Technical Solution Document

Author- Alexander Adu-Sarkodie

MSc Telematics MSc Engineering

LinkedIn - <https://www.linkedin.com/in/alexander-adu-sarkodie-063b4b4a/> GitHub - https://github.com/kukuu?tab=repositories

Progressive Web App

These are further steps with capability of enhancing the application’s usability and performance.

Performance

*Progressive Web App Performance*

Progressive Web Apps are judged by several performance standards, including Time to Interactive and First Meaningful Paint.

Time to Interactive (TtI) - This audit identifies the time at which a page appears to be ready enough that a user can interact with it.

First Meaningful Paint (FMP) - First Meaningful Paint is essentially the paint after which the biggest above-the-fold layout change has happened, and web fonts have loaded.

*Tools*

Capacitor: Native Progressive Web Apps

Capacitor is a cross-platform app runtime that makes it easy to build web apps that run natively on iOS, Android, Electron, and the web. We call these apps "Native Progressive Web Apps" and they represent the next evolution beyond Hybrid apps.

Capacitor provides a consistent, web-focused set of APIs that enable a web app to stay as close to web-standards as possible, while accessing rich native device features on platforms that support them. Adding native functionality is easy with a simple Plugin API for Swift on iOS, Java on Android, and JavaScript for the web.

*Running Natively and on the Web*

One of the key features of Capacitor is the ability to build one app that runs both natively (in the app stores), and on the web. Capacitor does this by providing a layer between the underlying platform and the APIs/Plugins you'd like to use.

If your app makes native plugin calls that don't have a web substitute, such as SplashScreen.show(), the app will allow those calls without crashing. Calls that return a promise will return a rejected promise, which you should be handling in your app anyways.

Additionally, Capacitor's JavaScript API has a number of utilities that make it possible to programmatically check whether certain APIs are available. It will then handle gracefully negative responses (promises) without the application breaking. This will be done programmatically.

For example, if your app would normally rely on the Camera app being used to take a photo, you could check if the Camera is available, and if not, ask the user to upload a file instead: