Bachelor's thesis

Information and Communications Technology

May 2023

Frank Olowoniyi

# Design and implementation of a PWA for ordering taxi

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

May 2023 | number of pages: 30

Frank Olowoniyi

# Design and implementation of a PWA for ordering taxi

This thesis addresses the need for the design and implementation of a Progressive Web Application (PWA) for ordering taxi services. The research focused on designing and implementing a PWA specifically tailored for efficient and user-friendly taxi booking.

This work was conducted to explore the potential of PWAs in enhancing the accessibility, convenience, and efficiency of taxi services. The PWA was developed using web technologies such as HTML, CSS, and JavaScript as well as 3rd party APIs and libraries such as React.js, Open Street Map (OSM) API for incorporating real-time location tracking and bootstrap for a responsive user interface.

The main results include a fully functional PWA for ordering taxi services with features like real-time location tracking, and intuitive user experience across multiple devices. The significance of these results lies in demonstrating the effectiveness of PWAs as a modern and cross-platform solution for streamlining taxi booking processes, offering convenience to users and enabling taxi service providers to offer an enhanced customer experience.

Keywords: Transport, Taxi, Cab, Progressive Web Apps (PWAs), Software design, Software development, User experience, Mobility, User interface design, Responsive design, Device Screens

# Contents

List of abbreviations ..... 5

List of Tables and Figures ..... 6

1 Introduction ..... 7

1.1 Background and motivation ..... 7

1.2 Scope and Objectives ..... 8

1.3 Assumptions and Limitations ..... 8

1.4 Structure of thesis ..... 9

2 Literature Review ..... 10

2.1 Overview of Progressive Web Applications ..... 10

2.2 Comparison of PWAs with Native Mobile Apps ..... 10

2.3 Key Features and Capabilities of PWAs ..... 12

2.4 Implications for mobile development ..... 13

2.5 Examples of Successful PWAs ..... 14

3 Tools, Design and Architecture of the taxi order PWA ..... 15

3.1 Architecture and Design of the PWA for Taxi Order Fulfillment ..... 15

3.2 User Interface and User Experience Design of the taxi service PWA ..... 15

3.3 Data and Business logic of the PWA ..... 16

4 Implementation and Testing of the PWA ..... 18

4.1 Implementation Details of the PWA ..... 18

4.1.1 App shell ..... 18

4.1.2 The Manifest ..... 19

4.1.3 Service worker ..... 20

4.3 Lighthouse: Testing of the PWA ..... 21

4.4 Integration with Third-Party Services and APIs ..... 22
4.5 Accessibility and Security Considerations ..... 23
5 Discussion ..... 24
5.1 Summary of key findings ..... 24
5.2 Recommendations for creating and deploying PWAs ..... 25
6 Conclusion and Future Work ..... 26
6.1 Conclusion ..... 26
6.2 Future Work ..... 26
REFERENCES ..... 28
APPENDIX 1 ..... 30

## List of abbreviations

| | |
|---|---|
| PWA | Progressive Web App |
| MTB | Mobile Taxi Booking |
| JS | JavaScript |
| JSX | JavaScript XML |
| ARIA | Accessible Rich Internet Applications |
| HTTPS | HyperText Transfer Protocol Secure |
| API | Application Programming Interface |
| HTML | HyperText Markup Language |
| CSS | Cascade Style Sheet |
| JSON | JavaScript Object Notation |
| UI/UX | User Interface /User Experience |
| JWT | JSON Web Token |

# List of Tables and Figures

Table 1: Feature comparison of PWA, native and standard web app

Figure 1: React component used for displaying notification in the PWA
Figure 2: Flow of data in the PWA
Figure 3: App shell
Figure 4: JSON-formatted manifest
Figure 5: Service worker in reactjs
Figure 6: Lighthouse report of the taxi order PWA
Figure 7: Performance of the PWA as reported by the lighthouse chrome tool
Figure 8: Browsers that supports PWA.

# 1 Introduction

## 1.1 Background and motivation

Several studies have been carried out on the importance of the taxi transport and its accessibility. According to Weng et al., Mobile Taxi Booking (MTB) Apps have been developed in cities as a bridge to connect passengers and taxis. With an MTB App, passengers can search for available taxis around them and make an order. The app sends the request to the nearest available driver who then either accepts or declines the trip. The MTB Apps, such as Uber and GrabCar, are varying the method in which passengers' book taxis. As developers release more MTB Apps, users are now left with many opportunities to access them. It is essential for the MTB App's service providers to know how to keep their present users. It is also beneficial for the MTB's service providers to know how the users develop their continuous intent so that these service providers can provide them with new MTB applications to satisfy their needs. (Weng et al., 2017 p.207)

Transportation and mobility of people from one location to another is a topic as old as human existence itself. People need travelling either for survival, business or leisure purposes. It is, therefore, essential to be able to access transport services timely and with ease in a given society (Habermann et al., 2016, p. 1).

The global population has in recent times recorded 59.5 percent of active internet users and 92.6% of these users have accessed the internet through their mobile devices. Mobile users are able to obtain different kinds of apps from the Apple store which has about 2 million apps or the Google play store which has about 3 million apps available for download and install. Although, there is a limit to the quantity of apps users can have on their mobile device due to certain restraints and capabilities of the device itself. (Jhala, 2021, p. 1).

## 1.2 Scope and Objectives

The main focus of this thesis project is to build a platform for users to order a taxi in a timely manner. According to Habermann et al. (2016), the new developments in Information and Communications Technology combined with smart desktop and mobile devices presents a great opportunity to offer potentially ubiquitous transport services.

The aim of this thesis is to design a full stack web/mobile platform for ordering a taxi ride. In order to achieve this, the concept of Progressive Web Application (PWA) would be used to achieve cross platform deployment. In essence, it would be an application fully functional and responsive in the web browser and also possible to install like a native mobile application.

## 1.3 Assumptions and Limitations

This thesis assumes that more than 90 percent of residents of the case study city uses their mobile devices to access the internet. However, it is essential that the mobile platforms are compliant with the PWA architectural design.

According to Jhala (2021), the main requirements for build PWAs include:

- An application shell
- Service worker
- Manifest

While it is widely true and acceptable that PWAs have the possibility of offering offline activities to users, this activity with regards to this project may not include order of the taxi because it requires an internet connection and GPS location. However, users can use the app to access the usage history, profile information and route planning on the map as offline capabilities of the software application.

## 1.4 Structure of thesis

This thesis work is structured in seven chapters in order to facilitate the proper layout of the conceptual framework used in the research process. The first chapter introduces the topic of this paper, discusses the background and what motivated the research while presenting research questions on the topic. This chapter also presents possible assumptions and the limitations of the research. The second chapter is a section for literature review of PWAs and its application to the taxi industry. The third chapter discusses the architecture of PWAs, the user interface design as well as third party integrations. The fourth chapter discusses how the app has been implemented to fulfil the purpose of design and also compared with other native apps. The fifth chapter discusses the results of the tests and analysis of the study performance metrics and satisfaction metrics. The sixth chapter discusses the findings of the study as well as the areas for further research. The seventh chapter reflects on the work carried out in this thesis and how this work could be further developed.

# 2 Literature Review

One of the key elements of this paper is that it relies on multiple sources of information through different databases and multiple academic disciplines.

## 2.1 Overview of Progressive Web Applications

Progressive Web Applications (PWAs) are web applications that utilize modern web technologies to provide an app-like experience to users. PWAs are designed to be fast, reliable, and engaging, with features that were previously only available in native mobile apps. PWAs are built using web technologies such as HTML, CSS, and JavaScript and can be accessed through any modern web browser on any device. [1]

Progressive web apps are the best of the web and the best of apps. They are convenient for users as no installation is required from the first time you access the app in the browser tab. While the user gradually establishes a relationship with the app, as time goes on, it becomes a bit more user-friendly because of the possibilities such as offline caching, faster load time, possibility to add the app to device home screen for quick accessibility and so on. Loading quickly even over unreliable network, sends relevant push notifications quickly and can have an icon on the home screen for launching a high-quality full screen app user experience. (Kumar, 2018 p. 1)

## 2.2 Comparison of PWAs with Native Mobile Apps

Native mobile apps are built using native programming languages and tools specific to the operating system of the device. They are downloaded from an app store and installed on the device, requiring a significant amount of storage space.

In contrast, PWAs do not require installation and can be accessed through a web browser. PWAs also require less storage space than native mobile apps. The tabular display of the comparison between native apps, standard web apps and PWAs is presented in the Table 1 below.

TABLE 1. FEATURE COMPARISON OF PWA, NATIVE AND STANDARD WEB APP

| Feature | Native App | Standard Web App | PWA |
|---|---|---|---|
| Installable | Yes | Not needed | Can add to home screen |
| Offline Access | Yes | Not needed | Need to use the app at least once online, the possible to access offline |
| Testable before Installation | No | Yes | |
| Push notification | Yes | Yes (with third party services | Yes |
| Discoverability | Comparatively limited | Not needed | Yes (varying as per browsers)<br><br>Good to make appear in search results, need to be optimized for SEO |

## 2.3 Key Features and Capabilities of PWAs

PWAs have several key features that make them appealing to developers and users alike. One of the main advantages of PWAs is their ability to work offline, meaning that users can still access the app even without an internet connection. This is achieved through the use of a service worker, which caches the app's content and resources. PWAs also have the ability to send push notifications, which can be used to keep users, engaged with the app. Additionally, PWAs can be added to the home screen of a user's device, allowing them to be accessed like a native mobile app.

Features of progressive web apps include the following:

- Being progressive in nature i.e., a task for every user irrespective of the browser choice since they are built with progressive enhancements as a core.
- They are responsive, which means that they fit very on various devices with difference screen sizes
- They are not dependent on connectivity because they have been enhanced with service workers to work offline or on low quality networks
- App-like – It is designed to fit on device screens like an app and also possibility to make launch icon on home screen for users.
- Always up-to-date due to the service worker’s update process.
- They are safe since they served via HTTPS to prevent spying and content integrity.
- They can be searched and discovered or identified as applications due to the manifests and service worker registration which optimizes them for search engines
- They utilize push notifications to engage and re-engage users effectively
- PWAs are installable and can allow users to keep on home screen where it can easily be found.
- It can also function like a website on desktop devices. (Jhala, 2021. p.208)

## 2.4 Implications for mobile development

In 2014, mobile devices surpassed desktops as the primary means of accessing the internet worldwide. This shift underscores the growing importance of optimizing web applications for mobile devices. To address the limitations of the web platform on mobile, companies often develop native or hybrid applications for specific operating systems like iOS and Android. Native applications are coded in device-specific languages and offer rich access to device hardware, while hybrid applications leverage web technologies but can mimic native app behavior with the help of frameworks like Apache Cordova. (Tandel et al., 2018, 9440.)

Introduced by Google in 2015, Progressive Web Applications (PWAs) present a compelling fourth option for mobile application development. PWAs are web applications that strive to deliver a native-like user experience on mobile devices; including features like offline support and push notifications. By combining the benefits of web technologies with the capabilities of native apps, PWAs offer a promising alternative that bridges the gap between web and native mobile experiences. (Tandel et al., 2018, 9439.)

Mobile users have a strong aversion to delays and unreliability. The frustration caused by mobile delays can be compared to the anxiety experienced while watching a horror movie. Approximately fifty percent of smartphone users prefer to use a company's mobile site rather than downloading an app when they browse or shop online. Limited storage space is one of the primary reasons why users uninstall apps. In contrast, Progressive Web Apps (PWAs) typically require less than 1MB of storage space. Mobile users are more likely to make purchases from mobile sites that provide relevant product recommendations.

Additionally, a significant 85% of smartphone users find mobile notifications to be useful. Based on these observations, it was identified that a client’s preferences

are for experiences that are fast, installable, reliable, and engaging (F.I.R.E.). (Fourault, 2022)

## 2.5 Examples of Successful PWAs

Several well-known companies have successfully implemented PWAs, including Twitter, Pinterest, and Forbes. For example, Twitter's PWA, Twitter Lite, has seen a significant increase in user engagement and a decrease in data usage compared by as much as 70% to the company’s native mobile app. Pinterest's PWA has resulted in a 60% increase in user engagement and a 44% increase in user-generated ad revenue. Forbes' PWA has resulted in a 43% increase in sessions per user and a 20% increase in ad viewability.

A particular cab booking app name Ola in India is another typical example of a successful case study. Ola is a leading cab aggregator in India with a great mission of driving mobility for the Indian society. It has achieved 68% increase in mobile traffic of tier 2 and 3 cities. It’s PWA application size 200kb, which is about 300 to 500 times smaller than native apps such as iOS and Android. A record of 20% of their users had previously uninstalled their app and now booked via the PWA.

# 3 Tools, Design and Architecture of the taxi order PWA

## 3.1 Architecture and Design of the PWA for Taxi Order Fulfillment

The prototype of the application was first laid out using Figma to model the layout of the user interface. The prototype designed using Figma can be found in Appendix 1 of this thesis. Figma provides designers with a collaborative platform to create and prototype digital experiences in real-time, all within a unified space. This facilitates the quick transformation of their ideas and visions into tangible products. [2]

The PWA for taxi order fulfillment has a server-side and client-side model architecture pattern. The model presents the data and business logic of the application on the server-side, the client-side represents the user interface where the users are able to interact with the PWA. The PWA utilizes several web technologies, including HTML, CSS, JavaScript (Node.js) and the ReactJS library. In order to achieve the PWA design architecture, the app mainly requires a service worker, a manifest and the app shell.

## 3.2 User Interface and User Experience Design of the taxi service PWA

The user interface components are designed using ReactJS and styled with Bootstrap CSS. According to the W3School, bootstrap 5 is the newest version and the bootstrap framework is regarded as the most popular CSS framework used to design mobile-first and responsive websites. The Bootstrap framework was used in this app to ensure a responsive design across mobile and desktop devices. [3]

React is a JavaScript library used for building components of user interfaces. A component could be a button on the web page or an entire section of an app’s page as a component is basically a piece of the UI (User Interface) that possesses its own

logic and appearance. React apps are made up of several components. These components are JavaScript classes or functions that utilize JavaScript XML (JSX) which is rendered to return a markup. [4] The figure 1 below is an example of a react component.

Figure 1. React component used for displaying notification in the PWA

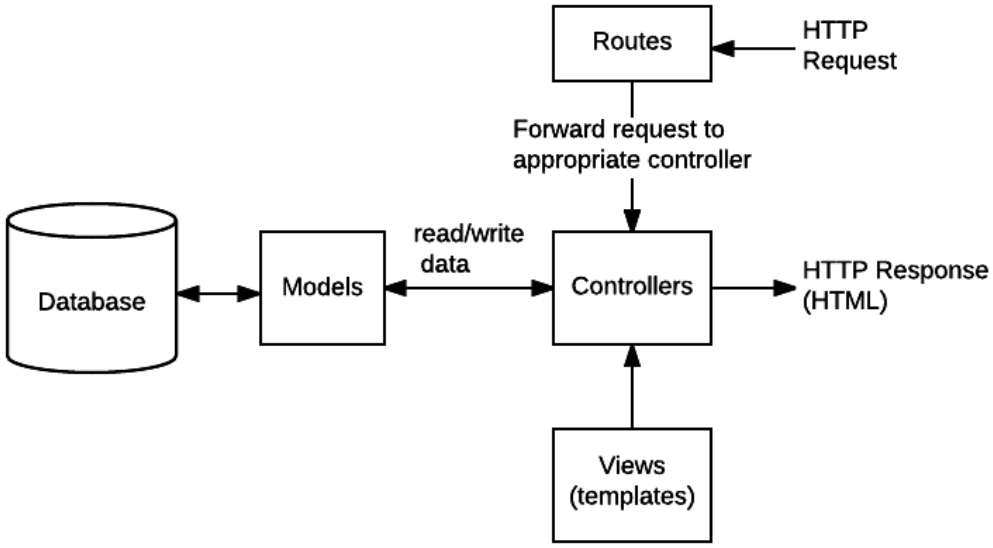
```
import React from 'react'

const Notification = ({notification}) =>{
  if(notification.length === 0) return null;

  if(notification[0] === 'success')
  return <div className='success'><h5>{notification[1]}</h5></div>;

  if(notification[0] === 'error')
  return <div className='error'><h5>{notification[1]}</h5></div>
}

export default Notification
```

The app also utilizes the React Router to handle navigation between different pages within the app. The user interface of the PWA for taxi order fulfillment is designed to be a responsive single page application (SPA) using the react router for navigation. The user experience is optimized for mobile devices, with features such as one-click ordering and real-time tracking of the user's location.

## 3.3 Data and Business logic of the PWA

The server-side of the application comprise mainly of the data and business logic of the application. The server is designed using the Node.js runtime and the data is stored using mongoDB. Figure 2. Below shows the flow of data with the Application.

Figure 2. Flow of data in the PWA




Node.js is a cross-platform JavaScript runtime environment and it is open source. Node.js along with express framework was used to design the model, controller and route responses that the server-side API provides to the client side. [5]

MongoDB is a noSQL database that stores data in JSON format which are referred to as documents. MongoDB was used to keep the app's data for persistent storage.

# 4 Implementation and Testing of the PWA

## 4.1 Implementation Details of the PWA

The PWA for taxi order fulfillment is implemented using ReactJS, a popular JavaScript library for building user interfaces. The app is designed as a single-page application, with ReactJS components for the user interface, data management using mongoDB.

Generally, there are 3 main components needed in a progressive web app namely: App shell, manifest and the service worker. These components are discussed in the following sub-paragraphs.

### 4.1.1 App shell

An app shell is the minimal HTML, CSS, and JavaScript required to run a Progressive Web App's UI and is one of the components that ensure consistently good performance. The first load of the PWA is very fast and caches immediately. This means you only get the content you need without having to load the shell every time instant loading web apps with an Application Shell Architecture. Figure 3 below shows the app shell of the PWA in react.

Figure 3: App shell (index.js) of the PWA

```
import React from 'react';
import App from './App';
import * as serviceWorkerRegistration from './serviceWorkerRegistration';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter as Router } from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Router>
      <App />
```

```
    </Router>
  </React.StrictMode>
);

serviceWorkerRegistration.register();

reportWebVitals();
```

The above figure is a code snippet taken from react PWA starter code index.js to demonstrate the PWA app shell in this thesis.

### 4.1.2 The Manifest

The file named manifest.json is JavaScript Object Notation (JSON)-formatted file that describes the app and it used to set app icon as well. See the sample code of a JSON-formatted manifest below in figure 4 using React JavaScript library.

Figure 4: JSON-formatted manifest (manifest.json)

```
{
  "short_name": "Taxiapp: order a ride from your device",
  "name": "Taxiapp",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192",
      "purpose": "maskable"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
```

```
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
 "background_color": "#ffffff"}
```

### 4.1.3 Service worker

The service worker gives the app the ability to work offline. It is mainly a script that enables background functionality without opening a web page or any interaction. See sample code using React JavaScript labeled as figure 5 below.

Figure 5: Service worker in React.js (service-worker.js)

```
import { clientsClaim } from 'workbox-core';
import { ExpirationPlugin } from 'workbox-expiration';
import { precacheAndRoute, createHandlerBoundToURL } from 'workbox-precaching';
import { registerRoute } from 'workbox-routing';
import { StaleWhileRevalidate } from 'workbox-strategies';

clientsClaim();
precacheAndRoute(self.__WB_MANIFEST);
const fileExtensionRegexp = new RegExp('/[^/?]+\\.[^/]+$');
registerRoute(

  ({ request, url }) => {

    if (request.mode !== 'navigate') {
      return false;
    }

    if (url.pathname.startsWith('/_')) {
```

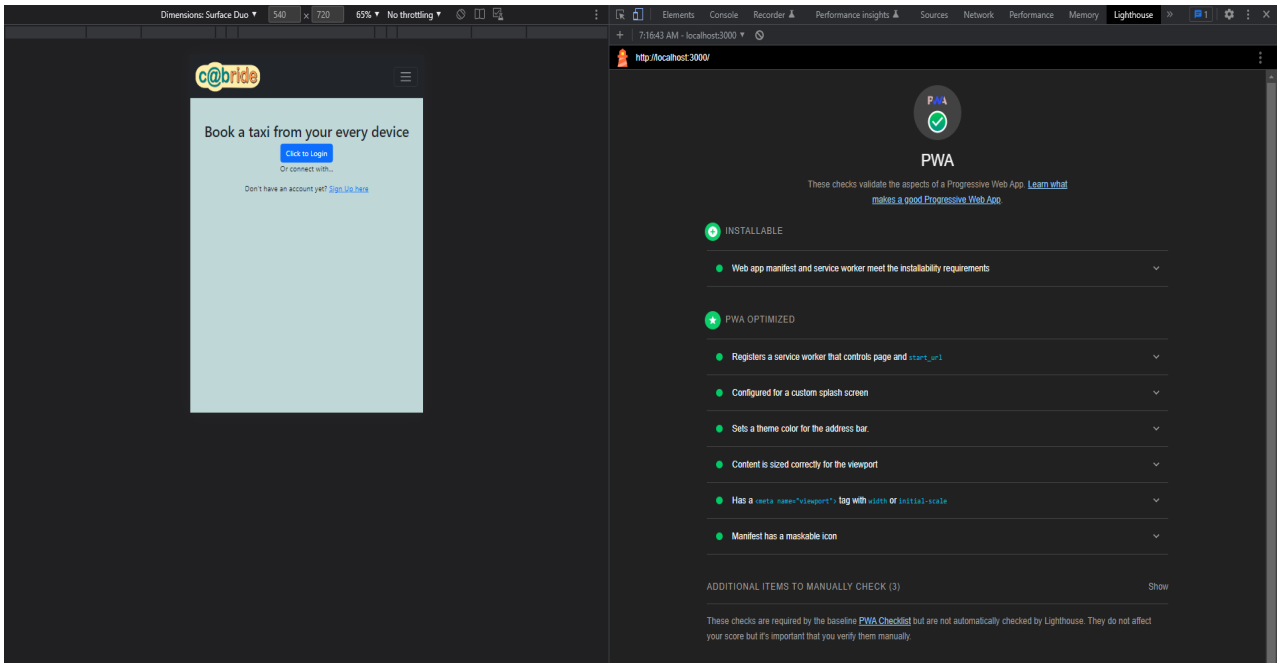
```
      return false;
    }
    if (url.pathname.match(fileExtensionRegexp)) {
      return false;
    }

    return true;
  },
  createHandlerBoundToURL(process.env.PUBLIC_URL + '/index.html')
);
registerRoute(

  ({ url }) => url.origin === self.location.origin &&
url.pathname.endsWith('.png'),
  new StaleWhileRevalidate({
    cacheName: 'images',
    plugins: [
      new ExpirationPlugin({ maxEntries: 50 }),
    ],
  })
);
self.addEventListener('message', (event) => {
  if (event.data && event.data.type === 'SKIP_WAITING') {
    self.skipWaiting();
  }
});
```

The above code snippet has been taken out the react PWA boiler-plate code for demonstration purposes in this thesis.

## 4.3 Lighthouse: Testing of the PWA

The Lighthouse Chrome extension, developed by Google, is specifically targeted towards Progressive Web App (PWA) developers. It serves as a tool for benchmarking websites based on specific metrics that Google deems significant, particularly for PWAs. While the extension can be used for non-PWA websites as well, their primary purposes is to assist in optimizing websites to improve rendering speed, reduce the time required for the first user interaction, and ensure compliance with Google's vision for PWAs and the future of the web, particularly on mobile devices. (Biorn-Hansen et al., 2018. p.74). Figure 6 illustrates the taxi order app Lighthouse report.

Figure 6: Lighthouse report of the taxi order PWA




The above figure was generated by the lighthouse chrome tool. The tool helps in determining if the app has been implemented as a PWA.

## 4.4 Integration with Third-Party Services and APIs

The PWA for taxi order fulfillment integrates with several third-party services and APIs, including Open Street Maps and Google Maps Geocoding API. Open street map is open source and is used to display the user's location and the location of available taxis in real-time.

## 4.5 Accessibility and Security Considerations

The PWA for taxi order fulfillment is designed to be accessible to users with disabilities. The app includes features such as alt text for images and Accessible Rich Internet Applications (ARIA) tags for screen readers. Security considerations are also taken into account, with features such as Hypertext Transfer Protocol Secure (HTTPS) encryption and JSON Web Token (JWT) used to authenticate and identify users and the login sessions.

# 5 Discussion

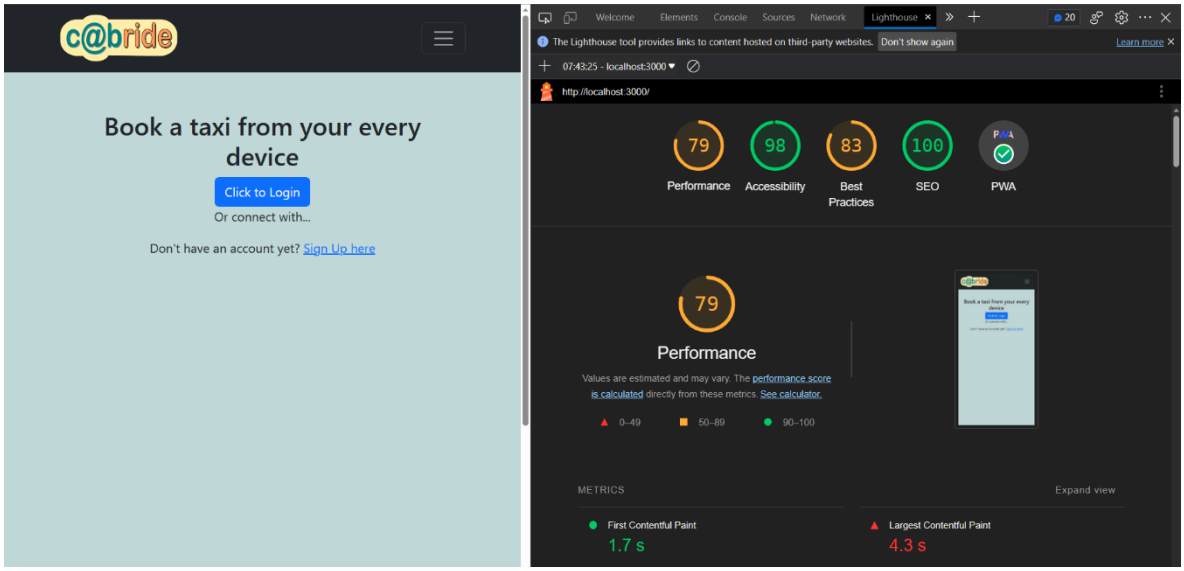
## 5.1 Summary of key findings

The implementation of the PWA resulted in certain key findings for evaluation and discussion in this thesis. After all the debugging necessary to achieve a successful PWA has been done, the lighthouse tool was used to perform further investigation into the current state and performance of the PWA.

When assessing a website, Lighthouse generates a detailed report indicating the current state of the tested website, offers resources on code and asset optimization, and provides an overall score. (Biorn-Hansen et al., 2018. p.74).

Figure 7 below shows a more detailed report on the taxi order PWA.

Figure 7. Performance of the PWA as reported by the lighthouse chrome tool



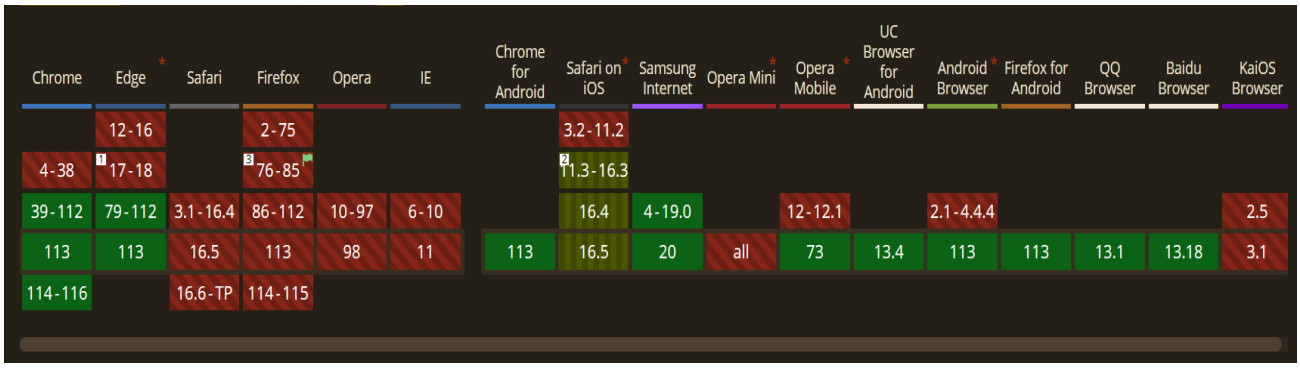

The performance of the PWA for taxi order fulfillment shows possible improvements to make the app perform even better as a PWA. The app loads

quickly, with an average load time of 2 seconds. The app also performs well on slow or unreliable internet connections, thanks to the app's use of service workers for caching and offline access.

## 5.2 Recommendations for creating and deploying PWAs

In many browsers, specific criteria determine whether a Progressive Web App (PWA) can be installed, and these browsers usually indicate to the user the installability of a PWA. For instance, indicators may include an Install button displayed in the address bar or an Install option within the overflow menu. Below is a picture labeled as picture 1 that displays a list of browsers that currently supports PWA usage and its features.

Figure 8. Browsers that supports PWA.




The numbers in the picture above are the browser versions that support using PWA and its features as can be found on caniuse.com.

# 6 Conclusion and Future Work

## 6.1 Conclusion

This thesis focused on the design and implementation of a Progressive Web Application (PWA) specifically for ordering taxi services. The work demonstrated the effectiveness of PWAs in streamlining the taxi booking process and providing a user-friendly experience.

The results of this research hold significant importance as they showcase the potential of PWAs as a modern and cross-platform solution for efficient taxi booking. The findings highlight the enhanced accessibility, convenience, and efficiency that PWAs can offer to both users and taxi service providers.

While the results demonstrate the advantages of PWAs, it is important to acknowledge certain limitations. Factors such as network connectivity and device capabilities may impact the performance and functionality of the PWA. However, the benefits of cross-platform compatibility, offline functionality, and improved user experience outweigh these limitations.

The results of this research can be applied in the transportation industry, specifically for taxi service providers aiming to enhance their booking systems. The developed PWA can be implemented to offer a seamless and convenient taxi booking experience to users across various platforms and devices.

## 6.2 Future Work

Further development of this work can be pursued in several directions. It could involve expanding the features of the PWA to include additional services like fare

estimation, driver ratings, and advanced booking options. Additionally, integrating with other transportation services or exploring partnerships with taxi companies could enhance the reach and effectiveness of the PWA.

There are several areas for future work to improve the PWA for taxi order fulfillment. One area of focus could be on improving the app's performance on slow or unreliable internet connections, such as through the use of additional caching strategies. Another area of focus could be on improving the app's accessibility, such as through the use of additional accessibility features for users with disabilities. Finally, continuous user feedback and iterative improvements can also contribute to the ongoing development and refinement of the PWA for ordering taxi services.

# REFERENCES

Brossel et al., 2023. Overview of Progressive Web Apps (PWAs). Microsoft Documentation Articles. Available at <https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/> [Accessed May, 2023]

Jhala, D. 2021. A Study on Progressive Web Apps as a Unifier for Native Apps and the Web. International Journal of Engineering Research & Technology, Vol. 10: 5. Pp. 207-210. ISSN (Online) 2278-0181. Available at <https://www.ijert.org/research/a-study-on-progressive-web-apps-as-a-unifier-for-native-apps-and-the-web-IJERTV10IS050139.pdf> [Accessed April, 2023]

Kumar, B NJ. 2018. Progressive web Apps a New way to Experience mobile. International Journal of Engineering Research & Technology, Vol. 4: 22. Pp. 1-2. ISSN (Online) 2278-0181. Available at <https://www.ijert.org/research/progressive-web-apps-a-new-way-to-experience-mobile-IJERTCONV4IS22069.pdf> [Accessed May, 2023]

Sayali, T. & Abhishek J. 2018. Impact of Progressive Web Apps on Web App Development. International Journal of Innovative Research in Science, Engineering and Technology. [Online Journal]. Vol. 7:9. Pp. 9439-9444. ISSN (Online) 2319-8753. Available at: DOI:10.15680/IJIRSET.2018.0709021. [Accessed May, 2023]

Sam, R. & Pete, L. 2020. What are Progressive Web Apps? Progressive Web Apps; Websites that took all the right vitamins. [Online article]. Available at: <https://web.dev/what-are-pwas/>. [Accessed May, 2023]

Fourault S. 2022. How Progressive Web Apps can drive business success. Progressive Web Apps; Websites that took all the right vitamins. [Online article]. Available at: <https://web.dev/drive-business-success/>. [Accessed May, 2023]

Biorn-Hansen, A., Majchrzak T. & Gronli T. 2018. Progressive Web Apps for the Unified Development of Mobile Applications. [Online article]. Available at <https://www.researchgate.net/publication/325823248_Progressive_Web_Apps_for_the_Unified_Development_of_Mobile_Applications>. [Accessed May, 2023]

[1] Microsoft (2023). Progressive Web App: Overview of Progressive Web Apps (PWAs). Retrieved from: https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/. [Accessed May, 2023]

[2] Adobe Figma Inc. (2016). Figma Documentation. Retrieved from https://www.figma.com/developers/api. [Accessed May, 2023]

[3] Refsnes Data AS (1998). W3Schools: what is bootstrap? Retrieved from https://www.w3schools.com/whatis/whatis_bootstrap.asp [Accessed May, 2023]

[4] Facebook, Inc. (2013). React Documentation. Retrieved from https://react.dev/learn [Accessed May, 2023]

[5] Mozilla Inc. (2023). MDN Web Docs: Routes and Controllers. Retrieved from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes. [Accessed May, 2023]

# APPENDIX 1

Prototype of the app pages designed using Figma.