University of Portsmouth

School of Creative Technologies

Final year Project undertaken in partial fulfilment of the requirements for the BSc (Honours) in Digital Media

Digital Media Bespoke Project:

One shoe fits all: Creating a bespoke progressive web app (PWA) with a headless CMS

By
Marcus Harding
UP818058

Project Module: CT6CTPRO2

Supervisor: Dr Claire Bailey-Ross

Project Type: Artefact

March 2020

Abstract

2019 has been a particularly good year for web development. The introduction of new technologies, frameworks and methodologies is exciting to say the least. In recent years JavaScript has quickly transitioned into the most popular programming language on the web (Sun, Bonetta, Humer & Binder, 2018, p. 196). Meanwhile a new term: ‘Progressive Web App’ (PWA) has been gaining traction, originally proposed by google back in 2015 (Cardieri & Zaina, 2018, p. 1). At present, the prevalence of mobile phones around the globe is monumental. Here in the UK alone around 93% of the population own a smartphone (Kuss et al, 2018, p. 141). These users also spend more time on their smartphones than any other device, and as a result the use of smartphone applications has risen significantly. This has had a major influence within the industry, creating a conscious shift towards a mobile first approach which is now considered industry standard. However, implementing responsive design is notoriously difficult and costly upfront (Mullins, 2015, p. 6). This in turn has created a high demand for a solution which can bridge the gap between multiple devices whilst retaining high levels of quality and bringing costs down.

As PWA’s become increasingly popular, claiming to solve some key issues within the industry, now is the perfect time to conduct some research around the area. The aim of this report will be to find out exactly what they are alongside the benefits or drawbacks to this technology in comparison to conventional methods. A PWA will be developed to discover and conclude first-hand whether they are indeed the revolutionary technological development users and developers have been looking for in this mobile first generation.

Table of Contents

Table of Figures

Introduction

As the digital sector continues to rapidly grow so too does the Web Development industry and its appeal. The number of software developers in the world is predicted to reach 27.7 million by 2023. The United States currently has the largest population of software developers in the world whilst the UK sits at the 2nd largest in Europe (Daxx, 2020). Compare these statistics to the number of U.S Americans that own a mobile phone and they almost seem trivial. About 95% of Americans own a mobile phone (Pew Research Center, 2018) with 26% of these users reporting that they are online “almost constantly” (Perrin & Jiang, 2018). This unanimous mobile usage has been the main influencer behind the transition towards a mobile-first approach within the development industry. Meanwhile several JavaScript frameworks have been gaining traction quickly, as of 2019, Angular, React and Vue JS have become some of the most popular JavaScript frameworks (Saks, 2019, P. 1). It is not surprising that these JavaScript based frameworks have now become the driving force behind the rise of PWA’s.

PWA’s are the web development industries response to the sudden and prevalent rise of mobile phone usage. It would not be unreasonable to say that mobile devices have now taken over. With content becoming progressively condensed for smaller screen sizes and a growing number of daily tasks being taken from desktop to mobile a conclusion could be made that the success of PWA’s is in fact make or break for the web development industry. A greater number of users are looking to access the same websites on mobile devices yet still expecting the same fluid and performant experience that native apps provide. Because of these heightened expectations the web development industry ultimately has no choice but to keep up or fall behind. This increase in pressure to stay relevant and competitive is what brought about the introduction of the Progressive Web App.

As the industry dives headfirst into this new mobile-first generation, any experienced developer or designer will now tell you just how important it is for a website of professional standard to be fully optimized and responsive for an optimal experience (Mohorovičić, 2013, p. 1209). Fully responsive, clean code is the foundation and backbone to any functional website. Developers should be morally obligated to ensure that client websites are fully responsive and high performing. Anything less simply will not survive the scrutinous, fast paced usage from users browsing on a mobile device or desktop for that matter. A participant study by Oulasvirta, Tamminen, Roto and Kuorelahti (2005, P. 926) found that continuous attention to mobile devices when used in an everyday setting quickly fragmented and broke down to bursts of just 4 to 8 seconds. Additionally, attention to the mobile device would be interrupted by glancing at the environment by up to 8 times during a subtask of waiting for a Web page to load. These statistics shed light on just how little room for error there is when it comes to website performance. For large scale

companies, this is even more important. A PWA is a website that takes the best practices of web development and serves them cross-platform whilst also simulating a native-like experience (Tahirshah, 2019, p. 8). Essentially, they can serve a single fully responsive code base across many different platforms. This is so advantageous for businesses since having one responsive code base eliminates expenses and cuts overall costs significantly (Yadav & Barwal, 2014, P. 154). For any business or client looking to tighten up on costs this revolutionary new technology could prove to be a highly attractive option. The ability to serve a website across multiple platforms whilst also behaving like a native app which in turn cuts costs almost sounds too good to be true.

First-hand development of a PWA for an external stakeholder really is the best circumstance to be able to put this state-of-the-art technology to the test. Developing a PWA under these conditions will make it possible to challenge the benefits that PWA's promise over Native apps through the analysis of this process in a real life setting. Having a stakeholder involved will increase the validity of the analysis by putting the application through real life situations and scrutiny. Undertaking this personally will also provide an individualized experience from which conclusions can be drawn. The alternative option would be to use second-hand research. This option would not be as reliable since the experience would not be unique to this study.

As the popularity of PWA's increases a variety of new, smaller technologies are introduced. These are the technologies that either go hand in hand with developing a PWA or are the essential technologies that a PWA comprises. These technologies in question are what make up several of the four minimum requirements for an application to be classed as a PWA. Firstly, there is the web app manifest, a simple JSON file which may include the application name, icons, splash screen image, status bar and location bar colour amongst other things (Břoušek, 2017, p. 14). By using this within the PWA it is possible to really give the application a native feel when installed on a mobile device. Secondly, it is important to make sure that an application icon is also referenced within the manifest for when the app is installed to a device. Without this the application will fail to meet the minimum requirements. The third key requirement is that the application makes use of a service worker. Service workers in mobiles allow users to work offline or on low quality networks (Pande, Samal, Somani & Kakkirala 2018, p. 195). This feature is an essential part to making a PWA feel like a native app. If a user's internet goes down the PWA will still be fully functional thanks to the service worker caching content. The last requirement is for the website to be served over Hypertext Transfer Protocol Secure (HTTPS). HTTPS is an end-to-end protocol, which establishes encrypted tunnels to deliver sensitive information between clients and web servers (Liang et al, 2014, P. 67).

By fulfilling these requirements, the application can have the best chance of achieving a truly native feel. Serving the application over HTTPS also minimizes the risk of deterring any users. This secure connection can build confidence and trust

within individuals who may otherwise be put off by the fact that their personal data is not encrypted if this requirement is not fulfilled. The presence of a padlock in the URL address bar is a key indication of whether a website is secured over HTTPS (See Figure 1).

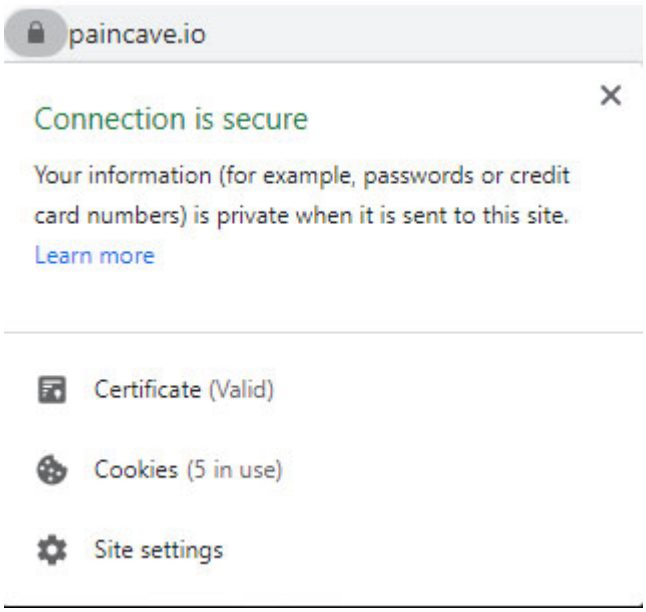

Figure 1 -*The padlock image in the URL is a clear indication as to whether a website is secured over HTTPS. As seen here the PWA connection was secure, successfully fulfilling this important requirement.*

Based on a study by Kelley & Bertenthal (2016, P. 171) participants were more likely to log in when a lock was present than when no lock was present. Despite this secure connection being a mandatory requirement in establishing your application as a PWA, this should arguably be a mandatory requirement for any website regardless.

The rapid advancement of mobile technology and apps is to thank for the introduction of new technologies and ways of working within the web development industry. React is a JavaScript library, for building declarative views (Components) (Mohammadi, 2010, P. 15). This framework has been very helpful in the development of PWA's; however, there is one drawback to using this technology to build client websites. Websites with React on the front-end are not compatible with conventional CMS's. For this reason, if a React application requires a CMS it must be served headlessly. Headless architecture refers to a situation where a database-driven CMSs content is accessible via a web-service API. The end-user experience is then delivered by a JavaScript application rendering the output of the API (Minna, 2019, P. 22). One advantage of serving your CMS Heedlessly is an increase in application security, users must take care of security of their code only, scalability and a fact that application lifecycle is not influenced by the CMS since it is decoupled (Headless) (Čechák, 2017, P. 4).

Technological breakthroughs have been made within the industry, the introduction of these new tools and frameworks has been exciting. However, it would be worth giving some consideration as to whether these advancements have any disadvantages. For example, PWA's performance has often been considered as lower than performance in native apps (Hiltunen, 2018, p. 6). Java has been the default language for Android development for a while now (Oliveira, Oliveira & Castor, 2017, P. 42). The introduction of PWA's can minimise the requirement for web developers to learn a new language when looking to create a 'native' application. Yet there is still a chance that an investment of time will be needed in order to learn the technologies that a PWA is built with. This poses the question: is it not more beneficial for developers to stick with learning a native language if the time investment required to learn PWA development works out the same? Especially since PWA's are trying to replicate what native apps do best. These are all questions that will be answered first-hand by formulating an opinion as to whether PWA's are a good alternative with reflection aimed back towards these questions. This opinion will be made possible by the process and subsequent learning curve involved with developing a PWA in a real life setting for a client.

The report will be structured as follows: An in-depth and concise review of current literature relating to the topic at hand. This review will seek to uncover a definition of what a PWA is exactly. The review will then continue to break down and elaborate on all the technologies that a PWA encompasses. After this point the review's objective will then transition toward identifying new areas of research around the topic of PWA's that have little to no published literature. Establishing areas that lack research will create a clearer direction toward answering the questions that will create new literature by being resolved. Next a succinct methodology with the intention of creating a fit for purpose development plan will be offered. Fulfilling this methodology will create an opportunity to put these questions to the test through a comprehensive discussion on the design and development phase of the project. Questions raised within the literature review and the project will be continuously reflected upon. This discussion will also be a good place to test the identified benefits that PWA's claim to bring to the table. After following this extensive process of first-hand research, a formulated conclusion will be presented as to whether PWA's are indeed a viable alternative to native apps.

Literature Review

Introduction

Native Apps have been making exponential growth in recent years, direct revenue from mobile apps is forecast to be $57 Billion by 2020 (Burroughs, 2017, P. 3). This growth can be attributed to the unanimous and global usage of mobile phones. As

this popularity continues to grow the web development industry pushes even more towards putting mobile design and development first. As mentioned, it is now imperative to serve modern websites in a fully responsive manner across all devices. This extensive growth of mobile devices and in turn mobile apps is no doubt the underlying factor to creating such a drastic shift within the web development industry. All the while PWA's continue to gain further traction with the aim of keeping the industry relevant. At present a modest understanding of what encapsulates a PWA is currently held: a collection of technologies, design concepts and web API's which work in conjunction to provide an app-like experience on the mobile web (Břoušek, 2017, p. 6). As definitive as this statement may be, a definition alone is not enough. A deeper understanding is required in order to conduct applicable research. The goal moving forward will be to identify areas of research that are lacking before moving on to creating a PWA first-hand. This process will make it possible to assess and subsequently fill any identified gaps in research.

Scope

The overall scope of this review will be to gain this 'deeper' understanding, whilst assessing current literature. As more clarity around the technologies which make up PWA's is created, in tandem with the literature which covers this, it will be easier to sculpt a distinct direction for this study. Creating direction will pave the way for targeted research into areas where research is inadequate, creating a well-defined plan to solve this. The organisation of this review will be as follows: The first objective will be to break down and gain a solid understanding of what a PWA is. This will be done in conjunction with assessing the literature around the topic area. By reviewing current literature and gaining greater clarity on the topic area it will become apparent which areas are lacking adequate research. From that point the literature review can then transition towards identifying an area within this topic where a useful contribution can be made. Having an area of contribution identified will then translate to the rest of the project, putting things in a good standing to build a purposeful methodology and discussion.

Review of literature

Initial searches into research on Progressive Web Apps (PWAs) returned a surprising amount of literature. The consensus appears to be that there is a lack of research covering this topic (Biørn-Hansen, Majchrzak & Grønli, 2017, p. 344). Which upon further investigation proves to be the case for articles published during or before the year 2017. Many papers published during this timeframe directly mention the lack of literature covering PWAs. These publications even reference other papers which make the same point of literature coverage falling short (Tim, 2018, p. 5739). In addition, there is a strong emphasis around the fact that during this time a search for 'Progressive Web Apps' only returned two search results

(Biørn-Hansen et al, 2017, p. 345). Contrary to this a search for the same keywords in November 2019 provides a 10-fold increase.

Progressive Web Apps started life as an up and coming web technology creating more questions than industry experts had answers to. This sparked a plethora of research between 2017 and 2018 looking to make more sense of this topic. Concrete answers and research around the role that PWA's fulfil is now far easier to access. It is abundantly clear that a PWA is a web application with the intention of delivering a native experience on a mobile device (Fransson & Driaguine, 2017, p. 4).

However, it is fair to say that in-depth research around the topic still falls short. Now that a technical understanding of the subject has been established through current research, the next step is to choose a more comprehensive area to focus on. Looking for Literature aimed at assessing the benefits and drawbacks, alongside advantages that PWA's bring over Native Apps poses the same issue. There is a lack of detailed literature. As a result, there is not a compelling argument as to whether the industry should be moving this way or not. A solution to this would be to conduct a case study to assess the areas identified as lacking investigation. From this study a justification could be made as to whether the industry should be adopting such practices based on practical assessment of the advantages and disadvantages of PWA's in comparison to native apps.

It is important to acknowledge the aptitude that progressive web apps have in closing the gap between web platforms across devices. It may be fair to say that they successfully fulfil the expectations of native apps and more (Ater, 2017, p. 15). If PWA's are indeed a viable option there will be a great deal of scope for comparison, review and assessment into the use of PWA's as an alternative to native apps. First-hand experience developing a PWA would be a great way to form an opinion as to whether PWA's can truly fulfil the role that Native Apps currently hold.

This new technology encapsulates a developmental movement bringing the native look and feel to the web (Hiltunen, 2018, p. 1). PWA's could indeed be a strong alternative to native apps, but research around their benefits and drawbacks is still lacking. To begin testing the pros and cons of this technology thought should be given towards native apps, this will ensure the assessment is not one sided. Native apps are developed to be compatible with a specific operating system (OS) (Hiltunen, 2018, p. 4). This is considerably restrictive not only for users but also for developers. IOS devices traditionally rely on the use of Objective-C and Android devices require the use of Java (Liu et al., 2015, p. 337). Therefore, it is not always possible to use the same codebase when looking to serve the same app across multiple native devices.

This is where PWA's come in, they use one codebase across multiple devices, ultimately solving this critical constraint. However, they are not without any caveats. One area they may never be equal is the consistent specifications that Native Apps must adhere to. Mobile Operating System manufacturers provide access to various libraries of code for developers to use (White, 2013, p. 9). This creates a level of consistency within apps for a specific device.

One technology is looking to replicate these levels of consistency found within Native apps. That technology is React, a JavaScript framework developed by Facebook (Gackenheimer, 2015, p. 1), React has grown rapidly in popularity and is built on components, with each one representing a view (Beshir, 2016, p. 5). It is this approach to re-usability and componentization which has allowed React to gain so much traction so quickly. The framework has played a crucial role in the growth of PWA's. React interacts with the Document Object Model (DOM) in an unconventional manner. Rather than interact with the DOM generated by the browser React creates an in-memory data structure cache which computes the changes made and then updates the browser. This results in far greater performance (Aggarwal, 2018, p. 133). Performance is one of the top priorities and a defining factor for PWA's (Tahirshah, 2019, p. 12).

Another base criterion expected of a progressive web app is that page transitions do not feel like they block on the network, transitions should feel snappy and seamless (like an app) (Johannsen, 2018, p. 7). React facilitates the creation of interactive and stateful components (Kumar & Singh, 2016, p. 226). By giving each component a state transitions can become seamless as React only has to re-render components who's state changes from page to page.

To summarize, by clearly articulating what a PWA is through a densely researched topic area and building a comprehensive understanding using this research, it has been possible to move on and identify areas where investigative study is deficient. There are very concrete sources of research that define the benefits and drawbacks PWA's have in relation to native apps. This makes it difficult to state whether they can be classed as a viable alternative. Based on this identification, and additionally identifying suitable frameworks which a PWA can be built with, it is now possible to move on and craft a clear methodology in order to begin answering these questions to fill the identified gaps in research.

Methodology

An accurate understanding of what a PWA is, in addition to identifying a research area which is lacking has been achieved through assessment of current literature. The goal now is to determine an appropriate research method in order to allow for assessment and analysis which will help form new decisive research. A fitting

research method for this is a case study, which is arguably defined as an in-depth study of a single area (unit) (Gerring, 2004, p. 341). Conducting an in-depth study will be made possible through the development of a PWA for an external client. One advantage of this is that case studies often produce detailed qualitative accounts, ultimately enabling the exploration of research in a real-life environment (Zainal, 2007, p. 4). The purpose of this study will be to provide a very specific insight into the process of PWA development. The subsequent results may not necessarily produce undeniable qualities of all PWA's. Therefore, it is more practical to say that any implications or problems that are produced and resolved throughout this study could then be used by other developers as guidance. The documented results will be able to aid individuals looking to adopt the same practices and integrate the use of PWA's within their own projects.

The choice of conducting a case study over other research methods will directly assist in answering the identified questions from the literature review. There is a very high chance that there will be positive and negative outcomes as a result of this case study. This will consequently allow for the direct analysis of the benefits and drawbacks that can be experienced through the development and use of PWA's. A mobile first approach will be taken to develop the progressive web app, keeping the client and end users in mind. The application will comprise two core technologies to make up the front and back-end. React will be used on the front-end, identified in the literature review as a highly efficient technology when used to build progressive web apps. Since React is not compatible with a conventional integrated Content Management System (CMS), the back end will be headless. A headless CMS is a back end only content management system (CMS) which makes content accessible via a RESTful API (Gustafsson, 2019, p. 20). This will allow for the presentation of content using any front-end framework on any device (Weber, Wesselman, Shyba, & Seifert, 2018, p. 1). Where a traditional CMS is integrated with the front-end, a headless CMS is not. These two technologies will be the driving force behind the PWA.

Using advanced technologies such as these will fulfil another aim of this project, which is to extensively increase developer skills, or in other words up-skill. Doing so would prove to be highly beneficial since a path of career progression can be characterized by a process of upskilling (Weber et al, 2018, p. 1). But before any developmental work can take place explicit designs will need to be created. Web designs are a crucial stage in the process of building a new website. They help speed up the development process with the goal of giving access to the information visitors desire in the most expedient way possible (Rosen & Purinton, 2004, p. 793). Web design is currently a weaker area of skill. Taking the time to create designs first will be another chance to up-skill further, stepping away from comfort zones.

Upon successful development of the application it will be possible to reflect back on this process, picking out benefits and drawbacks. This will build up an area of

discussion around the positives and negatives to PWA's as well as a clear comparison between the two technologies. Developing an application first-hand will provide the best experience of working with PWA's, allowing for a descriptive account of the development process throughout the upcoming discussion. An emphasis will be placed on creating the most app like experience possible with design and development taking a mobile first approach. Adopting a mobile-first design approach is essentially mimicking the same procedure seen with the design of native apps. It will ultimately make it far easier to form an opinion around whether PWA's can fulfil the things they promise.

Ultimately, the end goal will be an in-depth discussion on the experience of developing a PWA with the required and chosen technologies. A conclusion will then be formed as to whether it would be beneficial for the industry to transition towards greater use of PWA's as an alternative to native apps. This result will be attained by comparing specific pros and cons identified within the literature review against the development process of the application. In doing so any scepticism around PWA's can then either be dispelled or confirmed.

Project Management

Effective project management involves repeated performance of three key areas: Planning, executing and monitoring / controlling processes (Mohammadjafari, Ahmed, Dawal & Zayandehroodi, 2010, P. 11845). The online project management tool 'Trello' (https://trello.com/) was used to manage this project. The use of this platform enabled the fulfilment of the key areas that make up effective project management. Trello's task management system lets users create cards composed of various subtasks. From here it was then possible to organise these into sections based on the project timeline or urgency of completion (See Figure 2).

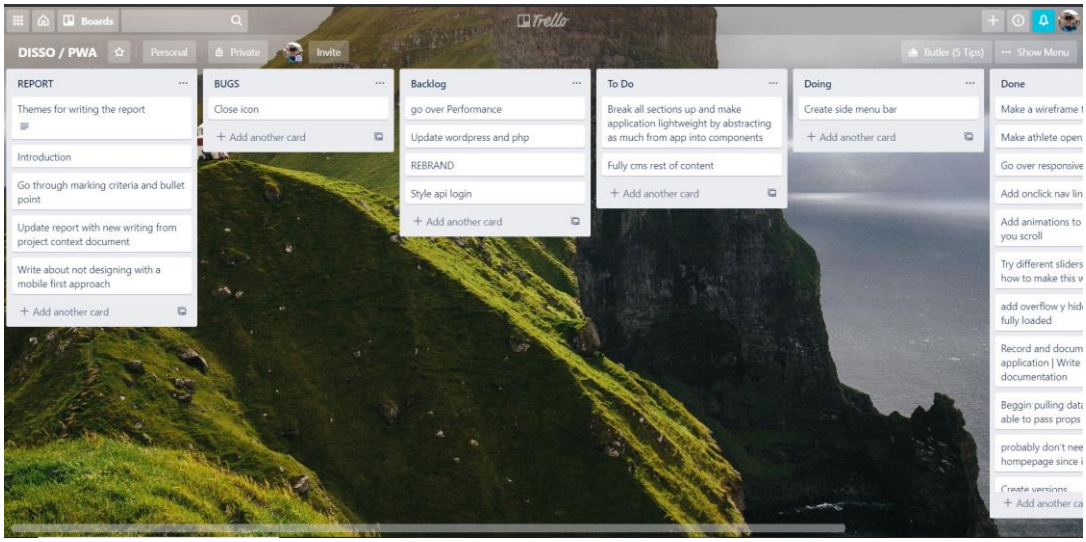

Figure 2- *The Trello project management board that was used for this project, each section represents a different stage that a task can be put into.*

Organising a project management board in this way creates a systematic order of execution based on task priority. This style of project management makes it easier to continually monitor and control the rate of progression with the goal being a steady and consistent progress.

A service called ‘Github’ (https://github.com/) was used to effectively manage the development of the project. Github is a code hosting repository based on the Git version control system (Dabbish, Stuart, Tsay & Herbsleb, 2012, P. 1278). A version control system (VCS) allows a user to track the iterative changes made to the code (Blischak, Davenport & Wilson, 2016, P. 1). This tool was critical to effectively managing the application development, without it the project may have encountered some serious issues. Committing the codebase up to a GitHub repository ensures that it remains backed up and secure. If the local copy were somehow deleted, there would be a saved copy on GitHub. Another key feature of Git is that it enables regular and structured reconciliation of all versions of all the files in the project (Bryan, 2018, P. 25). If something were to break in the codebase, it would be easy to look back at the past versions to see where this break occurred. The project could then be rolled back to its last working state, saving a lot of time and effort.

Carrying out effective Project Management made it easier to transition between different types of work during the project lifecycle. The workload was split between developing the application and documenting the process. Clearly structured task columns in Trello was a great way of quickly switching between the two types of work with minimal confusion.

Discussion

Introduction

This report started life with the objective of creating a clear understanding around what a PWA was. As this understanding grew it became even more apparent as to why this new technology was introduced in the first place. This was the web development industries solution to keep up / compete with the fast growth of mobile apps. From this point the technologies that a PWA consists of were broken down and assessed based on their individual functionality. Taking the time to learn the role that each comprising technology played undoubtedly made it easier when it came to develop the client’s application. This upcoming discussion will fulfil the need for an in-depth assessment around the benefits a PWA provides. The assessment will help determine whether the time to learn these new technologies is worth the investment compared with learning to code natively.

Building concrete definitions of what PWA's are in addition to their comprising technologies opened the door for a more purposeful search into literature. There was already a vast amount of literature surrounding the topic. However, this literature primarily sought to answer the question of what a PWA was exactly and nothing else. Admittedly, this was the question on everyone's mind when PWA's were originally introduced by Google in 2015 (https://thinkwithgoogle.com/). This overabundance of research meant that new literature with the aim of debunking what a PWA is, would not be required. In contrast, research directed towards the benefits and drawbacks that PWA's present in comparison to native apps began to fall short. This was an area which was lacking. From this point forward the literary goal became to uncover, and answer questions related to the benefits that PWA's bring over native apps. Is it worth investing time into learning this new technology? Constructing a thorough methodology was the best way to construct a plan of action to answer these questions first-hand through the development of a PWA. During this planning process the decision was made to conduct a case study. A case study approach will provide the opportunity to breakdown and analyse the development process, all the while creating new literature by linking the experience back to previously posed questions.

This upcoming discussion will analytically document every process that makes up the application build, ranging from design and development to shipping the production build and everything in-between. Reflective segments will be focused on the experience of using each identified PWA technology first-hand. This reflection will relate back to the questions posed at the start of the report, with the intention of answering them with a personal experience of developing a PWA. The end goal will be to form an opinion as to whether PWA's are indeed a better alternative based on the benefits or drawbacks experienced during this discussion.

Planning & Application Set up

Before any design or development could take place a comprehensive tech-stack needed to be created. One of the most popular older stacks is LAMP, this stack includes Linux operating system, Apache HTTP Server, MySQL database and PHP programming language (Vainikka, 2018, P. 10). However, with the introduction of headless CMS and new frameworks such as React it is now possible to create a customised tech-stack comprising many different technologies. An appropriate tech-stack was created (See figure 3).

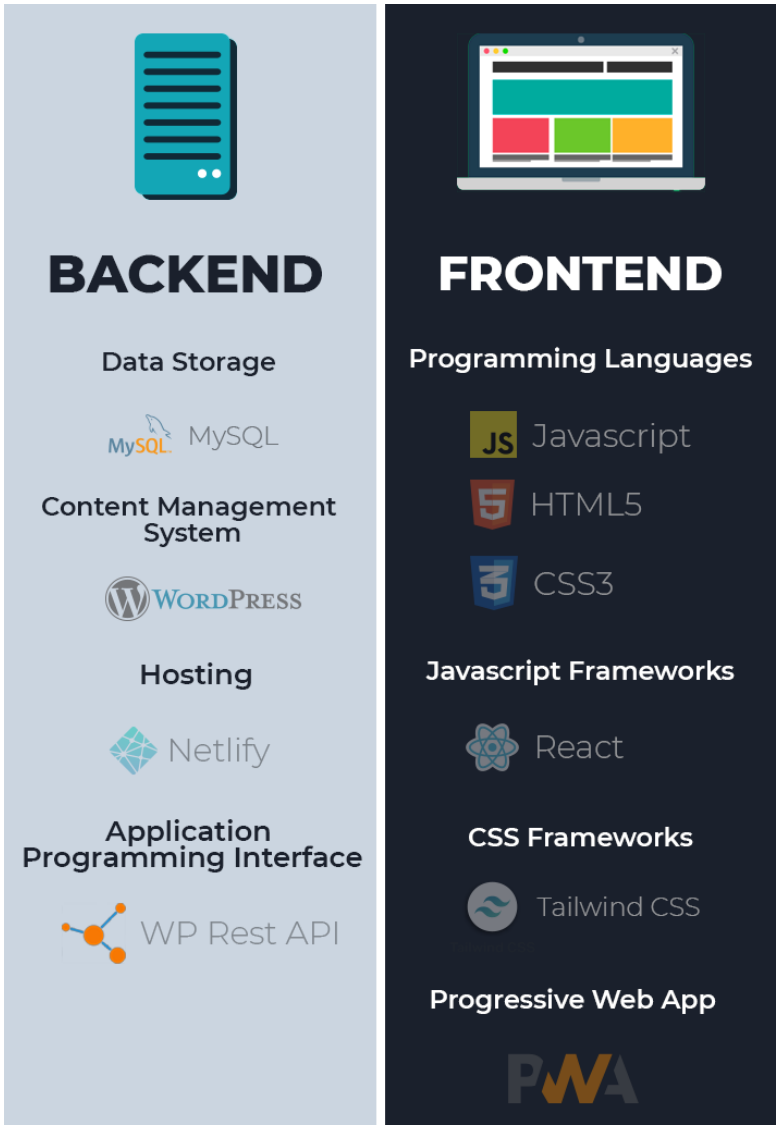

Figure 3 – *The final tech-stack used for the project: The front-end consisted of React framework which uses Javascript in addition to HTML5, CSS3 and TailwindCSS to be able to build fully functional web pages. On the back end Wordpress and MySQL were both hosted on a subdomain accessible via the WP Rest API. The whole application was hosted using the free service Netlify.*

The first point of focus was getting the front-end of the application set up. React is quickly becoming the go-to solution for frontend web applications (Domos, 2017, P. 10). However, any of the JavaScript based frontend frameworks would still work just as well for a PWA. The decision to use React was based on its component pattern, where UI's are split into distinct pieces (Domos, 2017, P. 10). Thus, making it a highly efficient and enjoyable framework to build with. The best way to get started with React is by using 'Create React App'. "Create React App is an officially supported way to create single page React applications, offering a modern build setup with no configuration" (https://create-react-app.dev/docs/getting-started/). The inclusion of a modern build setup saves an immense amount of time. Especially when the development of a build process is not within the scope of this project. In

addition, Create React App comes with a pre-configured service worker, a mandatory component of a PWA. With one command it is possible to spin up a new app using ‘npx create-react-app my-app’. ‘npx’ is a tool intended to help round out the experience of using packages from the npm registry (https://blog.npmjs.org/post/162869356040/introducing-npx-an-npm-package-runner).

Getting React set up and running on the front-end was surprisingly easy and efficient. The only prerequisite was the ability to use a code editor and the command line, which are both basic working knowledge requirements of any developer. Therefore, every intermediate developer should be able to run the previous steps to get React set up. Comparatively, the setup of a native android app using Java requires the developer to use a new program called Android Studio. Android studio is the official Integrated Development Environment (IDE) for Android application development (Golhar, Vyawahare, Borghare & Manusmare, 2016, P. 3661). Since this program is slightly more niche, one could assume there is less chance web developers will have previous experience using Android Studio. A conclusion could be made that the set up process required in order to begin developing a PWA is far more accessible and therefore quicker in comparison to the time it takes to get to grips with a new program like Android studio. With the bulk of the front-end set up complete, the last step before moving onto the back end was the inclusion of a framework called ‘TaildwindCSS’. Tailwind CSS is a highly customizable, utility based low-level CSS framework which can be used to quickly build the user interface (Nguyen, 2019, P. 3). Based on prior experience, the inclusion of this technology will allow for the styling of the web app in half the time of conventional methods.

With everything on the front-end in place attention could be turned towards setting up the back end. A choice was made that WordPress would be an appropriate CMS to use. Firstly, for the fact that this CMS is seen as the industry standard, with sixty percent of all sites using WordPress as their CMS (Cabot, 2018, P. 89). The use of such a popular CMS is a smart decision when building a website for a client, the risk of it becoming outdated or obsolete is minimized significantly based on its prevalence within the market. WordPress can also be very developer friendly when a plugin called ‘Advanced Custom Fields’ is integrated (https://www.advancedcustomfields.com/). This plugin allows for the creation of custom field types which in turn allows the developer to sculpt the back end around the client’s needs (See Figure 4).

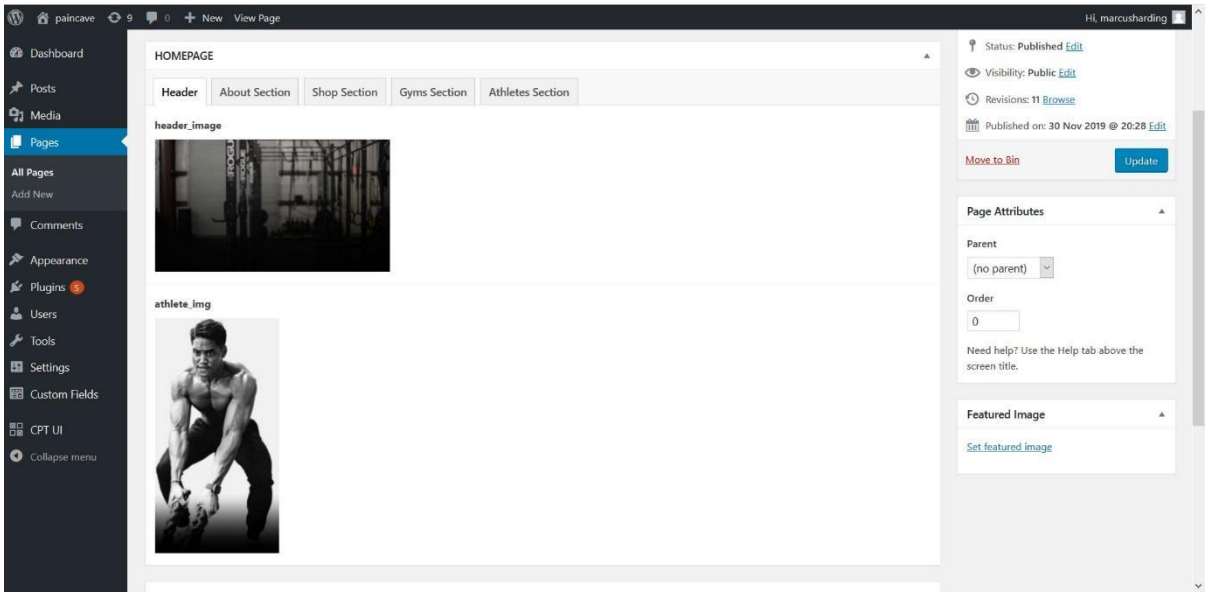

Figure 4 – *Using advanced custom fields made it possible to sculpt the back end into relative sections. This makes it far more intuitive for the developer and especially the end client.*

The CMS was then hosted on a subdomain of (https://www.marcusjh.co.uk) where an SQL database could also be created for data storage. With both back and front-end set up, the last and most important step was to create an API layer between the two. This has been made possible since 2015 by the advent of REST API to the WordPress core (Uzayr, 2016, P. 6). REST stands for 'Representational State Transfer', which effectively means transferring these representational states between the client and server (Lange, 2016, P. 4). An API is an application programming interface, a collection of tools built into a software application to interact with that application (Messenlehner & Coleman, 2019, P. 292). Which in this case, was interacted with externally. The combination of these two technologies allows for the creation of an API endpoint, which makes data accessible through the combination of a URL and HTTP method (Wittern et al, 2017, P. 245). By exposing the data stored in the CMS to this endpoint, it is then possible to fetch that data on the front-end, allowing for it to be used within the application.

Design

The importance of mobile first design and development has been stressed many times over. Purely focusing on consistency across platforms "fails to recognize that the situations that different devices are used in, in fact, already built into the devices that we're using" (Mullins, 2015, P. 5). A cleverly designed web application will change the design appropriately for each different platform. Coming from a web development background it was easy to get caught up in designing for desktop first (See figure 5).

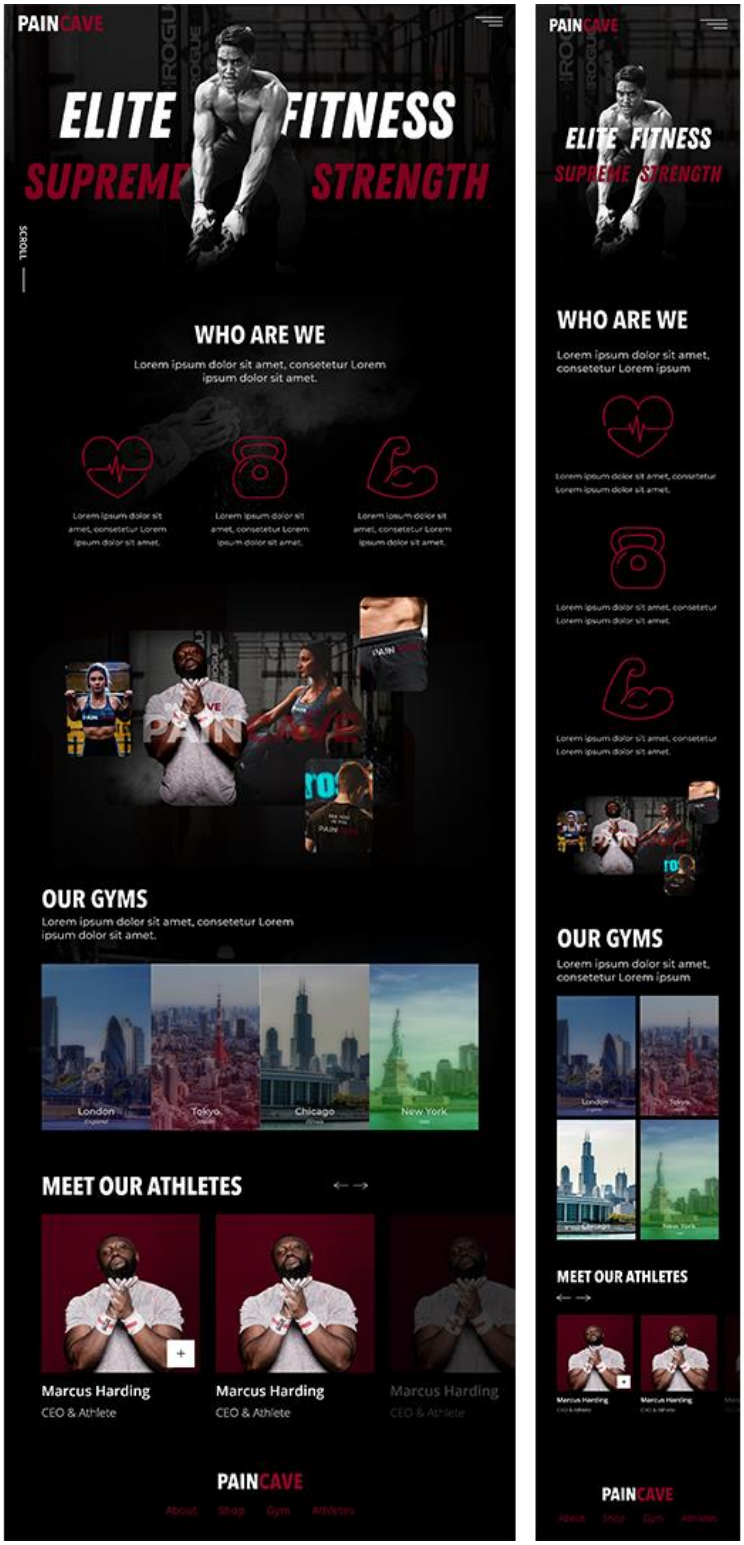

Figure 5 - *The first design phase: As seen here, the design quickly headed towards a desktop first approach. Old habits crept in and the importance of mobile-first was taken out of consideration.*

Adopting this recent mobile-first approach involves breaking old habits and ways of thinking. Choosing to develop and design exclusively for native platforms would cut out the requirement of consideration for any desktop screen sizes. Conventionally,

developing for a responsive mobile-first website involves the use of three media queries aimed at three different screen widths: mobile, tablet and desktop (See figure 6).

```scss
.responsive-element {

    // Every Mobile device up until 768px
    display: flex;

    // Every Tablet device up until 1024px
    @media only screen and (min-width: 768px)  {
        display: block;
    }

    // Every Desktop device greater than 1024px
    @media only screen and (min-width: 1024px)  {
        display: hidden;
    }
}
```

Figure 6 - *The use of SCSS makes it possible to nest media queries. It then becomes far easier to create fully responsive, mobile-first components. An industry skill is to nest media queries for each device width within an element's class. This technique prevents repetition of classes by containing the styles for each screen size inside the class itself, ultimately keeping the code clean and concise. It is also important that each media query uses 'min-width', this allows a website to be adaptive from small up to large screens, this is how a website becomes 'mobile-first'.*

By using percentage widths where possible the elements that make up a website can become adaptive to all sizes in-between each media query specified. An adaptive website that resizes itself accordingly in between each key media query saves the developer an incredible amount of time and effort. Trying to cater for every different screen width would be almost impossible. Apple devices alone account for 16 different screen widths, and that is not even counting desktop (Developer.Apple, 2020). It could therefore be said that the design process for native applications is considerably easier and will most likely take less time than web applications since desktop screens can be cut out of the equation. One less media query and screen size to develop and design for. Additionally, the fact that PWA's require designs for more than just mobile and tablet could ultimately raise questions around the quality of web application designs in relation to creating a native like feel. By additionally designing for desktop attention may be taken away from mobile, this may

consequently cause mobile design to suffer. Therefore, there is a strong chance that designs created purely for native apps will be of higher quality, conveying a ‘native feel’ with more success. This was certainly the case during the development of this application. The transition from building conventional websites to a PWA was evidently seen throughout the design process. Within a short space of time designs had taken a path of desktop-first iterations. It became difficult to backtrack resulting in a desktop feel throughout. This in turn made it creatively challenging to re-invent elements for mobile when a premature desktop iteration was already in place. If designs had taken a mobile first approach as intended the outcome would have no doubt been the opposite, a PWA with a mobile feel throughout which is the goal. An important lesson was learnt on the importance of designing for your primary or most difficult screen size first.

Adobe XD (https://www.adobe.com/uk/products/xd.html) was used to design due to its developer friendly nature, assets are easily exported for production whilst retaining original dimensions. In hindsight this could have been one of the reasons why it was difficult to stick to creating mobile first designs. The fact that many desktops first websites had been previously designed with this program may have made it harder to break old habits. Another way or averting old habits could have been to remove the desktop artboards, adding them back once mobile had been created. This would have been a good way to refrain from designing for the desktop too early.

Surprisingly one of the hardest parts of the design process was finding inspiration. The client had clearly stated they would like a website that conveys fitness as their brand. However, it is difficult to just go away and begin designing without a clear direction. More often Designers will habitually seek inspiration from pre-existing related designs (Bonnardel, 1999). By using Behance (https://www.behance.net/) it was possible to begin searching for work falling under the category of ‘fitness’ to begin drawing inspiration (See figure 7).

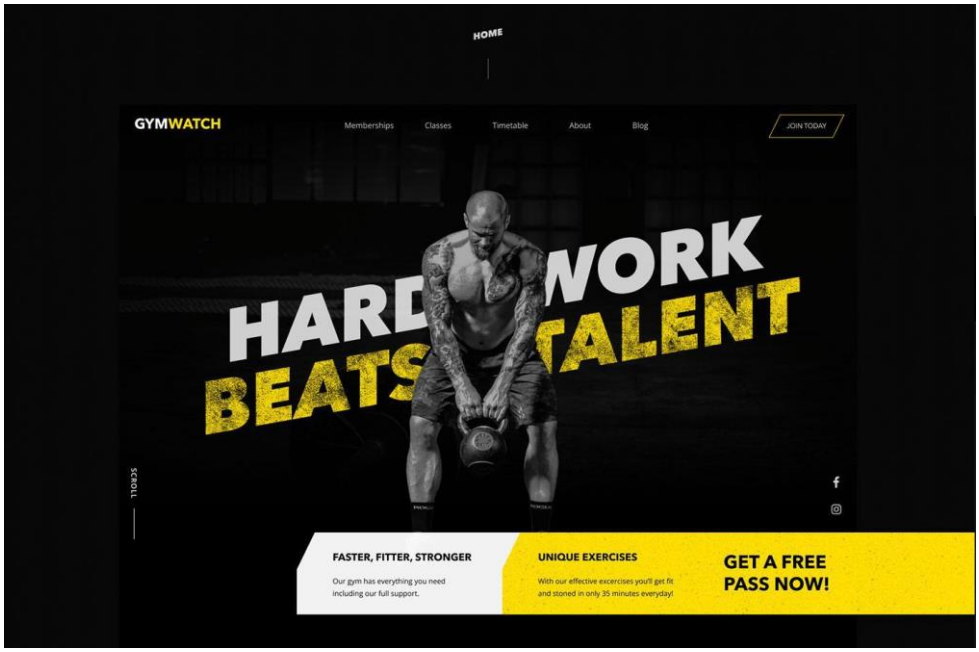

Figure 7 - *This was a key inspiration for the application hero header, the contrast in colours with a hero image in the middle whilst also being simplistic was a very striking design style.*

Figure 8 - *This was also a strong piece of influential design, the use of white chalk smoke amongst a dark black background was successfully carried over into the application design.*

The key focus when looking for inspiration was bright contrasting colours amongst a dark background. The idea being to symbolise hard work and grit strength. Both these projects on Behance proved to be a great inspiration.

Development

A strong foundation had been created thanks to a well-planned tech-stack coupled with TailwindCSS (https://tailwindcss.com). With the front and back-end setup and high-fidelity designs created the development process could now begin. The application was broken down into sections working with a top down approach. The first section to develop was the site's hero header. Two very distinct features found across most mobile apps are loading icons and splash screens. These are both great ways to convey a native-like feel whilst deferring content as it loads, this simultaneously keeps a user engaged before content can be displayed. The integration of a splash screen and a loading icon was very important. Not just to create a native feel but also to increase user experience, the loading of content should happen in the background and be finished by the time the application becomes interactive. A problem that was faced was how to incorporate this loading icon, specifically how to dynamically display it whilst content is loading. This was originally developed to display until the API fetch had been processed. However, this fetch occurred so quickly that images were still loading once the app had been revealed. This is not a good experience; images should not be loading once the application has become interactive. The header image of the athlete was the largest asset on the page in turn creating the largest data payload. To overcome this challenge the loading icon would need to appear based on whether the header image had loaded (See Figure 9).

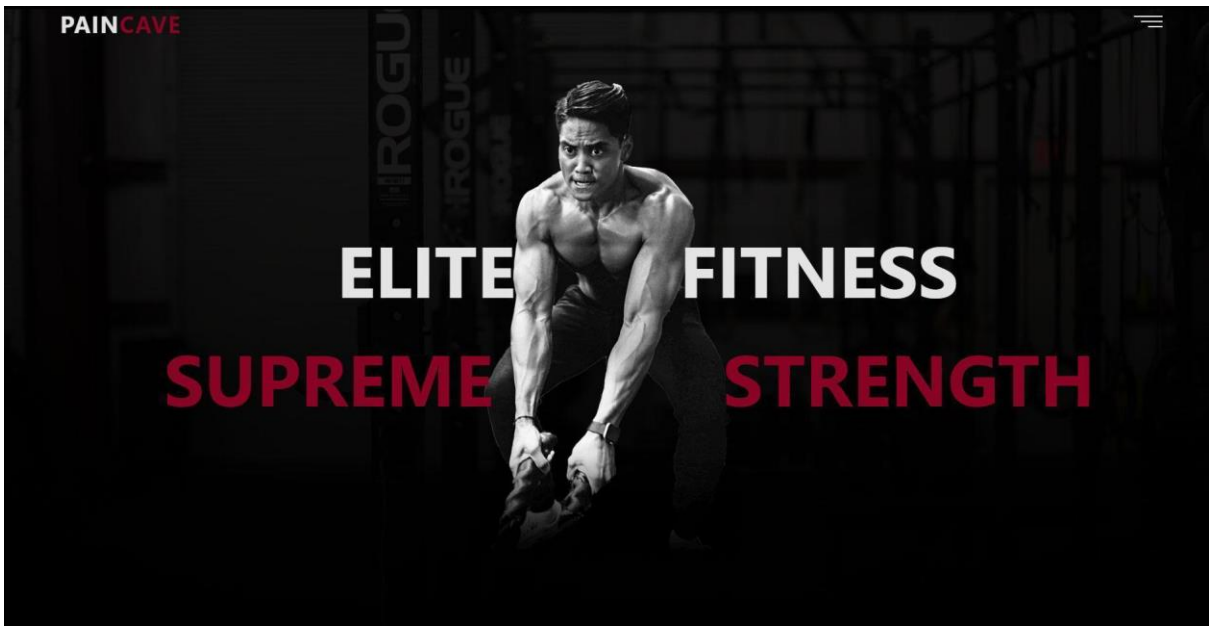

Figure 9 - *The athlete image needed to be a PNG in order to maintain some transparency below the graphic. However, since this file type is generally larger the image would take slightly longer to load so the logical thing to do was base the loading icon animation on whether this image has loaded or not.*

```
{/* Using the import to set a src image to be checked for load | adjusting state once loaded */}
<BackgroundImageOnLoad
    src={athleteImg}
    onLoadBg={() =>
      this.setState({
      bgIsLoaded: true
    })}
    onError={err => console.log('error', err)}
  />
```

Figure 10 - *Figure 10 - Creating the loading animation: By using an external node module called 'BackgroundImageOnLoad' it was possible to specify a source to check against. The image of the athlete in the hero of the app was set as the 'src'. Once this asset had loaded the state 'bgIsLoaded' was set to true.*

```
// conditional statement to check wether the state has been updated from the image load
if((this.state.bgIsLoaded === true)) {
  SpinnerHandler = 'hidden'
  body.classList.remove('fixed', 'w-full')
} else {
  SpinnerHandler = ''
};
```

Figure 11 - *Figure 11 - Once the state has changed to true an if statement will check this to change the 'SpinnerHandler' variable to a value of 'hidden' if the state is indeed true.*

```
{/* The loading spinner dynamically displayed by the 'SpinnerHandler'
variable controlled based on the loading of the athlete image */}
<div className={SpinnerHandler}>
  <Spinner
  bgState={this.state.bgIsLoaded}
  />
</div>
```

Figure 12 - *By wrapping the loading spinner component in a div with the 'SpinnerHandler' variable as its 'className' the display can be toggled based on the 'bgIsLoaded' state. This variable will change to 'hidden' once the 'athleteImage' has loaded, this will then hide the spinner, giving the user the message that the application has loaded.*

Incorporating loading icons when a page is loading to capture and maintain the user's attention is a baseline criterion of a PWA. Although the application has technically loaded, placing a loading screen over the page makes it possible to defer content until every element has finished loading too. Since the CMS was abstracted away, sitting on a separate URL there needed to be a way to grab this data using the WordPress REST API. The solution was an integrated API call to fetch the data in JSON format. All assets and data across the site were pulled from the CMS using

this API call. Taking advantage of the ‘res.json’ method is a convenient way of handling JSON data (Mardan, 2014, P. 89) (See Figure 13).

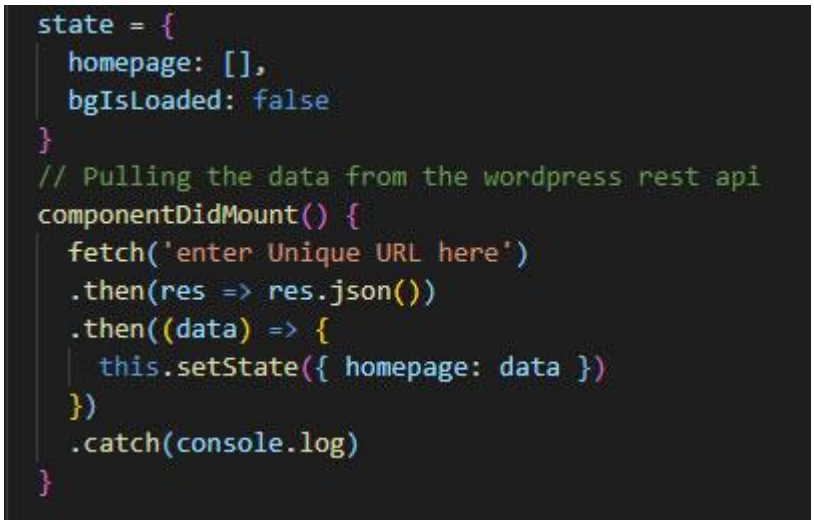

Figure 13 - *‘componentDidMount’ is an important function found within React, when the component, in this case the app, mounts onto the HTML element specified (gets loaded into the DOM) it becomes possible to run JavaScript functionality. A fetch call to grab the API was run within this and ‘res.json’ was used to format the data. Once the data has been formatted it is stored in the ‘homepage’ state. From here the data can be passed around the whole app to different components for use.*

No more issues were encountered as development progressed. A strong foundation of experience coupled together with the speed of TailwindCSS created a smooth build process. A conscious effort was made to keep the application minimal, refraining from packing too much functionality in, a common scenario that occurs when developing for a desktop application. An application saturated by too many functions makes scalability more challenging. When it comes to smaller screens, space is a premium. As mentioned before old habits creeped through into design. Therefore, this carried over into the development process as well, the application progressively transitioned toward desktop-first development. As a result, the process of condensing content down to mobile becomes significantly more challenging. A big lesson was learnt at this point: It is far easier to develop a complex website on desktop, so this should be developed last. The mobile first approach operates on the fact that if a website can look and function well on mobile, it should function on a larger screen with minimal difficulty.

A key characteristic that makes up a PWA is an ‘App-like’ feel made up by app-style interactions and navigations (Lindley, 2019). The inclusion of a splash screen and installable icon are the first steps to achieving this (see Figure 14).

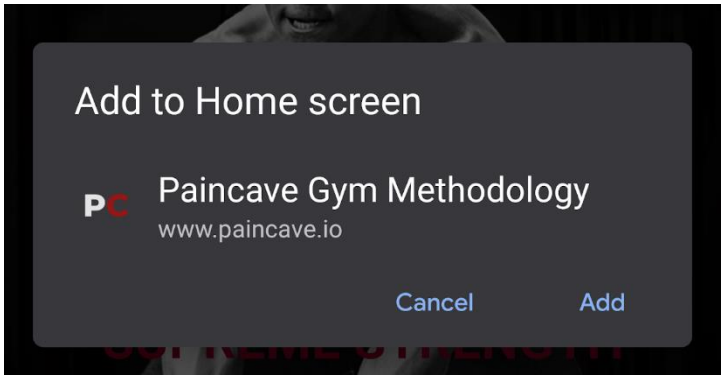

Figure 14 - *The option to add the application to the home screen when viewing in the browser.*

Figure 15 - *The desktop application icon once installed to the home screen.*

This is also where UI and UX go hand in hand to create said app-style interactions. One detail that appears regularly across Apps is a fade in effect of each element once the Application has loaded. By hooking into the state that controlled the display of the loading icon, the elements could fade in once the application had subsequently loaded (See figure 16).

```
${this.props.state.bgIsLoaded ? 'fade-in-extended': ''}`}
```

Figure 16 - *Using a ternary operator it is possible to add a class to fade in each element if the state 'bgIsLoaded' is true. This is the state which turns to true once the header athlete image has loaded.*

Additionally, since the application is a Single Page Application (SPA) a nice addition proved to be a progress bar at the top of the page. Enhancing the user experience by hand holding the user throughout the website, maintaining engagement by displaying current progress (See Figure 17).

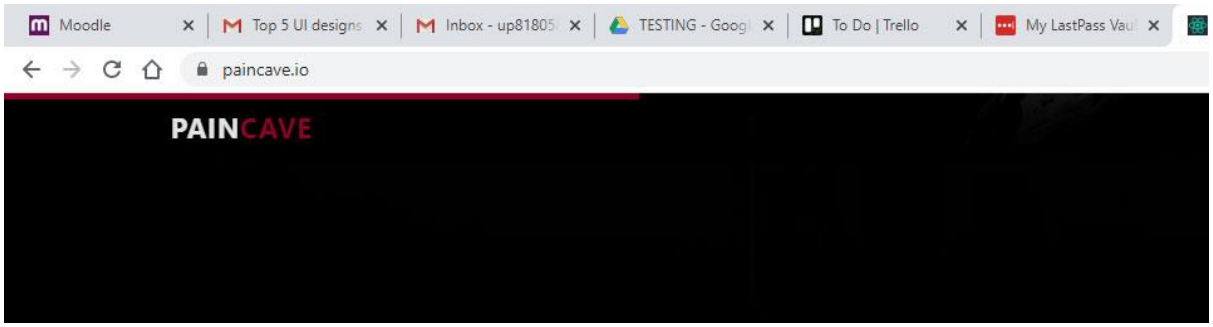

Figure 17- *The red line acts as a tracker that signifies the user's location as they scroll down the page.*

Testing

With the application build complete there were a few key things that needed to be tested. Google's lighthouse is an open source tool that can be used to test web pages (https://developers.google.com/web/tools/lighthouse). By using this tool, it was possible to test the most important thing, whether the application indeed met the criteria of a PWA (See figure 18).

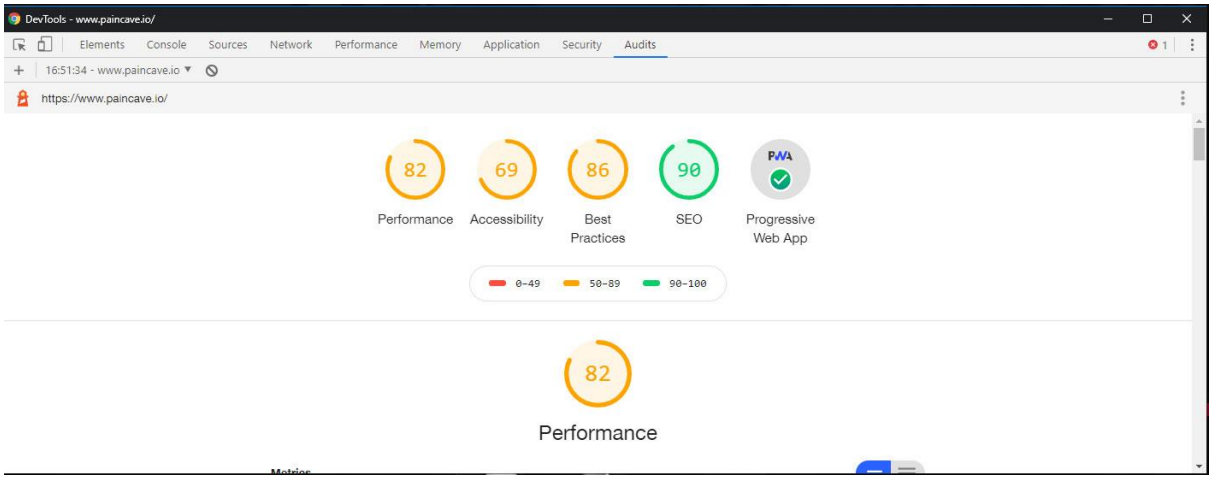

Figure 18 - *As shown here the application successfully fulfils the criteria of a PWA based on the check sign.*

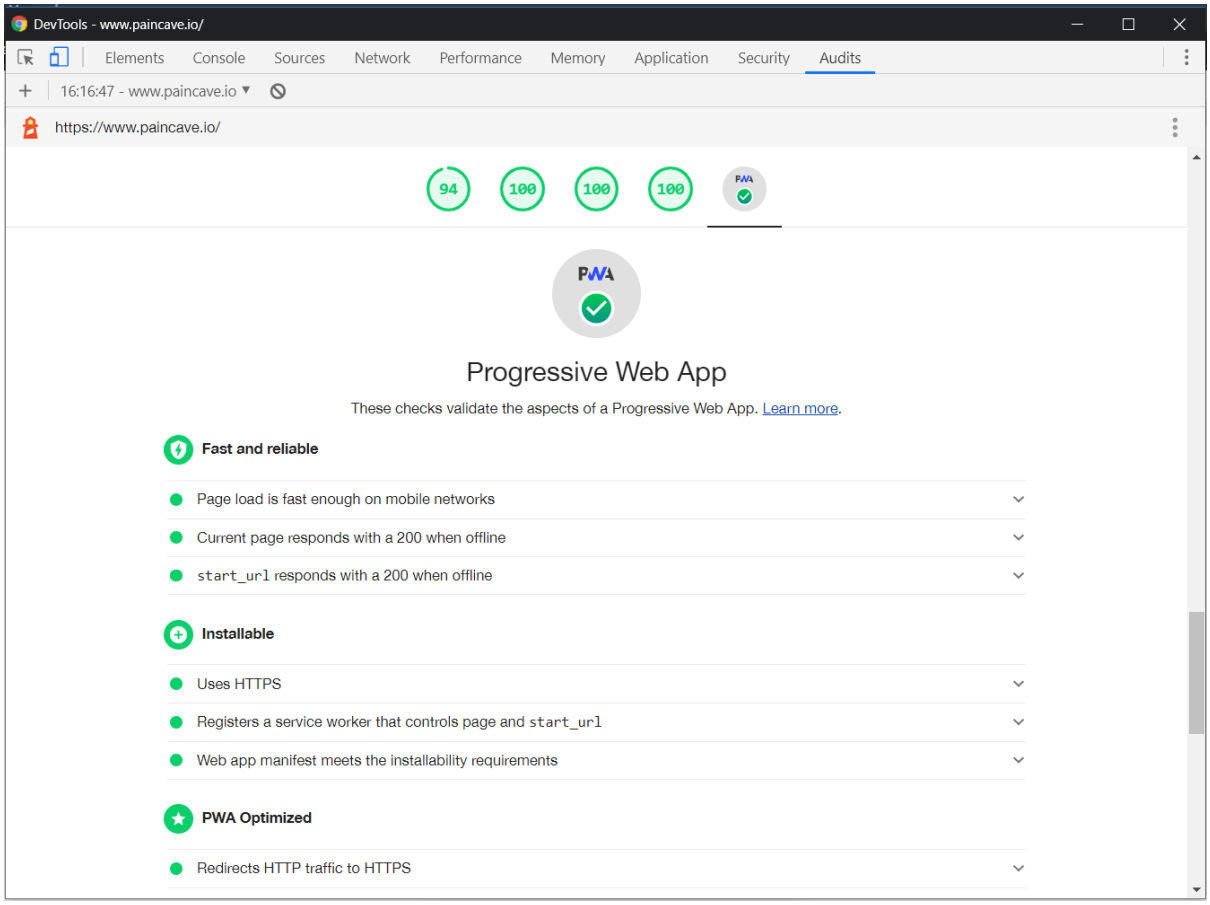

Figure 19 - *A more comprehensive breakdown shows each individual component of the application test which makes up the criteria needed to have the application be classed as a PWA.*

With the application fulfilling all requirements the next step was to improve the overall performance. The Performance, Accessibility and Best Practices were all still in the orange zone. High levels of performance are a critical requirement for any website to be successful, slow websites lose traffic and sales whilst fast sites rank higher in Google's search engine (Smith, 2012, P. 24). Paying close attention to the score given by Google is a sure-fire way to increase a web applications chance of success. Especially when a developer's goal should always be to provide the client with an application to the highest standard.

Performance

Performance was lacking for two reasons: Firstly, the inclusion of the TailwindCSS framework can massively increase the size of the main CSS file since it injects every single utility class into this one stylesheet. The solution was the inclusion of a module called PurgeCSS (https://purgecss.com/). This module checks every JS/HTML file in the application then proceeds to remove any class from the stylesheet that is not in use, greatly reducing the file size. The last issue came down to images that were not

sized properly. Thanks to Google's audit feature it was possible to see which images were being displayed at a suboptimal size. The solution was to resize each image, reducing file size in order to display these at their native size.

Accessibility

The cause of a lower accessibility score came down to a few elements across the site missing alt attributes. Additionally, a few buttons did not have an appropriate name which also lowered this score. When it comes to computers and software applications, accessibility relates to the usability of a computer system by people with disabilities (Paris, 2005, P. 293). Sifting through the code base and including these attributes pushed the score back up. Impaired users will now be able to hover over these elements and receive a voiced description of what they are.

Best Practices

The best practice score was being affected since there was a 404 error in the console of the live website. There should be no errors present in a website ready for production (See Figure 20).

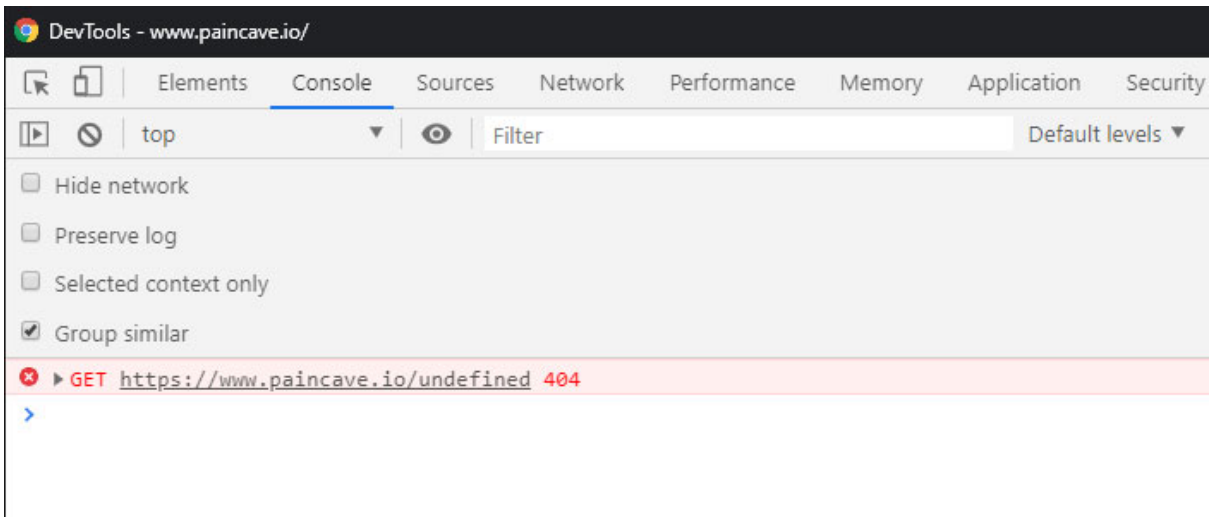

Figure 20 - *As seen a 404 error of undefined was being returned. This was an empty variable returning undefined since a value had not yet been assigned. Once the API call returns data it is stored in this variable. So, until then the solution is to set it to an empty string, giving it a temporary value until the API call returns new data.*

The solution was to set the variable to an empty string until it had been populated by the API to prevent it returning undefined (See Figure 21).

```
let headerImg = ''
let athleteImg = ''

this.props.data.forEach(element => {
headerImg = element.acf.header_image
athleteImg = element.acf.athlete_img
});
```

Figure 21 - *Assigning the variables an empty string until they had been reassigned the relative content in the 'forEach' loop.*

SEO

Search Engine Optimization (SEO) is the process of fine tuning a website / application to maximize its findability and in turn its ranking. The SEO score was adequate from the start. Thanks to the clever design of Create React App the SPA application was very SEO friendly without the need to change anything. The HTML title and description meta tags are the most relevant aspects of increasing a website visibility (Weideman & Kritzinger, 2003, P. 3). By ensuring the application included a descriptive title and accompanying Metadata the SEO score could be increased to a desirable value (See figure 22).

```
<meta
  name="Paincave elite performance method application"
  content="Paincave official application, find out about our gyms, athletes and official merchandise"
/>
<title>Paincave elite performance method</title>
```

Figure 22 - *Both the Metadata and title tags are very important aspects of successful SEO. The inclusion of both will significantly increase an application's SEO score.*

Once each key area had been tested and the resulting issues fixed the performance rating rose significantly. Providing the client with a website that performs to the best of its abilities (See figure 23).

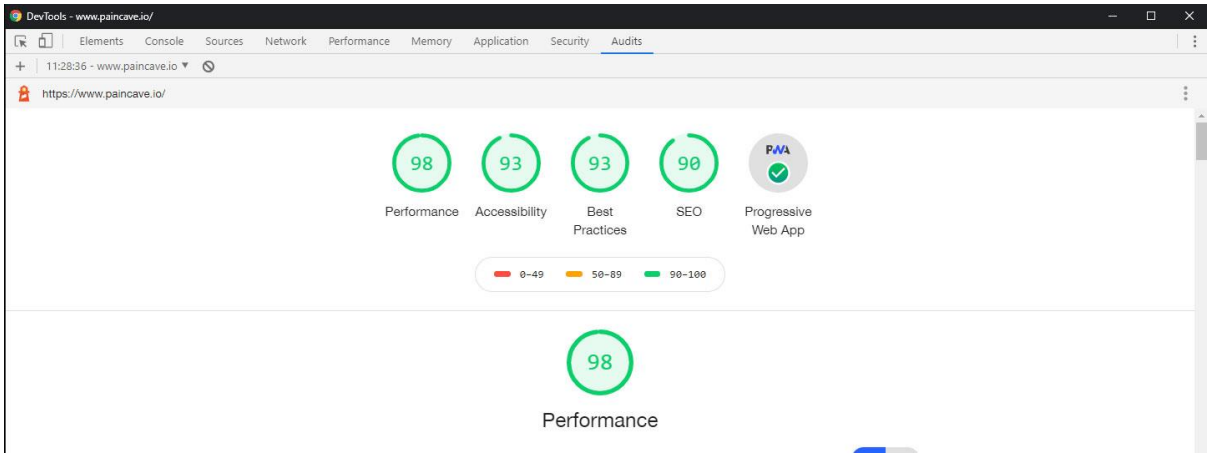

Figure 23 - *As seen here the overall performance of the application has been significantly improved to reflect the standards expected from a professional website.*

User Testing

The use of external participants to test an application can be highly beneficial. User testing analysis arguably should be used as a method for guiding design improvements (Tan & Bishu, 2002, P. 1258). By receiving feedback from the intended audience, it is far easier to make informed design changes with users in mind. This would have been a very important step in the testing process for the client application. The original plan was to sit down with a range of participants with the goal of collecting between 5-10 sets of results. Having the users use the application for 10 minutes, carrying out several tasks to then receive feedback. This would have been an effective way to identify any missed bugs or pitfalls in the design. However, due to the ongoing COVID-19 pandemic it has not been possible to carry out any face to face testing. The solution has been to increase the scrutinization of the testing phase, placing oneself in the user's shoes to make this first-hand testing as effective as possible.

Summary

The development process was not so difficult. Coming into this with a good understanding of JavaScript helped immensely with getting to grips with React. As mentioned previously one of the identified downsides of choosing to build an application natively would be the need to learn a new technology and the time this would take. This discussion has shed light on just how quick and advantageous it can be to develop a PWA using technologies where prior experience is held. New technologies are being created and introduced very rapidly and React has now created a framework to cater for cross platform native development called React Native (https://reactnative.dev/). React Native is being adopted by more and more organizations worldwide, it is a native scripting framework used for creating cross-platform mobile applications (Niclas & Tomas, 2016, P. 6). React Native allows for cross-platform development using a single codebase. Therefore, if the requirement was a native application across all mobile devices React Native may be a better

solution. Since React Native is also built upon React it would therefore be a great option for a developer with JavaScript experience. With the popularity of JavaScript this may well be worth investing in. Ultimately it could save a company time and money removing the need to employ a new developer or train an existing one.

When it comes down to it there are many different use cases for each technology whether it be Native, PWA or Cross-Platform Native. If the client wanted an application exclusively for Android then development would have to be done using either Java or Kotlin (https://kotlinlang.org/). Kotlin is a new general purpose, free, open source, statically typed “pragmatic” programming language (Infoworld, 2020). It was initially designed for the JVM (Java Virtual Machine) and Android. Therefore, is a more than capable alternative to Java for creating Android apps. Alternatively if the requirement was for an application exclusively for Apple phones then this would need to be coded using either Swift or Object-C. Swift is a quickly growing programming language created by Apple (https://www.apple.com/uk/swift/). The fact it is developed by Apple for their devices would arguably make it the most compatible language. No two client projects are the same, each one will have its own requirements and individual use cases. For this reason, there will be a need to research programming languages and choose the one most appropriate under the circumstance. The analysis of the development process during this discussion has created a very strong understanding on what it is like to develop a PWA using React. This will help build up an informed conclusion as to whether the use of React to build a PWA is a viable or even a better alternative to conventional methods of native application development.

Conclusion

A large amount has been learnt regarding this new technological approach that is having a drastic, but positive impact on the web development industry. Through a process of first-hand research and development a much more informed opinion can now be given in order to form an answer to the most critical question that's been surrounding PWA’s since their creation back in 2015; Are Progressive Web Apps a good alternative to Native Apps?. This conclusion will aim to answer this question with the intention of helping other developers who are unsure as to whether they should use this technology themselves. By providing an informed decision based on various scenarios others will be able to use this in-depth case-study and the subsequent conclusion as guidance.

There are so many different possible use cases as to why a PWA may or may not be needed. If the requirement is unclear it is important to identify what the motives are for certain technologies and approaches. If the requirement is a purely native application exclusively for a single mobile manufacturer then the requirement for a PWA is effectively removed altogether. Under this circumstance the suggestion

would be made to take advantage of a programming language that caters for the device in question. This would be advantageous for many reasons including performance. Alternatively, if for example the requirement was to transform an outdated website into a cross-platform application whilst still maintaining all customers that use the website in a browser. Then the suggestion of a PWA to fulfil these requirements could be made. It is important to inform clients of the power that PWA's can provide. The possibility to create a website capable of fulfilling previous requirements, whilst also being accessible on a mobile device and providing all the benefits that native apps promise.

An assumption could be made that a web developer is currently employed to maintain and develop the website of an agency looking to transition from a conventional platform into a cross-platform experience. In this case the benefit of developing a PWA would be a minimised learning curve. Using languages such as JavaScript maintain familiarity for web developers. Provided the developer is proficient with code it would not be too difficult to develop high levels of performance to match that seen on native platforms. As seen previously during the development stage it is more than possible to make a technically indistinguishable PWA from a native app using service workers, web app manifests etc. A client looking for a PWA which is indistinguishable by look is where the real test comes into play, as was experienced first-hand. The design process was the one which stood out as one of the most important stages. Without creating quality designs that successfully convey a native feel the application has failed before it has even begun development. Reflecting, it was clear how previous experience of developing websites negatively impacted the ability to create a native feel. Based on this the developer or designer's ability to design for native devices should be closely assessed. This is an area where a native designer would be highly advantageous. There is increased risk that the application will not achieve a native feel using an individual who only holds experience designing for the web. A solution to this would be to outsource a freelancer or employ a native designer.

A PWA can indeed be a very good alternative to a Native Application. But only under circumstances that truly justify the need for one. This can range from; a requirement due to business related factors such as monetary incentives to save on costs by serving the same code base across different platforms; or from a skills perspective. The developer creating the application may have no experience of coding for Native applications. Therefore, one could conclude that there will be more chance of a high-quality build by using technologies of expertise. This again would save time and money without the need to learn new native languages or tools. It is important to stress once again just how easy an individual with a background in web development can fall into old habits of designing and developing for the web. This indicates just how important a rigorous design process can be when working under conditions that normally facilitate the development of websites. If designs fall short at successfully conveying a native feel, then the development of these designs will no doubt follow

down the same path. It would be reasonable to assume that native developers know certain techniques that work in conveying an important 'native' feel and experience. Further assumptions can be made that native design teams can create designs that are unquestionably for a native device based on how they look. The consideration to hire individuals with native expertise as a stepping stone for a PWA may not be a bad one.

Ultimately, the main conclusion is that PWA's are and should be used as viable alternatives to native applications under the right circumstances. For these circumstances to be achieved there need to be a few prerequisites in place beforehand. For example, the designer or developer must have experience or gone through a process of learning how to design for native applications. As was experienced first-hand, the development process can become a waste of time and resources if the preceding design process has fallen short in successfully conveying a native like feel. Therefore, without the prerequisite of a competent, native set of designs the development of the application may as well never go ahead, it will be destined to fall short of the main goal if the designs have already done so. In addition to this, the developer must have a clear understanding and working practice of implementing mobile-first development. Mobile and tablet usage has exceeded desktop and actively drives the shift in industry to mobile-first development. Without a strong theoretical and practical understanding of responsive, mobile-first development the application will not scale appropriately or efficiently between screen sizes.

Limitations

One limitation of this study is that it could be interpreted as slightly one sided. This is for the fact that the only application developed throughout was a PWA. A different approach could have been to this same application as a PWA and a native application. In doing so it would be possible to create a direct comparison between both applications, one served as a PWA and the other purely native. If the application looked the same across devices, then a conclusion could be drawn as to whether a PWA will look the same based on a direct comparison. Given the chance again more research would have been undertaken around the area of native apps. A large amount was aimed at PWA’s which built up a very comprehensive understanding. Pointing more attention towards researching native apps could have created greater opportunities for comparison based on greater knowledge of both sides.

Nonetheless, it is time for developers across the board to begin sharpening their skills by giving mobile devices a priority. PWA’s have been the biggest driving force towards this shift. The web development industry has swiftly taken to these adjustments and upheld their reputation for being one of the quickest evolving industries. The ability to adapt quickly alongside a keen level of openness toward

new practices and change will be the new driving force behind successful developers in this mobile-first generation.

References

Aggarwal, S. (2018, March). Modern Web-Development using ReactJS. Retrieved from **http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf**

Ater, T. (2017). Building Progressive Web Apps: Bringing the Power of Native to the Browser (T. Ater, Ed.). Sebastopol: O'Reilly Media. (Original work published 2017)

Blischak, J, D. Davenport, E, R. Wilson, G. (January 19, 2016). A Quick Introduction to Version Control with Git and GitHub. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4718703/pdf/pcbi.1004668.pdf

Bonnardel, N. (1999). Creativity in design activities: The role of analogies in a constrained cognitive environment. In Proceedings of the 3rd Conference on Creativity & Cognition (pp. 158–165). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=317589

Bryan, J. (2018). Excuse Me, Do You Have a Moment to Talk About Version Control? Retrieved from https://amstat.tandfonline.com/doi/pdf/10.1080/00031305.2017.1399928?needAccess=true

Břoušek, P. (2017). Evaluation and usage of Google Progressive Web Apps technology. Retrieved from **https://is.muni.cz/th/433364/fi_b/bachelor-thesis-pavel-brousek-pwa.pdf**

Beshir, A. (2016, August). Cross-platform development with React Native. Retrieved from **http://www.diva-portal.org/smash/get/diva2:971240/FULLTEXT01.pdf**

Burroughs, B. (2017, May). YouTube Kids: The App Economy and Mobile Parenting. Retreived from **https://journals.sagepub.com/doi/pdf/10.1177/2056305117707189**

Biørn-Hansen, A., Majchrzak, Grønli, T-A., (2017). Progressive Web Apps: The Possible Web-native Unifier for Mobile Development. Retrieved from **https://www.scitepress.org/Papers/2017/63537/63537.pdf**

Cardieri, G. A. & Zaina, L. M., (2018, October 22 - 26). Analyzing User Experience in Mobile Web, Native and Progressive Web Applications: A User and HCI Specialist Perspectives. Retrieved from **http://delivery.acm.org/10.1145/3280000/3274201/a9-de_andrade_cardieri.pdf?ip=148.197.248.60&id=3274201&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2E9AADD49FC5ECC8B0%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1575485844_fc738fd9d522669de9b268d46b7c723d**

Cabot, J. (May/June, 2018). WordPress: A Content Management System to Democratize Publishing. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8354434

Čechák, D. (2017). Using GraphQL for Content Delivery in Kentico Cloud. Retrieved from https://is.muni.cz/th/qm0cs/thesis.pdf

Dabbish, L. Stuart, C. Tsay, J. Herbsleb, J. (2012). Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. Retrieved from https://dl.acm.org/doi/abs/10.1145/2145204.2145396

Daxx, (2020). How Many Software Developers Are in the US and the World?. Retrieved from https://www.daxx.com/blog/development-trends/number-software-developers-world

Developer.Apple. (2020). Human interface guidelines. Retrieved from https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/adaptivity-and-layout/

Domos, S. (October, 2017). Progressive Web Apps with React. Retrieved from https://books.google.co.uk/books?hl=en&lr=&id=8RhKDwAAQBAJ&oi=fnd&pg=PP1&dq=building+a+progressive+web+app+with+react&ots=6lXlloCExo&sig=iLger57rwN8q_GznJZWUOfV-q_o&redir_esc=y#v=onepage&q&f=false

Fransson, R., & Driaguine, A. (2017). Comparing Progressive Web Applications with Native Android Applications. Retrieved from **http://www.diva-portal.org/smash/get/diva2:1105475/FULLTEXT01.pdf**

Gackenheimer, C. (2015). Introduction to React: Using react to build scalable and efficient user interfaces. Retrieved from **https://link.springer.com/content/pdf/10.1007%2F978-1-4842-1245-5.pdf**

Gerring, J. (2004, June 21). What Is a Case Study and What Is It Good for? Retrieved from **https://www.cambridge.org/core/services/aop-cambridge-core/content/view/C5B2D9930B94600EC0DAC93EB2361863/S0003055404001182a.pdf/what_is_a_case_study_and_what_is_it_good_for.pdf**

Gustafsson, N. (2019). Developing modern web applications with the React library and WordPress. Retrieved from **https://www.theseus.fi/bitstream/handle/10024/172161/Degree_Thesis_NatalieGustafsson_2019.pdf?sequence=2&isAllowed=y**

Golhar, V, R., Vyawahare, A, P. Borghare, H, P. Manusmare, A. (2016). Design and Implementation Of Android Base Mobile App For An Institute. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7755391

Hiltunen, M. (2018, November 13th). Creating multiplatform experiences with Progressive Web Apps. Retrieved from **https://www.theseus.fi/bitstream/handle/10024/157245/Hiltunen_Mira.pdf?sequence=1**

Infoworld. (2020). What is Kotlin? The Java alternative explained. Retrieved from https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html

Johannsen, F., (2018). Progressive Web Applications and Code Complexity. Retrieved from **http://www.diva-portal.org/smash/get/diva2:1230204/FULLTEXT01.pdf**

Kuss, D. J., Kanjo K. E., Crook-Rumsey, M., Kibowski, F. Wang, Y. G., Sumich, A. (2018, January 08). Problematic Mobile Phone Use and Addiction Across Generations: The Roles of Psychopathological Symptoms and Smartphone Use. Retrieved from **https://link.springer.com/content/pdf/10.1007%2Fs41347-017-0041-3.pdf**

Kelley, T., & Bertenthal, B, I. (13, June, 2016). Attention and past behavior, not security knowledge, modulate users' decisions to login to insecure websites. Retrieved from https://www.emerald.com/insight/content/doi/10.1108/ICS-01-2016-0002/full/pdf?title=attention-and-past-behavior-not-security-knowledge-modulate-users-decisions-to-login-to-insecure-websites

Kumar, A., & Singh, R, K. (2016). COMPARATIVE ANALYSIS OF ANGULARJS AND REACTJS. Retrieved from https://s3.amazonaws.com/academia.edu.documents/54960538/148051944230.1245.pdf?response-content-disposition=inline%3B%20filename%3DCOMPARATIVE_ANALYSIS_OF_ANGULARJS_AND_RE.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20200304%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200304T212540Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=87926cc331b13a8565eeda39acd925140b2e1cc4a78bcc4621b0d121976debb9

Liang, J. Jiang, J. Duan, H. Li, K. Wan, T. Wu, J. (2014). When HTTPS Meets CDN: A Case of Authentication in Delegated Service. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6956557

Liu, Y., Liu, X., Ma, Y., Liu, Y., Zheng, Z., Huang, G., Brian Blake, M., (2015). Characterizing RESTful Web Services Usage on Smartphones: A Tale of Native Apps and Web Apps. Retrieved from **https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7195587**

Lange, K. (2016). The little book on rest services. Retrieved from http://www.kennethlange.com/books/The-Little-Book-on-REST-Services.pdf

Lindley, C. (2019). Front-end Developer Handbook 2019.

Mullins, C. (2015, July 16 - 17). Responsive, mobile app, mobile first: untangling the UX design web in practical experience. Retrieved from **http://delivery.acm.org/10.1145/2780000/2775478/a22-mullins.pdf?ip=148.197.248.60&id=2775478&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2E9AADD49FC5ECC8B0%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1575491732_8e4dc3d2a3fae466dfe63f06301f5b12**

Mohorovičić, S. (2013, May). Implementing responsive web design for enhanced web presence. Retrieved from **https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6596440**

Mohammadi, R. (2010). Performance, and Efficiency based Comparison of Angular and React in a case study of Single page application (SPA). Retrieved from https://s3.amazonaws.com/academia.edu.documents/62117344/monograph20200216-17036-1haeii6.pdf?response-content-disposition=inline%3B%20filename%3DPerformance_and_Efficiency_based_Compari.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20200320%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200320T132059Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=e696d0dc2dff07fdc3984ff826570e0e0a3220a9f599b92549518c9a41ae37fc

Minna, K. (2019). Enhancing E-Commerce with Modern Web Technologies. Retrieved from https://www.theseus.fi/bitstream/handle/10024/161196/Kankaala_Minna.pdf?sequence=2&isAllowed=y

Mohammadjafari, M. Ahmed, S. Dawal, S. Z. Md, Zayandehroodi, H. (29 November 2010). The importance of project management in small- and medium-sized enterprises (SMEs) for the development of new products through E-collaboration. Retrieved from https://www.researchgate.net/profile/Shamsuddin_Ahmed_Mohammed_Shahadat/publication/252063857_The_importance_of_project_management_in_small-_and_medium-sized_enterprises_SMEs_for_the_development_of_new_products_through_E-collaboration/links/542aceba0cf29bbc126a6f4d/The-importance-of-project-management-in-small-and-medium-sized-enterprises-SMEs-for-the-development-of-new-products-through-E-collaboration.pdf

Messenlehner, B., & Coleman, J. (December, 2019). Building Web Apps with WordPress. Retrieved from https://books.google.co.uk/books?hl=en&lr=&id=gXTDDwAAQBAJ&oi=fnd&pg=PR4&dq=wordpress+rest+api&ots=J4yniCOkRX&sig=Ar08acsE_9YeYZiViMQhTf3i1_s&redir_esc=y#v=onepage&q&f=false

Mardan, A. (2014). Express.js Guide: The Comprehensive Book on Express.js. Retrieved from https://books.google.co.uk/books?hl=en&lr=&id=5eGRAwAAQBAJ&oi=fnd&pg=PP6&dq=res.json&ots=njtgv_cnMH&sig=QzS8I0i8oeNWdefYcStK91DDCeo&redir_esc=y#v=onepage&q=res.json&f=false

Niclas, H., & Tomas, V. (2016). Effects on performance and usability for cross-platform application development using React Native. Retrieved from http://www.diva-portal.org/smash/get/diva2:946127/FULLTEXT01.pdf

Nguyen, L. (September, 2019). BUILDING E-COMMERCE SOLUTIONS WITH WOOCOMMERCE. Retrieved from https://www.theseus.fi/bitstream/handle/10024/261146/Linh_Nguyen.pdf?sequence=2&isAllowed=y

Oulasvirta, A. Tamminen, S. Roto, V. Kuorelahti, J. (April, 2005). Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. Retrieved from https://dl.acm.org/doi/abs/10.1145/1054972.1055101

Oliveira, W. Oliveira, R. Castor, F. (2017). A Study on the Energy Consumption of Android App Development Approaches. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7962354

Pande, N. Samal, S, P. Somani, A. Kakkirala, V. (2018). Enhanced Web Application and Browsing performance through Service-worker Infusion Framework. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8456349

Paris, M. (1 July 2005). Website accessibility: a survey of local e-government websites and legislation in Northern Ireland. Retrieved from https://link.springer.com/content/pdf/10.1007/s10209-003-0081-7.pdf

Pew Research Center (2018). Demographics of mobile device ownership and adoption in the United States. Retrieved from http://www.pewinternet.org/fact-sheet/mobile/

Perrin, A., & Jiang, J. (2018). About a quarter of U.S. adults say they are ‘almost constantly’ online. Retrieved from http://www.pewresearch.org/fact-tank/2018/03/14/about-aquarter-of-americans-report-going-online-almost-constantly/

Rosen, D, E., & Purinton, E., (2004, July). Website design: Viewing the web as a cognitive landscape. Retrieved from **https://reader.elsevier.com/reader/sd/pii/S0148296302003533?token=817C373F2B87DF8A6FE7**

8AAC84ED55A86CF9B940D5292DA72F243EA6A6C95B751DD3161C7CA8DA000D2821E970D61F89

Saks, E. (2019). JavaScript frameworks: Angular vs React vs Vue. Retrieved from https://www.theseus.fi/bitstream/handle/10024/261970/Thesis-Elar-Saks.pdf?sequence=2

Sun, H., Bonetta, D., Humer, C., Binder, W., (2018, February 24 - 25). Efficient dynamic analysis for Node.js. Retrieved from **http://delivery.acm.org/10.1145/3180000/3179527/cc18-p69.pdf?ip=148.197.248.60&id=3179527&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2E9AADD49FC5ECC8B0%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1575483480_abf0f280b700ab859386ecf385a96535**

Smith, P. (2012). Professional Website Performance: Optimizing the Front-End and Back-End. Retrieved from https://books.google.co.uk/books?hl=en&lr=&id=MHLJlUfXV4QC&oi=fnd&pg=PR23&dq=website+performance,+accessibility+and+best+practices&ots=81Huo6H2AD&sig=E8koDoW0dwbAiXvyDI7ir-ihO1w&redir_esc=y#v=onepage&q=website%20performance%2C%20accessibility%20and%20best%20practices&f=false

Tan, W-S., & Bishu, R. R. (2002). Which is a Better Method of Web Evaluation? a Comparison of User Testing and Heuristic Evaluation. Retrieved from https://journals.sagepub.com/doi/pdf/10.1177/154193120204601404

Tahirshah, F, S. (2019, June) Comparison between Progressive Web App and Regular Web App. Retrieved from http://www.diva-portal.org/smash/get/diva2:1334458/FULLTEXT02.pdf

Tim, A. M., (2018). Progressive Web Apps: the Definite Approach to Cross-Platform Development? Retrieved from **https://aisel.aisnet.org/hicss-51/st/mobile_app_development/7/**

Uzayr, B, S. (2016). Learning WordPress REST API. Retrieved from https://books.google.co.uk/books?hl=en&lr=&id=-82qDQAAQBAJ&oi=fnd&pg=PP1&dq=wordpress+rest+api&ots=3fsZZHX4XC&sig=zcu0q3Sj-KiFnkcFLGphG4kEEj4&redir_esc=y#v=onepage&q=wordpress%20rest%20api&f=false

Vainikka, J. (16, May, 2018). Full-stack web development using Django REST framework and React. Retrieved from https://www.theseus.fi/bitstream/handle/10024/146578/joel_vainikka.pdf?sequence=1

White, J., (2013, Jan 31). Going native (or not): Five questions to ask mobile application developers. Retrieved from **https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3575060/pdf/AMJ-06-07.pdf**

Weber A, K., Wesselman N, J., Shyba, O., Seifert, M., (2018, Aug 16). Headless content management system (cms). Retrieved from **https://patentimages.storage.googleapis.com/48/c5/65/631136d73a9913/US20180232782A1.pdf**

Weideman, M., & Kritzinger, W. (2003). Search Engine Information Retrieval: Empirical Research on the usage of Meta Tags to enhance Website Visibility and Ranking of e-Commerce Web Sites. Retrieved from https://shorturl.at/jDZ56

Wittern, E. T, T, A. Yunhui Z, Y. Dolby, J. Laredo, A, J. (2017). Statically Checking Web API Requests in JavaScript. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7985666

Yadav, P., & Barwal, P. N., (2014, November). Designing Responsive Websites Using HTML And CSS. Retrieved from **http://www.ijstr.org/final-print/nov2014/Designing-Responsive-Websites-Using-Html-And-Css.pdf**

Zainal, Z., (2007, June 9). Case study as a research method. Retrieved from **https://jurnalkemanusiaan.utm.my/index.php/kemanusiaan/article/view/165/158**

Glossary of terms

API - Application Programming Interface

CMS - Content Management System

CSS - Cascading Style Sheets

DOM - The Document Object Model

GIT - A distributed version-control system

HTML - Hypertext Markup Language

HTTPS - Hypertext Transfer Protocol Secure

IOS - A mobile operating system created and developed by Apple Inc.

IDE - Integrated Development Environment

JS - JavaScript

JSON - JavaScript Object Notation

Kotlin - Cross-platform, statically typed, general-purpose programming language

LAMP - LAMP stands for Linux, Apache, MySQL, and PHP

Objective-C - A general-purpose, object-oriented programming language

PWA - Progressive Web Application

React – A JavaScript library for building user interfaces.

RESTful API - An application program interface that uses HTTP requests

SQL - Structured Query Language

SPA - Single-page Application

SEO - Search Engine Optimization

Swift - Programming language created by Apple for building apps for iOS

URL - Uniform Resource Locator

UI / UX - User Interface & User Experience

VCS - Version control systems

Appendices

Client Brief:

The Client
A competitive athlete competing within the sport of ‘strong-man’, with a passion for all thing’s fitness.

Project Aim
The aim of the PWA build is to develop and deliver a brand which appeals to more serious athletes. The application should provide an effective and easy to use platform for prospective athletes and customers. Creating a place providing a place to view the brands products, athletes and gyms.

Requirements
The client requires the website to be accessible across all platforms, in a professional and performant manner. Therefore, a progressive web app is the best way to achieve this outcome. By creating a fully responsive PWA which behaves and performs like an application, all users will be able to access the website ultimately increasing the customer outreach for the client.

Proof-of-concept (URL)

https://www.paincave.io

Final Year Project: Preparation (2019) Submission page numbers:

Methodology - 11

Link to proof of concept video and Documentation:

https://drive.google.com/drive/folders/1oVsJ-fMjew192htQ32WWMSdc9hFwOVvq?usp=sharing