



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Investigate the use of Progressive Web Apps as a Cross Platform Mobile Development Approach in Government

by
Gomathi Shivakumar

Supervisor: Dr Kemi Ademoye
Second Marker: Dr Nik Whitehead

Project submitted as part of the requirements for the
award of
M.Sc Software Engineering (Web Development)

January 2020

DECLARATION

I, Gomathi Shivakumar declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : Gomathi

Date : 30/01/2020

FORM OF CONSENT

I Gomathi Shivakumar hereby consent that my Project, submitted in candidature for the M.Sc Software Engineering degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : Gomathi

Date : 30/01/2020

Abstract

From the first commercial device, Apple Newton message pad in 1993 to wearable devices, mobile devices have gone through a huge developmental transition over the years. Mobile technologies, such as tablets and smartphones provide portability and are powerful enough to compete with personal computers in performance [1][2]. With more than 5 billion of the global population owning a smart phone, the web is accessed on mobile phones more than ever as illustrated in figure.1 [3][4]. Governments have recognised the impact of this culture and multiplier effect of technology that creates a new breed of services and devised innovative transformation strategies to deliver their services. However, accessibility, cross-platform limitations, development and maintenance costs are the deterrents for mobile app development. In light of this, it was proposed to investigate if Progressive web apps (PWA) could be the alternative technology for governments to tackle the mobile gap. A literature review on cross-platform development approaches, features of PWA and existing research on PWA and mobile initiatives in governments was conducted. A progressive web app prototype was built for the vehicle enquiries services of the Driver and Licencing Agency in UK as a proof of concept. User research and usability study results of the PWA was conducted across several platforms with user testing and surveys. The results revealed a positive experience with 100% preference on PWA installations and a SUS score of 85.22. One in 12 persons also preferred the government to deliver their services through mobile apps. With major operating systems and browsers now providing enhanced support for PWA and the ease of use and nearly native app experience they provide along with push notifications and offline capability, it was recommended that governments explore PWAs as a cross-platform development approach.

Table of Contents

1	INTRODUCTION.....	1
1.1	DIGITAL GOVERNMENT	1
1.2	MOBILE APPS INDUSTRY	2
1.3	AIMS & OBJECTIVES	3
1.4	DOCUMENT STRUCTURE	4
2	LITERATURE REVIEW	5
2.1	HISTORY OF THE WEB	5
2.2	THE WEB ON THE GO - MOBILE TECHNOLOGIES	6
2.2.1	<i>Native App</i>	8
2.3	CROSS PLATFORM DEVELOPMENT APPROACHES (CPD).....	8
2.3.1	<i>Mobile Website</i>	9
2.3.2	<i>Responsive Design</i>	9
2.3.3	<i>Hybrid Development</i>	10
2.3.4	<i>Interpreted Approach</i>	10
2.3.5	<i>Cross Compiled Approach</i>	11
2.3.6	<i>Accelerated Mobile Page</i>	11
2.3.7	<i>Web Apps</i>	11
2.3.8	<i>WebAssembly</i>	12
2.3.9	<i>Summary</i>	12
2.4	PROGRESSIVE WEB APPS (PWA).....	13
2.4.1	<i>Key Features</i>	14
2.4.2	<i>Development Frameworks and Languages</i>	19
2.4.3	<i>PWA Usage and Relevant Research</i>	21
2.5	GOVERNMENT AS A PLATFORM (GAAP).....	24
2.5.1	<i>EGovernment</i>	24
2.5.2	<i>M-Government</i>	25
2.5.3	<i>User Satisfaction and Perception in mobile adoption</i>	26
2.5.4	<i>Technologies within Public services</i>	28
2.5.5	<i>Digital Government in UK</i>	31
2.5.6	<i>Conclusion</i>	32
3	RESEARCH METHODOLOGY	34
3.1	INTRODUCTION.....	34
3.2	POSTPOSITIVISM AND QUALITATIVE INQUIRY	35
3.3	APPROACH	35
3.4	STRATEGIES	36
3.5	DATA COLLECTION	36
3.6	CONCLUSION	38
4	DEVELOPMENT METHODOLOGY.....	39
4.1	INTRODUCTION.....	39
4.2	REUSE-ORIENTED SOFTWARE DEVELOPMENT	40
4.3	FEATURE DRIVEN DEVELOPMENT	40
4.3.1	<i>Requirements Engineering and Modelling</i>	41
4.3.2	<i>Build – Technical Stack</i>	48
4.3.3	<i>Technical architecture of a PWA</i>	53
4.3.4	<i>Test by feature</i>	58
4.4	CONCLUSION	59
5	RESULTS AND DISCUSSION	61
5.1	FUNCTIONAL TESTING.....	61
5.2	PWA CHECKLIST	61
5.2.1	<i>Performance</i>	62

5.3	CROSS-PLATFORM CAPABILITY	64
5.4	USABILITY TESTING	65
5.4.1	<i>PWA Installation Experience</i>	65
5.5	APP-LIKE EXPERIENCE	68
5.5.2	<i>System Usability Survey (SUS)</i>	76
5.6	AFFINITY TO GOVERNMENT DIGITAL SERVICES	78
5.6.1	<i>Post-Usability Test</i>	78
5.6.2	<i>DVLA User Insights Survey</i>	79
5.6.3	<i>Usage Data on existing DVLA and HMRC mobile apps</i>	79
5.7	CONCLUSION	81
6	CONCLUSION.....	82
6.1	RESEARCH BASED FINDINGS	82
6.2	SOFTWARE DEVELOPMENT CASE STUDY BASED FINDINGS.....	82
6.3	LIMITATIONS.....	83
6.4	RECOMMENDED FUTURE WORK	84
6.5	SUMMARY	84
6.6	REFLECTION	85
APPENDIX I		87
	VEHICLE RESPONSE DATA SET - 3 AUTHENTICATION LEVELS	87
	GOOGLE CHECKLIST FOR A BASE LINE PWA.....	88
	GOOGLE LIGHTHOUSE AUDIT WITH PWA CRITERIA PASSED	89
APPENDIX II		90
	USER STORIES – KANBAN BOARD	90
APPENDIX III		94
	DVLA USER RESEARCH COLLATED COMMENTS FOR A DVLA APP (SOURCE: DVLA).....	94
	USER RESEARCH – REQUIREMENTS ELICITATION	95
	PRE-INSTALLATION SURVEY.....	97
	POST USABILITY TEST SURVEY.....	98
	SYSTEM USABILITY SURVEY	100
	REFERENCES.....	101
	BIBLIOGRAPHY.....	113

Table of Figures

FIGURE 1 WEB ON THE MOBILE [4]	7
FIGURE 2: DOWNLOADS IN LAST SIX MONTHS- REACTJS	19
FIGURE 3: FACTORS INFLUENCING MGOVERNMENT	26
FIGURE 4: MOBILE APP VS MOBILE WEB SOURCE: COMSCORE MMX MULTI-PLATFORM, JUNE 2019.....	28
FIGURE 5: PERCENTAGE OF COUNTRIES WORLDWIDE	29
FIGURE 6: GOV.UK WEB TRAFFIC [8]	31
FIGURE 7: RESEARCH MODELS ADOPTED [78].....	34
FIGURE 8: FEATURE DRIVEN DEVELOPMENT[87]	41
FIGURE 9: DOMAIN MODEL	41
FIGURE 10 USER JOURNEY - UI FLOW	43
FIGURE 11: OFFLINE USER JOURNEY	44
FIGURE 12: LAYERED ARCHITECTURE	46
FIGURE 13: COMPONENT DIAGRAM	46
FIGURE 14: ENTITY RELATIONSHIP DIAGRAM - VEHICLESDB	48
FIGURE 15: GAZERECORDING USER1	51
FIGURE 16: GAZERECORDER USER 2	51
FIGURE 17: TAX RATES PROTOTYPE	52
FIGURE 18: PWA ON WINDOWS10.....	54
FIGURE 19: CACHE-FIRST STRATEGY	55
FIGURE 20: SUBSCRIPTION BASED NOTIFICATION WEBPUSH	58
FIGURE 21:PUSH NOTIFICATION THROUGH FIREBASE.....	58
FIGURE 22: LIGHTHOUSE AUDIT.....	62
FIGURE 23: THE APP SHELL ON ANDROID	62
FIGURE 24: LIGHTHOUSE PERFORMANCE SCORES FOR THE PWA.....	63
FIGURE 25: WEBPAGEPERFORMANCE.ORG AUDIT COMPARISON(INITIAL READINGS).....	63
FIGURE 26: A2HS AWARENESS	66
FIGURE 27: APP INSTALLATION BEHAVIOUR	66
FIGURE 28: CHROME ON EDGE ON DESKTOP	67
FIGURE 29:A2HS ANDROID AND IOS.....	67
FIGURE 30: EASE OF INSTALLATION.....	67
FIGURE 31: NUMBER OF PEOPLE WHO PREFER ADD TO HOMESCREEN INSTALLATION	67
FIGURE 33:EXTERNAL PAGE IN-APP BROWSER MACOS.....	71
FIGURE 33: PUSH NOTIFICATION WHEN APP IS CLOSED WINDOWS10.....	71
FIGURE 37: SCREEN 1.....	72
FIGURE 37: SCREEN 2.....	72
FIGURE 37: SCREEN 3.....	72
FIGURE 37: SCREEN 4.....	72
FIGURE 38: OFFLINE PAGE ALL PLATFORMS.....	73
FIGURE 39:OFFLINE EXPERIENCE SATISFACTION	73
FIGURE 40:PUSH NOTIFICATION SATISFACTION	75
FIGURE 41: PERCEPTION OF TIME TAKEN	75
FIGURE 42: POSITIVE QUESTIONS ON USABILITY	77
FIGURE 43: NEGATIVE QUESTIONS ON COMPLEXITY	78
FIGURE 44: PREFERENCES FOR GOVERNMENT SERVICES	79
FIGURE 45: USER RESEARCH – DVLA DATA ON REQUEST FOR MOBILE APP	80

List of Tables

TABLE 1 EGov/MGov ADOPTION	30
TABLE 2: SOURCES OF DATA	37
TABLE 3: FEATURES LIST (MoSCoW).....	45
TABLE 4: USER TESTING GROUPS	68
TABLE 5: USER TESTING - SUCCESS RATES AND ISSUES	69
TABLE 6: TIME TAKEN TO COMPLETE ALL TASKS.....	70
TABLE 7: SUS RANKING AFTER USER TESTING	76

Abstract

From the first commercial device, Apple Newton message pad in 1993 to wearable devices, mobile devices have gone through a huge developmental transition over the years. Mobile technologies, such as tablets and smartphones provide portability and are powerful enough to compete with personal computers in performance [1][2]. With more than 5 billion of the global population owning a smart phone, the web is accessed on mobile phones more than ever as illustrated in figure.1 [3][4]. Governments have recognised the impact of this culture and multiplier effect of technology that creates a new breed of services and devised innovative transformation strategies to deliver their services. However, accessibility, cross-platform limitations, development and maintenance costs are the deterrents for mobile app development. In light of this, it was proposed to investigate if Progressive web apps (PWA) could be the alternative technology for governments to tackle the mobile gap. A literature review on cross-platform development approaches, features of PWA and existing research on PWA and mobile initiatives in governments was conducted. A progressive web app prototype was built for the vehicle enquiries services of the Driver and Licencing Agency in UK as a proof of concept. User research and usability study results of the PWA across several platforms was conducted with user testing and surveys. The results revealed a positive experience with 100% preference on PWA installations and a SUS score of 85.22. One in 12 persons also preferred the government to deliver their services through mobile apps. With major operating systems and browsers now providing enhanced support for PWA and the ease of use and nearly native app experience they provide along with push notifications and offline capability, it was recommended that governments explore PWAs as a cross-platform development approach.

1 Introduction

The advances in communications technology and infrastructure with high speed internet access, affordable mobile devices in the thriving smart phones market has facilitated an unprecedented use of mobile devices across the world. ‘Mobile first’ has become the policy that provides exciting opportunities to serve larger population ubiquitously. This is not exclusive to the commercial sector, but Governments are also under pressure, with people expecting swift and effective services. One would say, it is time that the government sees a shift from its ‘digital by default’ declaration to embrace mobile by design. This would mean creating services around mobile device hardware and software capabilities rather than repackaging traditional sites for mobile use. Native and hybrid mobile apps being the most popular app categories, progressive web apps have seen a rising trend in the industry. The UK government digital services have recently revisited their policy on mobile apps and are open to its adoption on exceptional cases albeit reluctantly [5]. This project is an attempt to address such scepticism and provide recommendations on the way forward by way of determining if government should invest on apps and if progressive web apps can alleviate the issues surrounding mobile app delivery.

1.1 Digital Government

The way the government conducts its business with its huge machinery of departments and bureaucracies is a barrier to realising the transformative potential of digital technology [6]. In UK, since the Government digital strategy in 2012, significant progress has been made in the online services provided by the government. A data-informed analysis report by the Institute

for Government [7][8] states that there are 800 government services offered online in January 2019.

1.2 Mobile Apps Industry

With a market value of \$439 billion in 2018, the mobile applications market is overcrowded and fragmented in a diverse range of markets [9]. Conglomerates such as Apple, Alphabet (Google) and Facebook are leading the global market forecasted to reach a value of \$661.4 billion by 2023 [9]. Over 175 billion apps were downloaded in 2017 [10]. It is a highly competitive market with a large number of players including Apple and Google Play. With windows out of the mobile operating system market and Blackberry being a negligible proportion of the market, for developers, the choice is mainly between iOS or Android platforms. Developing for all platforms has led to a new breed of cross-platform development. Progressive Web Apps (PWA) came into the arena in 2015 as a cost effective and resourceful approach. For consumers, the propensity to download apps could be influenced by factors such as price, size, updates and available languages in addition to the value.

A case study of the vehicle enquiries services of the Driver and Vehicle Licencing Agency (DVLA) was chosen for development of a PWA. An overwhelming 33 million people taxed their vehicle online in 2015 – 2016 with 93% of the transactions done online [6]. This research was an effort to translate the success of mobile apps in commercial industry to government sectors in a cross-platform model using PWA.

1.3 Aims & Objectives

Schmitz et al [11] conducted a study on mobile app usage and outcomes achieved by public transport apps in Germany to conclude that value, convenience and speed of transactions were beneficial to the organisation. Yberg [12] developed a proof of concept PWA as an alternative native apps emphasising the advantages of avoiding distribution platforms with the limitation of having to use own servers. Several articles describe the merits of the device agnostic PWA with customised html based interactive content and speed that compares to that of in-app content [13][14]. The first outcome in the UK Government's digital transformation strategy 2018 [15] [6], is to focus on innovative ways to provide the best of digital services to citizens.

This raised the main question for this project

‘Would a Progressive web app be the way forward for Government services to tap into the potential of mobile apps as a cross-platform approach, for wider reach and ease of development.

The main objectives to aid in the study followed three streams.

- Critically analyse the development techniques employed in providing services through the web and mobile devices.
- Analyse the efficacies of PWAs over other technologies as a cross-platform development.
- Evaluate the current status in the use of mobile apps in e-government
- Implement a proof of concept progressive web app for a public service and evaluate its effectiveness

The outcome of the literature review, case study and development are expected to produce a positive evaluation on the use of PWA on usability and performance. However, there may exist some limitations that will also be discussed.

1.4 Document structure

The remaining chapters of this paper are organised as follows:

- **Chapter 2: Literature Review**

A literature review on related research and mobile technologies are discussed. Key features and criteria for a PWA are presented with examples from industry. A discussion on egovernment/mgovernment initiatives across the world and the current developments in the UK government is used to form the basis of a use case.

- **Chapter 3: Research Methodology**

A brief outline of the research methodologies employed are discussed with the philosophy, approaches and strategies that have been explored and employed.

- **Chapter 4: Development methodology**

This chapter explains, the design and developmental approaches, technical tools used in the development of the proof of concept PWA.

- **Chapter 5: Results and Discussions**

This chapter explains the evaluation of the product and an analysis of the data collected from surveys and user testing. Success and failures are detailed.

Chapter 6: Conclusion and Recommendations

A summary of the findings from literature review and case study with recommendations for future work are outlined.

2 Literature Review

The topic of interest being the dissemination of a digital service through the web, it would be interesting to see how the web and development technologies for mobile have evolved. Cross-platform technologies, PWA features, current government initiatives form the basis of this review.

2.1 History of the Web

The history of the web could be traced back to 1962 when Licklider [16] introduced the concept of the ‘Galactic network’, a future of networked computers for end user computing. However, it could be said that 1989 marked the beginning of a new era. Communication between previously isolated networks became a reality with TCP/IP standards and the World Wide Web (WWW), the browser and the Universal Resource Locator (URL) were first coined and prototyped by Berners-Lee [17]. It was not until 1991 that the browser became the catalyst that changed the web by establishing itself as one of the most ubiquitous and powerful software tools leading to the proliferation of web 2.0 technologies [18] such as microblog sites, networked social media, wikis, and web services with organisations embracing the power of collective intelligence[19][18][20].

Browser programming languages and technologies such as Ajax and high-speed internet saw an increase in browser applications paving way to the rise of web Applications(apps). Application Programming Interface(APIs) expanded the realm of data access by external applications to query several data stores seamlessly[20]. The shift to deploying applications over the web deprecated the cumbersome process of installing them in devices. However, Web 2.0 comes with its own downfalls relating to security concerns[20].

The prevailing technological eco system led to a natural progression to the third-generation web, called the web 3.0, sometimes referred to as the Semantic web, construed as intelligent and live, emulating human behaviour to think and generate customised micro content[21] [22]. They play a prominent role in logic-based applications that rely on multiple application sources for information[23]. Simultaneously, hardware developments in the mobile industry led to a newer revolution in the way web was accessed.

2.2 The Web on the Go - Mobile Technologies

Murugesan et.al [24] forecast the future of the web in 2011 as a ubiquitous and smarter landscape penetrating even the underdeveloped economies. It would be prudent to agree with them as we have witnessed the evolution of the web as a resource of multimodal access and technologies. Demand for quick information and social space, democratisation of the inputs, and shrinking device sizes led to the rise of mobile applications. The omnipresent battle between web applications and mobile applications has seen a number of technologies invested in the development of applications. What is of interest here, is the rationale behind mobile apps development.

- Current Trends in consumer preferences

StatCounter Global Stats data in 2016[25] revealed that mobile devices overtook desktops for webpage access as in figure1. It can be said that recent years have seen a sharp rise owing to wearables and affordable connectivity.

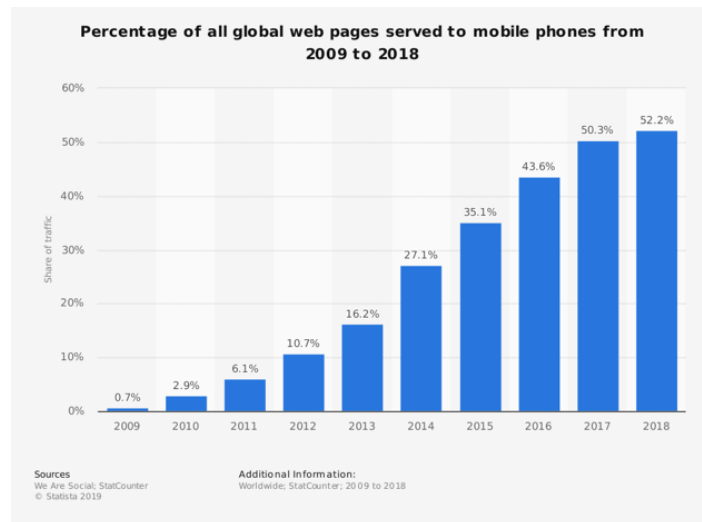


Figure 1 Web on the mobile [4]

The UK digital market overview from March 2019 [26] summarises the current trend of mobile devices as the preferred option with 78% of all adult online minutes. 74% of 18 – 24 age group and 53% of over 55s are on smart phones. 89% of the adult population use smart phones with 35% on mobile only. Mobile apps generate 65% of all online minutes. It is pertinent to discuss the mobile technologies that enable the distribution and consumption of web content.

Design and development considerations are dictated by several layers including operators, platforms, application framework and marketing. This has led to four main strategies to serve content on mobile devices and discussed in following sections:

- Native app
- Mobile website
- Responsive web site
- Mobile web applications

What makes designing for the mobile different is the context[27]. Designing for context with relevance to location, intent, intuitiveness and lively experiences, providing value to the user brings with it its own complexity. With the exception of the native app, all other three options utilise the mobile web browser with their own merits and demerits.

2.2.1 Native App

Native app often referred to as Mobile app, are made for a specific device and platform such as iOS, Android or Windows to harvest the functionalities and feature of the specific devices they operate on. Content is not rendered through a browser and the app resides in the device. The merits of native apps lay in access to hardware, e.g., camera and GPS with faster load times, offline content and background synchronisation. Buettner [28] states that in addition, the distribution through the Appstore marketplace ensures discoverability and common monetising mechanism. Access to native UI elements, contact lists and gestures are possible as it sits on top of the platform.

These very characteristics have led to the popularity of native apps especially in entertainment and games. There are around 5.08 million native apps available in the three major app stores – Apple App Store, Google Play Store and Windows Store[29].

However, the affiliation with platforms calls for separate development and maintenance of code for each of the platform available. This does however mean, despite the rich user experience and low development cost for a single device, native app development requires the developer to consider the device type, platform and UI design for a better UX. One could also argue that this has transported us back to the age of installing applications that are dictated by device and operating system vendors with developers having no control. Portability leads to the question if cross-platform solutions are any better.

2.3 Cross Platform Development Approaches (CPD)

Cross-platform development approaches involve a single developmental cycle bringing considerable saving of time and efforts by enabling writing the mobile application in a unified

codebase, while it is transcribed to run on several platforms thereby differing from native apps. Some of the available cross-platform approaches employed, not necessarily apps are discussed below.

2.3.1 Mobile Website

A mobile site is simply a separate, dedicated website interface developed for mobile devices and is typically accessed via a different subdomain such as “mobile.domain” or “m.domain” . Device detection redirects to the mobile specific URL [25]. There might be disparity in the content served through a desktop site as a mobile site is often a condensed version with a canonical desktop version. The backend, which provides core application functionality such as access to any data sources is developed as a single code base serving both the mobile and desktop versions. This typically means that only the user interface needs to be developed twice to support mobile devices but raises device optimisation, consistency and maintenance issues.

2.3.2 Responsive Design

Responsive websites go a step further to eliminate the additional user interface and separate URL used by mobile websites. A responsive web design (RWD) resizes a single user interface layout to respond to the size of the user’s device to render the content seamlessly from a single URL by dividing the screen into a flexible grid using percentages as the scale rather than fixed-width pixels [30]. The core element in RWD are the cascading stylesheets (CSS) media queries. The stylesheet is manipulated for varying screen sizes, dynamically rendering the page by recognising the screen size. Elements on the screen on mobile devices may be rendered differently to cater to the restriction in screen size by way of different CSS rules. The best experience is provided by a mobile first approach that involves design considerations to prioritise key functions and then modifying to expand for larger screens. Often misinterpreted as the process of designing the user interface to fit to screen, it is rather guided by the content,

context and user behaviour[25]. For instance, features like absence or presence of touch gestures, mouse over etc need to be considered and alternate solutions provided for accessibility in all device types. Single code base, accessibility and resilience to changes have made RWD revolutionise web development. However, slow performances is a trait to consider.

2.3.3 Hybrid Development

Hybrid apps are developed with cross-platform development principles. However, they still contain some element of platform dependency. Berners-Lee[17] in the Guardian claims that most apps are essentially built on a web platform and morphed into apps using device specific frameworks[17]. True to this, hybrid apps are built on most of the code written in HTML5 and JavaScript and then wrapped with native code to become device agnostic. Tools such as Adobe's PhoneGap and Apache Cordova and other user interface libraries such as Ionic allow for the app to be built with web technologies and use webview component to present inside a Native app. Hardware access to native functionalities is also enabled by the framework by providing a bridge for bi-directional transcription[31][32].

2.3.4 Interpreted Approach

Base code is written in JavaScript and is interpreted at runtime. As opposed to a hybrid approach where native components are mimicked, interpreted approach does not depend on web views but mapping of native UI widgets and has better UI performance[33]. React Native allows for third party components as well reducing development efforts considerably.

2.3.5 Cross Compiled Approach

Cross platform tools like Xamarin compile C# code into native code[34]. Google's flutter compiles Dart code to ARM code and is gaining a lot of momentum since its release in 2017. It compares to performance speed of native apps with its own rendering engine[35].

2.3.6 Accelerated Mobile Page

Introduced in 2015, Accelerated mobile pages is a HTML framework with JavaScript which make up the web components. Despite the pitfall of JavaScript and images slowing down page rendering considerably, they are increasingly adopted in web development. As a solution to optimise web pages, AMP provides prebuilt components to increase performance and dynamic experiences [36]. Pages are loaded with Google cache and loads within less than 2 seconds on mob and lacking UX features.

2.3.7 Web Apps

Web apps are normal websites but look and feel like an app on a mobile device. Initial web apps were rigid as such they lacked the ability to use the hardware functionalities of the device they operate on. The difference from hybrid lies in that they run inside a browser which makes them compatible across all supported devices. Trott [37] argues that the development in technology associated with the ever increasing speed of the internet enabled the advent of the web app and reliability enabled seamless download of applications to run on browsers.

The availability of several frameworks that produce the necessary code to load the packaged web content results in a simplified development process to port the app to different devices. One can argue that this is similar to the PhoneGap apps which is a hybrid development. This does in essence, signify the overarching approach of cross platform development. Web apps leverage the potential of JavaScript, HTML and CSS. JavaScript has been favoured for web

apps due to its dynamically typed nature with no requirement for prior compilation and browser optimisation. However, they still need to be parsed and compiled before execution as opposed to native apps that have compiled code. PWAs aim to close the gap between web apps and native apps.

2.3.8 WebAssembly

Web assembly is the newest entry into the web app scene in 2017, where code is in binary format. Code written in any programming language, such as C# is compiled into WebAssembly bytecode, stored in binary format for browsers to execute [38]. This is particularly beneficial in high performance reliant applications such as 3D rendering, video editing, encryption etc., where JavaScript could be seen as slower. This raises the question if WebAssembly would be an alternative to JavaScript and is one to be answered as more use case experiences emerge and will not be a point of discussion to be explored in detail here.

2.3.9 Summary

Though there are different approaches to cross platform development suggested by several researches, El-Kassas et.al. [33] identify 6 categories, whilst maintaining that several of the cross-platform solutions are still under R & D (MD2/XMLVM). They maintain that approaches such as Cross-Compilation, Virtual Machine, Model-Driven Development, and Web-Based ones have specific advantages that outweigh the efforts in their application. They further subcategorize these approaches to those that are Component-Based or Cloud-Based or even those that involve a merged approach. Some of the limitations expressed include issues such as the manual definition of user interface for each supported platform and consideration of common functions rather than platform-specific functions.

Rieger et.al [31] states that the cross-platform app development faces challenges as the complexity introduced by novel mobile devices involves much work in relation to assessment criteria infrastructure, development, app, and usage.

Is there a best of both worlds? Biørn-Hansen et.al[39][40] pose a question as to whether PWA could be the definite approach to cross-platform development and conclude that the PWA fulfil several of the requirements for a unified multi-platform development. However, they assert that it is difficult to say if the app would replace existing cross- platform development approaches and recommend further research which could be attributed to the status quo of PWA at that time, when support by major browsers were still underdeveloped.

2.4 Progressive Web Apps (PWA)

The earlier discussion on Web apps emphasises the versatility provided by web technologies with regards to cross platform development. The following sections will present the PWA as an evolution of the web app to provide additional functionalities on mobile devices as a potential cross-platform deployment approach.

PWA combine the features of all of the above. Russell and Berriman [41] introduced the term Progressive Web Apps in 2015. It is in essence, a web app but with progressive features with the core differentiation being an app-like experience in mobile devices.

A PWA is accessed via a URL and rendered through an app-like interface using web technologies, with increased features and possibilities that sets them apart from web apps[42]. The cross-platform development allows for a PWA to be accessed in any device but is dependent on browser support to render content. All major browsers including Google, Safari, Edge and Samsung internet support PWAs. Service workers and in-browser logic facilitate offline content and background synchronisation that are inherent in native apps.

Mobile application functions are traditionally classified as time-critical, location-sensitive and personal control functions[43]. Thus accessibility, performance and engagement are the key drivers for the rationale to decide upon the appropriate technology for developing a service. We discuss some of the core features of the PWA to understand why it could be attractive to both developers and users akin.

2.4.1 Key Features

The main characteristics of a PWA are that they are reliable, fast and engaging [41]. Ater [44] defines PWA as

“Progressive web apps start off as simple websites, but as the user engages with them, they progressively acquire new powers”

Archer[45][25] defines the core strategy of progressive enhancement is to design for the lowest device specifications and capabilities first and then add additional features to tackle complex capabilities. This is indeed the exact reverse of graceful degradation which was the common approach in the desktop era. This enhanced power transforms a website into an app, more of a disguise of a native app. These are defined by the ability to load faster even in slower networks such as 2G with an app like immersive user experience. This means responsive navigation without browser based navigational controls such as back/forward buttons, address and search bars, fluid transition, offline experience, faster loading and push notifications. These native-app-like experiences in a PWA are explored in the following sections.

2.4.1.1 Lightweight installation

One may recall, that a PWA was introduced as an app that does not require an installation from any of the app stores. With a PWA, only necessary basic code is installed in the device from a URL. The user is given the opportunity to add a shortcut to the home screen that is more than a mere bookmark as exhibited by a web app. However, since January 2019, Google Play app

store features PWAs, as Chrome 72 for Android shipped with Trusted Web Activity (TWA) which TWA allows Chrome tabs to open in a standalone mode. All static files that are the basic components to build the site such as JavaScript, html and css are installed in the device. However, they do not take up a large proportion of the device storage, akin to native apps. For instance, the Book My Show's PWA is 54 times smaller than their Android app and 180 times smaller than the iOS[41].

Ater[44] presents the case for the increase in reaching audience through a mobile site than a native app. Several research indicate that the conversion rate in installing an app tends to be low due to the multi-staged process of installing a native app [42][48][50]. It can be said that the process of discovering the app through the app stores and installing it is convoluted but might be productive for the most popular applications such as Facebook, Twitter, Spotify, Netflix etc. But Kho[49] argues that if these apps are too large, users would tend to just use the web instead. Their popularity might make them easily discoverable as the user knows what they are looking for. However, studies show that websites of other categories lose 20% of their users somewhere within the process. In addition, studies also show that installed apps are deleted at a similar rate with users prioritising device storage and tidier home screens[44]. In contrast, a PWA does not require to be staged through any of the closed or open app stores pertaining to any specific platform. Ater[44] describes the PWA as a one-step funnel.

2.4.1.2 Fast Loading

PWA typically load pages under 5s on 4G set by Google's criteria. PWAs are single page apps, downloading everything on first load. Hence subsequent loads are much quicker. A report on the state of the retail industry concludes that of the 19 PWA-using retailers studied, the 4.5s long median first page load reduced to 1.3 seconds for subsequent loads [50]. Network speeds

are no longer an issue as evidences show that pages with multimedia content load around 3 seconds even in 2G networks[41].;

Baraković and LeaSkorin-Kapov[51] have concluded that longer loading time of a web page in a smart phone has a negative impact on the usability of the website more than that on desktops. This echoes a similar finding by Varela et al.[51] who have demonstrated that loading time has no impact on perceived usability on desktops. Overall, 86% of participants were unhappy with websites loading longer than 5 seconds. 1.5 s or less loading time was categorised as high quality. The two fundamental features that facilitate faster loading in PWAs are offline caching and service worker.

2.4.1.3 Offline caching

One of the features that gains user trust in an application is reliability, the ability to access in poor network or even no network conditions. Offline caching involves processing of a client request at the client side and content delivered if already in cache. The request is only forwarded to the server if undeliverable from cache, thus increasing latency. This implies that the non-availability of a network to interact with the server is not an obstruction to service, but also a catalyst for speedier page loads in the presence of network. Service workers enable the PWA to render content and interact offline utilising the cache interface. Interactions such as posting messages will be realised once users go online. The fundamentals for the offline-first approach is to provide for these features regardless of connectivity strength.

Caching strategies and levels in a PWA can vary depending on the requirements from static content to dynamic content with full offline functionality. Braun and Fertig[52] have researched on the implementations of offline-first approach to RESTful systems and

recommended six levels of offline support. The lowest being caching for reading operations and queuing offline requests at second level.

Offline caching may not be a significant factor for desktop web users with perpetual network access. However, for mobile only users, offline content would increase user experience manifold. Traditional web apps and RWD despite client-side rendering and data manipulation have not circumvented the dependency on the server which exactly is the service worker's purpose.

2.4.1.4 Service Worker

Service workers are the core of a PWA that fulfils the requirement of offline access and push notifications in an event driven execution model. Service Worker performs four fundamental functions that emulate persistent connectivity.

- Caching assets
- Caching pages
- Push Notification
- Syncing data when transitioning from offline to online, to upload user input.

All interactions between the client and the server result in a sequence of events triggered by fetch calls from the client or server-initiated requests. Once registered at the browser, a service worker sits in the background as a proxy in a separate thread and works in parallel, processing the events associated with these transactions such as FETCH and PUSH [53]. The service worker intercepts network requests and scans the cache for a previous instance of the same request and responds with the cached response. In the absence of the duplicate request, the network request is processed[54]. What this means is reduced calls to the server, hence faster page loads.

2.4.1.5 Transport Layer Security

Since the service worker sits as an intermediary between the client and the server, it poses security issues and vulnerability on push notifications[55]. Hence, it needs to be served securely on https. This easily addresses the major security concerns relating to native apps according to Lynch, CEO [49][56] of the Ionic framework.

2.4.1.6 Web-App Manifest and a place on the HomeScreen

The home screen or desktop being the gateway to all apps, a PWA would never be complete without an icon on the home screen for easy launch. The Web-App Manifest defines the identity of the app with an icon, title, description and appearance. This generates the App shell and caches it when the user visits the site the first time. This uses a combination of server-side rendering and client-side rendering. When a site is identified as a PWA, chrome fires an event, to display the install feature and the icon is added to the home screen. Successive visits through this icon opens the standalone app with content rendered within the app shell.

2.4.1.7 Push Notifications

An app differs from a website in that it is capable of initiating re-engagement. Push notifications remind users of the existence of the app and urge them to revisit. From a business' perspective, push notifications fulfil their obligations towards their customers in keeping them up-to-date with information pertaining to their nature of business with them. Alerts on overdraft balances, utility bills and renewal reminders are valuable to businesses and consumers. A PWA's app like experience is enhanced by the ability to receive push messages from a server and display push notifications making it more immersive and is discussed in chapter 5.

2.4.2 Development Frameworks and Languages

Atwood's Law states that any application that *can* be written in JavaScript, *will* eventually be written in JavaScript. True to this, the following JS frameworks are the most popular amongst frontend developers.

Angular

Angular from Google uses the JavaScript ecosystem in an MVC framework model. With extensive support from Google, there is a clearly defined documentation for easily developing robust and reliable apps. It is however, a little complex as it involves typescript which has a steep learning curve and hence relies on developer's expertise.

ReactJS

ReactJS is an open-source JavaScript library developed by Facebook. It is extremely popular now as evident from figure3 with an appropriate 5 times more downloads in the last six months.

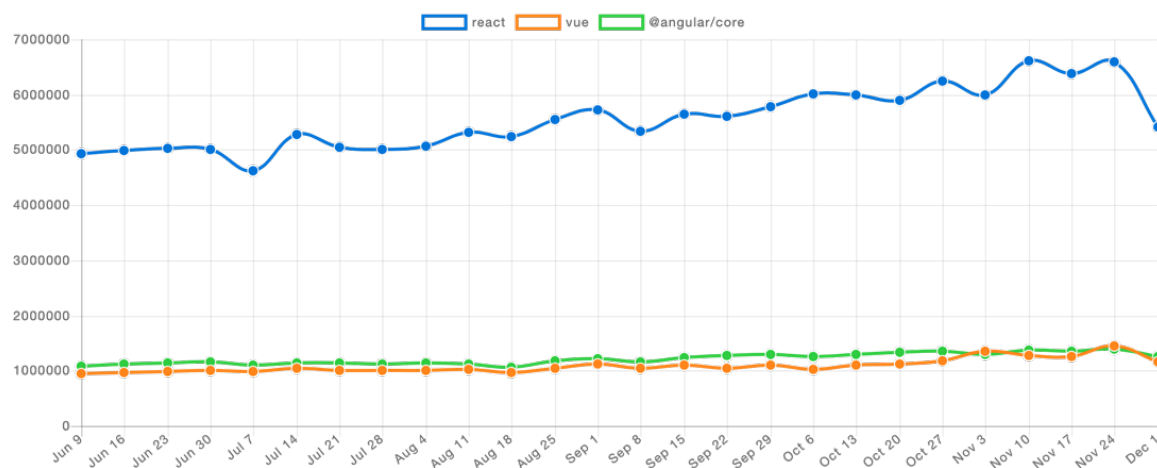


Figure 2: Downloads in last six months- ReactJS

source: <https://www.npmtrends.com/@angular/core-vs-react-vs-vue>

Some advantages include

- Highly cohesive and loosely coupled components rendering highly dynamic pages by employing JavaScript extensions known as JSX.
- Robust maintenance by Facebook, it is tried and tested, flexible and scalable.
- Very fast with its virtual DOM, that re-renders only updated elements and one-way data binding that reflects state changes in the model on the UI resulting in robust performance, less complexity and errors [57][58].

Vue

Vue also is a typescript and JSX framework borrowing many features from React. Easier to learn, simple code and flexible scaling are its merits. However, it lacks the support that the React community enjoy and migration from legacy code is complex.

Ionic

The ionic framework is supported by Angular and popular amongst hybrid app developers. It uses the Apache Cordova framework enabling cross platform capability. With its predefined components and service worker, and all PWA features enabled by default, it is easy to set up, but updates are cumbersome as the existing app would have to be refactored.

Polymer

Polymer is a template from Google, using plain HTML, JS and CSS, simple components and web view. It is well documented with extensive support and development tools embedded in browser alleviate the need for debugging tools. But, slow loading due to increased http requests and lack of server-side rendering make it not too attractive.

The decision of the framework depends on the individual business case and resources available. AngularJS is a full-fledged MVC framework whilst ReactJS is a JavaScript Library just incorporating the view. This means that the restrictions placed by Angular is non-existent in ReactJS with more freedom for developers [57][58].

However, since PWA involves client-side code execution, code can be in any language or framework such as PHP, Rails or C# as long as they can handle html, CSS and JS. An existing site can be transformed to a PWA by way of progressive enhancement.

2.4.3 PWA Usage and Relevant Research

A number of organisations including Twitter, Ola, Flipkart, Ali Express, Wiki offline are leading the PWA market with their success stories. Their primary goals are reach and conversion rates.

Case studies on the efficacies of PWA

Nugroho et.al. [47] developed a monitoring information system UI as a PWA. The PWA provided the means to access the API of a palm oil company's monitoring system with a device agnostic UI. Web sockets for real time data loading, home screen icon and push notifications were implemented to provide a seamless native like experience[47]. Overall, the PWA provided a valuable resource for operators on the field successfully.

Japanese publishing company Nikkei's motivation for a PWA were performance issues to their existing website. They developed a multi-page PWA using JavaScript. Service workers with caching strategies enabled faster loading of pages. Critical rendering path strategy, cache-first approach and prefetching successive pages ensured pages loaded 75% faster than the non-PWA site. Bundling all JavaScript and loading asynchronously resulted in optimising third-party

JavaScript. They saw drastic improvement in performances and business gains with the fully qualified PWA. Their data shows 85% accessibility and 53% conversion[59].

Comparison of Native vs PWA – User experience and performance metrics

Cardieri and Zaina [60] conducted an experiment comparing three different web technologies RWD, native apps and PWA. They performed two comparisons- web vs Native and PWA vs Native. They compared Tripadvisor and Trivago respectively. They concluded a negative experience on the native app over PWA on response to interactions or loading[60]. There does arise the debate that are they even comparable? However, considering that the entire app resides in the device as opposed to PWA where only critical assets are downloaded, one would expect the native app to be faster and the results proving otherwise adds credibility to PWA.

Gambhir [61] analysed the effect caching with service worker has on the performance of a PWA by using Google Lighthouse to measure performance metrics on iOS and Android apps in comparison with PWA. They conclude that caching improves performance and the number of users has an adverse effect on response times in Android apps whilst impacting positively on PWAs. One could argue that it is however difficult to ascertain the validity of their research in current conditions as browser support from Edge and Safari was not present then. It could also be contested that the apps don't have similar architecture, thus undermining the comparison[61].

Backlund and Hedén [62] developed an app implementing interactive map with Mapbox GL framework to monitor software quality ISO 25010 standards on two cross platform approaches. The app was created using React Native and Redux as well as a PWA with React and Redux. Their findings are rather inconclusive with no clear winner. PWA ranked better in portability, usability, reliability and maintainability whilst React Native scored well in performance efficiency and functionality of hardware components. This is rather expected as JavaScript

does not optimise hardware functionalities that one could hope for. It is also interesting to note that the PWA scored high on development context whilst React Native fared well in end user experience factors.

Majchrzak et.al [31] explored the potential of PWA in the cross platform development market with a view to unified development. It is understandable that they speak of fragmentation and browser capabilities posing as barriers for unified development. They pose questions on PWA compatibility with wearables and Internet of Things. However, since then most browsers have been shipping with increased support for PWA making it a realistic approach for device agnostic development.

Security implications

Lee [55] has identified security and privacy flaws associated with browsers and third party push services. Attacks that target the vulnerability of service workers, hijacking and spoofing of push notifications could be perceived as limitations. A systematic study of the top 100 Alexa sites were conducted to identify issues. They have proved their case by demonstrating a cryptocurrency attack and recommend solutions to mitigate risks. It is for this very reason that a PWA requires to be served via https and all push messages be encrypted with JSON web tokens.

Impact on energy efficiency

Service workers being the core of PWAs, there is much scrutiny on its impact on efficiency. Malavolta et.al, [54] investigated the impact of service workers on device energy efficiency. They tested 7 PWAs with three variables, device, network and service worker status. Their results showed no difference on the device and network conditions, but difference in energy efficiency. This is due to the difference in the sizes, business logic and service worker implementation.

There appears to be not much research on the developmental and usability aspects of the PWA for a large scale not for profit service, especially within public services. It would be interesting to see if the PWA that has gained momentum in the commercial sector in recent years has intrigued the public sector. It would be pertinent to briefly look into the existing developments within Government regarding the provision of digital services, especially in the mobile market. The following sections explore some Government initiatives in this regard.

2.5 Government as A Platform (GAAP)

Government as a platform is a move from a Government being a service provider to work within the digital ecosystem providing a citizen-centric single point of resource, driven by innovative technology for economic growth. A research report by Accenture in 2018, ranks Singapore and UK in the first and second place in GAAP readiness[63].

2.5.1 EGovernment

Studies show that public administrations in Europe have been formalising e-government strategies since 2007 on digital transformations of governance and services[64]. Since the initial inception, most governments developed most if not all of their services as web applications offering services ranging from engagement and participation to core services such as payment of taxes, passport applications, etc. Almost all governments also have a social media presence such as Facebook and Twitter. SOCITM conduct reviews on council websites by testing them on four criteria including site performances on mobile devices. They have concluded that almost half of all interactions to council websites are from mobile devices.

.

2.5.2 M-Government

M-Government is an extension to the e-Government with the aim to attain reach, availability to users 24/7 and involves interaction of the citizens with the government. The M-Government paradigm falls into web-based web portals, apps and non-web based short message services and interactive voice response platforms[65]. M-Government transcends into web applications delivering more personalised services by leveraging the advances in mobile technology infrastructure [66]. The potential of m-government for effective service delivery through easy access and usability even in remote locations is enormous due to the ubiquitous use of mobile phones[67]. The purview of m-government relies on the relevance, business value derived from effective administration and usage, contributing to several paradigms. Hussain et al [65], Ahmed Imran reveal m-government adoption as an expansion pattern of e-gov in developed countries even as early as 2014. Local need and innovation were catalysts in developing countries especially the exponential reach of m-gov in heavily populated countries[65].

However, the Social and economic inequalities in addition to the demographics will affect the key performance indicators and evaluation of m-Government services. Tien Nguyen et al [68] argue that despite these disparities the upsurge of mobile technologies will influence governments to invest in enhancing service quality and efficiencies. Citizen centred services, that are part of a wider trend have risen the demand for mainstream delivery. However, governmental challenges and factors such as those in figure 3 make mgovernment a subject of policy decision. The next few chapters describe some of the technological innovations embarked by public services and look into the wider implications for a delivery at pace.

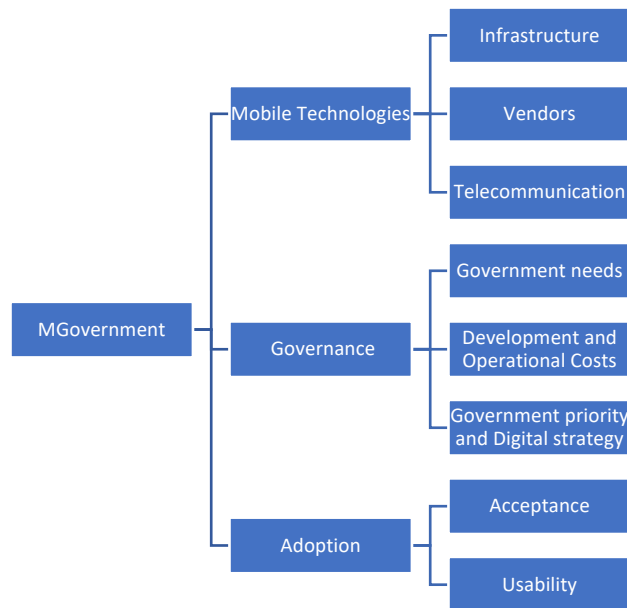


Figure 3: Factors influencing MGovernment

2.5.3 User Satisfaction and Perception in mobile adoption

There have been several studies conducted on the adoption of egov services across the world. Challenges and success have been studied from both the service provider as well as the citizen as the stakeholder perspectives. Studies ranging between 2003 and 2011 focused on the merits to the governments. Later years have seen an increase in publications pertaining to user satisfaction and their implications on the diffusion of eGov services.

Chen's[43] research on the technical factors that influence user satisfaction in mgov conjecture that time criticality, location sensitivity, MMS and personal control, portability, ease of use and active control as functionalities to measure user satisfaction in mobile services. We can ascertain that these functions have positive impact on transparency, procedural fairness and information accuracy, behavioural patterns and social issues such as trust, risk and perceived value[69][70].

Carter et al's [71] research on the citizen's perception of e-Government in the UK and United states is a study on the use of services offered by the DVLA in UK and DMV in the US. 250 UK citizens were surveyed. Their research primarily focused on the effect of trust and risk. The premise being EGov adoption and the population from two culturally similar countries. They have concluded that trust is not a construct for use of e-services but usability is a construct. Users were more inclined to use the service based on ease of use, navigation and information quality. They also emphasise that, value added services through advancement in technology that would enhance accessibility, availability and convenience would result in increased diffusion, especially in lower socio-economic segments. Their findings confirm the theory that efficiencies achieved through citizen centric online services lead to effectiveness in service, cost and process improvement for government organisations. They recommend Governments need to respond to citizen's needs. It could be argued that in today's world of cloud based services, Big data and information sharing, risk and trust are major factors. Trust tends to develop from positive past experiences and popularity which in turn diminishes the risk factor. However, over the years we have seen overwhelming success of crowd sourcing and social media platforms despite the risks of data piracy and privacy which confirms Carter's findings. Public services should have robust, responsive development strategies to instill trust for sustained success.

- Demographics

Shareef et al[72] in their study have confirmed the demographics on the popularity of use of the e-government and M-government model. E-Government is widely used by educated and technologically savvy people on the desktops while M-Government is used even by technologically inept users. Their findings indicate an increase in MGov adoption with mobile apps seeing a 49% increase higher than web apps.

2.5.4 Technologies within Public services

- *Responsive web and mobile web*

Figure3 indicates that Mobile web with responsive web design appears to be the most favourable choice of service delivery for almost all public bodies for core services.

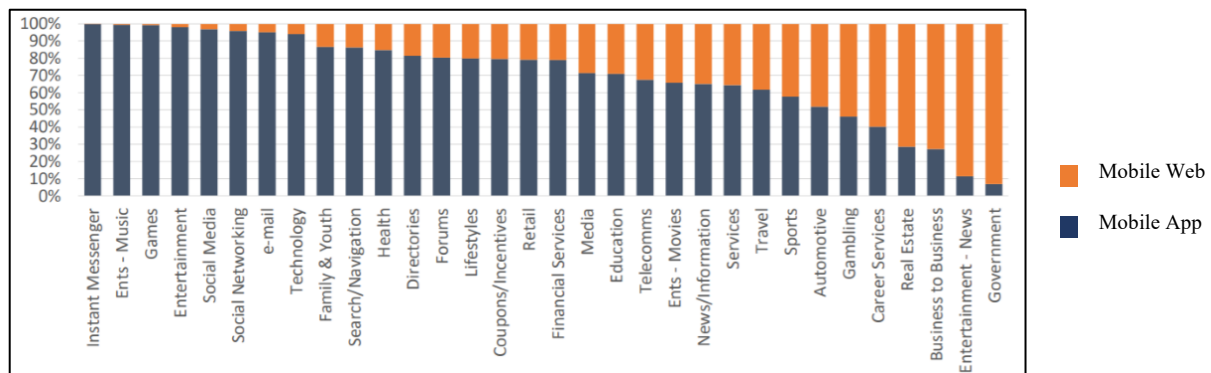


Figure 4: Mobile App vs Mobile Web Source: Comscore MMX Multi-Platform, June 2019

Arkansas.gov provides a digital government ecosystem experience through responsive and interactive site integrated with chat-bot, delivering more than 900 services. Static generator technology enables page loads of less than 1s emphasising the benefits of building static html pages. Since the launch of the redesigned site 2018, site visit analytics picture an 8% increase from mobile devices. Satisfaction on the experiences is at 96.6%. [73] and is testimony to the increased adoption of a digital Government and the ever-increasing appetite for mobility.

- *Mobile Apps*

Considerable research has been done in the adoption of M-Government, more so from developed nation's perspective. It has been found that the implementation of tools fall into a broad spectrum owing to the disparities in infrastructure, economy, digital literacy and voracity of citizens to consume information through technology[67]

- *MGov initiatives in national and local governments*

A ten year EU report identifies the main trends in the overarching EGov that encompasses all categories of online services including Mgov [64].

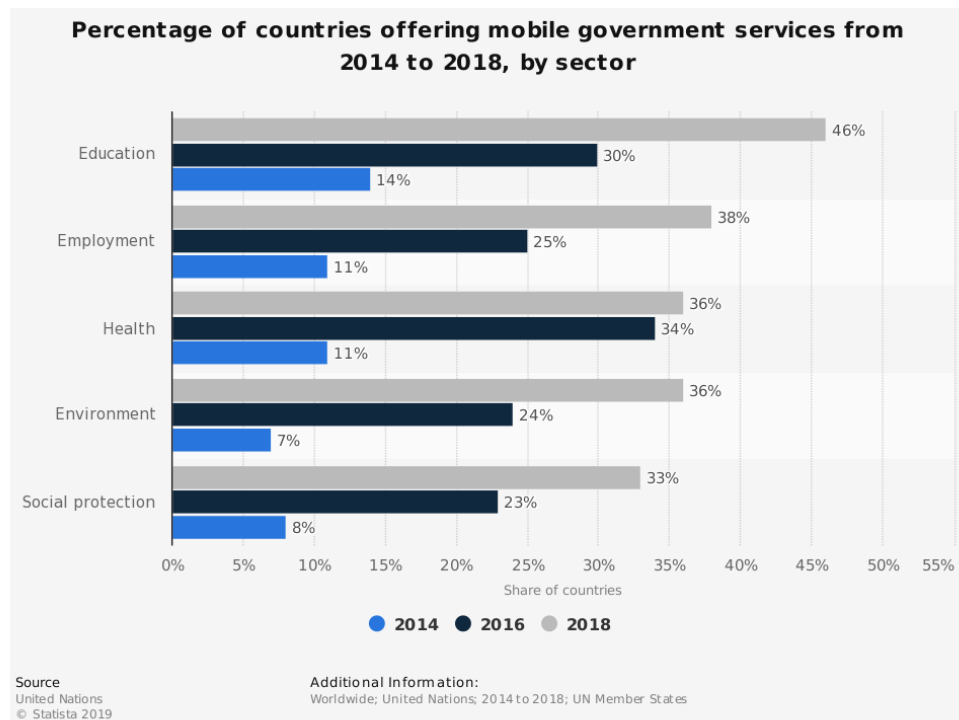


Figure 5: Percentage of countries worldwide

Web 2.0 technologies have seen an upsurge with public services harvesting on collective intelligence through citizen engagement[64]. Nils Walravens' mapping grid Brussels Government's mobile apps compares mobile initiatives and assist in strategy management[74].

A sample of MGov initiatives is given in the following table.

Name	Authority	Platform	Special features
Arkansas.gov		Mobile web	Static html files
Autolilija	Finland –Transport and Communication Agency	Android app	Mobile driving licence and vehicle services QR code access
DLS - Check driving licence	DVLA - UK	iOS, android	Add to apple wallet
Gov.MT	Malta Government	Mobile web	
HMRC Tax Calculator	HMRC - UK	iOS, Android and web	
Leeds bin	Leeds council - UK local government	Android app	Google calendar integration of dates and map
Noise App	Multiple councils	iOS, Android	Record and report antisocial incidents
Oxford council app	Oxford - UK local government	iOS, Android	
Roadside photo - Pronto	Sussex and surrey police, UK		
Singpass	Singapore	iOS, Android	Digital identity with biometrics
UK Exit -ID document check app	HomeOffice - UK	Android app	Photo and Document scan
Utah Legislature Bill watch	Utah gov	iOS app	Apple watch Push notifications Search Subscribe to watch

Table 1 EGov/MGov adoption

From an organisation’s perspective the cost of developing apps is the major deterrent. This has led to the increased implementation of Responsive Web Designs that work well in mobile and requires only a single code base.

2.5.5 Digital Government in UK

The UK Government has several future technology initiatives that encompass Artificial Intelligence, Machine Language, Cryptocurrencies, Blockchains and wearables. The Department of Work and Pensions have trialled Govcoin, digital money for benefits payment. The Ministry of Defence work on AI and robotics projects. The Food standards agency uses Blockchain to monitor cattle[75]. Gov.Verify, Gov.Pay and Gov.Notify are some services that have been implemented for cross departmental services enabling citizens to easily transact.

All digital services deployed through the GOV.UK with the above exceptions listed in Table 1 EGov/MGov adoption are responsive sites that has reached a wider audience. The design is minimalistic with no images, content-centric based on extensive user research and 100% accessibility compliant. However, it lacks features that are inherent in mobile apps that optimises the use of smart phones to capture interactive functionalities. Since its launch in 2013, traffic from mobile has risen to 43% from 18% and desktops dropped to 43% from 73%. Figure 5 below depicts the web traffic to the site by device type. This steady growth can be attributed to the changing model of information access by people.

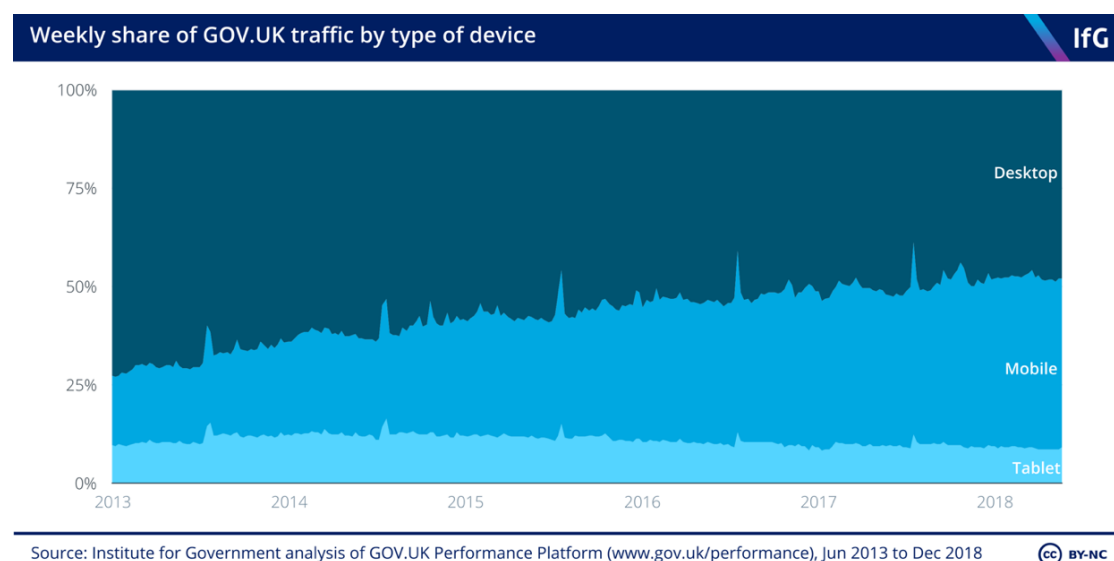


Figure 6: GOV.UK web traffic [8]

The Government Digital Service in the UK discourages the development of native apps and only consider it on exceptional business needs that benefit the citizens[15]. Income tax service provided by the HMRC is available as an iOS and android native app for citizens to manage their personal tax and benefits. The EU exit document check app allows for citizens to scan identity documents with near field communication technology as part of the settlement process. This is however currently available as an android app only. One attempt to provide a mobile app experience by the DVLA is the digital licensing service as part of licence check for drivers by third parties. This has been developed using some aspects of PWA and hence is discussed in chapter 5 Results and conclusions.

2.5.6 Conclusion

Several research compare the usability, performance and security principles between the web and native mobile apps, most of them that are in use in the industry. There is insufficient research analysing the benefits or demerits of a PWA as compared to a Responsive Web Design. With all Governments hosting their services in the web and the wide acceptance of EGov and MGov, it is apparent that public bodies need to think of better ways to continue to be digital. Although there is an appetite for mobile apps, there is a reluctance due to financial constraints. The need for a mobile app is justified by the nature of the service and the need.

It must be remembered here that native and hybrid apps are installable from the app distribution platforms and in a rather convoluted way if they are not indexed in any of the search engines. Native apps requires a certain level of commitment towards engagement from users that compels users to install them by trading off their storage and possibly security to some extent[76]. However, the second quarter of 2019 has also seen that user retention has decreased by 6% since last year and on average, a user uninstalls an app within 5.62 days after their last

session. Kho[49] argues that PWAs provide plausible solutions for organisation with their ease of discoverability, less storage, lower development and marketing costs.

Hence it is proposed to investigate if PWAs are the way forward to provide the benefits of an app without compromising on development costs, accessibility and usability. The research allows for analysing the added value to the citizens in terms of user experience and to the business in terms of ease of operations, communication and successful service.

3 Research Methodology

This section describes the philosophy, approach, strategies and data collection methods employed in this project.

3.1 Introduction

Seaman [77] describes research in software engineering as being complex and difficult as it involves issues arising from technical aspects associated with both machine and human actions that is defined by software development.

The complexity is captured in table 6 produced by Stol et al, where every software engineering research from 2002 have involved a hugely differing research methods. As software engineering (SE) tends to be techno-social in nature, the overall theme appears to be that every research is unique, and the methodology employed are on a case by case basis. The normal theme of one method fits all basis of generic research may not be best suited for the research.

Glass et al. 2002 [4]	Zannier et al. 2006 [27]	Tonella et al. 2007 [28]	Easterbrook et al. 2008 [9]	Wohlin et al. 2012 [11]
Replicated	Controlled experiment	No experimentation	Controlled experiment	Survey
Synthetic	Quasi experiment	Experience report	Case study	Case study
Dynamic analysis	Case Study	Case study	Survey	Experiment
Simulation	Exploratory case study	Quasi experiment	Ethnography	
Project monitoring	Experience report	Randomized experiment	Action research	
Case study	Meta-analysis	Observational study		
Assertion	Example application			
Field study	Survey			
Literature search	Discussion			
Legacy data				
Lessons learned				
Static analysis				

Figure 7: Research models adopted [78]

This research on PWA attempts to blend the methods that best suit the purpose as the research wishes to explore the application in a government setting where the delivery is focused on facilitating public with regulatory compliance.

3.2 Postpositivism and Qualitative Inquiry

Postpositivism engages a scientific approach to research believing in multiple perspectives from participants with a cause and effect inquiry. Pragmatism is to focus on consequence and arrive at means to get there. It does not fit into the paradigm of either quantitative or qualitative and but select the best methods that suit [79]. SE is a rapidly changing field and statistical quantitative inquiry of longitudinal study would be considered redundant given the short timeframes. Qualitative study relies on multiple sources of data and the meaning that the participants hold about the issue. This study was about studying the behaviour of the system as well as the participants and their impact. On this premises, a mix of qualitative cross-sectional study with a quantitative element was used in analysing user experiences.

3.3 Approach

Whilst exploring the best approach one finds that a deductive reasoning tends to start by generalising an idea in order to use it for a specific purpose. Whilst Hyde [80] suggests that in most cases a qualitative research tends to follow an inductive process, argues that using a formal deductive approach to a qualitative approach is perhaps a better way of assuring conviction in qualitative research findings. The deductive approach is based on previous researches and tests previously established research hypotheses instead of establishing new hypotheses. It is important to note that this method accepts or rejects a previously established hypothesis instead of formulating a new hypothesis. This research followed a ‘Deductive’ approach which in this case is synonymous to the primary object proposed by Run [81] as that of ‘Exploratory’. The approach was to determine if the following generally deduced hypotheses were true or false.

- PWAs are the innovative and efficient cross-platform approach for mobile delivery.

- Mobile apps are not popular in Government due to cost and operational implications.

3.4 Strategies

As an exploratory method SE is about evaluating the appropriateness of a technology within a bounded system. Run [81] in his guidelines proposes an approach that involves ‘Case Study’ which is ‘Exploratory’ and that adopts a ‘Flexible’ ‘Qualitative’ methodology. SE is an empirical study coinciding with evidence-based research. It is imperative to gather an understanding of the work carried out earlier in theory and in practice, thereby contributing in conceptualising the design of the research and the results themselves. It is interesting that Bowen’s [82] description of document analysis goes a step further than what is generally proposed by the Software Engineering Group for a ‘Literature Review [83]’. Following his suggestions, with technical specifications and a regulatory public governance context, and the nature of documents that will be available for the purpose, a mix of documents and literature review has been performed to elicit empirical knowledge of the social and historical context of PWA as well as identify prior research. Creswell[79] argues an intrinsic case study as one which focuses on the case itself and in turn is the artefact. Thus, software development forms the main aspect of the case study. Action research by way of making changes aims to improve a specific setting. To evaluate the case, it was required to build a system with a view to improving a service and hence a PWA was developed.

3.5 Data Collection

Various forms of data collection methods are employed in a qualitative study. In the instance of a case study, it is from observations, interviews and documents either printed or audio-visual. Sample survey, one of the most popular method in SE is useful in generalising an idea. This could produce qualitative as well as quantitative data in the field study setting. A combination

of these methods was employed with a purposeful sampling strategy, selecting a group of participants and sites who would be able to provide purposeful understanding[79]. The following table highlights the methods used.

Source	Method	Tools used	Data type
Product owners, Software managers, Software engineers, Business analysts	Open-ended Interviews, Questionnaires, Social messaging systems	Face to face and note taking, Email, Slack	Qualitative and quantitative secondary data
System users	Short surveys, Observations as an observer and participant, Visual recordings (eye tracking)	Survey monkey, eye gazer	Qualitative and quantitative primary data
Documents	Public documents, system requirement documents	Websites, Confluence, Email	Qualitative and quantitative Secondary data

Table 2: Sources of data

3.6 Conclusion

This chapter was a brief account of the research methodologies employed in conducting this research. It was a positivistic and pragmatic theory following a cross-sectional qualitative, deductive study owing to the short turnaround time and the nature of the case study. This was complemented by an extensive document and literature review and software development. The next chapter explains in detail the software methodology and logical steps followed with the data collection embedded within usability testing.

4 Development Methodology

This chapter discusses the development methodology chosen for this project, broken down into the four distinctive phases of the software development life-cycle (SDLC).

4.1 Introduction

SDLC have varied nomenclature in the different development models that exist. Regardless they all follow the same activities.

- Identify the problem.
- Identify the requirements criteria that could possibly be formulated into a solution.
- Identify the architecture and design that would satisfy the requirements.
- Construct and implement the solution guided by the design.
- Validate and evaluate the solution.
- Maintain and iterate.

The choice of a development methodology is dictated by several factors including the size of the project, timescale, resources and cost. The chosen methodology is Feature Driven Development model(FDD) integrated with Incremental development and Reuse-oriented Software Engineering.

The Waterfall methodology conceived by Royce is linear- sequential process with predictable deliverables at every stage [1][2]. Requirements analysis and planning dominate the development process. The pitfall lies in risks being identified at a very late stage. Iterative development involves building a skeleton system with minimum functionality and building on top of previous iterations. Conversely an incremental method relies on building one functionality at a time and adding on to increase functionality. An Agile approach can be said

to combine the principles of iterative and incremental development. The following sections explain the approach and methodology used and the rationale behind their choices.

4.2 Reuse-Oriented Software Development

Almost all software utilises a number of patterns that are similar including code, design and requirements. Frameworks, libraries and open source development exist for this very purpose where they are modified as needed. Reuse encourages the application of patterns. Integrating reuse into FDD would enable speedier development with tried and tested patterns. The core functional requirement of this service and user interface design will be reused from the current system and wrapped with additional PWA features. The user journey of the core service is thus the same as the existing service. The architecture follows the same client server model. However, the difference lies in the language and frameworks used for the web application where ReactJS replaces Ruby on Rails.

4.3 Feature Driven Development

Feature driven development (FDD) embraces the features of aforementioned agile practices. The very adaptive and incremental nature, short iterations and user centric model makes it an ideal candidate for the development methodology for this project. Somerville[85] states that plan-driven development methods like waterfall are not suitable for projects with a short turnaround time and not geared for rapid development. This is a small project which addresses only a part of the vehicle enquiries service with a short turnaround time. Although Zahid et.al.[86] recommend FDD for larger projects with fewer requirement changes, FDD was chosen, as the core business requirements have been derived from the existing system and not bound to change frequently, albeit any UI design considerations. Further, the fundamental concept is propelled by the decision to implement the predefined features of a PWA.

The 5 process in FDD as illustrated in figure 8 can be further grouped into two phases for the purpose of this project. Phase 1 comprised developing and planning the feature and phase 2 design and build the feature.

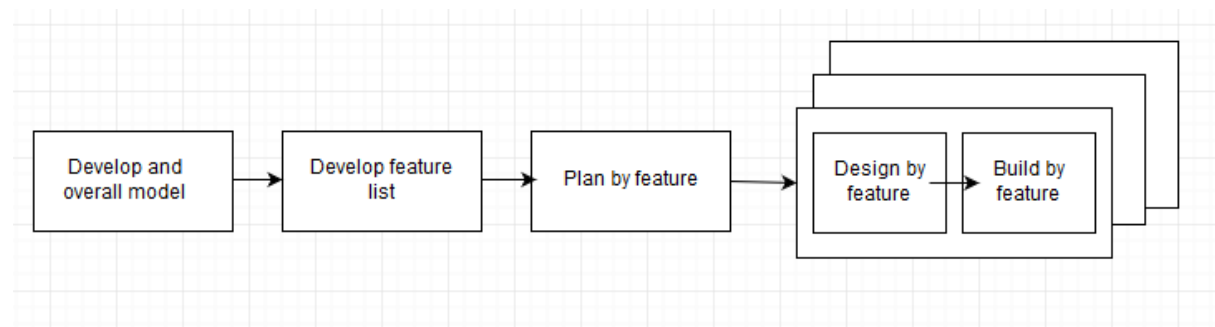


Figure 8: Feature Driven Development[87]

4.3.1 Requirements Engineering and Modelling

Arrive at an overall model, a prioritised feature list and plan based on business requirements.

Using ‘object modelling in colour’ technique the business domain model was identified [88].

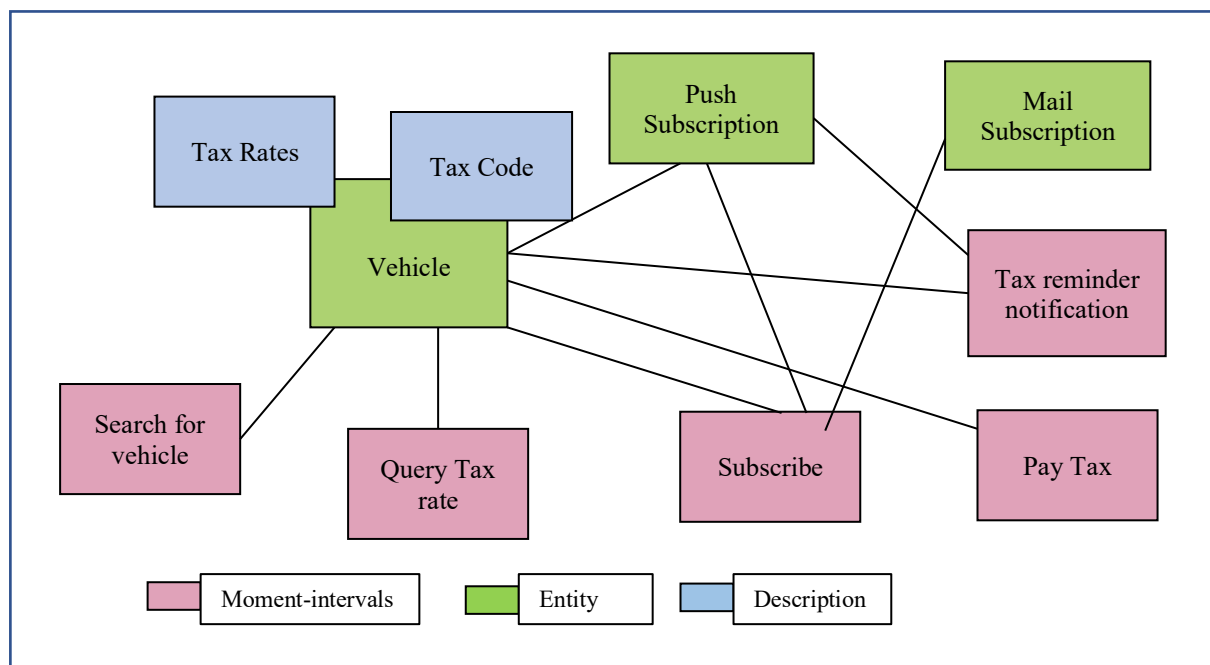


Figure 9: Domain model

Since the artefact is to implement a PWA as a proof of concept, the overall design would reflect the current existing live service.

4.3.1.1 Requirements Elicitation – Business Requirements

Requirements engineering for this project was in the form of unstructured interview with product owner, developers and usability testing of the live site to gather requirements and usability data.

Unstructured interview:

Discussions with the DVLAs vehicle enquiries services product designer, user insight team and business analyst were carried out. This gave an insight into the design and usage of the current live system which in turn defined the requirements.

Document review:

The current vehicle enquiries system, existing in two versions, Vehicle Enquiry Services Web, API caters to three different types of users with different levels of authorisation. The web serves all individual users and provides open data. The API serves developers as well as authenticated users with a large dataset and a non- exhaustive example dataset is given in the following table.

For the purpose of this project only the Basic level of service is considered and the hence the user and business requirements reflect that. The business requirement for this project does not include the collection and storage of the vehicle details but requires retrieving data from an existing data store.

Planning and designing by feature followed a pattern closer to Extreme Programming but with no roles and class ownerships. It rather followed a Kanban model with one feature at a time A sample of Kanban board with user stories is included in Appendix II.

4.3.1.2 Requirements elicitation – User requirements

Based on the findings from interviews, a user journey was mapped out. A number of use cases exist including online and offline view. Figure10 gives the overall user journey with add to homescreen and subscribing to reminders as epics.

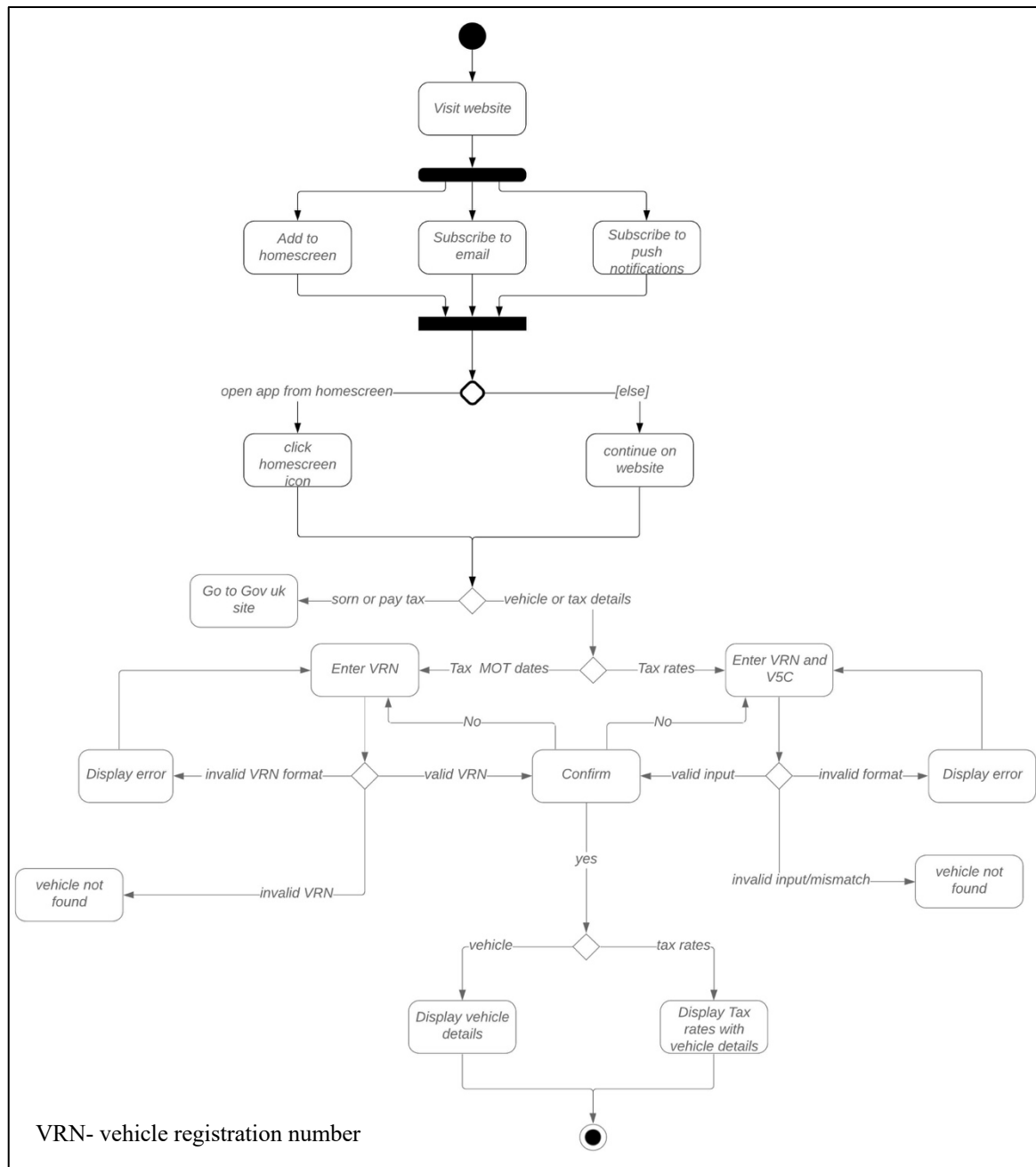


Figure 10 User journey - UI flow

Figure11 depicts the journey for the use case when the app is accessed with no internet and fulfils the offline access criteria.

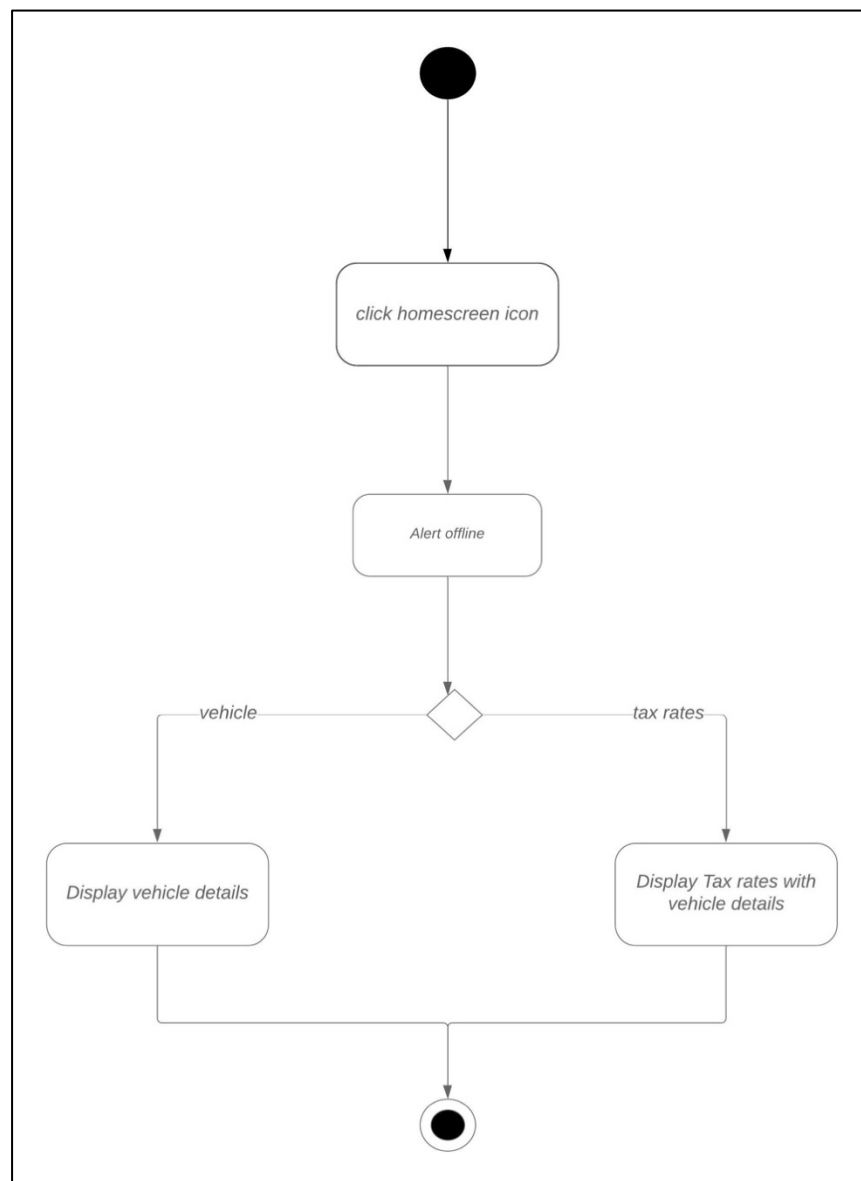


Figure 11: Offline user journey

As identified in the literature review, Google specifies the features that a website should deliver in order to be classified as a PWA. The MoSCoW mechanism of nominal scale is used to prioritise the list. All items within a group have same priorities in a nominal scale [89].

Transforming the methods from the moment-intervals above resulted in feature sets. A high order user story feature set is illustrated in table3.

Must have	Input vehicle registration and retrieve vehicle details. Add the application to device homescreen and access the site from the icon.
Should have	View some or all pages in the absence of network ▪ Homepage ▪ Vehicledetails page ▪ Tax rates details page Subscribe to push Notifications Receive push notifications
Could have	Home page with menu for navigation Navigate to Gov.uk view tax rates external link Navigate to Gov.uk SORN external link Subscribe to email notification
Won't have	Payment of vehicle tax Log in authentication

Table 3: Features list (MoSCoW)

4.3.1.3 Architectural Model

A 3-tier layered architecture better describes the vehicle enquiry system. This will follow a client-server model with the PWA client communicating with a web server through API calls.

The benefits to using a 3-layer architecture with disparate layers include scalability, performance, reliability and availability. Different parts of the application can be hosted on

different servers and scaled or modified independently. Fowler suggests the logical structure of the layers facilitate them to be physically placed together or apart [90].

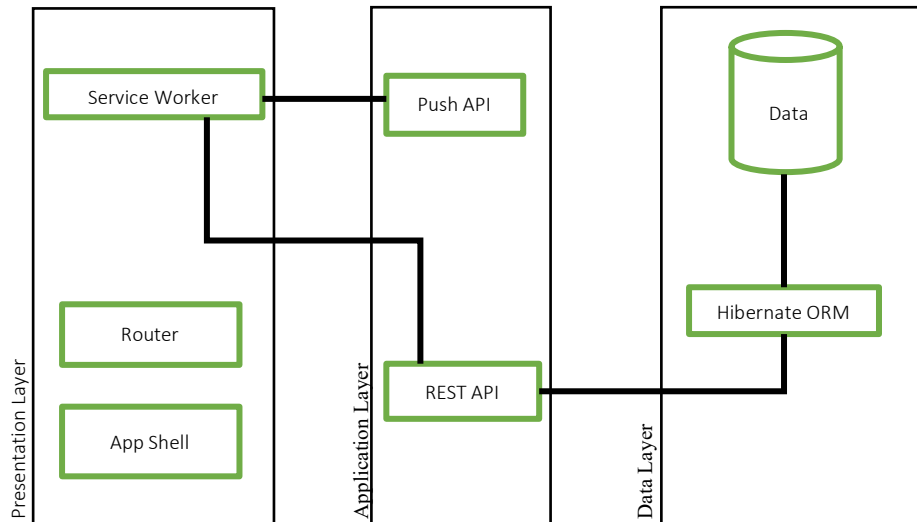


Figure 12: Layered Architecture

Figure12 depicts the layered architecture to reflect the functionalities of the system. The web app forms the front-end with a service worker that sits as an interface between it and the application layer. The Push API sits in between the server and the service worker facilitating push messages. The persistence layer comprises the data store with the Vehicle and push subscription details serving the data through Object Relational Mapping. Figure13 gives a breakdown of the components in the system and their relationships.

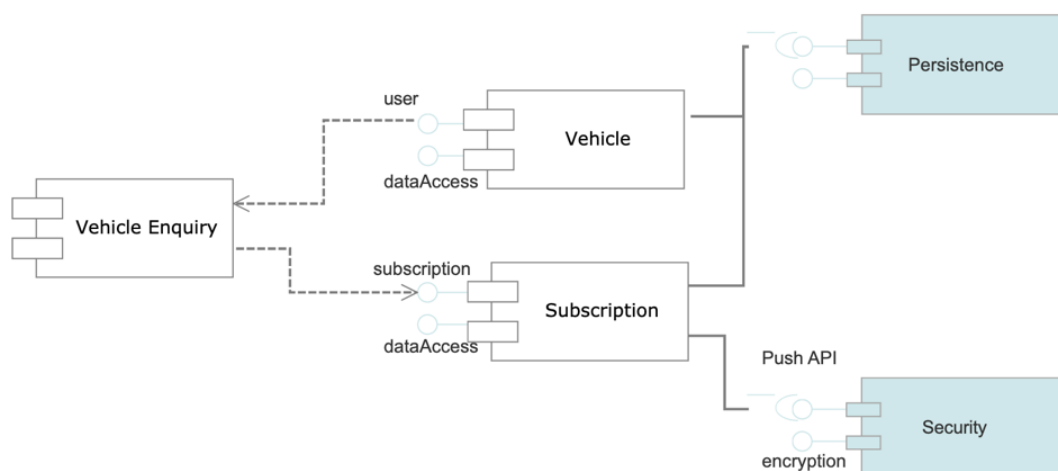


Figure 13: Component Diagram

4.3.1.4 Database Design

A Relational Database model (RDBMS) is proposed with a top-down approach. A RDBMS model stores data in related tables, with non-redundant data [91]. Normalisation techniques ensure that data records are not duplicated in a given relation. RDBMS are relational based as opposed to NoSQL databases that are defined by documents and collections rather than tables. Although NoSQL databases have gained popularity, RDBMS is chosen owing to the simplicity of the tables required and lack of defined standards for NoSQL. The current system captures a much larger data set in the database which includes data relevant to individual, industry, enforcement, environment agencies and so forth. The DVLA updates vehicle details in a separate registration service and is not part of the enquiry service. Hence, for the purpose of this dissertation, a condensed data set was pre-populated in the database. The rational being, the goal is not to conduct a study on the actual data base or the backend but focus on the front end with the required business logic in the backend server. No vehicle keeper details no user authentication are considered for this case study. User's subscription to a vehicle is mapped with the vehicle registration number used as the key as illustrated in the ERD.

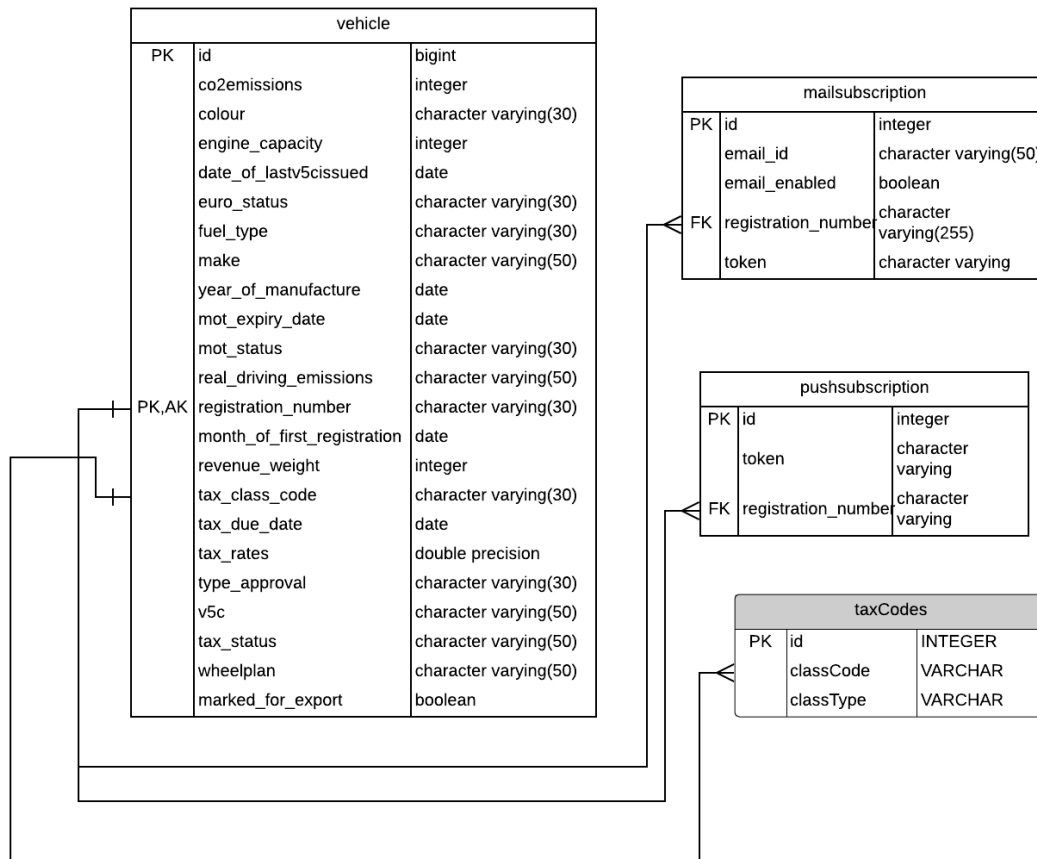


Figure 14: Entity Relationship Diagram - vehiclesdb

4.3.2 Build – Technical Stack

Armed with the conceptual design formulated into features list and components, the next phase was the actual implementation. Implementation followed an incremental process with the core enquiry feature and homepage icon as the minimum viable product. The high-level features were broken down into granular level and technical stories. Each of these formed just in time tasks in a Kanban model. The following sections describe the technical stack utilised in the build process and the rationale behind the choice.

4.3.2.1 Reactive Components with ReactJS

Somerville [85] recommends reusing of existing code as much as possible in the form of frameworks and libraries. A number of popular frontend frameworks such as Angular, Vue and

ReactJS are usually built with a NodeJS with Express server for the backend. The chosen framework was ReactJS with Create-React-App for the web app and Spring with Java for the backend. A Postgres database comprised the data layer.

This project could potentially be a prototype for a large and complex one. One could look into bringing all the services of DVLA together with customer data as the single source of data and link all driver data and vehicle data to provide a uniformed seamless service. This would make it a complex development process and it would be apt to use ReactJS which fares well in such scenarios. It also has an easy learning curve and due to the limitations in time, felt the appropriate framework to use.

4.3.2.2 Environment

Setting up the environment required a transpiler and bundler to render the JavaScript. Babel transpiles JavaScript Extension (JSX) and JS into a single ES5 file eliminating compatibility issues over the web. Webpack, static module bundler that runs on node.js, builds all the dependencies required in the project and bundles all the assets into an output. One could start with a skeleton project and configure Webpack and Babel or use a library that configures the environment and scripts.

On the contrary, Create React App is a framework from Facebook, that saves time-consuming setup and configuration. Using Webpack and Babel under the hood, it generates only the necessary files and the file structure. Installation of latest versions of react-scripts that manages all development dependencies, React and React-DOM are done along with the initiation of the project. Caching is also taken care by default with Workbox. Hence, the project was developed using Create-React-App.

4.3.2.3 Mobile-first design

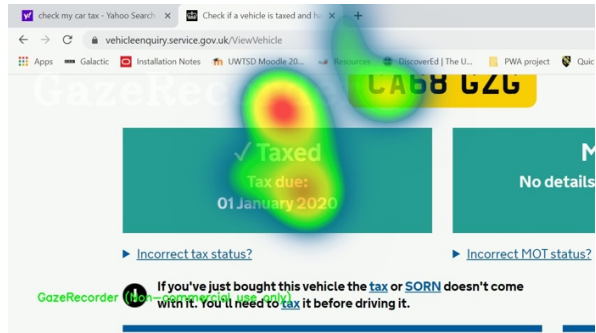
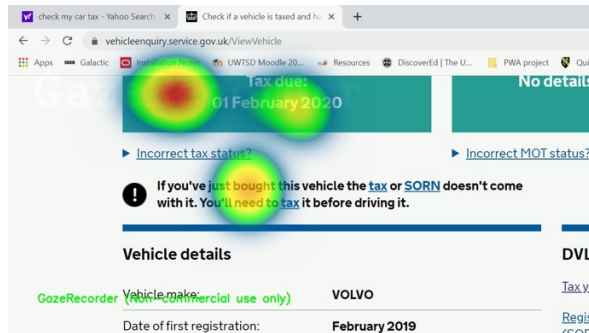
Progressive enhancement was the mobile first design approach for mobile browsers and then enhancing for complexity. Current mobile browsers are powerful enough that the focus has shifted to designing for miniaturisation.

User Research

The fundamental goal of this project, being the provision of better human-computer interaction, it was necessary to gauge the user experience of the current system. To this effect, a timeboxed cognitive walkthrough exercise was carried out with a small number of users. Users were asked to follow a journey to complete tasks on the site. Gazetracker was used to track eye paths and identify the content that most users viewed or ignored on a page. Sample heat maps from the test are shown below in figure 15 and 16.

Eliminating Noise

With space being critical on smart phones and other smaller mobile devices, it was necessary to design to context. One would agree that web users skim through a page for content of interest to them [92]. This was evident from the eye-tracker results from the eye tracker results. Content ignored by all participants was removed to declutter the page. For instance, links to sign up for MOT reminders and SORN that were removed. A sample of two sets of gazerecorder readings in figures 15 and 16 suggest the key points of interest on the page.



A questionnaire given to participants helped

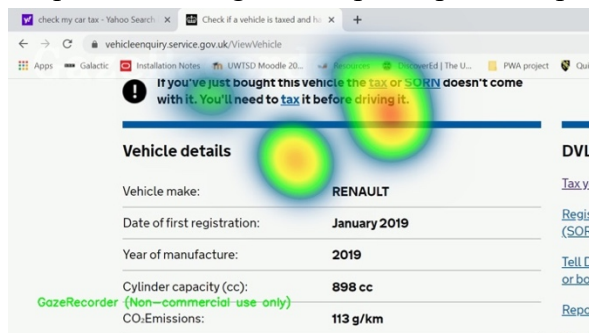


Figure 15: Gazerecording user1

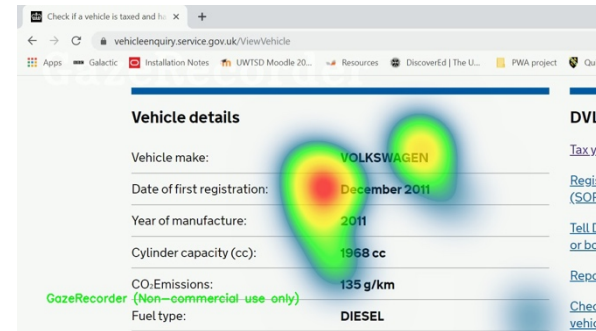


Figure 16: Gazerecorder user 2

displayed on the vehicle details page. This led to changing the UI to list the top 5 details that most people would be interested in the top and display the rest in an accordion that could be expanded or collapsed on click. Although, this meant additional clicks for users, the survey result was indicative of little or no impact on user experience as the basic level users were not interested in those details anyway. Their primary goal was to view the tax and Mot status and due dates.

UX screen prototyping

Prototypes of screens that mimicked a website were created using Sketch. The prototype included an add to homescreen button and two designs to view the tax rates. All users questioned responded with a preference of design1, but with the tax and MOT due dates to be in text rather than coloured panels.



Figure 17: tax rates prototype

The GovUK stylesheet was reused to keep it similar to the current site and conform to accessibility standards. Responsive design ensured that content was resized to viewport size. However, the current site does not contain a menu nor a homepage for the service and considering that mobile users would benefit from one and also to provide for any scalability of the app with additional functions, a homepage was designed. A list of links was added to keep in line with the GovUK design system.

4.3.3 Technical architecture of a PWA

An app manifest defined the app shell that would be installed on the device and a service worker was implemented.

```
{
  "short_name": "Vehicle Tax Enquiry",
  "name": "Vehicle Enquiry Service",
  "icons": [
    {
      "src": "assets/images/favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "assets/images/icon-144x144.png",
      "type": "image/png",
      "sizes": "144x144"
    },
    {
      "src": "assets/images/icon-192x192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "assets/images/icon-512x512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/",
  "display": "standalone",
  "theme_color": "#74818c",
  "language": "en-gb",
  "scope": "/",
  "background_color": "#e1edf7",
  "orientation": "portrait",
  "prefer_related_applications": false
}
```

The app when installed appears on the desktop and homescreen on mobile. Figure 18 illustrates the app installed in the apps launch centre on Windows10.

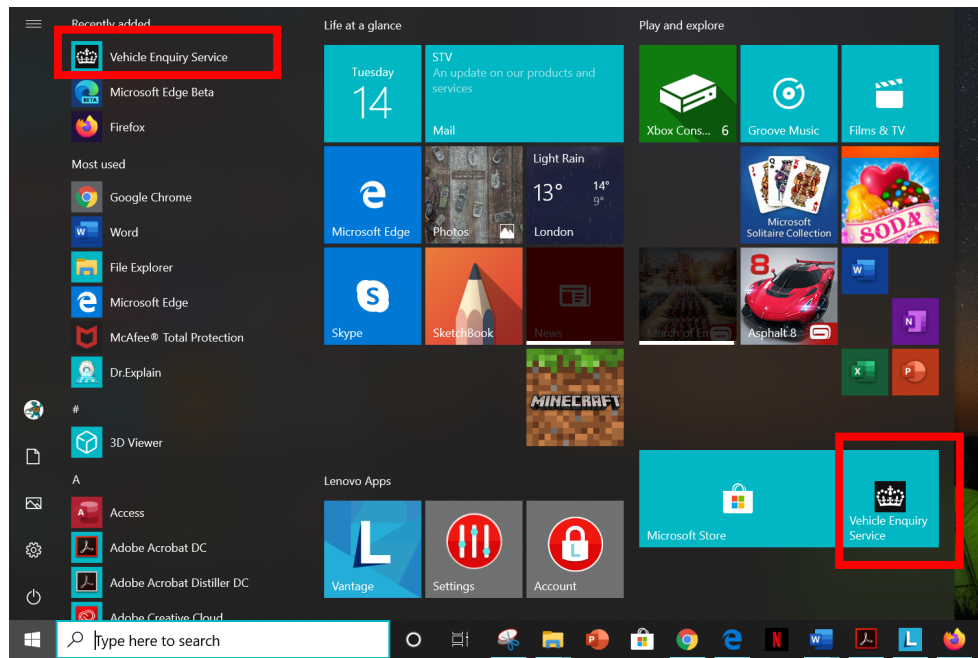


Figure 18: PWA on Windows10

4.3.3.1 Caching Strategy with Workbox for Offline-first

One of the features of a PWA is to maintain a cache for fast and reliable loading of pages even in shaky or no network conditions. A cache first strategy was implemented. The differences between strategies and an illustration of cache-first strategy is depicted below.

Caching Strategies[93][94]:

- Cache-first strategy – As the name implies, cache is checked for a match and if no match the request is routed through the network. Figure19 illustrates a cache-first strategy.
- Stale-while-revalidate strategy - Cache first but simultaneously go to the network to update cache if new data available.

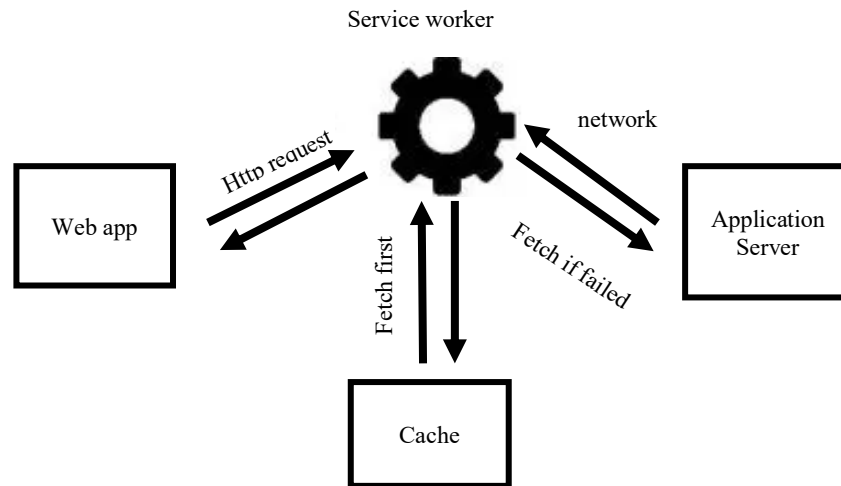


Figure 19: Cache-first strategy

Service worker if properly configured makes the web fast and reliable[94]. Precaching, caches critical assets such as css, html and JS ahead of time, involves creating a list of URLs in the manifest file on build and follows the cache-first strategy.

The issue though was Create-React-App generates a service worker and precache manifest at initial build and lacked flexibility to customise the cache. This meant the static html, css files and logos were cached, rendering only the home page and the Start page from cache. One could say this does satisfy the PWA requirement as some pages are available when offline. However, to make the app work completely offline, the solution was to inject the manifest using Workbox. This was achieved using the Workbox webpack plugin, as it allows for customisation. The code sample demonstrates the caching strategies used.

```

workbox.routing.registerRoute(
  new RegExp("/*.*"),
  new workbox.strategies.CacheFirst({
    cacheName: "my-cache",
    plugins: [
      new workbox.cacheableResponse.Plugin({
        statuses: [0, 200]
      })
    ]
  })
]

```

The choices available for caching dynamic data with a post request were to use the Cache API for URLs and IndexedDB, a low-level API for client-side database, to store the data in the user agent. Users would have to enter VRN and/or V5C numbers even when offline and follow the online user journey. An improved version was to take users to the details page without having to enter VRN or V5C when offline, thereby rendering cached results. This was implemented with the use of IndexedDB to save and retrieve data on the client side when offline.

Add data record	Retrieve data record
<pre> let openRequest = indexedDB.open('vehicle_db', 1); openRequest.onupgradeneeded = function (e) { let db = e.target.result; console.log('running onupgradeneeded'); if (!db.objectStoreNames.contains('vehicle')) { let vehicleOS = db.createObjectStore('vehicle', {keyPath: 'registrationNumber'}); } }; openRequest.onsuccess = function (e) { db = e.target.result; addItem(data); }; // add data to db function addItem() { let transaction = db.transaction(['vehicle'], 'readwrite'); let vehicle = transaction.objectStore('vehicle'); vehicle.clear(); let request = vehicle.put(data); request.onerror = function (e) { console.log('Error', e.target.error.name); }; request.onsuccess = function (e) { console.log('data in indexeddb'); } </pre>	<pre> export function getIndexdbData(setDetails) { console.log("i am here"); let myRecord = []; let DBOpenRequest = window.indexedDB.open("vehicle_db", 4); DBOpenRequest.onerror = function (event) { console.log("cannot open"); }; DBOpenRequest.onsuccess = function (event) { console.log("opened"); // store the result of opening the database in the db variable. let db = DBOpenRequest.result; // Run the getData() function to get the data from the database let transaction = db.transaction(['vehicle'], 'readwrite'); let objectStore = transaction.objectStore("vehicle"); let objectStoreRequest = objectStore.getAll(); objectStoreRequest.onsuccess = function (event) { myRecord = event.target.result; setDetails(myRecord); } </pre>

4.3.3.2 Serverless deployment:

Cloud services provided a very cost-effective solution for deployment of the app, the web service and the database. Amazon Web Services (AWS) Amplify was ideal for the app as it enables continuous deployment with its node engine and serves the app through SSL. Hosting the Spring web server in Elasticbeanstalk did conflict with the webapp over https incompatibility. An alternate option was to host the backend in Heroku which generated its own database and served the domain through HTTPS eliminating the need for a separate certificate.

4.3.3.3 Push Notification

Whilst it is not a requirement for a PWA to deliver push notifications, it was deemed fit to implement owing to the business need that individuals be kept informed of vehicle tax renewals. Push messages with payload require data to be encrypted with the public and private keys and JWT. The push library was utilised for this purpose. With the user granting permission for notifications and the subscription details stored in the database, a scheduler scanned for tax dues at midnight every night, generated and pushed messages through the push service. The sequence of operations is depicted in figure20. The design was to send messages only to intended devices whose tax was due, identified by the device token. The service worker utilises the notification API to display the notification on the webpage with the click event handler directing them to the GovUK page for tax payment.

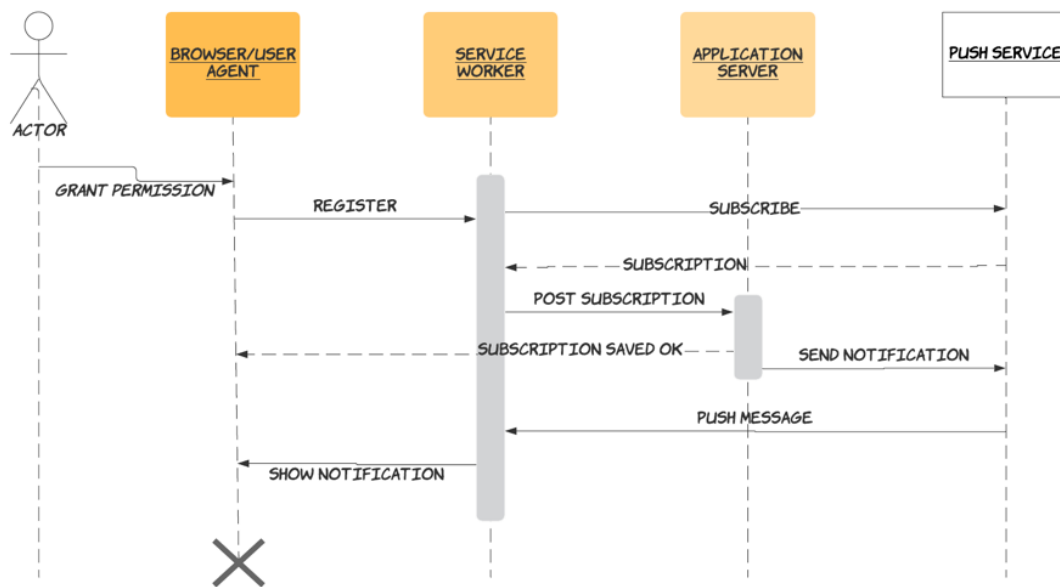


Figure 20: Subscription based notification webpush

Posts to the push API from the server resulted in JWT error and hence an alternate model was implemented by routing the push through Google's Firebase. The database schema had to be amended accordingly.

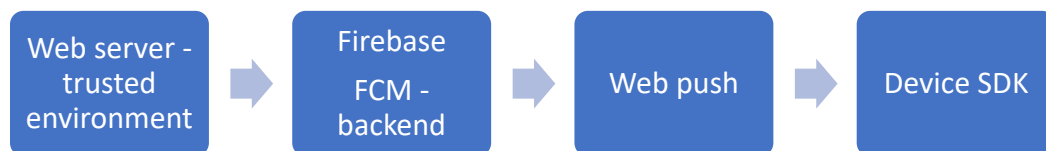


Figure 21: Push Notification through Firebase

4.3.4 Test by feature

Backend:

JUnit framework was used for testing the Spring component with a limited number of unit tests. Blackbox testing using Postman identified bugs in code and ensured the business logic modules and data persistence modules functioned properly.

Frontend:

Exploratory and usability testing were carried out with happy path testing with input data validation. Limited informal usability testing was also carried out with a view to receiving feedback for successive iterative improvements. This form of testing follows the ideas of Steve Krug [92] with periodic usability testing with an aim to detect and fix issues rather than collecting quantitative data.

The following suggestions were implemented in the next iteration.

- Add to homescreen button was removed and added as an extendable link and button on the homepage. Add button was disabled in mobile devices.
- User permission prompt for notifications was triggered on user click event by adding a subscribe button.
- Selecting the No option in the vehicle confirmation page took the users to the Start page. This was changed to stay on the same page for users to switch to yes, if they selected no by mistake.

Integration Testing

With lack of time to write automated behaviour tests, blackbox end to end integration testing was performed testing validation of input data, communication to the backend API and the data layer. Observations and results from further usability testing are discussed in detail in the results section.

4.4 Conclusion

The methodology started off with user research including investigating data DVLA held with regards to user requirements, testing of the current site with eye tracker and interviews.

Following the MoSCoW principle, a domain model and architecture was designed. The PWA app was developed using Create-React-App, a ReactJS framework and the web server was implemented with Spring and Postgresql database. Caching was implemented with Workbox library and client-side storage, IndexedDB. Push notifications was achieved with Google Firebase as a middleware and the Java mail sender library for email notifications on version 2.0 of the app. The next section discusses the evaluation of the product and the data analysis and results.

5 Results and Discussion

This chapter presents a set of tests carried out to evaluate the implemented PWA and analysis of the results obtained. The tests conducted include black box testing to evaluate the functional requirements of the app and a set of non-functional tests to evaluate its performance, cross-platform support, and usability. The app will also be evaluated against the Google checklist for PWAs. In addition, a survey aimed at measuring the usability and user experience was also conducted.

5.1 Functional Testing

The proof of concept PWA has to meet with the following requirements:

- Service requirements - Access an API to retrieve vehicle tax, MOT status and tax rates within the scope of the app.
- Add to homescreen - Install the app on the homescreen on mobile devices and on the desktop on MacOS and Windows.
- App-like experience - Upon opening the app, it should work seamlessly online and offline. Additional services such as payment of tax and declaring SORN should redirect to GOVUK pages and return to the app on closing the external pages.
- Notifications - Users should be able to subscribe to push and email notifications.
Should receive push notification on Android devices and desktops

5.2 PWA Checklist

The foremost specification for a PWA is an app manifest that prescribes the App shell, followed by the service worker. Google's criteria for a PWA is listed in Appendix I. An automated audit with Lighthouse was performed to crosscheck compliance, which produced the overall scores for the five categories shown in figure16. Appendix I lists all extended criteria set that comprises the categories.

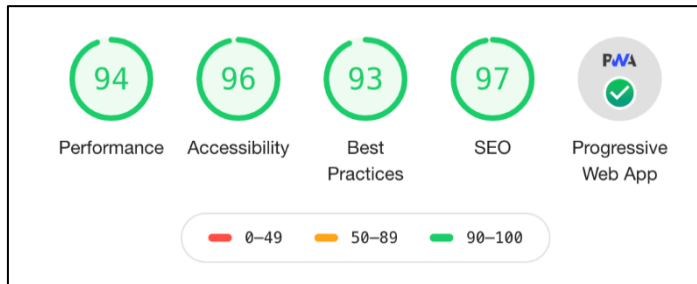


Figure 22: Lighthouse audit



Figure 23: The App shell on Android

The following sections only discusses the results of the tests carried out on performance and PWA features in detail. However, it should be noted that the other categories were also subjects of the implementation process.

5.2.1 Performance

The performance of a website is a metric of the time to load responses and how quickly the page is interactive. Slower responses would result in a higher bounce rate. The First Contentful Paint (FCP), First Meaningful Paint (FMP), and time to interactive are the metrics that would normally affect user retention. According to Google, a reading of less than 3 seconds is the benchmark. A time less than 10s time to interactive on slow 3G would satisfy a baseline PWA.

Figure 24 illustrates the Lighthouse scores on performance for the PWA. Additionally, 3 sets of tests run on webpagetest.org, emulating the same network conditions and device for the PWA and the current DVLA site resulted in the median scores as shown in Figure 25.

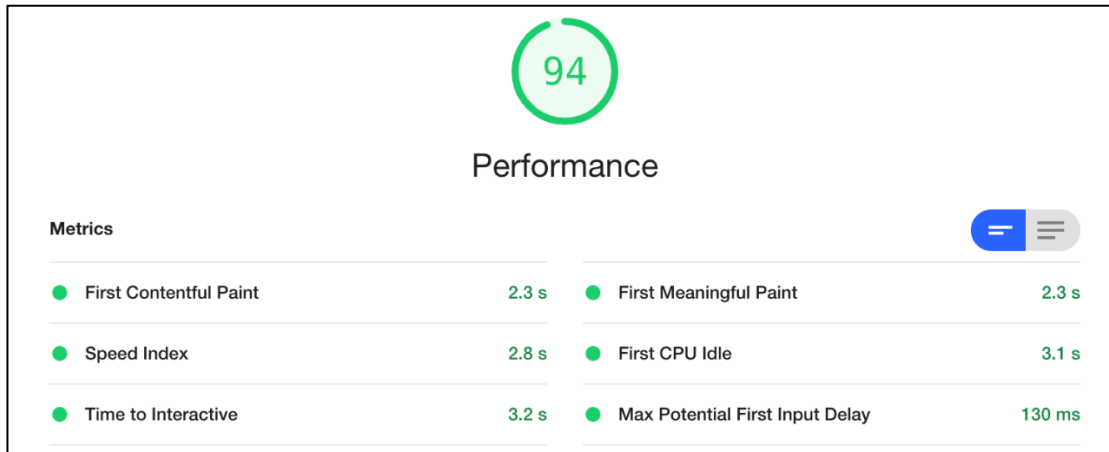


Figure 24: Lighthouse Performance scores for the PWA

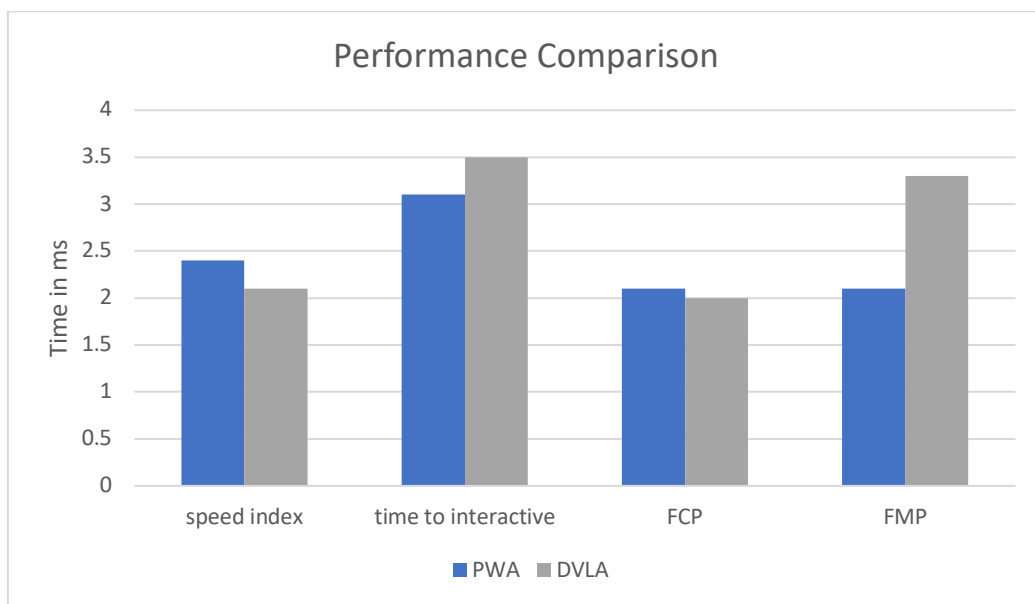


Figure 25: Webpageperformance.org audit comparison(initial readings)

The speed index is a metric of the loading time of the above-the-fold page content i.e. visible content within the viewport area. The smaller the page content, the lower and better the speed index will be. This is a calculated value of the first render and visually complete state. The first content might load quickly, but the page would complete only when it becomes fully interactive. Hence, speed index should be used in conjunction with the time to interactive as this is when users can actually interact with the page. Considering this, although the speed index of the PWA was slightly higher at 2.8 s as compared with the current DVLA website, the low 3.1s time to interactive suggests that the PWA was ready to accept user input earlier

than the DVLA and is also reflected in the first meaningful paint times. However, it should be noted that two compared sites do not have similar page content and the PWA requires background JS to be executed before rendering. As the components to be bundled increased, load time almost doubled. This was only on the first page as all other pages had same content as the current site. Efforts to code-split and lazy load did not help and requires more work. The measured metrics are also from the simulation of first visits and one must bear in mind that successive visits in a PWA follow the cache-first strategy and will be much quicker to load. In addition, the environment, language and framework used also differs, with the current site using Ruby on rails. Hence, it is not a like for like comparison and only indicative but not definitive.

5.3 Cross-Platform Capability

Develop once and work on all devices defines cross-platform capability. The objective is to ensure that the PWA functions across the multitude of devices and operating systems and hence was tested on Windows, Linux, MacOS, iOS on iPhone and iPad, Android on phone and tablet and Windows on XboxOne. The vehicle enquiries app was built with standard web technology frameworks html5, JavaScript, JSX and CSS. This did not require wrapping the code with a native container and rendering UI components in Webview that is characteristic of the hybrid approach. A web app built with web technologies and served through the browser depends heavily on browser support. The browsers used were Chrome, Safari, Opera, Firefox, and Edge. The earliest versions of browsers that support PWA include Chrome 40, Safari 11.1, Firefox 44 and Edge 17. However, the PWA was also installable on Windows and XboxOne on the new Edge with Chromium browser.

5.4 Usability Testing

Usability testing is about observing the actions of a user with a view to identifying ease of use. Following the action research methodology, users were given a list of tasks to perform and progress and errors were noted. MacOS, Windows 10 and Linux were tested on the same wi-fi network. Mobile phones (iOS and Android) were tested on wi-fi and 4G.

A sample of 12 participants were involved in testing. Testing was carried out individually with observation. The age range of the participants was between 22 and 60 approximately. One user had never used a smart phone other than to make calls but was familiar with installing games on desktops (non-mobile, app-distribution platforms). User testing involved performing a sequence of tasks. 2 participants used the app freehand with no prescribed task list. The usability tasks focused on users' experience of the installation of the PWA and the app-like experience of the app. In addition, a system usability survey was also given to participants.

5.4.1 PWA Installation Experience

Since the installation process for PWA differs from that of a native app, the aim of this test was to gauge how the users found the installation experience. The test required participants to complete a questionnaire before and after the users installed the app.

5.4.1.1 *Pre-Installation Test - User Survey*

A short questionnaire (Appendix III) was given before the users started the usability test to understand their exposure to mobile apps. The goal was to find out if the add to homescreen (a2hs) feature was familiar to them. Figures 25 and 26 show the responses to the questions:

1. Are you aware of the add to homescreen feature available in your device?
2. Do you uninstall and reinstall apps frequently?

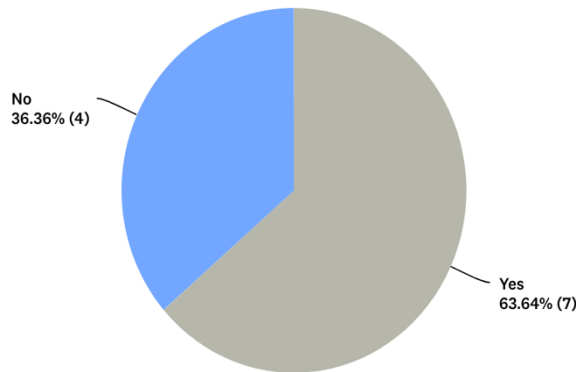


Figure 26: a2hs awareness

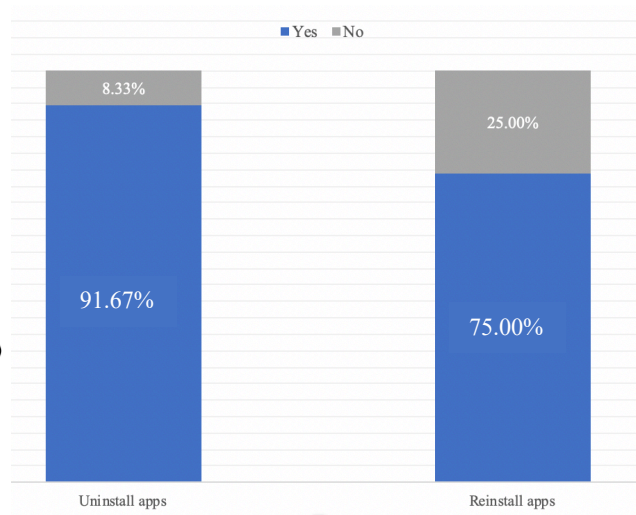


Figure 27: App installation behaviour

A little more than half the number of people who were aware of a2hs had not used the feature to bookmark an app. It was also evident that an equal number of persons uninstalled apps due to irregular use and insufficient storage. This stands to support the initial argument, government services that are infrequently used do not warrant the efforts of a native app. The post-installation questionnaire aims to find if user's installation preferences will be different after they have tried out the a2hs feature.

5.4.1.2 Installation Task Test

All users were asked to install the app. The installation banner was visible on both Android and iOS devices. Installation on Android was seamless with the app packed with WebAPK, where chrome packs the app with a native container on the fly. The app appeared on the app drawer with its own info. As Webkit, the browser engine for iOS, not yet supporting the beforeinstall prompt event, clicking the Add button displayed a modal with instructions to utilise the native add to homescreen feature. On desktops, users had to be instructed to use chrome or edge for installation as other browsers lacked support. Clicking the add button again displayed a modal

directing the user to click the + icon the address bar. Both scenarios are illustrated in figure 28 and 29.

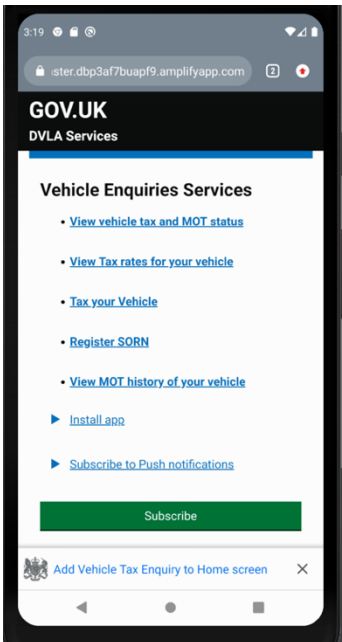


Figure 29: a2hs Android and iOS

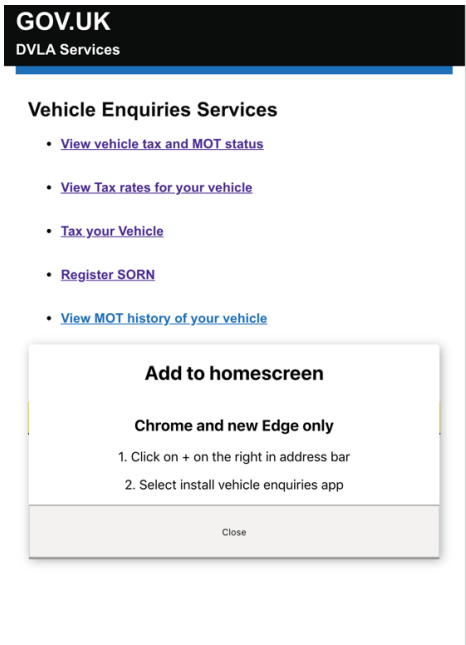


Figure 28: Chrome on Edge on desktop

5.4.1.3 Post-Installation Test Feedback

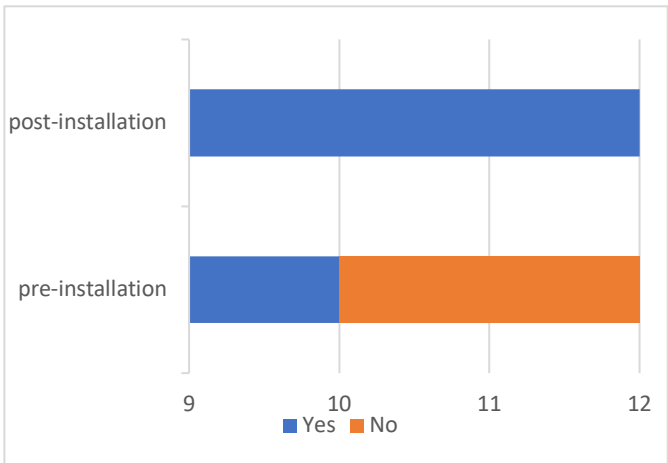


Figure 31: Number of people who prefer add to homescreen installation

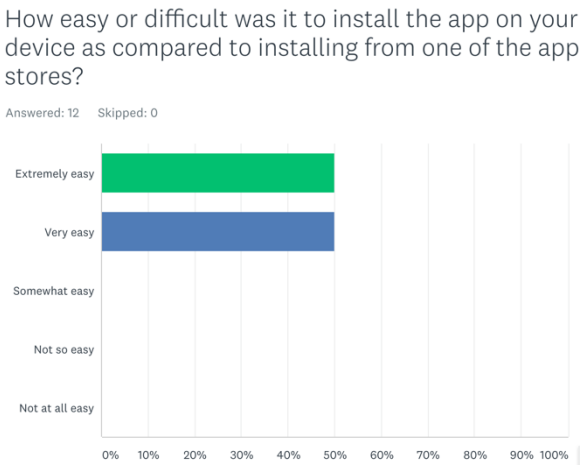


Figure 30: Ease of installation

All users were satisfied with the app installation process. Ease of installation was at 1:1 ratio on extremely easy and easy. Participants conducting the tests on mobile devices found it easier

than those conducting the test on desktops. On desktops, chrome browser produced a 100% success rate. Prior to installation, 17% had said that they have bookmarked sites and would not prefer this method. However, there is no information if those users had merely bookmarked a non PWA website.

The install app information link and the add button on the home screen were seen as good visual aids. All respondents unequivocally preferred the PWA installation to the installation of a native app. The convoluted rigmarole of discovering native apps was the reason cited. Users found the installation straight from the website as a reliable source. As cited in the literature review, trust is an important factor for adoption and the multitude of apps available in the app stores raised questions of validity and authenticity of apps. One participant emphasised that they often had to verify the source and PWA alleviated such concerns regarding authenticity.

5.5 App-Like Experience

The second usability test aimed to observe and evaluate the usability of the PWA. Participants were of two groups of 6 testing on desktop and mobile devices respectively as per table 4. Participants were asked to complete the tasks from the below mentioned task list.

Desktop		Mobile	
Windows	2	iOS	3
MacOS	3	Android	3
Linux	1		

Table 4: User testing groups

User Testing Task List

- Open the app from the launch icon.
- Navigate to the link to find their tax due date.
- Navigate to the link to find their tax rates.
- Navigate to an external link and return to the app.
- Use the swipe action to go back and forth the pages.

- Subscribe to push and email notifications.
- Click on notification when received.
- Check email for notification – including validating email

Email notification was tested with a test email address. No personal email addresses were collected. For testing purposes notifications were triggered by API calls rather than a scheduling periodic notification. The results of the usability tasks will be discussed first followed by observations of the splash screen, navigations, and offline experience.

5.5.1.1 Usability Task Results

The success rates and time taken by users to complete the tasks were recorded and shown in table 5 and 6.

Tasks	Success	Issues
Open the app from the launch icon.	100 %	None
Navigate to the link and find vehicle tax due date.	100 %	None
Navigate to the link and find vehicle tax rates.	100 %	None
Navigate to an external link and return to the app.	100 %	None
Use the swipe action to go back and forth the pages.	50 %	Did not work on desktops
Subscribe to push and email notifications.	100 %	No error message displayed on invalid data or format
Receive Push Notifications	50 %	iOS does not support
Receive email notification	100 %	One user did not receive due to user error and was fixed immediately

Table 5: User testing - Success rates and issues

All mobile device users were able to perform the swipe action successfully and all but iOS device users received push notifications successfully. However, clicking the notification did not open the app and this has been marked for future iterations.

User	1	2	3	4	5	6	7	8	9	10	11	12
Total time taken minutes	7.56	8.43	9.37	6.47	8.02	8.28	7.22	7.59	6.52	6.29	6.36	9.25

Table 6: Time taken to complete all tasks

The mean and median time taken to complete tasks was 7.6 minutes with mobile device users taking shorter time than desktop users with 2 standard deviations from the mean. It can be said that these results suggest that touch and tap gesture on mobile devices provide optimum experience than the touchpad or mouse.

5.5.1.2 Splash Screen

The splash icon was visible on android devices. With webkit for iOS still not providing full support for the web-app manifest and splash icon, a white screen was displayed on iOS but was insignificant as the first contentful paint was quick enough as indicated by the performance tests.

5.5.1.3 Navigation Experience

Navigation included forward, backward, back to home page, closing and reopening the app and switching between different apps (multitasking) akin to a native or hybrid app. Multitasking in versions iOS 12.1 loaded the app from scratch without maintaining the state of the app, thereby always opening at the home page. iOS13.0+ preserved the storage context and restored the current page. External URLs opened in the in-app browser and switched back to the preserved

context similar to native apps. However, in iOS13.3, a bug occasionally displayed ‘untitled’ instead of the app name. Webkit is working on fixing the bug.

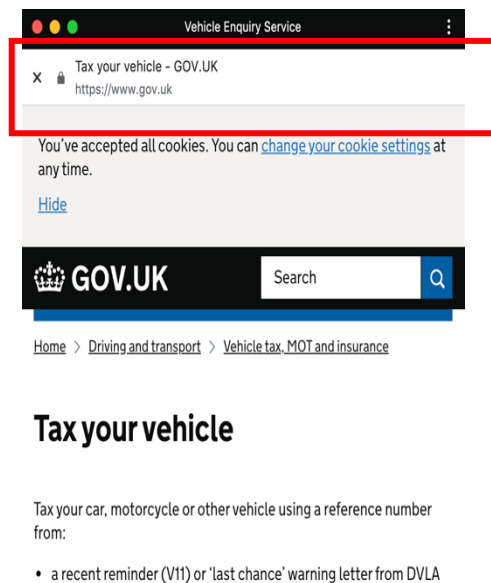


Figure 33: External page in-app browser MacOS

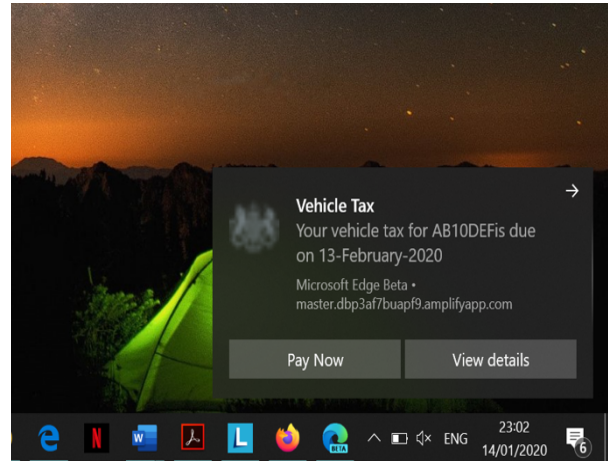


Figure 33: Push notification when app is closed Windows10

The swiping gesture was comparable to a native app in all devices. However, in a native app, the backwards navigation would take the user to the start of an action point. With the browser-based navigation of the standalone PWA, the user had to navigate through all previous pages before reaching the starting point of the current operation. The physical back button on Samsung devices also exhibited the same behaviour. For instance, as shown in figure 34 - 37 when a user visits the vehicle details page (screen 4), one would expect the back button in a native app to take them to start page or home page. Instead, the navigation took them to home page through screen 3 and screen 2 by way of successive backwards navigation. This was fixed by rerouting the navigation in the second iteration to take to the homepage. One user found when going back to the homepage by clicking new search displayed the bottom of screen 1 within the view rather than the top of the screen as a not so positive experience.

Figure 37: Screen 1

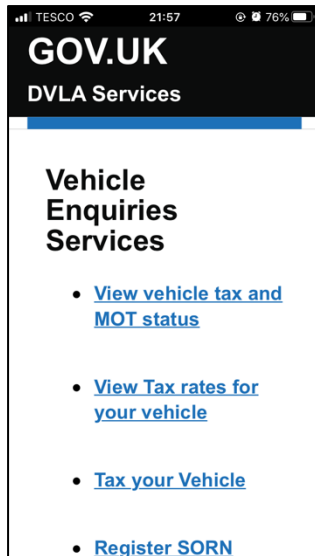


Figure 37: Screen 2

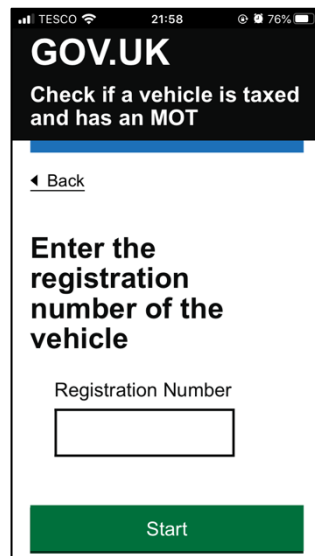


Figure 37: Screen 3

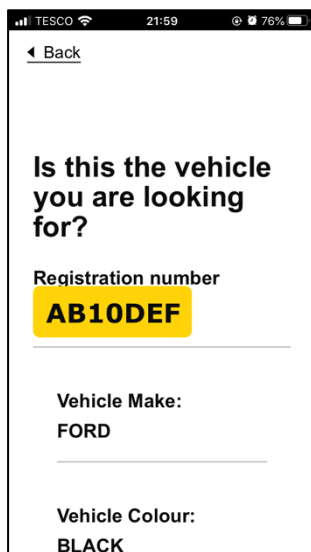
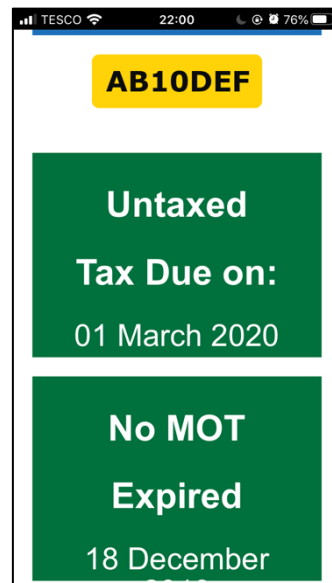


Figure 37: Screen 4



5.5.1.4 Offline Experience

The pre-test questionnaire given to users recorded that of the 20% of respondents who had bookmarked a site, none of them said the site worked offline. To test the offline capability and measure user feedback, participants were asked to turn off wi-fi and mobile data and find their vehicle details.

Figure 38: Offline page all platforms

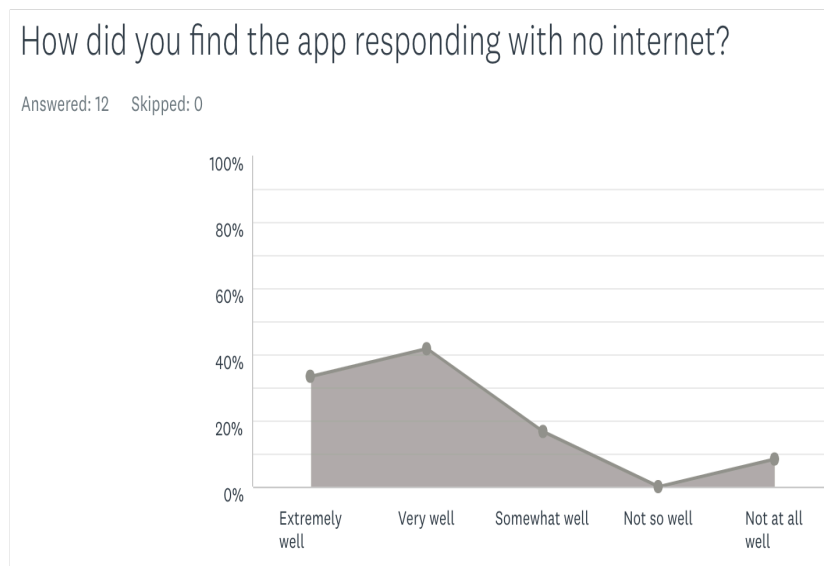
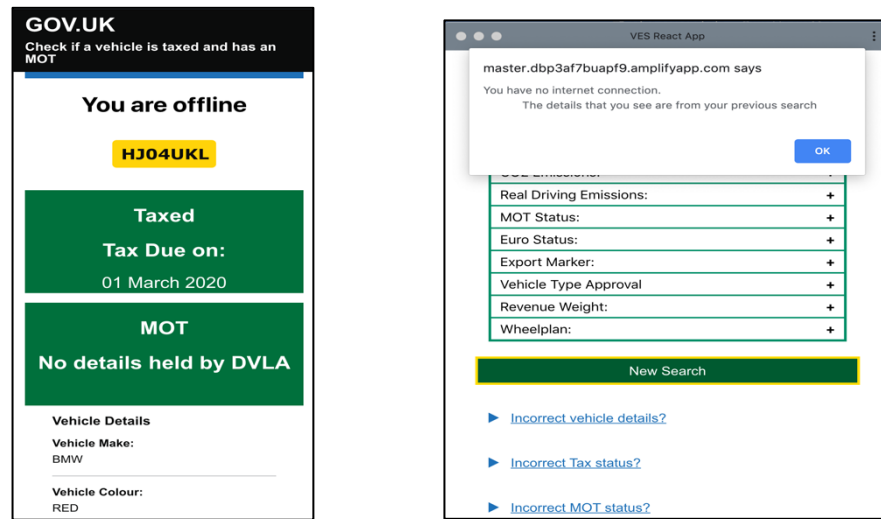


Figure 39: Offline experience satisfaction

Figure 39 shows that offline experience was overall satisfactory with 75% rating between extremely well and very well. Workbox scripts used for caching pages was not supported in

iOS version 11.2.5 and 12.1 resulting in a network error page which prompted one user to respond negatively on offline capability. Two inconsistencies were noted on offline capability.

1. Users had to type in the registration number and go through the confirm page even when offline.
2. The second inconsistency was that when they went offline in the vehicle details page, which displayed dynamic data, when the app was closed and reopened to the cached page, a backward navigation, followed by forward navigation resulted in error in dynamic data.

Issue 1 was a redundant operation from the user's perspective as new vehicle search would not be possible anyway. This was mitigated in the next version by navigating straight to the cached vehicle details page from the homepage. Issue 2 was a consequence of the way that the cached page and the indexeddb communicated when offline. Earlier testing did not replicate this error as older iOS versions always opened the app from the root URL. However, in later versions of iOS state preservation took them to the current page and hence the error. This is reflected in the post-test survey, where 30% responded with somewhat well for the offline experience. The issue was programmatically fixed in the second iteration.

5.5.1.5 Push Notification

Android and desktop users received push notifications and for the question in figure 40, one in three users were extremely satisfied. The onclick action on the notification did not open the app or redirect to the payment URL and this was a disappointing experience. All users were also asked the following questions. iOS users were not asked to rate the service as they were unable to receive any due to lack of feature support.

1. Do you think the push notification feature is useful to you?
2. The notification content was precise and clear.

All users felt it was a very useful feature and found the message title and content to be precise and clear.

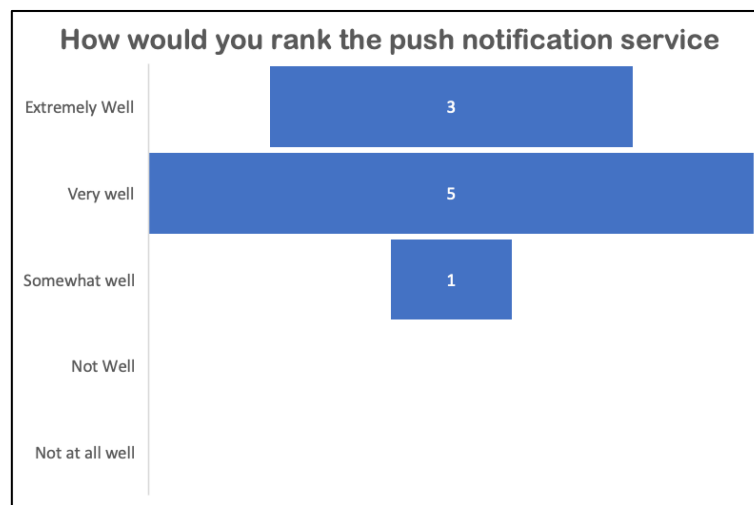


Figure 40: Push Notification satisfaction

5.5.1.6 Summary

User feedback were suggestive from figure 27 and direct quotes from users, that they were overall satisfied with the experience.

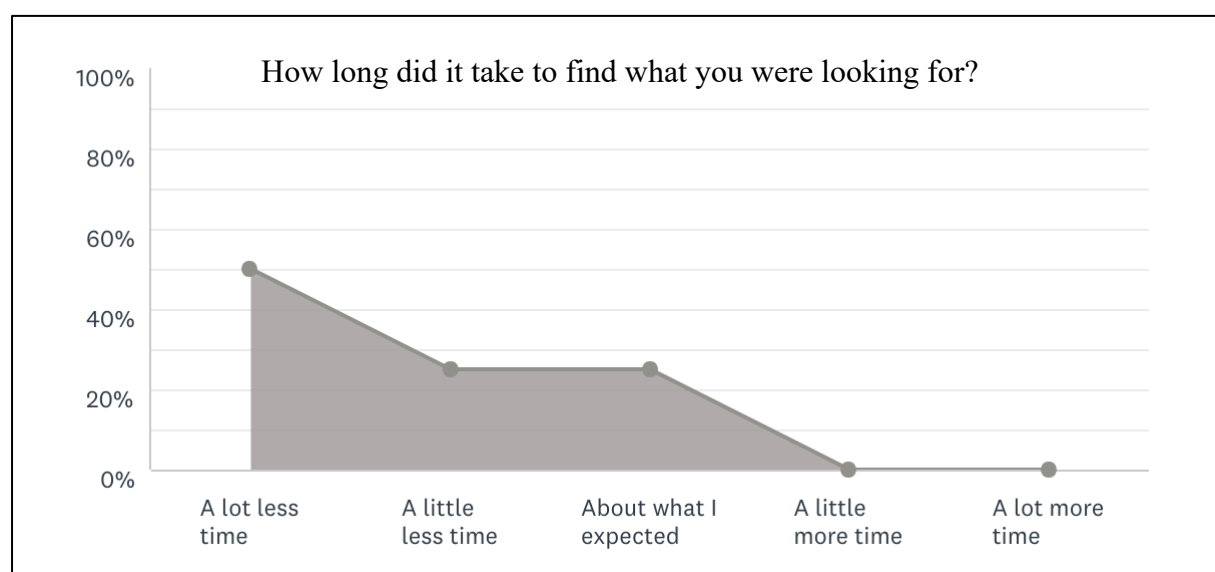


Figure 41: Perception of time taken

- **Direct quotes**

“I was surprised with how easy the application is to install and how quickly the information which you are looking for is provided. The only suggestion would be minor styling changes with the way the accordion works on the vehicle details page.”

“ I find the push notification very helpful, as it would help me remember to pay my tax even if I was travelling”

The general themes were easy and quick to install, authenticity and notifications. There were a few comments regarding the styling and navigation.

5.5.2 System Usability Survey (SUS)

The SUS is a tried and tested questionnaire set used for nearly 30 years and has a proven track record for quantitative analysis on the usability of a system. The likert scale system assigns a numeric score as a representation of 100 and a mean score of 68 or more is considered as acceptable industry standard usability. 11 participants completed the SUS questionnaire after exploring the PWA by completing tasks from the Installation Task Test. Table 7 shows the SUS score for each of the users

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10	User 11
SUS score	72.5	90	90	90	90	80	100	72.5	77.5	100	77.5

Table 7: SUS ranking after user testing

With a score of 80.3 regarded as grade A, the overall mean test value of 85.46 indicates that participants were overall very satisfied with the experience. It is by no way diagnostic and is

an indirect measure of the experience. We can safely say that participants had a satisfying experience with the steps that involved installing the PWA, viewing content online and offline and features such as push notification from both desktop and mobile. However, it should be noted that the scores were in the range of 72 – 78 in the case of desktop users as opposed to higher scores with mobile. Breaking down the SUS into two themes, namely ease of use and complexity, yielded the results as shown in figure 42 and 43.

60% of the participants found it extremely easy to use and only 1 in 10 persons felt they would not use it frequently.

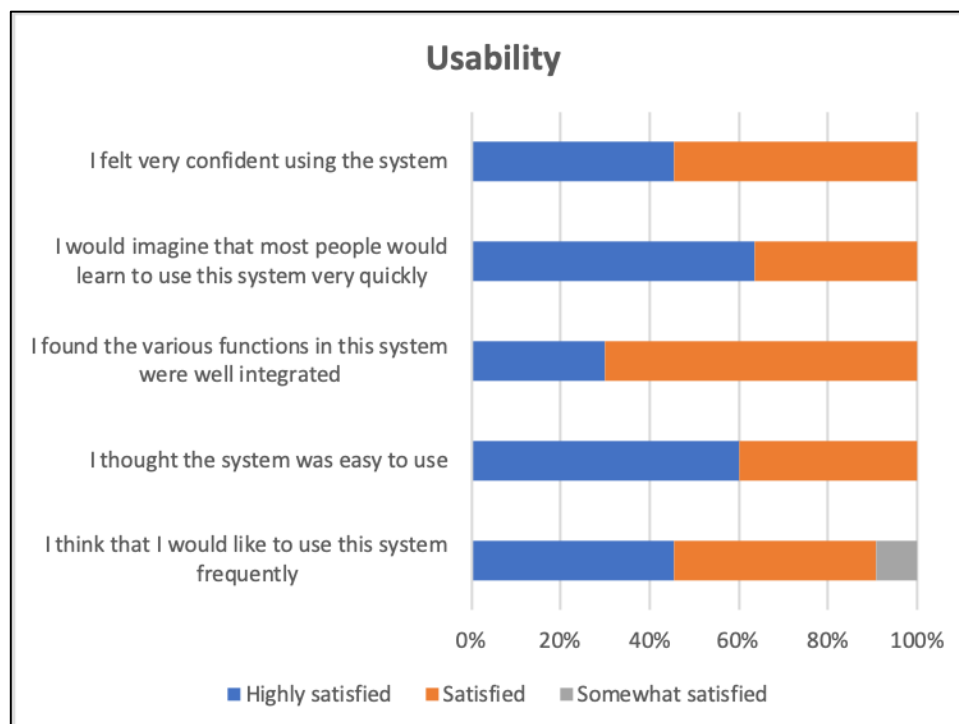


Figure 42: Positive questions on usability

1 in 10 persons felt that it was difficult to learn how to use the app. The pre installation test indicated that 1 person was a stranger to apps and this SUS result appears to suggest that it could be lack of knowledge that led to this high disappointment score.

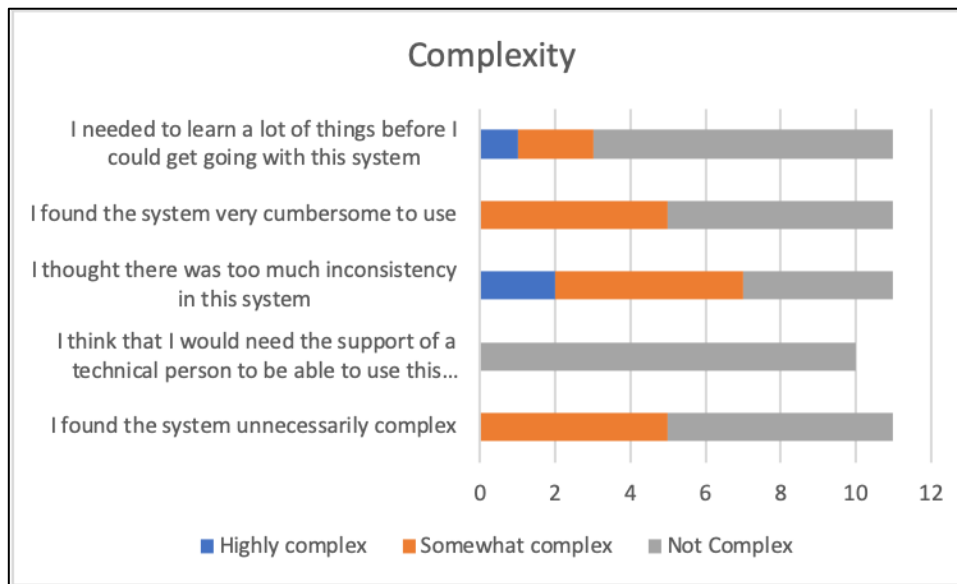


Figure 43: Negative questions on complexity

5.6 Affinity to Government digital services

Additional investigations and analysis were carried out to discover user preferences regarding Government apps via:

- Survey conducted after the usability testing of the PWA
- Data from previous surveys conducted by DVLA
- Usage data from Digital licencing service (DLS) and HMRC app

The results from this investigation lead us to believe that there is an appetite for government mobile apps.

5.6.1 Post-Usability Test

Participants were asked 3 questions as part of the post usability test, regarding the mobile apps in government. All respondents had used some form of government online services with only 16.67% having installed a mobile app. The preference for the provision of a mobile app could be considered unanimous with only one person against it.

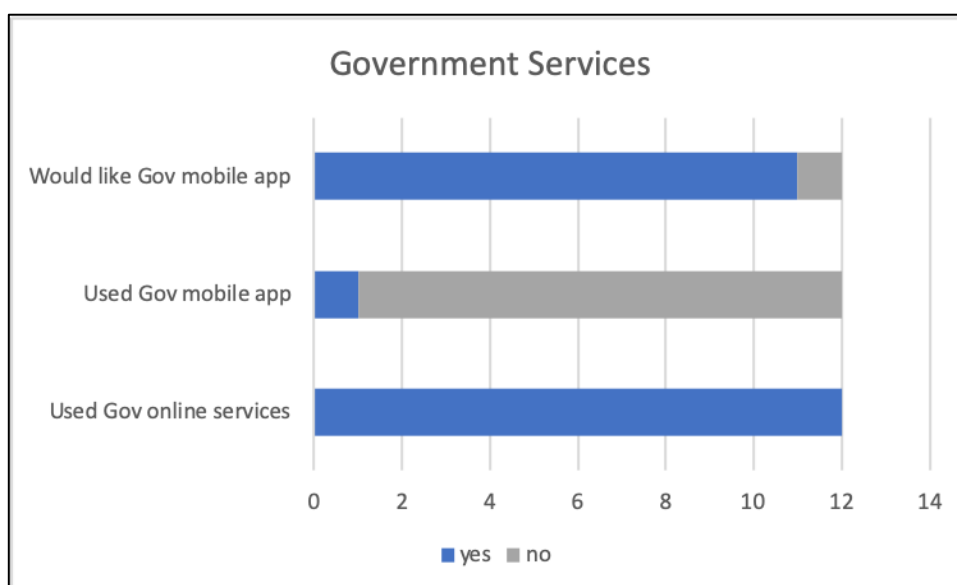


Figure 44: Preferences for Government services

5.6.2 DVLA User Insights Survey

A number of data sets [95] from mobile initiatives used by the UK Government, point to a positive trend towards experience and satisfaction. In a survey conducted by the DVLA as part of user insight, 57.6% of respondents preferred a digital licence of which 84.5% preferred to view it on a smartphone.

5.6.3 Usage Data on existing DVLA and HMRC mobile apps

Interviews with DVLA and HMRC product owners yielded some usage data. Between the period February to July 2019, 2,042 android and 1,476 iOS users had used the DLS service successfully with a 29.60% conversion rate. HMRC app's usage data is sitting at 40% of all sessions in the personal tax space, with users using the app 8-9 times a month whilst the web is being accessed once or twice in a month. Around 400,000 people are using the app every month with a much higher satisfaction than average apps at 4.5 stars.

Interviews with service designers revealed that the rational for not developing an app for the service is based on the guidelines issued by GDS. However, feedback from users indicate

(figure 45) that users were expecting the DLS to be an app and suggest that they were not quite satisfied with the experience. The DLS allows for the licence code to be added to the wallet or download a code. Comments from users who tried the DLS reiterates that users expected an app and the DLS fell short of being an app. Some comments included phrases such as

"I thought it was an app", "Cannot install on iPhone8".

Words like not able to install, phone app, save to phone etc were frequently seen. It has also proved to be a very lengthy and cumbersome process to add the licence to the wallet and many could not follow the instructions.



Figure 45: User research – DVLA data on request for mobile app

Further user research data results also suggest that there is an appetite for mobile apps as comments collated in Appendix III.

5.7 Conclusion

The key features that differentiate the PWA from the responsive web and native apps include:

- **Developer's perspective:** Cheaper, single code development, cross-platform functionality.
- **User experience:** Ease of installation, provide a native app like experience, faster loading performance even on weaker network conditions and offline caching after adding to the homescreen.

It could be said that users would more likely learn to use features that are only relevant to them and app stores are familiar due to their robust marketing by sites and the stores whilst, apps developed as PWAs are yet to enjoy the same level of publicity as those published in the app stores. However, it was surprising to discover that a higher proportion of users were aware of the add to homescreen feature. All participants preferred the PWA installation to native or hybrid app installation. Nearly half of them found it extremely easy and half of them said it was very easy to find the information that they were looking for. Device detection ensured only relevant user interface was rendered to users. Only 1 in 12 found the offline experience unsatisfactory. All non-iOS users who subscribed for push notifications received notifications. As for the requirements of a PWA stipulated by Google, all criteria were satisfied with first contentful paint under 3s and all user requirements according to the MoSCoW scale were implemented with grade A on the system usability scale. Primary data from survey and secondary data suggest that there is an appetite for mobile apps in the government and with user preference leaning towards PWAs in the usability test and surveys conducted, PWA appears to be a feasible solution.

6 Conclusion

6.1 Research Based Findings

Previous research on PWAs, have considered them as promising and cost-effective alternatives to native apps. Both academia and industry give testimonials on their increased performance, lower device storage, offline caching, and background synchronisation. Although one research suggests that more robust testing for vulnerabilities is required, PWAs are generally considered secure with its TLP and browser level data storage. Several research conclude it as the most efficient cross platform approach due to its web technology-based development with HTML, CSS and JS. This is also evident from the fact that some of the big players such as Twitter, Instagram and Facebook adopting and advocating PWAs. The UK government is hesitant on mobile apps at national level but local governments have a number of them. Existing MGov apps are popular and there is a demand for more.

6.2 Software Development Case Study Based Findings

The aim of the project to investigate if PWA is an option for mobile app provision in governments was met by building the proof of concept PWA for the DVLA's vehicle enquiry service. Usability testing has confirmed that PWA is installable in Android, iOS, iPadOS, Windows 10, Xbox One, macOS, and Linux 64 bits. Firefox for android and chrome on all operating systems installation worked. With a 100% satisfaction on installation experience and 83.33% on content loading and offline caching we could say that although not perfect scores, there is still a good scope for implementing a PWA. With user study from Google suggesting that 53% of users abandon a site if loading is not under 3 seconds and most sites on 3G take around 19 seconds, the readings from this PWA were promising with just 2.1 seconds and 9.6 on slow 3G on MVP. However, as efforts were made to improve the UX, the JS bundle size increased the load times. One must not forget that PWAs gain power progressively. It is evident

that users are interested in the offline capabilities' that a mobile app can provide and can be perceived that most users are only aware of mobile apps as the medium for offline usage.

Major challenges for mobile adoption are mobile device management, app store management, identity management and data security. The browser-based technology of PWA eliminates cost and concerns regarding the above. Unlike a native app, whitelisting, and app wrapping, routing through app stores were not required. Data privacy concerns are alleviated as information is stored at the browser level. As for app-like experience, with Webkit providing support for safari, testing confirmed that swiping gestures for back and forward and seamless transition to external links and back were possible. Users were able to trust the app as it was installed directly from the website thereby establishing authenticity. One major positive is the push notifications which reminds users with tax renewal dates. With 1100 letters being undelivered only in April 2019 and 67% of surveyed respondents requesting digital notifications, this stands to be advantageous to DVLA. Further feedback from DVLA user research suggest that citizens prefer apps. PWAs are cost effective, cross-platform mobile apps.

6.3 Limitations

It has not been possible to ascertain the cost factor to compare with existing responsive site or mobile apps as no data was made available due to sensitivity. However, research suggests that organisations have implemented PWA for a fraction of the cost of developing for multiple platforms. However, one caveat is that it is heavily reliant on JavaScript rendering. In a survey conducted by Webaim 98.93% had JS enabled. 0.7 disabled [96]. None of the participants in the testing had their JS disabled by default. There may be other scenarios where JS might fail as well. GDS argues that no user should be excluded due to JS failures. However, they are revisiting the topic of PWA. With only 0.2% of the population having disabled JS in 2016 and no recent data available on this, it is the prerogative of government departments to decide on

policy level taking into consideration the evolving technology eco system and business landscape to strike a balance to meet user expectations.

6.4 Recommended Future Work

Future work would involve some work with the user interface, addressing issues relating to older browsers, improving performance by offloading uncritical JS to webworkers or server-side rendering. Redux allows for storing app states and can be explored in lieu of IndexedDB. Implementing authentication of users either through OAuth or Gov.verify, the government's user identification service could also be investigated. The results from this work suggest that the benefits outweigh the JS issue, it is recommended that public services should invest in PWA and perhaps look into other alternatives to ReactJS including combining PWA and WebAssembly which allows to write with *c#* or progressive enhancement of the existing responsive site. DVLA is also looking into developing enterprise apps for staff and with Apple and Google store introducing new restrictions on apps with their notarisation mandate and framework restrictions, PWA could be considered an alternative.

6.5 Summary

With Progressive web apps being the technology trend in industry, it was aimed to investigate if they are suitable for government services as a cross-platform approach. Literature review, a proof of concept PWA for the DVLA as a case study, Usability tests and survey formed the study. Primary and secondary data from the DVLA and HMRC were also analysed. The PWA was implemented with ReactJS, Spring backend and RDBMS with test data. It has been identified that the citizens would like government to deliver mobile apps and they prefer the PWA over native and hybrid apps installable from one of the app distributions stores. Ease of installation, trust and push notifications were the main reasons. It is also evident that the single

code development keeps resources at minimum and the PWA functions on a range of cross-platform devices with similar robustness. The government's agenda on accessibility will be met with PWA which guarantees best experience with the fully downloaded app with super-fast internet connection and most modern browser whilst even shaky connection and initial visits would also give a quality experience.

6.6 Reflection

The topic for the project was suggested as an option by my lecturer and with no prior knowledge on the topic of progressive web apps it was quite intriguing. What came as a surprise, is the realisation that it was such a trending topic in the field of apps. With not much prior experience on front end development, it was intimidating and daunting to decide where to start from. Extensive reading on PWA literature was a good starting point. It made me aware of all the research that has been going on and has motivated me to keep me updated with the latest trends in research and development in the realm of technology.

The project being timeboxed and limited knowledge on JavaScript, ReactJS appeared to be a best fit with its not so steep learning curve. With full stack development being the norm at the DVLA, I took to working on stories at work which involved frontend UI. I was not hesitant to speak to service designers, UX designers and other fellow developers to gather all the information that I deemed pertinent. I had some experience on Spring and Java and was quite confident with the backend. However, tackling the cross-origin issue was challenging. Progress being made quite smoothly, I hit my first hurdle with push notifications. What was meant to be a simple process proved to be agonising with the server to frontend communication failing. In

an attempt to solve it, I developed a backend with nodeJS nodeJS, but no success. It was an exhilarating experience when finally, I was successful with firebase.

It has been an exciting journey, learning several new technologies and tools such as ReactJS, Workbox, IndexedDB, service workers and above all the nuances and finer details of mobile app interfaces. I found it awkward to approach people for user testing and questionnaires as I was perturbed that they might find it a time-consuming affair. It was comforting to find that volunteers were accommodating and genuinely liked my product.

I wish the notification didn't take so long as I would have had time to bring some finesse into the UI and the code and some automated tests. My initial attempts to use bitbucket were also rusty and time and effort was wasted on fixing issues, which were solved with increased practice. It has definitely been invigorating to have produced an end-to-end service that mimics a live service. However, I realise that this is just the beginning and there is always a lot more to learn in this rapidly evolving field.

Appendix I

Vehicle response data set - 3 authentication levels

	<i>Web and Basic API</i>	<i>Standard</i>	<i>Enhanced</i>
<i>VRN (vehicle registration number)</i>	X	X	X
<i>Tax status</i>	X	X	X
<i>Tax due date</i>	X	X	X
<i>ART end date</i>	X	X	X
<i>MOT status</i>	X	X	X
<i>MOT expiry date</i>	X	X	X
<i>Make</i>	X	X	X
<i>Month of first DVLA registration</i>	X	X	X
<i>Month of first registration</i>	X	X	X
<i>Year of manufacture</i>	X	X	X
<i>Engine capacity</i>	X	X	X
<i>CO2 emissions</i>	X	X	X
<i>Fuel type</i>	X	X	X
<i>Marked for export</i>	X	X	X
<i>Colour</i>	X	X	X
<i>Type approval</i>	X	X	X
<i>Wheel plan</i>	X	X	X
<i>Revenue Weight</i>	X	X	X
<i>Eurostatus</i>	X	X	X
<i>RDE (Real Driving Emissions)</i>	X	X	X
<i>V5C issue date</i>	X	X	X
<i>Tax rates only to keeper</i>	X		
<i>Tax code only to keeper</i>			
<i>Previous colour</i>		X	X
<i>Engine number</i>		X	X
<i>Mileage reading</i>		X	X
<i>Change of keeper date</i>			X
<i>Date of SORN</i>			X

Google Checklist for a Base line PWA

Criteria	Implementation
Site is served over HTTPS	Implement HTTPS
Pages are responsive on tablets and mobile devices	Implement a responsive design
All app URLs load while offline	Use a Service Worker
Metadata provided for Add to Homescreen	Add a Web App Manifest file
First load fast even on 3G	time to interactive <10s for first visit
Site works cross-browser	
Page transitions don't feel like they block on the network	
Each page has a URL	

Google Lighthouse Audit with PWA criteria passed

Progressive Web App

These checks validate the aspects of a Progressive Web App. [Learn more.](#)



Fast and reliable

- Page load is fast enough on mobile networks ✓
- Current page responds with a 200 when offline ✓
- start_url responds with a 200 when offline ✓



Installable

- Uses HTTPS ✓
- Registers a service worker that controls page and start_url ✓
- Web app manifest meets the installability requirements ✓



PWA Optimized

- Redirects HTTP traffic to HTTPS ✓
- Configured for a custom splash screen ✓
- Sets a theme color for the address bar. ✓
- Content is sized correctly for the viewport ✓
- Has a <meta name="viewport"> tag with width or initial-scale ✓
- Contains some content when JavaScript is not available ✓
- Provides a valid apple-touch-icon ✓

Appendix II

User Stories – Kanban Board

- A sample of major stories

ID	VE1	Version number	V1
Date		Author	
Title	Get Vehicle Tax and MOT Details		
As a...	user		
I want...	View my vehicle details		
So that...	I can know tax due date and MOT due date		
Dependencies	Vehicle details should already exist in database		
Acceptance Criteria:			
<u>Scenario 1</u>			
Given: There exists a vehicle			
When: a user navigates to the search page and types number plate and confirms			
Then: tax due date and MOT due date will be displayed with all other vehicle details			
And: the user can perform a new search			
<u>Scenario 2</u>			
Given: Vehicle does not exist			
When: a user types in number plate			
Then: No vehicle details found will be displayed			
And: a user can perform a new search			

ID	VE2	Version number	V1
Date		Author	
Title	Get Vehicle Tax rates		
As a...	user		
I want...	View my vehicle details		
So that...	I can know tax amount		
Dependencies	Vehicle details should already exist in database		
Acceptance Criteria:			
<u>Scenario 1</u>			
Given: There exists a vehicle			
When: a user navigates to the search page and types number plate and V5C number, and confirms			
Then: monthly, 6 monthly and yearly tax amount will be displayed with all other vehicle details			
And: the user can perform a new search			
<u>Scenario 2</u>			
Given: Vehicle does not exist			
When: a user types in number plate			
Then: No vehicle details found will be displayed			
And: a user can perform a new search			

ID	VE3	Version number	V1
Date		Author	
Title	Install app		
As a...	user		
I want...	Install the app in desktop or mobile device		
So that...	I can know vehicle details without navigating in browser		
Dependencies	Vehicle details should already exist in database		
Acceptance Criteria:			
Constraints: Use Chrome or Edge browser on desktop			
<u>Scenario 1</u>			
Given: A user navigates to the website on chrome desktop or mobile, firefox for android			
When: a user clicks on install			
Then: the app should be added to homescreen			
And: the user can open the app from desktop or homescreen by tap action			
<u>Scenario 2</u>			
Given: A user navigates to the website on safari iOS			
When: a user clicks on install			
Then: the instructions to manually add are displayed			
And: a user can manually add to homescreen			
And: the user can open the app from desktop or homescreen by tap action			

ID	VE4	Version number	V1
Date		Author	
Title	Receive Push Notification		
As a...	user		
I want...	Receive notification		
So that...	I can be reminded of vehicle tax due date		
Dependencies	VE5 – Send Push Notification, VE6 Subscribe to Push Notification		
Acceptance Criteria:			
Constraints: Android or desktop			
Scenario 1			
Given: A user has subscribed to push notification			
When: the tax due date is one month away			
And: the notification has been sent from the server			
Then: the notification should be displayed.			

ID	VE5	Version number	V1
Date		Author	
Title	Send Push Notification		
As a...	service		
I want...	To send push notification		
So that...	I can remind users of vehicle tax due date		
Dependencies	VE6 Subscribe to Push Notification		
Acceptance Criteria:			
Constraints:			
<u>Scenario 1</u> - Scheduler scans records every night			
Given: The service is running			
And: The vehicle records are scanned for tax due dates 1 month away			
When: there are subscriptions for the vehicles due			
Then: send a push notification to those users			
<u>Scenario 2</u> – Testing for demo purposes			
Given: The notification service is being tested			
When: The request to send notification is received			
And: The vehicle records are scanned for tax due dates 1 month away			
When: there are subscriptions for the vehicles due			
Then: send a push notification to those users			

Appendix III

DVLA user research collated comments for a DVLA app (source: DVLA)

In Favour

- If DVLA had an app that could be used for notifications.
- App is the way forward
- I would prefer an app, it's easier
- DVLA Account, prefer an app
- The app can tell/update you on any DVLA services.
- Dvla app? Web chat would be really helpful. Likes notifications “that would be really good”
- Check your ID, especially for those who drive for a living. They would not need to carry cards around.
- They would want to generate a security code to get into the App, especially if they were going to change any details.
- Log in – touch id with an app is good, someone could guess your password
- Have an app similar to Boots that is really clever, tells you all about new offers etc. so something like that for DVLA would be really great as it could update them on any new online services etc.
- “Being dyslexic, having a way where things can be done automatically is easier for me. I don't have to put my details in all the time then.”

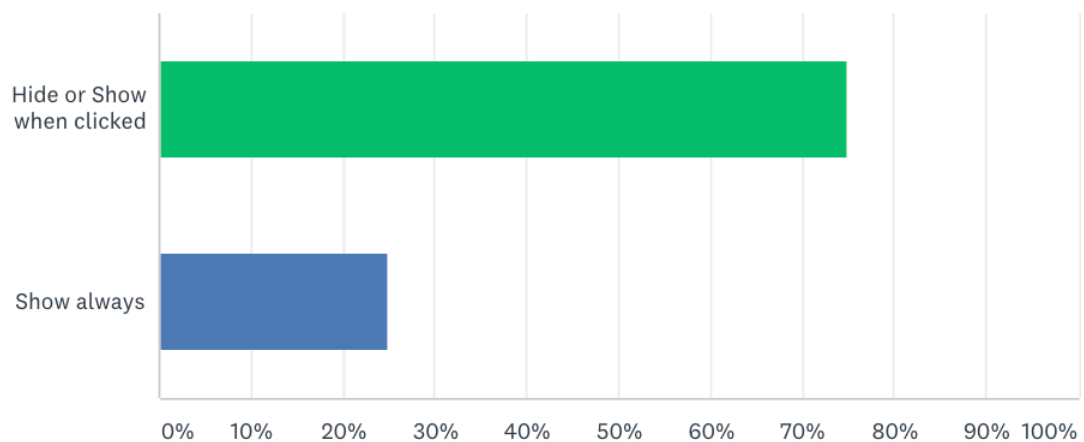
Reluctance

- Apps can take up too much storage.

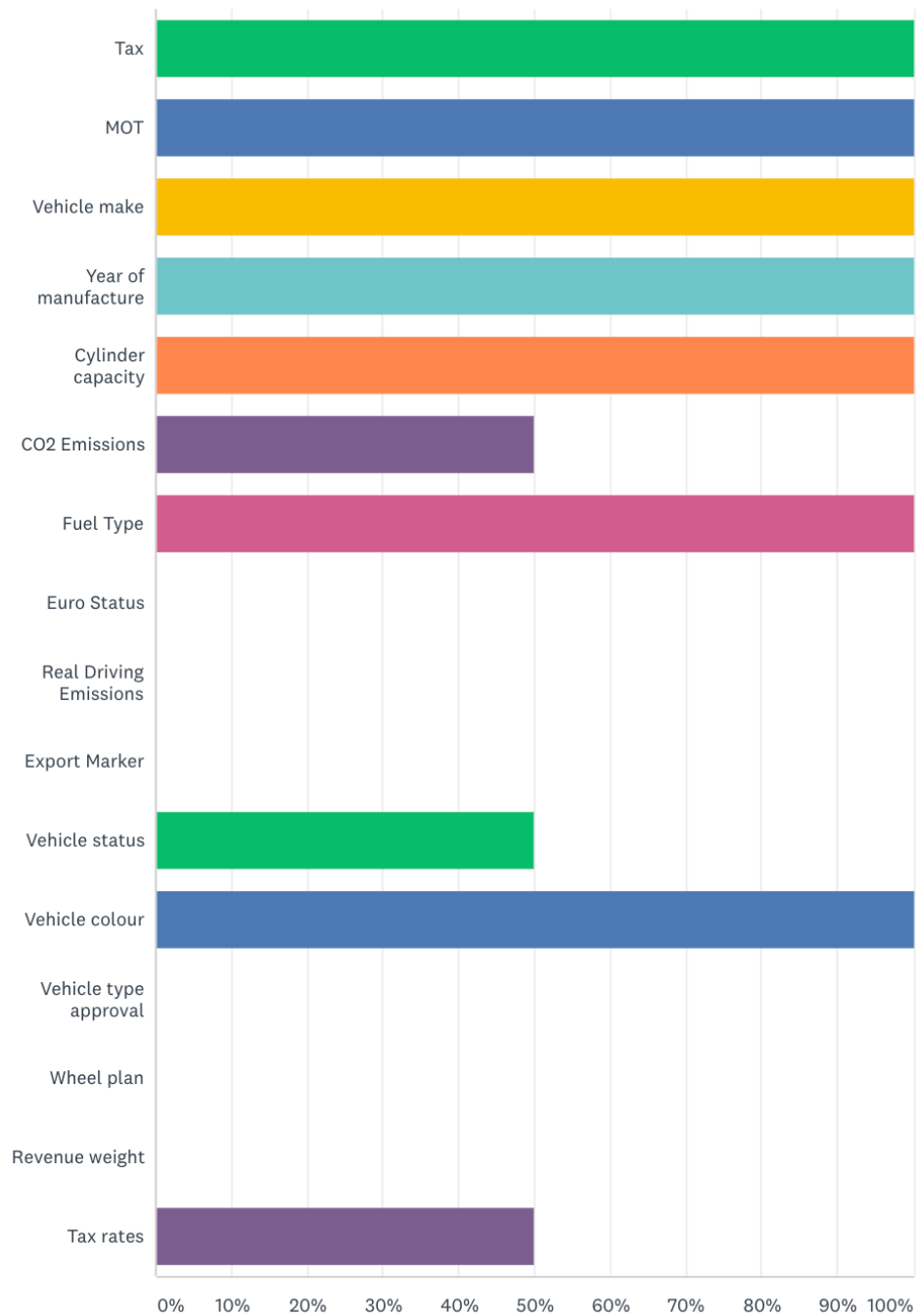
- Screen can feel cluttered if adding more apps.
- Reluctant to add any more apps to their phones as felt they were already cluttered and apps take too much storage space.
- Probably the younger generation would like an App.

User research – Requirements elicitation

Would you like all the information to be visible straight away or visible only when clicked on it?



What information about the vehicle would you like to see on the first screen after you submit the query? All other details can be displayed on the following screen or hidden and shown when clicked.



Pre-Installation Survey

1. Do you know about the Add to Homescreen feature in your phone or tablet?
 - ☐ Yes
 - ☐ No
2. If yes, have you used to bookmark a website using the feature?
 - ☐ Yes
 - ☐ No
3. If yes, does the website you bookmarked work when there is no internet connection?
 - ☐ Yes
 - ☐ No
4. Where do you usually install your apps from?
 - ☐ Apple App store
 - ☐ Google Play store
 - ☐ Windows store
 - ☐ Other
5. Have you ever installed an app without having to go to any of the above apps marketplace
 - ☐ Yes
 - ☐ No
6. If you were provided with an option to install an app by just bookmarking it, would you prefer that installation method?
 - ☐ Yes
 - ☐ No
7. Do you think apps take up a lot of your device storage?
 - ☐ Yes

- ☐ No
- 8. Do you uninstall apps more often?
 - ☐ Yes
 - ☐ No
- 9. What is the reason for uninstalling?
 - ☐ Not used regularly
 - ☐ Need of storage space in device
 - ☐ No longer require the app
 - ☐ Other
- 10. Do you uninstall apps that you don't use frequently and then reinstall when required?
 - ☐ Yes
 - ☐ No

Post Usability Test Survey

1. How easy was it to find what you were looking for on our app?
 - ☐ Extremely easy
 - ☐ Very easy
 - ☐ Somewhat easy
 - ☐ Not so easy
 - ☐ Not at all easy
2. Did it take you more or less time than you expected to find what you were looking for on our website?
 - ☐ A lot less time
 - ☐ A little less time
 - ☐ About what I expected
 - ☐ A little more time
 - ☐ A lot more time
3. How visually appealing is our app and website?
 - ☐ Extremely appealing

- ☐ Very appealing
 - ☐ Somewhat appealing
 - ☐ Not so appealing
 - ☐ Not at all appealing
- 4. How easy or difficult was it to install the app on your device as compared to installing from one of the app stores?
 - ☐ Extremely easy
 - ☐ Very easy
 - ☐ Somewhat easy
 - ☐ Not so easy
 - ☐ Not at all easy
- 5. Would you prefer this bookmarking method to install apps to installing from the app stores?
 - ☐ Yes
 - ☐ No
- 6. How did you find the app responding with no internet?
 - ☐ Extremely well
 - ☐ Very well
 - ☐ Somewhat well
 - ☐ Not so well
 - ☐ Not at all well
- 7. Do you have any other comments?
- 8. Have you used any of the Government online services?
 - ☐ Yes
 - ☐ No
- 9. Have you installed any Government services mobile apps?
 - ☐ Yes
 - ☐ No
- 10. Would you like Government services to provide mobile apps?
 - ☐ Yes
 - ☐ No

System Usability Survey

Strongly disagree					Strongly agree
1	2	3	4	5	

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

References

- [1] V. Persson and J. Nouri, “A systematic review of second language learning with mobile technologies,” *Int. J. Emerg. Technol. Learn.*, vol. 13, no. 2, pp. 188–210, 2018.
- [2] B. Fling, *Mobile Design and Development*. O’Reilly Media.
- [3] K. T. and L. Silver, “Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally,” *Pew Research center*, 2019. [Online]. Available: <https://www.pewresearch.org/global/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/>.
- [4] J. Clement, “Percentage of all global web pages served to mobile phones from 2009 to 2018,” *Statista.com*, 2019. [Online]. Available: <https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/>.
- [5] GDS, “Making sure your service works well on mobile,” *Government Digital Service GovUK*, 2019. [Online]. Available: <https://www.gov.uk/service-manual/technology/working-with-mobile-technology>.
- [6] Cabinet Office, “Government Transformation Strategy,” p. 93, 2017.
- [7] S. Howes and T. Kidney Bishop, “The hidden obstacles to government digital transformation,” pp. 1–109, 2018.
- [8] G. Freeguard *et al.*, “Whitehall Monitor 2019,” pp. 13–23, 2019.
- [9] M. I. Profile, “Global Mobile Apps,” no. December, 2017.
- [10] W. N. Picoto, R. Duarte, and I. Pinto, “Uncovering top-ranking factors for mobile apps

- through a multimethod approach,” *J. Bus. Res.*, vol. 101, no. January, pp. 668–674, 2019.
- [11] C. Schmitz, S. Bartsch, and A. Meyer, “Mobile App Usage and its Implications for Service Management – Empirical Findings from German Public Transport,” *Procedia - Soc. Behav. Sci.*, 2016.
 - [12] V. Yberg, “Native-like Performance and User Experience with Progressive Web Apps,” *Degree Proj. Comput. Sci. Eng.*, 2018.
 - [13] “Need To Know About...,” *Econtent Wilt.*, vol. 41, no. 2, pp. 20–24, 2018.
 - [14] N. Sobotker, “Progressive web apps - Why brands might want to get progressive,” *Bizcommunity.Com*, 2016.
 - [15] M. & S. and T. R. H. K. B. M. Department for Digital, Culture, “UK Digital Strategy,” *GOV.UK*, 2017. [Online]. Available: <https://www.gov.uk/government/publications/uk-digital-strategy>.
 - [16] Pekley James, *Entrepreneurial Capitalism and Innovation: A History of Computer Communications 1968-1988*. 2007.
 - [17] A. Hern, “We can get the web we want,” *The Guardian*, 2019.
 - [18] C. Simmons, *Instant Responsive Web Design*. Packt Publishing, 2019.
 - [19] K. Jacksi and S. M. Abass, “Development History Of The World Wide Web,” no. September, 2019.
 - [20] K. Nath, S. Dhar, and S. Basishttha, “Web 1.0 to Web 3.0 - Evolution of the Web and its various challenges,” *ICROIT 2014 - Proc. 2014 Int. Conf. Reliab. Optim. Inf.*

- Technol.*, pp. 86–89, 2014.
- [21] T. Berners-Lee, J. Hendler, and O. Lassila, “the Semantic Web.Pdf,” *Text*, vol. 8, pp. 1–10, 2008.
 - [22] J. Zhang, X. Li, and L. Zhang, “Exploring the virtual reference service based on Web 3.0 environments in the library,” *Proc. - 2015 8th Int. Conf. Biomed. Eng. Informatics, BMEI 2015*, no. Bmei, pp. 862–866, 2016.
 - [23] Y.-C. Pauwels, Pieter, Zhang, S., & Lee, “Semantic web technologies in AEC industry : a literature review,” *Autom. Constr.*, vol. 73, pp. 145–165., 2017.
 - [24] S. Murugesan, G. Rossi, L. Wilbanks, and R. Djavanshir, “The future of web apps,” *IT Prof.*, vol. 13, no. 5, pp. 12–14, 2011.
 - [25] R. Margea, C. Margea, B. Veche, and C. Hurbean, “Mobile First. Current Trends and Practices in Website Design,” *Ann. Dunărea Jos Univ. Fascicle I Econ. Appl. Informatics*, vol. 23, no. 3, pp. 164–173, 2017.
 - [26] “UK Digital Market Overview MMX multiplatform,” 2019.
 - [27] B. Fling, *Mobile Design and Development*. O’Reilly Media.
 - [28] K. Buettner and A. M. Simmons, “Mobile Web and Native Apps: How One Team Found a Happy Medium,” in *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, 2011, pp. 549–554.
 - [29] J.Clement, “Number of apps available in leading app stores as of 2nd quarter 2019,” 2019.
 - [30] L. Giurgiu and I. Gligorea, “Responsive Web Design Techniques,” *Int. Conf.*

KNOWLEDGE-BASED Organ., vol. 23, no. 3, pp. 37–42, 2017.

- [31] C. Rieger and T. A. Majchrzak, “Towards the definitive evaluation framework for cross-platform app development approaches,” *J. Syst. Softw.*, vol. 153, pp. 175–199, 2019.
- [32] A. Biørn-Hansen, T. M. Grønli, and G. Ghinea, “A survey and taxonomy of core concepts and research challenges in cross-platform mobile development,” *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–35, 2019.
- [33] W. S. El-Kassas, B. A. Abdullah, A. H. Yousef, and A. M. Wahba, “Taxonomy of Cross-Platform Mobile Applications Development Approaches,” *Ain Shams Eng. J.*, vol. 8, no. 2, pp. 163–190, 2017.
- [34] L. Delia, N. Galdamez, P. Thomas, L. Corbalan, and P. Pesado, “Multi-platform mobile application development analysis,” *Proc. - Int. Conf. Res. Challenges Inf. Sci.*, vol. 2015-June, no. June, pp. 181–186, 2015.
- [35] “Flutter.”
- [36] Conor McGann, “Unite AMP and PWAs,” *net*, Bournemouth UK, pp. 90–93, May-2019.
- [37] G. Tott, “The Rise of the Web App,” 2019.
- [38] “WebAssembly,” *webassembly.org*. [Online]. Available: <https://webassembly.org/>.
- [39] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, “Progressive Web Apps: The Possible Web-native Unifier for Mobile Development,” 2017.
- [40] A. Biørn-Hansen, T. M. Grønli, G. Ghinea, and S. Alouneh, “An Empirical Study of

- Cross-Platform Mobile Development in Industry,” *Wirel. Commun. Mob. Comput.*, vol. 2019, 2019.
- [41] Developers.Google.com, “PWA Showcase,” *Developers.Google.com*, 2017. [Online]. Available: <https://developers.google.com/web/showcase/2017/bookmyshow>.
- [42] T. A. Majchrzak, A. Biørn-Hansen, and T.-M. Grønli, “Progressive Web Apps: the Definite Approach to Cross-Platform Development?,” *Proc. 51st Hawaii Int. Conf. Syst. Sci.*, pp. 5735–5744, 2018.
- [43] Z. J. Chen, D. Vogel, and Z. H. Wang, “How to satisfy citizens? Using mobile government to reengineer fair government processes,” *Decis. Support Syst.*, vol. 82, pp. 47–57, 2016.
- [44] Tal Ater, *Building Progressive Web Apps Bringing the Power of Native Apps to the Browser*. O’Reilly Media, 2017.
- [45] J. Archer, “Mobile first: why are we getting it wrong ?,” 2016. [Online]. Available: <http://jamesarcher.me/mobile-first>.
- [46] A. Ritiņš, A. Sergejevs, and S. Kodors, “PERSPECTIVES OF PROGRESSIVE WEB APPS,” *Human. Environ. Technol.*, 2018.
- [47] L. E. Nugroho, A. G. H. Pratama, I. W. Mustika, and R. Ferdiana, “Development of monitoring system for smart farming using Progressive Web App,” *2017 9th Int. Conf. Inf. Technol. Electr. Eng. ICITEE 2017*, vol. 2018-Janua, pp. 1–5, 2018.
- [48] V. Karpagam, “PERFORMANCE ENHANCEMENT OF WEBPAGE USING PROGRESSIVE WEB APP FEATURES,” *Int. J. Innov. Res. Adv. Eng.*, 2017.

- [49] N. D. Kho, “Everything you need to know about: Progressive web apps,” *EContent*. 2018.
- [50] G. Taylor, “PWAs Remain A Work In Progress, But Faster Load Times Show Promise,” 2019.
- [51] S. Baraković and L. Skorin-Kapov, “Modelling the relationship between design/performance factors and perceptual features contributing to Quality of Experience for mobile Web browsing,” *Comput. Human Behav.*, vol. 74, pp. 311–329, 2017.
- [52] T. Fertig and P. Braun, “Towards Offline Support for RESTful Systems,” pp. 1875–1880, 2018.
- [53] N. Pande, A. Somani, S. Prasad Samal, and V. Kakkirala, “Enhanced Web Application and Browsing Performance through Service-Worker Infusion Framework,” *Proc. - 2018 IEEE Int. Conf. Web Serv. ICWS 2018 - Part 2018 IEEE World Congr. Serv.*, pp. 195–202, 2018.
- [54] I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirovic, “Assessing the Impact of Service Workers on the Energy Efficiency of Progressive Web Apps,” in *Proceedings - 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2017*, 2017.
- [55] J. Lee, H. Kim, J. Park, I. Shin, and S. Son, “Pride and Prejudice in Progressive Web Apps:, Abusing Native App-like Features in Web Applications,” in *CCSI8*, 2018.
- [56] R. Harris, “The future of Progressive Web Apps,” *AppDeveloper*, 2018.
- [57] A. Kumar and R. K. Singh, “Comparative analysis of angularjs and reactjs,” *Int. J.*

- Latest Trends Eng. Technol.*, vol. 7, no. 4, pp. 225–227, 2016.
- [58] S. Aggarwal, “Modern Web-Development using ReactJS,” *Int. J. Recent Res. Asp.*, vol. 5, no. 1, pp. 133–137, 2018.
- [59] Google Developer, “Nikkei achieves a new level of quality and performance with their multi-page PWA | Web | Google Developers,” no. March 2016, 2017.
- [60] G. D. A. Cardieri and L. M. Zaina, “Analyzing User Experience in Mobile Web , Native and Progressive Web Applications : A User and HCI Specialist Perspectives.”
- [61] A. Gambhir and G. Raj, “Analysis of Cache in Service Worker and Performance Scoring of Progressive Web Application,” in *Proceedings on 2018 International Conference on Advances in Computing and Communication Engineering, ICACCE 2018*, 2018.
- [62] O. H. Ludwig Bäcklund, “Evaluating React Native and Progressive Web App development using ISO 25010,” 2018.
- [63] Bernard le Masson, “Government As A Platform,” *Accenture*, 2018. [Online]. Available: <https://www.accenture.com/gb-en/insights/public-service/government-as-a-platform>. [Accessed: 16-Sep-2019].
- [64] D. Isa and P. Directorate-general, *eGovernment factsheets anniversary report*. 2019.
- [65] M. Hussain and A. Imran, “mGovernment services and adoption: Current research and future direction,” *IFIP Adv. Inf. Commun. Technol.*, vol. 429, pp. 311–323, 2014.
- [66] J. M. Marc Holzer, Aroon Manoharan, *E-Government and Information Technology Management: Concepts and Best Practices*. Melvin & Leigh, 2019.

- [67] R. K. Kanaan, "Implementation of M-government : Leveraging Mobile Technology to Streamline the E-governance Framework," vol. 8, no. 3, pp. 495–508, 2019.
- [68] J. T. Tien Nguyen, Amit Goyal, Subodh Manicka, M Hazli M Nadzri, Bhargav Perepa, Sudhir Singh, "IBM MobileFirst in Action for mGovernment and Citizen Mobile Services," IBM Redbooks, 2015, pp. 3–9.
- [69] I. Almarashdeh and M. K. Alsmadi, "How to make them use it? Citizens acceptance of M-government," *Appl. Comput. Informatics*, vol. 13, no. 2, pp. 194–199, 2017.
- [70] S. K. Sharma, A. Al-Badi, N. P. Rana, and L. Al-Azizi, "Mobile applications in government services (mG-App) from user's perspectives: A predictive modelling approach," *Gov. Inf. Q.*, vol. 35, no. 4, pp. 557–568, 2018.
- [71] L. Carter, V. Weerakkody, B. Phillips, and Y. K. Dwivedi, "Citizen Adoption of E-Government Services: Exploring Citizen Perceptions of Online Services in the United States and United Kingdom," *Inf. Syst. Manag.*, vol. 33, no. 2, pp. 124–140, 2016.
- [72] pp. 39-49 Shareef, M. A.; Archer, N.; and Dwivedi, Y. K. (2012). . , 53(2), "Examining Adoption Behavior of Mobile Government," *J. Comput. Inf. Syst.*, vol. 53, no. 2, pp. 39–49, 2012.
- [73] NIC, "Arkansas.gov." [Online]. Available: <https://www.egov.com/who-we-serve/partners/arkansas>.
- [74] N. Walravens, "A Business Model Approach to Local mGovernment Applications - Mapping the Brussels Region's Mobile App Initiatives," *2015 12th Int. Jt. Conf. E-bus. Telecommun.*, vol. 02, pp. 113–124, 2015.
- [75] M. Shepherd, "Future Technology in Government," *Institute for Government*, 2019.

- [Online]. Available: <https://www.instituteforgovernment.org.uk/explainers/future-technology-government>.
- [76] S. Domes, *Progressive Web Apps with React*. Packt Publishing, 2017.
- [77] Carolyn B Seaman, “Qualitative methods in empirical studies of software engineering,” *IEEE Trans. Softw. Eng.*, vol. 25, no. 4, pp. 557–572, 1999.
- [78] K. J. Stol and B. Fitzgerald, “A Holistic Overview of Software Engineering Research Strategies,” *Proc. - 3rd Int. Work. Conduct. Empir. Stud. Ind. CESI 2015*, pp. 47–54, 2015.
- [79] J. Creswell, *Qualitative Inquiry and Research Design*, 2nd editio. USA: Sage Publications, 2007.
- [80] K. F. Hyde, “Recognising deductive processes in qualitative research,” *Qual. Mark. Res. An Int. J.*, vol. 3, no. 2, pp. 82–90, 2000.
- [81] P. R. and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empir. Softw. Eng. online J.*, 2008.
- [82] G. A. Bowen, “Document Analysis as a Qualitative Research Method,” *Qual. Res. J.*, vol. 9, no. 2, pp. 27–40, 2009.
- [83] S. E. Group and S. S. School of Computer Science and Mathematics Keele University Keele, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” *Biomed. Environ. Sci.*, vol. 13, no. 1, pp. 37–43, 2007.
- [84] S. N. Kollár Lajos, “Case study in system development - Notes,” 2013.
- [85] I. Sommerville, *Software Engineering*, 10th Editi. Pearson, 2016.

- [86] S. Aftab *et al.*, “Using FDD for small project: An empirical case study,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 3, pp. 151–158, 2019.
- [87] S. A. K. Gahyyur, A. Razzaq, S. Z. Hasan, S. Ahmed, and R. Ullah, “Evaluation for feature driven development paradigm in context of architecture design augmentation and perspective implications,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 3, pp. 236–247, 2018.
- [88] R. Anthony, “6 Feature-Driven Development,” 1999.
- [89] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah, “Requirements Prioritization Techniques Comparison,” *Mod. Appl. Sci.*, vol. 12, no. 2, p. 62, 2018.
- [90] M. Fowler, *Patterns of enterprise application architecture (Паттерны корпоративных приложений)*. 2003.
- [91] M. J. Hernandez, *Database Design For Mere Mortals*. Michigan: Addison-Wesley, 2013.
- [92] Steve Krug, *Don't make me think*, 3rd editio. USA: New Riders, 2014.
- [93] Google, “Progressive Web App Checklist,” *Google Developers*, 2017. [Online]. Available: <https://developers.google.com/web/progressive-web-apps/checklist>.
- [94] D. Sheppard and D. Sheppard, “Background Sync for Offline Apps with Service Workers,” in *Beginning Progressive Web App Development*, 2017.
- [95] DVLA, “Documents.”
- [96] Webaim projects, “Screen reader survey,” *WebAim*, 2019.
- [97] M. Firtman, “Whats new on iOS 12.2 for PWA,” 2019. [Online]. Available:

- <https://medium.com/@firt/whats-new-on-ios-12-2-for-progressive-web-apps-75c348f8e945>.
- [98] S. F. Express, N. Delhi, and N. Delhi, “PWA will become mainstream in next 12-24 months,” no. May, pp. 1–3, 2017.
 - [99] “The Architect’s Guide to Progressive Web Apps.”
 - [100] “Autoilija app,” *Finnish Transport and Communication Agency*, 2019. .
 - [101] S. Thakur and H. Singh, “FDRD: Feature driven reuse development process model,” *Proc. 2014 IEEE Int. Conf. Adv. Commun. Control Comput. Technol. ICACCCT 2014*, no. May 2014, pp. 1593–1598, 2015.
 - [102] A. Javeed, “Performance Optimization Techniques for ReactJS,” *Proc. 2019 3rd IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2019*, pp. 1–5, 2019.
 - [103] A. Doupé, P. Mutchler, J. Mitchell, C. Kruegel, and G. Vigna, “A Large-Scale Study of Mobile Web App Security CTFs for Education View project A Large-Scale Study of Mobile Web App Security,” no. June, 2015.
 - [104] T. Mikkonen and A. Taivalsaari, “Apps vs. open web: The battle of the decade,” *Proc. 2nd Work. Softw. Eng. Mob. Appl. Dev.*, pp. 22–26, 2011.
 - [105] K. T. and L. Silver, “Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally,” *Pew Research center*, 2019. .
 - [106] T. O. and J. Battelle, “Web Squared: Web 2.0 Five Years On,” in *Web2.0 summit*, 2009.
 - [107] N. Mishra, “How to Select App Development Frameworks: Native vs. Web vs.

- Hybrid,” *DZone/Mobile Zone*, 2016. [Online]. Available:
<https://dzone.com/articles/how-to-select-app-development-frameworks-native-vs>.
- [108] J. Jewer and D. R. Compeau, “Understanding IS Adoption and Success : Integration of IS Success and Technology Adoption Research Full Paper,” no. 1991, pp. 1–10, 2017.
- [109] M. Saunders, “(PDF) Understanding research philosophies and approaches,” no. January 2009, 2016.
- [110] N. Bouraqadi and D. Mason, “Test-driven development for generated portable Javascript apps,” *Sci. Comput. Program.*, vol. 161, pp. 2–17, 2018.
- [111] F. Besson, P. Moura, F. Kon, and D. Milojicic, “Bringing Test-Driven Development to web service choreographies,” *J. Syst. Softw.*, vol. 99, pp. 135–154, 2015.
- [112] J. Itoi, M. Sasabe, J. Kawahara, and S. Kasahara, “An offline mobile application for automatic evacuation guiding in outdoor environments,” *Sci. Phone Apps Mob. Devices*, vol. 3, no. 1, pp. 1–10, 2017.
- [113] & P. di M. Capgemini, Sogeti, IDC, *eGovernment Benchmark 2018. Securing eGovernment for all*. 2018.
- [114] Krishna Hari, “Building an offline Web App that works in very low internet conditions using ServiceWorkers,” *Progressive Web Apps*, 2017. .
- [115] F. Shahzad, “Modern and Responsive Mobile-enabled Web Applications,” in *The 12th International Conferene on Future Mobile and Communications*, 2016.

Bibliography

- M. Firtman, “Whats new on iOS 12.2 for PWA,” 2019. [Online]. Available: <https://medium.com/@firt/whats-new-on-ios-12-2-for-progressive-web-apps-75c348f8e945>.
- S. F. Express, N. Delhi, and N. Delhi, “PWA will become mainstream in next 12-24 months,” no. May, pp. 1–3, 2017.
- S. N. Kollár Lajos, “Case study in system development - Notes,” 2013.
- “The Architect’s Guide to Progressive Web Apps.”
- “Autoilija app,” *Finnish Transport and Communication Agency*, 2019. .
- Prioritization Techniques Comparison,” *Mod. Appl. Sci.*, vol. 12, no. 2, p. 62, 2018.
- S. Thakur and H. Singh, “FDRD: Feature driven reuse development process model,” *oc. 2014 IEEE Int. Conf. Adv. Commun. Control Comput. Technol. ICACCCT 2014*, no. May 2014, pp. 1593–1598, 2015.
- A. Javeed, “Performance Optimization Techniques for ReactJS,” *Proc. 2019 3rd IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2019*, pp. 1–5, 2019.
- A. Doupé, P. Mutchler, J. Mitchell, C. Kruegel, and G. Vigna, “A Large-Scale Study of Mobile Web App Security CTFs for Education View project A Large-Scale Study of Mobile Web App Security,” no. June, 2015.
- T. Mikkonen and A. Taivalsaari, “Apps vs. open web: The battle of the decade,” *Proc. 2nd Work. Softw. Eng. Mob. Appl. Dev.*, pp. 22–26, 2011.
- K. T. and L. Silver, “Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally,” *Pew Research center*, 2019. .
- T. O. and J. Battelle, “Web Squared: Web 2.0 Five Years On,” in *Web2.0 summit*, 2009.
- K. Nath, S. Dhar, and S. Basishta, “Web 1.0 to Web 3.0 - Evolution of the Web and its

various challenges,” *ICROIT 2014 - Proc. 2014 Int. Conf. Reliab. Optim. Inf. Technol.*, pp. 86–89, 2014.

N. Mishra, “How to Select App Development Frameworks: Native vs. Web vs. Hybrid,” *DZone/Mobile Zone*, 2016. [Online]. Available: <https://dzone.com/articles/how-to-select-app-development-frameworks-native-vs>.

J. Jewer and D. R. Compeau, “Understanding IS Adoption and Success : Integration of IS Success and Technology Adoption Research Full Paper,” no. 1991, pp. 1–10, 2017.

C. Simmons, *Instant Responsive Web Design*. Packt Publishing, 2019.

M. Saunders, “(PDF) Understanding research philosophies and approaches,” no. January 2009, 2016.

N. Bouraqadi and D. Mason, “Test-driven development for generated portable Javascript apps,” *Sci. Comput. Program.*, vol. 161, pp. 2–17, 2018.

F. Besson, P. Moura, F. Kon, and D. Milojevic, “Bringing Test-Driven Development to web service choreographies,” *J. Syst. Softw.*, vol. 99, pp. 135–154, 2015.

J. Itoi, M. Sasabe, J. Kawahara, and S. Kasahara, “An offline mobile application for automatic evacuation guiding in outdoor environments,” *Sci. Phone Apps Mob. Devices*, vol. 3, no. 1, pp. 1–10, 2017.

& P. di M. Capgemini, Sogeti, IDC, *eGovernment Benchmark 2018. Securing eGovernment for all*. 2018.

N. Pande, A. Somani, S. Prasad Samal, and V. Kakkirala, “Enhanced Web Application and Browsing Performance through Service-Worker Infusion Framework,” *Proc. - 2018 IEEE Int. Conf. Web Serv. ICWS 2018 - Part 2018 IEEE World Congr. Serv.*, pp. 195–202, 2018.

Krishna Hari, “Building an offline Web App that works in very low internet conditions using ServiceWorkers,” *Progressive Web Apps*, 2017. .

F. Shahzad, “Modern and Responsive Mobile-enabled Web Applications,” in *The 12th*

International Conference on Future Mobile and Communications, 2016.

P. R. and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empir. Softw. Eng. online J.*, 2008.