ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 5, May 2019

RxMedicus: An Application to Facilitate Consultation in Healthcare

Ajith N S [1] , Mitul Bipin [2], Nikhil Choudhary [3] , Sameer Paisari [4]

Final Year UG Student, Dept. of ISE, School of Engineering and Technology Jain University, Bangalore, India[1,2,3,4]

ABSTRACT: The main aim of the project is to develop an app that adds value to the healthcare industry. The project aims to serve the best interests of the general public, a large portion of which will possibly be in need of healthcare consultation and also doctors alike. With the growing advent of information technology and the rapid strides it is taking in today's world, the accessibility it has and the kind of outreach it sees with the modern-day population is mind boggling. Bringing healthcare to one's palm and trying to ensure accessibility to one and all regardless of social status is one of the project's primary goals**.**

KEYWORDS: Healthcare, consultation, app, information technology

I. Introduction

Healthcare in India has become one of the largest sectors in India in terms of revenue andemployment. This sector is expected to have the fastest employment growth between 2014-24.This constant growth leaves a void in the domestic market in terms of accessibility and outreachgiven our country's diversity. With the increasing emergence of digital platforms, the aim of theproject is to provide an all inclusive solution regarding healthcare on an easy-to-use application.There are several existing platforms offering snippets of information but the idea is to expand onthem and diversify solutions based on customer requirements. This may greatly help in ensuringbetter solutions provided to those in need and making healthcare an accessible service regardless of one's social status.

II. Existing System

With the rise in the increase in the consulting charges in the medical field, It becomes quite difficult for the patients to afford such medical treatment and therefore people tend to skip the medical attention which is required, This leads to unnecessary rise of a disease which is rather not curable.

Maintaining the medical history of all the records till date is rather difficult in recent times since each and every disease, cure, prescription requires a document which becomes difficult in handling. These records contain valuable information which needs to be acknowledged by the doctor and having the hard copy of these documents makes it a more confusing task for the doctor to prescribe medicines.

Finding a suitable doctor for the required concern is a difficult task in the present situation since there are various medical fields and It is quite overwhelming when it comes to the patients. Also, the approach to the doctors is not possible without proper considerations made by the developers.

Locating a hospital/clinic for further consultation which cannot be done through the means of application is a difficult task since the patient doesn't have the required knowledge about the specialization about the doctor nearby the patients' location.

The increasing availability of Information Technology gives the rise to the possibility of it being implemented in the medical field, making healthcare/ consultation accessible to the ever-rising number of smartphone and internet users the project aims at developing an app which holistically caters to the needs of doctors and patients alike.

DOI: 10.15680/IJIRCCE.2019. 0705064

III. Tools Used

A. Android Studio

Designed specifically for Android development, Android Studio is the official integrateddevelopment environment for Android OS. It is built on IntelliJ IDEA software. It has multiplefeatures available to enhance user experience as enumerated upon subsequently.

B. Firebase Realtime Database

The Firebase Realtime Database is a cloud-hosted database, developed by Google. The USPof this database is that it enables real-time syncing of data across all clients and remainsavailable even when the app goes offline. Data is stored as JSON (JavaScript Object Notation)and synchronised real-time to every relevantly connected client. JSON is a lightweight formatwhich makes the storage and transportation of data a feasible or even a relatively easy task. Itis most often used during the process of sending data from a server to a webpage. It is alsodubbed “self-describing” which exemplifies the ease with which it can be understood

IV. Implementation

A. Firebase Realtime Database

The Firebase Database is built to facilitate the building of rich, collaborative applications, enabling secure access to the database using code from the client’s end. The data is persisted locally to ensure that users have a responsive experience even when offline. The local data

changes are seamlessly merged to avoid conflict as soon as the device regains connection. The API is designed to allow only operations that can be quickly executed, with the database being a NoSQL database decked with numerous optimizations and functionality when compared to a real time database.

- Integrate Firebase real-time with Database SDKs.
  Via script includes or so on
- Create Realtime Database references
- Reference JSON data or subscribe to data changes.
- Set Data and Listen for changes.
- Use reference to write data or subscribe to changes.
- Enable offline persistence.
- Write data to local disk so as to enable offline access
- Secure your data
- Using Firebase Realtime Database Security rules

B. Cloud Storage

Built for app developers who rely on storage and serve user-generated content, cloud storage for firebase is a simple, powerful and cost-effective tool built keeping the scale of Google in mind. The firebase SDKs for cloud storage implement Google’s security to file uploads and downloads for Firebase apps, in spite of whatever the network quality is. Inherent SDKs can be used to store images, audio, video or any other format of user-generated content. Google cloud storage can be used to access the same files on the server.

Implementation path:

- Integrate Firebase SDKs for cloud storage
- Create a reference
- Upload or download
- Secure files

C. Android Studio

Designed specifically for Android development, Android Studio is the official integrated development environment for Android OS. It is built on IntelliJ IDEA software. It has multiple features available to enhance user experience as enumerated upon subsequently.

Visual Layout Editor - Enables creation of complex layouts and also provides the option to preview it on any screen size by selecting a device configuration or even just resizing the preview window.

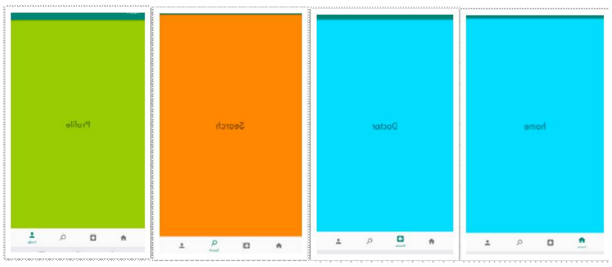
APK Analyzer - Provides the option to reduce app size by inspecting contents of the APK file.

Intelligent Code Editor - Provides code completion for Kotlin/Java/C++ thereby increasing productivity.

Flexible Build System - Allows for customization of build to generate multiple variants for different devices, stemming from a lone project. This feature is powered by Gradle.

D. *Bottom Navigation*

The bottom navigation bars make it extremely easy for users to explore top-level views with a single tap. Tapping on the button enables the user to see the relevant section with all the requisite/associate information. Android studio has developed a default predefined template to create a bottom navigation view, which can be used by creating New Activity -> Bottom Navigation Activity. There are also provisions to customize the view to suit our requirements.



The processes involved in the creation of the bottom navigation view are as follows:

- *Creation of an Android Studio Project:*
  Creating a new project with an empty activity.

- *Addition of*BottomNavigationView*:*
  Involves import of design support which is added to the module’s build gradle file.
  Inclusion of BottomNavigationView widget, which includes a constraint layout and a frame layout. Once the menu is created containing the menu items, each item includes an ID, an icon and a title.

- *Initialization of components:*
  Involves initializing an instance of the BottomNavigationView.

- *Testing the app:*
  Involves running the application.

- *Configuring click events:*
  This process is undertaken to ensure that each click on a particular activity takes the user to his/her desired section.

  The subsequent processes involve creation of fragments (pages) which will contain the requisite information of the particular sections created in the aforementioned processes, after which the fragments can be launched which sets the app into motion and gets it up and running.

V. Authentication

A. Email Verification

The privacy and security of a user is of utmost importance and in order to provide users with thebest-in-class security, the feature of authenticating a user via email verification has beenimplemented.

Firebase authentication is used to sign in a user, by sending them an email containing the link,which is thereby used to sign-in.

The process involves setting up an android project, enabling email link sign-in for Firebaseproject, sending an authentication to the user’s email address and finally - completing the sign-in with the relevant email link.

There are valid security concerns raised regarding this, not least of which are methods toprevent a sign-in link from being used by an unintended user. This is why the authenticationsystem of Firebase requires the email ID of the user to be provided during the process ofsigning in. In-order to successfully log in, the email address must be the same as the one towhich the sign-in link was sent originally. After the completion of the process of signing in, anyprevious unverified attempts to sign in will be disbanded from the user and existing sessions willbe invalidated to prevent an impersonator from trying to sign in again with the same unverifiedusername and password.

VI. Testing

A. Static Analysis

an Android application is in the type of an Android bundle, or apk, archive, which is a zip bundle. The apk archive incorporates the manifest, alongside different resources and folders. To separate the highlights of intrigue, we first need to figure out the apk files, which we achieved utilizing the APK tool in Virustotal.

The record AndroidManifest.xml contains a few highlights that could be utilized for static analysis. Here, we center around the permissions mentioned by the application. The AndroidManifest.xml contains a rundown of all authorizations required by the application. Android utilizes an exclusive binary xml format, so we structured our very own custom xml parser to extract the authorization features from AndroidManifest.xml files.

Given an Android application the accompanying advances portray the procedure we use to extricate the permissions features.

1. Figure out the Android application. This figuring out is accomplished utilizing the APK tool in Virustotal12.

2. Extract the permissions mentioned from the AndroidManifest.xml record utilizing our custom xml parser.

3. Produce a binary feature vector, as in (1).

4. At long last, we constructed a permission vector dataset for every one of the applications in our dataset store it in an ARFF13 file format.

Of the 135 potential permissions, many were never mentioned in any of the Android applications in our datasets. These authorizations were expelled from thought since they don't contribute anything to the analysis. Moreover, a few features (i.e., permissions) give next to zero valuable data. Along these lines, to further lessen the length of our feature vectors, we have utilized feature selection based on a direct information gain calculation.

B. Dynamic Analysis

True to form, an Android application interfaces with the working framework through framework calls. We have separated framework calls utilizing dynamic investigation. To accomplish this, we have utilized the Android emulator that is incorporated with Android Studio14. Each Android application in our dataset has been executed in a different emulator, with the recurrence of each framework call recorded.

We interface with the emulator occurrence utilizing the Android Debug Bridge (ADB)15, which is an order line instrument found in the Android SDK. The ADB comes with an alleged Monkey Runner16, which can be utilized to imitate arbitrary UI communications. These occasions incorporate snaps, volume connections, contacts, thus on, which trigger framework calls. We record the subsequent framework calls utilizing the checking apparatus Strace17

In detail, the copying and information gathering comprises of the accompanying advances.

1. Open the AVD Manager in Android Studio also, click on Create New Device. This makes an emulator occurrence and runs it.

2. After the emulator is running, we open the terminal and explore to the stage apparatuses organizer of the Android SDK. There we enter adb help to confirm that the ADB is working true to form.

3. Next, we issue the order adb gadgets which records the emulator ID that is running.

4. Accepting the Android application is named ApplicationName.apk, we give the order

adb introduce ApplicationName.apk

(by means of a group document). Now, we can confirm that the application document has been introduced in the emulator.

5. Next, we enter the emulator shell by composing

adb - s emulator-5646 shell

at the terminal.

6. We dispatch the application and check theprocess ID utilizing the order

ps<package name>.

7. The command

strace - P <ProcessID> - c - o

<path in emulator>Filename.csv

<package name>

starts the account of framework calls.

8. We begin Monkey Runner utilizing the order

adb shell - p <package name> - v 500 - s 42.

As referenced over, this creates arbitrary occasions through the UI. At the same time, Strace will record the recurrence check of the framework calls that are created.

9. After the Monkey Runner occurrence stops, we ex-tract the log document utilizing the order

adb pull <path in emulator>

<path in destination>.

Obviously, the exact grouping of framework calls created will fluctuate, contingent upon the irregular determination made by the Monkey Runner. Be that as it may, the recurrence of the different framework calls is moderately steady for a given application.

The recurrence portrayal of framework calls conveys data about the conduct of the application (Burguera, 2011). A specific framework call may be used more in a pernicious application than in a kindhearted application, and the framework call recurrence portrayal is proposed to catch such data.

VII. INTERACTION

A. Interaction between Doctor - Patient:

The app is built to serve the purpose of facilitating a smooth interaction between doctors and patients. The very aim of the project is to ease the process of consultation and give users - be it doctors or patients, on both ends of the spectrum, exactly what one is looking for and enable a seamless experience of communication. The patient will be able to view the available doctors on his/her home screen/ home section of the app. The question that remains is how this interaction will be implemented. A child node is created in the previously mentioned database for doctors from which details are retrieved and a list view is created. The list view contains details such as the name, specialization, clinic address and number(optional) of the particular doctor. Based on these details, a patient will be able to select the appropriate doctor as per one’s requirements and make a well-rounded decision about which doctor to consult. There is also umpteen scope to add to these features and inculcate many others, keeping in mind the enhancement of user experience in the future.

B. Interaction between Doctor - Doctor:

To add to the previously mentioned feature of facilitating interaction between a doctor and a patient, there is also an undying need for a feature which facilitates interaction between two doctors. Picture a scenario where patient is interacting with a doctor who feels that he/she can refer the patient to another doctor with greater/required expertise to make way for enhanced consultation or treatment. Implementing a feature in the app that would enable a doctor to share details of a particular patient for matters of reference is of paramount importance keeping in mind the efficiency of treatment received via the app. This feature cur5rently is being implemented via redirecting a doctor to the particularly available in-built messaging app through which details can be shared. Future scope demands can be met by implementing a chat box or a messaging box within the app for purposes of easing communication and enhancing the overall ease-of-use.

REFERENCES

[1]Adhistya Ema Permana sari, Warsun Najib, Reza Rully Kusuma “Prototype Development for an Android based Medical Record System” IEEE 2017

[2] Navdeep Singh” Study of Google Firebase API for Android”International Journal of Innovative Research in Computer and Communication Engineering Vol. 4, Issue 9, September 2016

[3] L Jaya, B. Putra, T. Suryadi, M. H. Irfani, and J. S. Informasi, “SistemInformasiRckamMedis Rawat Jalan Pada RumanSakatUmumDacrah Palembang BARI,” J. Sist. Inf. STMIK MDP, 2013.