Is Hybrid Mobile App Development A Fesisable 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 .....

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 .....

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 apps 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: a native app, a web app and a hybrid app. 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 and iOS apps are developed in Swift (formerly Objective C) this means that code has to be written in multiple languages to develop the app for different platforms. Native apps allow for platform APIs to be used and therefore the developer has access to resources such as the device's camera or contact book.

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' API therefore it is very challenging for a developer to use native resources such as a device's camera.

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.

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 want 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 API's however 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 that they choose the correct framework for developing hybrid apps as they 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 hybrid apps performance 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.

As this project is an investigation the requirements of the app may change as 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.

Scrum was the agile methodology I felt was most appropriate for this project. 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.) Whilst Scrum was the methodology chosen the project didn't strictly use Scrum as Scrum requires multiple different people in a group to do multiple different roles such as a Scrum Master.

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. The below figures show the templates:

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

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

January 31, 2017

Contents

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

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 sprint 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 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.
- System for creating events 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.

Chapter 2

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

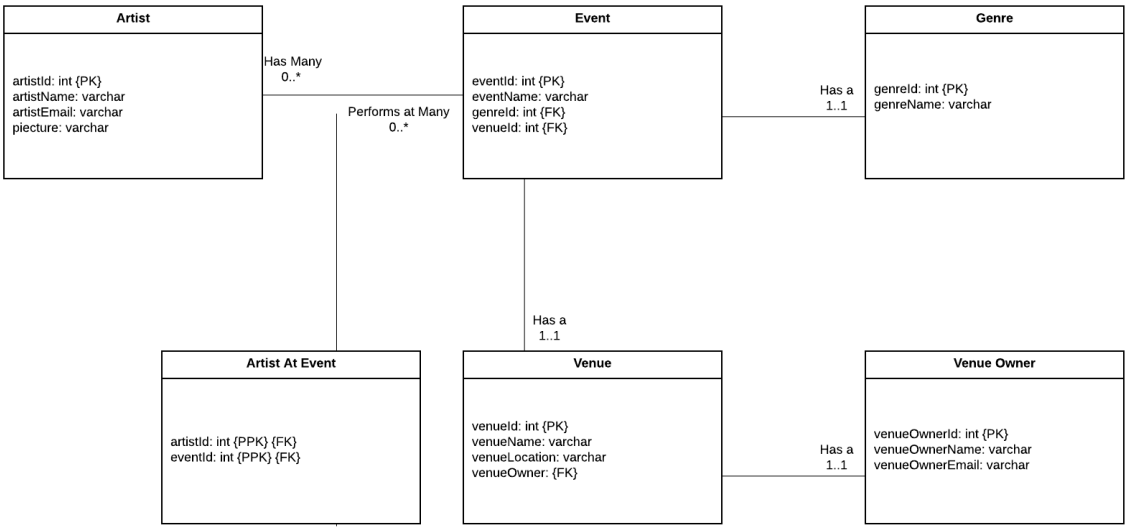

Figure 2.1: UML diagram which illustrates the appropriate relationships between the different tables in the database in relation to an event.

2.1 Overall Architecture

2.2 Some detailed design

2.2.1 Even more detail

2.3 User Interface

2.4 Other relevant sections

Chapter 3

Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

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

4.2 Automated Testing

4.2.1 Unit Tests

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

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

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

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;
  }
}
```