Project title: “MembaShip”

1. Introduction

Problem Statement:

The climbing walls I use in Bristol each come with their own membership card that has a unique barcode printed on it that is scanned every time you climb. It’s not just climbing memberships that cause this, any card with a barcode like loyalty cards fall into the same problem. Each card takes up the limited space in a wallet or purse which could mean that curtain cards wouldn’t be kept in there until they are needed. This creates the problem of forgetting your cards and leaving them behind when you need them.

Our phones hold a high priority in our lives, so we take better care not to lose it and keep it on us, it’s rare for us to not have our phone nearby. So I’m going to create an app that stores all your membership card details (barcodes that you scan on entry) within it to save the user having to carry every card they use on their person and eliminating the potential struggle of rummaging through their wallet or purse to find the card they need.

1. Method

Suitable Tools:

With the advice of my supervisor I will be designing my app with JavaScript along with its libraries, mostly react.js. I could end up using AJAX, if this is the case then ill also utilise PHP to handle the data that is returned and to write the pages/files that AJAX is calling too. Then use visual studio code as my IDE.

It makes sense for me to use these coding technologies as these are the ones I’ve had the most experience with excluding react.js, I’ll need to do some research into this technology.

There is also the option for me to implement a framework like Ionic and React Native. I feel ill gain a better learning experience building my app from the ground up on my own but, there is the issue of neither android or IOS being able to run JS natively so I will need to use React Native as this will bridge the gap.

For the UML I’ll be using Astah for the same reasons for my code language choices as this is the software I have the most experience with for producing UML.

Methodology:

I have chosen extreme programming as the methodology I’ll be using due to its focus on frequent releases in short development sprints that encourage change when needed. Therefore, I can retroactively reflect on my work and make changes before moving onto the next section of the project.

Extreme programming also requires developers to plan and understand the customers user stories. This is also a benefit for me as I will start coding after I have created the user stories and UML.

The practice of designing with simplicity in mind and testing your code often are practices that I believe will be a successful routine for me and the project.

I made this decision based on the detail in the article (LeanKit, 2019).

Risks:

The organisations issuing the membership cards taking issue with the app therefore not being happy with the card details being used within a third-party setup.

Another program/app existing that works either similarly or completely like my proposed idea would.

Schedule/deadline issues with my other modules affecting the time I’m dedicating to this project.

Issues with learning the new and already known languages I need to use.

Having not used UML and use cases for a while this could pose some time-wasting issues.

Version Control:

<https://gitlab.uwe.ac.uk/c23-day/dissertation>

1. Research

Primary Research:

Informal Sources (User Stories):

“User” = customer of membership card organisation.

“Business user” = worker within membership card organisation.

* As a user I want to be able to store my card barcodes on my phone, so I don’t have to take the cards with me. (Priority: 1)
* As a user I want my phone to display the selected card barcode to be scanned so I can gain access to the card’s benefits. (Priority: 2)
* As a user I want the app to be easily navigable so I can have a stress-free experience when using it. (Priority: 3)
* As a user I want the app to be dependable enough as to warrant its use over the normal cards (Priority: 5)
* As business user I want the app to be as easy for me to work with as the physical cards, so my workload isn’t increased. (Priority: 6)

Use Cases:

Functional/Non-Functional Requirements:

Functional Requirements:

Non-Functional requirements:

Case-studies:

Existing market:

When I was asking the climbing walls around Bristol for their permission to use the membership cards they supply, one of the companies highlighted the existence of an existing app that provides the same functionality that I’m aiming to provide called “Stocard”. After finding “Stocard” I was suggested more apps that work the same from the Google play store. I think this would be a good reference point for the design of my app as I can highlight what these apps do well and the areas I can improve or new features I can create if it’s lacking some functionality.

Stocard Pros:

* UI is simple and easy to understand and use
* The app provides exclusive offers and coupons to its users
* Can store bank cards
* Uses fingerprint scans and code security, can freeze cards
* The app works well on smartwatches
* Well reviewed, 4.8\*/ 5\* out of 500k reviews, 10M+ downloads, implies very few bugs

Stocard Cons:

* Although the UI is simple, It’s a little bland
* Finite number of free spots to use bank cards, pay needed otherwise

Cards Pro:

* Prettier UI
* Has the option to open any card instantly from outside the app
* Offers same functionality and security as Stocard
* Well reviewed, 4.6\*/5\* out of 61k reviews, 1M+ downloads

VirtualCards Pros:

* Can create an account to act as a card backup
* Can apply for new cards from select partners
* Contains a shopping list you can populate with desired items and any offers relating to those items are displayed
* Users can use their voice to add items to the shopping list
* Well reviewed, 4.7\*/5\* out of 21k reviews, 500k+ downloads

VirtualCards Cons:

* Similar UI to Stocard
* Can’t be used on smartwatches
* Can’t be used with bank cards

From this research I’ve gained a solid perspective on the market for these types of apps and what direction I should go in for my design. With combined downloads of 11M+ there is definitely a demand for this type of service. It also seems users are more interested in the app’s functionality over its presentation with these types of services, I feel this is true as the reviews are exceptionally high for the apps which have bland UI experiences. This gives me a lot of freedom with the UI as the example set from these apps is relatively low.

Some of these apps are offering the option to store bank cards to use the phone for contactless payments. This aspect does interest me as it follows the problem solution my app is aiming for, that of reducing the risk of leaving the cards you need at home/work and saving space with a wallet/purse. The only set back here is that there will need to be ridged security in place seeing as something as sensitive as bank details are being handled. I feel this could be quite time consuming, with me needing to learn react.js I don’t want to set the scope to wide then not have enough time to fulfil the project.

Some of these apps can run on smartwatches, which Is a great addition to the app’s functionality as it is increasing the ease of use but getting rid of the need get your phone out of your pocket. With the number of smartwatches being bought on the rise (Liu, S. 2019) this seems like it makes sense to include this functionality whenever possible. This would be a smart decision to include this in my app given what problem I’m trying to solve with it. The issue is I’m unsure about the setup of smartwatch apps, like is it easy to port existing apps over or do you need to build the app again specifically for the platform and could there be new technologies I haven’t seen before be required to do so. This could lead to the same issue of time consumption mentioned with implanting the use of bank cards but it’s worth researching to find out the answer.

Voice Recognition:

Use of voice recognition is an interesting aspect provided by VirtualCards and opens the potential positives that can be added with speech to text/text to speech functionality. In terms of accessibility it can extend the usability of my app, by allowing anyone with disabilities that would either struggle to use or just couldn’t use the app a more manageable experience. Even for users with out disabilities it can offer a better user experience, an example of this would be if the user had something in their other hand so it might be easier for them to use voice recognition. There is the possibility that most users will find using their hands the more natural and easiest choice, but it would still be a good idea to offer the option. With a search on google I’ve found a couple API’s and some video tutorials so I feel this should definitely be a piece of functionality I should try to implement into my final design if I have enough time to spare after completing the main functionality of storing card details.

Mapping:

When looking at the existing apps I thought about how useful a map function might be to the users. My thought process behind this is having a map that displays all the locations around you that you could use the cards that have been stored in the app with, this is more suited to something like loyalty cards. Something like Tesco as there are going to be more of these around to make use of this feature. Not to say climbing walls couldn’t benefit from this as there are climbing companies that span across cities and can have multiple venues, like TCA have two venues in Bristol. If you’re in a new city for a few days and want to climb, it would provide a handy way to see if there are any venues that you’re a member of near you. Although this seems like it’s an added step when using Google maps to find somewhere. For this reason along with wanting to keep my app simple enough for my timeframe, I don’t think there is a good enough reason for me to work this functionality into my app.

Smartwatch (wearable app) implementation:

There is some solid documentation being provided from Android, IOS and Samsung on developing wearable apps, only Samsung allows for JavaScript to be used within their SDK (Samsung, 2020). With this being limiting, I believe that it would be better for me to focus on using a JavaScript framework called React Native to build my main app but keep the option to create a wearable app open for implementation if I have time available to do so. This could be possible with my timeframe as React Native can create wearable apps as well, so I should be able to carry my knowledge over from my original app.

New Tools:

As I did my research into the case studies lead me to the smartwatch idea, it came to my attention that my idea of creating an app with plain JavaScript and React.js wasn’t the best wait to approach development. This is because the Android and IOS don’t natively support apps written in JavaScript, this led me to search for technologies that would allow me to keep JavaScript and its libraries as the code language. This is where I discovered the React Native framework.

React Native framework and Expo:

Whilst doing the research for the smartwatch idea, the issue of code language became visible to me. Android apps being written in Java (Android, 2019) and IOS apps being written in Objective-C or Swift (Apple, 2020), I’ve used a little Java but not enough to be confidant with this kind of project and the other languages I’m not entirely familiar with. This led me to discover React Native. I’m going to be using React Native as this will allow my app to be implemented to both of these OS’s. This is because React Native can allow for a single codebase to be applied to many platforms like Android and IOS (Facebook, 2019), this will save on time during development whilst allowing for a wider reach of my app. JavaScript is a language that is becoming increasingly popular within the web development space which is the space I’m aiming to get a job in so, I want to learn more about. This framework lets me keep it as my primary language.

This does mean that time is going to have to be dedicated to React Native, although react.js was also an area that needed the same attention so they should fit together nicely and reduce time dedicated to learning them.

There is a couple of extra components to install to before React Native can be used Node.js and Expo CLI (Command line interface). There is the option to use React Native CLI, but the React Native docs recommend the Expo CLI as its better suited to newcomers in the mobile app development space, which I am. Node.js will allow me to execute JavaScript code outside of a browser. Expo is set of tools built around React Native with many features like, working around the need for XCode (IOS IDE) or Android Studio, access to the Expo SDK which provides a wide variety of native API’s on IOS and Android, Expo can manage your assets, take care of push notifications, build your final native binary for submissions to the app stores and can be used in most code editors (Expo, 2020). Expo also provides a mobile app that will read a QR code from the CL, this enables testing how the code you’ve written will perform on mobile setups as this will simulate the typical app environment on mobile devices. They do provide a service called Snack which allows to test React Native code without having to install any tools, but I feel I would be better suited to use the tools provided with Expo and test my app on the platform its being designed for.

The React Native CLI provide more customisation for a more experienced app developer as you can select what you’re developing OS is (macOS, Windows, Linux) and what target OS you’re developing for (IOS, Android), what dependencies you need, specific version control and relevant SDK set up. as this is the more experienced way to use React Native the assistance Expo provides mentioned above is lost. when comparing this with Expo, it’s clear to me that the use of Expo would be better for my situation as I’m not at the level and don’t have time available to be to get there in time for the end of this project. The assistance that Expo offers will help me spend more time into the core code functionality and design of my app.

The original plan for the layout/design of the app was for it to be in a web app format with just plain JavaScript and React.js, as this would have been easier for me to create due to lack of experience and skill with the tools required to develop a app for either the Android or IOS markets. With the discovery of the React Native framework and the tools that can be used alongside it, this puts me in a position where I can create my app in a more comfortable and appropriate environment for mobile app development. Especially with the added functionality that Expo provides with the option to easily test your design on a physical mobile platform will allow me to have a greater control over the design and functionality of my app as I can constantly test and retest my program as I develop it. With these technologies in use my app won’t be in the web app format but more Android focused.

Technology Selection (Pugh Matrix):

References:

# LeanKit (2019) *Top 6 Software Development Methodologies* Available from: <https://leankit.com/blog/2019/03/top-6-software-development-methodologies/> [Accessed 02/02/2020].

# Liu, S. (2019) *Number of Connected Wearable Devices Worldwide By Region From 2015 to 2022* Available from: <https://www.statista.com/statistics/490231/wearable-devices-worldwide-by-region/> [Accessed on 10/02/2020].

# Samsung (2020) *Create A Web App Project* Available from: <https://developer.samsung.com/galaxy-watch-develop/creating-your-first-app/web-companion/create-project.html> [Accessed on 16/02/2020].

# Android (2019) *Create An Android Project* Available from: <https://developer.android.com/training/basics/firstapp/creating-project> [Accessed on 16/02/2020].

# Apple (2020) *Swift* Available from: <https://developer.apple.com/swift/> [Accessed on 16/02/2020].

# Facebook (2019) *React Native* Available from: <https://facebook.github.io/react-native/> [Accessed on 16/02/2020].

# Expo (2020) *The Fastest Way From An Idea To An Native Experience* Available from: <https://expo.io/features> [Accessed on 17/02/2020].