Elsevier Editorial System(tm) for Journal of Biomedical Informatics

Manuscript Draft

Manuscript Number: JBI-19-181

Title: A Fast Prototyping Approach for Cross-Platform mHealth Applications: A Case Study for Endocrine Hormonal Therapy Adherence

Article Type: Research paper

Keywords: Smartphones; mobile devices; mobile computing; cross-platform; mobile health (mHealth); applications (apps)

Corresponding Author: Mrs. Devasena Inupakutika,

Corresponding Author's Institution: University of Texas at San Antonio

First Author: Devasena Inupakutika, MS

Order of Authors: Devasena Inupakutika, MS; Sahak Kaghyan, PhD; David Akopian, PhD; Patricia Chalela, DrPH; Amelie G Ramirez, DrPH

Abstract: With the advent of cross-platform development frameworks, mobile Health (mHealth) is bridging the gap between patients, health professionals and healthcare. These frameworks are constantly developing in terms of sharing the significant portion of the application (app) codebase and their ability to leverage device hardware features based on the app's requirements. There exists a need in fast prototyping of apps to reduce deployment cycles. This paper explores such a design of device independent native mobile and web app prototyping approaches, which harness existing Google resources. However, it is known that these frameworks come with a performance penalty that is often critical for mHealth apps. The paper assesses comparative performance figures in various modes. The approach is analyzed for practical case study app developed for medication adherence management of breast cancer patients undergoing the Endocrine Hormone Therapy (EHT). The app is built on cloud computing technologies that provide a portable, cost-effective and convenient monitoring environment. Experimental results are presented that verify the efficacy of the proposed prototyping approaches as compared with the other state-of-the-art performance parameters for the app development. The performance analysis methodology could form the out-set for developers in order to assess their apps.

Suggested Reviewers:

Opposed Reviewers:

Cover Letter

February 8, 2019,
Editorial Department of Journal of Biomedical Informatics

Dear Editor-in-chief,
I am submitting an original manuscript for consideration of publication in the Elsevier Journal of Biomedical Informatics. The manuscript is entitled "A Fast Prototyping Approach for Cross-Platform mHealth Applications: A Case Study for Endocrine Hormonal Therapy Adherence".

It has not been published elsewhere and that it has not been submitted simultaneously for publication elsewhere.

We proposed and demonstrated a fast prototyping approach for cross-platform mHealth applications. Our proposed approach can be followed relatively easily in the existing mHealth apps, solving the interoperability and portability challenges faced by healthcare/ digital health today, improving the performance aspects, some aspects of the User Interface and more specifically shorter app deployment times. With this approach, we accomplished three things: Firstly, development of native mobile and cross-platform web apps for providing necessary care to the patients. Secondly, the approach also preserves trustworthiness by ensuring that the patients know who is receiving their health data and for what purposes it will be used. Thirdly, the apps support all platforms and would be easy to work with any stakeholder in the healthcare community.

The corresponding author is: Mrs. Devasena Inupakutika
(email: mmn609@my.utsa.edu).

Thank you very much for your consideration. We are looking forward to hearing from you.

Yours Sincerely,

Devasena Inupakutika
University of Texas at San Antonio
One UTSA Circle, San Antonio, TX, 78249, USA

A Fast Prototyping Approach for Cross-Platform mHealth Applications: A Case Study for Endocrine Hormonal Therapy Adherence

Devasena Inupakutika[1], Sahak Kaghyan[1], David Akopian[1], Patricia Chalela[2], Amelie G. Ramirez[2]

***Abstract*— With the advent of cross-platform development frameworks, mobile Health (mHealth) is bridging the gap between patients, health professionals and healthcare. These frameworks are constantly developing in terms of sharing the significant portion of the application (app) codebase and their ability to leverage device hardware features based on the app's requirements. There exists a need in fast prototyping of apps to reduce deployment cycles. This paper explores such a design of device independent native mobile and web app prototyping approaches, which harness existing Google resources. However, it is known that these frameworks come with a performance penalty that is often critical for mHealth apps. The paper assesses comparative performance figures in various modes. The approach is analyzed for practical case study app developed for medication adherence management of breast cancer patients undergoing the Endocrine Hormone Therapy (EHT). The app is built on cloud computing technologies that provide a portable, cost-effective and convenient monitoring environment. Experimental results are presented that verify the efficacy of the proposed prototyping approaches as compared with the other state-of-the-art performance parameters for the app development. The performance analysis methodology could form the out-set for developers in order to assess their apps.**

***Index Terms*—Smartphones, mobile devices, mobile computing, cross-platform, mobile health (mHealth), applications (apps)**

I. Introduction

OVER the past decade, mobile technology ownership has been tremendously increasing. Ubiquitous connectivity is opening new possibilities for the healthcare professionals. Technology ecosystems integrating mobile devices, cloud computing and wireless communication technologies provide mediums for the deployment of mobile Health (mHealth) applications and the broader penetration of health informatics concepts [3]. mHealth is changing the way of healthcare delivery today due to almost global mobile coverage. The large-scale adoption of applications depends on their usefulness, effectiveness and entertaining features to engage their users for prolonged times.

This paper discusses a fast-prototyping cross-platform mHealth app development approach, which exploits Google resources and evaluates performance figures. A hormonal therapy (HT) Patient Helper prototype app is used as a case study, which includes integration of new generation of mobile and cloud computing and messaging solutions for healthcare. We also consider mHealth design, accessibility, usability, and user experience from patient's, healthcare professional's (Patient Navigators (PN)) and application developer's perspectives. HT Patient Helper mHealth platform is designed to incorporate most of the aspects of the Endocrine HT (EHT) for patients. The backend services provided by Google Firebase called Firebase Cloud Messaging (FCM) are used to monitor and send messages and notifications to the user. Key features include symptoms monitoring, educational content and videos, peer modelling closed group that helps patients learn about the symptoms and available resources, understand the intensity of the symptom, improve and gain coping skills to manage the level of tension and build a network of support to enervate the fear of recurrence.

Since the late 1970s [4], breast cancer has been on the rise and becoming a common threat to women's physical and mental health. According to the research in [2], about 296,980 women in the United States are diagnosed with breast cancer each year and approximately 75% of them have hormone receptor-positive breast cancers. Thus, the prevention and treatment of breast cancer are among important and urgent tasks. EHT is an effective and appropriate for nearly all women who have hormone receptor-positive tumors. Patients who used EHT for about 5 years have shown 50% better results in survival rates and reduction in recurrence [2]. The case study application aims at promoting patients adherence to their therapy and prescriptions for ensuring successful treatment.

There exists a portability challenge in the adoption of mobile technologies due to diverse spectrum of devices. Similar research [49] that presents the assessment of a rigorously designed, low cost custom exergaming mHealth platform that utilizes contemporary controller hardware off-the-shelf assesses the intervention impacts as well as mHealth

Manuscript submitted . The study is funded by Susan G. Komen (Award No. SAB160005), the Mays Cancer Center (Grant No. P30 CA054174) and Redes En Acción (Grant No. U54 CA153511).

[1]D. Inupakutika, S. Kaghyan and D. Akopian are with the Department of Electrical and Computer Engineering, The University of Texas at San Antonio, TX 78249 USA (e-mail: devasena.prasad@gmail.com; sahak.kaghyan@gmail.com; david.akopian@utsa.edu).

[2]P. Chalela and A. G. Ramirez are with the Institute for Health Promotion Research, University of Texas Health San Antonio, San Antonio, TX 78229 USA (e-mail: chalela@uthscsa.edu; ramirezag@uthscsa.edu).

platform's usability for long-term adherence. Concerned about the operability of applications across multiple platforms in heterogeneous mobile devices, one can explore the solutions for fast prototyping of cross-platform applications for such fragmented mobile landscapes and its possibilities. Since smartphones are easily accessible by 80 percent of the mobile phone users [4], one of the approaches to this problem is to design an Android application and then convert it to a portable mobile web service. This method exploits rich existing resources and development tools from Google.

Our case study application employs this approach, is simple to navigate, have access to basic as well as necessary chronic-condition dependent information that is important to patients, social media access and better user interactivity. It includes scale-driven functionality for symptoms tracking, some statistics in the form of weekly symptom summary graphs, tips and information to manage common symptoms and promote positive health behaviors. Google's FCM is used to send notifications to patients. The app is linked to user's specific Google account. All the patients enrolled in the adherence program are monitored by the PN. Facebook is integrated in the app for peer modeling intervention and building a network of support. The same application with all the features is further converted to a cross-platform mobile web service for broader deployment.

The structure of this paper is as follows. In Section II, we review related works on mHealth apps and some challenges in their development. In Section III, we introduce the idea of our system architecture on the basis of the native and web apps architectures considering HT Patient Helper as a case study prototype app. In Section IV, we describe the implementation methodology of the proposed prototyping approach. In Section V, we discuss the performance metrics of the developed native and web apps and analyze the differences of both. In Section VI, we summarize our analyses based on differences between the two types of apps using web services. In Section VII, we conclude the work with the future viewpoint.

II. Background

The major challenges of chronic diseases treatments include regular and continuous patient-physician communication, lack of adherence, potential side effects, and impact on quality of life and other reported outcomes by patients. Over the last years, these challenges are being overcome and have been providing new opportunities [9] [10] [11] by the increase in the number of mHealth solutions.

mHealth is an emerging intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies so that the healthcare is improved globally using the information and communication technology [12]. mHealth solutions also have the potential to support clinical decision making through the continuous symptoms tracking. However, mHealth in cancer therapy management [13] is underused for several reasons such as lack of relevant experimental and scientific validations, compliance-related, regulatory, organizational and technological questions.

A. Existing mHealth applications and technologies

The mHealth technologies over the last few years have given rise to the development of significant number of platforms that offer various opportunities for healthcare providers to deliver high-quality care remotely. Researchers and practitioners have been developing mHealth related behavioral as well as technical protocols for different kinds of cancer patients targeting diverse communities (e.g. underserved and rural) as well as different age groups. As discussed in [14] there is an expected increase in the number of smartphone owners using mobile medical apps and it is expected to increase to 3.4 billion by 2018. It is stated that nearly 85% of physicians are already adopting smartphones and medical apps to remotely monitor their patients' chronic conditions. Currently, the application related to this technology that are in use include access to patients health records via web portals for providers and doctors and patient communication, access to medical education via internet educational content, videos, disease-specific documentation and clinical trials for patients. [15] [16] The collection of patient-specific data related to their symptoms management and drug or therapy adherence and mHealth interventions already span social media to foster healthy living and lifestyle through peer modeling. Actual study [23] on cancer survivors for such apps, illustrate that the apps enable them to have close and personal relationships with PNs and immediate access to providers acted as safety element for possible emergencies and issues in their therapy and treatment. Many other studies have also demonstrated the effectiveness and potential of increase access to comprehensive cancer related care to patients spread over various geographical locations through remote consultations by PNs, doctors and healthcare providers, medication adherence and symptom management. Latest systematic review protocol [50] has also been released based on the detailed study conducted to identify and describe the various mHealth intervention strategies that are used for breast cancer. It also assessed the impact of such strategies on breast cancer awareness and screening.

From technology perspective, the extensive review of current state of art in mHealth services in [28] surveys the most significant research works in this field and presents a detailed analysis of the novel mHealth services and applications as proposed by the industry. It also represents the typical mHealth architecture that uses Internet and web services to enable authentic and pervasive patient-doctor interaction. The apps that were reviewed concern mostly in the healthy lifestyle/ prevention, finding a healthcare professional, filling prescriptions, compliance, diabetes, mental health, behavioral disorders, oncology, women's and children health etc. The three oncology apps are for breast cancer [29], chronic myeloid leukemia (CML) [30] and skin cancer patients [31] but not specific to those undergoing cancer therapies. This review also highlights the open issue of cooperation between mHealth apps that use same or different services to accomplish common objective. This could improve

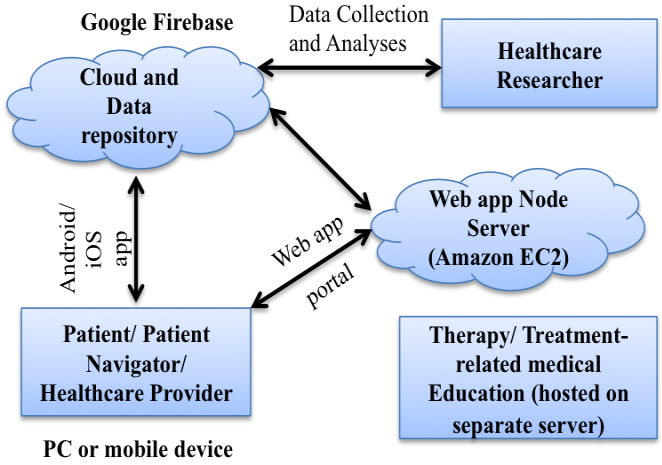

Fig. 1. High-level architecture for the proposed system

the performance and efficiency of the devices being used for accessing the apps. Another systematic literature review of mHealth apps [51] classifies smartphone-based healthcare technologies discussed specifically in academic literature. The study is conducted on a total of 84 apps according to their functionalities, six major Operating System (OS) platforms (Android, iOS, Windows, Blackberry and Palm OS), categories (disease diagnosis, drug reference, medical calculations, chronic disease self-management, remote monitoring etc.), advanced mobile communication and portable computation technologies that eventually effect the development and performance.

In the current work, we aim to explore the full potential of mHealth technology by exploiting a state-of-the-art mHealth architecture and Google Firebase as real-time backend database, developing native and web apps with API and providing access to broader information systems that ensures high degree of interoperability, performance as well as pervasiveness.

B. Challenges

With all these developments in the electronic tools for supporting patients during their cancer therapy and other chronic conditions [1] [11] [18] [19] [20] [21], the trend towards outpatient monitoring and treatment for chronic diseases leads to new challenges in terms of performance, efficiency, privacy, security, confidentiality and ownership of data for patients, PNs and doctors. There are also concerns related to the insecure data collection, transfer, integration to third-party apps like REDCap [25] that stores Electronic Health Records (EHR), cloud technology and storage. It is also required that the medical apps and mHealth technology is integrated to traditional health IT systems with the current paradigm shift. Applications Programming Interfaces (APIs) are the solution for the same. SMART onFHIR [26] and Epic EHR [27] are some of the relevant examples resolving interoperability in mHealth. However, they are for integrating with proprietary vendor systems.

Extensive engineering and interdisciplinary effort is directed in addressing performance and efficacy challenges to achieve more robust delivery and operation of mHealth systems and expand their coverage to as many smartphone OS users as possible. Serious efforts [52][53] in this direction for providing technology assisted solutions for engaging patients for maximizing long-term adherence to therapies are being put up. Another similar mHealth telecardiology system [54] that provides 24/7 health monitoring service called WE-CARE helps in real-time monitoring over mobile networks combining the user mobility and intelligent clinical function.

The proposed prototyping approaches aim to address these challenges by designing and developing native mobile and web apps exploiting open source technologies which can be accessed on any device. The apps are designed to be simple for anyone to use, even those with little to no prior computer experience. The level of efficiency and performance is improved and made possible in the proposed system by combining mobile device and the real-time backend server for faster request and response that is crucial for mHealth apps.

The implementation also reduces the learning curve that is required for successful development across diverse platforms for the developers. The web implementation also offers API and makes use of the existing REDCap API to fix the interoperability issues between mHealth and EHRs for data exchange. The experimental and measurement results indicating the technical, user-centric and performance parameters for each of the apps will be discussed in Section V.

III. Proposed System

Breast cancer is most common among women over age of 40, and they are especially affected by usability problems, for instance not being able to perform tasks of setting reminders for a doctor appointment or medication, on switching to a new device (e.g. smartphone) it takes time for them to get used to the new user interface. This population is at risk to be excluded from the benefits of using mHealth apps or services [36]. Usability issues are often caused by device platform restrictions, for instance application's fragmentation, availability of similar widgets, user interface designs and limitation in screen sizes. These restrictions force developers to build application code for specific target platform: Android, iOS etc., separately resulting in increased development cycles and resources. Eventually users have to adapt to varied user interfaces and navigation [37].

The proposed fast prototyping methodology of cross-platform mHealth apps for EHT therapy patients exploits Google Environment. The proposed system architecture can be applied to any mHealth app in general. The apps developed as part of this research use Google Firebase as database server, Amazon Elastic Cloud Compute (EC2) [33] instance as a Node.js [34] server for web app hosting and Github pages [35] for medical education content. The idea is to reduce the impact of fragmentation that mobile technologies suffer, which stems from the diversity in devices (ranging from tablet, to smartphone), operating system platforms (Android, iOS, Windows Phone, and Symbian), and diversity in user preferences and needs (i.e., accessibility). This approach for

multiplatform apps use development tools from Google, and a user-centric method for design validation.

A. System Architecture

Fig. 1 shows the high-level architecture of the proposed system. Google Firebase is used as the database server, which is the same across native mobile and web apps. The native android app can be accessed on android smartphones or tablets. Web application and portal is deployed using Node.js as the web server with the same functionality as the mobile app. The web app can be accessed on any mobile or fixed device (desktop computer) using its URL. Node server for web app is deployed using Amazon web services. Table I shows the specifications for Amazon EC2 instance. It is scalable and can be configured depending on the number of concurrent users accessing the app. The educational content for patients is deployed centrally using Github pages for medical educational management and to provide them access to broader information about the symptoms and EHT therapy itself. The information is open and hence, it is available for anyone to access and educate self. The idea is to provide access to PN or provider to be able to update the content easily without having to make any updates to the apps themselves whenever the educational content is changed. Firebase database for both native and web apps are the same, and its structure is explained in Section IV.

The core component in adherence to EHT is to increase patients' self-efficacy and motivation, by providing education via the apps (native or web) on general information about hormone receptor positive breast cancer, importance of EHT adherence and potential side-effects, in addition to information on how to prevent or treat common side effects, self-monitoring/management of symptoms, coping and self-care skills, etc. The apps provide regular medication reminders, motivational messages, and facilitate direct communication with the medical team, as well as help them build a support network. In particular, the built-in Google Calendar is integrated into the apps. Regular motivational messages and social media group communication allow for educating and motivating patients through sharing of similar experiences from fellow patients as well as health care professionals. Since the apps use Google account based authentication, it is easier to synchronize the events (Google Calendar) and reminders from the PN. When the user signs in for the first time into the apps, she is prompted to choose a Google account from the account picker. This Gmail ID for the chosen account is recorded on the Firebase. PN uses this ID to send relevant information to the patient, such as emails, reminders, and notifications.

The key components of the apps are categorized based on the standard Open Systems Interconnection (OSI) [38] model in computer networks for better understanding of the proposed system architecture. The layered architecture of native and web apps are implemented as part of the research in this paper. The topmost layer is the Presentation layer, which consists of mobile and web user interfaces (UI). All the HTTP services and web services that the native and web apps will be using for rendering of the relevant information for users on the UI form the core research logic layer. This layer is the heart of the system where all the processing is done. Activities and Views in Android and html and embedded JavaScript [39] (EJS) pages in web app are all rendered based on the information exchange through HTTP services. HTTP services also consist of retrieving the educational content hosted on Github pages. The bottom layer is the data layer that consists of the Google Firebase database. Google Firebase stores all the data collected and transferred by the apps.

TABLE I
AMAZON EC2 NODE SERVER SPECIFICATIONS

Metric	Description
Operating System	Linux x86_64
Storage	Elastic Block Store (EBS)
EC2/ API Type	T2.small
CPU	1 virtual core
Memory	8GiB
Network Performance	Low to Moderate
Cost Estimate per month	$ 8.64 (24 hours used over 30 days)

B. System Features

This section describes the features of the apps developed using the proposed approach. As discussed in the Section II.A and Section II.B, from the shortcomings of the current mHealth apps for the self-management of the cancer survivors' care and overall wellbeing, we exploited the Patient Engagement Framework (PEF) [40] in the designing and implementation of the cross-platform apps. PEF model consists of the engagement strategies for the apps that are useful in improving the adherence in breast cancer patients. Besides the automated and PN-side monitoring, all the app features are developed based on the PEF model and are as mentioned below:

1. Provides basic information on how to find the services, clinical facilities and PNs through patient's therapy related education information sources.
2. Tracks symptoms status and intensities.
3. Facilitates patients to manage symptoms history or summary.
4. Sends notifications and reminders for health, daily care and symptoms assessment.
5. Offers social media (Facebook closed group) to create network of support and information exchange among other patients undergoing or have undergone therapy.
6. Provides support environment through Facebook group as well as PNs (administrator) interacting with patients on a regular basis.
7. Provides technical assistance with the issues related to apps usage, technical glitches with the existing features and adding new functionality.
8. Provides web-based dashboard for PNs for easy database CRUD (create, read, update and delete) operations on patient records without the necessity to visit Google Firebase JSON structure database. This is demonstrated in

Fig. 2. This functionality has proven to facilitate mHealth data collection for further analysis and processing, which are desirable attributes in health intervention research.

All the stated app features are incorporated as individual activities or web services or web pages. These are shown in detail in Fig. 3. With the Google Firebase, all user data can be saved on the cloud, making it secure and maintaining the confidentiality of patients' personal and health records.

IV. Implementation Methods

The native android mobile and cross-platform web apps have been developed through an iterative process mostly influenced by the UTSA and UTHSCSA teams' usability studies, constant feedbacks and focus group interviews with actual breast cancer patients undergoing EHT therapy. The developed apps can be accessed in two ways: Firstly, by downloading the android app from Google Play Store. Secondly, by installing the *.apk* file, which can be obtained through a trusted source. Another one is through web app, which can be accessed via its URL in any device's browser. The *.apk* file can also be downloaded from the web version of the app. The methods employed in developing the native and web apps are described in the following. The architectural differences between native and web apps in using the health-related web services are also highlighted.

As mentioned earlier, in the proposed prototyping approach an android app implementation is mapped to a cross-platform web with all the apps having the same functionality and the activities or web pages as shown in Fig. 3. The web services for apps are all available in RESTful fashion. For e.g. Firebase services for storing the data collected from patients. The contents of these service responses are JSON based in both web and native apps. As per the difference in the architectural anatomy [41] of both types of apps, the server side implementations of the same feature are deployed separately for native and web apps, although both still access the same back-end.

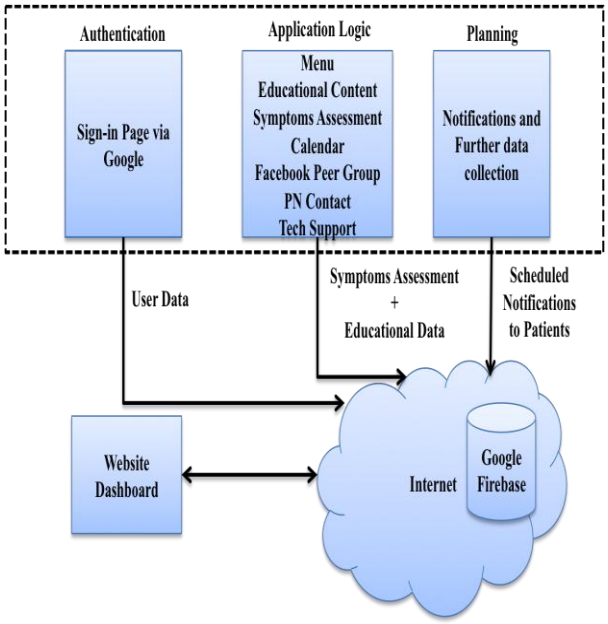

Fig. 2. Web-based dashboard for accessing admin related features

A. Phase 1: Implementation of Native Android Mobile Application

As a first step to systematic approach of designing interoperable cross-platform app, a native android app is developed. Android application is made up of Android activities [12]. Each activity includes one or more features of the application. All the activities interact with the user, so the activity Java class creates a window where we can place our User Interface (UI) defined by an XML file. A set of activities along with services, broadcast receivers and content providers, make an android application. Hence, the core components of the project were developed as different activities. Although the activities work together to form a cohesive user experience, each activity is independent from others. The *Intent* class is used to maneuver to another activity; this also allows few variables to be passed to the other activity.

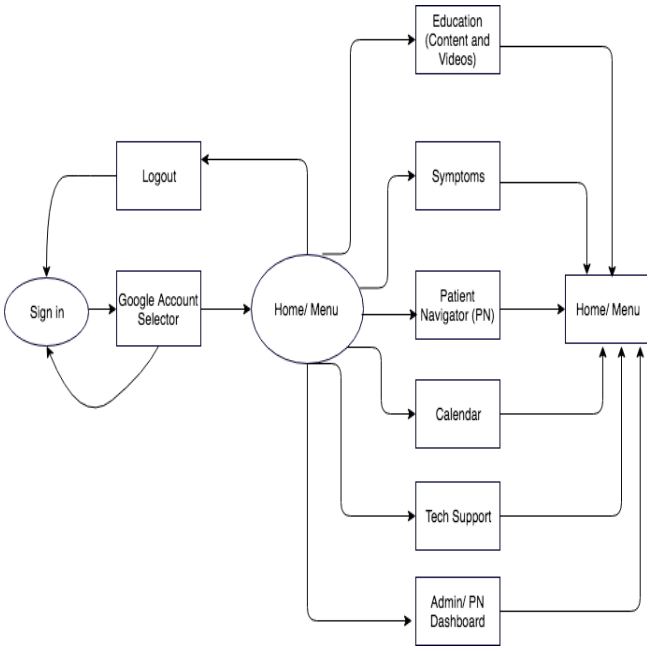

Fig. 3. Mobile and Web App Form-based chart

Application Structure. Android application is developed using an open-source build automation tool called Gradle [13] for adaptable and fast automation. The application consists of various activities corresponding to the application's functionality. It has following features:

- Clinical use and assistance in diagnosis (through symptoms reporting activity that provides the possibility to check up on symptoms and personal health record access through symptom weekly summary, through contact activity for providing easy way to communicate with the PN if necessary)
- Reminder (scheduling of appointments, help patients manage their appointments and events through calendar

activity)

- Easy medical referencing (Included reference guides and other breast cancer specialized medical referencing through Educational content and videos activity)
- Maintaining healthy lifestyle (through all the activities)

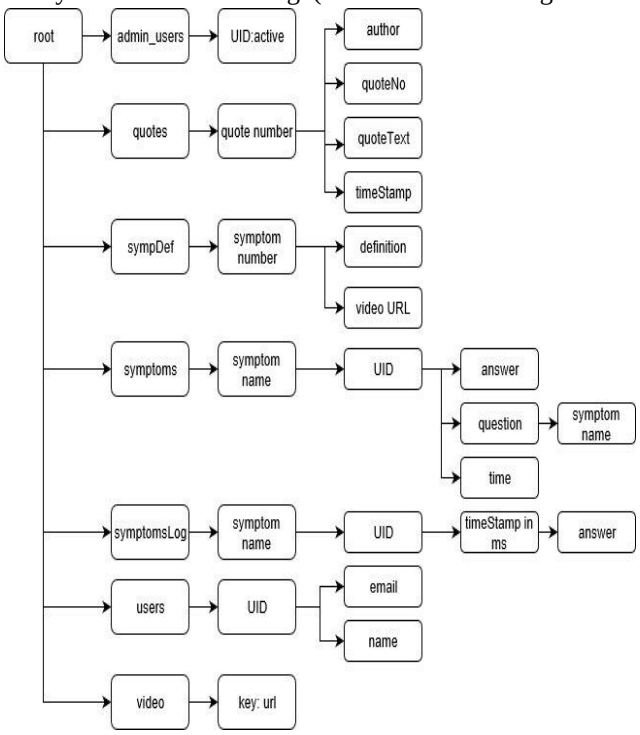

Fig. 4. Firebase database structure for the apps

All the activities are individually described in the sub-sections below.

Splash and Sign-in Activities: Android apps take some time to start up, usually when the app is opened for the first time. There is a delay that is unavoidable. The splash activity shows a logo or a visually appealing image. The amount of time the user takes looking at the splash screen is the same time the app takes to start. Sign-in integrates standard Google Sign-in into the app. We retrieve the user's unique ID, along with their name and email ID to be stored in Firebase.

Home Activity: Home activity is the primary screen where the user navigates to different features of the app through buttons. Users can access the features through the buttons specific to each feature. Each feature is described in the following sections.

Reminder and Events on Calendar App: The application allows users to keep track of their medication, doctor appointments, and peer group interactions. It also allows users to sync in phone calendar app with the reminders from HT app (shared Google Calendar) for increasing adherence to their treatment. This is useful when an assigned doctor shares calendar data with a patient.

Educational Content: The application delivers educational videos, motivational quotes, and other information to help Hormone Therapy patients keep moving forward with their therapy.

Symptoms: This activity consists of a list of breast cancer symptoms for patients to keep track of. Patients can submit their symptom intensity levels and these get recorded in Firebase, along with patient information and timestamps. This data allows for PN and doctor to be aware of the patient's situation and advise patient accordingly.

Patient Navigator (PN): This is an activity class that displays the information related to PN. The user can contact the PN by email, text message, or phone call.

Contact Technical Support: This is an activity where the user can contact the developer or tech support team in case she finds issues with the app. The activity displays the phone number and email address of the team.

B. Backend for all the applications: Google Firebase:

Firebase is an Internet service, which is provided by Google. It includes several products such as Analytics, Real-time Database, and Authentication that can easily be integrated into the backend. The data stored in the Firebase is in the form of a JSON tree. Unlike SQL, there are no tables with rows. When the data is added to the JSON tree it becomes a node in the existing JSON structure with an associated key. A node can be accessed in source code by using the set of keys that index it in the JSON structure. Fig. 4 shows the database structure. The top-most node is called the root node. A symptom level submission under a particular symptom name, such as "decreased appetite", for example, is saved under the respective UID and the timestamp at which the most recent submission was done.

C. Facebook Closed Group for Peer Networking

A closed secret group is made under the Patient Navigator Facebook account. A secret group is where only the participating members added by the patient navigator are allowed access to read and write posts. For the Android application, a Facebook button is placed on the action bar. This button opens the Facebook group on the Facebook app. If a Facebook app does not exist on the user's device, the same group is opened using the device's default web browser. For the web application, the Facebook group is opened in browser.

D. Google Play Store Publishing

Publishing is the general process that makes the Android application available to users. When we publish an Android application we perform two main tasks:

1) *Prepare the application for release:* During the preparation step, a release version of the application is built, which users can download and install on their Android-powered devices.
2) *Release the application to users:* During the release step, we publicize, sell, and distribute the release version of the application to users.

Usually, the application is released through an application marketplace, such as Google Play Store. However, an application can also be released by sending its file directly to users or by letting users download it from our website. The publishing process is typically performed after testing of the application in a debug environment is finished. Also, as a best

practice, the application should meet all of the release criteria for functionality, performance, and stability before its release.

Once a native android app is installed on the mobile device, the native code, the data parsing, network management, UI rendering are maintained locally on the device. When a user launches the native app and issues the interaction requests by clicking the UI buttons, the native code processing logics are executed. This may trigger requests to the Web services, and send synchronous/ asynchronous requests to fetch data. The native apps have lots of cached or preloaded elements and when requesting Web services, they fetch only the updated data from the server rather than the contents of the entire application from scratch. The native app can create a new thread to issue an asynchronous request when a resource is required. When the data is downloaded, the native code can parse the data, process the data with the preloaded logic, and re-render the new UI.

E. Phase 2: Implementation of Web Application

The HT mobile app that has been discussed runs on the device once download from the Google Play Store [14]. The application makes use of web views and is hybrid in a way that some features are embedded as web pages. We have also developed a web application that runs on a Node.js web server and can be accessed through a web browser like Chrome, Firefox, Safari, or Internet Explorer. All the data is typically saved in Google Firebase (similar to native mobile application). The application is designed and developed for multiple device (mobile devices, laptops and desktops) usage. This implementation also extends to number of platforms, which can run the application, as its web-based nature ensures the application must not necessarily be platform-specific. Our web mobile application is a set of HTML, CSS, JavaScript and other related files that reside on the Node.js server. These web pages are formatted for smartphones and tablets, and they can be accessed through the mobile devices' web browser. For the web implementation, we focused on the most recent and popular way of Responsive Web Design [16], as a way to approach the mobile web presence of our application. Web content in the application can be adapted to make it more accessible to mobile users.

Used Technologies. Based on the architecture of the app as described in the Method A, android mobile application implementation section, the web application was developed using JavaScript as a programming language and the technologies below:

- **Node.js:** Event driven server-side JavaScript environment.
- **EJS:** Embedded JavaScript. A templating language that generates HTML markup with plain JavaScript. It is primarily HTML, but with additional features which allow us to efficiently re-use pieces of our project.
- **Session.js:** For secure and persistent login (keeping users authenticated/logged in).
- **Express:** Node.js web framework.
- **Passport:** Authentication middleware for Node.js. Help in authenticating with different methods.
- **Google's OAuth2 Authentication Strategy:** For creating a login flow for users and using the profile information to provide users with personalized functionality.

TABLE II
WEB APPLICATION ROUTES

Routes	Description
/	Sign-in page
/menu	Menu Page
/calendar	Calendar of events
/patnav	PN contact
/support	Contact technical support
/symptoms	Symptoms tracking and summary
/educational	Educational, motivational content and videos
/logout	Logout of Google account

- **Google Firebase:** NoSQL-based real-time database that provides a backend as a service. For patient and educational data storage and retrieval.

Application Structure. The application source code is hosted on a server, which makes it possible to make updates to the app without the process of submission and approval typically required by the app store. Patients and patient navigator (i.e., nurse) can access the application with his/her Google account. The following are some of the features of the Node.js web application:

- Google login and registration (using OAuth2 with *passport-Google-oauth2*)
- Require login for certain sections of the application (i.e. paths to different sections like symptoms, educational etc.)
- Linking Google account to user's application account in Google Firebase through the use of Unique User ID (UID)
- Registering newly signed in Google user to Firebase's *user* database.
- Storing user's symptoms data to Firebase's *symptoms* database for retrieval and analysis purposes.
- Retrieving updated educational content and videos from Firebase

To set up the base application, we set up *npm packages*, *node application*, *configuration files*, database *structure* and *routes* as mentioned in the following section.

Routes: Routes are the application end points and they determine how the application responds to user requests from the client. For our web application, we have the following routes as shown in Table II. All the routes to different pages of this web application are protected using route middleware**.** A user has to be logged in for accessing all the pages. Every web page of the website has the *Google Play Store button* that links to the Android mobile app.

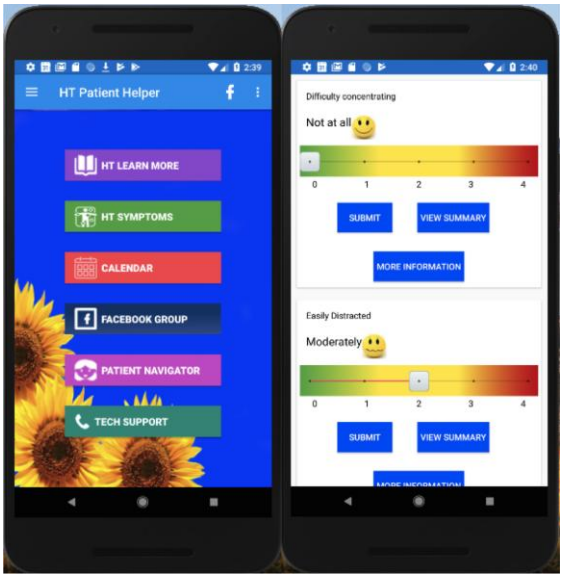

Fig. 5. 1) Menu page with access to all the app's features and 2) Symptoms Tracking Page

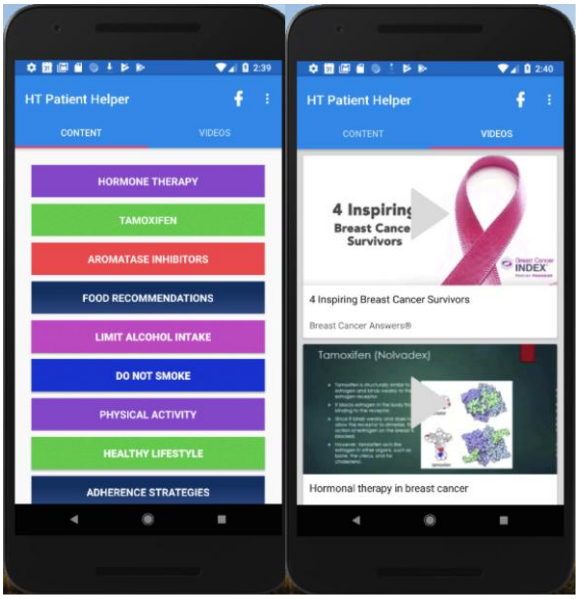

Fig. 6. 1) Educational Content Page; 2) Educational Videos Page

Web apps can be accessed in the smartphone's web browser such as Chrome, FireFox, Safari, and can update themselves without manual intervention. All the updates are pushed directly to the web server it is deployed on. The Web apps have to download all the resources such as web page files, images, JavaScript and CSS files to render the UI in any browser. The browser initiates a request to fetch the structure of the main page when a Web app is first loaded. Then the browser interprets the HTML, JavaScript, and CSS files to build up UIs. There are two forms of user interactions over the Web apps. The first one is to input some text fields or submit some forms, then the Web apps transition from one page to another page. The other one is that the Web app generates an AJAX (Asynchronous JavaScript and XML) request, and deals with the callback methods (for e.g. in login using Google OAuth2) without page transitioning. As the Web apps fetch the latest resources of a Web service, the user always sees the latest version of the app.

F. The Complete Interactive HT Patient Helper Application

The web implementation is released as a website. The user interface design of the android and web apps consider the user's technical abilities, the type of data displayed and collected and the varied sizes of mobile devices. They are also simplified by making the buttons larger and including appropriate graphic elements throughout the apps. Fig. 5 and Fig. 6 show the HT Patient Helper application screenshots.

V. Performance Assessment

This section discusses several measurement results of the proposed approach to mHealth app development for EHT breast cancer patients to assess the overall performance of both Native and Web apps. The current application development approaches [14] are adopted to enhance the implementation. The section also summarizes performance metrics such as throughput and response times collected during the load testing [42] of the apps. Table III shows the metrics and tools used for monitoring the performance indicators for the Android and Web Apps.

TABLE III
LIST OF TESTING TOOLS FOR PERFORMANCE ANALYSIS

Metrics	Tools
CPU Utilization	Chrome Dev Tools and Node.js Keymetrics PM2 for Web App. Android Studio profiler for Android App.
Memory Usage	Chrome Dev Tools and Node.js Keymetrics PM2 for Web App. Android Studio profiler for Android App.
Stress Testing Node.js Server	EC2-fleet
Throughput	Apache JMeter
Response Time	Apache JMeter
Stress Testing Database	Firebase Profiling

A. Cross-platform Development and Evaluation Approach

Although, web apps can run on any mobile platform, native apps have their own advantages too. Table IV summarizes the advantages of both native and web apps [43] as developers will undoubtedly ask themselves this open question whether to develop a native app or a web app.

For the evaluation and validation of the proposed Web App prototype, we used three different platforms: 1) Android OS, Moto G model, Version 5.1. 2) iOS, Apple iPhone 6s model,

Version 10.3.2. and 3) MAC OS X, Macbook Pro model, Version 10.9.5. The app was observed on Chrome Web Browser in the three devices. Native Android Mobile App was tested on Moto G model with Android Version 5.1.

B. Measures

Further, a set of technical and non-technical evaluation parameters described in [19] helped us identify the right type of solution.

User Experience. The golden rule for better end-user performance is that 80-90% of the end-user response time is spent on the frontend. Application launch time is one of the most important performance parameter for any smartphone as far as user experience is concerned.

Delivery Platform. Table V highlights some of the key differences that have been observed in terms of method of delivery of native mobile app and cross-platform web app.

We integrated application functionalities based on the feedback from the focus groups and in-depth interviews with breast cancer patients, oncologists, nurses and patient navigators. Application components include self-reporting of EHT adherence, side effect, symptoms tracking and accessing weekly summary reports in the form of graphs, educational and encouraging testimonial videos, emojis, Facebook support group information and a "Contact Us for Technical Assistance Tool". Focus group including breast cancer patients undergoing EHT therapy are involved during the study to improve and pilot-test these personalized, interactive mobile and web apps and through iterations, we achieved more than 80% of the key data and results on the effectiveness of our applications.

C. Web App Validation

1) Network Performance: Time To First Byte (TTFB) is one of the important aspect of website optimization and is the amount of time it takes from when a client device (experimental platform here) makes an HTTP request to it receiving the first byte of data from Node.js web server.

Values of TTFB between 200-500ms are standard. From the performance results, the First Byte time is 120 ms, 184 ms and 304 ms on our three test devices. We performed controlled experiments and examined how the web app loads, in a variety of circumstances. As the sites that load fast at one moment in time may load slower the next. It also includes one round trip of latency. It all depends on the current congestion in the routes our traffic takes. To smooth out these variances, we choose to perform tests 3 times.

The median values of the runs were collected on our three experimental devices after smoothing out the time anomalies. The tests were carried out on the University's Wi-Fi network. Taking into account just a pure network speed, the web app's home page was also loaded with the three devices in order to investigate the effect of the devices for the same application on an identical network.

The single run of the Web App on the same network and different devices such as MotoG 5.1, iPhone 6s 10.3.2 and Macbook Pro 10.9.5 takes the load times of 1.660 s for 391 Kb, 1.792 s for 880 Kb and 0.500 s for 909 Kb respectively. From this analysis, we see that the Web App can recognize a smartphone device and a laptop/ desktop, and take the user automatically to the mobile web app. The difference in the amount of bytes loaded can be observed with a totally different application, functionality and screen, thus indicating the optimized prototyping of the app.

2) CPU (load), Memory Usage (MB) and Memory Bandwidth: Data for our experiments was collected via three devices and monitoring the web app. Results of CPU load and Memory usage were monitored using the data collected by the advanced production process manager PM2 [17] for Node.js applications. The average CPU (load) and memory usage were observed to be 49.99 and 68.12 MB respectively.

TABLE IV
ADVANTAGES OF NATIVE MOBILE AND CROSS-PLATFORM WEB APPS

Native Mobile App	Web App
Extensive access to device hardware/ sensors and native mobile platform features (e.g. contact list, file systems or notifications)	Reaches broad community of audience (smartphone as well as desktop/ laptop users)
Better Performance	Fast multi-platform development
Apps' visibility via platform's app stores or marketplace such as The App Store or Android Apps on Google Play	Easy upgrades across all platforms

TABLE V
DIFFERENCES IN DELIVERY METHODS FOR THE APPS

Native Mobile App	Web App
Downloaded onto a mobile device	Can be accessed through a device's browser
Typically installed and run as a standalone application	Doesn't require any new software to be installed
Users need to manually download or approve the app and install app updates	Upgrades are made to the web server avoiding user intervention
App Stores and Marketplace help users to find the app	It can be hard for users to find the app

D. Native Android App Validation

1) Launch Time: In Android, an application can be launched in two ways [18]: 1) when the process is not present in the background, zygote can be forked to create a new application process and 2) already existing process in background can be

resumed. Hence, depending on the way application is launched, the launch times differ. Launch time is the time taken from the instant the user touches the application icon to the instant the app's activity shows up. It is observed that the launch of EHT Android app collected from Moto G through android logcat [19] takes the median application launch time of 0.493s. The measures were taken from the 3 runs of the app on Moto G 5.1 and LG Nexus 5.

The android application was validated by performing 20 tests in 3 scenarios for collecting the application's launch times: (a) Initially, many default applications like Calendar, Gallery, Camera, Video, Facebook that are already existing in the device are opened to consume more RAM and make the device slow. (b) Making sure not to kill the EHT Android application early. In this case, application launch time improves by loading it from background rather than forking it. 3) Closing the previous launch of the app entirely as soon as it is launched. Our application was opened in aforementioned different scenarios in order to test the actual performance by eventually calculating the medial launch time, which is shown in Table VII.

2) CPU and Memory Usage: CPU usage and Memory allocation of the android app was collected through a tool Android Debug Bridge (ADB) [20]. This tool monitors android apps performance along with the CPU load and memory consumption. CPU load is the percentage of CPU load spent in the app and the memory allocation is the total amounts of memory in megabytes allocated to the app. We observed that the average memory usage is approximately 68.71 MB, which is comparable to the server memory usage of 68.12 MB by Web App.

E. Performance Analysis Based on Load Testing

The objective of load testing is to obtain the response time of Native and Web Apps while varying the number of users. This type of testing simulates user access in order to acquire apps performance. Load tests were ran on the Android as well as Web apps. We monitored the hardware resources on the Node.js server in the case of Web app. For the Web app, initially stress testing is performed to determine the Node.js server's ability to handle the heavy load (concurrent users). This is in order to test the app's robustness and availability.

1) Stress Testing Node.js server for Web App: As discussed in the Section IV.E, we have our suitable production-ready Node.js server on our Amazon EC2 instance. It is Ubuntu-based Linux operating system and we have root access to it. Since, Amazon does not let us pick up more than 20 instances in one region, we used *ec2-fleet* utility [5] that creates many Amazon micro EC2 instances as clients across multiple regions to test our target web server [44]. Each instance has a T1.micro [45] configuration. We set up to 10 instances evenly distributed across regions. We ran the tests with the following client distributions:

a. 10 concurrent connections: 10 instances simulating 1 client each.
b. 100 concurrent connections: 10 instances simulating 10 clients each.
c. 500 concurrent connections: 10 instances simulating 50 clients each.
d. 1000 concurrent connections: 10 instances with 100 clients each.

The Node.js Web App could handle 1000 concurrent users without any special optimizations on the Web Server and the Client Instances. However, predictably, performance started decreasing beyond this point. This is due to the smaller amount of consistent CPU resources of the Client Instances and limited memory and CPU configuration for our server mentioned in Table I. This can be resolved by deploying the Web App Server on an instance with additional compute cycles in case of increase in the number of breast cancer patients undergoing the EHT therapy.

2) Load Testing of the Apps: As part of load testing we performed benchmark and user scenario tests. Apache JMeter [46] was used to create scenarios (user behaviours) for load tests for both the Android and Web Apps. JMeter can record browser or device's behaviours and simulate real situations for the evaluation of performance. User Scenarios were configured according to the Fig. 3 with various app's activities and web pages such as Login Scenario, Education Scenario and Symptoms Scenario etc. In the tests, we choose maximum of 1000 users to be simulated simultaneously as indicated by the stress test for current server configuration. We conducted tests for 5 minutes each with 10, 100, 200, 500 and 1000 users for each of the scenarios with the ramp-up period of 1 second. The performance plot for the CPU Utilization on the server for accessing educational content, videos, submitting symptoms and retrieving symptom summary for current week with the change in the maximum number of simultaneous users is obtained as in Fig. 7. This indicates the CPU ratio. All the user scenarios follow the similar performance curve even in the situations involving Firebase IO-Intensive operations. During the testing, the memory usage was around 300 MB for 1000 users as opposed to the minimum usage of 68.1 MB. It is also observed that the average response time and total requests per second become constant after the initial ramp-up time once all the users are concurrently using the app to 30 requests per second and 300 ms respectively.

Since the proposed prototyping approach and system architecture use separate Web Server (deployed on AWS EC2 instance) and Database Server (Google Firebase); it is expected to have impact on bandwidth when generating symptoms summary graphs. However, at a given time, the number of users simultaneously connecting to the database extending 100,000 as per the real-time database limits [47] is highly unlikely.

At any given time on a mobile device, a single user accesses android app. Since, the app explicitly relies on Firebase database backend for users, symptoms and educational videos data, Firebase profiling command line interface is used to measure read, write and broadcast speeds and Bandwidth. The average read speeds for symptoms and video nodes were observed to be 1 ms and 0.25 ms respectively. Since, we are using the free plan under Firebase, it can simultaneously support 1000 maximum database accesses, which justifies the choice of T2.small Amazon EC2 for Node.js web server. The Android and Web Apps conform to the standard operation.

VI. Future Work

The discussed prototyping approach makes the mHealth apps for chronic disease management accessible to patients on any device. The methods employed in this research also consider the UI and performance aspects of mobile and web apps when accessed on different mobile OS platforms. An important note is although its purpose is to result in optimal utilization, accessibility, efficiency and viewing the cross-platform app on multiple platforms, a responsive Web App doesn't automatically respond very well on every available device given context-based requirements of mHealth apps for different users. The future work will address these aspects by developing an iOS app as well as optimize the performance of the mHealth apps by incorporating caching and code offloading to the cloud environment making it easier for developers by having shorter app development cycles as well as user feedback. It is also planned to conduct more thorough testing using tablets, conducting focus group studies on actual patients undergoing therapy, increasing the capacity and utilizing the scalability of the AWS servers.

VII. Conclusion

This paper proposed and demonstrated a fast prototyping approach for cross-platform mHealth applications. Our proposed approach can be followed relatively easily in the existing mHealth apps, solving the interoperability and portability challenges being faced by healthcare/ digital health today, improving the performance aspects, some aspects of the UI and more specifically shorter app development times. With this approach, we accomplished three things: Firstly, development of native mobile and cross-platform web apps for providing necessary care to patients e.g. through access to relevant educational content, videos and motivational messages. Secondly, the approach also preserves trustworthiness by ensuring that the patients know who is receiving their health data (e.g. symptoms data) and for what purposes it will be used. Thirdly, the apps support all platforms and would be easy to work with any stakeholder in the healthcare community. The development of the API also adds to the interoperability and would help in bridging the divide between healthcare systems and patient data. This could also be a solution to an optimized availability of the chronic conditions specific content to the patients. Smartphone-based and Web-based adherence app for breast cancer patients developed a prototype with the proposed approach is scalable, inexpensive and easily accessible to anyone with mobile devices either through android marketplace or as open link to the website.

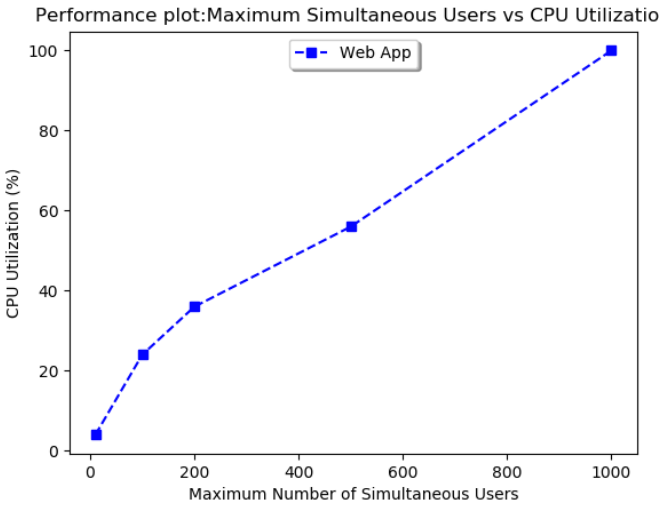

Fig. 7. Performance Curve for Load Tests

An extensive performance evaluation and measurement study was conducted on the two prototype android and web apps that were developed in the first stage of our work. The article in [41] conducted a comparison study on the performance of native and web apps from the perspective of web services, HTTP(S) requests, traffic, round-trip time, and energy consumption. Different from [48], this paper studied the performance of mHealth android and web apps on smartphones by tracking response time, app load time, CPU utilization and memory usage. This paper also studied the impact of concurrent users on the CPU utilization on web server as well as database server Firebase in the case of android app.

It was observed that although native apps usually performed better for e.g. app's load time on an android device takes 0.493s as opposed to 1.660s for web app on the same device, some alternate and exception scenarios were observed. However, when accessing the web services, web apps reuse the existing network connections to reduce the latency of fetching new resources. This observation was made use in our android app too for hosting medical educational content for patients. This RESTful pattern of using web services on mobile platform boosts the performance of the android app on the resource-constrained mobile devices. In particular, following the approach of having Firebase as a separate database server, Node.js web server on AWS and implementing the native and cross-platform web apps leveraging their services led to the platform neutrality and easy deployment of mHealth apps.

Acknowledgment

The study is funded by Susan G. Komen (Award No. SAB160005), the Mays Cancer Center (Grant No. P30 CA054174) and Redes En Acción (Grant No. U54 CA153511).

REFERENCES

[1] D. T. Dunsmuir, B. A. Payne, G. Cloete, C. L. Petersen, J. Lim, P. Von Dadelszen, G. A. Dumont, J. M. Ansermino, "Development of mHealth applications for pre-eclampsia triage", *IEEE J. Biomed. Heal. Informatics*, vol. 18, no. 6, pp. 1857-1864, 2014.

[2] The University of Texas Health Science Center at San Antonio. *Improving Adherence to EHT among Breast Cancer Patients.* Accessed: July. 2017. [Online]. Available: http://clinicaltrials.gov/ct2/show/NCT02850939. NLM Identifier: NCT02850939.

[3] N. Saranummi, "In the spotlight: Health Information Systems", *IEEE Rev. Biomed. Eng.*, 4:17-19, 2011.

[4] J.Laurance, *Breast Cancer Cases Rise 80% since Seventies; BREAST CANCER*, The Independent. London. 2006.

[5] EC2-fleet. *EC2-fleet.* Accessed: May. 2018. [Online]. Available: https://github.com/ashtuchkin/ec2-fleet

[6] R. M. Califf, MD. Lawrence, H. Muhlbaier, "Health Insurance Portability and Accountability Act (HIPAA), Must There Be a Trade-Off Between Privacy and Quality of Health Care, or Can We Advance Both?," *Circulation, AHA Journals*, August 25, 2003.

[7] N. Serrano, J.Hernantes, and G. Gallardo, "Mobile web apps", *IEEE Softw.*, vol. 30, no. 5, pp. 22-27, Sep./Oct. 2013.

[8] Hybrid apps. *Hybrid Apps.* Accessed: June. 2017. [Online]. Available: https://en.wikipedia.org/wiki/Multi-Channel_app_development

[9] Castelnuovo G, Mauri G, Simpson S, Colantonio A, Goss S, "New technologies for the management and rehabilitation of chronic diseases and conditions." *Biomed. Res. Int.* 2015. 180436 (2015).

[10] Kearney N, Mccann L, Norrie J *et al*, "Evaluation of a mobile phone-based, advanced symptom management system (ASyMS) in the management of chemotherapy-related toxicity." *Support. Care Center* 17(4), 437-444 (2009).

[11] Bennett AV, Jensen RE, Basch E, "Electronic patient-reported outcome systems in oncology clinical practice." *CA Cancer J. Clin.* 62(5), 337-347 (2012).

[12] Eysenback G, "What is e-health?" *J. Med. Internet Res.* 3(2), E20 (2001).

[13] Eggersmann K T, Harbeck N, Schinkoethe T, Riese C, "eHealth solutions for therapy management in oncology." Future Medicine. *Breast Cancer Manag.* (2018) 6(3), 101-106.

[14] Christine Tran, Adam P.Dicker, Heather S.L. Jim, "The Emerging Role of Mobile Health in Oncology." *The Journal of Targeted Therapies in Cancer*, (2017).

[15] Kay M, Santos J, Takane M, "mHealth: new horizons for health through mobile technologies." *World Health Organization website*. Accessed: May. 2018. [Online]. Available: http://www.who.int/goe/publications/goe_mhealth_web.pdf

[16] Moumtzoglou A, "mHealth Innovations for patient-centered care." *IGI Global*. 2016 doi: 10.4018/978-1-4666-9861-1.ch016.

[17] Kessel KA, Vogel MM, Schmidt-Graf F, Combs SE, "Mobile apps in oncology: a survey on health care professionals' attitude towards telemedicine, mHealth and oncology apps," *J Med Internet Res.* 2016; 18(11):e32.

[18] Lewis J, Ray P, Liaw ST, "Recent worldwide developments in eHealth and mHealth to more effectively manage cancer and other chronic diseases – a systematic review," *Yearb. Med. Inform.* 10(1), 93-108(2016).

[19] Bender JL, Yue RYK, To MJ, Deacken L, Jadad AR, "A lot of action, but not in the right direction: systematic review and content analysis of smartphone applications for the prevention, detection, and management of cancer," *J. Med. Internet Res.* 15(12), e287 (2013).

[20] Jenson RE, Synder C.F, Abernethy AP *et al*, "Review of electronic patient-reported outcomes systems used in cancer clinical care," *J. Oncol. Pract.* 10(4), E215-E222 (2014).

[21] Kuijpers W, Groen WG, Aaronson NK, Van Harten WH, "A systematic review of web-based interventions for patient empowerment and physical activity in chronic diseases: relevance for cancer survivors," *J. Med. Internet. Res.* 15(2), e37 (2013).

[22] McGillicuddy JW, Weiland AK, Frenzel RM, *et al*, "Patient attitudes toward mobile phone-based health monitoring: questionnaire study among kidney transplant recipients," *J. Med. Internet. Res.* 2013; 15(1): e6. doi: 10.2196/jmir.2284.

[23] Cox A, Lucas G, Marcu A, *et al*, "Cancer survivors' experience with telehealth: a systematic review and thematic synthesis," *J. Med. Internet. Res.* 2017; 19(1): e11. doi: 10.2196/jmir.6575.

[24] Yamin CK, Emani S, Williams DH, *et al*, "The digital divide in adoption and use of a personal health record," *Arch. Intern. Med.* 2011; 171(6): 568-574. Doi: 10.1001/archinternmed.2011.34.

[25] Research Electronic Data Capture. *REDCap*. Accessed: March. 2017. [Online]. Available: https://www.project-redcap.org

[26] Mandel JC, Kreda DA, Mand KD, Kohane IS, Ramoni R, "SMART on FHIR: a standards-based, interoperable apps platform," *J. Am. Med. Inform. Assoc.* 2016;23(5): 899-908. Doi: 10.1093/jamia/ocv189.

[27] Bloom eld RA Jr, Polo-Wood F, Mandel JC, Mand KD, "Opening the Duke electronic health records to apps: Implementing SMART on FHIR," *Int. J. Med. Inform.* 2017;99:1- 10. Doi: 10.1016/j.ijmedinf.2016.12.005.

[28] Bruno M.C. Silva, Joel J.P.C. Rodrigues, Isabel de la Torre Diez, Miguel Lopez-Coronado, "Mobile-health: A review of current state in 2015," *J. Biomed. Inform.* 56(2015) 265-272. Doi: 10.1016/j.jbi.2015.06.003.

[29] Lingopal Holdings Pty, *Dr K's breast checker, 2014*. Accessed: May. 2018. [Online]. Available: https://itunes.apple.com/us/app/dr-ks-breast-checker/id385045662?mt=8

[30] Pcr tracker, 2014. *Pcr tracker*. Accessed: May. 2018. [Online]. Available: https://itunes.apple.com/ch/app/pcr-tracker/id592097118?l=en&mt=8

[31] SkinKeeper, 2014. *SkinKeeper*. Accessed: June. 2018. [Online]. Available: https://itunes.apple.com/lv/app/skinkeeper/id486413797?mt=8

[32] Preethi R Sama, Zubin J Eapen, Kevin P Weinfurt, Bimal R Shah, Kevin A Schulman, "An Evaluation of Mobile Health Application Tools," *J. Med. Internet. Res.* 2014;2(2):e19. Doi: 10.2196/mhealth.3088.

[33] Amazon EC2. *Amazon Elastic Cloud Compute*. Accessed: February. 2017. [Online]. Available: https://aws.amazon.com/ec2

[34] Node.js. *Node.js*. Accessed: March. 2017. [Online]. Available: https://nodejs.org/

[35] Github pages. *Github Pages*. Accessed: February. 2018. [Online]. Accessed: https://pages.github.com

[36] Harte R, Quinlan LR, Glynn L, Rodriguez-Molincro A, Baker PM, Scharf T, OLaighin G, "Human-Centered Design Study: Enhancing the Usability of a Mobile Phone App in an Integrated Falls Risk Detection System for use by Older Adult Users," *J. Med. Internet Res.* 2017;5(5):e71.

[37] P. Harbig, I. Harbig, EM. Damsgaard, "Suitability of an electronic reminder device for measuring drug adherence in elderly patients with complex medication," *J. Telemed Telecare*, vol. 18, no. 6, pp. 352-356, 2012.

[38] Open Systems Interconnection Model. *OSI Model*. Accessed: June. 2018. [Online]. Accessed: https://en.wikipedia.org/wiki/OSI_model

[39] Embedded JavaScript Templating. *EJS*. Accessed: March. 2017. [Online]. Avaiable: http://ejs.co/

[40] Yimin Geng, Sahiti Myneni, "Patient Engagement in Cancer Survivorship Care through mHealth: A Consumer-centered Review of Existing Mobile Applications," *AMIA Annu Symp Proc.* 2015; 2015: 580-588. PMCID: PMC4765566.

[41] Yun Ma, Xuanzhe Liu, Yi Liu, Yunxin Liu, Gang Huang. "A 41 of Two Fashions: An Empirical Study on the Performance of Native Apps and Web Apps on Android," *IEEE Transactions on Mobile Computing*, Vol. 17, No. 5, May 2018. Doi: 10.1109/TMC.2017.2756633.

[42] S. Okuboyejo and E. Omatseyin, "mHealth: Using Mobile Technology to Support Healthcare," *Online Journal of Public Health Informatics* 5.3 (2014): 233. *PMC*. Web. 10 July 2017.

[43] Holzinger A, Treitler P, Slany W, "Making apps useable on multiple different mobile platforms: On interoperability for business application development on smartphones," *International Federation for Information Processing*. 2012.

[44] Node.js Web App Server. *Node.js Web App Server*. Accessed: March. 2017. [Online]. Available: http://ec2-52-10-29-128.us-west-2.compute.amazonaws.com/.

[45] Amazon T1 Micro Instances. *AWS T1 Micro Instances*. Accessed: February. 2017. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts_micro_instances.html

[46] Apache JMeter. *JMeter*. Accessed: January. 2018. [Online]. Available: https://jmeter.apache.org/

[47] Realtime Database Limits. *Firebase Limits*. Accessed: May. 2017. [Online]. Available: https://firebase.google.com/docs/database/usage/limits

[48] Z. Hemel and E. Visser, "Declaratively programming the mobile web with mobl," *ACM SIGPLAN Notices*, vol. 46, no. 10, pp. 695-712, 2011.

[49] Evdokimos I. Konstantinidis, Antonio S. Billis, Christos A. Mouzakidis, Vasiliki I. Zilidou, Panagiotis E. Antoniou, and Panagiotis D. Bamidis, "Design, Implementation, and Wide Pilot Deployment of FitForAll: An Easy to use Exergaming Platform Improving Physical Fitness and Life Quality of Senior Citizens", *IEEE J. Biomed. Heal. Informatics*, vol. 20, no. 1, pp. 189-200, 2016.

[50] Temitope O Tokosi, Jill Fortuin and Tania S Douglas, "The impact of mHealth Interventions on Breast Cancer Awareness and Screening: Systematic Review Protocol", *J. Med. Internet. Res. Research Protocols*, vol. 6, iss. 12, 2017.

[51] Abu Saleh Mohammad Mosa, Illhoi Yoo and Lincoln Sheets, "A Systematic Review of Healthcare Applications for Smartphones", *BMC Medical Informatics and Decision Making*, 2012. Doi: 10.1186/1472-6947-12-67.

[52] Trung Q. Le, Changqing Cheng, Akkarapol Sangasoongsong, Woranat Wongdhamma and Satish T. S. Bukkapatnam, "Wireless Wearable Multisensory Suite and Real-time prediction of Obstructive Sleep Apnea Episodes", *IEEE Journal of Translational Engineering in Health and Medicine*, 2013. Doi: 10.1109/JTEHM.2013.2273354.

[53] Kemal Serdaroglu, Gamze Uslu and Sebnem Baydere, *e-Health Pervasive Wireless Applications and Services (eHPWAS'15),* 2015.

[54] Anupeng Huang, Chao Chen, Kaigui Bian, Xiaohui Duan, Min Chen, Hongqiao Gao, Chao Meng, Qian Zheng, Yingrui Zhang, Bingli Jiao, Linzhen Xie, "WE-CARE: An Intelligent Mobile Telecardiology System to Enable mHealth Applications", *IEEE J. Biomed. Heal. Informatics*, vol. 18, no. 2, pp. 693-702, 2014.

***Declaration of Interest Statement**

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: