**Technical report**

by Jan Hanson

1339404

Section 1 - technology

What have you learned about the technologies used?

Android

At the start of the project I had read little about android programming except that it was essentially a framework based on a modified version of java running on the Dalvik vm as opposed to the normal java vm. This is true but there is much more to it than I initially realised.

I learned that the android framework has a built in MVP type structure, with the views being defined by XML files, these in turn are controlled by activities and fragments which make up the presenter element of the architecture (presenter as opposed to controller as there is generally a one to one relationship between a view and a presenter and the input comes directly into the view as opposed to the controller).

hardware functionality is usually implemented by inheriting from the appropriate interface doing setup appropriate to the particular hardware element in question, and then querying the hardware.

Overall the Android framework is pretty clear to build functionality around, and is exceptionally well documented, googles' android developer website is amazingly extensive and covers pretty much every base when it comes to learning android and using the api documentation.

The last thing I learned about android is that it is significantly easier to develop natively on an android device rather than on an X86 architecture, even though eclipse/android studio provide all the functionality you could possibly need, there seem to be endless problems when loading projects, whereas using AIDE natively on an android tablet there was no need to use the adb to push the app to a device for testing or use the infernal emulator (it really isnt a practical way to get any work done).

Jetty

At the start of the project I had never heard of jetty before, so the first thing I learned was that jetty is a java based lightweight http server for the writing of java servlets. What makes jetty a little bit special is that it can be embedded into a java project rather than having to setup a separate web server and running your application through that.

There are two ways to set up jetty from what i can tell, the first is to set up jetty as a webserver and then to write servlets that run on that jetty server, the other is to write your application and then to have a jetty server within that application that takes handles specified request from the application, a subtle difference but an important one as I came to find out.

Jetty has an easy way to handle HTTP requests and responses i.e. by overriding certain methods and passing response/request parameters to these methods. Jetty handles the guts of it, the user then just has to instantiate request/response objects and use them as required for the specified task. This allowed me to write one servlet as an interface the application to the jetty http server, the requests could then be passed down to other classes to do the actual computations, then passed to other classes to format the results and passed back out to the servlet which handles the http response. This makes the whole setup conceptually elegant; the application submits requests to the http server which then passes those requests on to the application server, and responses from the application sever go the other way.

MySQL/MariaDB

Being a user of linux, and a fan of opensource software i used MariaDB in conjunction with JDBC on my development server and then when transferring to the group server all I had to do was change the JDBC driver and it worked more or less straight away. This is because MariaDB is a drop in replacement for oracle's MySql.

There was however one rather confusing problem that cropped up in transferring code from my development server to the production group server, and that was to do with the use of java's ResultSet. On my development server, i opened a JDBC connection to the server, but as i didnt wish to waste resources (network/memory bandwidth, CPU time etc) i closed the database connection after the execution of a query, this seemed to me a good way to do things as then i dont keep a DB connection open the whole time the application is in use. However on transferring this code to the production group server errors cropped up telling me that certain operations could not be carried out while the ResultSet was closed. Thanks to Michael Scott I managed to figure out a way to fix this, by first copying the data in the ResultSet to another object and then closing the connection. However it was a problem that i was not expecting between the two database implementations. Other than that JDBC is a straightforward way to connect to a database.

How did you connect your android app to the server?

Rather than using Jetty and java servlets to connect to the database stored on the groups linux server, I decided to interface directly with database server through JDBC, thus eliminating the need to use a webview in the android application or unnecessary code for turning a http response into text usable by the application. This also has the benefit of reducing the load on the server, not that that is ever going to be a concern for a non-production application such as mine; but it would be relevant to a future employer when showcasing my previous work.

In the model element of my MVP application architecture I included the code to connect to and interact with the database server (www.jpchanson.jetos.com:8080/CONTENT/dev-ARC-3.php). It consists of using a 'Connection' and 'DriverManager' object to initialise the JDBC connection and connect to the database, using a 'PreparedStatement' to execute a query (in my case creating/viewing posts etc) and a 'ResultSet' to store the results of the query. This data is then copied to an appropriate object(again, in my case, a post object). Upon completion the database connection is closed, so that resources dont stay locked up unnecessarily.

The downside to this approach is that if(when) the database implementation, an update to the application would be necessary, probably just changing the driver. However in the event that the database is changed to something not supported by JDBC, then major changes will be needed to make the app compliant. However if I had gone with a http implementation rather than a raw JDBC connection, then only server code would have to change as long as the format of the http response stays the same. I don't consider this to be too likely, however, as social networking data is inherently relational, switching to a non relational database implementation would be unnecessary.

What data did you generate and/or use? how was it used to support the application?

Currently, the app generates post data, this is both created by and used by the application: Post data is created when a user wants to update their status or notify friends of something etc. On the post creation screen there are editText boxes for the title and the content of the post, the other data generated by the application is the location that the application is currently in and the current time. I would have liked to implement more media, user icons, etc but due to time constraints this was not possible, as more time than I would have liked was tied up figuring out the augmented reality functionality.

This same data is used by the application for viewing posts, on all of the views, camera preview, map view and the list view. This is the core functionality of any social networking application. However with the fact that I was only able to implement this application is both disappointing and makes the app severely lacking. In order to be even semi usable, the ability to view friend details, and create/update user information, and it is regrettable that this was not done.

How would you develop the project differently (from a technical perspective) if you had to attempt a similar project?

If I had to attempt a similar project, the first thing that I would do differently is change the emphasis of my implementation. In this project I spent too much time figuring out the 'shiny' functionality and not enough time concentrating on the core functionality such as those mentioned above. So rather than starting with the augmented reality functionality and the views, I should have started with the model, taking into account that there would ultimately be a database backend providing the data. Then implement the business logic parts of the presenter and those views that relate to the input of data to the system, such as the post construction, user signup/data, and the friends system. Only after this (or the best part of this) was implemented, should I have implemented the data views such as the camera view and the map view.

This should ensure that i end up with product that has the necessary guts it may not be as pretty or impressive to the user, but it would make it much easier to implement the 'shiny' functionality at a later date. Rather than having a product with impressive looking features but less to back it up than I would like. It is much better to have a product that with unimpressive features that is easy to extend, than to have a product that looks more impressive but is missing backend functionality that is harder to implement.

How did you use models and design patterns in your work? (www.jpchanson.jetos.com:8080/CONTENT/dev-ARC-1.php)

(www.jpchanson.jetos.com:8080/CONTENT/dev-ARC-2.php)

(www.jpchanson.jetos.com:8080/CONTENT/dev-ARC-3.php)

I used a couple of design patterns during the project: The first one is the MVP pattern, although as I have mentioned this is inherent in using the android framework. However I was not overly happy with androids implementation of MVP, this is because of the relationship between the view and the presenter. While the XML does define the Views, much of the guts of the view is handled by the android framework itself. This lack of control was frustrating so I decided to modify the given MVP structure. So I decided to make each view an XML plus an android activity. This gave me more or less full control of the structure, the XML defined the view layout and the activity handled the input to the view. There is one other modification in that I made a one to many relationship between the presenter and the views, in this sense it is more like an MVC structure than an MVP.

In order to make it easy to add views to the system without having strong coupling between the views and the presenter, i used the 'strategy pattern' (Gamma et al, 1994), with the presenter being the context class and defining a views interface. Each of the views then implements this interface and is a concrete strategy. I wasnt entirely sure whether to use the strategy or the state pattern, the motivation for the strategy patterns use is fairly clear, but there was an argument to make that each view constitutes the application being in a certain state. However I felt that the behaviour of the application would not dependent enough on the state(view) to warrant the inclusion of a state variable and the increase in complexity.

During the group server servlet work I used a variant of the command pattern[Gamma et al, 1994] to encapsulate individual queries as command objects, Reducing the coupling between the http functionality and the database. This also makes it relatively trivial to add new queries to the system as needs arise. In this case the main servlet functioned as the client class, the invoker was a class called 'QuerySelector' holds a reference to the query to be executed. I then created a DBQuery interface to act as the abstract command interface, and any query that needs to be executed is then encapsulated as a command object inheriting from this interface. The last part of the command pattern is the reciever, and in my case that was the JDBCQueryConnector class, this class was responsible for connecting to the database and actually running the query in question. I felt that by using this pattern i made life both conceptually and actually easier for all involved. The only thing i didnt really think about is that there are members of the group who are not overly comfortable with object oriented techniques, and especially design patterns; and it is difficult to explain any design pattern if you are having trouble with the uses of inheritance and the importance of interfaces. However I spent a long time going over these concepts with the group and even if they may not be able to implement design patterns I believe that they now understand (at least conceptually), the reasons behind the use of the command pattern and how it works in the context of our group project.

What usability concerns did you have with respect to your target user type?

There was one major concern that I had with regard to my target user type, and this was that the user type is not particularly well defined. By this I mean that it 'professionals' are not one homogenous demographic, they are spread over a wide group of ages, and skill sets, from personal assistants to architects, software developers to CEO's and everything in between. I was concerned that in trying to design an application to appeal to everyone who fits into this category, i might create something that appeals to no-one. I think that these concerns were well founded, and could easily have been realised if care was not taken.

In order to limit the chance of this happening I eliminated demographics that would have been relatively minor in the larger scheme. I did this by removing the elderly and those under the age of about 21 from consideration. This made life somewhat easier.

However given that the outcome of the project is a less finished app than I would have liked, it is difficult to judge whether I actually hit the mark or not. Given the results of the usability evaluation that I carried out on the app, it is clear that general consensus points to ease of use and a small learning curve, but there was also a consensus that the app was too basic to really be a usable SNS.

Section 2 - organisation

INTRODUCTION

Did the project idea chosen work well for the group?

Initially we, as a group, were a little stumped with regards to what idea to use, very few of the ideas we came up with allowed much scope for diversity between applications. This caused us a problem for a while but then we collectively came up with the idea for an augmented reality social networking application. After having several brainstorming sessions [REFERENCE] on the topic we determined as a group that there was sufficient depth to the idea that it warranted being used.

After we decided to go with augmented reality social networking, we came up with a number of different spins on the topic for a series of different user groups: business/marketing, romantic, artistic, professional, children, the elderly, and general users:

business/marketing - a focus on business marketing and advertisement

romantic - an app focused on romantic relationships, similar to a dating site.

artistic - allowing artists/musicians to showcase their work.

professional - a professional/collaborative emphasis.

children - a social networking app focused on children.

the elderly - the same as the above but focused on the elderly.

general users - a social networking application for the general populace.

we each chose one of these 'spin's' as the focus our individual app's, this worked out pretty well, in so far as most of us managed to chose a spin that played to our individual strengths and interests: Ian Muigai chose business/marketing, he already owns a ForEx trading and as such he jumped on the chance to showcase his business skills in the app. Mathu chose artistic, he decided that he could draw on his interest in music to try and produce an innovative app. Mark Passey as an avid user of social networks decided his skill base would be best used in producing a more all-purpose app aimed towards the 'average' user. Emmet Wray, in this context, was a bit of an odd one out, he had no particular interest in romantic/dating sites yet he chose to go down the romantic route. for myself, I chose 'professional'. I chose this mostly because I felt it was probably the option most likely to go down well with future employers.

While the idea that we chose was broad and imaginative, it may have been too broad for timely implementation of a good level of functionality. Unfortunately this became clear too late into the project to be able to do anything about it, and from there we just had to make the best of what we had chosen.

How did the team meetings evolve over the coursework(and were they useful)?

Team meetings fall under two headings: Group meetings with the tutor, and group meetings without the tutor. The initial meeting with the tutor turned out to be difficult to organise, this was due to difficulties finding times at which both the tutor and all members of the group where available, as well as a regrettable communication gap between the group and the tutor. The first meetings where attended by a subsection of the group, and the proceedings were disseminated to those members who couldn't attend.

As time progressed, we were able to become more organised with regards to these group meetings and we were able to meet up as a full group more often. The frequency with which we met up with the tutor varied as time progressed. At the start of the project we met up every week, but after a point, we decided (the group and the tutor), to only meet up as and when necessary. This freed up more time for us to concentrate on the work, as less time was spent mired in communication between the group and the tutor.

The tutor meetings themselves were of mixed usefulness, as the majority of the time the focus of the meetings was placements. Although in hindsight we could have pushed harder to drive the meetings more towards the group project. This mixed usefulness was also part of the reason why the tutor meetings became more infrequent.

The internal group meetings however are a different matter; from the start we met up on a regular basis to talk about the project and how it was going. At the start of each task, we would meet up multiple times per week to talk about and work out the specifics of that particular task, and to plan how we would attack the problem. After this initial period, usually about 2 weeks, the meetings would slow down to an irregular schedule. This (roughly) weekly meeting was used to update the group on the progress that individuals had made, and as an opportunity for members of the group to air any problems they may have encountered in that last week so that the group could help with whatever problems individuals were having.

From the very beginning there was a (semi)active decision to not have meetings set on regular days or with any kind of formal format, this reduced the amount of bureaucracy inherent in formal meetings, and allowed meetings to be as long or as short as was necessary given the topics at hand and allowed for us to schedule meetings as and when they were necessary.

On the downside, however, it did mean that we would sometimes go for two weeks without a group meeting and that would sometimes put us out of sync with each other. This would sometimes negate the lack of bureaucracy and cause delays due to catching each other up on our respective progress. However on balance I would say that the system of meetings provided more benefits than detriments; and that the evolution, while unstructured, ended in a system of meetings that worked for our group. I would probably not recommend the system we used for other projects or teams but given the people in our group and the nature of the personalities involved it did work in this case. For other projects having more structure is probably more beneficial than the relatively unstructured approach that we used.

What were the benefits of using the Wiki/web collaboration tools?

Throughout the project we employed web collaboration tools to aid in our project. However we didn't have one particular web collaboration method, there were many different methods. This was a problem on occasion, as we could never reach consensus within the group as to what method to use. Over time we fell into a rhythm and became used to the various methods in play. And while having a unified collaboration method would have made life easier and more consistent, the use of multiple applications had some benefits, namely that the limitations of one implementation are covered by a different one.

The one type of collaborative implementation we didn't use was a wiki. There are a couple of reasons for this: firstly our group didn't have the discipline to use and update a wiki on a regular basis, secondly some of our group members weren't particularly tech savvy and as such, having to use a wiki would have been more troublesome than it would have benefitted. Lastly and most importantly, we didn't actually think to use a wiki.

The first method of collaboration that we used was facebook. We set up a private facebook group purely for the group project, we used it to collaborate on certain problems in conjunction with the file upload facility of facebook. The collaborative functions weren't used as extensively as other methods though, and we ended up mainly using it for communication, and as an electronic substitute for physical face to face meetings whenever we didn't have anything that required face to face communication. The limitation of facebook were that the file sharing facility is not available from the mobile app, which is one of the reasons why other forms of collaboration were used.

The next form of electronic collaboration we used was dropbox and occasionally Google drive. These were used primarily for the exchange and storage of files. They both do a good job in this regard, with both of them having an easy to use interface, mobile apps that allow the use of all common functionality and are free. Dropbox was used more than google drive but the two services were more or less interchangeable for all intents and purposes.

The blackboard learn website also provided a host of useful tools and services which saw mixed use throughout the project. However due to our lack of a strong organising principle these facilities were used in a rather inconsistent fashion. We originally decided that the blog was going to be used for general communication, asking questions of each other etc; but after a couple of group members erroneously used the journal and tasks sections(which were originally designated for finished subtasks and subtask specific communications respectively), replies got posted to those sections and the organisational system broke down. The blackboard facilities where useful to a degree but there are some serious limitations. Chief among these is that while there is a blackboard learn mobile app, it doesn't feature push notifications meaning that it is not possible to get real time activity notifications making it less than ideal for collaborative activities, and as a result was used sporadically and required other forms of communication to be effective. This proved to be the downfall of our use of blackboard learn facilities. It remained in use till the end of the project, but not on a regular basis.

By far the most used collaboration tool was email, the reason for this was its simplicity, near real-time notifications and zero setup time (everyone is already set up with and familiar with the use of email). It is easy to use email for the sharing of files, as well as commenting on them and collaboration. There were very few downsides to using this system, the only noticeable one being that different email services have limits on the size of attachments i.e. if files were over a certain size then email could not be used to share them. This was overcome via the use of other mediums (Google Drive/Dropbox) in those cases where the file size was too large.

In summary we made use of a variety of electronic collaboration methods, and as much as from the descriptions given, it looks like it was a very haphazard affair. However despite how it looks it worked very well in general, and after figuring out the limitations of the various methods we had no problems in web based collaboration. This is not to say that it couldn't be improved upon, in fact it definitely could be. While the methods of collaboration worked, there were the occasional hiccups with regards to redundancy of collaborative methods and communication. This would not have been a problem if we had used a one or two well defined methods covering all the bases.

How would you organise the team differently if you had to undertake a similar project?

If I were to undertake a similar project i would organise a fair few things differently: firstly the team meetings, while effective enough to get the job done could have been more efficient if there had been a stronger organisational principle in place. As such having a once weekly meeting in place as a regular thing could have been beneficial and also avoided lengthy catching up during some of those two week periods where we went without a meeting, even though the flexibility of our particular system of meetings was useful and for the most part efficient, a little bit of red tape could have reduced some of the inefficiencies inherent to that flexibility.

The next major thing I would do differently if undertaking a similar project is to do with the collaboration methods that we, as a group, used. Again while our system of collaboration tools worked for the most part, we could have reduced inefficiencies inherent to a system without strong organisation principles by using a more centralised approach. In my opinion the best option for a project such as this would have been to use the Google suite of tools plus the use of a wiki. I would put a gmail account for the project in place and set up forwarding so that any email coming in from team members gets forwarded to the rest of the team(akin to a mailing list) this would give us a centralised communication method that everyone is already set up for. Google drive gives us 15GB to play with to store and share files on. The other benefit of google drive is that files can be collaborated on in real time meaning that team members could e-commute to work on the project. Couple this with a centralised group project calendar as well as project management plugins provides a professional, free, easy to use system to work on and manage the group project.

While the google suite of tools would be used as the day to day collaboration on the project, the wiki would be used for project documentation, i.e. diagrams, reports and other project artifacts. This would give us a running commentary on the current state of the project, and upon its completion full documentation on the entire project. This contrasts with our project in so far as we got all the tasks in the project completed and all the deliverables are there, but there is no centralised full documentation on the various project artifacts, making it difficult to pull together the various artifacts to show to the various stakeholders, and to future employers as part of a portfolio.

Section 3 - people

Was your team efficient and effective?

The group worked very well together on the whole. We were in almost daily contact with regards to the various aspects of the different tasks that we undertook, and as discussed in the previous section we met up often for group meetings, and spent a fair bit of time helping each other out whenever a member had problems with a particular task.

Our methods of communication were relatively centralised and efficient. However our collaboration methods were a little bit sprawling and a little uncoordinated. Realistically this is a relatively fair reflection on how our team worked, effective but a little bit uncoordinated. Although in our case it worked, in most cases, and with a more substantial project it probably would not have produced good results. However in our case it did, and really speaking we got our work done in an efficient and effective manor, and while the method is not professional or conventional, I would stand by that assessment.

How were tasks allocated?

At the start of each part of the project, we would sit down and work out how we would proceed vis-a-vis the tasks necessary to complete. A major part of this meeting was deciding how the work would be allocated. On the first meeting that the group held, we went around the group and talked about our strengths and weaknesses regarding various skills: organisation, programming, technical design, business sense, databases, project management etc.

At first we assigned tasks based on these strengths and weaknesses as well as how much people wanted to do it: assigning organisational tasks to someone who said that was their strength and technical design tasks to someone who's said that was their strength. After the end of the first part of the group project we, as a group, had more of a concrete idea of the various group members strengths and weaknesses and assigned tasks, as much as possible, on the basis of a meritocracy.

We did this as we believe that this would produce the best result on all the coursework tasks. Of course there were tasks that had to be carried out by all individuals. In these tasks those with more experience and skill in the particular areas, helped those who were less experienced or skilled. It did take some extra time for the individuals who were strong in the specific area, but we had a good spread of people strong in different areas, we figured, what goes around comes around.

Did you have specific roles?

There were no formal roles defined within the group, however as the project progressed people fell into particular roles. This was driven by both the group and individuals strenths and weaknesses. There was no defined leader, as it was more of a democratic process, but certain pseudo-leadership roles were established in a non-formal context. The first was Ian Muigai, who assumed a role akin to a CCO, taking over communications between the tutor and the group, as well as taking an active role in the internal organisation and communication of the group, this was mostly on the request of the group because of his previous business experience.

The other pseudo-leadership role that emerged was something akin to technical director. This role ended up being assumed by myself, this was not as active a decision as that of Ian, but after the first couple of meetings it became relatively clear that I had more confidence and experience with the technical side of things: programming, object oriented design and database design. And many members of the group ended up looking to me for technical advice.

I believe that the lack of specific roles was(in our case) a good thing; the group being relatively small would probably have been hindered by having specific roles, it could hinder progress by causing arguments based on the common 'you were'nt doing your job' scenario, or people feeling like others arn't pulling their weight based on job description.

On the flip side, being able to fix responsibilities to individuals does make accountability easy. However in a project such as this accountability isn't hugely important and only adds to possible problems between teammates.

What communication mechanisms were used?

During the project we used the Facebook, Email, SMS and phone calls as our methods of communication. As mentioned in the previous section we set up a private facebook group for use within the group. This was our primary form of communication, any questions that team members had or updates on tasks were posted on the facebook group. This had the benefits that there is a mobile app allowing almost 24-hour semi-real time communication. However there are certain tasks that wern't possible to communicate effectively on; things like in-depth task specifics involving project artifacts, or items where it was necessary to exchange files. For tasks such as this, it was necessary to use email. Email turned out to be our secondary form of communication, we didnt use it as often, but for some tasks it proved more effective than facebook.

SMS and telephone communication was also used as methods of communication, although these methods were used mostly for organising meetings, and communications that were time sensitive or high priority. This form of communication was used minimally in comparison to email and facebook.

Our methods of communication wern't as spread out as our electronic collaboration methods, and this probably worked in our favour. It was easy to get hold of group members when it was needed, and the facebook group gave us a centralised method of general communication, and the limitations of that form of communication were eliminated by using the other forms of communication mentioned previously.

Section 4 - research topic

For my research topic i have chosen to take a look a human centered design, I have split this into two sections: requirements gathering and a usability testing. First taking a look at the literature available and relevant to this project, a presentation of the results of some of my experimentation into the area is included in the appendix. The reason for splitting this topic into requirements gathering and usability testing, is that I did some experimentation at the requirements gathering phase which shaped many of the thoughts I had with regards to the design of the application. Usability testing was carried out afterwards in an effort to discover whether the design was effective and what might need to be changed/implemented in order to make the app more suitable to my user group.

### **Requirements Gathering**

(M.F. DiAngelo & C.J. Petrun, 1995) work through a very detailed and well specified methdology known as CRTS (Custormer Requirements and Task Specification), which focuses on gaining requirements based solely on customer input, through an extensive series of expert-led sessions. Unfortunately utilizing this in its full capacity would have taken a restrictive amount of time. I would like to take a key point from this and apply it to my own requirements gathering methods, namely: that user requirements should come as much as possible from the customer (with the caveat that, there exist some pre-existing requirements for this project)..

At the outset of this project some arbitrary requirements were prescribed for the project (David Bell, 2014) Which include the use of augmented reality, and the requirement that the project must be completed using Google's Java Android framework. The justification for the use of AR in this particular context is given in the background section. This makes the project a little limited as people may not want an augmented reality SNS app.

The first method that I elected to use for the purpose of requirements gathering is a set of semi structured interviews performed on a test population of potential users. (Doody & Noonan, 2013) building on the work of (Holloway & Wheeler, 2010), describe the effective use of semi-structured interviews to allow a freeflow of ideas and context, difficult to obtain via structured interviews, to be developed; while allowing the researcher to eliminate some of the observational bias inherent

in the unstructured interviews. The question themes used to direct the course of the interview, within the context of this project where as follows: What role do you think an SNS could play in your day to day professional activities, how comfortable are you using GPS/Sensors, What features do you use most/would you most like.

In conjunction with the method specified above it was decided upon to use brainstorming. Both (M.F. DiAngelo & C.J. Petrun, 1995) and (AL-Hothali, 2012) describe it's use for the elicitation of requirements in software developement. In the context of ARC, this was performed by a group of 9 Potential users and the system developer using the group passing technique described in (Osborne,1963)

Lastly a survey was released onto Facebook and Twitter, asking a series of descriptive, true/false, selection and Likert style questions. These questions attempt to ascertain(amongst other things) potential users' most favoured/wanted features from an SNS and users current SNS usage (Hanson,2014).

### **Usability Testing**

As recommended by (Speicher, 2015), the actual usability is evaluated acording to three metrics (internal, external and in-use).

###### *Prototype Design*

A series of prototypes was used to aid both the design process and the usability of this project. This process was started by using the POP application to produce an interactive paper-style prototype as advised by (Wilson Fletcher, 2014). Following this a HTML mockup was created to more closely simulate the look and feel of the end product (Van Buskirk & Moroney, 2003).

The look and feel of these prototypes was deliberately left rough while engaged in usability testing as (Van Buskirk & Moroney, 2003) noted that by doing so it is possible for the test user to concentrate more on the functional usability, as they are less likely to forgive errors because of a polished appearance.

###### *Internal*

This is the metric that is hardest to quantify, and from the consumers point of view, the least important. However, if this project were to continue in the long run, maintainable, easy to read code ultimately aids usability as new features are less likely to be prone to crippling bugs, and those bugs that do crop up should (by defnition) be easier tofind and fix.

Because of this, I have decided to spend some time evaluating these attributes of ARC by giving the source code to 5 developers not involved with this project and asking them to implement some test functionality. A time limit of one week was given to them, after which their code was submitted and the developers interviewed.

The developers were of varying skill levels and asked to implement a new text based view with for- ward and backward buttons to view the next/previous post based on chronological order. The only information available to the developers was the sourcecode documentation and a brief description amounting to what is contained in the rst paragraph of this report. The code that they submitted was not used in the final product.

###### *External*

This metric defined the testing of the initial POP prototypes and the HTML Mockup. Test participants were asked to review these prototypes in the context of completing a series of test tasks aimed at assessing the 4 aspects of usability proposed by (Shackel, B., 1991). These test tasks were performed under observation in laboratory conditions and the time taken to complete the tasks was recorded. The same tests(where applicable) where performed on facebook and linkedin. No information was given to the test subjects other than what is contained in the first paragraph of this report. Semi-Structured interviews were then carried out given the theme of (Shackel, B.,1991)'s 4 criterion:effectiveness, learnability, flexibility and attitude.

###### *In-Use*

This context denes the usability in a real word scenario, therefore the app was given to test participants, and they were asked to use it off site as if it were any other mobile application. A time limit of two days was set after which they were given a questionnaire based on a modified set of questions from (Thuseethan et al, 2014) to ascertain the usability of the product in its current state.

Lastly unstructured interviews were conducted (Doody & Noonan, 2013) where participants were asked for there overall thoughts on the product. Test participant comments were used to drive the interview in an attempt to reduce researcher bias.

# **References & bibliography**

[1]Maximilian Speicher. (2015). What is Usability? A Characterization based on ISO 9241-11 and ISO/IEC 25010. arXiv:1502.06792. 1 (1), p1-10.

[2]M. F. DiAngelo., C. J. Petrun.. (1995). Collecting Product based Usability Requirements. IBM Systems Journal. 34 (1), p4-19.

[3]David Bell. 2014. CS2001 - Group Project. [ONLINE] Available at: https://blackboard.brunel.ac.uk/webapps/portal/frameset.jsp?tab\_tab\_group\_id=\_2\_1&url=%2Fwebapps%2Fblack- board%2Fexecute%2Flauncher%3Ftype%3DCourse%26id%3D\_17217\_1%26url%3D. [Accessed 16 March 15].

[4]Doody 0, Noonan M (2013) Preparing and conducting interviews to collect data. Nurse Researcher. 20, 5, 28-32.

[5]Holloway I, Wheeler S (2010) Qualitative Research in Nursing and Healthcare.Third edition. Wiley-Blackwell. Oxford

[6] Samaher Abdullah AL-Hothali, Noor, Anusuyah Subbarao Abdulrahman AL-Zubaidi, . (2012). Requirements Elicitation For Software Projects. International Journal of Computer Science and Information Security. 10 (11), p64-71.

[7] Osborn, A.F. (1963) Applied imagination: Principles and procedures of creative problem solving (Third Revised Edition). New York, NY: Charles Scribner's Sons.

[8]Wilson Fletcher, (2014), 'Prototyping - The Tools: What We Use and When To Use Them', CS2003: Usability Engineering, Available at: https://blackboard.brunel.ac.uk/webapps/portal/frameset.jsp?tab\_tab\_group\_id=\_2\_1&url=%2Fwebapps%2Fblack- board%2Fexecute%2Flauncher%3Ftype%3DCourse%26id%3D\_17219\_1%26url%3D, [Accessed 16 March 15]

[9]Shackel, B. (1991). Usabilitycontext, Framework, Definition, Design and Evaluation. Shackel,B. and Richardson, S., Ed. Human Factors for Informatics Usability. pp.21-37. Cambridge, UK, Cambridge University Press.

[10]Thuseethan, S., Achchuthan, S., Kuhanesan, S.. (2014). Usability Evaluation of Learning Man- agement Systems in Sri Lankan Universities. Global Journal of Computer Science and Technology. 15 (1), p15-24.

[11]Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994)Strategy. Design Patterns, Elements of Reusable Object-Oriented Software. Westford, MA: Addison Weseley.

# **Appendix**







