Membaship

Dissertation Report

Chris Day 17033796

Software Development Project

Contents

[Introduction 2](#_Toc38284120)

[Problem Statement: 2](#_Toc38284121)

[Method 2](#_Toc38284122)

[Suitable Tools: 2](#_Toc38284123)

[Methodology: 3](#_Toc38284124)

[Test Driven Development: 3](#_Toc38284125)

[Project Cycle: 4](#_Toc38284126)

[Risks: 5](#_Toc38284127)

[Version Control: 6](#_Toc38284128)

[Research 6](#_Toc38284129)

[Case-studies: 6](#_Toc38284130)

[Existing market: 6](#_Toc38284131)

[Voice Recognition Implementation: 7](#_Toc38284132)

[Mapping Implementation: 8](#_Toc38284133)

[Smartwatch (wearable app) implementation: 8](#_Toc38284134)

[Existing Apps Using React Native: 8](#_Toc38284135)

[Research Outcomes: 8](#_Toc38284136)

[Informal Sources (User Stories): 9](#_Toc38284137)

[Use Case: 9](#_Toc38284138)

[Functional/Non-Functional Requirements: 10](#_Toc38284139)

[Design 10](#_Toc38284140)

[Software Architecture: 10](#_Toc38284141)

[Quality Assurance: 10](#_Toc38284142)

[Usability: 11](#_Toc38284143)

[Functionality: 11](#_Toc38284144)

[Reliability: 12](#_Toc38284145)

[Portability: 12](#_Toc38284146)

[Efficiency: 13](#_Toc38284147)

[Maintainability: 13](#_Toc38284148)

[Conclusion, Next Steps & Reflection 14](#_Toc38284149)

[Conclusion: 14](#_Toc38284150)

[Next Steps: 14](#_Toc38284151)

[Reflection: 15](#_Toc38284152)

[References 15](#_Toc38284153)

[Appendix 16](#_Toc38284154)

[Technology Selection: 16](#_Toc38284155)

# Introduction

## Problem Statement:

The climbing walls 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 this project exists to create an app that stores membership card details (the 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.

# Method

## Suitable Tools:

The app will be designed with JavaScript along with its libraries, mostly react.js. The JavaScript framework called React Native will also be used.

It makes sense for these coding technologies to be used as these are the ones that will be able to create the functionality this app needs and relatively high experience has been gained with these technologies before this project excluding react.js and React Native, there will need to be some research into these two technologies.

There is also the process needed to implement the framework React Native. There is this issue of neither android or IOS being able to run JS natively so the React Native framework will need to be used as this will bridge the gap between language and platform. This will be a great learning experience using this framework as its use it rather popular despite its relatively young age, so it’ll be useful to have this knowledge. Its use will also improve knowledge with JavaScript and react.js, with these become more and more popular too is a huge benefit also.

The React Native GitHub was made public in 2015. This has plenty of documentation available to users of all kinds of experience levels, offering help from start to finish of a project. This will a great resource to use throughout the app’s development.

Codecadamy is a website that offers tutorials on a range of coding languages including JavaScript and its libraries like react.js, although it doesn’t have any tutorials on React Native. this will be used to make sure the core concepts of JS and react.js are understood before starting app’s development.

LinkedIn learning would be useful for tutorials on React Native, JavaScript and react.js. If it becomes a struggle to understand any part of the documents from the React Native GitHub or Codecadamy attention will be given to LinkedIn learning to find someone to explain the process to help gain a better understanding.

YouTube has plenty of videos explaining React Native, JavaScript and react.js. this will be used as a backup just in case LinkedIn lacks any tutorials that are needed.

For the UML designs Astah will be used for the same reasons for the code language choices as there has been a lot of experience with this software’s for producing UML.

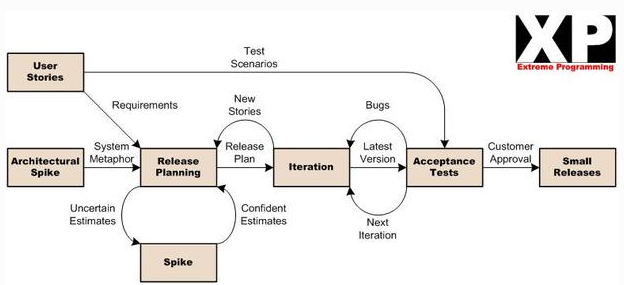
## Methodology:

The main methodology to use in this project is extreme programming as its focus on frequent releases in short development sprints that encourage change when needed. Therefore, retroactive reflection can be applied on any work and changes can be made before moving onto the next section of the project.

Extreme programming also requires developers to plan and understand the customers user stories. This is a benefit for the project as coding will start after the user stories and UML have been created. This will help keep the project’s development focused and progressing within a suitable time frame, as reference can be made to the UML and user stories to help reduce the apps functionality drifting from its original design.

The practice of designing with simplicity in mind and testing your code often are practices that will help lay down a successful routine for the project.

This decision was based on the details in this article (LeanKit, 2019).



This diagram can be found at antossoftwaretopics site (Antossoftwaretopics, 2016).

### 

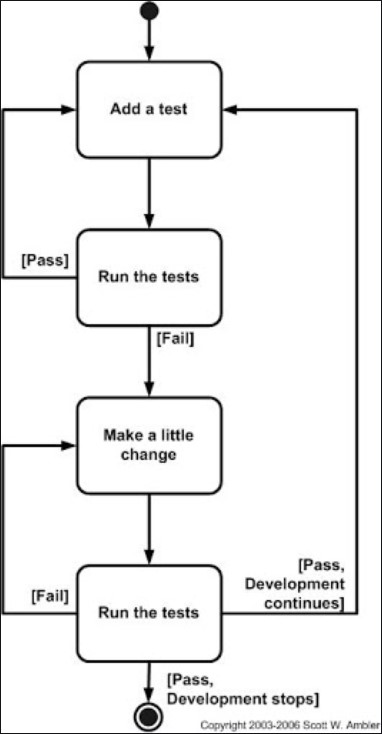
### Test Driven Development:

With this project being written in a coding language and we are making use of a framework that allows for functional programming, this will allow for a form of test-driven development to be performed on this project.

The React Native framework borrows heavily from the react.js library like its structure and syntax while introducing some of its own like “View”. Because it uses react.js structure it means that apps structure under the framework is split into components. Components are like JavaScript functions that allow you to split the UI into independent and reusable pieces, each component handling one aspect of the UI or functionality.

Due to the nature of components utilising function behaviour and their independency, projects using React Native and the tools that can be used beside it (like Expo) becomes an ideal project for the use of the underlying themes of test-driven development. This is because functions can be tested after they have been written due to their behaviour staying consistent to how they are coded. The parameters (“props” in React Native terms) passed into the components can change, but that shouldn’t affect the actions the components are coded to do.

In the case of this project the test-driven methodology followed with ease as the supporting software called Expo along with help from Node.js allows the use of a test environment to test the written code, not only through a browser but through a mobile device. Once the test environment has been set up the code file can be changed, once the changes have been saved the test environment will automatically compile the changes and display the new UI. This demonstrates more reason why this project’s development would benefit from this aspect of test-driven development.



This diagram can be found at agiledata.org (Agiledata, 2018).

### Project Cycle:

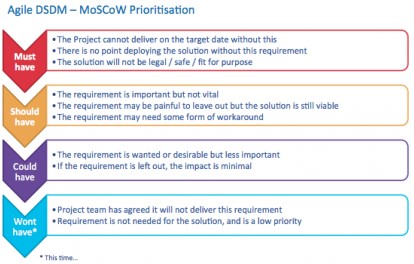
Through the entirety of any app’s life it will be broken down into development cycles. From the initial release until its deemed unresourceful to dedicate any more time to its development. Each cycle will focus on either adding new functionalities or updating or patching current issues. Many of the agile project management methods follow this structure. The MoSCoW method plays a big part in the project development cycle.

The MoSCoW (Must have, Should Have, Could have, Wont have) method is a way to prioritise requirements based on the importance the identified stakeholders would place on each one, Must have being the most important working down to Wont have being the least important.

Using the MoSCoW method in the functional/non-functional section of this document, its easily laid out what order the priorities of the development cycles should be. Because this project will be starting at the beginning the first development cycle will be focused on the M (Must have) in the MoSCoW method, depending on size of the project, size of the development team and project timeframes then the list of “Musts” might be further broken down and carried over into the next cycle.

The cycles following the completion of the “Must” functionalities would then work focus on the rest of the prioritised functionality starting with S (Should have) then C (Could have). Although the order could change by the time the next cycle starts, as new requirements could be added. With reflective practices encouraged in a lot of development environments, potential feedback from stakeholders after the first development cycle (if releasing the project was a goal at the end of that cycle) and/or changes in the respective markets, previous requirements could change by the time the next development cycle starts. Pushing unworked requirements from previous cycles down the priority list as newer requirements are deemed more important.

The development of this project will follow this method exactly, first focusing on the Must’s for this cycle, then for working through the priorities in the following cycles. With the project scope being relatively small the Must’s should be achievable on with the first cycle.



This diagram can be found at apm.org.uk (Apm, 2014).

## 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 the proposed app idea would.

Schedule/deadline issues with other modules affecting the time dedicated to this project.

Issues with learning the new and already known languages that need to be used.

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>

# Research

## Case-studies:

### Existing market:

When 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 this project is aiming to provide called “Stocard”. After finding “Stocard” more apps were suggested that work the same from the Google play store. This would be a good reference point for the design of the app as it can highlight what these apps do well and the areas that can be improved or new features can be created if it’s lacking some functionality.

Stocard (Stocard, 2020) 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

Stocard (Stocard, 2020) Cons:

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

Cards (Cards, 2020) Pros:

* 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 (Virtualcards, 2020) 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 (Virtualcards, 2020) Cons:

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

From this research a solid perspective has been gained on the market for these types of apps and what direction should be taken for the apps design. With combined downloads of 11M+ there is enough evidence to show that 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, this is true as the reviews are exceptionally high for the apps which have bland UI experiences. This gives the developer 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 perk interest as it follows the problem solution this projects 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. This could be quite time consuming, with the needing to learn react.js the scope of the project should remain relatively small in the first cycle of its development to ensure enough time is available 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 the app given what problem is trying to solve with it. The issue is the uncertainty 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 haven’t been 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 Implementation:

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 the 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 without 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 the results found a couple API’s and some video tutorials so this should definitely be a piece of functionality that should try to end up implement into the final design for this cycle, if there exist enough time to spare after completing the main functionality of storing card details.

### Mapping Implementation:

When looking at the existing apps an idea occurred about how useful a map function might be to the users. The thought process behind this is having a map that displays all the locations around you that you could use with the cards that have been stored in the app, 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 the app simple enough for the timeframe, there isn’t a good enough reason for the app to feature this functionality.

### 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, it would be better for the focus on the main app but keep the option to create a wearable app open for implementation if there is time available to do so. This could be possible within the timeframe as React Native can create wearable apps as well, so the option to carry knowledge over from the original app will be available.

### Existing Apps Using React Native:

The framework being used for this app’s development is becoming more and more popular after Facebook opened its availability on GitHub in 2015, as an alternative method to develop native mobile apps outside of the standard languages (Java/Kotlin and Swift/Objective C). Some popular apps developments are transferred over to or created from the ground up with this framework like; Facebook, Instagram, UberEATS, Discord, Discovery VR and Airbnb (Kakar, K. 2017). With its wide variety of use and with big names behind it, this gives more motivation to use this framework as It would be a valuable skill to know for any future projects like this one, as its use is increasing.

### Research Outcomes:

With the research completed in this section, the project has developed its scope. In terms of the first project cycle the scope hasn’t be developed by expansion but through refinement. With the first cycle not set to provide any functionality besides taking and storing pictures of cards, the research being utilised for this cycle will be centred around UI and presentation. Looking at how the existing apps are presented and taking influence from their decisions and creating something new and improved.

When applying this research to multiple project cycles the scope may grow to include the development suggestions like the smartwatch and voice recognition ideas. Not to say these will be implemented, as it may be deemed through further research in future cycles that other functionalities will be more important. These ideas for functionality will remain for potential development if they are deemed useful.

## 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 by taking a picture, 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 Case:

A picture containing text, map

Description automatically generated

Store barcode – allow user to store their barcodes on their phone through the app.

Display barcode – app displays selected barcode for use.

Simple to use – easily navigable design for ease of use.

More dependable – app should be reliable enough over the normal method of scanning cards to warrant its use.

Easy to work with – app shouldn’t make extra work for staff as this could lead them to banning it.

## Functional/Non-Functional Requirements:

Functional Requirements:

* My app must store barcodes locally in the form of images.
* My app must display those images so the barcodes can be scanned.
* My app could have voice recognition for ease of use.
* My app could have wearable portability (use on smartwatches).
* My app won’t have any mapping functionality.

Non-Functional requirements:

* My app must be easily navigable.
* My app should be more dependable than the current card system.
* My app should be easy for business users to work with.

# Design

## Software Architecture:

Native components

Core libraries

Style and layout

My App

React Native framework

Operating system

## 

## Quality Assurance:

To ensure the development of this project is performed with high quality, it would be best to be stick to the software engineering quality standard ISO/IEC 9126. The main goal of this standard is to try and reduce some of the biases that can have a huge effect on the development and final delivery of the software. Issues like changing project priorities during development or not having an agreed idea of what the project end is are the kind of problems that are trying to be avoided to ensure the highest project efficiency and quality. This standard can be easier to follow when its structured into six characteristics, each focusing on one aspect of software development.

### Usability:

In software engineering projects usability is an area that focuses on how well the developing software can be used by the target consumers to achieve expected outcomes.

The sub-characteristics of usability for a software system are understandability (easy to use), learnability (easy to learn), operability (keeping system in safe and reliable condition) and attractiveness (being helpful to the user). There should be an early focus on end users and the tasks they need completed by the system (Gould et al., 1985), so use cases are vital for this area of the ISO standard.

With this project being about developing an app usability is a very important part of the ISO standard to focus on. More obscure software used in older industries for example, might not have the usability standard at a high priority due to lack of competing software, so they could feel resources would be spent in other areas. Due to the nature of the huge and diverse markets that mobile apps exist on and with app design heavily relying on usability, poor usability could lead to users moving to an alternative app offered on the app stores or going back to using traditional methods, in this projects case using the original cards. To ensure constant reuse this app will need to follow the usability principles mentioned above to keep users happy with the service.

This project will conform to the usability sub-characteristics by its design being simple. The app will only be able to take pictures of new cards and display saved cards, this means that there will only be a couple of clear menus with simple functionality making it conform to understandability and learnability. For operability React Native will help keep the software reliable across platforms. The apps simple design will help keep reliability as there only a couple of “moving parts” in its design, so this reduces the chances of the software falling over. Being able to test the components of the software as there written will aid in reliability as well. The projects reason for existing will cover attractiveness as it solves the issue of carrying/forgetting membership/loyalty cards which will be helpful to the user.

### Functionality:

Functionality is an area that focuses on the essential purpose of the project. Functionality is shown as a totality of essential functions that the software provides (Medium, 2017).

The sub-characteristics of functionality for a software system are suitability (providing suitable functions for specified tasks), accuracy (functions providing right/expected results), interoperability (software interacting with separate systems), compliance (laws and guidelines that need to be complied with) and security (preventing unintended access and resist attacks) (SQA, 2019). With this characteristic, functional requirements are important as they define what functionality the project should have.

For this project its functionality included in its first development cycle makes it easy to define as the number of functions is small. The more functions a piece of software is meant to have creates greater complexity which makes it harder to define its functionality (SQA, 2019), therefore making it harder to explain the sub-characteristics of the project.

This project will easily conform to high standards with the functionality sub-characteristics due to its limited functionality in its first development cycle. With the initial functionality shown to be relatively simple through the functional requirements for this project, the functions needed to achieve the requirements shouldn’t be that complex making it easier for suitable functions to be written. The functional simplicity would also aid in function accuracy This would cover the suitability and accuracy sub-characteristic. In terms of interoperability this project doesn’t interact with many other systems. The only system it interacts with is the barcode reader, so the only requirement needed for this cooperation to work is the picture the barcode reader reads to be in a high enough resolution for easy reading. This requirement relies on the hardware of the device that’s taking the picture rather than the app itself, therefore out of the apps control. With this project being self-contained and separated from other systems plus with its functionality not including any form of data collection there isn’t any issues concerning the compliance sub-characteristic. The security sub- characteristic isn’t applicable for this project as there isn’t any incorporation on online services or connection to of systems that do.

### Reliability:

Reliability focuses on the capability of software to keep itself at the performance and functionality under certain conditions for a defined amount of time (SQA, 2019).

The sub-characteristics of reliability for a software system are maturity (frequency of software failure), fault tolerance (software’s ability to withstand from component or environmental failure) and recoverability (ability to recover a failed system to full operation) (SQA, 2019).

The reliability characteristic for a project is reflected through its complexity, reduced software complexity means reducing the chance of software failure and increases software rigidity. More complexity generally means more functions which could be reliant on each other or other external services which could lead to brittle software. This projects first development cycle functionality leaves it with low complexity making its reliability characteristic a high standard.

This project will have a high standard reliability characteristic due to its functionality creating a simple design, avoiding complexity. This means in terms of maturity this projects software will have very little room for failure due it only containing a few components. The functionality of the software only allows for the use of the camera, accessing saved photos, UI and touch navigation. These functions/components will take simple inputs with no option to provide other inputs providing strong software failure protection. This can be applied to fault tolerance as the components are only passing limited props between each other reducing complexity and the problems that come with it. The testing that can be done through the Expo tool will help in finding and solving component failures. Environmental failures will be based on React Native as that handles the compiling of the JavaScript into the different mobile environments. Attention will have to be given to updates for React Native as this could affect this project after deployment. With the design of the project being so simple the whole app would be one system and it doesn’t interact with any systems; this makes the recoverability sub-characteristic hard to define. Looking at the system this project is creating, if it fails that means the app has failed which would mean an app reboot would be needed.

### Portability:

Portability focuses on a piece of software’s ability to adopt changes in its environment or its requirements. Object oriented design and implantation practices can add to the level to which this characteristic is present in each system (SQA, 2019).

The sub-characteristics of portability are adaptability (ability of a system to change to new specs or environments), instability (the effort required to install the software), coexistence (capability of software to share a common environment and resources with other independent software) and replaceability (the ease of exchanging software components in a specified environments) (Ques10, 2019).

The portability of this project is quite limited with its independent design and functionality. It doesn’t rely or interact with exterior software, so in terms of coexistence the app doesn’t share environments or resources. Its limited and simple components make it easy to change to new specs and/or new components entirely. The React Native framework makes changing to new environments on mobile very easy as it handles the conversion to new platforms itself. When thinking about instability the effort required is no more than the normal process of installing an app on your phone through the app store, so this will be a very comfortable experience for users.

### Efficiency:

Efficiency focuses on the relationship between software’s performance and the resources used under conditions created by the software’s functionality. The amount of disk space, memory network etc are good examples of this characteristic (SQA, 2019).

The sub-characteristics of efficiency are time behaviour (software ability to produce correct response and processing times and throughput rates) and resource utilisation (software’s ability to use appropriate resources in a suitable time) (Ques10, 2019).

Taking time behaviour into consideration for this project’s development cycles, it would be best to use the testing environment that’s provided with Expo to test all components in the app to keep the response and processing times in a reasonable tolerance. This will aid in keeping the entire system responsive for the user. Its hard to say for sure at this point in this project’s life cycle what the resource utilisation will look like, so its very difficult to explain how the system will behave in relation to using appropriate resources in suitable times.

### Maintainability:

Maintainability (in other software quality models, supportability) is focused around the ability to identify and fix issues within software components. The main impacts on maintainability is code readability or complexity as well as modularisation (SQA, 2019).

The sub-characteristics of maintainability are analysability (ability to identify root cause of failure within software), changeability (effort required to change a system), stability (capability of software to reduce unexpected effects from changes to said software) and testability (effort needed to test a system change) (SQA, 2019).

As with all the main characteristics of ISO 9126, complexity plans a huge part in maintainability. The less complex a system is the easier it will be to identify cause of failures and reduce the amount of effort to change components. This is because there are less/more efficient components that are easier to understand, when a greater understanding is achieved its easier to find an issue or change a component within the system. This will also make testing more manageable for the same reasons. Stability will also improve as systems with reduced complexity try to keep their components independent as to reduce chain affects of failures passing through the system.

Taking the previous paragraph into consideration, this project’s maintainability will be easy to identify. With this projects app only consisting of a few components it makes error finding more manageable within the development cycle, because with fewer components a system has the fewer occasions of processes being passed to and from components are happening making failure less likely. The code can still be written in a way that can cause major failures, so it will have to be written to be as clear as possible so each components functionality can be easily understandable which in turn will make the failures easier to understand and solve. This will apply to analysability, changeability and stability as changes to the system by editing existing components or adding new components will have a reduced effect on the system due to its simple design. As mentioned before the tools that can be used with React Native provided by Expo will allow testing of the app. When the test environment is set up and running on either browser or mobile device, any saved changes made to the code will automatically update the test environment and display the changes or error messages if something could compile. This makes testing system changes easy on the developer.

# Conclusion, Next Steps & Reflection

## Conclusion:

The research completed show that there is a market for this type of product. The apps in this market show through their combined popularity of 11m+ downloads the level of demand that they’re fulfilling. With the number of apps in this market small, this provides more breathing room for innovation and improvements on what the current apps deliver. The research also led to the discovery of new potential functional requirements for future development cycles. Extending the usability and what the app can provide to users, along extending the overall life cycle of the project. All these points provide context and validity to this project’s existence, meaning the project is worth the resources to develop the idea.

The inclusion of the use of the framework React Native will be helpful experience. The knowledge gained from having used this relatively new technology could be a valuable skill for any potential future projects that could benefit from its implementation. It will help in diversifying skillsets which is always a positive as this can help with understanding difficult issues and the solutions required to resolve them and lead to a deeper understanding of the behaviour of different languages, like their strengths and weaknesses.

Overall this project will be successful if the information from this document is applied within its development cycles. The methodologies will keep the project approach structured and aid in the continuous development cycles with help from the MoSCoW method. The case studies into the existing market will help substantiate the use cases and functional/non-functional requirements which then feed into the MoSCoW method. The information about the React Native framework will give evidence and justification for its use. Then the quality assurance section will detail how and why the characteristics of ISO/IEC 9126 should be followed for the best project outcome. The resulting app would be completed to a high standard through this proof of concept.

## Next Steps:

The next steps for this project would be to complete the first cycle with the most important functional requirements to be completed as the end goal. With the functionality included in the first cycle, the app will be in a position for deployment after completion the cycle. With each cycle after the first, the functional requirements should be reassessed to allow for new features that could come from a changing market or updates to improve the current system. Once this has been completed MoSCoW should be applied to show what each cycle should prioritise as end goals.

## Reflection:

learning outcomes have been an important part of this project as this allows for the growth of skillsets which can then be applied to future projects. The most prevalent outcome from this project would be learning React Native.

As mentioned before, React Native is a valuable skill to have due to its versatility when creating apps for more than one platform with one codebase. If this project were to be restarted with this learning outcome being applied to it, more functionality could be implemented within the first cycle. This could happen due to the knowledge base of React Native already existing so the framework could be used with less time would be spent looking through tutorials. Meaning more work could be completed within the same project cycle. Implementing more functionality would then lead to more experience with the framework, further expanding the skillset.

Another learning outcome would be the use of Test-driven Development and the MoSCoW methodology with future project cycles. With this being the first time these methodologies are being applied it took a little time to understand their strengths and weaknesses and to ensure they are being utilised properly within this project. If this project where to be restarted with this learning outcome being applied to it, more efficient project progression would be achieved as the knowledge can be used to properly apply the strengths of these methodologies. This would help when applying to future projects when deciding what would be the best methodology to use and being able to provide evidence for the decision.

Combining all the learning outcomes makes this project a very rewarding experience. With the project allowing the growth of existing knowledge and the inception of new ideas. With the new and improved knowledge based gained from the entire process being carried forward to all future projects, regardless of scope or project team.

# 

# 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].

Antossoftwaretopics (2016) *Extreme Programming* Available from: <https://antossoftwaretopics.wordpress.com/2016/08/26/extreme-programming/> [Accessed on 31/03/2020].

Agiledata (2018) *Introduction to Test Driven Development (TDD)* Available from: <http://agiledata.org/essays/tdd.html> [Accessed on 31/03/2020].

Apm (2014) *Making Progress In An Uncertain World* Available from: <https://www.apm.org.uk/news/making-progress-in-an-uncertain-world-webinar/> [Accessed on 31/03/2020].

Stocard (2020) *Stocard* Available from: <https://stocardapp.com/en/uk> [Accessed on 10/02/2020].

Cards (2020) *Cards* Available from: <https://cards.app/> [Accessed on 10/02/2020].

Virtualcards (2020) *Virtualcards* Available from: <https://www.virtualcardsapp.com/en/> [Accessed on 10/02/2020].

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].

Kakar, K. (2017) *10 Amazing Apps That Are Built Using React Native* Available from: <https://insights.daffodilsw.com/blog/10-amazing-apps-that-are-built-using-react-native> [Accessed on 26/02/20].

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].

Hackernoon (2018) *Understanding React Native Bridge Concept* Available from: <https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8> [Accessed on 31/03/2020].

Gould, J.D., Lewis, C. (1985) *Designing for Usability: Key Principles and What Designers Think.* Communications of the ACM.

Medium (2017) *ISO 9126 Software Quality Characteristics* Available from: <https://medium.com/@leanardbuenaflor/iso-9126-software-quality-characteristics-a25a26e7d046> [Accessed on 17/03/2020].

SQA (2019) *ISO 9126 Quality Characteristics* Available from: <http://www.sqa.net/iso9126.html> [Accessed on 17/03/2020].

Ques10 (2019) *ISO 9126 Quality Characteristics* Available from: <https://www.ques10.com/p/2303/iso-9126-quality-characteristics-1/> [Accessed on 17/03/2020].

# Appendix

## Technology Selection:

Android and IOS don’t natively support apps written in JavaScript, this led to a search for technologies that would allow the development of mobile app’s whilst being able to keep JavaScript and its libraries as the code language. This is where the framework React Native was discovered.

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), with Java having been used a little but not enough to be confidant with this kind of project and the other languages that aren’t entirely familiar. This led to the discovery React Native. React Native is going to be used as this will allow the app to be implemented to both OSs’. 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 for the app.

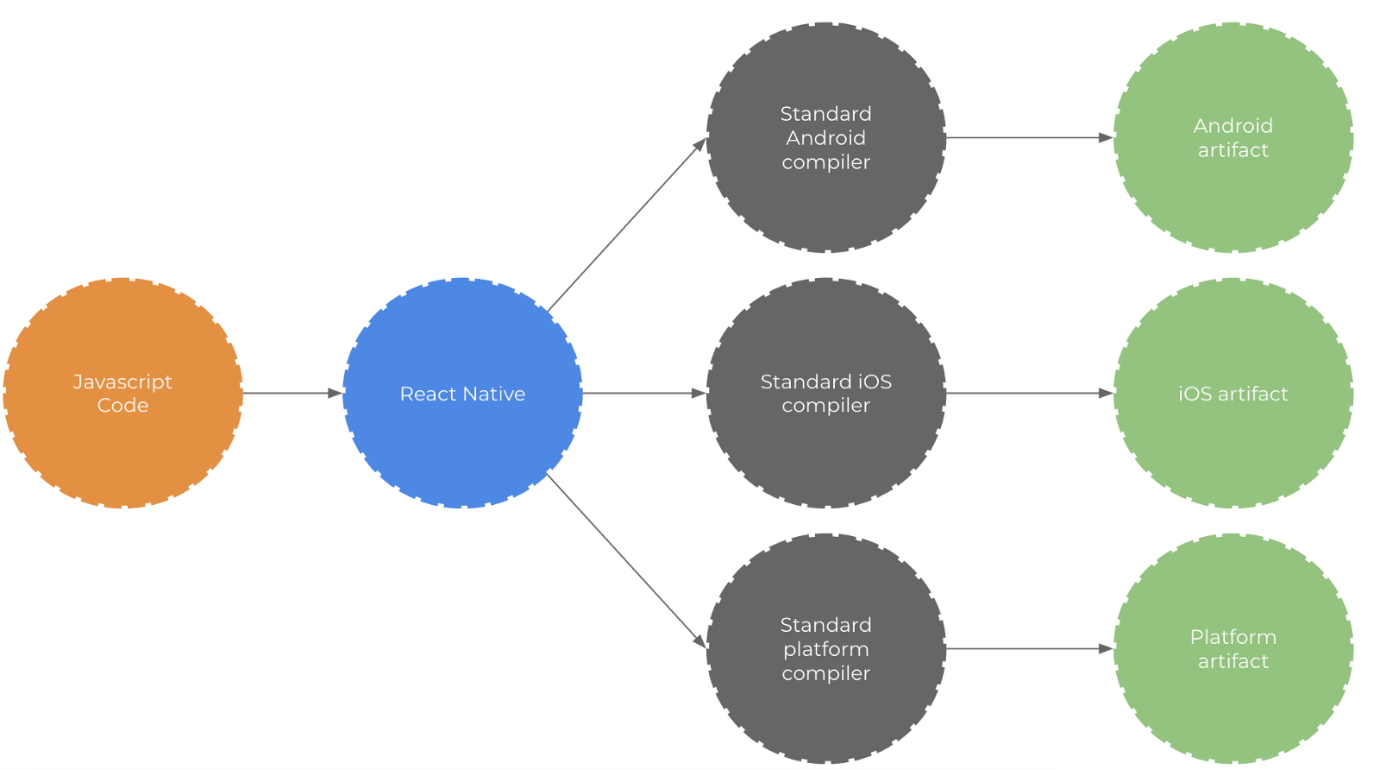
With more experience with Java then React Native it seems like Java should be the focus as there less learning is required for app development, more focus on improving skills with JavaScript and its libraries would be more desirable. JavaScript is a language that is becoming increasingly popular within the web development space which is an area of great interest, so it would be beneficial to learn more about. It would be more beneficial to learn to use React Native as this framework allows the uses of JavaScript as the primary language with the bonus of learning a popular framework.

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. Node.js will allow the developer 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, this would be better suited to use the tools provided with Expo and test the 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 that the use of Expo would be better for this app’s situation as the level of knowledge required to use the other method besides Expo would take more time to learn than is available for the end of this project, also Expo offers all that’s needed for this project. The assistance that Expo offers will allow for more time being spent in the core code functionality and design of the 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 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 the developer in a position where they can create the 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 the developer to have a greater control over the design and functionality of my app as they can constantly test and retest my program as they develop it. With these technologies in use the app won’t be in the web app format but more mobile focused.



This diagram showing the React Native framework processing JavaScript can be found at hackernoon.com (Hackernoon, 2018).