International Journal of Innovations in Engineering and Technology (IJIET)
http://dx.doi.org/10.21172/ijiet.101.13

Meridian : Multi-featured Android app serving an Offline first Progressive Web application

Jalaj Maheshwari[1], Roshani Kavishwar[2], Osunam Panggam[3], Komal Deoda[4]
[1,2,3,4] *Department of Computer Science and IT, Veermata Jijabai Technological Institute, Mumbai, India*

Abstract-- Nowadays the weather forecasting is available as a widget or application on our smartphones. But often, it becomes difficult to find an application which provides a centralized features of Weather forecasting which works well for low internet connectivity, Calamity / Disaster Updation, Air Quality Index and Elevation Reporting. However, the proposed solution assures to provide all these functionalities in a single android application which provides an extension to a Progressive Web App hosted on Firebase. The Progressive Web App will fulfill the objective of weather forecasting & air quality index reporting in an efficient manner using Service Workers whereas the android application will provide notifications and detailed updates regarding any Calamity and predicted alerts. In addition, a user can also check the Elevation of current / customised user location with reference to the Mean Sea level and can also make use of chat feature to proceed with a quick text conversation with its peers.

Keywords-- Progressive Web App, Service Workers, Firebase, Weather forecast, Android App, Air Quality Index , Mean Sea Level.

I. INTRODUCTION

In today's era, web and mobile applications are two of the most prevalent domains that any user can rely on. But both of these areas comes with some shortcomings in their own ways and sometimes these limitations causes end user to pay the price in the form of either poor user experience or less reliable applications. Attempts have been made in the past to develop systems that can provide Weather, Air quality, Natural calamity notification and predictions but no approach had been taken to bring together all these phenomenon under a single hood and provide these services with low internet connectivity support. This project typically strives to resurrect such a loophole. The following elaborates the key concepts involved:

A. Android Application

An Android app is a software application running on the Android platform. Because the Android platform is built for mobile devices, a typical Android app is designed for a smartphone or a tablet PC running on the Android OS [1]. Android apps are mostly written in the Java programming language and use Java core libraries. They are first compiled to Dalvik executables to run on the Dalvik virtual machine, which is a virtual machine specially designed for mobile devices.

B. Progressive Web Application

*Pro*gressive Web Apps are user experiences that have the reach of the web, and are [2]:

- *Reliable* - Load instantly, even in uncertain network conditions. When launched from the user's home screen, service workers enable a Progressive Web App to load instantly, regardless of the network state. A service worker, written in JavaScript, is like a client-side proxy and puts you in control of the cache and how to respond to resource requests. By pre-caching key resources you can eliminate the dependence on the network, ensuring an instant and reliable experience for your users.
- *Fast* - Respond quickly to user interactions with silky smooth animations and no janky scrolling. 53% of users will abandon a site if it takes longer than 3 seconds to load! And once loaded, users expect them to be fast—no janky scrolling or slow-to-respond interfaces.
- *Engaging* - Feel like a natural app on the device, with an immersive user experience. Progressive Web Apps are installable and live on the user's home screen, without the need for an app store. They offer an immersive full screen experience with help from a web app manifest file and can even re-engage users with web push notifications. The Web App Manifest allows you to control how your app appears and how it's launched. You can specify home screen icons, the page to load when the app is launched, screen orientation, and even whether or not to show the browser chrome. This new level of quality allows Progressive Web Apps to earn a place on the user's home screen.

C. Service Workers

A service worker is a script that your browser runs in the background, separate from a web page, opening the door to features that don't need a web page or user interaction [3]. Today, they already include features like push notifications and background sync. In the future, service workers might support other things like periodic sync or geofencing. They allow developers to support offline experiences, giving them complete control over the experience.

D. Firebase

Firebase is a combination of Platform and Backend as a Service provided by Google Inc to help developers build, manage and deploy their applications using cloud solutions. Users can create databases, provide user profiles to firebase, host their applications and also perform data and usage analytics of their applications. Precisely, firebase facilitates the developers to focus more on the user experience and frontend of their application by providing the key backend and platform oriented services.

E. Weather Forecasting

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time [4]. Human beings have attempted to predict the weather informally for millennia and formally since the 19th century. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere at a given place and using meteorology to project how the atmosphere will change.

F. Natural Disaster

A natural disaster is a major adverse event resulting from natural processes of the Earth; examples include floods, hurricanes, tornadoes, volcanic eruptions, earthquakes, tsunamis, and other geologic processes [5]. A natural disaster can cause loss of life or property damage, and typically leaves some economic damage in its wake, the severity of which depends on the affected population resilience, or ability to recover and also on the infrastructure available.

G. Air Quality Index

An air quality index (AQI) is a number used by government agencies to communicate to the public how polluted the air currently is or how polluted it is forecast to become [6]. As the AQI increases, an increasingly large percentage of the population is likely to experience increasingly severe adverse health effects.

H. Mean Sea Level

Mean sea level (MSL) (often shortened to sea level) is an average level of the surface of one or more of Earth's oceans from which heights such as elevations may be measured [7]. MSL is a type of vertical datum – a standardised geodetic reference point – that is used, for example, as a chart datum in cartography and marine navigation, or, in aviation, as the standard sea level at which atmospheric pressure is measured to calibrate altitude and, consequently, aircraft flight levels. A common and relatively straightforward mean sea-level standard is the midpoint between a mean low and mean high tide at a particular location.

II. RELATED WORK

Related work for each of the proposed system module is addressed as part of research and understanding. AQI results the index of maximum concentration of a pollutant in air. Research work in this field is varied. Web-based AQI system, which is designed for online calculation and display of nation-wide AQI [8]. *InAir* relies on stationary gas sensors placed inside users' homes, that display on each sensor node visualizes the air quality values measured in the room and the values acquired by the other nodes in the other parts of the house. *MAQS*, considers a mobile wearable sensing system that provides air quality information for each room visited by the user. *CitiSense* and *Common Sense*, both these solutions rely on small, battery-powered sensor nodes that measure the concentrations of polluting gases and send air quality data to users' smartphones through Bluetooth. *uSense*, a user can monitor the air quality near her/his house, just by placing a small sensor node in her/his property. The obtained data are sent to the uSense database via the Internet, and are made accessible to all the other uSense users. Existing APIs include Breezometer and Real Time Quality API.

The Natural disaster management consists of four fundamental steps [9]. The steps are mitigation, preparedness, response, and recovery. Work in this area reflects idea of an android-based mobile application that provides early disaster warning and evacuation progress of the upcoming disaster by using Google Map, Disaster Management Server (DMS) as a third-party server, which stores disaster prone areas and the details about the users in its database. Application has two major parts:

- *Early Disaster Warning:* To notify automatic or manual risk messages to the user.
- *Tracking Evacuee User*: To display shortest path to nearest safe shelter via Google map.

In elevation report, Sea level rise or Land elevation poses greater impacts on the coastal environments. The present research methodology in case of elevation, can be analysed to have two parts [10]. The first part considered the trends of sea level rise through the available sea level data, via Permanent Service for the Mean Sea Level (PSMSL) which contains annual mean values of sea level from almost 2000 tide gauge stations around the world. After collecting the data fitted by 5th order polynomial, the regression coefficient is calculated for the analysis of future sea level trend. Another part of methodology involves to the analyses of sea level rise effects by using Shuttle Radar Topographic Mission (SRTM) Global Digital Elevation Model (DEM) and the satellite images.

Progressive Web App (PWA) is known for its ability to work on low network at an impressive speed. It has two function: Application Shell Architecture and Service Workers [11]. Application Shell Architecture empowers user interface with HTML, CSS and Javascript. It is responsible for initial load and is cached by Service Worker to avoid re-fetch during repeated visits. Service Workers make use of APIs, and are responsible to serve the pages requesting an event and push the notifications back to the server. Service Workers run in background and help in caching and delivering content.

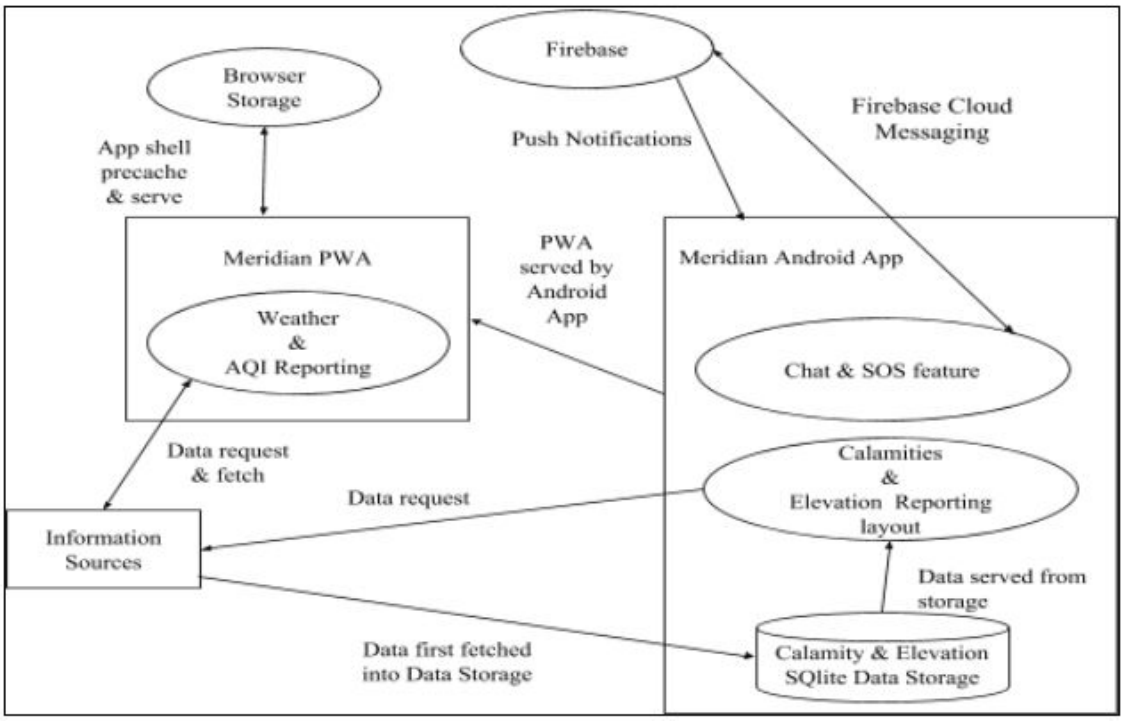


Figure 1. System Architecture of Meridian

III. PROPOSED SYSTEM ARCHITECTURE

The System Architecture (Figure 1.) gives a pictorial representation of how data is fetched, stored and transferred across different modules of Meridian. It also puts some limelight on features provided by Meridian.

The following provides a key description of System Architecture:

1. Meridian is divided into 3 modules namely
   i. Weather & Air Quality reporting
   ii. Natural Calamities Reporting
   iii. Elevation Reporting
2. Weather & Air Quality Reporting will be hosted on Google Firebase as a Progressive Web Application (PWA) and will be accessible for initial visit using Meridian android app as it will be present on the android app's home screen as one of the primary option, clicking on which the PWA will open in the default browser of user's mobile. Users can later directly access the progressive web app using its 'Add to Homescreen' feature.
3. Natural Calamities and Elevation Reporting will be present as a native feature inside the Meridian Android app.

4. Weather and Air quality details for a user preferred location will be fetched as a RSS feed from respective information sources and will be then displayed in Meridian PWA.
5. User preferred locations will be stored in browser storage using Localforage API and these preferences can be later used to auto-populate the PWA with Weather and AQI details on subsequent user visits.
6. The key advantage of using a Progressive Web App is the ability to pre-cache the app shell architecture in the browser storage on initial user visit using Service Workers and then serving the app from browser storage on future visit.
7. The Natural Calamities and elevation details are requested from the android app and these details are fetched as RSS feed from respective information sources and are first stored in a SQlite database created on the mobile device by Meridian App. These details are then fetched from the SQlite database and displayed to the user in the android app. The main motive behind storing the information into the device storage is to facilitate the user with last updated details even if the user goes into offline mode. The corresponding details are fetched from information sources at regular intervals of time and persisted accordingly.
8. Firebase Cloud Messaging is used to implement the Peer to peer chat feature and SOS emergency functionality, provided by the Meridian Android App. The SOS feature let's user send its location coordinates to preferred users at any instant. Individual login details of a user are also stored to Firebase for facilitating above services.
9. Instant news notifications relevant to Weather, AQI, Natural Calamities and Elevation reporting domain are sent to the Mobile or Web Interface of a user using Push Notifications.

IV. IMPLEMENTATION METHODOLOGY

Meridian focuses more on efficient and timely updates to its user with Cached Technology and Greater User Experience. Certain Steps are involved to create Meridian.We focus on the steps as below:

A. Steps for Building a Progressive Web App

1. Initializing a web Serve - Chrome web server that supports PWA used for development purpose to simulate the server in a real time environment.
2. Building the App Shell - The app's shell is the HTML, CSS, and JavaScript required to power the user interface of a progressive web app and is one of the components that ensures reliably good performance. Its first load should be extremely quick and immediately cached. Every subsequent time that the user opens the app, the shell files are loaded from the local device's cache, which results in blazing-fast startup times.
3. Implementing the Service Worker to pre-cache the App Shell - Progressive Web Apps have to be fast, and installable, which work online, offline, and on intermittent, slow connections. To achieve this, app shell is cached using service worker, which enables it to be always available quickly and reliably, i.e., all the UI and the scripts will be pre-cached in the local app. The updates are maintained, because every time the service worker checks for any changes between the current app UI and then matches it with that on server.
4. Deploy the PWA to a secure server - Meridian is then deployed it to Google Firebase, which is a remote cloud-like platform which allows to remotely manage applications, perform analytics, provide notifications and can provide version updates.

B. Steps for using Web API into an Android / Web Application

1. Deciding the Web API and information source.
2. Making the layout in android / web app for respective feature.
3. Getting API key for a particular API which we intend to use.
4. Setting up AsyncTask in Main activity to fetch and manage the information in case of android app and setting up the REST service for fetching the response in case of web app.
5. Displaying the results on the UI.

V. OFFLINE CACHING AND FETCH ANALYSIS

A. Context:

In online mode, the app fetches the Weather and AQI details for a user preferred location and displays it to the user. The analysis describes the process of Weather & AQI information caching and fetching from browser storage which forms the basis for PWA to work in an offline mode. The below analysis illustrates the use of key life cycle states of a service worker (viz Install, Activate & Fetch), role of service worker in caching and providing the resources and the entire process of app shell caching and serving it to the user under different conditions and will be used each time a User visits the PWA.

SW-precache is a tool used during the development phase to create a basic version of service worker file and calculate the hash values of app shell files including the service worker file itself. The hash values thus generated are used to detect any future changes in any of these files. The developer modifies the basic version of service worker to modify the caching and fetch logic for app data and appshell files as required.

B. Analysis:

If (PwaVisitCount = 1) ^ (Browser B.serviceWorkerSupport = True) ^ (App.netConnectivityMode = Online) then do the following:

1. Register Service Worker
2. Install Service Worker in Browser
3. Open Browser Cache C
4. Cache all the App Shell files
5. Cache all the Weather ^ AQI information using a key value pair pattern.
6. Close the Cache
7. Activate Service Worker S

Else if PwaVisitCount > 1, do the following while (S.state = Activated ^ B.serviceWorkerSupport = True) :

1. If (newServiceWorkerVersionDetectedFlag = True) ^ (newServiceWorker.hashValue != currentServiceWorker.hashValue) ^ (App.netConnectivityMode = online) then do the following:
    a. Register ^ Install the new Service Worker S file.
    b. Transfer the control from old S to new S
    c. Activate the new S
    d. Unregister the old S
    e. For each of the cached appshell file, do the following to compare the newly calculated hash values of all the appshell files calculated & saved in ServiceWorker file with values of those already cached to determine the updated files:
        If (cachedFile.hashValue != File.newHashValue) then
            i. Cache the new version.
            ii. Delete the old version of file from cache.
2. If (App.netConnectivityMode = Offline) then do :
    a. To serve the app, fetch all the app shell files via URI requests from cache that are otherwise meant to be fetched from remote server through network.
    b. For each of the user preferred location saved in browser storage, fetch the relevant information from the browser cache in the form of json response ^ Display it to the users.
    c. If (newLocationAddedFlag = True) then add the location in userPreferredLocations array inside browser storage whose details for same will be fetched when user accesses the app in online mode.
3. If (App.netConnectivityMode = Online) then do :
    a. Fetch all the app requests for appShell files using Service Worker.
    b. For each of the file URI requests, scan the browser cache as follows:
    If (cacheUriKeys.has(fileRequestUri) then return the file as response.
    c. For each location stored in browser Storage in offline mode, do the following:
        i. Fetch the latest details for location from Informations Sources as RSS feed in the form of json response.
        ii. Display the fetched information to user
        iii. Update the Cache with all the new values of Weather & AQI as follows:
            1. Set locationInformationUri.key = cacheUri
            2. Set locationInformationUri.value= responseFromInformationSource

A similar storage first strategy is used in case of information management of Natural Calamities and Elevation reporting to facilitate the user to get almost recent details displayed on the Meridian Android App screen.

C. Key Formulae:

1. The performance of cache memory is frequently measured in terms of quantity called Hit Ratio.

Hit Ratio = hit / (hit + miss) = no. of hits / total accesses

Here, hits denote the number of times URI requests made for the appshell files returns the required file. The only scenario for cache miss could be one when the user enters a new location in offline mode for which the weather details are not yet cached until that instance

2. Response time of a single appshell file contributing in making the user interface interactive, on repeat visits, in a traditional fully network dependent application is termed as follows [12] :

Response time without caching = Server Processing Time + (Page Size / Minimum Bandwidth) + Ideal Round Trip time × Turns + Network Latency + Client Processing Time

Here, Server Processing time includes searching for a desired file and creating the specific response to be sent to client and parsing the code on server in case of server side rendering.

Network latency involves delay in transfer of message packets from server to client & vice versa due to bandwidth restrictions & network strength fluctuations

Client processing time includes decoding the URI requests, computing time to display the web pages by fetching them from server response and additional time to parse the code in case of client side rendering.

The above response time can be reduced to significant levels in case of PWA during repeat visits, as the appshell files are cached on initial visit. Hence, the app does not need to make a request to fetch file from remote server on repeat visits, instead it can fetch it from cache memory of client itself thus eliminating the overhead of latency, round trip time and server processing time.

Hence the Response time for making a web page/ Mobile web page interactive using a caching strategy for a single file can be calculated as :

Response time with caching = Client Processing time taken to decode the URI request + time taken to fetch the appshell file from cache + time taken to render the file. ----------(A)

Thus, response time for loading the required files to facilitate the basic functionality of a progressive web app inorder to make the web app interactive to users can be deduced as:

Total response time = $\sum_{i=1}^{n}$ (response time to load an individual appshell file i under caching strategy)

reference : A)

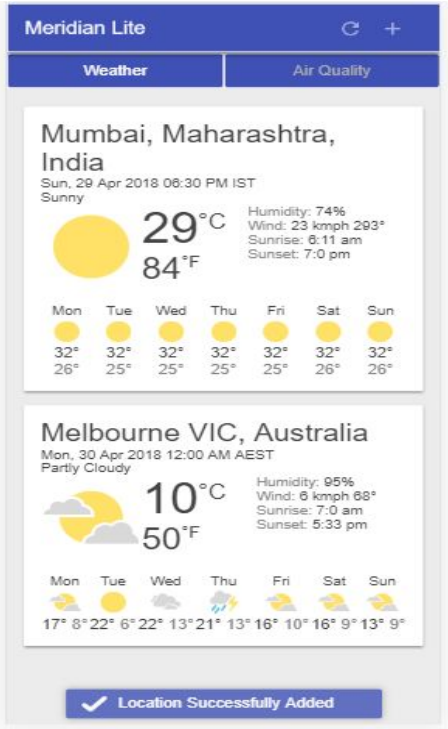
The above analysis is self sufficient to determine the rate at which caching can reduce the response time of an application because at each stage of file cache, the network latency, round trip times and server processing time can be made almost negligible. However, Service Worker caching may seem to similar to traditional browser caches, but in reality Service Workers caching technique have a great edge over traditional caching which are as follows:

1. Service workers can help browsers show cached data in offline mode.
2. We can control the way we want to update the Cache and can intercept the network requests to modify them to provide a response as per our business case.
3. Service workers can support push notifications.
4. Cache only those resources which are most important for basic functionality of app and then fetching the required resources as required on later stage thus reducing the initial response time.
5. Though traditional browsers can cache and showcase details in online mode but they still lack the benefit of reflecting the updated content to the user every time as :
   a. if max-age of cache is set to a predefined value then the browser may have to wait for an updated copy to be fetched from server till the max-age time elapses.
   b. If max-age of cache is set to 0, then the network latency may peek in as the app checks for an updated copy from remote server each time the fetch method is executed this incurring extra data charges & latency.

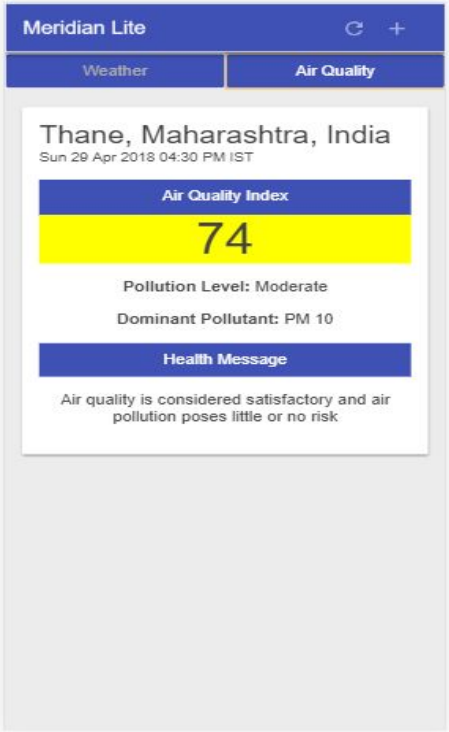
Service workers handle both these scenarios as they provide the user to push the updated versions of files to client and cache them until the next version becomes available thus eliminating the need to check for file updates and getting the most recent versions of app shell files at the same time thus proving it's worth over traditional browser caching technique.

VI. IMPLEMENTATION SNAPSHOTS

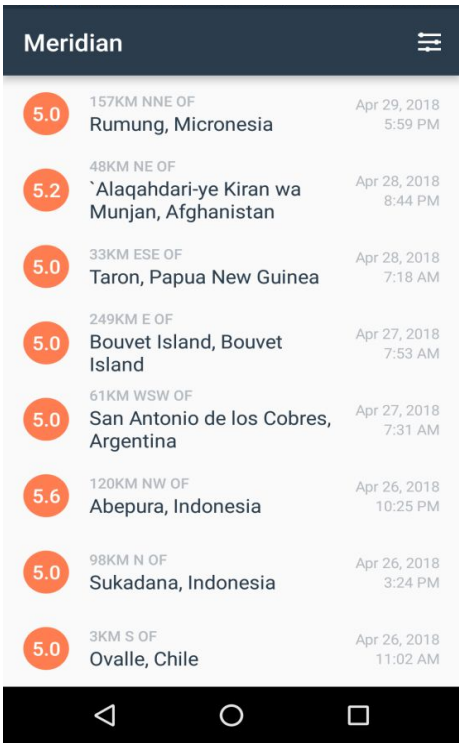
http://dx.doi.org/10.21172/ijiet.101.13



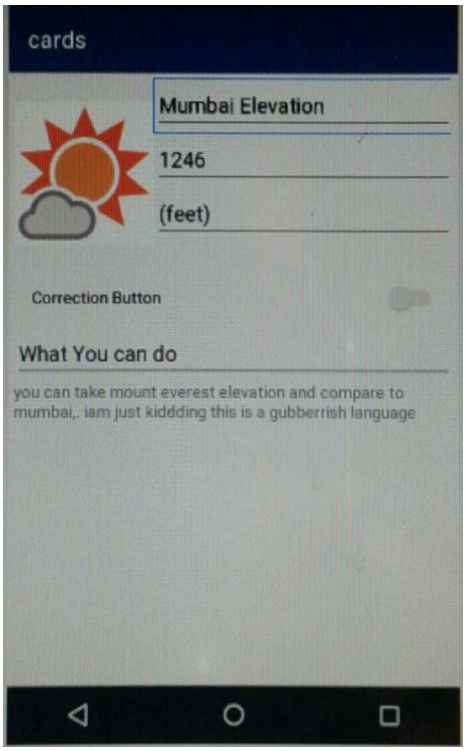
A PWA Mobile View: Weather Module



B. PWA Mobile View: Air Quality Module

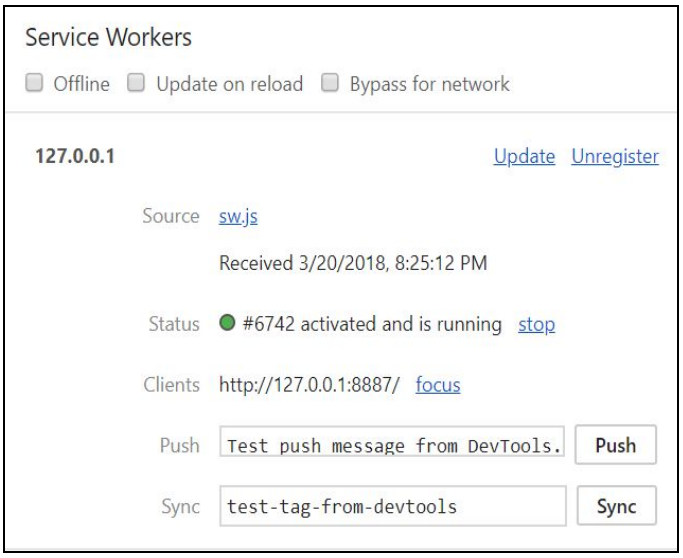


C. Android App View: Earthquake Module



D. Android App View: Sample Elevation Module

E. Live Service Worker of PWA visible in Browser Console



VII. ADVANTAGES

- Early Accessibility to Weather, Air Quality & Natural Calamity forecasts and Information
- Centralized informative application that can work on low to Offline connectivity modes
- Weather & Air Quality Progressive Web App provides a worth noting user experience of native mobile app.
- Inbuilt Peer chat feature in Meridian Android Application
- Flexible Push Notifications to keep users updated
- Cache support leading to low Internet data usage and faster data fetch timelines.
- Travel Partner to keep you informed about the weather, air quality,physical location elevation & Calamitic conditions of your locations
- Emergency SOS feature to family members, close acquaintances and any individual within a stipulated distance from the user.

VIII. APPLICATIONS

- Extension to Meteorological forecast & notification domain
- Can assist Tourism Industry by providing location specific weather information to tourists and their safety using SOS module.
- Alternative to applications that require high internet bandwidth and connectivity for providing similar functionalities thus making application usage possible in rural and low connectivity regions.
- Location specific Elevation reporting that can facilitate the travel routes and engine requirements decision during journey of heavy vehicles and locomotives.
- Pollution Regulatory and Healthcare Domain

IX. Conclusion

People who travel often requires the need of an efficient application that can work on fluctuating networks and at the same time gives information about their current location with reference to weather forecast and air quality details, natural calamity notification and details and sometime elevation level too. For such people, Meridian can prove a one stop destination. Moreover Meridian not only fulfills the basic expectations but also tries to meet the some of need-of-the-hour features such as peer chat, SOS from both quick conversations and safety point of view. The combination of Progressive web app and the android app of Meridian has tried to achieve minimum disk storage and support to work in low internet connectivity regions. The project will also prioritize, in future releases, to make an AI approach in order to further simplify things for its potential users.

X. ACKNOWLEDGMENT

The authors gratefully acknowledge the contributions of Prof. S. S. Suratkar, Department of Computer Science and IT, Veermata Jijabai Technological Institute, Mumbai, India for her inputs during writing this document.

XI. REFERENCES

[1] Android App [Online]. Available: https://www.techopedia.com/definition/25099/android-app.
[2] Progressive Web App: Open Source Google Developer. [Online]. Available: https//developers.google.com/web/progressive-web-apps/
[3] Service Workers: An Introduction. [Online]. Available: https://developers.google.com/web/fundamentals/primers/service-workers/
[4] Weather forecasting [Online]. Available : https://en.wikipedia.org/wiki/Weather_forecasting
[5] Natural Disaster [Online]. Available : https://en.wikipedia.org/wiki/Natural_disaster
[6] Air Quality Index. [Online]. Available: https://en.wikipedia.org/wiki/Air_quality_index
[7] Mean Sea Level. [Online]. Available: https://en.wikipedia.org/wiki/Sea_level
[8] Dr. A.K.Agrawal, S. Mahajan, S. Parihar and R. Singh, "National Air Quality Index", Dr. A.B Akolkar, CPCB: Public Journal, pp. 5-32, October 2014. [Online]. Available: http://www.indiaenvironmentportal.org.i/files/file /Air%20Index.pdf . [Accessed December. 11, 2017].
[9] Amit Gosavi and S.S. Vishnu, "Disaster Alert and Notification System Via Android Mobile Phone by Using Google MAp", International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 11, pp. 2-5, November 2014. [Online]. Available: https://pdfs.semanticscholar.org/a90d/20867148ebc26cbc569022ed591f3bc9d7a.pdf . [Accessed December. 11, 2017].
[10] M.K. Pramanik, S.S. Biswas and T. Mukherjee, "Journal of Remote Sensing & GIS", Pramanik et al: Research Article, pp. 2-6, 2015. [Online].Available:https://www.omicsonline.org/open-acess/sea-level-rise-and-coastal-vulnerability-along-the-eastern-coast-of-indiathrough-geospatial-technologies-2169-0049-1000145.pdf . [Accessed December. 10, 2017].
[11] R.S. Mishra, "Progressive WEBAPP: Review, "International Research Journal of Engineering and Technology, Volume 3, No. 6, pp. 2-3, June 2016. [Online]. Available: https://www.irjet.net/archives/V3/ie/IRJET-V3I6568.pdf . [Accessed March. 12, 2018].
[12] Web Page Response Time [Online]. Available: http://ericgoldsmith.com/wp-content/uploads/2009/02/web_page_response_time_101.pdf