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 .....

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 Application Programming Inteface (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 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 API's. 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.

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.

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. 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

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

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 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

TODO: This section.

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 2.1 shows the architecture as whole. Whilst figures X seperate the model, view and controller.

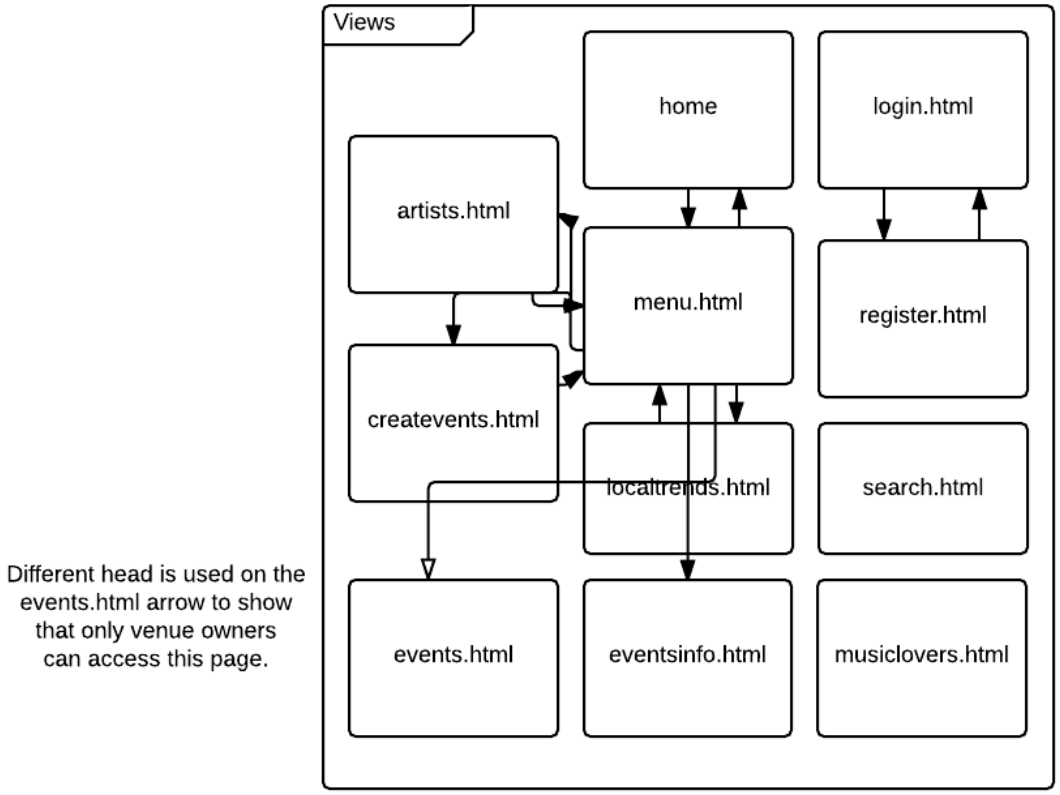

Figure 2.1: UML Diagram which illustrates the view. Created using https://www.lucidchart.com/

Figure 2.1 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.

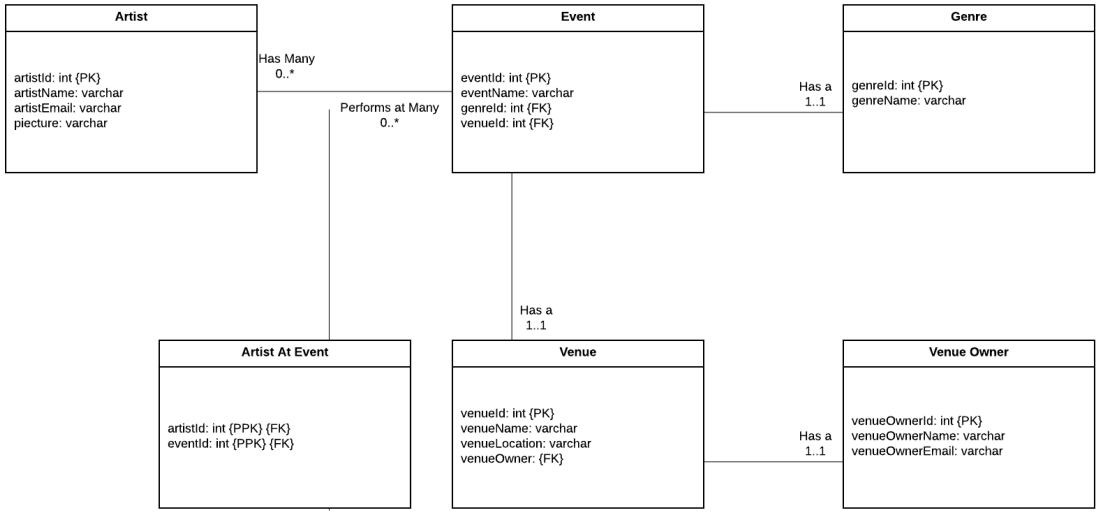

Figure 2.2: UML Diagram for the model in relation to events. Created using https://www.lucidchart.com/

Figure 2.2 is a UML diagram 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

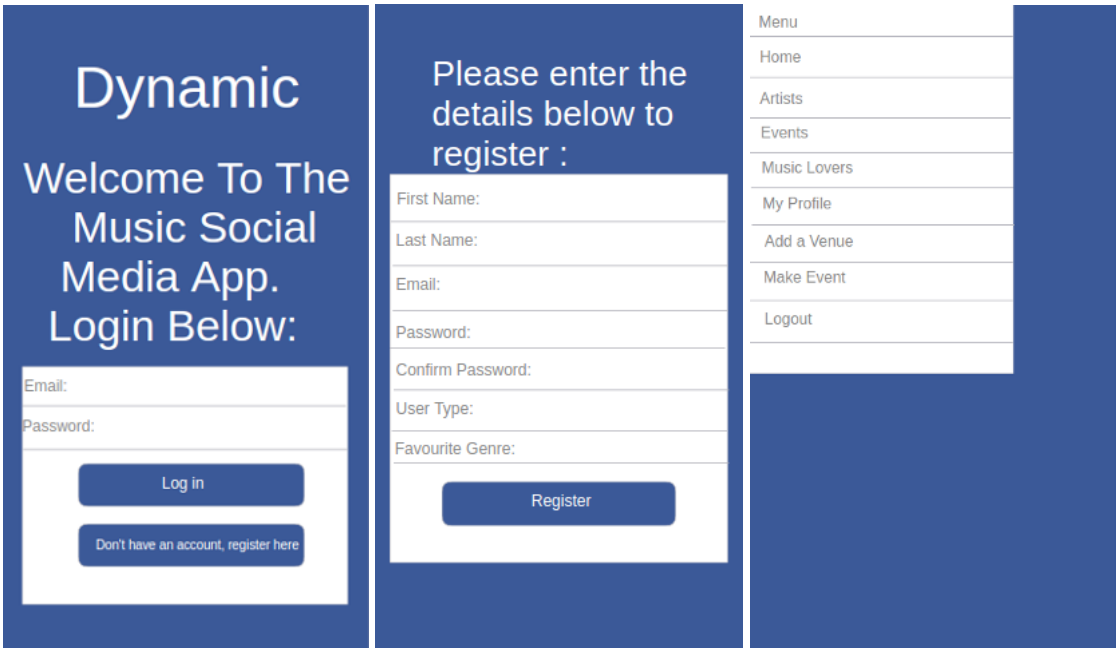

Figure 2.3: User interface mock up designed using https://www.fluidui.com

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 Story 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 there picture. The genres table was simply a table which contained different genre's 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.

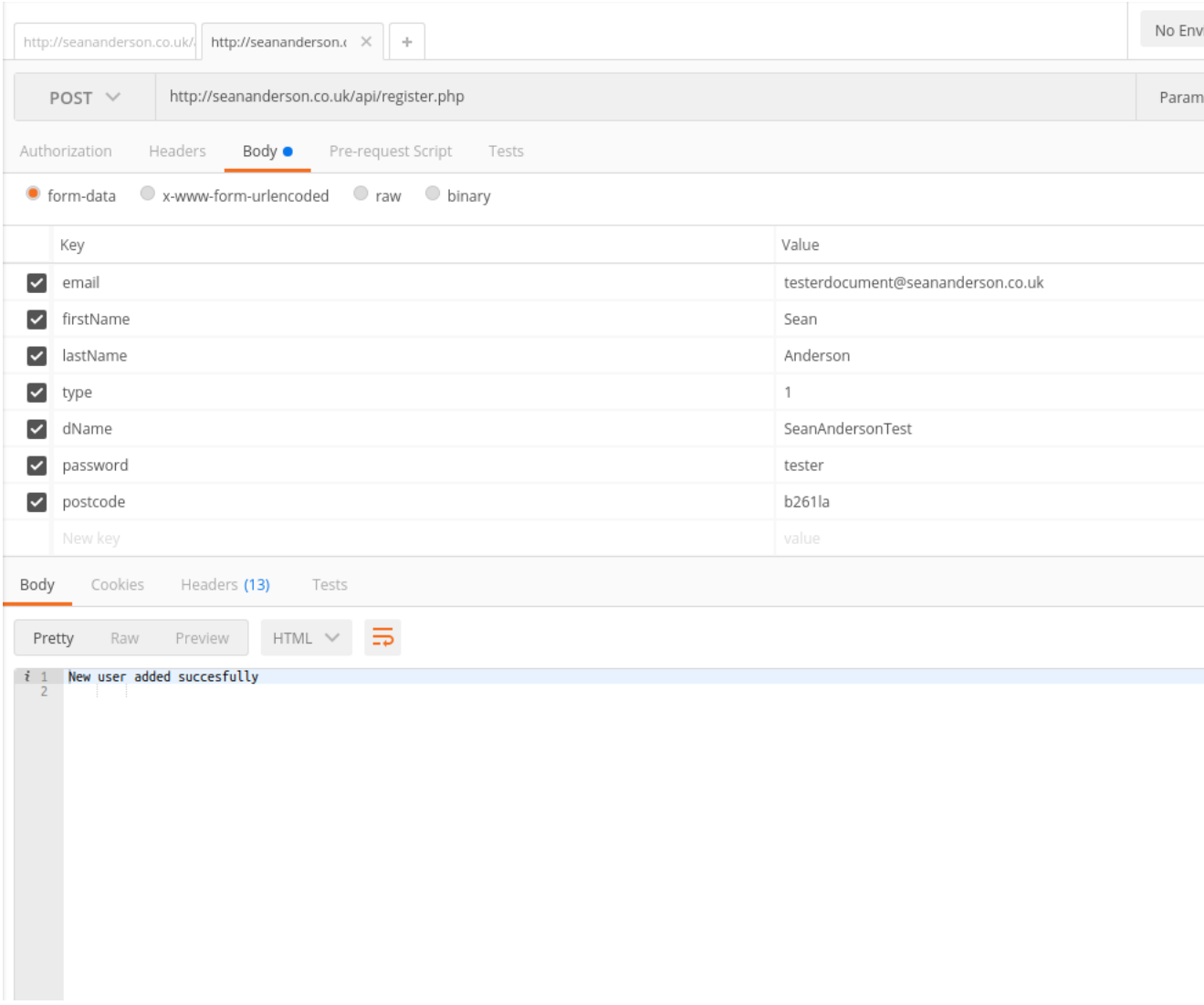

Figure 3.1: Postman testing register.php

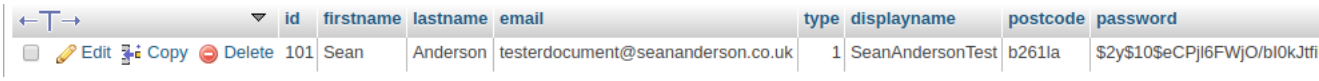

Figure 3.2: User added to the database

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 it's 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.

//TODO: Add sample code here.

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`) //TODO: Add code here.

Figure X 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.)

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 X displays an example of what happens when a user tries to register an account with an email address which has already been used.

3.2 Neil Info

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;
   }
 }
```