

iWhisky: A social whisky application for iPhone

Report to the Examiners

By Joshua Lewis Marketis

Student Number: 07823631

Exit Award: BSc (Hons) Internet Computing



Table of Contents

Introduction	3
<i>The Problem</i>	3
<i>A mobile world</i>	3
<i>A social world</i>	4
<i>The Solution</i>	4
<i>Background research</i>	4
<i>Aims and Objectives</i>	6
<i>Development Methodology</i>	7
<i>Project Planning</i>	7
Development Process	9
<i>Learning Object-C and iPhone Application Development</i>	9
<i>Designing an iPhone application</i>	9
<i>Building an iPhone application</i>	11
<i>Testing an iPhone application</i>	12
<i>Legal and ethical issues</i>	12
Reflection	14
References	15

Introduction

This document describes and evaluates the development and learning experience of Joshua Marketis in producing the product “iWhisky” for his final year project.

The Problem

The product was developed for a real life client who runs an online whisky retail website in Royal Tunbridge Wells. The client was introduced during the optional placement year in which valuable learning experiences and practical knowledge was gained. The client’s e-commerce site (www.masterofmalt.com) sells whisky, rum, brandy, champagne and wines and is a growing business, however as a small player in a populated market the client is seeking to differentiate themselves and acquire brand loyalty.

The client hopes to achieve this goal by using utilising two growing trends in today’s technology; mobile and social.

A mobile world

The mobile phone landscape is dominated to this day by ‘feature phones’ that account for 79% of the industry (Nielsen, 2010). These devices, produced by companies such as Nokia and Motorola, provided basic features such as making phone calls and sending text messages and are incredibly inexpensive. The growing trend in developed markets such as the United States and Europe however, is towards more advanced devices called ‘smart phones’. These devices surpass the typical limitations of a phone bringing it closer to the capabilities of a personal computer, with email, programs and multimedia functionality. This explosion in capabilities on mobile devices has attracted developers who can bring exciting new applications and experiences to users on the go.

The smart phone market is occupied by two major players in 2011; Apple and Google. In 2007 Apple launched its iPhone, running an operating system called iOS, which has gained 23% of the UK market, while Google’s Android operating system, released in 2009, has acquired 38% (Arthur, 2011). The two companies have radically different businesses strategies in this market, with Apple’s iOS running on a single device created by Apple, while Google allows Android to run on any device a company (such as HTC, Samsung and LG) would like to create. Apple’s defense of this policy is that they can create a better, seamless experience for the user this way, while Google argues that open creates more innovation.

Other players attempting to get into this market are Microsoft with its Windows Phone 7 operating system and Hewlett-Packard (who purchased Palm) with webOS. Research In Motion (RIM) and its BlackBerry is a long standing holdout in the industry, although its losing market share fast (Nielsen, 2010).

While Android’s dominance seems assured thanks to the number of handsets the operating system is installed on, many forget that Apple also sells a WiFi only version of the iPhone called the iPod Touch and a larger 10 inch tablet called the iPad, both of which also run iOS and don’t require a phone contract. According to recent number tracking individual users (not devices), iOS (including iPod Touch and iPad) reaches 16.2% of the 234 million mobile users in the US while Android only reaches 10.2% (comScore, 2011). Numbers from NetMarketShare seem to agree with this data, showing 1.87% of Internet browser usage coming from iOS devices while only 0.56% comes from Android devices (NetMarketShare, 2011).

A social world

While mobile devices are a growing trend in people's lives, an even bigger trend is the social web. Facebook, the social networking site where users communicate with their friends about their daily lives, share pictures, play games and 'poke' each other has more than 500 million active users who spend 700 billion minutes per month on the site, with more than 50% of users accessing it from a mobile device (Facebook, 2011).

Twitter is another incredibly popular social network, in which people send and read short messages (up to 140 characters) being sent in real time. Twitter has over 200 million users, generating 65 million posts a day and 800,000 search queries (Shiels, 2011). Some have come to refer to it as the 'SMS of the internet' (D'Monte, 2009).

Social networking has rocketed into popularity as a way to meet new people, find old friends, communicate effortlessly and find people who have similar interests.

The Solution

At the start the project just under 7% of visitors to the client's website were using mobile devices. By the end of the project that number has increased to just over 10% of visitors (Google Analytics, 2011), validating the original desire to get into the mobile space. These 10% of visitors are getting a sub-optimum experience, as the client's current site uses small text and buttons that are difficult to use on a small mobile screen. Visitors to the website are also becoming increasingly active in trying to discuss whiskies in user reviews and with the client's Twitter and Facebook account. With the success of whisky social site Connosr (www.connosr.com), which allows users to build a collection, write reviews and discuss whisky, the client was also eager to make the site more social.

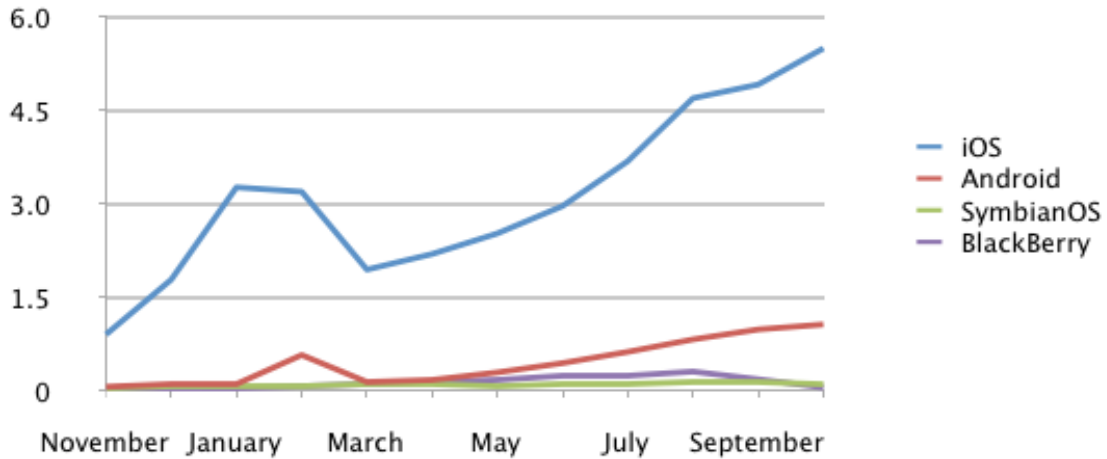
The proposed solution was to build a native application for mobile devices that allows the user to access many of the features of the client's website, such as browsing and buying whiskies, but also added the social component of being able to build a list of whiskies they wanted, owned or had previously drunk. Whisky enthusiasts enjoy talking about whisky as much as they do drinking whisky, so it seemed like a natural fit to build social features that allow users to compare the whisky they own and tell each other what they want in the future.

Background research

Unlike the PC world, where if you are going to develop a native application the developer would almost certainly choose Windows, which owns over 85% of the market, the mobile industry is still new and changing, so some background research needed to be conducted in order to choose which mobile platform to focus on. Each platform has its own development environment and languages, so its not possible to develop for multiple simultaneously. The client from the outset was attached to the idea of developing for the iPhone from word of mouth, however research was needed to confirm it was the right course of action.

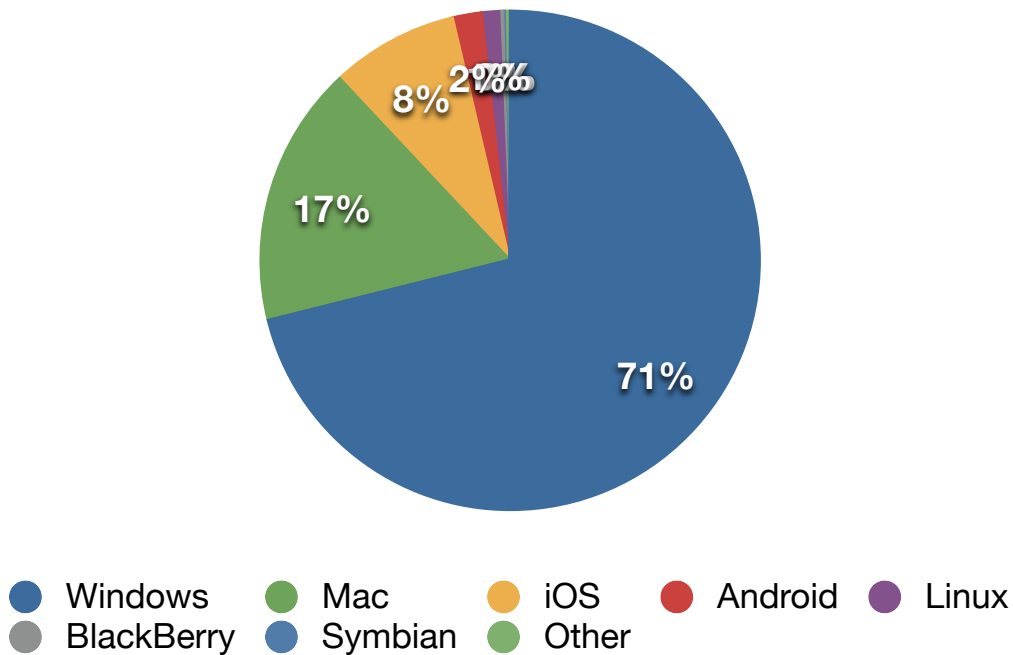
The graph on the next page shows statistics pulled for visitors to the client's website in the year leading up to the start of the project (Google Analytics, 2011). At the time of the 7% of visitors using mobile devices, 5.5% were using iOS, with just over 1% using Android.

Percent of visitors using mobile OS



If we compare these numbers to the site statistics at the end of the project, mobile users now account for 10% of devices, with just over 8% of visitors to the site using iOS, an increase of 2.5% in less than seven months (Google Analytics, 2011).

Visitors by Operating System



With an overwhelming majority of visitors using iOS devices, it seemed like the most ideal platform to focus on. Research was done (see appendix) into the multiple methods of developing for the iPhone. The biggest barrier to using Apple’s “approved” method of creating applications is learning the programming language Object-C, an object-oriented (a programming paradigm that uses data structures that contain values and methods) version of C, one of the most popular programming languages of all time. There are ways to develop for the iPhone using more familiar languages taught on the course, such as C# and ActionScript, however both these solutions are hacked on and have little developer support at this time. Its also questionable whether Apple will continue to allow them, as

they have blocked applications that use third party tools from being sold in the App Store in the past.

Research was also done into Apple and the controversy surrounding the App Store. Unlike Android which supports an open market, Apple requires all applications to be sold through their store, and approves or rejects submissions based on their somewhat vague guidelines. According to Apple's Phil Schiller (Hesseldahl, 2009), of all the apps that get rejected "about 90% of those cases, Apple requests technical fixes—usually for bugs in the software or because something doesn't work as expected". The other 10% are rejected because they try to steal customer data, break the law or contain inappropriate content.

The last reason is the most controversial amongst developers because Apple gets to define what is and isn't inappropriate. Some high profile rejections such as Google Voice, which was rejected for having "duplicated features that come with the iPhone" and cartoonist Mark Fiore, who's app was deemed inappropriate because it ridiculed public figures, have gained much attention in the press, however for the most part app's can be approved by Apple with minor modifications, so the risk was deemed acceptable by the client.

Aims and Objectives

The goal of my client in this project is to begin development on a new mobile and social strategy. In the future this strategy may include the development of apps for other mobile platforms, a universal web app and bringing the social features to their current website, however the client plans to launch this endeavor with a native application for iPhone, which is the focus of this project.

In entry two of the project log the scope of the project was defined with the client before the project began. The app being developed would include two primary components; a retail component and a social component. The retail component is required to deliver much of the functionality of the client's website within the app, including the ability to navigate and search for products, browse a product page to read about the whisky, add it to a basket and proceed to checkout where the product can be purchased.

The social component is required to deliver a social element to the app by introducing features not present in the client's website, such as building a whisky collection and wish list and adding tasting notes to the whiskies.

Other features were also proposed for the app, such as user profiles, friend lists, price comparisons and commentary from the web, however they were regarded as features the client would ultimately like to have in the app, but not required by the end of the project.

As all the final features of the app would not be delivered by the end of the project the goal of the project was agreed to not produce the final product that would be submitted to the app store, but to produce an effective working prototype of the product that the client would be able to see and use, that would later be expanded to become feature complete.

It was also agreed that the client would do the work necessary to produce a web service that would receive the application's order details, process and charge that order and record it in the client's current order processing system. The app would not do any of this work itself.

The choice of project may seem odd at first, as it involves learning a significant amount of new skills not taught on the course (the most obvious being the Object-C language, but also developing a native application for a mobile platform), however everything about the project is built on foundations learnt during University studies.

Object-C by its very design is an object-orientated programming language that relies heavily on the MVC (model-view-controller) architectural pattern. The course laid out the foundation of object-orientated programming in CI101 (Introduction to Programming) and expanded that knowledge in CI228 (Object Oriented Software Design and Implementation), which also introduced the MVC pattern. Although both these modules used Java as the programming language, the principles learnt carry over to Object-C.

The course also has a solid foundation in database theory and design in CI102 (Introduction to Databases), CI204 (Client-server Databases) and CI330 (Data Management) which will help immensely in designing a database for the application and writing SQL queries to process and send information from the client's database. CI203 (Web Application Development) also provided an understanding of the .NET framework and CI310 (Advanced Internet Applications Development) expanded upon it with an understanding of web services and Microsoft's entity framework which works remarkably similar to Core Data in the iPhone SDK.

Development Methodology

The application's development will be done using the agile software development model, as opposed to more typical development methodologies such as the waterfall or spiral model. The waterfall method is possibly the most traditional model, using sequential design where you move from requirements, to design, implementation and testing in a strict logical order, only moving from one stage to the next when the stage is complete and perfected. Many criticise this method of development as it is very difficult to declare a stage complete. For example, a client may need to see a working version of the product before they can completely define all its requirements. This would require you to return to the requirement and design stages after implementation, something not supported by the waterfall method.

Other development models, such as the spiral model, support such a process, where you repeat each stage of the process multiple times, however agile seems most appropriate for this project. The agile model is designed around the idea of iterative and incremental development, where a whole development lifecycle takes place within a short amount of time. In agile the "highest priority is to satisfy the customer through early and continuous delivery of valuable software" and it welcomes "changing requirements, even late in development" (Agile Manifesto, 2001). While this won't be followed to the letter in this project (in true agile a complete working product is produced at the end of each lifecycle), this methodology will allow for modifications to original plans as development progresses, and the ability to test working features on users before the end of the project. This methodology is ideal for a project that has to do a lot in a limited amount of time.

Project Planning

For a project that needed to be executed within such a short period of time, planning was paramount to its success. To achieve the goals of the project the time available was split into four stages.

Planning and Design Stage (four weeks: 6th December - 2nd January)

Development Stage (eleven weeks: 3rd January - 20th March)

Testing and Polishing Stage (two weeks: 21st March - 3rd April)

Documentation Stage (four weeks: 4th April - 29th April)

Within the largest stage (development) milestones were set on when necessary features needed to be complete. In the original draft of this timetable (see appendix) these were outlined as the retail and social features, however they were later refined to precise features such as the stages defined in entry six of the project log for the start of the project:

Stage 1) Develop an app that can communicate with a web service and pull down a feed of products.

Stage 2) Extend the app so that the web service is a component called the first time the app is used, and store the data feed in the local database using Core Data.

Stage 3) Use the data stored in the app to construct the navigation and views.

Some dates in this plan slipped in the final implementation (development didn't end until April 1st, requiring a much shorter period of time for testing), however it did assist greatly in keeping the project organised and on track.

In order to protect the project from data loss a rigorous backup solution was devised. On the primary development computer the program "Time Machine" was used to backup the computer to an external hard drive every night. An offsite backup was also kept using a service called DropBox, which not only syncs data between multiple computers (meaning there is a copy of the project on every computer DropBox is installed) but also keeps a backup in the cloud.

To assist in the development of the project the project log was kept on an online blog where posts could be tagged under categories such as "designs" or "development issues", other students could post comments and feedback and the entire log is easily searchable.

Development Process

Learning Object-C and iPhone Application Development

As stated in the aims and objectives this project builds on the many programming foundations laid out in the course, however possibly the biggest challenge of the entire project was learning an entirely new development environment and language, while completing the project in a very short amount of time.

It was decided very early on to learn iPhone development from a book, rather than online tutorials. Books have the disadvantage of being out of date when a new software version is released (most currently published books do not cover changes made in iOS 4.0), however they are much more comprehensive. Its very easy to find an online tutorial on how to do a certain thing in Object-C, but very difficult to find a site which fully covers everything you need to know about the subject. iTunesU has a lot of videos on iPhone development produced by Stanford University, but for the most part these are difficult to follow and see what the lecturers are doing. Learning started with the book "Learning iPhone Programming" by O'Reilly, chosen due to familiarity with other works by the publisher.

Learning from the O'Reilly book was a fairly difficult experience as it seemed to jump ahead pretty early on and presume you understood what the author was talking about. After becoming disheartened and lost in what the O'Reilly book was trying to teach the project as a whole was almost reconsidered in its entirety. It wasn't until "iPhone Programming: The Big Nerd Ranch Guide", recommended on a podcast on the TWiT Netcast Network, that things started to turn around.

The Big Nerd Ranch Guide takes the reader through the steps of learning iPhone development very well, always laying out the theory behind an idea before showing how it is practically implemented. Each chapter of the book contains step by step guides that can be followed to produce working examples and then conclude with challenges that encourage the reader to advance the code written in the chapter without help. An online message board is available for people who get stuck, and contains different working solutions to the challenges in the book. The book was produced as teaching material for an actual class on iPhone development taught by the authors, one of whom is a former employee of Apple, which is likely the reason the book works so well.

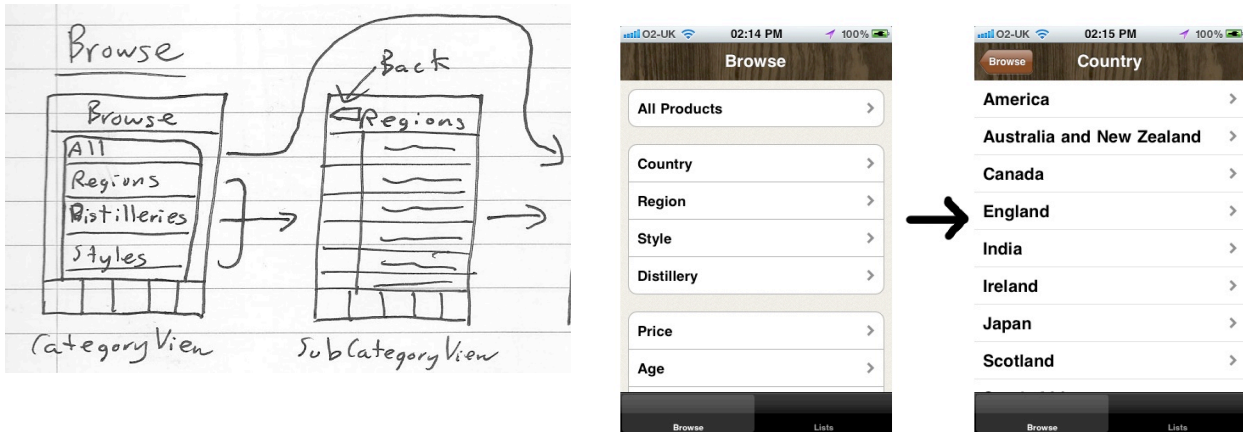
One of the small applications that was made following the book can be seen in entry three of the project log. Aside from the foundation teaching, the book also provided practical examples on how to use core data to talk to a database, implement list views and pull information from a web service, all of which were invaluable examples for the project.

Designing an iPhone application

The book demonstrated the theory behind each application's design using object diagrams, a UML diagram that shows a complete or partial view of the structure of a system at a specific time. Unlike a class diagram which shows the classes in a system, an object diagram shows the instances of those classes that are created by the system as it is running. Following the principles laid down by the book, an object diagram was used to create the design for the project's application.

Interface designs were drawn up using typical iOS interface layouts as inspiration and discussed in a meeting with the client (entry five in the project log). The client suggested

multiple changes to the design as it was discussed, including making the application UK only to reduce complexity with shipping, suggesting a better way to do accessories and requesting the user not be required to create a username and password to use the app. This input follows on from the client's work on their website, where the user is asked if they wish to create an account at the very end of the checkout, rather than being forced to create one at the start.



Through discussions with the client it was decided that in order to make an application better than those available on the App Store currently, the app needed to work offline as much as possible (an advantage of a native app) and be a fast download from the store (so a small file). In order to achieve the offline functions all the whisky data from the client's database would have to be stored locally on the device, and in order to be a fast download no information could be stored in the app.

The solution agreed upon with the client for achieving this was to create a XML feed of product data on the server that would be downloaded the first time the application launched. This requires the user to have an internet connection the first time the app is used, but then it will have the ability to work offline using the cached data in its database. This data could then be updated with changes to price and availability the next time the app gets an internet connection.

Competing applications that were analysed have massive file downloads from the store (up to half a giga byte) because they include everything, including product images, in the app download. This requires the creator to release an update to the app, which takes time to get approved by Apple and requires the entire app to be re-downloaded, to update product information. The alternative used by competing apps is to get all information from a web server, which requires the app to be online to function. On a mobile device that may have a data connection one minute and no signal the next, this really isn't an acceptable way to design an application, so the XML feed cached in a local database seemed like the best solution.

The iPhone SDK has multiple methods of storing information on the device. The app can write objects to a file (called archiving), use the SQL query language on a SQLite database or use a library called Core Data, which works in a similar fashion as Microsoft's entity framework, communicating with the database on the developers behalf and returning data in objects instead of records. Archiving was not a suitable solution because it doesn't support incremental updates to the data, and using SQL directly requires a lot more work with raw C code, so the Core Data library was chosen. It doesn't provide access to all the power of SQL, however for what the application required it was perfect for the job.

Building an iPhone application

From the designs and plans that had been created and signed off by the client, and with the knowledge gained from both the course and the reading material the project was ready to begin development. The experience and examples from the book made it very easy to make a head start into development and it wasn't long before a working version of the app existed that could pull product data from the web server, store it using Core Data and display it with basic navigation (entry seven in the project log).

From here a couple online tutorials were used to help expand knowledge not contained in the book, such as how to develop a search box for a product list page.

Developing the product page was one of the surprise difficulties in developing the project. A long panoramic page that scrolls from left to right was chosen for the design to introduce freshness and remove user fatigue, as all the previous screens in the application require a lot of scrolling up and down. When testing this feature with users however, they seemed confused as to what to do at this point and didn't understand where the screen arbitrarily stopped when they worked out you could scroll left and right. This issue was solved using inspiration from a flash application the client had briefly worked on in the past that showed whiskies on cards. The card metaphor for each "page" that the user arrived at when scrolling left and right made much clearer sense to the users it was tested on, and was loved by the client.

From a technical stand point, the product page also introduced challenges. When the user selects a product an image has to be downloaded from the server and displayed on screen. This initially locked up the entire application while the download happened, which could be a long time on the worst cellular data connections. This issue was eventually solved by using a more complex solution to retrieve the image, which downloads it in the background while the user can still interact with the app.

Another unforeseen issue with the product page was that product descriptions and tasting notes from the client's server were being stored in the database with HTML formatting. This is clearly fine if the only place the text would be used is a web page, however inside an application the only option seemed to be to strip out the HTML tags. After some research a better solution to this was devised after it was discovered that Apple's Web View designed to display web pages within applications could be used with a string of text. The web view itself was hidden from the user by giving it the same background as the card.

At this point the navigation to a product and viewing its information was complete and the project was well on time. The client signed off on the work done so far and made a suggestion regarding how lists should be handled. In the prototype application each of the individual lists (wish list, collection and basket) that the user could make were separate icons at the bottom of the app available at all times. The client requested this become a single icon for all lists so that the space at the bottom of the screen could be used for other features in the future. The list page itself was redesigned to allow for the user to change the list they were viewing and a forth list (empties, for bottles that have been drunk) was added upon request of the client.

Development of the list page caused the most delays in the entire project. There was no one issue that caused the delay, instead development of the feature was just more difficult than expected. Working with the data structure behind the list and allowing the user to

reorder and delete items was considerable more work than first thought, and in order to change quantities of an item the client's strict (and some what confusing) business rules (illustrated in the product report) had to be followed.

These delays meant that less time in the project was devoted to the checkout process than originally planned, and resulted in no time to implement the functionality of adding a tasting note to a product and caching images. Developing the checkout introduced its own challenges, as the client had a rather complex set of business rules for how delivery options should be provided. The checkout was designed in a meeting with the client, where use cases were used to agree how the user should step through the checkout and a flow diagram was drawn up to illustrate the client's shipping rules (see product report). The delivery algorithm itself (created on the client's server) took more time out of the app's development, meaning certain corners had to be cut in order to meet the deadline. As a result no validation was added to the screens that accepts the users address and payment details.

The final submission of the order to the client's server was also not complete, however this was agreed as work the client would do on their server to process and charge the order and was not implemented on their end in time. At present once the order is submitted into the application the user is simply sent back to the list view and told that their order was a success, despite the fact it was never sent to be processed and charged.

Testing an iPhone application

Testing was carried out throughout development of the project, revealing a number of bugs and application crashes related to memory leaks. As a developer more experienced with building web pages than actual applications, the principles behind memory management had to be learnt quickly. This is particularly important on a mobile device in which memory is limited. Understanding how to allocate and release memory is possibly the most difficult thing to learn in Object-C development and is the cause of most application crashes.

It was particular useful midway through the development process when the client registered as an Apple developer. Prior to this development was being done on a simulator, but as an approved developer applications can be deployed to real hardware, which highlighted issues not previously shown in simulation.

Documented testing was also carried out at the end of the project, in which a number of issues were highlighted that needed correcting, such as crashes if the server cannot be contacted, missing range checks and a logic bug in the delivery algorithm (see product report). There was nothing experience destroying however that stopped the prototype app from functioning in most use cases.

Legal and ethical issues

As an application centered around the consumption of alcohol it would have to be submitted to the App Store with an 18 rating in order to conform with UK laws on the age that alcohol may be sold to a person and Apple's own guidelines. Ethically, it would also be a good idea to display a statement about responsible drinking when the application is launched. The client uses the line "Master of Malt supports responsible drinking - Sip, don't Gulp." on its website, which would also work within the app. A prominent link to the Drink Aware website which provides alcohol advice would also be useful.

In order to conform to the Data Protection Act the app would also need to be modified. It should be clear to the user when they are storing addresses and payment details to be reused in future orders and credit card information should be encrypted and the full card number not displayed. The three digit security code should not be stored and the checkout should require a password to reuse saved details. When the order is being sent to the client's server for processing this should also be done using a secure connection.

Reflection

Although not completed with all the features it was originally planned to include (tasting notes, sending an order to the client's server for processing), the final product at the end of this project achieved the goals it set out to. The working prototype of what will ultimately be an application available on the App Store runs on real hardware, has no show stopping bugs and impressed the client who is excited to see the product complete.

Of the many components the product has the developer is most proud of the user experience delivered when browsing products, as using the cards to swipe between pages is a rich and engrossing way of viewing information on a mobile screen. The list building social component should also be noted, as a feature unique to the application which works particularly well and is very cleanly designed.

The application produced at the end of this project is clearly not the final product. With more time many more features would be expanded and added, such as a functioning checkout and the ability to add tasting notes. With further research and time however, it would be great to expand the application to have a native interface for the iPad, which with a large 10 inch display could allow for a much larger, more exciting user interface on top of the same backend code.

Ultimately however, if the developer was to redo this project or give advise to another student about to do the same undertaking, it would be recommended that the developer didn't decide to jump head first into a development environment they had never experienced before. Although initially interesting, learning Objective-C without knowledge became a larger than anticipated challenge. Learning a new language is non-trivial - implementing it in a project as you learn is more non-trivial.

With a lot of hard work and late nights studying the goals of the project were ultimately achieved, however it seems like an effectively as impressive project could have been produced without the stress this extra, volunteered amount of work introduced.

Many lessons in time and project management were also learnt during a project of this scale. Planning should have started earlier to allow more time for development and testing. Issues that arose due to unfamiliarity with the development environment, particularly memory management would have been easier to solve had more time been allocated.

Overall the project was a success and a fantastic learning experience in independent study, cooperating with a real client and working to a deadline.

References

Allan, A (2010). *Learning iPhone Programming*. O'Reilly Media.

Arthur, C (2011) Nokia and RIM bleeding smartphone share while Android cleans up [Online] <<http://www.guardian.co.uk/technology/2011/apr/18/smartphone-market-android-win-nokia-rim-lose>> [accessed 15 April 2011]

Agile Manifesto (2001) Manifesto for Agile Software Development [Online] <<http://agilemanifesto.org/>> [accessed 14 November 2010]

comScore (2011) Apple iOS Platform Outreaches Android by 59 Percent in U.S. When Accounting for Mobile Phones, Tablets and Other Connected Media Devices [Online] <http://www.comscore.com/Press_Events/Press_Releases/2011/4/Apple_iOS_Platform_Outreaches_Android_by_59_Percent_in_U.S> [accessed 17 April 2011]

Conway, J. & Hillegass, A (2010). *iPhone Programming: The Big Nerd Ranch Guide*. Pearson Technology Group.

D'Monte, L (2009) Swine flu's tweet tweet causes online flutter [Online] <<http://www.business-standard.com/india/news/swine-flu%5Cs-tweet-tweet-causes-online-flutter/356604/>> [accessed 16 April 2011]

Facebook (2011) Statistics [Online] <<http://www.facebook.com/press/info.php?statistics>> [accessed 16 April 2011]

Google Analytics (2011) Google Analytics [Online] <<http://www.google.com/analytics/>> [accessed 13 April 2011]

Hesseldahl, A (2009) Apple's Schiller Defends iPhone App Approval Process [Online] <http://www.businessweek.com/technology/content/nov2009/tc20091120_354597.htm> [accessed 14 November 2010]

NetMarketShare (2011) iOS vs. Android in Browser Usage [Online] <<http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustom=iOS,Android,Linux&sample=29>> [accessed 16 April 2011]

Nielsen (2010) Smartphones to Overtake Feature Phones in U.S. by 2011 [Online] <<http://blog.nielsen.com/nielsenwire/consumer/smartphones-to-overtake-feature-phones-in-u-s-by-2011/>> [accessed 16 April 2011]

Shiels, M (2011) Twitter co-founder Jack Dorsey rejoins company [Online] <<http://www.facebook.com/press/info.php?statistics>> [accessed 16 April 2011]

TWiT LLC (2011) The TWiT Netcast Network with Leo Laporte [Online] <<http://twit.tv/>> [accessed 17 April 2011]