mPOW – Mobile Support for Health Care Students on Work Placements

Eric Bonnici[1] and Samia Oussena[2]
[1]*ARC Technology Ltd, Byfleet, Surrey, U.K.*
[2]*University of West London, Department of Computing, London, U.K.*

Keywords: Mobile App, Mobile Learning, Higher Education, Work Placements.

Abstract: This paper outlines the development of a mobile application called "mPOW" as part of a KT (Knowledge Transfer) partnership project between the University of West London and Arc Technology Ltd. The aim of this project is to provide improved support for HE (Higher Education) students while on a work placement. The design considerations and solutions for designing a cross-platform application are discussed and presented, and the result is an app that is applicable to students from various universities who attend a range of distinct work placements. The application may also be used online as well as offline with the result that students are not tied down to space and time.

1 INTRODUCTION

Practice based learning, or work based learning (WBL), is "that learning which is explicitly designed to relate to professional practice standards. It includes learning which is work-based, undertaken in placements and which aims to enhance learners' employability" (Quality Assurance Agency for Higher Education, 2011).

Practice based learning has become a crucial part of the curriculum for a selection of courses offered at HE (Higher Education) institutes. The popularity of WBL has also increased due to the increase in knowledge led demands of the UK industry illustrated by the Leitch Report (Leitch, 2006). This report challenged institutions to deliver learning opportunities so that 40% of adults of working ages have a higher education qualification.

The courses in HE which require students to complete both theoretical as well as practical classes result in student timetables which are somewhat dynamic in nature. Further to this, work placements in the UK are in high demand, many placements may only accommodate a few students at a time and, consequently, placement allocations are only issued a few weeks prior to the start of the placement – with students being notified some days after that. This exacerbates the urgency with which placement allocation information needs to be distributed to the students.

Students who are about to attend a placement need to be well informed on the location (including route guidance and public transport), contacts, and times of their placement – which may also include shift patterns if applicable. Other forms of vital information may include dress code as well as any procedures which need to be followed (such as sickness procedures). Less urgent, but still important, information may come in the form of the following: parking space availability, changing facilities, locker availability, any practice preparation, possible learning opportunities, and many more. Once a student has been allocated to a placement, it is vital to get all this information out to them before they commence their practical training. What if a student's placement is changed at the last minute? How can this information get sent to the student in a timely and efficient manner?

2 STUDENT APP REQUIREMENTS

For various reasons it is not generally possible for universities to supply their students with mobile devices and, therefore, any app targeted at students will need to run on their personal devices – of which

Bonnici E. and Oussena S..
mPOW – Mobile Support for Health Care Students on Work Placements.
DOI: 10.5220/0005446300810088
In *Proceedings of the 11th International Conference on Web Information Systems and Technologies* (WEBIST-2015), pages 81-88
ISBN: 978-989-758-106-9

there is a vast range. Developing an app which runs on a range of devices brings with it some important challenges and design considerations, this is especially true for companies (or development teams) with limited resources. We will discuss these challenges in Section 3.

As mentioned earlier, distinct universities publish different placement information and therefore an app offering placement information to students from a range of universities must cater for this. Section 4 discusses the challenges presented when developing such an app.

Finally, students on work placement are currently making use of a paper based system to register attendance. Once a students' paper based timesheet is submitted to the university, then university staff must enter the information into their electronic systems manually. mPOW may offer an automated solution to registering student attendance while still on placement. Unfortunately mPOW is required to work in an online as well as an offline mode and therefore any location based services cannot be relied on. Section 5 discusses a solution to attendance registration without the use of location services which is as secure as the current paper based system.

3 TARGETING A RANGE OF DEVICES

In recent years, there has been exponential growth in global sales of mobile devices. The development of these various mobile devices has forced vendors to continuously improve their mobile platforms by launching new features. The constant launch of new platforms has caused a major problem for mobile application developers; i.e. a native application has to be developed separately for each platform. Consequently this has hampered the development in terms of cost and time and introduces complexities with code management. Currently, the most widely used platforms are iOS (Apple) and Android (Google), with Windows Phone (Microsoft) trailing behind. Figure 1 shows the share of the market in the second quarter of 2014.

According to IDC the total combined market share of iOS and Android devices swelled to 96.4 percent throughout the second quarter of 2014. This is up from 92.6 percent the year before. However, IDC claim that this is due to the increase in actual mobile devices worldwide - which has increased past 300 million phones for the first time in history. iOS devices tend to dominate the high end of the market while Android devices dominate the low end. At the moment this leaves little room for Windows phones, although Microsoft has recently lowered the platform's licensing costs to help propel it in the market. In either case, targeting these most widely used platforms comes with some challenges.

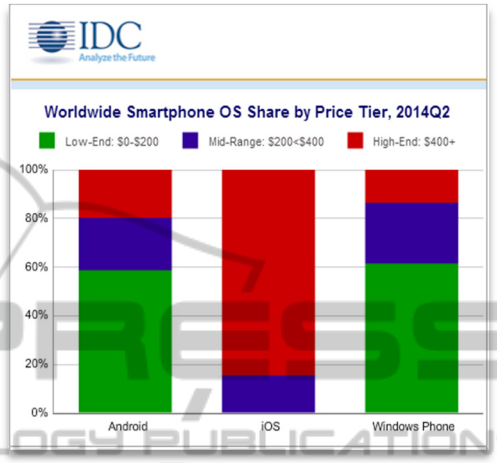

Figure 1: The smartphone market. ((IDC), 2014).

3.1 Developing Native or Cross-Platform

A prototype of mPOW was first developed as a native Android application due to the fact that Android are the most popular on low cost devices. This prototype served as a proof of concept allowing quick feedback from potential clients but also served as an investigation into the time and resources required to develop for a single platform.

3.1.1 Native Apps

Native apps are only specific to mobile platforms for which they have been developed for and do not run on other platforms (Nithiyanantham.C & Kirubakaran.R, 2013). Native apps are developed separately for each platform with different languages, for example Java for Android, Objective-C for iOS, C++ for BlackBerry, C# or VB.Net for Windows Phone, and so forth (Smutny, 2012). Therefore, the application code used for one platform is not reusable for another platform. The required tools for developing and testing native apps are done using an IDE (Xinogalos, 2013), for instance Eclipse for Android apps, Xcode for iOS apps, MS Visual Studio for Windows Phone apps and so on.

Although, by using a native approach, there is an

additional overhead in creating an app for several platforms, the quality of native apps is usually very high. They are fast, reliable and powerful (Smutny, 2012). Native approach allows apps to use all the features or components of the device as programs written for the device's native runtime environment have rich APIs (Smutny, 2012). Native apps provide higher level of 'user experience' (Xinogalos, 2013). However, the development of several mobile platforms has created fragmentation of mobile platforms with differences in their architecture, supported programming languages, SDKs, frameworks or APIs, feature support and other technologies.

Most software companies have some common functions gathered together in a core library. If each mobile platform has a distinct client application, then there is a probability that either the common code will be duplicated per platform, or, if shared libraries are used, each client (depending on the platform SDKs and features) may use different parts of the shared code base. This may have serious ramifications later on when bug fixes and updates to the mobile application is required - and will have to be carried out on a per platform basis.

The cross-platform mobile application development approach has emerged as a solution for this problem. Over the past years, many cross-platform mobile application development tools have been introduced so as to assist developers in quick and easy development of mobile applications that supports multiple platforms. Each of these tools has certain benefits and limitations as compared to others. Several techniques are being followed for developing cross-platform mobile apps. These will be discussed in the following sections.

3.1.2 Hybrid Apps

The hybrid approach is the combination of native and web approaches, where apps are developed using web technologies and are executed inside a thin native container of the device (Raj & Tolety, 2012). The same source code is used to build various platform installation files. Hybrid approach takes advantage of browser engine and the device capabilities to build more mobile friendly apps (Raj & Tolety, 2012). Hybrid apps are distributed through app stores and these apps, as pointed out by (Raj & Tolety, 2012), are downloaded and installed into the mobile devices just like native apps. They can perform as standalone applications or as server based applications (Raj & Tolety, 2012). A couple of popular examples of developing mobile apps with hybrid approach are Phonegap/Cordova and Sencha. By taking an approach such as Phonegap, the developer is allowed to access certain device features which would not be accessible from a standard web page, an example of this is access to local storage on the device. The developer is not solely limited to the features already provided since there is a whole range of 3rd party plugins available that add features (such as the Google AdMob plugin for phonegap) which allows control of any advertising banners.

3.1.3 Web based Apps

The web approach allows development of cross-platform mobile apps which can be viewed using the web browser of mobile devices. Web apps are similar to traditional web pages used in computers but are specifically designed for mobile devices or in other words they are web pages that are optimized for viewing on mobile devices. These apps are developed using HTML, CSS, JavaScript and other web programming languages. The web apps do not require any installation onto the mobile device. Thus, it can be accessed on any mobile device that has a web browser and an internet connection. The data and application are hosted from server, so there is no need for updating the web apps (Raj & Tolety, 2012). However, although recent advances in HTML5 have improved access to services such as storage (via web storage) and location (via the geolocation API), there is still limited access to the device's hardware and other native features (Xinogalos, 2013). This confines the services that can be offered to the users.

3.1.4 Mapping to Native

A similar method for developing mobile applications utilises a tool that creates a native app from an SDK other than what is used for the native platform. Examples are:

a. Xamarin Studio
b. Appcelerator Titanium
c. Embarcardero FireMonkey

These tools use their own GUI components with the result that the app has the same look and feel on either platform, but the result is that it does not have the native look and feel on either platform. Some of these solutions use native components wherever possible and then use emulated components for those which are not available on a particular platform. This may result in a user experience where things work mostly like the native controls, but with

just enough minor differences to throw users off (Fowler, 2011).

3.2 Approach Taken

mPOW was initially developed as a native Android app and this was then showcased to a group of 50 potential users. Feedback was gathered from users which included their thoughts on:

- interface design and layout,
- speed and fluidity of the user interface,
- features available to users,
- customisable section of app,
- methods to register attendance of users on placement,
- device, and platform, compatibility,
- any features they would like to see included.

From the feedback gathered we took the decision to carry on with a pure web based approach (i.e. develop the app as a mobile web page), since HTML5, CSS3 and JavaScript would allow us to develop the features required. The benefit of this comes in the form of a quick turnaround from when changes are made, to displaying and gathering feedback from users. This can all occur without any need for the test users to install anything to their devices. Collaboration with a group of test users is made much easier and, by incorporating a large number of test users in the development process, constant feedback can be gathered throughout the development life cycle of the product.

Upon completion of the product, the web based version may then be wrapped up as a Phonegap app to allow installation from the app stores and take advantage of the distribution system provided by these stores, if desired.

Taking a web based approach also allowed us to work on one single code base and, in the future, will allow for bug fixes and updates to be carried out for all devices simultaneously.

There are some disadvantages to this approach which mainly come in the form of decreased performance and lack of the platforms' native look and feel. A point to take into consideration is that it is not always possible to take a pure web based approach during the development cycle – this is mainly dictated by the features being implemented by the app being developed. From our prototype and initial user feedback we were able to determine that the main features will be able to run from a web based app.

3.3 MPOW Showcase Results

mPOW is required to provide its users with a quick and intuitive way to gather information regarding their work placements throughout their academic career. Not only is it important for students to know what placements are coming up, but it is also as important for previous placement information to be available as needed. The reasons for this are as follows: students are required to perform a certain number of hours every year of their course, produce reports on placements attended, and also keep track of specific experiences gained. All this information will be useful to the students as they work on assignments given in class and prepare for their exams.

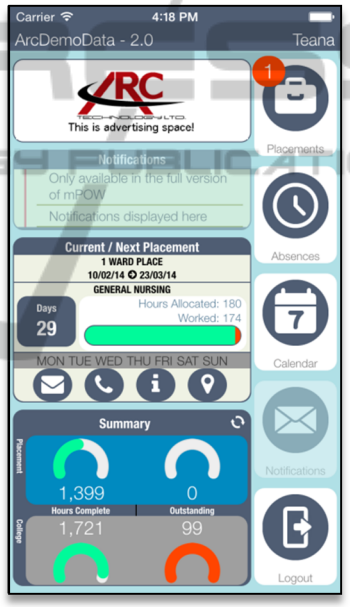

Figure 2: mPOW homescreen.

The main page that the students are presented with after logging in to mPOW is made up of information tiles (on the left) and navigation buttons (on the right). The tiles display the information that was considered the most important by our feedback group. Tiles may be configured according to screen size. If there is more space available then additional tiles can be displayed. Figure 2 shows the homepage on an iPhone 5 sized screen. Figure 3, on the other hand, shows the homescreen on an iPad sized screen in landscape mode. As can be seen the calendar and personal tutor information are also visible.

The "Current/Next Placement" tile was indicated to be the most important information to provide. Students can see their current placement, or their

upcoming placement if the current one is finished. The length of the placement in days, and an indicator on the number of hours worked are shown. Links are also provided to enable the user to contact the placement, find the placements locations, or view further information regarding the placement, if required. Other tiles provide the student with a summary of their yearly attendance, and notifications which can be sent from the university administration to groups of students on particular placements. The first button from the set of navigation buttons takes the user to the list of placements. This button also indicates if there have been any changes since the last time the user opened the app via a notification badge. This can be seen in Figure 2.

Figure 3: The homescreen on an iPad.

Figure 4 shows the list of placements in chronological order with the oldest at the bottom. A student may expand any of the placements to view additional details. This is also visible in the "Current Placement" tile on the homescreen. Different types of placements have different background colours. In the example blue corresponds to a work placement. A black background corresponds to "Made-Up Time". This is used when a student misses some hours on placement (due to sickness for example) and makes it up at another time. A yellow background corresponds to leave, and grey to theory (i.e. the student needs to be at university). A purple background corresponds to a class which is run during a work placement. These colour codes also apply to the calendar as can be seen in Figure 3.

The next section will explain what further information can be viewed on each placement, and how different universities can display different types of information regarding placements.

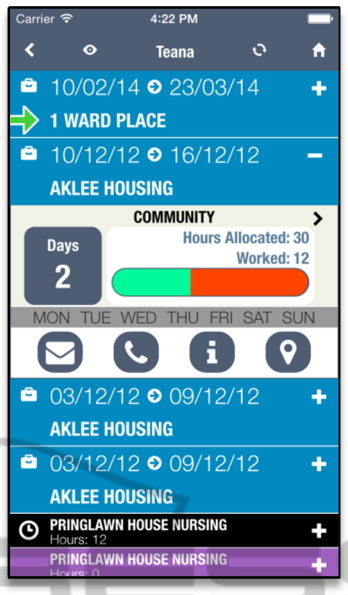

Figure 4: The list of placements.

4 CATERING FOR DIFFERENT ORGANISATIONS

Students using mPOW may come from a variety of universities where each university has their own way of storing and publishing placement information. Differences range from minor changes in layout to completely different type of content. One university may decide that parking information is a vital piece of information for the students while another university might, instead, choose to focus on public transport to and from the placement. Yet another university may decide to publish both parking and public transport information. In this case, such a decision may be influenced by the geographic location of the university and the placement and is a typical example on the importance for the mobile application to cater for the different customers (i.e. universities).

4.1 Customer Specific Backend

A decision had to be made on whether to develop several client applications, one for each client – where each one is customised to that particular customer's needs – or whether to develop one client application that will be used by all students from all universities. There are significant challenges in doing the latter where all customisation for a particular customer will need to occur on the server side. This also needs to be accomplished without

transferring large amounts of data – applications running on mobile devices need to use both power and data sparingly (GSM Association, 2012).

However, by taking the option to develop a separate client for each customer, one would have to setup a distinct project for each customer. Problems may arise from the management of several code bases sharing different parts of some common core. Both version control and bug fixes may quickly become complex tasks, especially as the different applications evolve through updates and routine maintenance. These are the same problems faced with native development of a mobile application on several platforms - however in this case there could be tens or even hundreds of customers. The decision to distribute the application through the major app marketplaces is also a deciding factor on whether to have a specific client application for each customer or a general one for all. When an application is submitted to such a marketplace there is a review period. This provides some control as to what applications are distributed through the marketplace but can also be a prolonged and tedious task. Even after an app is reviewed, many times there are changes required before the application must be sent for review again. Furthermore, even once the application has been reviewed successfully and accepted, any additional development (such as updates or bug fixes) requires the whole process to start all over again.

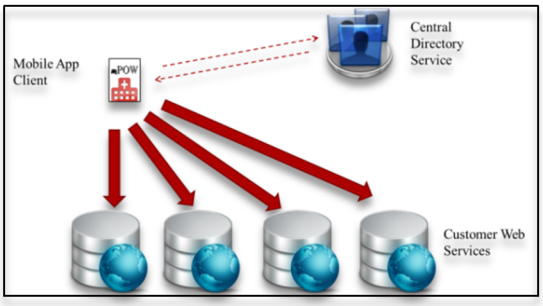

Figure 5: The system architecture.

By providing a flexible backend to the system, all the client code is contained within a single code base which allows for all bug fixes and updates to be handled in one place. Any differences between customer configuration is handled on the server side which does not require any sort of review process - changes are therefore instant.

mPOW was designed to make use of a central web service (common to all customers) which redirects the client mobile application to the customised version of the web service for a particular university. As an example, when a student from UWL launches the app, they are presented with a login screen. Upon entering their login details, the app makes a connection to the central web service which responds with a URL for the UWL specific web service - after which no further communication between the mobile app and the central web service is needed. The use of a central web service as a directory service makes it possible to add (or remove) customers from the system without any need for modification to the client mobile application. mPOW then retrieves all the student and placement information from a web service hosted on the customer's server.

4.2 Custom Placement Information

When a user selects a placement from the placement list, they are taken to a selection of pages containing further information regarding that particular placement. The user may navigate from one page to the next by simply swiping the screen left or right. The idea here is that each page encapsulates a selection of similar types of information (Figure 6).

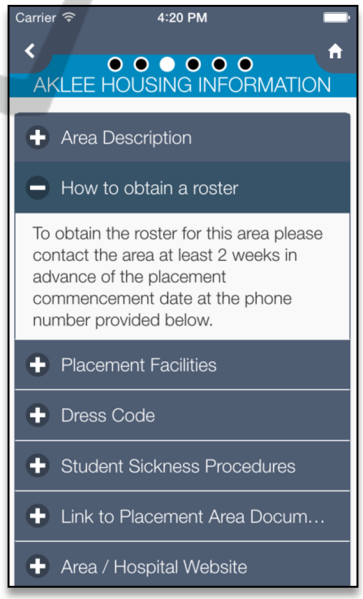

Figure 6: Placement information.

An example of this is a page for all contact information or a page for location information which includes information such as geographical location, links to navigation apps, and also information such as public transport to the placement (Figure 7). These pages are, however, fully customisable by the customer organisation. This means that the

placements department at a particular university are free to include the specific information they would like to show to students, how many pages to include, and even the layout of items on each page. These pages may also be changed at any time and the users will see the latest configuration whenever they browse their placements.

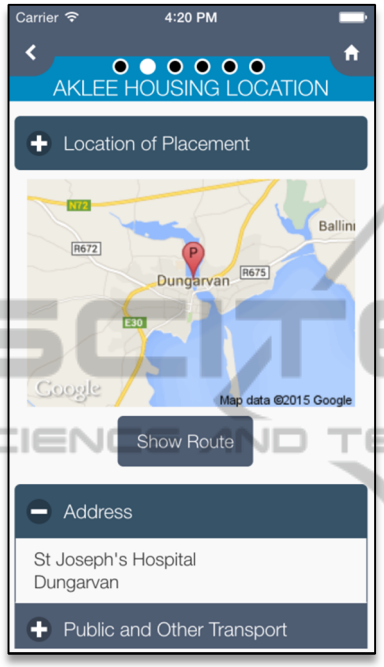

Figure 7: Location of placement.

5 STUDENT ATTENDANCE WHILE ON PLACEMENT

As mentioned in Section 3, a group of 50 potential users were shown the prototype of the product and contributed their ideas to the features available in the final release version of mPOW. One of the main features discussed was attendance registration.

Several approaches to attendance registration were initially considered. The first option was (what seemed like the most obvious to the authors) location based attendance. The idea was that the location of each placement will be saved, and as long as a student is within a certain radius from that location, then he/she will be able to register their attendance.

Upon speaking with our group of test users we quickly came to the realisation that many of the placements take place in fairly old buildings with very thick walls. This ruled out the use of location based services. The use of QR codes was also considered (possibly dynamically generated), but was quickly opposed by our test users as there are strict regulations to the use of cameras inside most hospitals.

The general reaction from this group was that human interaction would be preferred and that several employees are resistant to change (which ruled out a second app running on the mentors device). Therefore we developed a verification system which works as follows:

- The student creates a timesheet on the mobile app for that particular placement.
- The student fills in timesheet on days applicable to the placement.
- At the end of the week the student reports to the mentor and gets that week signed off on the mobile device.
- At the end of the placement the student closes the timesheet (which is then locked for any modification). The student can then send the timesheet to the mentor or to themselves for printing.

If sent to the mentor (via email), then the mentor will be presented with the signed timesheet. After verification, the mentor may click on a link (present in the email) which verifies the timesheet for that placement. The data is then sent to the university's electronic placement system.

In the case that electronic verification is not possible (such as mentor has no email), the student may choose to send the timesheet to themselves. This will allow them to print it, and get the hard copy signed. In this case, the timesheet is then submitted to the university in the same manner that it is currently submitted.

6 FUTURE SCOPE

mPOW is currently in a trial phase by some students at the University of Cork. The main reason for this is to get feedback on how it runs on several devices, by several users. This pilot phase will be followed by the official release of the system to a few customers. The features available in the first release are placement information and attendance information. This will be shortly followed by attendance registration as described earlier in this paper.

Another feature planned for mPOW is evaluations. This feature will enable students to submit evaluations to the placement management team via their mobile device. Currently this is done via a web application and has to be completed from

beginning to end. Further to this, students are required to fill out evaluations of previous placements to see the details of their future placements. mPOW will allow evaluations to be filled out while in an offline state (while student is on the underground for example) and submitted when the student is back online.

7 CONCLUSIONS

As mentioned in section 4, mPOW is flexible to the various dissimilarities in the information provided by each university - this is achieved via server side configuration. Information such as the student's list of placements, along with basic placement details (such as time and location), are required by all students from all universities, however once a student navigates to the comprehensive details section for that particular placement, then the application will display the specific views generated by the mPOW web service, and unique to that particular customer. This is accomplished via use of the Razor view engine which allows the creation of HTML page templates for the content layout, and then populating the content for the specific student upon request. What is returned to the mPOW mobile client (from the service call), is HTML which is rendered and displayed inside the mobile application container (ie. header, footer, pageindicator, etc).

Mobile devices are today prevalent amongst students and offer a wide range of possibilities to support, and improve, education. mPOW is one such example of this. The presence of several mobile platforms do, however, raise some issues when developing a mobile application that will require to run on the vast range of mobile devices available. Native development may produce slicker looking applications with access to native features, however they also require completely independent development per platform. This may be quite costly in term of time and resources and may complicate the release of updates and bug fixes later on. mPOW has been designed using a hybrid web application approach allowed for a shared code base on several platforms, and access to native features using phonegap. By using mPOW, students will be able to access placement information whenever required and will be able to register their attendance while on placement. This eliminates the need for a paper based placement attendance system.

REFERENCES

(IDC), I. D. C., 2014. s.l.: s.n.

Fowler, M., 2011. *Martin Fowler.* [Online] Available at: http://martinfowler.com/bliki/CrossPlatformMobile.html.

GSM Association, 2012. *Smarter Apps for Smarter Phones!,* s.l.: GSM Association.

Leitch, S., 2006. *Prosperity for all in the global economy - world class skills: executive summary and foreword,* s.l.: s.n.

Nithiyanantham.C & Kirubakaran.R, 2013. Cross Platform Application Development with Compatible GUI Solutions. *International Journal of Engineering Science and Technology (IJEST),* 5(7), pp. 1427 - 1433.

Quality Assurance Agency for Higher Education, 2011. *Key aspects of practice-based learning in teaching nursing and social work in Scotland,* s.l.: SMCI Associates.

Raj, R. & Tolety, S., 2012. *A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach.* s.l., s.n.

Smutny, P., 2012. *Mobile development tools and cross-platform solutions.* s.l., s.n., pp. 653 - 656.

Xinogalos, S. X. a. S., 2013. *A comparative analysis of cross-platform development approaches for mobile applications.* New York, s.n.