Creating a Development Process for Cross-Platform Progressive Web Applications

Christopher Johnson

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the Degree of BEng (Hons) Computing

School of Computing

April 2020

Authorship Declaration

I, Christopher Johnson, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the School's ethical guidelines

Signed:

[illegible]

Date: 01/04/2020

Matriculation no: 40275286

General Data Protection Regulation Declaration

Under the General Data Protection Regulation (GDPR) (EU) 2016/679, the University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below *one* of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

[illegible]

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Abstract

This dissertation explores the extent to which a progressive web application (PWA) can mimic a natively developed application. PWAs are an upcoming form of web application made possible by advances in web browser technologies that offer functionality such as installation, offline capabilities, notifications and more. This means that web applications, previously viewed as a slower, less functional and harder to distribute form of application have become a viable alternative to native development.

PWAs are built as websites, utilising standard web technologies HTML, CSS and JavaScript but when designing a PWA a developer must consider the user experience across different devices. Users will interact with a PWA as if it is a native application and so considerations must be made to ensure the experience is as close to that of a native application as possible.

The aim of this dissertation is to develop a process for developers to create cross-platform progressive web applications that preserve as much of the functionality of a native application as possible. The process developed is hosted via an online PWA and guides are developer through the development of a PWA starting from a standard web application, outlining the importance of certain features and providing the necessary information and resources to aid.

The URL for the live PWA was shared amongst online and university communities and some feedback was gained via a questionnaire. Though there was not much feedback received, a large limiting factor for the evaluation of development process, the site was received well with positive feedback.

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I would like to express my sincere gratitude to my project supervisor, Dr Brian Davison, for his continual support and guidance throughout this project.

I would also like to extend thanks to Edinburgh Napier University's School of Computing staff specifically Dr Simon Wells, Prof Sally Smith and Dr Babis Koniaris for helping me obtain feedback.

Thanks also to Dr Valerio Giuffrida, my second marker, for the time he's given to assess the project.

1 Introduction

1.1 Background

Progressive web applications (PWA) are an upcoming form of web application designed to provide the most native app-like user experience via the web. PWAs are slowly becoming more popular as an alternative to native applications. Many companies are making the switch from native applications to PWAs as they are easier for users to access and in some cases easier to develop whilst still providing the same software and hardware features.

PWAs are built as websites utilising HTML to structure individual pages, CSS for presentation and JavaScript to control how the application behaves when interacted with. PWAs are different to normal web pages as they can be installed onto a user's device to be used as an application with offline functionality. PWAs are supported on both desktop and mobile devices.

The user experience is different when interacting with a desktop application compared to a mobile application. Mobile applications are optimised for touch interaction on smaller displays whereas desktop applications are mostly accessed on larger displays using a mouse and keyboard. Users expect applications to feel like they were designed for the device they're being used on and this extends further than just desktop and mobile. Applications developed for different mobile operating systems are often designed in a way that complements the visual and functional design of the operating system itself.

1.2 Aims and Objectives

The aim of this project is to develop and evaluate a process for developers to create cross-platform progressive web applications that include a range of commonly-used software and hardware features.

1.3 Research Questions

- To what extent can a progressive web application preserve native functionality?
- Are progressive web applications viable alternatives to native applications?

2 Literature Review

Progressive web apps (PWA), a term coined by a Google engineer in 2015 (Russell, 2015), are web applications that behave like an installable app. They are fully functioning applications that can be installed to a user's device but are served using a web browser. Many app developers have switched to developing PWAs over Native or Cross-Platform applications as they can be installed on many devices, mobile as well as desktop, and are based on popular pre-existing web technologies.

This literature review will overview the current state of web applications, native & cross-platform mobile applications and PWAs. It will explore the development options currently available for creating applications and the benefits or drawbacks to these different development approaches.

This review will look at web and native applications separately before looking at PWAs which can be considered a combination of the two approaches to developing mobile applications. Most of the literature available discussing PWAs is located online as web pages as it is a relatively recent concept, meaning a lack of relevant academic research.

2.1 Web Application Development

Web applications are apps that do not need to be downloaded and can be accessed via a web browser. Web Applications can only be used by a user who has a network connection as it is the same as visiting any other website. A web application resides on a remote server and is delivered to the user's device over the Internet (Rouse, n.d).

2.1.1 Front-End Development

The front-end of a web application utilises HTML, CSS and JavaScript. This is rendered by a web browser to create the user interface. The visual elements of the application are best designed to scale up or down depending on the device the application is being viewed on. The interface may even change completely if viewed on a mobile device instead of desktop.

2.1.2 Back-End Development

There are many options for back-end development, the most popular frameworks amongst professional developers being ASP.NET[1], Express[2], Spring[3] and Django[4] ("Developer Survey Results," 2019). Back-end development can be written in Java, Ruby, PHP, Python and more which gives developers a range of options and the choice to pick a language they are experienced using.

A web application can be designed for mobile use specifically; however, due to the lack of persistence, installation and offline functionality they do not feel like mobile applications. This is because most applications we interact with on mobile devices were developed natively for the device or set of devices they are being used on.

2.2 Mobile Application Development

In 2018 there were 5.1 billion unique mobile users (67% of the world's population) ("The Mobile Economy", 2019). With that number expected to increase it is easy to see why mobile applications have become so popular and why it is important for businesses and organisations develop applications to reach their target audience or customers.

Mobile applications are typically installed onto the device, requiring a download from an app marketplace. Unlike a web application, mobile applications are specifically designed for mobile devices and can make use of hardware features that these devices have such as cameras, gyroscopic sensors and GPS. There are several ways to develop mobile applications which are discussed in the next few sections.

2.2.1 Native Development

Native development means an application is being built for a specific device or operating system. Natively built applications can take full advantage of the

[1] https://dotnet.microsoft.com/apps/aspnet
[2] https://expressjs.com/
[3] https://spring.io/
[4] https://www.djangoproject.com/

device’s hardware capabilities. Native app development has a variety of advantages over cross-platform development; however, if a developer wants to create a native application for different operating systems (iOS and Android, for example) two separate pieces of software must be created.

A big advantage to developing natively is that an application can be designed to have its UI and interactive elements fit the style and feel of the devices operating system. Often users expect an application to “fit” their operating system of choice to provide a more satisfying experience.

Native applications are generally faster than applications built with cross-platform tools, leveraging a devices hardware components more efficiency. In performance analysis tests native applications are faster than cross-platform applications (Grzmil et al, 2017).

A device built natively for a specific operating system or device, as opposed to being generic for a range of systems, can also be easier to develop. Developers that want to develop an application natively for more than one operating system will potentially have to know or learn two different programming languages.

2.2.1.1 Android Development

Java is the official language for developing Android applications, with Kotlin being recently introduced as a secondary official language (Sinicki, 2019). The most popular android development IDE is Android Studio[5]. Developed by Google, Android studio is the official IDE for Android Development. It supports Java, Kotlin and C++ and has development-friendly features like a Visual layout editor, Android package (APK) analyser, Intelligent code editor, Flexible build system and Real Time profilers (“Top 10 Android Development Tools ”, 2019).

Apps developed for Android are distributed with the Google Play Store which has a one-time developer fee of $25 for developers to upload an app.

[5] https://developer.android.com/studio

2.2.1.2 iOS Development

iOS applications are developed using Apple's IDE called Xcode[6]. The main languages it uses are Swift, Objective-C, C and C++ and it includes many tools and technologies that allow you to manipulate the iPhone's camera, voice interaction and more ("Getting Started with iOS App Development", n.d.).

Xcode is only available on Mac computers, meaning it is challenging to develop a native iOS application if a developer does not have access to a Mac. Apps developed for iOS are distributed through Apple's App Store. In order to upload an app to the App Store developers must pay an annual $99 fee.

2.3 Cross-Platform Application Development

It is important, for businesses especially, that an application is available on multiple platforms to reach as many of the application's target audience as possible. Cross platform development is the process of writing a single piece of software, in one language, that will then run on multiple mobile operating systems. This is better for developers that need an application to run on as many devices as possible without increasing the work to be undertaken dramatically by introducing a second application or language. There are several tools available that will produce native-like applications from one code base:

2.3.1 Flutter

According to a Stack Overflow survey conducted last year ("Developer Survey Results", 2019) Flutter[7] was the most loved cross-platform mobile app development tool, with React Native and Xamarin in second and third respectively. Flutter is Google's mobile UI framework for creating iOS and Android applications. It uses the language Dart, which was developed by Google specifically for building mobile, desktop, backend and web applications. When developing with Flutter and Dart changes made to code can be viewed within less than a second using the Hot reload tool within the Dart Virtual Machine ("Flutter Development Tools," n.d.)

[6] https://developer.apple.com/xcode/
[7] https://flutter.dev/

2.3.2 React Native

React Native[8] is Facebook's open-source mobile application framework. It uses JavaScript to create Android and iOS applications. When using React Native around 90% of code is shared between iOS and Android deployments, resulting in the application being easier to develop and maintain.

2.3.3 Xamarin

Xamarin is Microsoft's open-source platform for building iOS, Android and Windows applications using C# (Johnson and Britch, 2019). Apps built with Xamarin use native user interface controls so that they feel like they were built for the device they are being used on. Xamarin Native can be used to support up to 70% code reuse but a separate UI is required for each platform or a developer can use Xamarin Forms to allow 90-100% code reuse and a single UI for all platforms (Sharma, 2018).

Mobile applications that are developed using native or cross-platform tools are usually installed via a device's app marketplace (App Store on iOS and Play Store on Android). This gives developers and organisations an insight into their app's performance, with download and usage statistics as well as user reviews provide a form of quality assurance.

Web-based solutions have been suggested to avoid the often complicated task of cross-platform app development (Trajkovik, 2016). Web-based applications use standard web development languages (HTML, CSS and JavaScript) to create a website that delivers and app-like experience.

2.4 Progressive Web Applications

A progressive web application is a website that functions like a natively or cross-platform developed mobile app. It is a more advanced and capable than a standard web application. The application is launched via the home screen of a mobile app and is served in an instance of a browser that supports PWAs.

A website must meet certain requirements advocated by Google for it to function as a PWA ("Progressive Web App Checklist", n.d.): The application should

[8] https://reactnative.dev/

always provide some level of experience, regardless of network connection. This is done by caching files, ensuring that the latest version is always cached whenever the user uses the app with a network connection. This removes the need for users to download updates for the app like they would for a natively developed app.

A PWA must also be fast, as performance is an important factor for how a user perceives a website. The relationship between performance and user retention has been demonstrated multiple times:

- In 2015 Pinterest Engineers found a 40% decrease in perceived wait time led to a 15% increase in traffic and a 15% increase in signup rates (Meder, Antonov and Chang, 2017).
- DoubleClick by Google found 53% of mobile site visits were abandoned if a page took longer than 3 seconds to load (Wagner, n.d.).
- Krishnan and Sitaraman (2012) showed that viewers start to abandon a video if it takes more than 2 seconds to start up, with each incremental delay of 1 second resulting in a 5.8% increase in the abandonment rate.

Although PWAs are developed and presented to the user differently they must engage the user in the same way a normal app would.

2.4.1 Development

As progressive web applications are served over web browsers, they are written in HTML and JavaScript. All of the front and back end tools discussed in the web application can be used to create PWAs. Architectural features such as service workers and web app manifests that make PWAs different to a standard web application are all executed in the browser at the client side.

There are however frameworks and tools available to developers specifically for the creation of progressive web applications.

2.4.2 Service Workers

Service workers provide the technical foundation for features such as offline experiences, background syncs and push notifications to web applications (Gaunt, n.d.).

Service workers are JavaScript processes that run in the background, even when the application is closed, controlling network requests. A service worker reacts to events and intercepts network requests of application or website with server and resources. They work as a proxy between the network and the browser as shown in Figure 1.

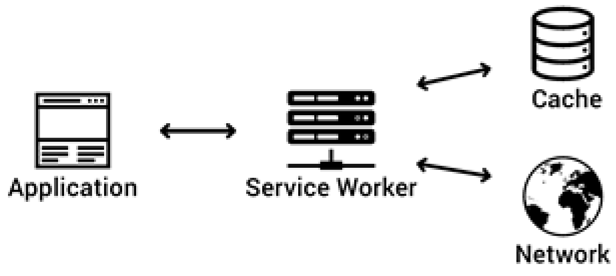


Figure 1 - How a service worker acts as a proxy. (Zlatov, 2019)

One of the most important features that service workers provide is providing an offline experience for the app. If it is detected that there is no network connection the service worker will use cached files to provide a functional offline experience. This is important as users expect applications to still provide an experience of some sort when they do not have an internet connection.

A service worker has a life cycle separate from the web page it is serving. It has three events:

- Install – When a service worker is registered within the application's JavaScript the install step is triggered and it caches the applications files that the developer has deemed necessary. The files chosen will be used to provide the offline experience to the user.
- Activation – When a service worker is activated it updates its cache if any of the files have changed, removing the old cache.

• Fetch – A service worker will handle fetch and message events when a network request or message is made from the PWA. These events contain information on resources the browser is trying to access. The way in which the service worker handles fetch events will depend on the cache strategy implemented by the developer. A selection of caching strategies are described later this section.

Once a service worker has been installed it remains idle waiting for a fetch or message event, if it is not needed it will be terminated until it is required again. Figure 2, below, is a simplified diagram of the service worker's life-cycle:

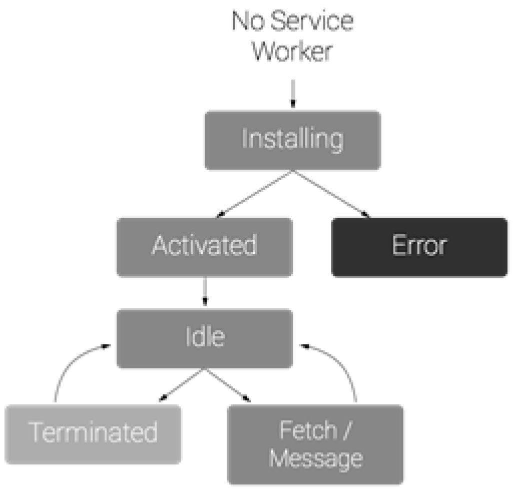


Figure 2 - Service worker lifecycle. (Gaunt, n.d)

There are a number of caching strategies that can be implemented in a service worker for serving pages. Different types of request can result in a different response from the service worker. The different options for offline caching are covered in detail by Google's offline cookbook. Some of these strategies include:

• Cache Only – The service worker will always serve this file from the cache. These will have been cached upon visiting the page. This is for the application's static resources such as images that aren't updated often.
• Network Only – For requests that would not function offline regardless of cache such as any non-GET requests.

- Cache, falling back to network – This is how the majority of requests for an offline-first application should be handled. The service worker will serve a file from the cache, falling back to the network if the file is not present in the cache.
- Network, falling back to cache – Resources that change often such as profile pictures, user posts and leader boards should be served this way. Online users will always see the most up to date versions of these resources whilst offline users will be served a cached version which is better than nothing in most cases.

2.4.3 Web App Manifest

The web app manifest is a simple JSON file (see Appendix 4 for an example) that tells the browser how the about the application and how it should behave when installed (Gaunt and Kinlan, n.d.). For a user to install the application a manifest is required, and it will be used to determine any app icons, the application's name, background colour, start URL, orientation and for configuration of the application's splash screen.

Without a web app manifest Chrome will not let a user install the application. iOS only uses some properties defined in a web app manifest. Apple's official documentation on configuring web applications show how the missing properties are to be implemented for iOS (Configuring Web Applications, 2016).

2.4.4 Background Sync

Background sync is a web API that lets you defer actions until the user has stable connectivity (Archibald, n.d.). Service workers will serve pages and content from the PWAs file when a user is offline, but server requests when offline are handled by background syncing. For example, offline viewing of posts on a blog website would be handled by the service worker but posting a new blog would be delayed by background sync until a network connection has been established.

During normal mobile phone use a user will drop internet connection throughout the day. Background sync ensures that the user does not miss out on notifications and any actions completed whilst offline will be synced with the server once a connection is acquired. This is a feature that most mobile applications have but is not usually possible with a normal website or web application.

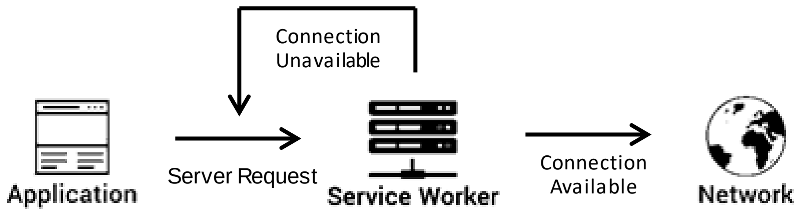


Figure 3 - A simplification of the background sync process

2.4.5 Evaluation and Audits

A progressive web application is not usually installed from a device's app marketplace; they are often advertised to users visiting the creator's website or by visiting the PWA via their web browser. This means developers and organisations do not have easy access to reviews and statistics to get feedback from users and to check that a web application can be used as a PWA. Google's audit tool, Lighthouse, can be used to check that the requirements are met. Lighthouse is an automated tool that is run from the Chrome DevTools or as a node module ("Lighthouse", n.d.).

Lighthouse runs a sequence of tests to check a website. Previous research has suggested that there are no significant limitations with PWA's compared to native Android applications with Frannson and Driaguine (2017) finding that PWAs are actually faster in some benchmark tests. Tools such as Lighthouse can measure the speed of the site and provide suggestions on how to reduce the loading time.

In order for a PWA to be available on the Google Play Score, one of the criteria that must be met is a Lighthouse score of 80/100 (Firtman, 2019). Therefore it is important when developing a PWA to keep running audits to see how the application is performing and how it can be improved.

2.4.6 Desktop Support

As well as mobile devices, PWAs can also be installed onto desktop computers, both Windows and macOS using Chrome. If Chrome detects a PWA the user will have an install option that will add a shortcut to the desktop. Desktop PWA's will work offline and can be uninstalled easily from the options in the apps window header.

2.4.7 Android Support

Android has the highest level of support for progressive web applications, supporting access to most of a mobile's hardware and software features such as cameras, GPS, push notifications and more.

Providing they meet Google's criteria, PWAs can also be distributed via the Google Play Store on Android devices. Firtman (2019) explains that this also provides extra features such as:

- Monetization of the App
- Deeper OS integration (such as app shortcuts)
- Re-installation after a hard reset or backup restore on a new phone. The application being available for the Play Store also makes it easier for users to find and install the application.
- Access to the play console, providing usage reports, pre-launch reports and review analysis.

2.4.8 iOS Support

Though Apple had the first platform to support web apps, with the first iPhone's apps being HTML5 based, the current version of iOS does not support many of the PWA features supported on Android (Love, 2019). PWA's do work on iOS and still provide an app like experience, as Safari supports service worker caching, and home screen installation.

There is no way to implement push notifications in a PWA for iOS. It is likely though that this feature will be available in the next year or so as Apple has been making good progress supporting more PWA features in the last year.

Background Sync is also not supported in iOS for PWAs. It is possible to have synchronisation using offline detection but the user will need to open the app for the sync to work. This does not execute in the background when a network connection returns, which is the main feature of the background sync API, but is better than nothing (Love, 2019).

iOS also limits the offline data for a PWA to 50mb and will delete the app's files if the app hasn't been used for around two weeks. This would create an issue if a user opened the PWA for the first time in a while whilst offline, the application wouldn't work.

One of the worst iOS PWA restrictions is the lack of persistence. Every time a PWA loses focus it is restarted. Freestone (2019) explains that it is possible to manually save the application state and restore it on load but this would have to be written as custom functionality that could impact performance. This may be worth the performance loss however as users expect that a mobile application will "remember" where they were within the app if they switch apps.

2.4.9 Existing Progressive Web Applications

2.4.9.1 Starbucks PWA

Starbucks is a coffee company that had a PWA developed that would allow customers to browse the menu, customise their orders, and add items to their carts, even if they weren't online ("Starbucks PWA", n.d.). Before the PWA they had a mobile iOS app that was 148mb in size, by converting to a PWA they reduced the size of their application by 99.84%. This resulted in a faster, more responsive application that was favoured by its users ("Starbucks PWA", n.d.)

2.4.9.2 Housing.com PWA

Housing.com is an Indian real estate company. In 2016 it received 50 million visits making it one of the most popular real estate platforms in India. In India, most users are accessing the internet via slow 2G and 3G networks, therefore the speed of their application was particularly important for maintaining user retention. After building their PWA House Go they had 38% more conversions, 40% lower bounce rate, 10% longer average session and 30% faster page load times ("Housing.com increases conversions and lowers bounce rate by 40% with new PWA", n.d.).

2.5 Conclusion

There are many options for developing mobile applications. There are many cases where developing an application natively or using a cross-platform development tool is appropriate, however progressive web applications are becoming more viable as mobile operating systems offer more support for them.

PWAs are easier to develop for developers who have only worked with web-based applications and offer many of the hardware features previously only available to native apps. Currently PWAs are more functional on Android devices as Apple has been slow to incorporate support for PWAs on iOS but the level of support offered on iOS should increase in the coming years.

3 Methodology

The purpose of this project is to create a process for web developers that guides a user through the various methods involved in creating a progressive web application. Though PWAs are cross-platform this process will mostly focus on the preservation of native mobile application functionality. This is important, as discussed in the literature review, as it offers developers the information required to build web applications that are more accessible with better usability for mobile users.

The process that will be developed aims to be as simple and clearly presented as possible, walking through the key features needed in a web application for it to be considered and installed as a PWA. It will also provide alternatives to OS restricted features, especially for iOS devices. A developer who has followed the process will have a good understanding of how a PWA works, what makes them special and give them a good framework from which they can continue with their own development.

How the process was created will be described in this section, as well as a discussion on the methods used to evaluate and test the process.

3.1 Frameworks and Technologies Used

A web application is transformed into a PWA using mostly JavaScript and therefore developers are not required to use a specific framework. The framework chosen for this project and to be used for presenting the process is Flask[9]. Flask is a python based web development framework. Flask works by providing a micro-framework (that does not require additional libraries) that developers can then add extensions to in order to implement additional features. This framework provides a number of advantages:

[9] http://flask.palletsprojects.com

- Easy to set up – Flask is well documented and lightweight. It includes an integrated development server which enables developers to easily set up a running web application.
- Extendable – There is a wide range of extensions available for Flask. This is useful for developers as they can easily expand upon a simple PWA created following the development process that will be created during this project.
- Templating – Flask comes with a template engine installed, Jinja2. This is a powerful tool for developers to simplify document creation and keep pages consistent.
- Debugging – The included development server and debugging functionalities are incredibly useful for troubleshooting and solving problems related to the application. This is useful for developers that are less experienced for web development who are interested in the created development process.

There are many detailed Flask tutorials and development processes available online however; the documentation available for PWA development with Flask is extremely limited. Flask is a very popular framework so the development process being created will fill a gap in knowledge available to developers and hopefully be helpful to numerous individuals.

3.2 Experimental Development

Before beginning the development process development, an initial phase of experimentation was needed to gain a deeper understanding of how PWAs work from a technical perspective. This included creating simple Flask applications and modifying them to be functional and installable PWAs.

This phase, combined with research conducted for this project, was useful to identify the key components of the development process and what information developers would need to be given to implement them.

3.3 Development Process Components

The development process was broken down into key components. These sections will be presented to developers in the order that they appear here and have code snippets to supplement development. These components are essential for achieving the aims of the project as described below.

3.3.1 Introduction

This part of the process will explain to users the purpose of the process and the tools that will be used to follow it. It will also cover any prerequisite knowledge anyone looking to follow the process will need with links to other recommended available tutorials and documentation that they will need if this knowledge is not already present.

3.3.2 Initial Set Up

Users will need to either have a pre-existing Flask application to modify or create a simple Flask application. This section covers required python installs and the folder structure that the process will assume is being used.

Simple additions that are required for a web browser to recognise an application as a PWA will also be included here. This includes Meta tags required in HTML and suggestions on how to force HTTPS using Flask. These steps are too simple to be given their own sections as they do not require a lot of explaining and are easy to implement.

3.3.3 Web App Manifest

The web app manifest will be the first significant stage of the development process. This is to be implemented first as it gives users a good idea as to how the PWA will be used and installed as a native-like application. The web app manifest is a simple JSON file defining different attributes.

Although the web app manifest is easy to implement, this section will include detailed descriptions of the different attributes available as well as how to provide iOS devices with the same information.

To aid users, a sample web app manifest will also be provided to show its structure and a set of sample icons of all required sizes will be offered as a download so that developers can quickly implement one if they have not yet created icons for their web application.

3.3.4 Service Workers

In this section of the process, service workers and their significance will be described. It will also outline different caching strategies that can be utilised by a service worker and their use cases. It is important that PWA developers have a thorough understanding of service workers and how they are used to provide installability and offline functionality.

For creating an application that passes Google's Lighthouse PWA audit a developer would only need to follow the process this far. However, in order to fulfil the aims of the project it is important that native mobile app functionality is implemented.

As well as code snippets this section will also include downloads for relevant JavaScript files as examples to aid developers.

3.3.5 Push Notifications

Most mobile applications serve push notifications to users. Notifications are important to engage users and to supply information that the user may rely on the application to receive. This section will run developers through implementing push notifications. This is also a suitable section to introduce browser permissions and their importance for PWAs with native functionality as well as advice on utilising a database with Flask to store user preference information.

iOS does not currently support push notifications for PWAs but alternatives will be offered by this process such as email notifications. Instructions for setting this up with Flask will be included.

As an example, the process PWA will offer interactive examples with which developers can see what a user will experience for granting the browser permissions to send notifications.

3.3.6 Background Sync

For some mobile applications, especially applications that have a messaging feature, background sync ensures that the impact a poor connection has on user experience is kept to a minimum. This section will describe the use of background syncing and demonstrate how it works within a PWA.

3.3.7 Additional Functionality

At this point in the process developers have an excellent starting point to continue with development. This section will introduce additional functionality that can be implemented for both Android and iOS. It will not be as in-depth as the other sections but will have a lot more suggested resources and tutorials.

3.3.8 Testing

This section will show how developers can test their PWAs. It will include Google's Lighthouse audit as an example, to ensure their PWA is recognised and available as a PWA, as well as other tests that can be carried out for various functions.

3.3.9 Troubleshooting

Throughout the process there will be troubleshooting sub sections that offer advice on common issues. These problems are those experienced during the experimental and process development as well as problems frequently found online. The solutions will include code snippets and external references.

3.4 Process Presentation

In all stages of development, the process will be presented using a PWA built to provide a clear view of the process with each section readily available. This will include code samples, diagrams, descriptions and resource downloads to aid developer understanding. The website will also be used to demonstrate features described within the process such as app installability and provide developers for the source code of the presentation PWA.

The site will be built using the process it was built to present, using Flask and hosted using the free tier available from Amazon Web Services. Having the process presented this way is important as it allows anyone with an internet connection to have access which makes testing and generating feedback much easier.

3.5 Initial Process Development

For the development of the presentation PWA and development process an iterative waterfall approach was taken. In order to make the development process as informative and useful as possible it was important to receive feedback before the process was completed so that informed revisions could be made before completion of the project.

A waterfall development methodology splits the development into five stages; these can be viewed below in Figure 4. This allowed for better time management to ensure the development was completed on time; the implementation of these stages for the project can be viewed in the Gantt chart in Appendix 3.

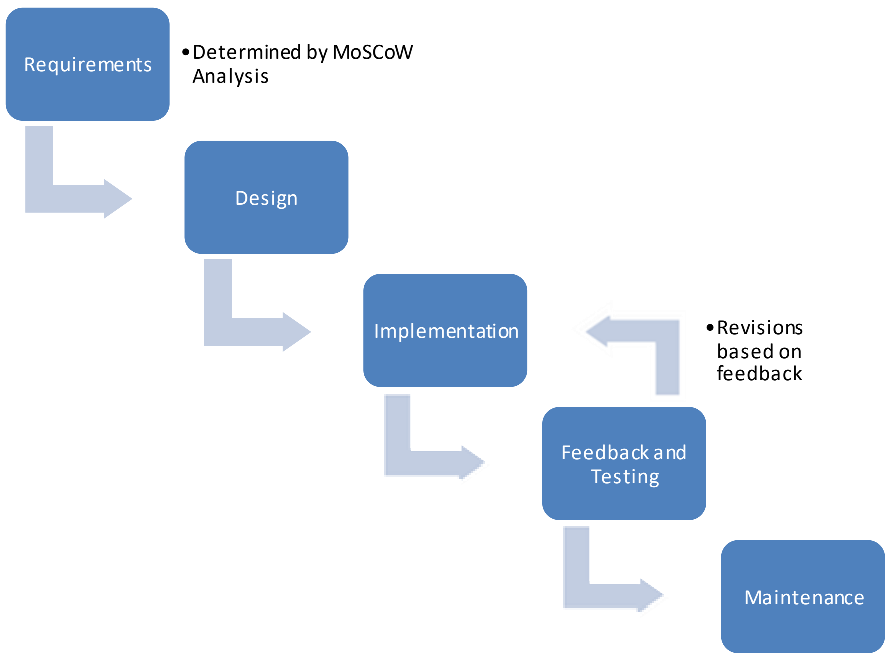


Figure 4 - Waterfall development methodology

The requirements were identified based on knowledge gained from research and the experimental phase. The requirements were analysed using the MoSCoW method to prioritise the processes features, this full analysis can be viewed in Table 1 below.

Must Have	- The necessary information for users to develop a PWA as identified in key components of the development process - A presentation website including: o Code snippets and examples o Resource downloads o Testing and troubleshooting advice
Should Have	- The information for users to preserve native functionality o Hardware support for cameras, Geolocation etc. - A solid user experience for desktop and mobile users on the presentation website - A way for users to give feedback via the presentation website
Could Have	- A fully functional PWA as the presentation website - Example PWA for the development process stage progression - Visualisation of the feedback data within the website - A section on mobile design considerations - A shorthand version of the process for developers to reference
Won't Have	- External example application for demonstrations

Table 1 - MoSCoW analysis of development process features.

Waterfall development also accommodated for the need for a revised development phases based on feedback. The implementation and feedback/testing stages were iterated, with the new features being implemented and once again evaluated after feedback was received.

3.6 Initial Process Evaluation

Once the initial process development was completed, it was shared online and to classes of students from modules relevant to the dissertation (mobile application development and web development). A variety of sites and online communities were chosen to share the process on in order to receive feedback from users of different skill levels and experience.

It was important to try and get feedback from the type of users that the process is aimed at, developers with experience in Flask who are looking at learning to develop PWAs. They will be able to provide feedback on how easy the process was to follow, what information was useful and what information needed to be included for the next iteration of the process. The process was shared, with requests for feedback on the following sites with the goal of finding these users:

- https://www.reddit.com/r/flask/
- https://www.reddit.com/r/webdev
- https://www.freecodecamp.org/forum/
- https://www.webdeveloper.com/
- https://www.webdesignerforum.co.uk/

Feedback from users that already have experience developing PWAs is also valuable as they can offer advice for improvements and suggestions for additional information. To find users with this perspective the process was shared on the following sites:

- https://www.reddit.com/r/PWA/

To collect feedback, a page was built on the presentation site that includes a questionnaire for users to give feedback easily. The questionnaire includes questions to find out the level of experience a user has with PWA development and their intentions for visiting the site. There have been many instances of developers using questionnaires to receive feedback about a website and so for this project questions related to the presentation site were collected and adapted from the Website Evaluation Questionnaire developed by Elling, Lentz and Jong (2007) (Appendix 5).

As the users giving feedback are under no obligation to do so, an alternative to the questionnaire will be supplied to just give general comments in text form. This means users that have feedback but do not want to spend the time filling in a questionnaire can do so to the extent they want.

The feedback received from this was collected in a database for analysis and used to develop a revised version of the process. Any feedback received during the development of the revised process will also be taken into consideration where applicable.

3.7 Revised Process Development

User feedback was collected and analysed to determine what changes and improvements needed to be made. This then set off another phase of development, shorter than the previous, in which the development process and presentation site were improved.

3.8 Revised Process Evaluation

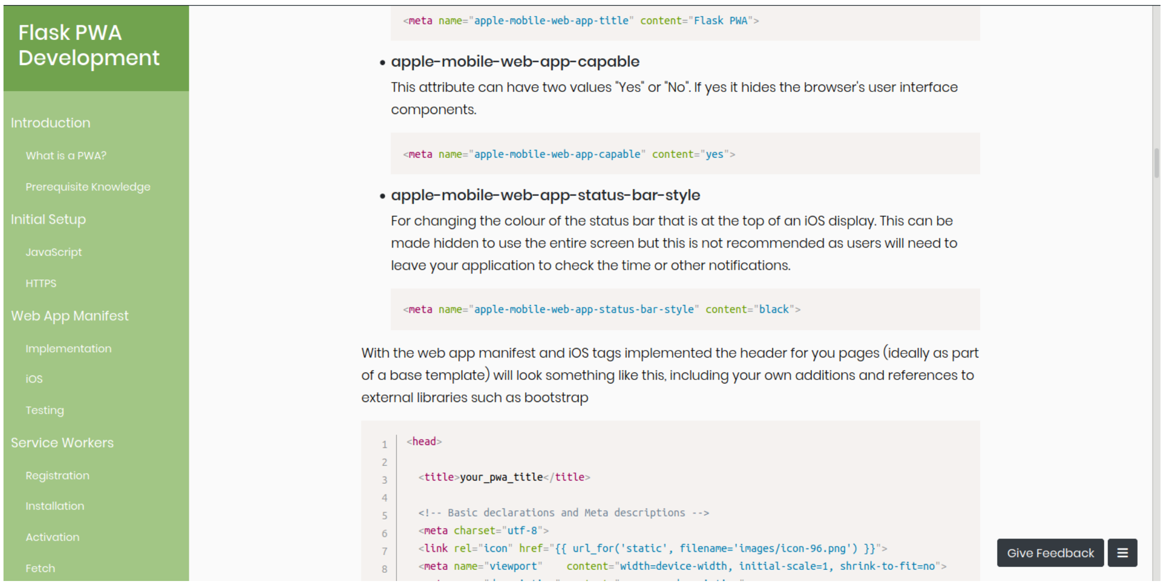
In order to test the final iteration of the development process one to one sessions were held in which students and developers were asked to follow the process whilst under observation. This testing was crucial as it provides feedback from the target audience and can be used as a measure of how successful the project was.

These sessions were useful for obtaining information on how the process is perceived by its target audience and offer a broader range of communication than the more impersonal approach used for the initial process testing.

The goal of the development sessions is to receive feedback from developers with different levels of experience. Therefore testers would ideally have varying levels of web development, Flask and PWA experience and would be encouraged to use external resources whenever necessary as if they were following the process themselves without supervision.

4 Results and Discussion

The development process was created and is available at www.flaskpwa.com. The website is a functioning progressive web app that can be installed onto desktop or mobile devices. The process is displayed as text, diagrams, code snippets and images on the site's main page with a navigation sidebar to easily locate different sections of the process.



```
<meta name="apple-mobile-web-app-title" content="Flask PWA">
```

- apple-mobile-web-app-capable

  This attribute can have two values "Yes" or "No". If yes it hides the browser's user interface components.

```
<meta name="apple-mobile-web-app-capable" content="yes">
```

- apple-mobile-web-app-status-bar-style

  For changing the colour of the status bar that is at the top of an iOS display. This can be made hidden to use the entire screen but this is not recommended as users will need to leave your application to check the time or other notifications.

```
<meta name="apple-mobile-web-app-status-bar-style" content="black">
```

With the web app manifest and iOS tags implemented the header for you pages (ideally as part of a base template) will look something like this, including your own additions and references to external libraries such as bootstrap

```
<head>

  <title>your_pwa_title</title>

  <!-- Basic declarations and Meta descriptions -->
  <meta charset="utf-8">
  <link rel="icon" href="{{ url_for('static', filename='images/icon-96.png') }}">
  <meta name="viewport"    content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

Figure 5 - Screenshot of www.flaskpwa.com

For gathering data there is a button prompting users to give feedback that is always displayed on the page as users are browsing through the development process. This takes users to the feedback questionnaire on another page. A Google analytics tag was included within the websites HTML which allowed for the number of visitors to be tracked as well as the platform the site was accessed on and where they come from. The tracking implemented by the analytics tag is blocked by users running any advertisement blocking software on their browser which makes the Google analytics data incomplete but good for giving a rough idea of site activity.

This site encourages users to interact with its' PWA design. The website itself serves as an example of what developers following the development process will be creating. In the installation section the installation of a PWA can be easily demonstrated by having users download the website to their device as a PWA. Features such as push notifications and hardware access are also demonstrated this way.

Each major component as described in the methodology is displayed as a section which is then split into sub sections. This allows for easier navigation as the page contains a lot of content and gives a potential user a good overview of what they will be undertaking should they follow the process.

4.1 Initial Process

The initial development process was created using the waterfall software development process which worked well for this project. With time restrictions it allowed for the process and website development to be split into manageable sections.

By the end of the initial process development the site contained sections for each of the components outlined in the methodology as well as a functional feedback page.

4.1.1 Questionnaire Feedback

The live data for the questionnaire, which will include any feedback given after the revised process was created, is available at www.flaskpwa.com/feedback/data. The data for feedback given for the initial process will be discussed in this section and is available to view in Appendix 6.

Unfortunately, despite efforts to share the website with online and university communities, there were not many responses to the questionnaire. Out of the 206 users Google analytics showed to have visited the site a total of eight users gave feedback via the website and only two of those leaving textual feedback. Probably due to having a specific target audience, this meant that the sample size was too small to properly analyse the data. There are few things that can be drawn from the responses given which will be discussed next.

Overall the feedback for questions about the website and development process was positive, with 80.0% of question responses being positive and 46.7% of responses selecting the “Strongly Agree” option. This is a good indication that the website offered a good user experience that did not incline users to leave negative feedback.

There were only two negative responses given, both in response to “I am likely to revisit this website”. It is not possible to tell if this is because the website was not perceived well or if they are users that do not and have no plans to develop PWAs. Both of these users did, however, state that they had no experience creating PWAs and limited web development experience.

4.1.2 Forum and Community Feedback

Some users did not provide any feedback by the site but left comments on the posts made sharing the website to various forums and web communities. Two of the five users who gave feedback this way expressed a dislike to the mobile experience, the long page with poor mobile navigation meant they found it tedious to scroll so far down.

There was no negative feedback regarding the development process content with four of the users praising the site. The full set of responses from forums and communities can be found in Appendix 7.

4.2 Revised Process

Following the feedback received through hosting the initial development process a short development phase was undertaken to make improvements and changes based on user feedback.

One of the main focuses for this revision was to improve the mobile experience. This involved changing the websites layout slightly on smaller screens and implementing the navigation sidebar in a different way so that it can be toggled on smaller devices, offering easier navigation. As this dissertation is focused on cross-platform functionality the sidebar was tweaked to close when the user taps anywhere other than the sidebar, this consideration was made as this is the behaviour users expect from natively developed applications. The desktop and mobile view of the site can be viewed in Appendix 8.

Images and GIF content was also changed to be optimised for smaller screens and not take up too much room on a desktop device following user feedback. Some of the suggestions made, such as including video tutorials or having an example PWA for demonstration purposes, were deemed too time consuming to be implemented for this project but have been noted as options for future development.

Unfortunately, running face-to-face development sessions as a form of review for the revised development process, as planned in the methodology, was not possible at the time due to the on-going global pandemic at the time of writing. Instead, the website was shared with two student volunteers each with previous web development experience. The two students were asked to attempt to follow the development process, whilst accessing the website from both desktop and mobile browsers as well as an installed application. Neither student ran into significant problems when following the process, praising the user experience.

4.3 Critical Analysis

4.3.1 Development

The iterative waterfall methodology used for development of the process was proved suitable for this project, had the project taken place across a longer time period or involved a team of developers a more complex methodology such as Agile may have been used.

4.3.2 Questionnaire and Feedback

One of the major problems encountered during this dissertation was a lack of feedback. There are too many possible reasons that this could have occurred but there are a number of ways that the response count could have potentially been improved. A shorter questionnaire may have resulted in more responses as users are more inclined to give feedback if it takes up less time, a review of response burden and questionnaire length by Rolstad et al (2011) found that response rates were lower for longer questionnaires.

Had more feedback been received proper analysis of the feedback data could have taken to place to reveal how users of different experience and backgrounds perceived the site.

4.4 Conclusions

The aim of the project was to create a development process for creating progressive web apps that include as many hardware and software features as possible. The developed process created does not cover many of the hardware features available as there are so many to possibly cover. Whatwebcando[10], a site that tracks device integration for web browsers, contains a full list of possible features available.

The features covered by the development process can be split into two categories, essential and non-essential. The essential features include notifications, offline mode, fullscreen view, home screen installation and background sync. These are the features most important for creating a native-like experience as most native applications take advantage of these and users will associate.

The non-essential features, Geolocation and camera access were added as examples for adding additional functionality to a PWA. Location and camera access are commonly used hardware features, including every feature would have made the development process too long and much of the content would be irrelevant to many developers who do not require additional functionality. These two features were implemented however as they give a user a good introduction to how hardware access works within a PWA making any future development to include the features that weren't included easier to understand.

[10] https://whatwebcando.today/

4.5 Suggestions for Future Work and Research

The presentation website will remain live and development of the process, to include any remaining requirements from the MoSCoW analysis, will continue after this projects completion. Users have already indicated that the website may be useful for them and therefore it's important to keep it up to date with relevant information and to improve it where possible. Any feedback received in the future will closely monitored and taken into consideration for future development.

Through the dissertation and following user feedback, it was found that an applications graphical user interface is also important for preserving native functionality. A PWA may have as many features as a native application but if the users perceive the application as a website and not a mobile app it will have significant impact on the user experience, as shown by feedback from forum users. It is suggested that more thorough research is to be undertaken regarding GUI and user experience for progressive web applications.

4.6 Self-Appraisal

I had no prior knowledge or experience working with PWAs before starting this project but through research and experimentation I am now confident in discussion and development within this area of study. This has been an exciting topic to study as it is becoming more relevant as the popularity of PWAs increases.

I think that the project ran fairly smoothly, managing my time with different phases of the waterfall methodology being incorporated into the Gantt chart (Appendix 3). The weekly diary sheets were important for breaking down the project into smaller goals each week. This resulted in successfully completing the project on time with no major issues or missing features.

If a GitHub[11] repository had been properly used during the development of the presentation PWA it could have been used another resource to offer developers on the site as demonstration code. Using a repository effectively could have also

[11] https://github.com/

been useful during development, allowing for past changes to be viewed to assist with debugging, but I do not think that and absence of this had a significant effect on development.

I hope that the development process I have created is a valuable resource for developers and I look forward to using the knowledge gained during this project for future PWA projects.

References

Archiblad, J. (n.d.). Introducing Background Sync. Retrieved October 11, 2019, from https://developers.google.com/web/updates/2015/12/background-sync

Configuring Web Applications. (2016). Retrieved December 4, 2019, from https://developer.apple.com/library/archive/documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html

Developer Survey Results 2019. (2019). Retrieved October 22, 2019, from https://insights.stackoverflow.com/survey/2019

Elling, S., Lentz, L., & De Jong, M. (2007). Website evaluation questionnaire: Development of a research-based tool for evaluating informational websites. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4656 LNCS*, 293–304. https://doi.org/10.1007/978-3-540-74444-3_25

Firtman, M. (2019). Google Play Store now open for Progressive Web Apps. Retrieved October 17, 2019, from https://medium.com/@firt/google-play-store-now-open-for-progressive-web-apps-ec6f3c6ff3cc

Flutter Development Tools - Hot reload. (n.d.). Retrieved November 1, 2019, from https://flutter.dev/docs/development/tools/hot-reload

Frannson, R., & Driaguine, A. (2017). Comparing Progressive Web Applications with Native Android Applications. Linnaeus University, Faculty of Technology.

Freestone, J. (2019). The current state of progressive web apps. Retrieved October 11, 2019, from https://www.browserlondon.com/blog/2019/04/15/current-state-progressive-web-app-pwa/

Gaunt, M. (n.d.). Service Workers: an Introduction. Retrieved October 25, 2019, from https://developers.google.com/web/fundamentals/primers/service-workers

Gaunt, M., & Kinlan, P. (n.d.). The Web App Manifest. Retrieved October 10, 2019, from https://developers.google.com/web/fundamentals/web-app-manifest/

Getting Started with iOS App Development. (n.d.). Retrieved November 30, 2019, from AWS website: https://aws.amazon.com/mobile/mobile-application-development/native/ios/

Grzmil, Paweł & Skublewska-Paszkowska, Maria & Łukasik, Edyta & Smołka, Jakub. (2017). PERFORMANCE ANALYSIS OF NATIVE AND CROSS-PLATFORM MOBILE APPLICATIONS. Informatics Control Measurement in Economy and Environment Protection. 7. 50-53. 10.5604/01.3001.0010.4838.

Housing.com increases conversions and lowers bounce rate by 40% with new PWA. (n.d.). Retrieved October 10, 2019, from https://developers.google.com/web/showcase/2016/housing

Johnson, J., & Britch, D. (2019). What is Xamarin? Retrieved November 5, 2019, from https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin

Krishnan, S. S., & Sitaraman, R. K. (2012). Video stream quality impacts viewer behavior. 211. https://doi.org/10.1145/2398776.2398799

Lighthouse. (n.d.). Retrieved October 10, 2019, from https://developers.google.com/web/tools/lighthouse

Love, C. (2019). Progressive Web Applications (PWA) on iOS Provide a Rich Channel to Reach Customers Despite the Platform Limitations. Retrieved November 5, 2019, from https://love2dev.com/pwa/ios/

Meder, S., Antonov, V., & Chang, J. (2017). Driving user growth with performance improvements. Retrieved October 22, 2019, from https://medium.com/pinterest-engineering/driving-user-growth-with-performance-improvements-cfc50dafadd7

Offline Cookbook. (n.d.) Retrieved January 14, 2020, from https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook

Progressive Web App Checklist. (n.d.). Retrieved October 23, 2019, from https://developers.google.com/web/progressive-web-apps/checklist

Russell, A. (2015). Progressive Web Apps: Escaping Tabs Without Losing Our Soul. Retrieved October 25, 2019, from https://medium.com/@slightlylate/progressive-apps-escaping-tabs-without-losing-our-soul-3b93a8561955

Rolstad, S., Adler, J., & Rydén, A. (2011). Response burden and questionnaire length: Is shorter better? A review and meta-analysis. *Value in Health*. https://doi.org/10.1016/j.jval.2011.06.003

Rouse, M. (n.d.). Web Application Development. Retrieved October 24, 2019, from https://searchcloudcomputing.techtarget.com/definition/web-application-development

Sharma, A. (2018). Why Use Xamarin for Mobile App Development. Retrieved November 9, 2019, from https://dzone.com/articles/why-use-xamarin-for-mobile-app-development

Sinicki, A. (2019). I want to develop Android Apps – What languages should I Learn. Retrieved October 15, 2019, from Android Authority website: http://www.androidauthority.com/wantdevelopandroidappslanguageslearn391008/

Starbucks PWA - Universally Accessible Ordering for Established and Emerging Markets. (n.d.). Retrieved October 23, 2019, from https://formidable.com/work/starbucks-progressive-web-app/

The Mobile Economy (2019). Retrieved from https://www.gsma.com/r/mobileeconomy/

Top 10 Android Development Tools. (2018). Retrieved October 20, 2019, from Amar InfoTech website: https://www.amarinfotech.com/best-android-development-tools-2018.html

Trajkovik, V. (2016). Ict innovations.

Wagner, J. (n.d.). Why Performance Matters. Retrieved October 23, 2019, from https://developers.google.com/web/fundamentals/performance/why-performance-matters

Zlatkov, A. (2019). How JavaScript works: Service Workers, their lifecycle and use cases. Retrieved November 6, 2019, from https://blog.sessionstack.com/how-javascript-works-service-workers-their-life-cycle-and-use-cases-52b19ad98b58

Appendix 1 Project Overview

There will be many discrepancies between the dissertation and the initial project overview, this is because the project was reviewed and changed after the submission of this document.

Initial Project Overview

SOC10101 Honours Project (40 Credits)

Title of Project: Developing an Outdoor Activity Social Network Application

<u>Overview of Project Content and Milestones</u>

The aim of the project is to develop a cross-platform app in the form of a progressive web application (PWA). I will create a desktop web version of the app that will run as a social network and a mobile version of the app which will grant access to some of the desktop features as well as providing utilities and information that will be helpful for users partaking in outdoor activities such as hiking and camping. The application will aim to make outdoor activities more social and to serve as a tool to make outdoor trips easier and safer.
Within the desktop site users will be able to share hiking routes, photos, points of interest and use maps to view, comment on and share other user's submissions. The mobile application will be able to access some of the desktop features but will also be an offline tool for activities with first aid information, gear checklists and more. The app will behave as a regular android/iOS application although it will be running using HTML/CSS/JavaScript.

The Main Deliverable(s):

The most important element that will be delivered at the end of the project will be a functional version of the application itself, along with thorough documentation.

The Target Audience for the Deliverable(s):

The application's users are likely to be people who enjoy outdoor activities, such as hiking and camping, who also have an interest in technology and the internet. The project will also serve as a resource for other developers/researchers as an example of PWAs being used to minimise development time for a cross-platform application.

The Work to be Undertaken:

I need to decide on additional features and begin to plan how they will be implemented. I will also research PWAs further and discover which tools I can use for development.
I need to design a suitable architecture for the software and prepare a server/database before I can start development. I will also need to plan out

development, allocating time to different sections or tasks that need to be completed. Code for the project will be managed using GitHub.
Throughout development I will need to continually test code and I can use multiple devices as well as peers to test the application. I can also use audit tools such as Google's Lighthouse to measure performance and ensure the application is at a professional standard.
Once the application is completed, I will need to produce documentation and evaluate the quality of the software I've produced.

Additional Information / Knowledge Required:

The bulk of this project will require me to develop using HTML, CSS and JavaScript. During my studies I have been introduced to these languages and have become familiar with using them to create simple websites. The nature of PWAs means I will not have to do any separate native mobile app development. Creating websites for mobile use is new to me but a lot of the user interface skills acquired during previous mobile app development projects will be useful here.
I will need to become more capable with these languages and be prepared to learn how to use new tools to aid development.

Information Sources that Provide a Context for the Project:

Websites and References:

- https://developers.google.com/web/progressive-web-apps
- Kvist, J., & Mathiasson, P. (2019). Progressive Web Apps and other mobile developing techniques
- https://developers.google.com/web/tools/lighthouse
- Grill-spector, F. A. K., Gross, J., Knutson, B., & Mcclure, S. (2017). Comparing Progressive Web Applications with Native Android Applications. *Linnaeus University, Faculty of Technology*, *1*, 59.

Prior Art:

- Twitter (https://arstechnica.com/gadgets/2018/09/progressive-web-apps-moving-mainstream-as-twitter-makes-its-mobile-site-the-main-one/)
- Uber (https://www.windowscentral.com/uber-talks-about-progressive-web-app-why-desktop-important)

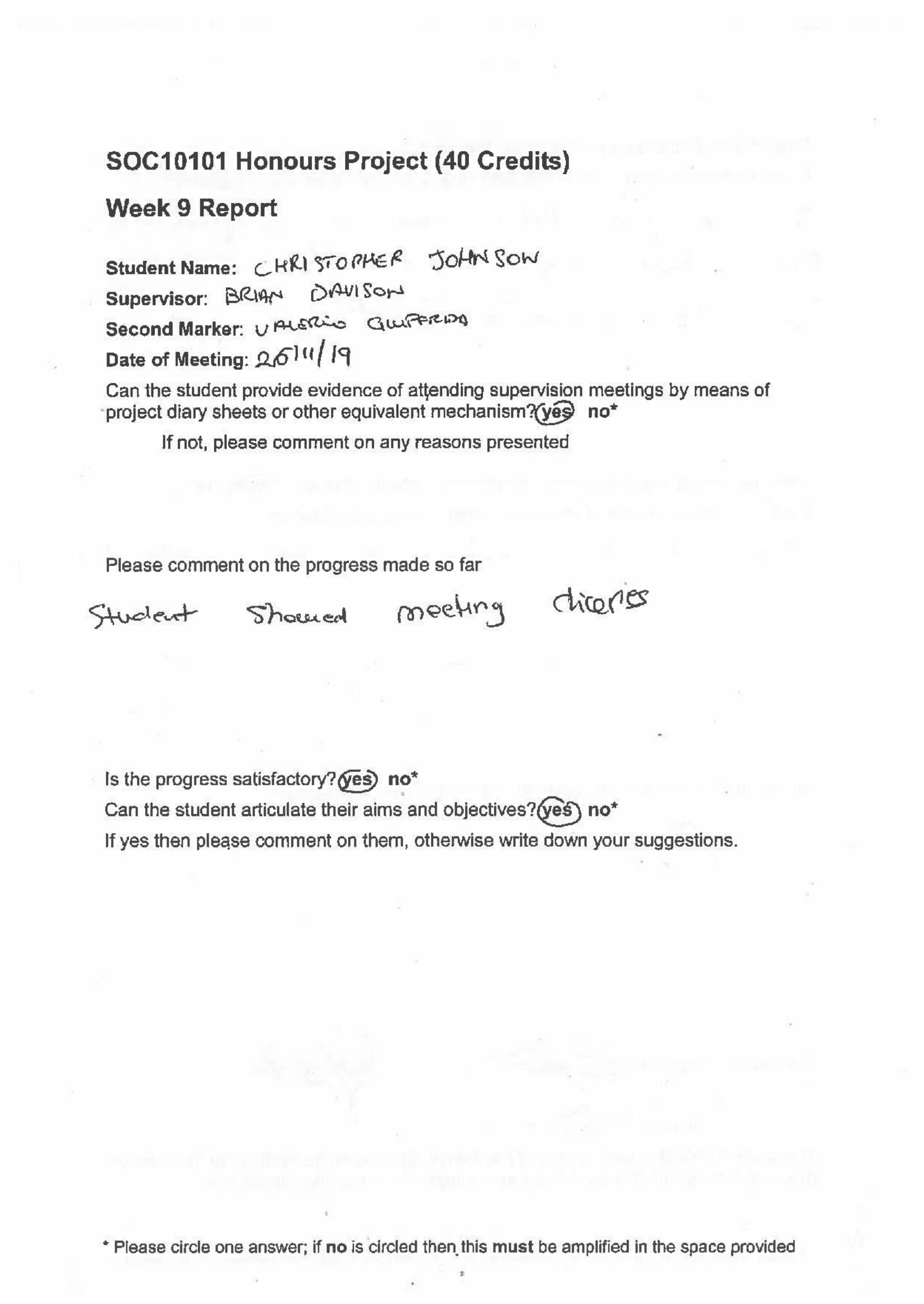
The Importance of the Project:

The project will illustrate the advantages of using PWAs and will serve as a demonstration of my skills as a programmer. The application will hopefully be a way to create a small community of individuals sharing a common interest of the outdoors, aiding trips making them safer and easier.

The Key Challenge(s) to be Overcome:

This is a large project compared to any previous projects I have undertaken, so time needs to be managed effectively. This will also be the first time I am developing a PWA.

Appendix 2 Second Formal Review Output



SOC10101 Honours Project (40 Credits)

Week 9 Report

Student Name: CHRISTOPHER JOHNSOW

Supervisor: BRIAN DAVISON

Second Marker: VALERIO GIUFFRIDA

Date of Meeting: 26/11/19

Can the student provide evidence of attending supervision meetings by means of project diary sheets or other equivalent mechanism? [✓ **yes**] **no***

If not, please comment on any reasons presented

Please comment on the progress made so far

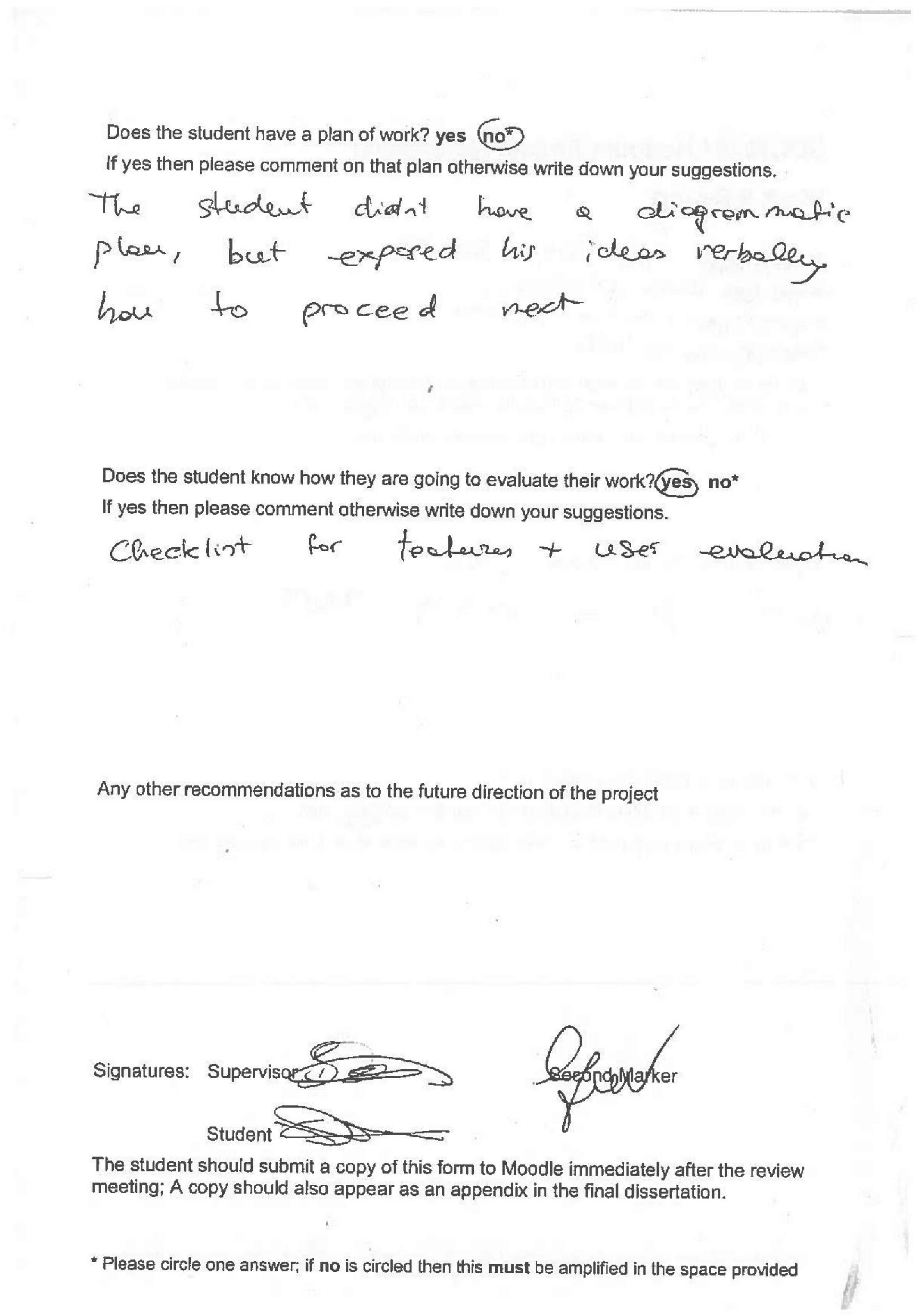
Student showed meeting diaries

Is the progress satisfactory? [✓ **yes**] **no***

Can the student articulate their aims and objectives? [✓ **yes**] **no***

If yes then please comment on them, otherwise write down your suggestions.

* Please circle one answer; if **no** is circled then this **must** be amplified in the space provided



Does the student have a plan of work? **yes** [✓ **no***]

If yes then please comment on that plan otherwise write down your suggestions.

The student didn't have a diagrammatic plan, but expored his ideas verbally how to proceed next

Does the student know how they are going to evaluate their work? [✓ **yes**] **no***

If yes then please comment otherwise write down your suggestions.

Checklist for features + user evaluation

Any other recommendations as to the future direction of the project

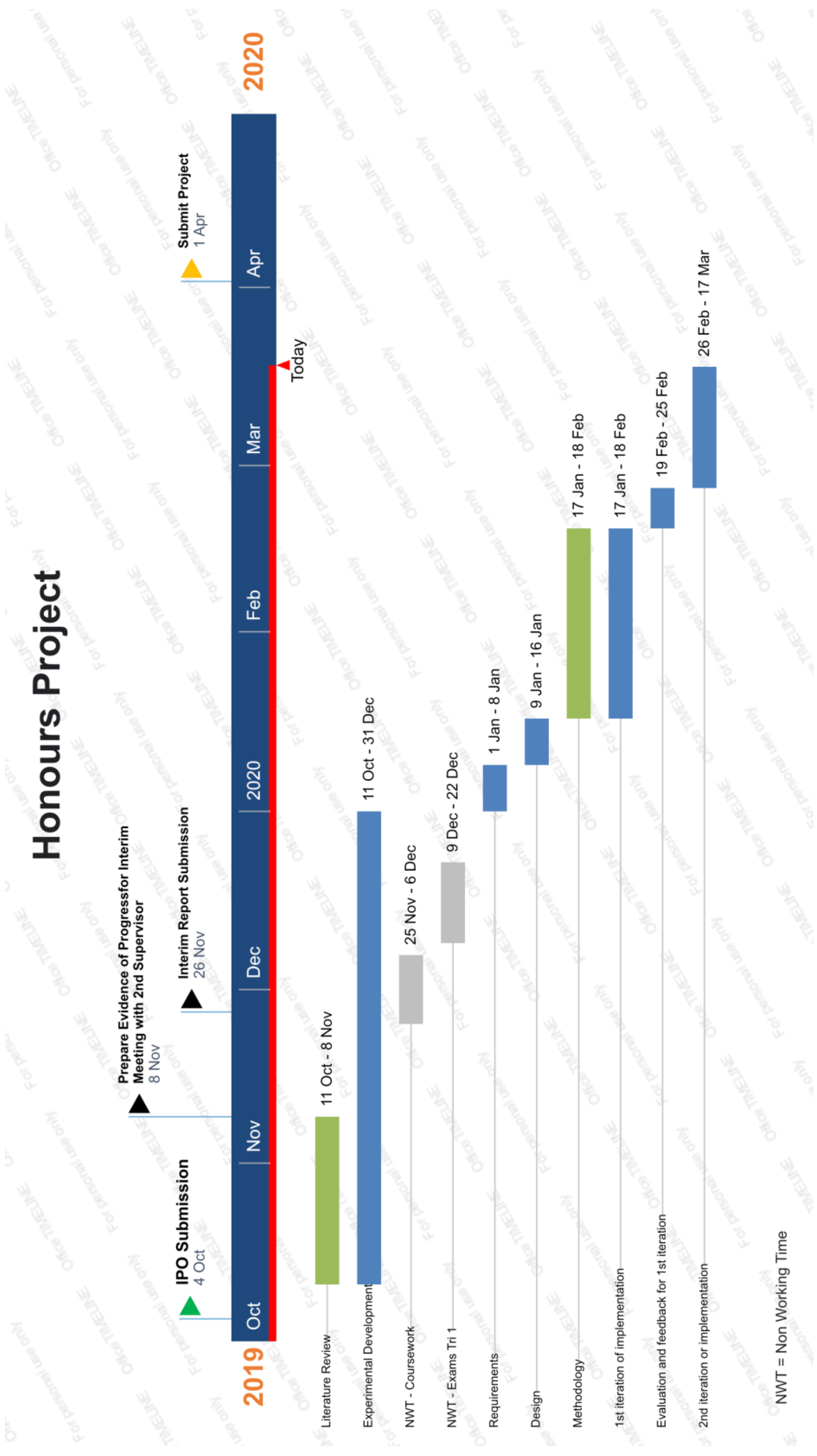
Signatures: Supervisor [illegible] [illegible] Second Marker

Student [illegible]

The student should submit a copy of this form to Moodle immediately after the review meeting; A copy should also appear as an appendix in the final dissertation.

* Please circle one answer; if **no** is circled then this **must** be amplified in the space provided

Appendix 3 Project Management - Diary Sheets and Gantt Chart



EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson **Supervisor: Brian Davison**

Date: 11/10/19 **Last diary date:**

Objectives:

- Notes & references for the PWA section of the Literature Review
- General notes on different sections of the Literature Review
- Complete Gantt Chart (main deliverables & non-working time)
- Experiment further with PWA development, find hardware/software features available
- Discuss project with friends/family to figure out what questions need answering in the literature review

Progress:

- Discussed with 2 friends and a parent, written up main points to better understand what information I need to
- Started Gantt chart, some more information needed
- Going over web tech module practicals on express
- PWA section of lit review draft complete
- Whatcanthewebdo.today for hardware/software features, iOS/Android

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 18/10/19	**Last diary date: 11/10/19**

Objectives:

- Go through academic articles, become more familiar with academic papers
- Think about the aim of the project and the problem being addressed
- Continue with lit review
- Different browsers

Progress:

- Lit review up to >1000 words. Mobile Applications section drafted and more info added to PWA section
- Browser/Mobile operating systems researched, findings written up for later use
- General reading of academic papers when searching for sources for the lit review
- Some possible project aims/problems written up

Supervisor's Comments:

- Good structure so far, generally coming along well
- Going forward, don't forget to include information that is assumed as obvious
- Current PWA section

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 25/10/19	**Last diary date: 18/10/19**

Objectives:

- Look at adding diagrams to literature review for clarity and explanation
- Section introductions
- Self evaluation

Progress:

- Project aims written up
- Browser research
- Lit review continued

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 1/11/19	**Last diary date: 25/10/19**

Objectives:

- Diagrams for lit review
- Introductions to sections
- Find an open source pre-existing website that could be converted
- Have a look at web framework support with PWAs (especially python)
- **Figure out why?**
- Does student know how they're going to evaluate themselves/project (from project aim)
- Finish off literature review draft

Progress:

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 8/11/19	**Last diary date: 1/11/19**

Objectives:

- Finish literature review

Progress:

- Finished off literature review draft and diagrams
- Looked at frameworks, flask and express

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 15/11/19	**Last diary date: 8/11/19**

Objectives:

Progress:

- Sorted references for lit review, added diagrams etc
- Finished final lit review draft

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 22/11/19	**Last diary date: 15/11/19**

Objectives:

- Make amendments to literature review based on feedback
- Start looking at PWA development
- Draft methodology
- Start to think about Christmas work

Progress:

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Chris Johnson **Supervisor: Brian Davison**

Date: 13/12/19 **Last diary date: 29/11/19**

Objectives:

- The flask mega tutorial
- Build a CRUD web application with python flask (Python 3)

Progress:

- Updated lit review based on feedback from interim meeting
- Added sections to lit review based on revised aims & objectives
- Played with PWA development with Flask and Express
- Produced draft methodology

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 23/01/2020	**Last diary date:**

Objectives:

- Azure, AWS – look into deployment options
- Further develop methodology
- Update gantt chart
- Troubleshooting

Progress:

- Added section on service worker caching strategies to literature review
- Followed Flask mega tutorial recommended in previous meeting
- Tinkered with the microblog site from tutorial, turning into an installable PWA.
    - Added service worker
    - Added app manifest
    - Added a like system
    - Enabled push notifications
- Mostly completed methodology, some further guidance needed before it's finished.

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 30/01/2020	**Last diary date: 23/01/2020**

Objectives:

- Continue website development. Get first few sections of process completed.
- Develop set of questions for feedback "online tutorial" evaluation questionnaire
- More completed methodology
- Google analytics
- Talking through process next week. Meeting Wednesday 11:30am

Progress:

- Continued with methodology, another couple hundred words.
- Updated Gantt chart
- Registered a domain for hosting process (flaskpwa.com)
- Hosting website with AWS

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson **Supervisor: Brian Davison**

Date: 05/02/2020 **Last diary date: 30/01/2020**

Objectives:

- Prism
- Scrollspy

Progress:

- Methodology rewritten and almost complete, not much left
- Started/almost finished multiple process sections
- Questionnaire research, questions are ready
- Google Analytics set up
    - TODO: install and bookmark events
- Attempted to host Flask with apache on aws for HTTPS, no luck yet
- Will have access to iPhone as of tomorrow hopefully

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 12/02/2020	**Last diary date: 05/02/2020**

Objectives:

- Tweak first part of methodology to flow following the lit review
- Overview section – orients the reader, what to expect
- Diagrams
- Refinement
- Table of platform differences
- Footnotes in methodology
- Incompatibilities, push notification/background sync
- Split testing and evaluation

Progress:

- Website finally hosted with apache, HTTPs working fine
- Confirmed iOS tags are working
- Confirmed Android stuff is working
- Continued developing process/process site
- Prism code snippets
- Working on scrollspy

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson **Supervisor: Brian Davison**

Date: 19/02/2020 **Last diary date: 12/02/2020**

Objectives:

- Separate website feedback from process feedback
- Pilot check
- “research survey methods”
- jsTree
- local data storage

Until Sat:
- Tidy up site
- Finish up a few sections
- Expand on text sections
- Add an analytics tag for installation
- Make feedback functional

On /After Saturday
- Share site
- Mobile version of site

Next meeting - Friday 2:30

Progress:

- Continuing with website development
- On track to launch/share on Saturday
- Updated list of sites to share to

Supervisor’s Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 28/02/2020	**Last diary date: 19/02/2020**

Objectives:

- SET08114 Mobile application development – Simon Wells
- SET08101 Web Development – Sally Smith
- SET11112 Masters Module – Malcolm Rutter
- Software engineering practices for methodology
- Website's mobile functionality
- Friday 11:30 for meeting

Progress:

- Got website ready for launch
- Launched website
- Posted around net looking for feedback

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson **Supervisor: Brian Davison**

Date: 06/03/2020 **Last diary date: 28/02/2020**

Objectives:

- Look at how appropriate methodologies are for process (agile)
- Repo
- Think for self evaluation
- Assessment guide Moodle, self appraisal
- Think about poster
- Report
- Next week: 11:30

Progress:

- Website optimised for mobile
- Spoke at lectures/practicals to get feedback
- Software Engineering Methods for methodology
- Data presentation for feedback

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Christopher Johnson	**Supervisor: Brian Davison**
Date: 11/03/2020	**Last diary date: 06/03/2020**

Objectives:

- Full draft for next week
- Software methods, templating perhaps
- Development patterns, OO
- Deviations, delivered app, results of survey

Progress:

- Methodology progress
- Abstract draft

Supervisor's Comments:

Appendix 4 – Example Web App Manifest

```
{
  "name": "Developing a PWA in Flask",
  "short_name": "Flask PWA",
  "theme_color": "#CBD394",
  "background_color": "#F1F7C6",
  "icons": [
    {
        "src": "/static/images/icon-64.png",
        "type": "image/png",
        "sizes": "64x64"
      },
      {
        "src": "/static/images/icon-96.png",
        "type": "image/png",
        "sizes": "96x96"
      },
      {
        "src": "/static/images/icon-
192.png",
        "type": "image/png",
        "sizes": "192x192"
      },
      {
        "src": "/static/images/icon-
512.png",
        "type": "image/png",
        "sizes": "512x512"
      }
  ],
  "start_url": "/",
  "display": "standalone",
  "orientation": "portrait"
}
```

Appendix 5 – Questionnaire Questions

About user – Range from no experience to very experienced

How much experience do you have with Flask?

How much experience do you have with PWA development?

How much experience do you have with web development?

Site Discovery – Option for each website the site was shared to

Where did you find this site?

About the website – Likert scale

Q1. I find the information available on this website helpful.

Q2. I personally find this website useful.

Q3. I think the information in this website is described clearly.

Q4. I think the information in this website is presented clearly.

Q5. I find this website easy to use an navigate.

Q6. I find the design of this website appealing.

Q7. I am likely to revisit this website.

About the development process

Q8. I find the development process to be helpful.

Q9. I personally find this development process useful.

Q10. I find the information in this development process easy to understand.

Q11. I consider this development process user friendly.

Q12. The development process is presented in a suitable way.

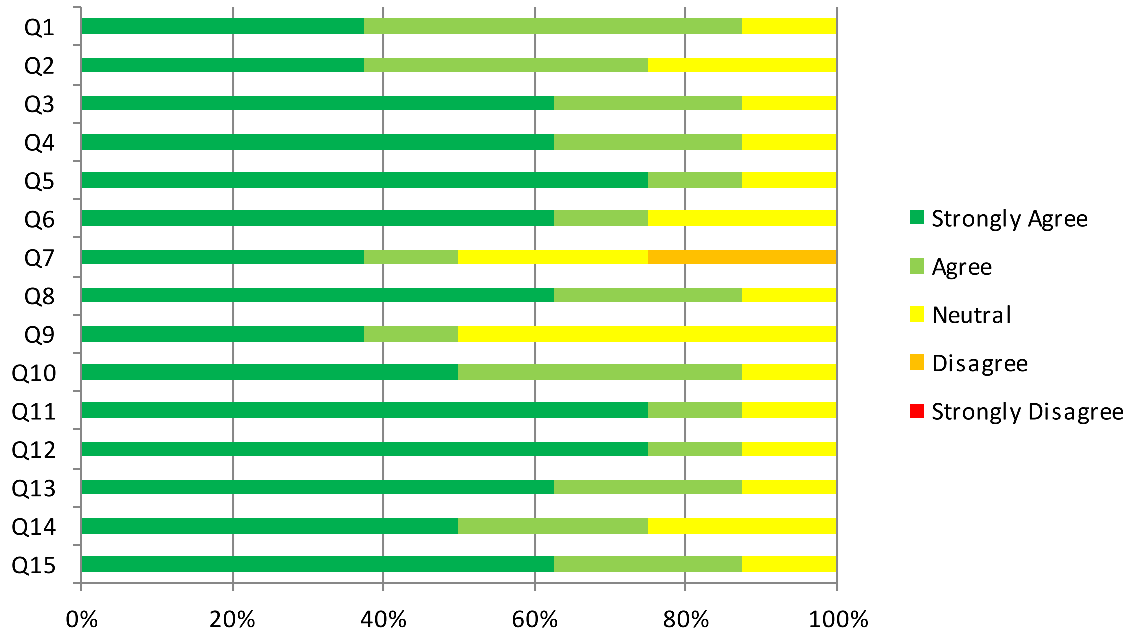
Q13. The development process provides me with sufficient resources to create my own PWA in Flask

Q14. I find the structure of the development process clear.

Q15. I find the development information available to be accurate.

Appendix 6 – Questionnaire Results

Using the questions from Appendix 6



Appendix 7 – Forum and Online Community Comment Feedback

Web Develop Forum
“Great Job ”

Web Designer Forum
“I gave up, it's all one huge long page without any navigation. Tried the feedback but it’s way too complicated”

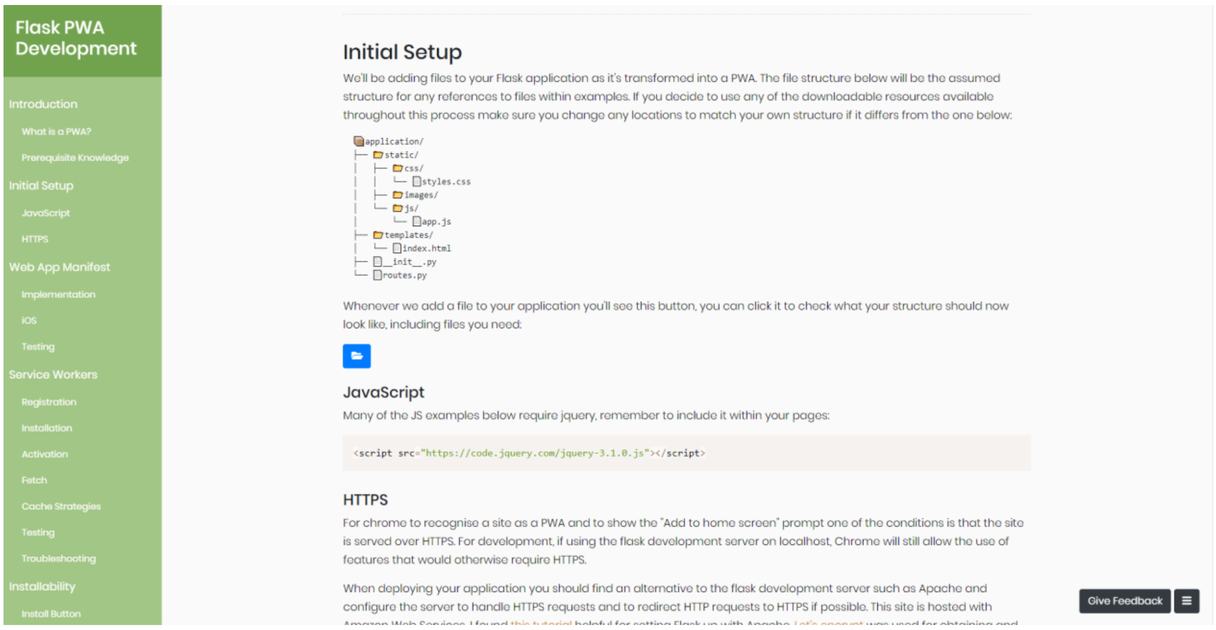
Progressive Web Application Subreddit
“Site looks great”

“This looks reasonably comprehensive. I've previously worked through the Google PWA tutorial. Converting the final product for submission to the play store was painful.
I'm seeing some overlap in structure of content with the Google tutorial. That’s not a bad thing, has been a while since I did the Google PWA and haven't dived into your tutorial in detail yet.
For more traction, I'd suggest doing at least an intro tutorial on YouTube. Maybe a demo of a project you've built with the material in your website. There's a lot of upside potential to turn the content into a course.
I've been looking for an excuse to get back into PWA's. This could be it.”

“You should make a responsive overview of the content. On mobile I literally scrolled down and down and down. Nevertheless it's a good overview of a PWA.”

Appendix 8 – Website Desktop and Mobile Views

Desktop



Initial Setup

We'll be adding files to your Flask application as it's transformed into a PWA. The file structure below will be the assumed structure for any references to files within examples. If you decide to use any of the downloadable resources available throughout this process make sure you change any locations to match your own structure if it differs from the one below:

```
application/
├── static/
|   ├── css/
|   |   └── styles.css
|   ├── images/
|   └── js/
|       └── app.js
├── templates/
|   └── index.html
├── __init__.py
└── routes.py
```

Whenever we add a file to your application you'll see this button, you can click it to check what your structure should now look like, including files you need:

JavaScript

Many of the JS examples below require jquery, remember to include it within your pages:

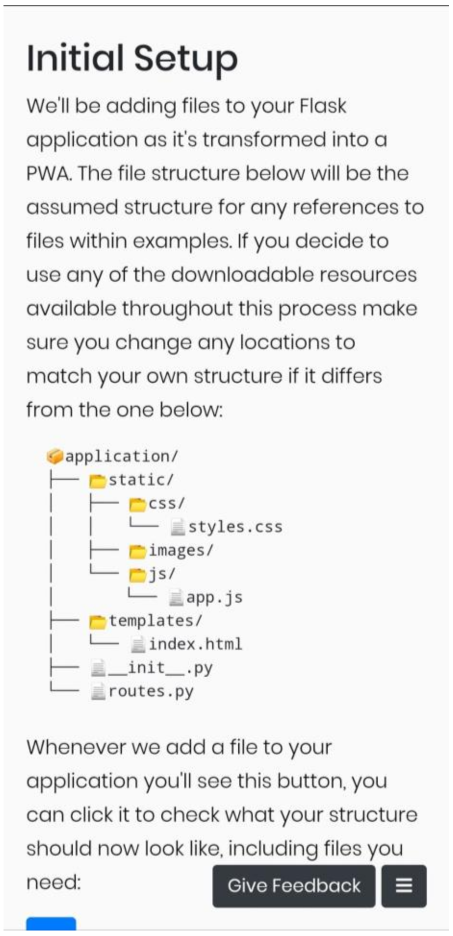
```
<script src="https://code.jquery.com/jquery-3.1.0.js"></script>
```

HTTPS

For chrome to recognise a site as a PWA and to show the "Add to home screen" prompt one of the conditions is that the site is served over HTTPS. For development, if using the flask development server on localhost, Chrome will still allow the use of features that would otherwise require HTTPS.

When deploying your application you should find an alternative to the flask development server such as Apache and configure the server to handle HTTPS requests and to redirect HTTP requests to HTTPS if possible. This site is hosted with

Mobile with navigation sidebar disabled and enabled

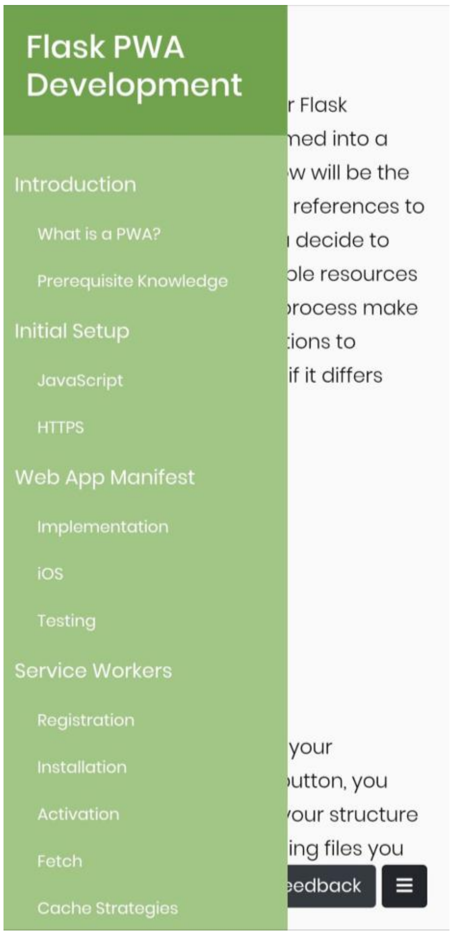


Initial Setup

We'll be adding files to your Flask application as it's transformed into a PWA. The file structure below will be the assumed structure for any references to files within examples. If you decide to use any of the downloadable resources available throughout this process make sure you change any locations to match your own structure if it differs from the one below:

```
application/
├── static/
|   ├── css/
|   |   └── styles.css
|   ├── images/
|   └── js/
|       └── app.js
├── templates/
|   └── index.html
├── __init__.py
└── routes.py
```

Whenever we add a file to your application you'll see this button, you can click it to check what your structure should now look like, including files you need:


Flask PWA Development
Introduction
What is a PWA?
Prerequisite Knowledge
Initial Setup
JavaScript
HTTPS
Web App Manifest
Implementation
iOS
Testing
Service Workers
Registration
Installation
Activation
Fetch
Cache Strategies