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 development. This will involve exploring, examining and developing a mobile application in a hybrid manner. For this project, a music social media app will be developed. The app will have various features e.g. 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 which will be in PHP.

1.1 Background

There are three different types of mobile applications: native apps, web apps and hybrid apps [9] [6] [17]. A native app is an app which is developed in a platform specific language and is usually downloaded from a device's app store e.g. Google Play Store for Android and the App Store for iOS devices. As native apps are developed in a platform specific language e.g. Android apps are developed in Java [5] and iOS apps are developed in Swift (formerly Objective C) [2] this means that code has to 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 [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 where 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 [12] [9] .

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 [12] [7].

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.

Hybrid mobile apps can access native APIs. This means that the middleware technology has to be set up to do so. As a result 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 [8].

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. This is why 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 I felt was most appropriate for this project this is because Scrum doesn't require planning of how the software will be developed as opposed to other methodologies such as eXtreme programming which actually informs the programmer that they should be using test driven development and pair programming. Whilst Scrum was the methodology chosen the project is really used an adaptation of scrum. This is because Scrum involves multiple roles in a team and this was a single person project. [4]

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) [11]. 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. Anybody 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 of 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.
- 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 [10] [1] [14] [16] [15] [3].

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 lean. Nice native like styles built in 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 that in mind.
The model of the app would be the database which is stored on the backend of the system. The view is the html files and the controllers are both the Angular JS controllers and the RESTful PHP API which I created.

2.1.1 Overall Architecture

Figure X shows the architecture as whole. Whilst figures X separates the model, view and controller.

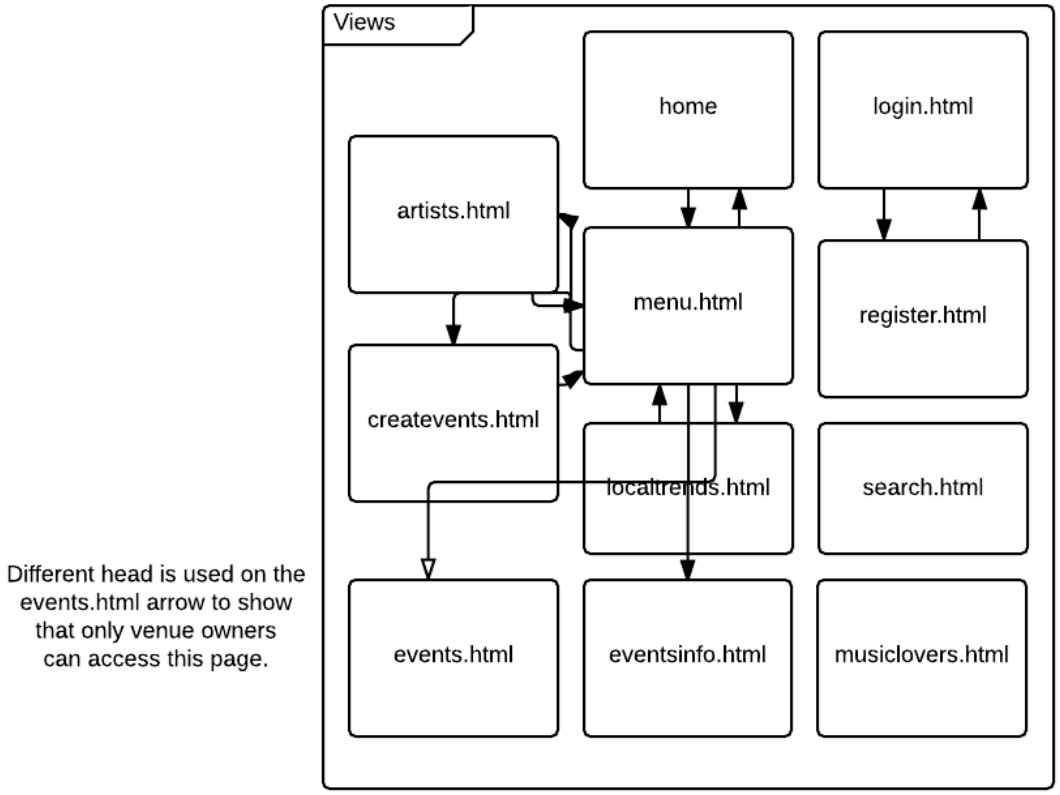

Figure 2.1: UML Diagram which illustrates the view

Figure X (created using `https://www.lucidchart.com/`) 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.

2.1.2 Some detailed design

The following figure highlights the different relationships between the tables (model) for events.

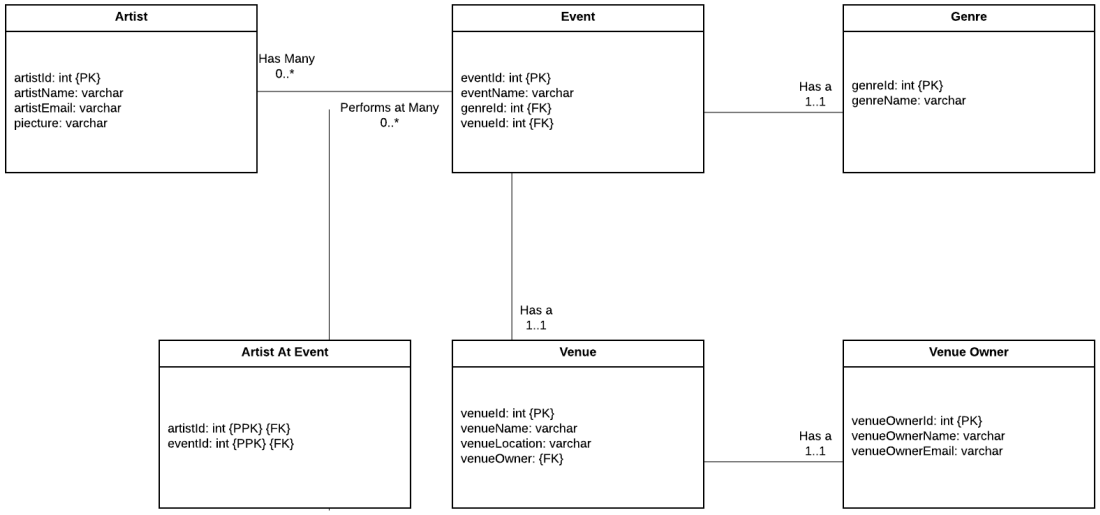

Figure 2.2: UML Diagram for the model in relation to events.

Figure 2.2 is a UML diagram (created using `Createdusinghttps://www.lucidchart.com/`) which shows the appropriate relationships between the different tables in the database in relation to an event.

2.1.2.1 Even more detail

2.1.3 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 important colours.

2.1.3.1 Schniderman Principles

Figure X (created using `https://www.fluidui.com`) shows some UI mockups which were created.

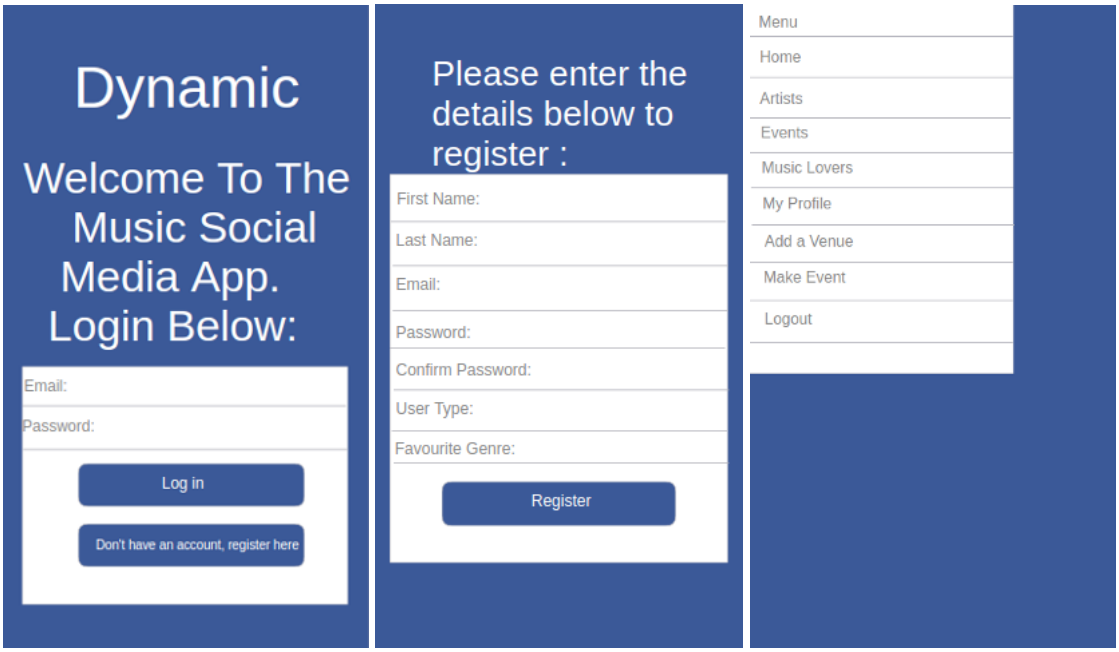

Figure 2.3: User interface mock up

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.

Chapter 3

Implementation

This section discusses the implementation of the music social network it is split into the multiple stories which have previously been listed.

3.1 The Registration System

3.1.1 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 they could just simply be added to the 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 and as genres to users is a many to many relationship a users_genres table was stored 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 the 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 sql commands which are ran from within the php. Some validation was done in the php file to check that things such as the email given are valid emails however I wanted to do most of the validation actually on the front end as this would allow for error messages to get out to the user quicker.
As at this stage in time there was no front end it was not possible to test that sending post data from an app would work. So, a piece of software called postman (which is a Google Chrome extension) was used as this software allows for post data to be pushed to a website as illustrated in figure X. 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 X.
Having completed this stage I did notice a significant weakness in using hybrid technology, the act

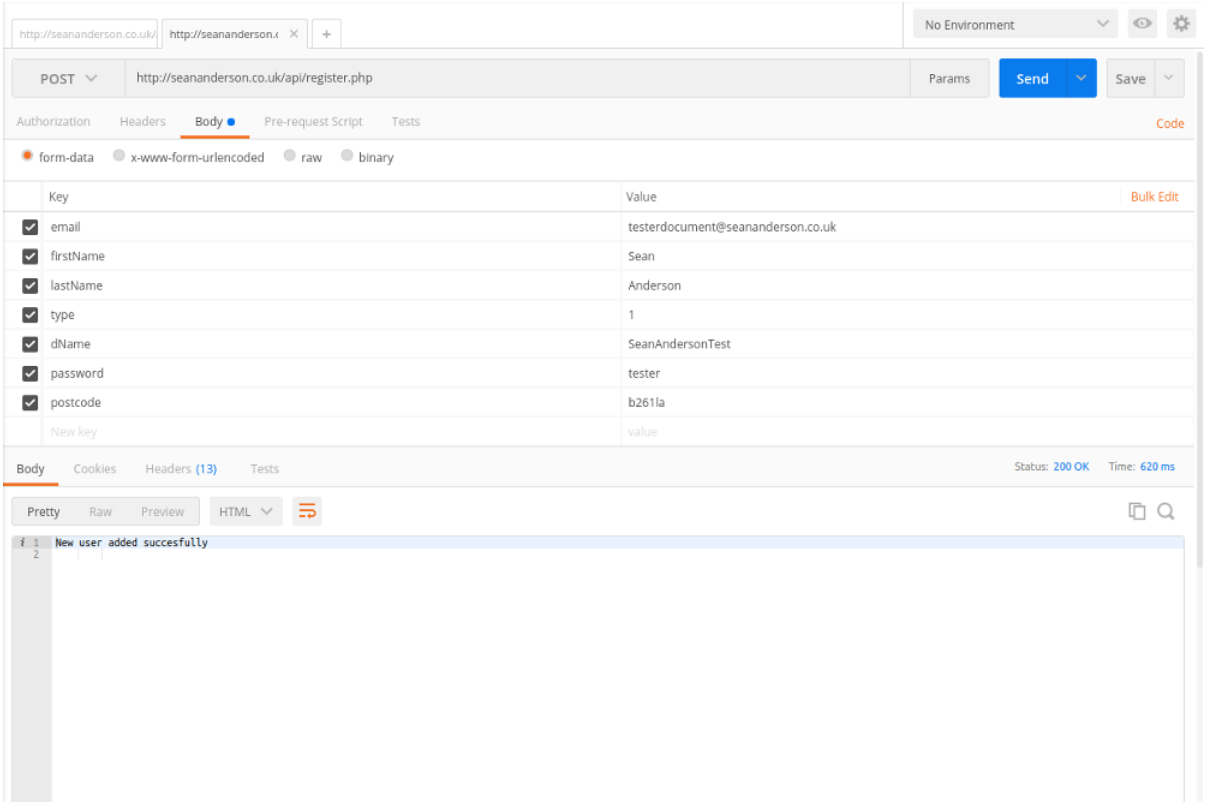

Figure 3.1: Postman testing register.php

Figure 3.2: User added to the database

For setting up registering different users' genres the system was implemented in the exact same way as the registration system only that there was far fewer variables. The login API simply checks that the user provides an email and password and that they match and if they do match returns the user details and if they don't match returns a message saying that the password is not correct.

3.1.1.1 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 will be imputed for the user (for all backend files for the music social media) 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.

3.1.2 Front End

After the back end for the registration of the app was created development then started on the front end of the app. The default ionic menu bar template was used as this would allow for a menu bar to appear on multiple pages (like in my 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 those 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.

```
$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 ar
  }
})
```

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. (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 3.3 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.)

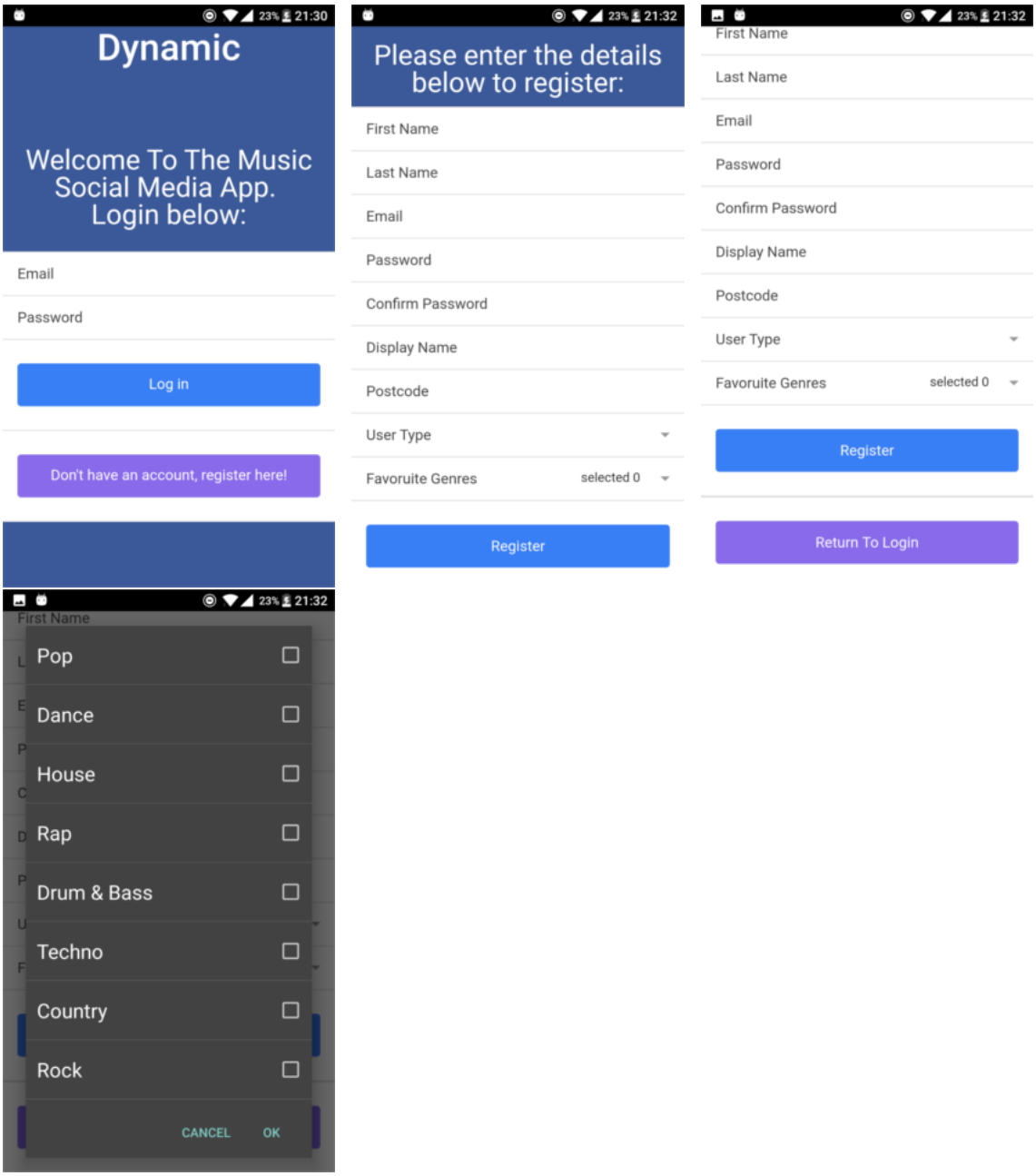

Figure 3.3: Screenshots of registration system.

3.1.3 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 so I ensured that the user had filled in all of the appropriate boxes and used a regex to ensure that the email address they entered was correct.
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 3.4 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.

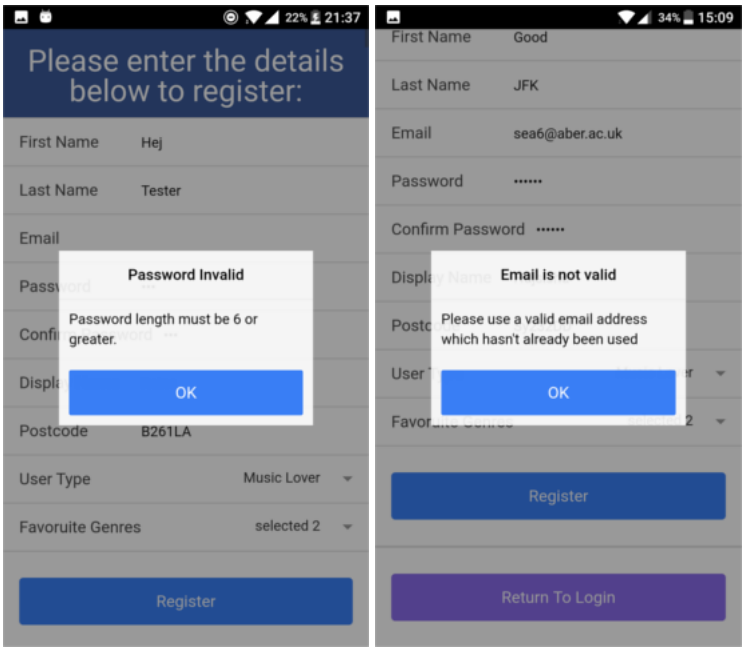

Figure 3.4: Validation Screenshots

3.2 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 the php file runs a sql command which updates the users table to have the correct information for the bio.

3.2.1 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, and the ask of transferring that photo to the backend.
Installing the cordova camera plugin [13] 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 plguin 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 3.4 highlights the two different buttons and what happens when they are clicked.

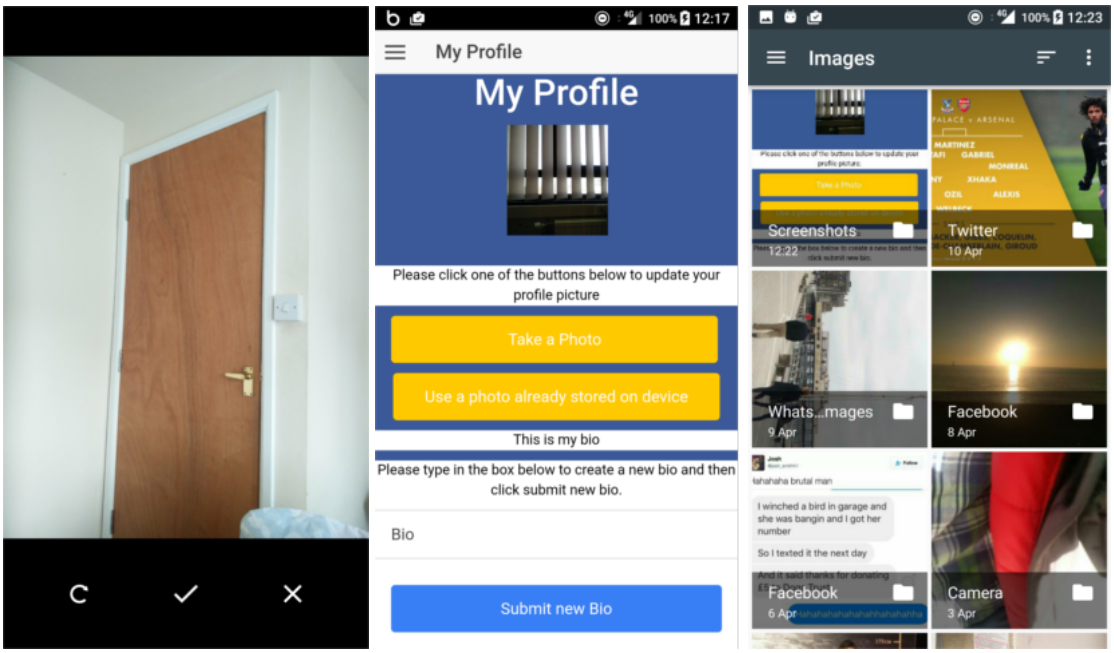

Figure 3.5: Screen shot of camera functionality.

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 the photo to upload was to create a API file, this file was entitled imageupload.php and is hosted at seanandersson.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 (seanandersson.co.uk/api/uploads). Afer 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 significantly longer than it takes when using a native app. This will be discussed in more depth during the testing section of this report.

3.2.1.1 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.
A variety of different ways was used to try and tackle the 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.

```
$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>');
})
```

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

3.2.1.2 Problem with Different Devices

The image 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.

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

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

3.3 Styling

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. 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.)

3.4 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.

3.5 Actual Implementation vs Plan

Chapter 4

Testing and Experimenting

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.
Have you tested your system on "real users"? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.
The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing and Experiments

4.2 Automated Testing

4.2.1 Unit Tests

It is important to unit test each of the 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.

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.
There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.
Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

Appendices

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.
As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.
Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.
As an example, you might include a definition such as:
Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the clients existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

Appendix B

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 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [**?**].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*-------------------------------------------------*/
  /* Minimum Standard Random Number Generator        */
  /* Taken from Numerical recipies in C              */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity   */
  /* Always call with negative number to initialise  */
  /*-------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
```

```
double temp;

if (*idum <=0)
{
  if (-(*idum) < 1)
  {
    *idum = 1;
  }else
  {
    *idum = -(*idum);
  }
  idum2=(*idum);
  for (j=NTAB+7;j>=0;j--)
  {
    k = (*idum)/IQ1;
    *idum = IA1 *(*idum-k*IQ1) - IR1*k;
    if (*idum < 0)
    {
      *idum += IM1;
    }
    if (j < NTAB)
    {
      iv[j] = *idum;
    }
  }
  iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
  *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
  idum2 += IM2;
}
j = iy/NDIV;
iy=iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
{
  iy += IMM1;
}
if ((temp=AM*iy) > RNMX)
{
```

```
    return RNMX;
  }else
  {
    return temp;
  }
}
```

Annotated Bibliography

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

Official website for Apache Cordova Framework.

[2] 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.)

[3] 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.

[4] 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.

[5] 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.)

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

Documentation in relation to the Cordova Camera Plugin.

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

Official website for JQuery Mobile.

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

Official website for Apache Cordova Framework.

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

Official website for Ionic Framework.

[17] 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.