platform specific language e.g. Android apps are developed in Java [14] and iOS apps are developed in Swift (formerly Objective C) [8] this means that code must be written in multiple languages to develop the app for different platforms. Native apps allow for platform specific Application Programming Interface (APIs) to be used and therefore the developer has access to resources such as the device's camera or contact book [25] [12].

Web apps are essentially just web pages which are mobile responsive and therefore display well on mobile devices. They are accessed via a device's browser when the user either inputs the Uniform Resource Locator (URL) or clicks a link which takes them to the web app. Web apps do not (easily) have access to the platforms' APIs therefore it is very challenging for a developer to use native resources such as a device's camera [25] [22].

Hybrid apps are apps which are written in web technologies (usually HTML5, CSS and JS.) They are usually downloaded through an app store and a piece of middleware enables hybrid apps to have access to native APIs, allowing hybrid apps to access resources such as a device's camera. Unlike native apps, hybrid apps only require one code base for multiple platforms [25] [19].

## 1.2 Analysis

It is important to consider the advantages and disadvantages of the different types of mobile app development as this will allow for a reasonable judgement to be made as to whether hybrid apps are feasible alternatives to native apps.

As hybrid mobile apps use one code base across multiple different platforms the development cost is cheaper as companies do not need to hire multiple different programmers to work across different platforms. If a business decided they wanted to release an app on both Android and iOS, if they took a native approach they would have to write the code for the app both in Java and Swift (possibly Objective C instead of Swift). However, if they were to develop the app in a hybrid manner then they would only have to write the app in the web technology framework which has been chosen.

Through using middleware technologies hybrid mobile apps can access native APIs. Different hybrid frameworks do this in different ways. Because of this, the developer needs to be very careful in choosing the correct framework for developing hybrid apps. They would need to ensure that the framework they want to use has access to all the native features they require. Most frameworks use plugins to access native features.

It is commonly thought that the performance of hybrid apps' is poor and that they are slow. Whilst this does appear to be a slight exaggeration, hybrid apps are generally slower then native apps and therefore the performance is generally poorer as a user may have to wait longer for an app to load.

## 1.3 Process

There is a common debate within software engineering about whether to use a plan driven methodology or an agile methodology. Plan driven methodologies rely on the requirements for the project not changing whereas agile methodologies try and embrace changing requirements [20].

As this project, is an investigation the requirements of the app may change it may be worth spending more time than initially planned building specific features as building those features will help make a judgement as to whether hybrid apps are feasible alternatives to native apps. Therefore an agile approach was used for this project.

There are multiple agile approaches to software development. Some of these approaches include: eXtreme Programming, Kanban and Scrum. Scrum was the agile methodology which was used for this project this is because Scrum doesn't dictate how the software will be developed as opposed to other methodologies such as eXtreme programming which informs the programmer that they should be developing software in a specific manner, such as using test driven development and pair programming. Whilst Scrum was the methodology chosen, the project didn't use all of the rules from the Scrum methodology. This is because Scrum involves multiple roles in a team and this was a single person project [11].

Scrum (as with most other agile methodologies) splits the requirements for the app into multiple stories. These stories then form the basis of each sprint (a work iteration) [24]. One of the most important things with using Scrum is to ensure that each sprint is planned and reviewed as this will help reduce errors when developing the app. The creation of templates seemed like a

good idea as this meant that less time would have to be spent formatting sprint planning and sprint review documents. Figures 1.1 and 1.2 show the templates.

Sprint Plan Week: 0 (6th to 12th Feb)

Sean Anderson

March 26, 2017

## 1 Sprint N Title

- Feature goes here.

## 2 Design (if necessary.)

## 3 Sprint Timing

Figure 1.1: The Sprint planning template

Sprint Review Week: N

Sean Anderson

## 1 Title of Sprint

- Sprint requirements

## 2 What was achieved

- This was achieved.

## 3 What was missed

- I missed this functionality this week.

## 4 Overall Review and Future Planning

- As a result of missing X functionality in the next sprint I must do.

Figure 1.2: The Sprint review template

### 1.3.1 Initial Requirements

Before splitting the different tasks into stories, it is important to consider the overall requirements of the system. Anyone must be able to register for an account; there will be three different types of account (a general music lover, an artist and a venue owner.) The general music lover should be able to follow artists and view events which are relevant to them. An artist should have the same functionality of a music lover however they will appear in a different section of the app. This is so it makes it easier for people to know who they are following. Both artists and music lovers should be able to post and people who follow them will see this post.

Venue owners need to be able to create events and add artists to these events. Music lovers should then be able to see all the appropriate information about these events. In terms of suggesting to the user who they should follow and what events they should attend, there will be some machine learning code placed on the back end of the system. As well as this the app, will need to be styled.

### 1.3.2 Stories

As Scrum is an agile methodology the project was split into multiple different stories. These include:

- A register system so that all user types can create an account.
- A profile system which will allow for music lovers and artists to write a bio and upload a picture.
- A follow system so that users can follow other users.
- A post system so that when a user posts, all people who follow them can see that post.
- A create events system which can only be used by venue owners.
- Styling of the app.
- Recommendation system (which would be machine learning code) on the back end of the system which will mean users will get appropriate suggestions.

### 1.3.3 Framework

There are multiple different frameworks for developing apps using hybrid technologies. These frameworks all use web technologies however it differs from framework to framework as to what specific web technology is used. For example, apps made using the JQuery Mobile Framework are wrote in JQuery and apps made using the Ionic Framework are written in AngularJS.

As well as considering the language of the framework it was also important to consider other factors before deciding which framework would be most appropriate. These factors include things such as page change speed, access to native APIs and external documentation and community available for support.

Table 1.1 summarises information which was gathered from multiple resources and summarises different hybrid app frameworks [23] [6] [29] [31] [30] [9].

| Framework Name | Summary | Advantages | Disadvantages |
|---|---|---|---|
| JQueryMobile | Framework uses HTML, CSS and JavaScript | Simple to learn. | Limited plugins. Page transitions are slow. Awkward to style No longer a large community so lack of support. |
| Phone Gap | Framework uses HTML, CSS and JavaScript Built on top of JQuery Mobile | Easy to learn - most developers have experience with necessary languages. Many Plugins available for access to platform's native APIs | Can be awkward to style. Page transitions can be slow. |
| Ionic | Framework uses HTML, CSS (Sass if developer prefers) and AngularJS. | Simple to style. Large community support. Lots of plugins. Considered fast. | Steep learning curve requires developer to understand AngularJs. |
| Framework 7 | Framework uses HTML, CSS and JavaScript. | Easy to learn. Nice styling which looks native. Can use in combination with other frameworks. | Not a very large community so a lack of support. Not many plugins easily accessible by default. |

Table 1.1: Table Comparison of Hybrid Frameworks

Despite Ionic having a steep learning curve due to it using AngularJS as opposed to JQuery it seemed like the most appropriate framework to use. This is because not only does Ionic have a fairly substantial amount of middleware plugins which will allow for platform specific APIs to be accessed but it is also considered one of the fastest frameworks, so this will aid the project aim of discovering whether hybrid apps are feasible alternatives to native apps.

# Chapter 2

# Design and Experimental Methods

## 2.1 Design

The ionic framework for developing mobile hybrid applications encourages the use of the Model View Controller (MVC) compound design pattern. MVC is something which is generally considered good practice within the software engineering industry and therefore the app was designed keeping this in mind.
The model of the app would be the database which is stored on the back end of the system. The view is the html files and the controllers are both the Angular JS controllers and the RESTful PHP API which was created.

### 2.1.1 Overall Architecture

The overall architecture of the system is complex and involves a lot of different files, this is due to the API having over 35 PHP files where each file performs a specific task. Figure 2.1 shows the overall architecture of the system it doesn't go into each of the specific files however it does provide a good visual representation of the system.

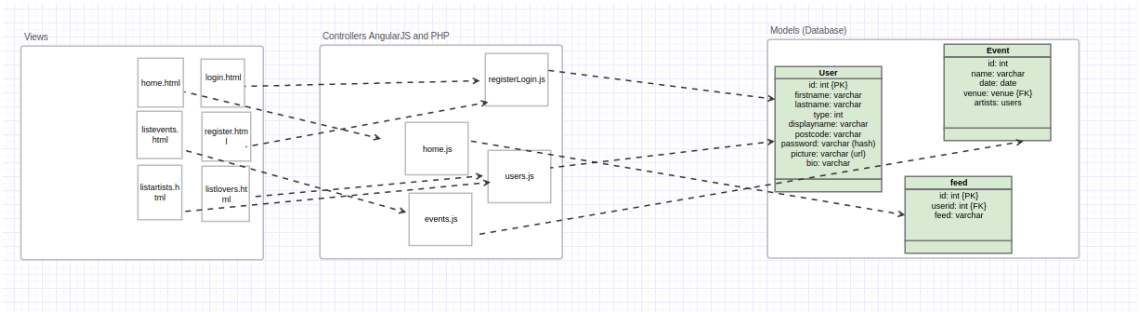


Figure 2.1: Overall architecture of the system.

Figure 2.1 shows the system split into the front end view, controllers (which includes both the front end controllers (AngularJS controllers) and the API, PHP file controllers) and the model which is the database. This figure does not show every little detail of the system.

#### 2.1.1.1 Use Case Diagram

Figure 2.2 is a use case diagram which shows the different features of the app different users can access.

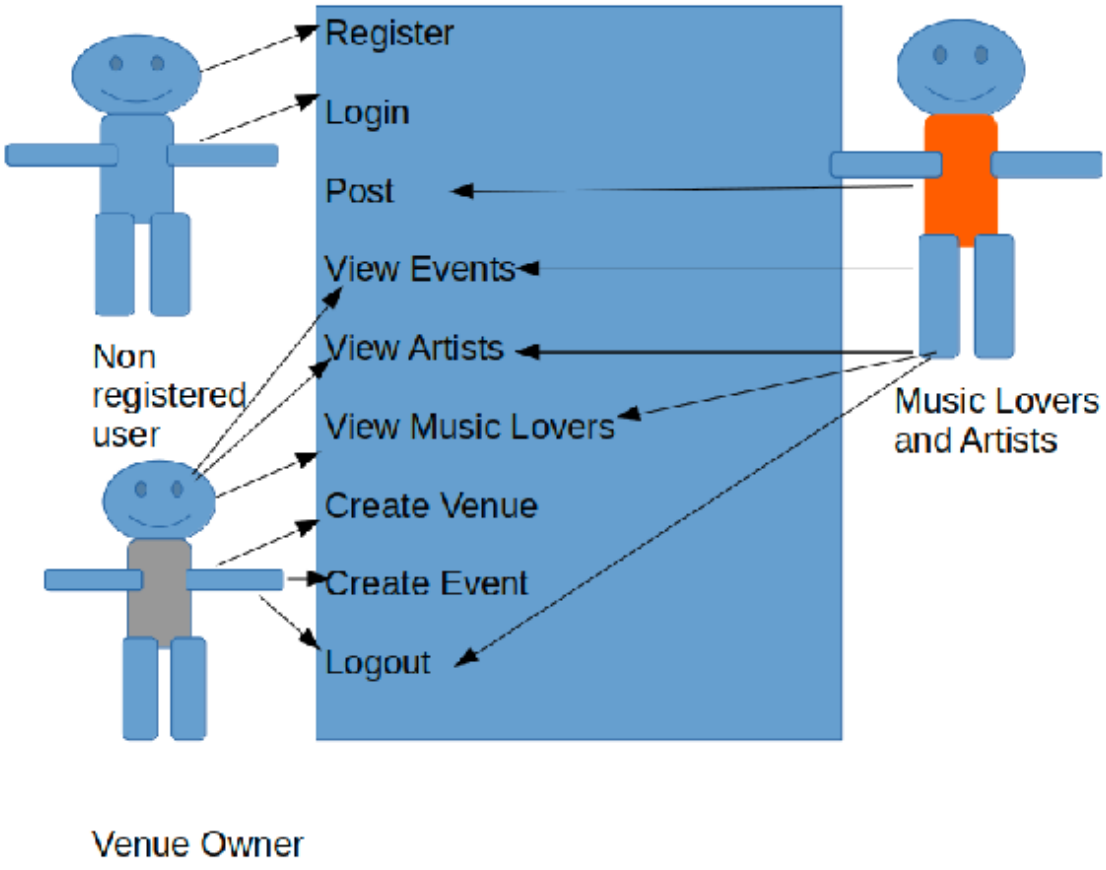


Figure 2.2: Use case diagram

### 2.1.2 Navigation System

The navigation system is done mainly through view files; a controller is called by the menu. This controller then checks what type of user the logged in user is, if they are a venue owner then it adds more links to the menu. Figure 2.3 (created using `https://www.lucidchart.com/`) shows this.

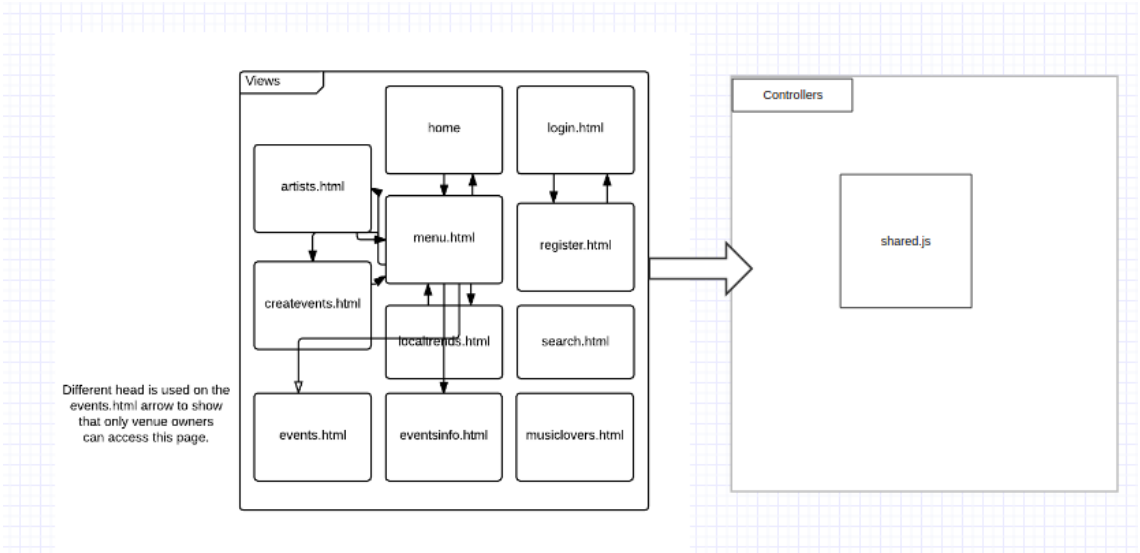


Figure 2.3: UML Diagram which illustrates the view

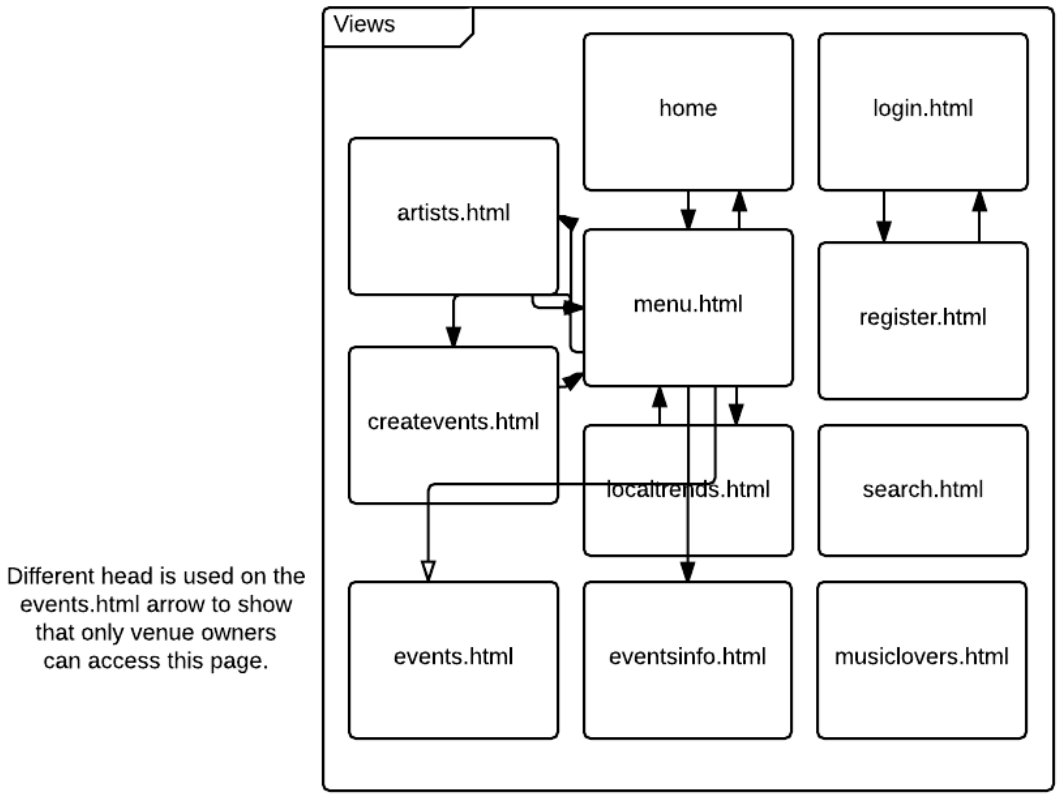


Figure 2.4: UML Diagram which illustrates the view interacting with controller.

Figure 2.4 shows just the view (created using `https://www.lucidchart.com/`) this illustrates that once a user has logged in they can access multiple different pages through the menu bar (menu.html). The only page which is restricted is the events.html page as only venue owners can access this page to create events. The register.html page can only be accessed through the login page as it is accessed when a user needs to create a new account to login.

### 2.1.3 Design of registration system

The design of the registration system considered that there are three different user types. The music lovers and artists have the same details whereas venue owners can also create events and venues. Figure 2.5 shows both the UML model for the registration system and the interaction between the model view and controllers. Figure 2.6 contains a flow chart which shows how the login and registration system work when the user launches the app.

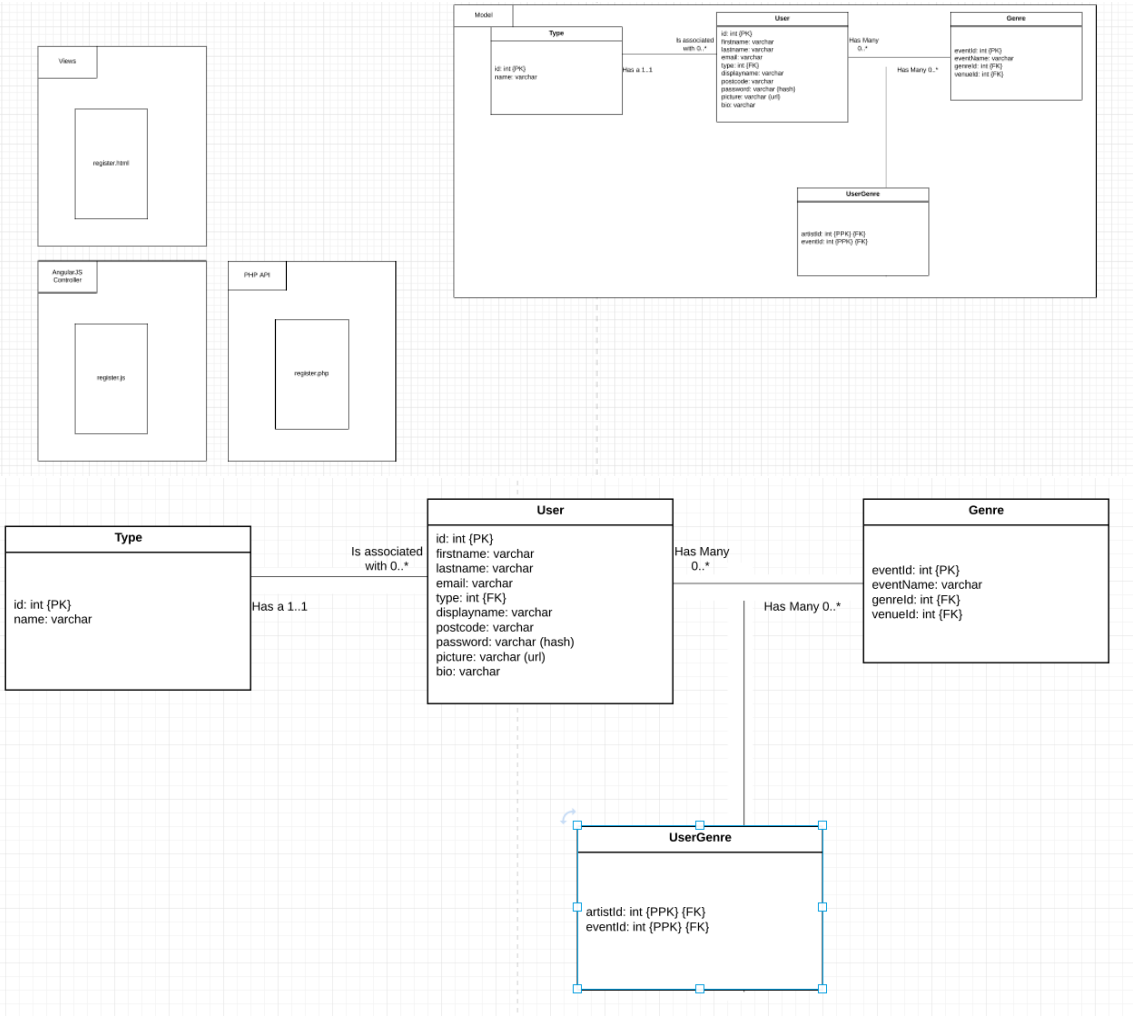


Figure 2.5: UML Diagram for the model in relation to a user registering.

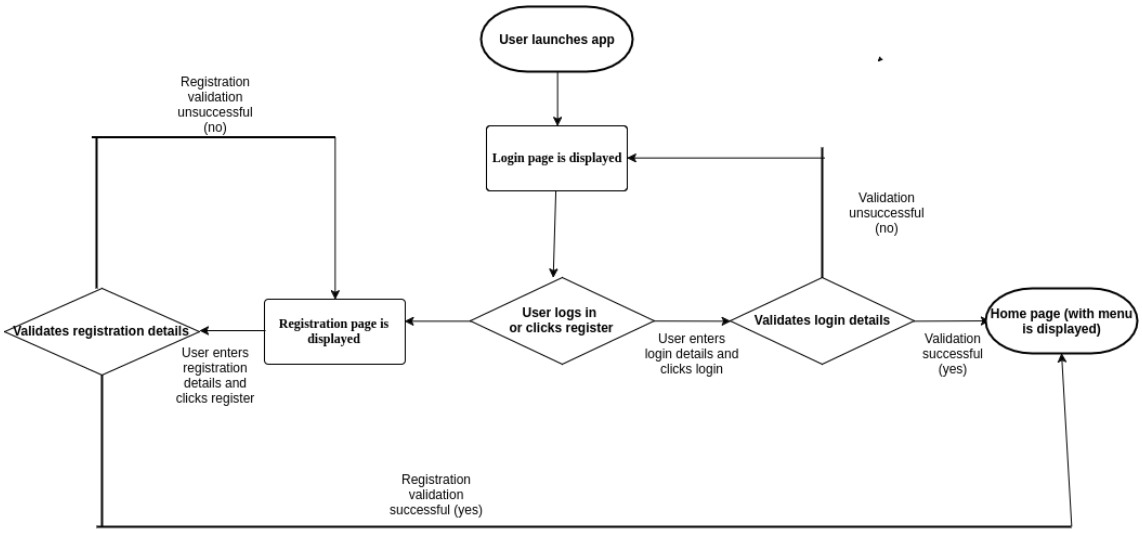


Figure 2.6: Flow diagram for registration system

### 2.1.4 Design of events

Figure 2.7 is a UML diagram (created using `https://www.lucidchart.com/`) which shows the appropriate relationships between the different tables in the database in relation to an event. For the interaction with controllers etc. it works in the same way that the registration system does.

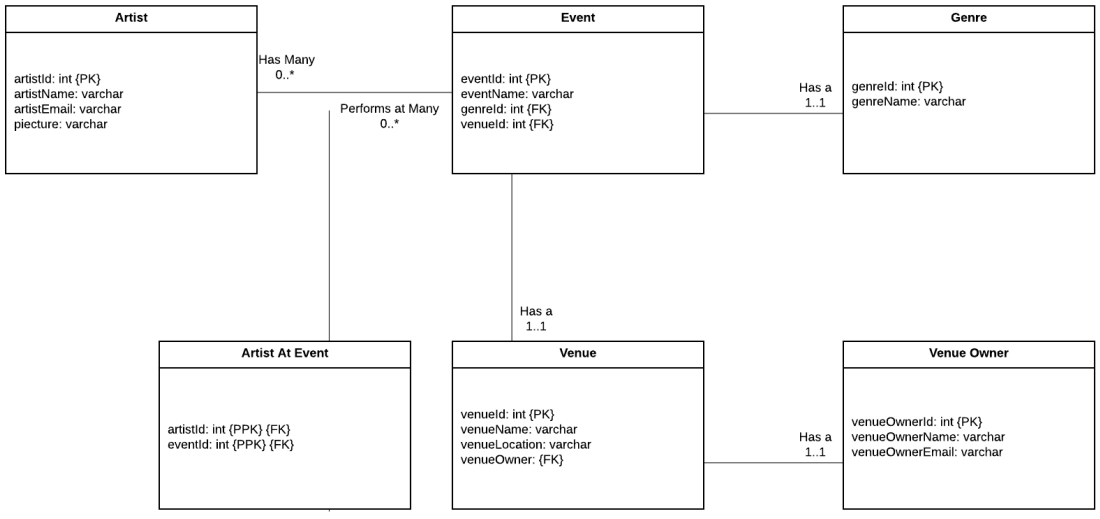


Figure 2.7: UML Diagram for the model in relation to events.

### 2.1.5 User Interface

In considering the music social network app, it was decided that the name of the app would be Dynamic. Research into different social media sites was carried out and it was decided that blue would be the main colour used for the app, this is because blue is considered a relaxing colour and in terms of accessibility it is one of the most accessible colours [18].

#### 2.1.5.1 Shneiderman Principles

Schniderman came up with eight golden rules for interface design, below the rules are listed followed by how they will be applied to the UI of the app. [27]

1. 'Strive for consistency' - There will be a consistent colour scheme throughout the user interface of the app, along with this navigation (where appropriate) will be the same throughout the app.

2. 'Enable frequent users to use shortcuts' - There are multiple ways in which this can be applied to a mobile phone app. One way it was applied to the app was by allowing the menu (for navigation) to be opened simply by the user swiping the screen to the left. It is also possible to close the menu by swiping to the right.

3. 'Offer informative feedback' - Every time the user clicks a button on the app, the app provides them with feedback in a variety of ways such as by navigating to a different page or by a popup telling them the action they performed was successful.

4. 'Design dialog to yield closure' - Popups were designed in such a way that the user is notified of exactly what has happened and thus this will give them closure.

5. 'Offer simple error handling.' - When things go wrong, for example if the user doesn't have GPS enabled they are informed of what the problem is and what a potential fix for the problem may be.

6. 'Permit easy reversal of actions.' - The interface was designed so that the user can easily go back and change the options they selected.

7. 'Support internal locus of control' - The interface has been designed so that the user is fully in control in terms of the data they are submitting and the way they are navigating between different pages.

8. 'Reduce short-term memory load.' - The only thing which the user must remember is there username and password to access the app.

Figure 2.8 (created using `https://www.fluidui.com`) shows some UI mockups which were created. The designs do not show any popup boxes this is because the ionic framework has a default pop up box which uses the device's native popup box.

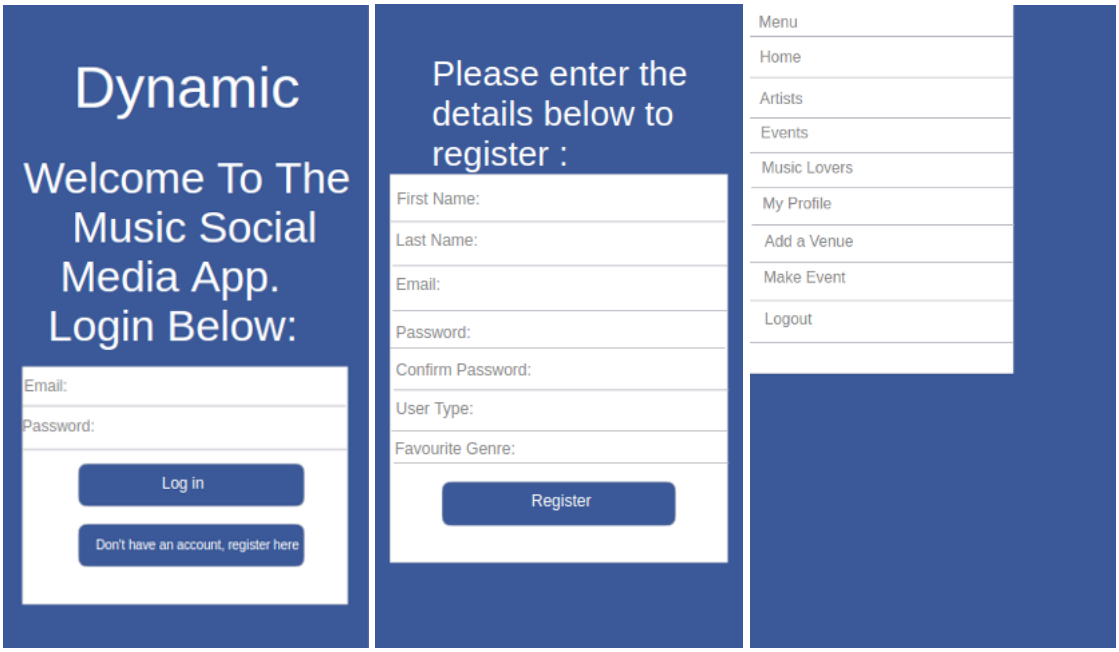


Figure 2.8: User interface mock up

## 2.2 Experimental Methods

As the purpose of the project was to determine whether hybrid apps are feasible alternatives to native apps tests had to be derived. User testing is key to this as it allows for general users (people who use apps on a regular basis) to give feedback about whether they think the hybrid app is as good as a native app, if not why not etc. Another key aspect to answering the question is comparing the resource usage on the device of the hybrid app to a standard app, this will include comparing things such as amount of RAM being used.
Finally, during each story in the development process, it was considered whether using hybrid app technology was as challenging, more challenging or less challenging than developing using native app technologies.

### 2.2.1 User Testing

In establishing whether hybrid apps are feasible alternatives to native apps it was important to get general users thoughts about the app which had been produced. Within the development community there is a bit of a stigma associated towards hybrid mobile applications therefore to remove bias the majority of people who fill in the questionnaires associated with the app will be lay people. Whilst clearly it would be a good idea to get as many people as possible to fill in the questionnaire, realistically only ( whatever number) were chosen. Before asking each individual question, the project was explained to them.
The following questions were asked to everyone who participated (a blank copy of the questions asked is in Appendix 4.)

1. On a scale of 1 to 10, with 1 being slowest and 10 being very fast in your opinion how quick is the app at processing the registration form?

2. On a scale of 1 to 10 with 1 being significantly slower than a native app 5 being about the same time as a native app and 10 being significantly faster than a native app. In your opinion, how quick is the app at processing the registration form?

3. On a scale of 1 to 10 with 1 being slowest and 10 being fast in your opinion how long does the process of uploading a profile picture take?

4. On a scale of 1 to 10 with 1 being significantly slower than a native app, 5 being about the same and 10 being very fast, faster than a native app. In your opinion how long does the process of uploading a new profile picture take?

5. On a scale of 1 to 10 with 1 being slowest and 10 being fast in your opinion how long does the process of viewing events and sorting them by using nearest events (GPS) take?

6. On a scale of 1 to 10 with 1 being significantly slower than a native app, 5 being about the same as a native app and 10 being very fast, faster than a native app in your opinion how long does the viewing events and sorting them by using nearest events (GPS) take?

These questions were decided upon as they ensured that all of the stories which were relevant in terms of testing different elements of hybrid technologies can be tested.

# Chapter 3

# Implementation

## 3.1 Overall approach to implementation

This chapter discusses the implementation of the music social network app 'dynamic'. It is split into multiple stories which have previously been listed. Not all of the stories are discussed here as some stories were very similar to others, and this chapter would have been very repetitive if every story had been discussed.

## 3.2 Toolset

It was very important to use an appropriate toolset for this project. Using an appropriate toolset would not only save time but would also aid in the creation of code.
Git is one tool which was used throughout the creation of the project. Git is a very useful tool as it not only allows for just backups but also for version control [3]. The main difference being version control means that all different revisions of the project could be viewed; this is extremely useful as it means that if app code was to break then the code could be restored to a working version. There are many different applications of git, consequently there are multiple providers of git services including gitlab and github [4] [5]. For this project github was used. As shown in Figure 3.1 a private git repository was setup which is what what used for version control and back ups.

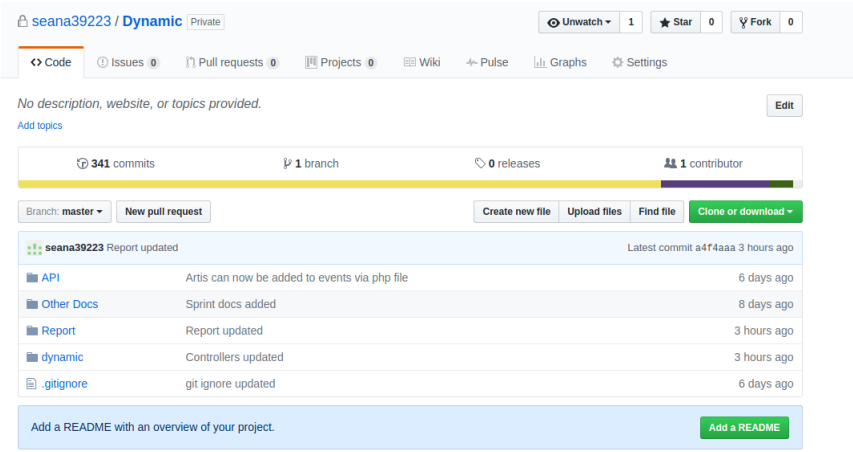


Figure 3.1: Git repository

Sublime is the text editor which was chosen for the project. Sublime was chosen as it has a large range of plugins, which is able to check the syntax of code along with tabulation etc.

## 3.3 The Registration system

### 3.3.1 Back End

Work began on the registration system by creating the appropriate tables on the back end. These tables were user_types, users, genres and user_genres. The user types table contained the three different types of account which a user could have: Music Lover, Artist and Venue Owner. The user type was done as a separate table to the users table itself for extensibility purposes as if additional user types needed to be added then they could simply be added to this table. The users table just contained all of the users' details including name, display name, email, password (discussed in more depth during the security section) and a url link to their picture. The genres table was simply a table which contained different genres of music and an appropriate id for each genre. As genres to users is a many to many relationship a users_genres table was created which stores a user_id and a genre_id.
After the tables had been created the php files (hosted at `seananderson.co.uk/api`) were created. These php files essentially formed the RESTful API which would allow for the front end to connect to the database for this system. The first file which was created was register.php this file takes variables which the user passes in (through POST data) and adds them to the database via my sqli commands which are ran from within the php. Some validation was done in the php file to check that things such as emails given are valid email addresses, however I wanted to do most of the validation on the front end as this would allow for error messages to get out to the user in a quick manner.
As at this stage in time there was no front end, consequently it was not possible to test that sending post data from the app would work. So, a piece of software called postman (which is a Google Chrome extension) was used. This software allows for post data to be pushed to a website or API as illustrated in Figure 3.2. As well as relying on the PHP file telling me 'A new user was added successfully' which is the message what is printed when the sql commands all run correctly, the database was also checked to ensure that the new user had been added correctly as shown in Figure 3.3.

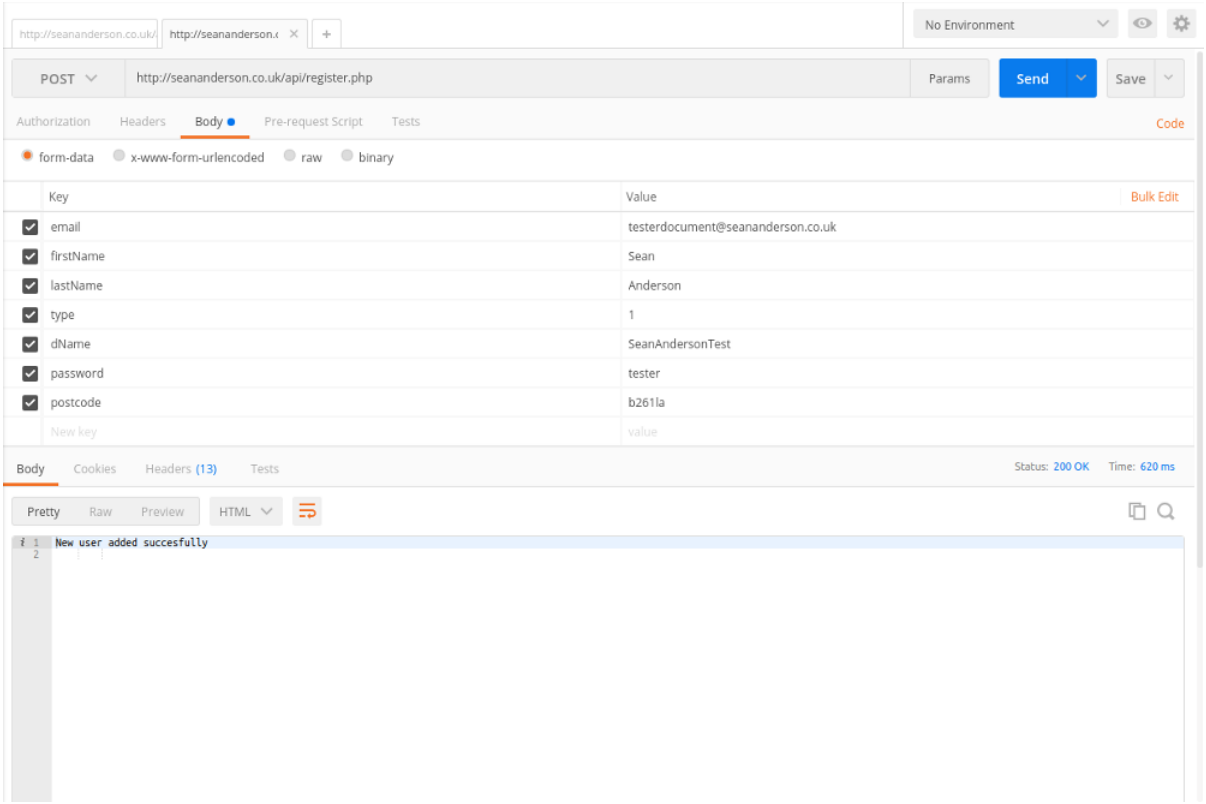


Figure 3.2: Postman testing register.php


Figure 3.3: User added to the database

#### 3.3.1.1 Security within the registration system

It is important that users are not allowed to enter dangerous characters into the database, as a result of this all strings which are inputted by the user are escaped using the mysqli_escape function which is a default php function which ensures that any characters which could potentially do damage to the database are escaped and therefore not ran in the mysqli statements.
As the registration and login system used a password it was important to ensure that if somebody was to get access to the database that they wouldn't get access to the password. There are multiple different methods of encryption which could have been used for this. PHP has its own built in function called password_hash which takes in two parameters the password itself as a string and the encryption type. For this project password_default which is a predefined php hashing method was used. Password default was used as it was a simplistic and relatively secure way of turning a password into a hash. The password_verify function in PHP is used to check that a plain text password matches the hashed password.

### 3.3.2 Front end

After the back end for the registration of the app was created, development then started on the front end. The default ionic menu bar template was used as this would allow for a menu bar to appear on multiple pages (like in the design) easily.
The views for both the login and registration system were created using html (as are all views in ionic) and the controllers for these views were created in AngularJS and were simply js files. Both the register and login view files contained a form which gathered appropriate information from the user and then passed that information onto the controllers when submitted. The controllers then passed that information onto the API's by using $http.post which is an AngularJS method used for calling post API's. Figure 3.4 shows how the AngularJS controller passes the appropriate data to the API. The full code for the registration system is shown in Appendix 3.

```
$http.post(api, data).then(function(res){
  apiReturns = JSON.stringify(res);
  if (apiReturns.includes('New user added succesfully')>=0) {
    localStorage.setItem('email', $scope.register.email);
    localStorage.setItem('dName', $scope.register.dName);
    if ($scope.registerGenre!=undefined) {
      $scope.registerGenre();
    }
    $state.go('app.profile');
    popUp('Welcome', 'Welcome to Dynamic please fill in
    your profile page and then follow some users');
  }
})
```

Figure 3.4: AngularJS code for posting to register account

In passing the information from the controller to the API I encountered a problem. AngularJS sends post data in a different way to most other languages. As a result of this the following lines of code had to be added to the PHP files as shown in Figure 3.5. (This code was taken from `http://corpus.hubwiz.com/2/angularjs/15485354.html`)

```
if ($_SERVER['REQUEST_METHOD'] == 'POST' && empty($_POST)) {
    $_POST = json_decode(file_get_contents('php://input'), true);
}
```

Figure 3.5: Code to make API work properly

Figure 3.6 shows what the front end for the login and registration screens look like on the front end on the system (from the perspective of a One Plus Two device.)

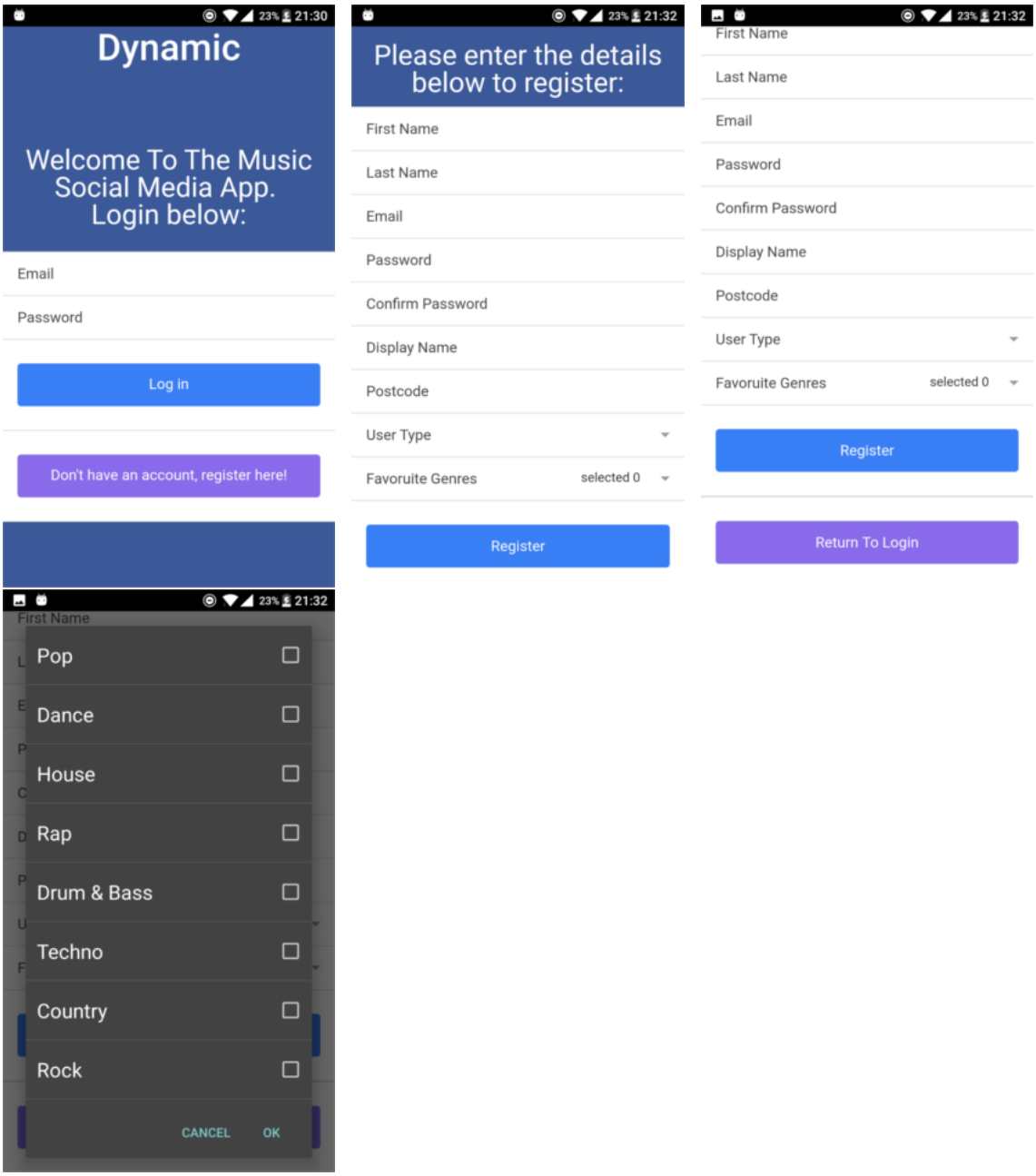


Figure 3.6: Screenshots of registration system.

### 3.3.3 Validation

Having successfully got the front end interacting with the back end it was important to ensure that all of the data being sent over was valid. I decided first of all to validate the data on the front end. This ensured that the user had filled in all of the appropriate boxes and a regular expression was used to ensure that the email address they entered was valid.
It was then necessary to create some more API files as each user had to have a unique email and display name, files called checkemail.php and displayname.php were created. These files run SQL

queries to ensure that the email address and display name which the user entered are unique. Figure 3.7 displays an example of what happens when a user tries to register an account with an email address which has already been used and when a password is not long enough.

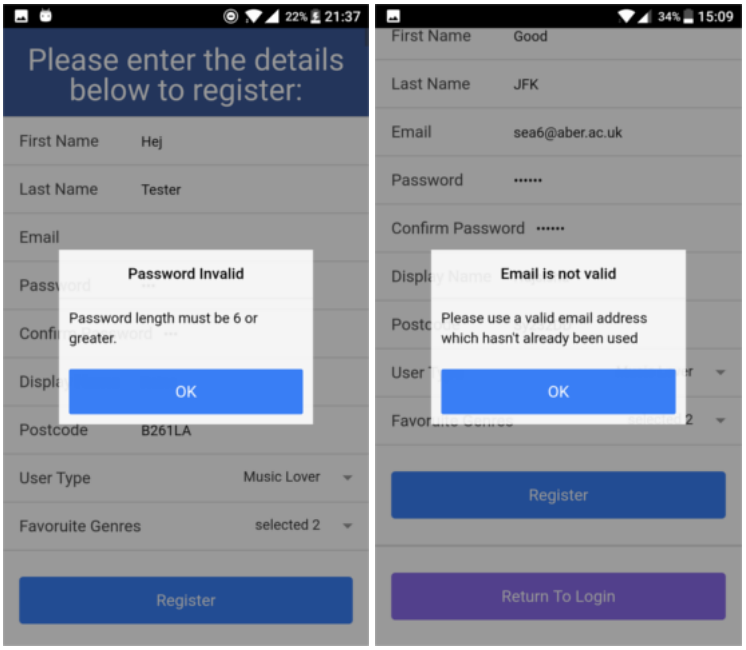


Figure 3.7: Validation Screenshots

## 3.4 Posting to a feed

There were multiple ways in which a user posting to a feed could be carried out. It was decided that a users posts would be stored in the database and would be retrieved from the database at appropriate times. As the relationship between a user and a post is a many to many relationship, a new table was created in the database called feed.
An API file called feed.php was created, this file takes in two post variables a users email and a users post (the string the user would like to add to their feed.) This file then uses the users email and runs a sql command to get the users id. This id is then placed into the feed table along with the users post variable.
The front end of the system worked in a way which was very similar to the registration system. The view of the front end was simply a text input field and a button (as shown in Figure 3.8) which when clicked would call an action in the controller.
This controller would then pass the appropriate variables to the API, using $http.post in the same way as the registration system. Finally a message would display to the user to let them know they had posted, this is shown in Figure 3.8.

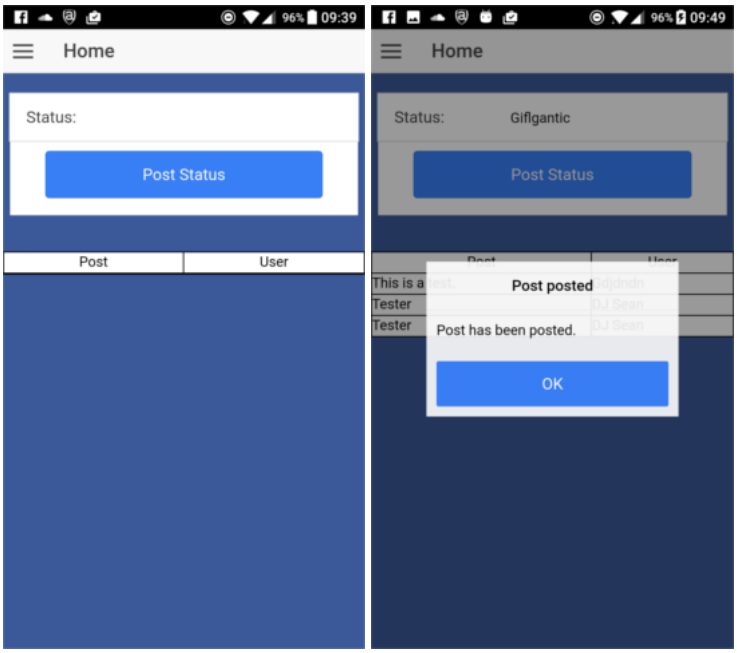


Figure 3.8: Validation Screenshots

## 3.5 Creation of Events

An important component of the mobile application was that users with the type venue owner could create events. Creating events is split into two different parts, actually adding the venue where events will take place to the app and then the actual creation of an event.

### 3.5.1 Adding a Venue

The purpose of adding a venue to the app is so that a venue owner can quickly create multiple events at the same venue and they don't have to enter the venue's details every time. The system for adding a venue was designed in a way that should be simple for the venue owner to add a new venue. They simply had to navigate to the 'Add a venue' page and enter the appropriate details as shown in Figure 3.9.
The system worked in the same way as the registration system in how it passes the data to the back end.

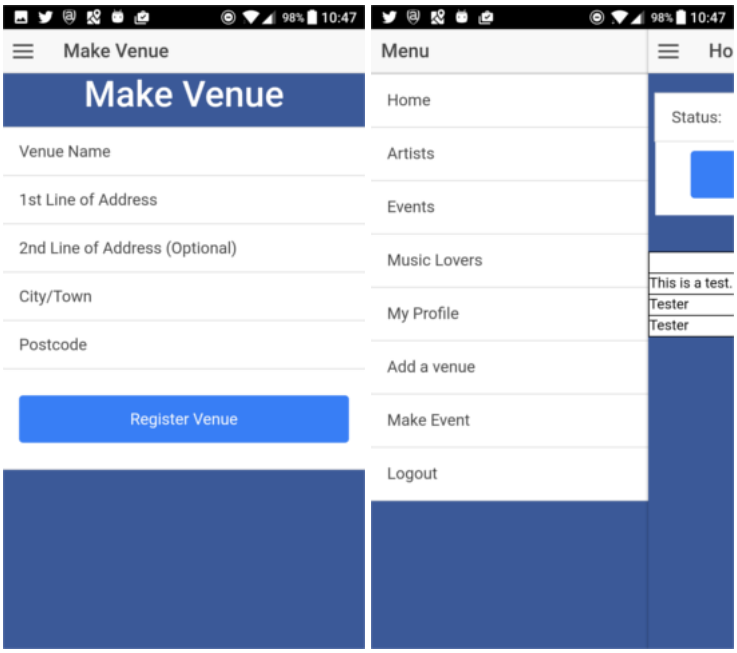


Figure 3.9: Front End for adding a venue

### 3.5.2 Adding an Event

Adding an event requires the user to already have a venue associated with them, consequently a message is displayed telling the user to add a venue if they don't already have one as shown in Figure 3.10.


Figure 3.10: Message which displays if the user has no venue.

Then once the user has created a venue they can create an event. The user simply fills in the appropriate details to create an event. The three drop down options on the page event venue, closest genre of event and artists performing are populated through API's. In terms of the event venue an API is called which gives all of the venues associated with the user logged in. The closest genre of the event is simply all of the genres which are in the database and the artists performing at event is made up of all artists who have an account on the app as shown in Figure 3.11. Once the details the user has entered are validated in that all of the information they have provided is valid the event is then added to the app and both music lovers and venue owners can add events to their favourites.

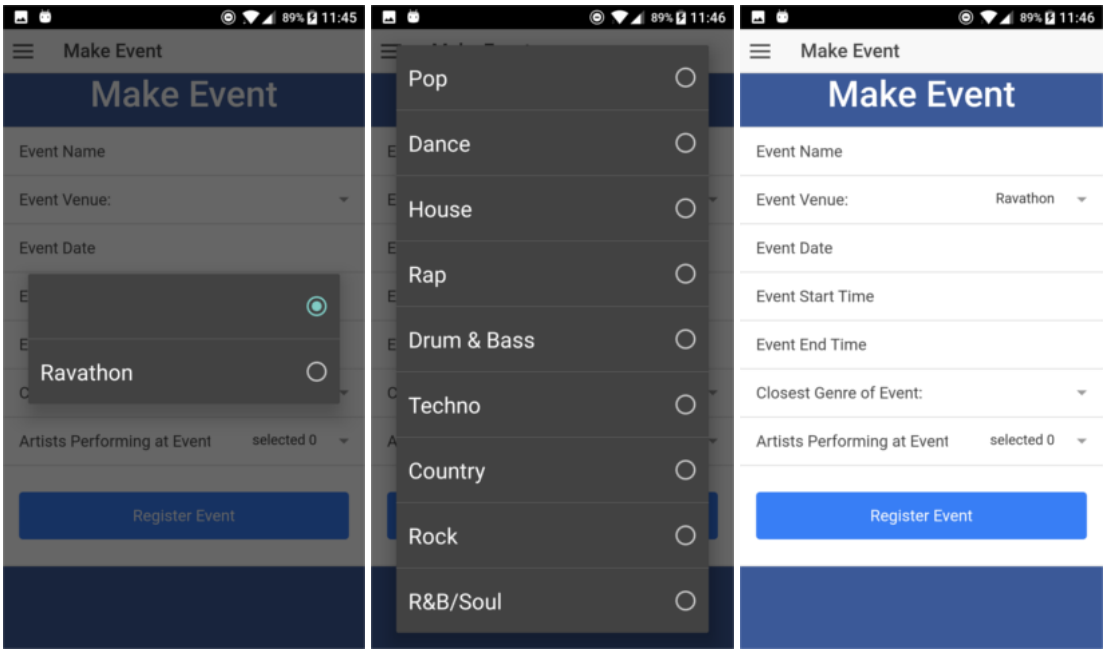


Figure 3.11: Creating an event

## 3.6 Users profile

Having created the registration and login system the next story to tackle was allowing the user to add a bio and to upload a picture. This picture would act as the users profile picture. Having previously completed the registration system allowing the user to add a bio was rather straightforward. It was just simply a matter of creating a front end which passes a variable to a php file (called update bio.php), the php files then takes the post variable and turns it into a php variable. Finally as shown in figure X the php file runs a sql command which updates the users table to have the correct information for the bio.

### 3.6.1 Camera and FileTransfer plugins

The task of allowing a user to upload a profile picture can be broken down into two seperate tasks. The task of actually allowing the user to take the photo or pick a photo from their devices library,

```
$sql = "UPDATE users SET bio = '$bio' WHERE displayname = '$dName' ";
```

Figure 3.12: Snippet of updatebio.php code which shows SQL command.

and the task of transferring that photo to the back end.
Installing the cordova camera plugin [28] was necessary to allow the user to take a photo or select a photo from their devices library. The actual installation of the plugin was simple, following the documentation for the plugin was straightforward and the app was programmed so that if the user clicked a button saying upload a picture or choose a picture from my gallery then the Cordova Camera plugin was called with the correct options configured.
Figure 3.12. highlights the two different buttons and what happens when they are clicked.

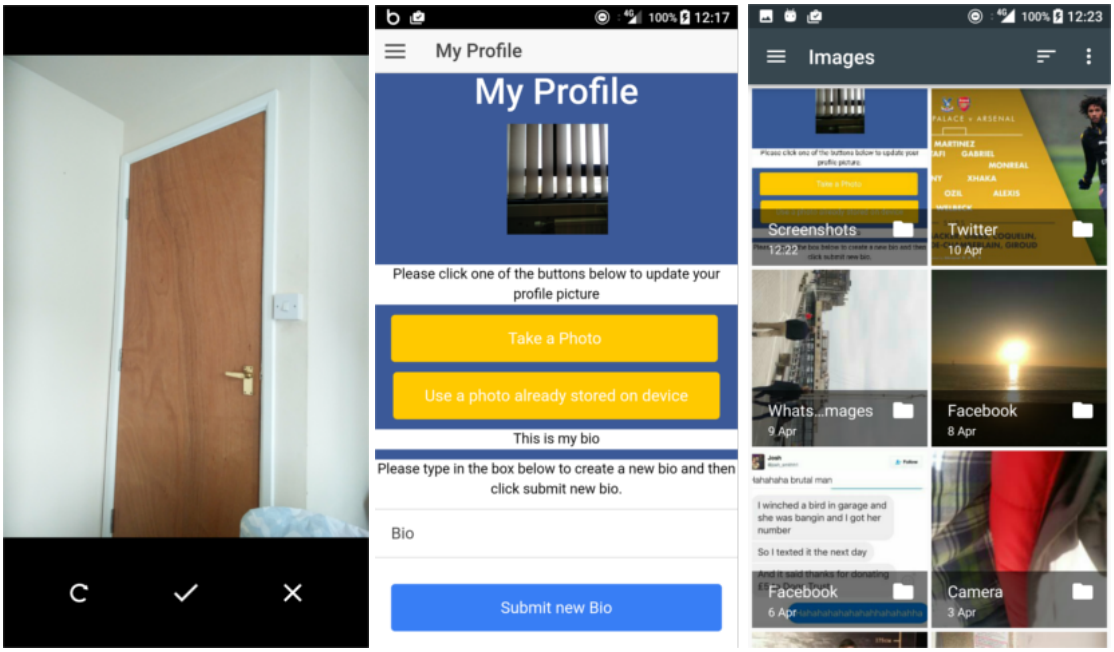


Figure 3.13: Screen shot of camera functionality.

After the user confirms (by pressing the tick) that they want to use the photo as their profile picture the photo is then uploaded to the server. The first step in getting the photo to upload was to create a API file, this file was entitled imageupload.php and is hosted at `seananderson.co.uk/api/imageupload.php`. This file contains some fairly trivial code which just gets the data of the image which has been uploaded and places it in an appropriate folder (`seananderson.co.uk/api/uploads`). After creating the php file the next step was to pass the photo from the device to the server. This was done by using the Cordova File Transfer plugin [**?**]. This plugin simply takes a file (which in this case was the image) and uploads the file to the specified URL.
After implementing the camera plugin one clear disadvantage of hybrid apps became apparent. The time taken for the device to take a picture and then process that picture was noticeably longer than it takes when using a native app such as Facebook. This is discussed in more depth during the testing section of this report.

### 3.6.2 Caching issue

At this stage of the development of the app it became apparent that there was an issue with caching. The issue was noticed as when the picture loaded on the users profile page after the user uploaded a new picture the picture would not be updated and would display the users old display picture even though the new image had successfully been uploaded.
A variety of different ways were used to try and tackle this issue. Ionic allows for the apps cache to be cleared using $ionicHistory.clearCache(); however this unfortunately didn't resolve this issue. As clearing the apps cache wasn't getting rid of the issue it was clear that the problem lied within the browser (which is how the app is ran.)
It turned out that the Chrome browser was caching the results of the API which displayed the picture, as the same API was being called on the page refresh, which was ran once the photo was uploaded. Despite not being the most elegant solution a random number was added to the end of the pictures url as shown in Figure 3.13.

```
$http.post(api, data, { cache: false }).then(function(res){
  //Below line is a hack, justified in report.
  var image = (res['data']['picture']) + '?random=' + Math.random();
  var photoDiv = angular.element(document.querySelector('#profile-photo'));
  photoDiv.html('<div id ="profile-photo"><img  height="60 px"
  width="60 px" src="' + image + '"</img></div>');
})
```

Figure 3.14: AngularJS Photo Code.

This meant that the browser would load the new image uploaded into the app as it would have a different url from the old image. The caching issue is clearly a negative to hybrid app development, if the app was developed in a native manner then it would allow for the developer to have full control over the cache and therefore this issue would not have arose had the app been developed in a native manner.

#### 3.6.2.1 Problem with Different Devices

The camera plugin worked fine on a One Plus Two mobile phone, however when the functionality was tested using a Samsung Galaxy A there was a problem, the orientation of the image was wrong. The Samsung Tablet would take the photo fine but when it actually came to uploading it, it would rotate the image.
One of the options when using the camera plugin is photo orientation as shown in Figure 3.14.

```
var options = {
  quality:80,
  targetWidth:500,
  targetHeight:750,
  sourceType : Camera.PictureSourceType.CAMERA,
  encodingType: Camera.EncodingType.PNG,
  correctOrientation: true
};
```

Figure 3.15: Camera Plugin Options.

Unfortunately enabling that to be true still didn't solve the problem. Despite there not being any official sources suggesting why the correct orientation doesn't always work the feeling within the Ionic Community is that Samsung devices actually ignore the correctOrientation option [2]. This is clearly another disadvantage to hybrid apps as native apps would give the developer more control [1].

## 3.7 Styling and Porting over to iOS

As ionic uses HTML for its views the styling is done in CSS, SASS can be used as an alternative to CSS if the developer wants. The majority of the app was styled whilst working on each story however it was important to spend some time to ensure that the styling of the app was consistent across multiple devices and platforms.
iOS mobile applications can only be released and launched through using xCode which is only avaliable on Macs. Clearly this requires some effort however it requires significantly less effort than if the apps were developed in a native manner. If the apps were developed in a native manner then the code would have to be completely rewritten for launch on iOS devices where as when using a hybrid technology it is simply a matter of installing any plugins on the Mac which were installed when developing the Android version and potentially some slight tweaks to the styling as Android and iOS browsers interpret CSS slightly differently.
This did actually bring up some issues with hybird app development. Whilst a key feature of hybrid app development is the ability to deploy an app to multiple different code bases; iOS and Android interpret CSS differently. Therefore time had to be spent tweaking the CSS of the iOS version. (Add this if possible Figure X shows a particular part of the app running on iOS and Android however it is clearly inconsistent.)

Figure 3.16: iOS not the same as Android image goes here

## 3.8 Recommendation System

When creating the recommendation system it was considered that information about the user to recommend events can be obtained explicitly and implicitly [17]. It was important to consider that if the app was released commercially then the ethics of obtaining data implicitly would need to be considered and the user would need to understand exactly how their data would be used.
The recommendation system was made for events and in terms of explicit data available, two pieces of information were relevant these were the users favourite genre's of music and the users postcode. This is because each event has a genre associated with it and a postcode so comparing this information allows for the establishment of how similar in terms of music the user likes is to the event and the distance between the user and the event. When using the explicit data to recommend events the system would be using content based filtering [21].
There was only one relevant type of implicit data available about users which would help in creating a recommendation system, this is the users current favourite events. If the user has already favourite a event then it is possible to compare the events they currently have favourite with other events and make recommendations based of how similar the events are. When using implicit data to recommend events the system would be using collaborative filtering algorithms [16].

```
$sql = "SELECT genre_id from user_genres WHERE user_id = '$userId'";
$genres = $connection->query($sql) or trigger_error($mysqli->error."[$sql]"
    foreach ($genres as $genre) {
        $genreIdSql = $genre['genre_id'];
        $sql = "SELECT * from events WHERE genre_id = '$genreIdSql'";
        $events = $connection->query($sql) or trigger_error($mysqli->error.
        while($event = mysqli_fetch_assoc($events)) {
            $eventsArray[] = $event;
        }
    }
}
```

Figure 3.17: Code which shows how steps 1 and 2 work for Content Based Filtering

Figure 3.18: Insert figure showing sorted events here.

### 3.8.1 Content Based Filtering

The recommendation system runs when the user decided to sort events by recommended events. The first thing which happens when the user does this is that the app makes a call to the back end which checks if the user has any events in their favourites. If they do have events in their favourites then the system uses collaborative filtering algorithms to recommend events if not it uses the following method to recommend events using content based filtering.

1. Looks up genres of music associated with the user.
2. Looks to see if any of those genres of music are being played at events and adds appropriate events to an array.
3. Sorts the array in terms of distance between the users home and event.
4. Displays the recommended events on the app itself.

Figure X shows how steps 1 and 2 work via php code (the full code can be seen in Appendix 4 under recommendedevents.php). Figure X shows how the events display on the front end of the app once they have been sorted.

### 3.8.2 Collaborative filtering

When the user does have some favourite events collaborative filtering is performed to give the user recommended events which they will hopefully want to attend. The first thing which is done is that the events which the user is attending are added to an array. //TODO: Get this working and rest of section done.

## 3.9 Debugging and other features of hybrid app development

One massive advantage of the ionic framework is that the developer can actually launch the app in a Google Chrome browser by running the 'ionic serve' command from a terminal. This is a very useful feature as it also allows for code to be edited and the page will simply refresh as shown in

Figure 3.15. This is a clear positive of hybrid app development as it means that the code can be 'live edited' so the developer can see the changes they are making and because the user doesn't necessarily have to use emulators which can be resource heavy.

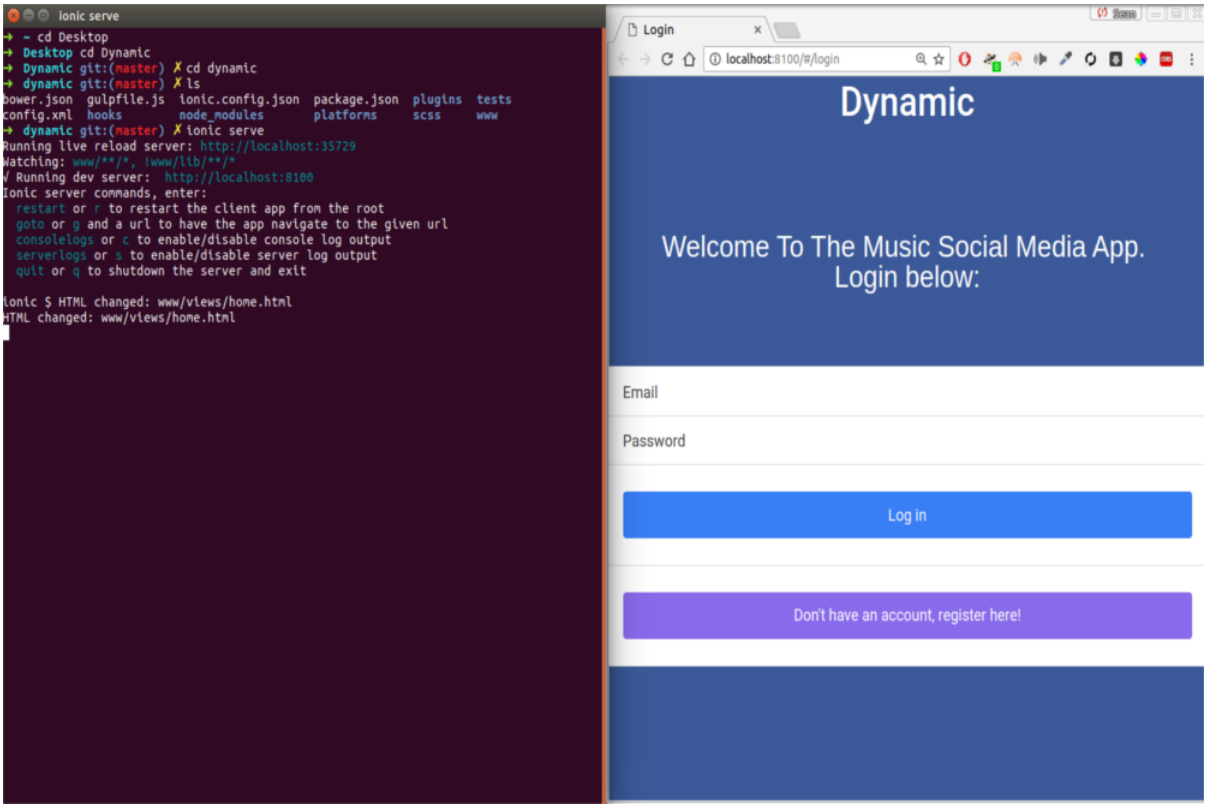


Figure 3.19: ionic serve command running

Debugging the app is also rather straightforward as Google Chrome's developer tools can be opened up whilst running ionic serve as shown in Figure 3.16 this allows for various commands to be typed to test a variety of things and also for console logs to be displayed which clearly aids the developer in checking variables.

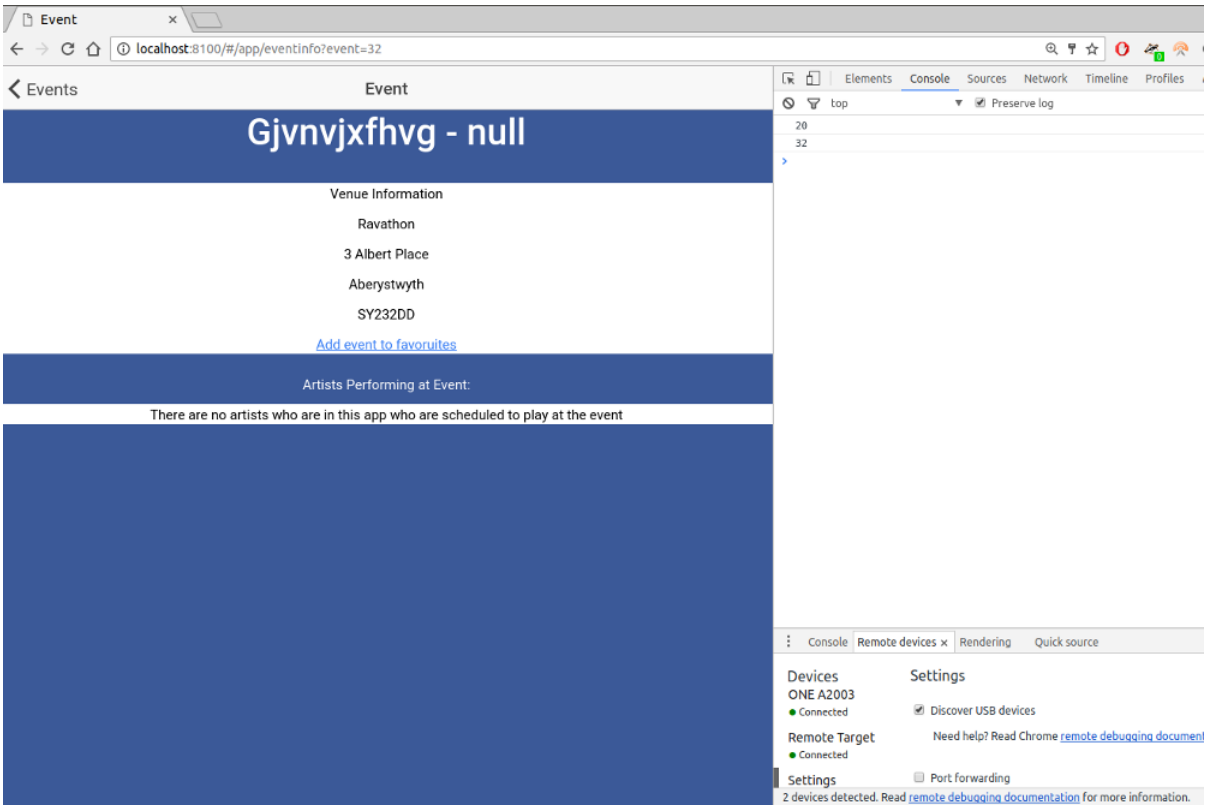


Figure 3.20: Chrome developer tools

## 3.10 PHP files

The back end of the app had over 30 PHP files, this had clearly taken a lot of time to develop. There is a lot of overlap in the files and a lot of them are very similar they just perform slightly different functions. When the system was initially thought off it wasn't expected that there would be so many PHP files and that is why all of the files were wrote manually, rather than using a framework.
This is where using an agile approach for the project had a negative impact, as if a plan driven approach had been used then it would have been realised much earlier on in the project the sheer amount of PHP files which would have to be created and so an appropriate framework would have been selected.

## 3.11 Technical challenges

There was multiple technical challenges which were faced when developing the app. The first of which was actually learning AngularJS, this was simply a case of trying different things until they worked and looking at multiple different sources online.
The caching issue was a rather interesting technical challenge, despite it not taking a long time to resolve it clearly highlighted an issue with hybrid mobile apps. The photo orientation with the multiple different devices was also an issue as it meant that more lines of php had to be added to

the API to ensure that the photo was uploaded in right orientation, therefore a developer would have to spend more time working on the photo upload using hybrid technologies than if they were to use the native app development.

## 3.12 Implementation vs design and plan

The advantage of using an agile methodology as opposed to a plan driven methodology was that if the initial plan was not the best option the project could easily be changed. Generally the initial plan and design of the system did end up being part of the implementation however in certain circumstances the plan was not stuck too. These include

- Slight changes to how venue and event creation work so that a venue has to be created before an event.
- When the project was thought of initially, the amount of PHP files without using a framework was vastly underestimated.

As a whole whilst the implementation didn't always stick to the design and plan it did stick to the initially planed stories and requirements.

## 3.13 Conclusion of implementation

Overall the implementation of the creation of the app went well. The actual development of the app meant that many advantages and disadvantages to hybrid app development could be established. The main advantages which were discovered were: quicker to port over than a native app would be, easy to use tools for debugging, language was relatively straightforward to pick up for a developer with some web experience and the app does seem to be relatively fast. Some of the disadvantages which were discovered include: not having full control over platform native features such as a cameras orientation, having very little control over memory usage of a device and a few caching issues.

# Chapter 4

# Testing and experimenting

## 4.1 Overall Approach to Testing and Experiments

In the sense of this project testing was carried out for multiple purposes. From a software development point of view testing was carried out to ensure that the app performed as expected, where as from the research perceptive of this project testing was carried out to determine whether hybrid apps are feasible alternatives to native apps.
The acceptance and unit testing was carried out to test the actual functionality of the app and that the software performs as expected. The user testing and resource testing was more the experimental side to testing as it helped answer the research element to this project.

## 4.2 Acceptance Testing

It was important that all of the stories and features of the app worked properly as this would help with the user testing. Whilst there was no business or end user who specifically gave the project the requirements this stimulated. A table was derived which broke down all of the stories into multiple different features. These features were then tested, and a comment was left if the test did not pass.

| Story | Feature Name | Pass/Fail | Comment |
|---|---|---|---|
| Registration and Login | Register User Details | Fail | Users postcode does not get added properly |
| Registration and Login | Login with previously created account | Pass | Works as expected |
| Registration and Login | Logout | Fail | System appears to logout okay however it loads previous users feed. |
| Profile Page | Upload picture from library | Pass | Works as expected |
| Profile Page | Take picture using device's camera and upload. | Pass | Works as expected |
| Profile Page | Update bio | Pass | Works as expected |
| Post System | User Posts | Pass | User posts get added to feed table as expected. |
| Follow System | User can follow other users | Pass | adds data to the correct table and shows as followed on the app |
| Follow system | User can unfollow other users | Pass | Can successfully unfollow other users |
| Post System and Follow System | Correct posts display on users home page | Pass | The correct posts display |
| Create Events System | Create a Venue | Pass | All necessary details are stored in the back end appropriately |
| Create Events System | Create Events Themselves | Pass | Events are properly created and have a correct link to the appropriate venue id. |
| Search | User can search for nearby events | Pass | Uses device's GPS correctly and compares with events location |
| Search | User can search for nearby music lovers and artists | Pass | Uses device's GPS correctly and compares with events location |

| Story | Feature Name | Pass/Fail | Comment |
|---|---|---|---|
| Search | User can search for events which are happening soon | Pass | Correctly sorts events by dates |

Table 4.1: User Acceptance Testing Table

It is clear from Table 4.1 that in terms of acceptance testing some of the code contained bugs for example the logout system was not working properly. Once all of the issues in Table 4.1 had been addressed another table; Table 4.2 was created, this goes through the tests which failed in Table 4.2 and evaluates whether they now pass.

| Story | Feature Name | Action Taken | Now Pass/Fail |
|---|---|---|---|
| Registration and Login | Register User Details | API amended so that the postcode variable is now assigned properly | Pass |
| Registration and Login | Logout | Now clears cache on logout | Pass |

Table 4.2: Corrections of failures from initial user acceptance testing

## 4.3 Unit tests

It is important to unit test each of the main functions within the controllers to make sure that the user of the app wont encounter any weird behaviour. The Jasmine and Karma frameworks were used for unit testing the app [10].
Figure 4.1 gives a small extract of the login unit test (the whole logintest.js file can be found in the appendix) which was wrote this unit test actually tests login controller. The unit tests work in such a way that the developer describes the function, followed by saying what the function should do and finally

```
describe('#doLogin', function() {
  it('Should Login using correct details', function() {
    expect(loginMock.doLogin).toHaveBeenCalledWith
    ('sea6@aber.ac.uk', 'b261la');
  });

  describe('when the login action is performed,', function() {
    it('if successful, should change state to home', function() {
      expect(stateMock.go).toHaveBeenCalledWith('app.home');
    });

    it('if unsuccessful, should show a popup', function() {
      expect(ionicPopupMock.alert).toHaveBeenCalled();
    });
  });
})
```

Figure 4.1: Sample of unit test for login

Table 4.3 illustrates the different unit tests which were created (the tests can be viewed within the technical hand in.) and what functions were tested using these unit tests. Not every function was tested as some functions were very similar to functions in other controllers.

| Controller Name | Function Tested | What should happen | Name of Testing File |
|---|---|---|---|
| Login | doLogin | Login with details | logintest.js |
| Login | register | Move to registration screen | logintest.js |
| Register | doRegistration | Register with details | registertest.js |
| Register | returnToLogin | Change state to login | registertest.js |
| Home | postStatus | Post a status and popup should appear | hometest.js |

Table 4.3: Table which shows different unit tests

## 4.4 User Interface testing

For a mobile app user interface testing is very important. As the user interface is essentially how the user natvigates around the front end and how they submit data to the back end. Whilst there is no simple automatic way to test the UI on a ionic application, a table as shown in Figure 4.4 was derived which shows all of the different buttons and navigation options to ensure that the UI worked as expected.

| Location | Button/Navigation | Pass/Fail | Comment |
|---|---|---|---|
| Login Screen | Register Button | Pass | Takes to registration screen as expected. |
| Login Screen | Login Button | Pass | Logs the user in if credentials are correct if not it displays an error message. |
| Registration Screen | Return to Login Button | Pass | App returns to login screen on button click. |
| Registration Screen | Register Button | Pass | Creates an account if user has entered correct information or displays an error if invalid info entered. |
| Menu (Shared across multiple pages once user is logged in.) | 'Home' link | Pass | Take user to the home page. |
| Menu (Shared across multiple pages once user is logged in.) | 'Artists' link | Pass | Takes user to artists page. |
| Menu (Shared across multiple pages once user is logged in.) | 'Events' link | Pass | Takes user to events page. |
| Menu (Shared across multiple pages once user is logged in.) | 'Music Lovers' link | Pass | Takes user to music lovers page. |
| Menu (Shared across multiple pages once user is logged in.) | 'My Profile' link | Pass | Takes user to my profile page. |
| Menu (Shared across multiple pages once user is logged in.) | 'Logout' link | Pass | Logs user out and returns them to login page. |
| Menu (extras only for venue owners.) | 'Add a venue' link | Pass | Takes venue owner to add a venue page. |
| Menu (extras only for venue owners.) | 'Make event' link | Pass | Takes venue owner to add an event page. |
| Home Page | Submit post button | Pass | Button posts the data as expected. |

| Location | Button/Navigation | Pass/Fail | Comment |
|---|---|---|---|
| Artists | Sort by 'Recommended Music Lovers' | Pass | Displays correct results |
| Artists | Sort by 'Nearby Music Lovers' | Pass | Displays correct results |
| Artists | Sort by 'Newly Joined Music Lovers' | Pass | Displays correct results |
| Artists | Click 'more info' link on a artist. | Pass | Correctly navigates to the individual artists page. |
| Artist Individual Page | 'Follow' button | Pass | Correctly follows artist. |
| Artist Individual Page | 'Un Follow' button | Pass | Correctly unfollows artist. |
| Events | Sort by 'Recommended Events' selected and sort button clicked. | Pass | Correctly sorts the events |
| Events | Sort by 'Nearby Events' selected and sort button clicked. | Pass | Correctly sorts the events |
| Events | Sort by 'Near Your Home' selected and sort button clicked. | Pass | Correctly sorts the events |
| Events | Sort by 'Date of Events' selected and sort button clicked. | Pass | Correctly sorts the events |
| Events | More info pressed on event. | Pass | Correctly navigates to the individual event. |
| Individual Event | 'Add event to favourites' clicked on event. | Pass | Correctly adds event to user favourites. |
| Individual Event | 'Remove event from favourites' clicked on event. | Pass | Correctly removes event from favourites. |
| Add a venue | Register venue button pressed | Pass | Creates venue on button click or displays error message if fields are missing. |
| Add an Event | Register event button pressed | Pass | Creates event on button click or displays error message if fields are missing. |

| Location | Button/Navigation | Pass/Fail | Comment |
|---|---|---|---|
| General | User swipes screen left | Pass | Shows menu as expected. |
| General | User swipes screen right when menu open | Pass | Closes menu as expected. |

Table 4.4: UI testing table

## 4.5 User testing

In terms of user testing the actual questions which were discussed in the experimental methods section (see Appendix 4), all of the questions were measure on a scale of 1 to 10 with 10 meaning that the user found the hybrid app fast and 1 meaning it was slow. table X shows the results.

| Participant | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|---|
| 2 | 9 | 6 | 8 | 4 | 9 | 9 |
| 3 | 10 | 9 | 9 | 4 | 8 | 4 |

Figure X gives a graphical representation of the results.

## 4.6 Resource Usage

When considering if hybrid mobile application it is also important to consider how much of the devices resources the app is using. For comparison the resource usage of Facebook has also been listed.
One of the main resources which was compared was the amount of CPU, as shown in Figure X when Facebook was looking for events it used 6% where as when the hybrid app which was developed was also 6%. It does need to be considered that Facebook is significantly larger than the hybrid app which was created consequently the hybrid app uses more CPU for less data than Facebook.
Figure X shows

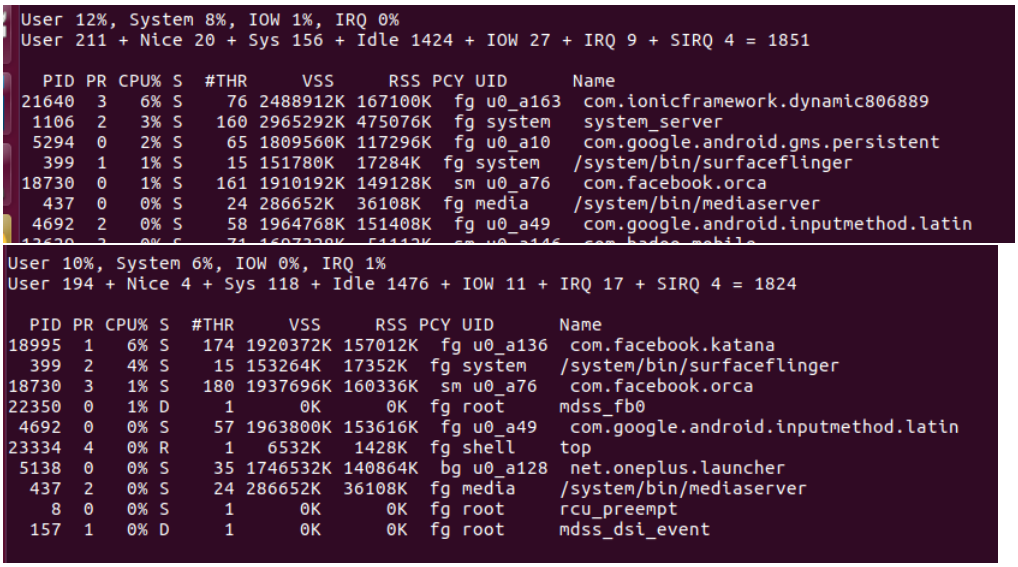


Figure 4.2: Images showing Facebook and Dynamic CPU.

# Chapter 5

# Evaluation

## 5.1 Are hybrid apps feasible

In determining whether hybrid apps are feasible alternatives to native apps it was important to consider both the advantages and disadvantages not just of the actual hybrid app technologies but of the actual development.

### 5.1.1 Advantages

- Plugins allow for simple access to platform native APIs.
- Porting to multiple platforms is relatively simple.
- More developers are familiar with web technologies and languages than platform specific languages.
- As with native apps, they can be on platform stores such as Google Play Store and App Store.
- Developer only needs one language for multiple platforms.
- Useful debugging features such as Google Chrome's developer tools.

### 5.1.2 Disadvantages

- Plugins do not give full control, such as issues with orientation on the camera plugin.
- Developer does not have full control over apps resource usage such as memory.
- Caching issues caused by the browser (which is displaying the app) caching even when developer tells app to not cache.
- Apps do generally seem slower for navigating around different pages as opposed to native apps.
- Styling can be challenging, where as styling a native iOS app using xCode's story board is rather straightforward.

### 5.1.3 Users thoughts

As shown in the testing section the general thoughts on the mobile app by general users were positive, the majority of users felt that the hybrid app which was created was as fast as a native app. This is rather interesting as there is generally a stigma in the development community that hybrid apps are significantly slower than

### 5.1.4 Feasibility

As the testing stage shown it would be fair to state that hybrid apps are feasible alternative apps for the majority of applications. This is because users did not seem to notice hybrid apps being significantly slower than native apps even when using a device's platform specific features such as a camera. There are clear times where hybrid apps would not be feasible such as for games or graphic heavy applications. Even though hybrid apps are feasible there are negatives in that native features do not always work properly such as the issue with the camera orientation on Samsung tablets.

## 5.2 Future Work

For the mobile app to be released commercially there is some more work which would have to be carried out. This work would include:

- Improving the server - If the app was released commercially then the server would have to be able to handle multiple requests at once and large data, this means stress testing would have to be performed.
- Push Notifications - Would add push notifications to tell users when upcoming events are happening.
- Promotion - Appropriate flyers, web blogs, publications etc. would be produced to promote the app.
- Email System - An email system would be set up, this would mean that a user would have to verify an account after creating it to ensure the email they provided was genuine.
- Password Reset System - A system which would allow for users to reset there password if they have forgotten them.

Another key thing which would have to be considered if the app was released in a commercial sense would be ethics. It would be important to have a terms and conditions which explains to the user exactly how their data would be used and that it would not be sold on. It would also be worth considering whether adding a minimum age to the app so that only users above a certain age can use the app, as the app does not have a profanity filter it would be important to ensure that young children aren't exposed to content which could be deemed offensive. Finally in relation to ethics it would possibly also be worth looking more into cyber bullying and how the app could aim to prevent cyber bullying.

## 5.3 Evaluation of app development itself

For the most part using the Scrum methodology worked well as it allowed for each different function of the app to be broken down into a story. At first it was rather difficult to estimate how

long each story would take but after doing a few stories estimation became much easier.
Using a plan driven methodology could have also worked well for developing the app and if a plan driven methodology had been used then a PHP framework would have probably been used rather than having lots of different PHP files.
The app development itself went rather smoothly once the basics of AngularJS had been learned. There were some problems which were encountered such as issues with caching and issues with converting postcodes to lat, long locations. These technical challenges were resolved mainly through research and looking at various online.

### 5.3.1 Requirements correctly identified

As this project was an investigation as well as the development of the app. It was important that the development was carried out to aid the investigation. In this case it certainly appears that the requirements were indeed correctly identified as the development of the app enabled a judgement about whether hybrid apps are feasible alternatives to native apps to be made.
There was more requirements which could have been added which would have allowed for a more solid judgement to be made. These requirements would have involved using more platform native api's so features such as push notifications along with access to a device's email could have been added to aid the judgement.

### 5.3.2 Design Decisions

There were some good decisions made in relation to the design of the app. Using ionic was certainly a good idea as it encourages the MVC framework and therefore means that the code created was easy to read and easy to understand.

### 5.3.3 Project aims achieved

The projects aims were achieved as I established that hybrid apps are feasible alternatives to native applications for the majority of purposes. The aims were achieved by correctly identifying requirements which enabled a sound judgement to be made.

### 5.3.4 What would do differently

### 5.3.5 Conclusion

The project was most certainly a success in that it proved that hybrid apps are feasible alternatives to native apps. As well as this through the actual development of a hybrid app the negatives and advantages of hybrid apps were made clear.
Whilst hybrid apps are feasible alternatives they do have some way to go especially in terms of giving the developer full control over a devices native features such as a camera. For such things to happen then hybrid apps probably need to become more common. In terms of making hybrid apps more common developers need to be made aware of exactly what hybrid apps are and when it can be useful to use them.

# Appendices

# Appendix A

# Third-Party Code and Libraries

The project was created using the menu ionic template, this was created by running the ionic start dynamic sidemenu from a terminal. [32]
As well as this an external API was used for coverting postcodes into lat, long. [7]
Code was taken from [13] to check the distance between two events.

# Appendix B

# Ethics Submission Assessment

## 2.1 Ethics Submission Assessment reference number: 6663

**AU Status**
Undergraduate or PG Taught

**Your aber.ac.uk email address**
sea6@aber.ac.uk

**Full Name**
Sean Anderson

**Please enter the name of the person responsible for reviewing your assessment.**
Reyer Zwiggelaar

**Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment**
rrz@aber.ac.uk

**Supervisor or Institute Director of Research Department**
cs

**Module code (Only enter if you have been asked to do so)**

**Proposed Study Title**
Are hybrid apps a feasible alternative to native apps?

**Proposed Start Date**
1/2/17

**Proposed Completion Date**
8/5/17

**Are you conducting a quantitative or qualitative research project?**
Mixed Methods

**Does your research require external ethical approval under the Health Research Authority?**
No

**Does your research involve animals?**
No

**Does your research involve human participants?**
Yes

**Are you completing this form for your own research?**
Yes

**Does your research involve human participants?**
Yes

**Institute**
IMPACS

**Please provide a brief summary of your project (150 word max)**
My app will be exploring whether using hybrid development are feasible alternatives to native development. I will be using human participants to review and judge the app.

**I can confirm that the study does not involve vulnerable participants including participants under the age of 18, those with learning/communication or associated difficulties or those that are otherwise unable to provide informed consent?**

# Appendix C

# Code Examples

## 3.1 Registration AngularJS Controller (register.js)

```
angular.module('register.controllers', [])
.controller('RegisterCtrl', function($scope, $ionicModal,
$http, $state, $ionicPopup) {
  var apiReturns;
  var api = "http://seananderson.co.uk/api/listgenre.php";
  $http.get(api).then(function(res) {
    var length = (res['data']).length;
    var select = angular.element(document.querySelector('#genre'));
    for (i=0;i<length;i++) {
      select.append('<option>' + res['data'][i] + '</option>');
    }
  })

   // Form data for the login modal
  $scope.register = {};

  //Generic popup as there may need to be
  //multiple popups for this page. Code is based of
  //ionic documentation for popup.
  popUp = function(title, message) {
    var alertPopup = $ionicPopup.alert({
      title: title,
      template: message
    });
    alertPopup.then(function(res) {
    });
  }

  //Return to login page i.e user has decided they dont need to register.
  $scope.returnToLogin = function() {
    $state.go('login');
  }
```

```
//Function which checks to make sure all fields
//on the form have been filled in.
$scope.checkFields = function() {
  if ($scope.register.fName==undefined){
    popUp("Field Missing", "First Name Field Was Not Entered");
    return false;
  }

  if ($scope.register.lName==undefined) {
    popUp("Field Missing", "Last Name Field Was Not Entered.");
    return false;
  }

  if ($scope.register.email==undefined) {
    popUp("Field Missing", "Email address was not entered.");
    return false;
  }

  if ($scope.register.password==undefined) {
    popUp("Field Missing", "Password was not entered");
    return false
  }

  if ($scope.register.cPassword==undefined) {
    popUp("Field Missing", "Confirmation password was not entered");
    return false;
  }

  if ($scope.register.pCode==undefined) {
    popUp("Field Missing", "Postcode was not entered");
    return false;
  }

  if ($scope.register.dName==undefined) {
    popUp("Field Missing", "Display Name was not entered");
    return false;
  }

  if ($scope.register.userType==undefined) {
    popUp("Field Missing", "User Type was not selected.");
    return false;
  }

  if ($scope.register.genre===undefined) {
    popUp("Field Missing", "You haven't selected your
    favoruite genre of music.");
    return false;
```

```
    }
    else {
      return true;
    }
  }

  //Function for regsitering genre of music which user likes.
  $scope.registerGenre = function() {
    var api = "http://seananderson.co.uk/api/registergenre.php";
    $scope.register.genre.forEach(function(genre) {
    var data = {
      email: $scope.register.email,
      genre: genre
      }
      $http.post(api, data).then(function(res) {
        console.log(res);
      })
    })
  }

  //Function which valides that the users postcode they have entered is cor
  $scope.checkPostcode = function() {
    var postcode = $scope.register.pCode;
    $scope.register.pCode = postcode.replace(/[\s]/g, '');
    if ($scope.register.pCode.length!=6) {
      if ($scope.register.pCode.length!=7) {
        if ($scope.register.pCode.length!=8) {
          popUp("Postcode is incorrect length must be 6,7 or 8 characters."
          return false;
        }
      }
    }
    return true;
  }

  $scope.checkPassword = function() {
    if ($scope.register.password.length<6) {
      popUp("Password Invalid", "Password length must be 6 or greater.");
      return false;
    }
    return true;
  }


  //Function for when the user clicks the validation button.
  $scope.doRegistration = function() {
    //Checks all necessary fields have values.
    if ($scope.checkFields()==false){
```

```
  return;
}
if($scope.checkPostcode()==false) {
  return;
}

if ($scope.checkPassword()==false) {
  return;
}

//Validates email address is a new email.
var api= "http://seananderson.co.uk/api/checkemail.php";
var data = {
  email: $scope.register.email
}
$http.post(api,data).then(function(res) {
  var apiResponse = JSON.stringify(res);
  var correct = "This email is fine"
  if (apiResponse.includes(correct)) {
    if ($scope.register.password==$scope.register.cPassword) {
      if($scope.register.userType=="Music Lover") {
        var type = 1;
      }
      else if ($scope.register.userType=="Artist") {
        var type = 2;
      }
      else if ($scope.register.userType=="Venue Owner") {
        var type = 3;
      }
      else {
        popUp('No User Type', 'No user type has been selected.' );
      }
      var api = "http://seananderson.co.uk/api/displayname.php";
      var data = {
        dName: $scope.register.dName
      }

      //Checks Display Name isn't already being used.
      $http.post(api, data).then(function(res) {
        apiReturns = JSON.stringify(res);
        var incorrect = "This user name has already been used";
        if (apiReturns.includes(incorrect)) {
          popUp('This display name has already been
          used please choose another
          display name.');
        }
        else {
          var api = "http://seananderson.co.uk/api/register.php";
```

```
            var data = {
              firstName: $scope.register.fName,
              lastName: $scope.register.lName,
              email: $scope.register.email,
              type: type,
              password: $scope.register.password,
              postcode: $scope.register.pCode,
              dName: $scope.register.dName
            }
            $http.post(api, data).then(function(res){
              apiReturns = JSON.stringify(res);
              if (apiReturns.includes('New user added succesfully')>=0) {
                localStorage.setItem('email', $scope.register.email);
                localStorage.setItem('dName', $scope.register.dName);
                if ($scope.registerGenre!=undefined) {
                  $scope.registerGenre();
                }
                $state.go('app.profile');
                popUp('Welcome', 'Welcome to Dynamic please
                fill in your profile page and then
                follow some users');
              }
            })
          }
        })
      }
      else {
        popUp('Passwords do not match', 'The confirmation
        password is not the same as the initial
        password you entered');
      }
    }
    else {
      popUp("Email is not valid", "Please use a valid
      email address which hasn't
      already been used");
    }
  })
 }
})
```

## 3.2 Registration API (register.php)

```
<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: Origin,
X-Requested-With, Content-Type, Accept");
```

```
header('Access-Control-Allow-Methods: GET, POST, PUT');

include "config.php";
$connection = new mysqli($server, $dbuser, $dbpass, $dbname);

if ($connection->connect_error) {
    die ("Connection to the database failed."
    . $connection->connect_error);
}

if ($_SERVER['REQUEST_METHOD'] == 'POST' && empty($_POST))
    $_POST = json_decode(file_get_contents('php://input'), true);

$firstName = mysqli_real_escape_string($connection, $_POST['firstName']);
$lastName = mysqli_real_escape_string($connection, $_POST['lastName']);
$email = mysqli_real_escape_String($connection, $_POST['email']);
$type = mysqli_real_escape_String($connection, $_POST['type']);
$dName = mysqli_real_escape_String($connection, $_POST['dName']);
$password = mysqli_real_escape_String($connection, $_POST['password']);
$password = password_hash($password, PASSWORD_DEFAULT);
$postcode = mysqli_real_escape_String($connection, $postcode);

if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $sql = "INSERT into users (firstname, lastname, email, type,
    displayname, password, postcode)
    VALUES ('$firstName', '$lastName', '$email',
    $type, '$dName', '$password', '$postcode')";

    if ($connection->query($sql) === TRUE) {
        echo "New user added succesfully";
    }
    else {
        echo "Error: " . $sql . "</br>" . $conn->error;
    }
}

else {
    echo "Email is not valid";
}

$connection->close();
?>
```

## 3.3 Unit Tests for Login and Register

```
describe('LoginController', function() {
```

```
    var controller,
        loginServiceMock,
        stateMock,
        ionicPopupMock;

    beforeEach(module('app'));

    describe('#doLogin', function() {

        // TODO: Call doLogin on the Controller

        it('should call login on dinnerService', function() {
            expect(loginServiceMock.login).toHaveBeenCalledWith('sea6@aber.a
        });

        describe('when the login action is performed,', function() {
            it('if successful, should change state to home', function() {

                // TODO: Mock the login response from DinnerService

                expect(stateMock.go).toHaveBeenCalledWith('app.home');
            });

            it('if unsuccessful, should show a popup', function() {

                // TODO: Mock the login response from DinnerService

                expect(ionicPopupMock.alert).toHaveBeenCalled();
            });
        });
    })
});
```

## 3.4 Login Unit Test (logintest.js)

```
describe('LoginCtrl', function() {

    var controller,
        loginMock,
        stateMock,
        ionicPopupMock;

    beforeEach(module('app'));

    beforeEach(module('#'));

    describe('#doLogin', function() {
```

```
        it('Should Login using correct details', function() {
            expect(loginMock.doLogin).toHaveBeenCalledWith('sea6@aber.ac.uk'
        });

        describe('when the login action is performed,', function() {
            it('if successful, should change state to home', function() {
                expect(stateMock.go).toHaveBeenCalledWith('app.home');
            });

            it('if unsuccessful, should show a popup', function() {

                expect(ionicPopupMock.alert).toHaveBeenCalled();
            });
        });
    })

    describe('#register', function(){

        it('Should go to register screen', function(){
             expect(stateMock.go).toHaveBeenCalledWith('register');
        })
    })


});
```

# Appendix D

# Questionnaire

## 4.1 Questionnaire handed out for user testing

Question 1: On a scale of 1 to 10 with 1 being slowest and 10 being very fast in your opinion how quick is the app at processing the registration form?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

Question 2: On a scale of 1 to 10 with 1 being significantly slower than a native app, 5 being about the same time as a native app and 10 being very fast, faster than a native app in your opinion how quick is the app at processing the registration form?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

Question 3: On a scale of 1 to 10 with 1 being slowest and 10 being fast in your opinion how long does the process of uploading a new profile picture take?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

Question 4: On a scale of 1 to 10 with 1 being significantly slower than a native app, 5 being about the same time as a native app and 10 being very fast, faster than a native app in your opinion how long does the process of uploading a new profile picture take?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

Question 5: On a scale of 1 to 10 with 1 being slowest and 10 being fast in your opinion how long does the process of viewing events and sorting them by using nearest events (GPS) take?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

Question 6: On a scale of 1 to 10 with 1 being significantly slower than a native app, 5 being about the same time as a native app and 10 being very fast, faster than a native app in your opinion how long does the viewing events and sorting them by using nearest events (GPS) take?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

# Annotated Bibliography


[1] "Camera api," https://developer.android.com/guide/topics/media/camera.html, accessed on 28th March.

Android Camera API Documentation

[2] "Camera: Wrong orientation with android," https://forum.ionicframework.com/t/camera-wrong-orientation-with-android/8583, accessed on 28th March.

Forum post dicussing the correct orientation option not working

[3] "Git hub features," https://git-scm.com/about, accessed on 25th March.

Git features

[4] "Github," https://github.com/, accessed on 25th March.

Github website

[5] "Gitlab," https://about.gitlab.com/, accessed on 25th March.

Gitlab website

[6] Apache, "Cordova framework," https://cordova.apache.org///, 2017 (Most recent update.).

Official website for Apache Cordova Framework.

[7] ——, "Postcode api," http://api.postcodes.io/, 2017 (Most recent update.).

Open Source API for converting postcodes to lat, long.

[8] Apple, "Apple developer site," https://developer.apple.com/develop/, 2017.

Apple Developer Site; explains how Native apps are made using Swift. (Please note the year is this year as the site is constantly updating.)

[9] V. Bhagat, "7 best hybrid app development frameworks for 2017," http://www.pixelcrayons.com/blog/7-best-hybrid-app-development-frameworks-for-2017/, March 2017, accessed 18th March 2017.

Blog post which compares different hybrid app frameworks.

[10] A. Biharisingh, "How to write automated tests for your ionic app," https://gonehybrid.com/how-to-write-automated-tests-for-your-ionic-app-part-2/, accessed on 25th March.

Site which explains how unit testing can be done in ionic

[11] J. Bowes, "Kanban vs scrum vs xp  an agile comparison," https://manifesto.co.uk/kanban-vs-scrum-vs-xp-an-agile-comparison/, July, accessed = 1st March 2017.

Post which explains the fudimental differences between Scrum, Kanban and XP.

[12] Drupal, "Drupal site," https://www.drupal.org/docs/7/mobile/native-mobile-apps, 2017.

Drupal 7 Mobile Guide; explains various things about native apps including how they can access native APIs.)

[13] GeoDataSource, "Distance between points." http://www.geodatasource.com/developers/php, 2017 (Most recent update.).

Code for working out distance between points.

[14] A. (Google), "Android developer," https://developer.android.com/develop/, 2017.

Android Developer Site; gives information about native apps including that they are made in Java. (Please note the year is this year as the site is constantly updating.)

[15] I. in Action, *Ionic in Action*. Manning, 2016, pp. 3–6.

Section of book explains the different types of hybrid apps and the benefits to each type.

[16] J. H. J. Ben Schafer, Dan Frankowski and S. Sen, "Collaborative filtering recommender systems," *The Adaptive Web: Methods and Strategies of Web Personalization*, pp. 291–295, 2007, accessed on 24th April.

Journal article which explains about what collaboartive filtering is and how it works

[17] A. H. A. G. J. Bobadilla, F. Ortega, "Recommender systems survey," *Knowledge-Based Systems*, 2013, accessed on 22nd April.

Journal which explains how a Recommender based system was created.

[18] A. Karr, "Color and user experience," http://interactions.acm.org/blog/view/color-and-user-experience, website which explains different colours.

Site which explains about different colours

[19] M. Korf and E. Oksman, "Understanding your mobile application development options," https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options, June 2016, accessed 10th February 2017.

Blog post gives a run-down of different development techniques and explains about hybrid apps accessing native API's.

[20] M. (listed on site.), "Manifesto for agile software development," http://agilemanifesto.org/, 2001.

The Agile Manifesto which clearly highlights welcoming change over following a plan.

[21] A. Ng, https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations, accessed on 22nd April.

Lecture given at Sanford about Machine Learning

[22] R. Rodger, *Beginning Mobile Application Development in the Cloud*. Wiley, 2011, p. 2.

Section of book explains the three different types of mobile app development.

[23] D. Rust-Smith, "Should you build phonegap or native," http://davidrs.com/wp/should-you-build-phonegap-or-native/, March 2014, accessed 14th March 2017.

Page which discusses whether to use Phone Gap or Native.

[24] K. Schwaber and J. Sutherland, *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game*. N/A, July 2013, pp. 3–16.

Scrum Guide which illustrates how the Scrum methodology should be applied to a project.

[25] J. Stangarone, "The mobile app comparison chart," http://www.mrc-productivity.com/blog/2016/06/the-mobile-app-comparison-chart-hybrid-vs-native-vs-mobile-web/, June 2016, accessed 10th February 2017.

Blog post which provides a nice comparison of the different types of app development, has a table which is a visualisation of the different features available to each type of app development technique.

[26] Unknown, "Ginger bread man image," https://pixabay.com/en/gingerbread-man-ginger-bread-man-213742/, free to use for commerical use. Accessed = 16th April.

Generic image used for users default image.

[27] Unkown, "Schneiderman's 8 golden rules," https://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html, website which lists Schneiderman's 8 Golden Rules.

Schneiderman's 8 Golden Rules

[28] Various, http://ngcordova.com/docs/plugins/camera/, accessed 20th March 2017.

Documentation in relation to the Cordova Camera Plugin.

[29] ——, "jquery mobile," http://https://jquerymobile.com/, 2016 (Most recent update.).

Official website for JQuery Mobile.

[30] ——, "Apache cordova framework," https://framework7.io////, 2017 (Most recent update.).

Official website for Apache Cordova Framework.

[31] ——, "Ionic framework," https://ionicframework.com//, 2017 (Most recent update.).

Official website for Ionic Framework.

[32] ——, "Ionic framework," https://ionicframework.com/docs/v1/getting-started/, 2017 (Most recent update.).

Guide to getting started with ionic v1.

[33] O. Yevtushenko, "What are the popular types and categories of apps," https://thinkmobiles.com/blog/popular-types-of-apps/, Dec. 2016, accessed 10th February 2017.

Blog post gives a run-down of different app development techniques.