Android application to store and analyze medical data

An industrial training report submitted

to

MANIPAL UNIVERSITY

For Partial Fulfillment of the Requirement for the

Award of the Degree

of

Bachelor Of Technology

in

Information Technology

by

Aman Chopra

Reg. No. 140911358

Under the guidance of

Miss. Jyoti Puri

Project Manager

Humble, Software Development as it should be

MANIPAL
INSTITUTE OF TECHNOLOGY

A Constituent Institute of Manipal University, Manipal

October 2017

CERTIFICATE

To whomsoever It May Concern

This is to certify that Mr. Aman Chopra completed his 8 weeks internship with our organization as an intern.

Project Start Date: 23/05/2016
Project End Date: 18/07/2016

During the internship he was very actively involved in android application development. He worked for projects of Humble's clients and has also contributed to open-source and community project of Humble.

Jyoti Puri
Project Manager

Humble
http://humble.net.in/
A 35/5 MANDAWALI FAZALPUR NEAR JOSHI COLONY DELHI 110092

ACKNOWLEDGEMENTS

During the course of my industrial internship I got an exposure to the corporate work culture. I received an opportunity to adapt to the office environment and learn about android application development, web development and version control tools. Since the company is a startup by a group of freelancers, working on the internship helped me gain knowledge and experience on how the freelancing community works and how it is different from the established companies. This experience has helped me shape myself in accordance with the current industrial developments and trend. The internship was a good learning curve of the professional environment and overall it was a very insightful and wonderful learning experience.

I would like to take this opportunity to thank the Institute and Department of Information and Communication Technology in giving me this opportunity to do an internship during my Summer Vacations. I would also like to extend my gratitude to Humble, my mentors, my manager, and all the other people who helped me in successfully completing this training.

ABSTRACT

With the country turning smarter every day, it's only fair that its citizen turn even smarter. In the time where carrying documents has become almost obsolete, when it comes to carrying prescriptions it becomes even more annoying. CARDIGRAM, an interactive android application aims to give its users a wide range of functionality in the medical world, making it their most utility companion. The platform conferences between the doctors and their patients from the very start to ensure full time assist for those in need. The main objective of CARDIGRAM is to reduce the communication distance in the medical world where even the slightest of seconds count.

The front end of the application is made following the principles of material design which is a design language developed in 2014 by Google. The application uses Google firebase for real time database, storage, authentication and analytic. Git which is a Version control tool is used to work in collaboration inside the company.

The goal of this application is to help patients store their personalized medical data securely and use it on the go. The application provides many functionalities including booking appointments with doctors, chatting and visual analysis of textual data using animated charts.

[Keywords-features]: Android, Firebase, Material Design, Git

Contents

List of Tables

List of Figures

ABBREVIATIONS

API	:	Application programming interface
WYSIWYG	:	What you see is what you get
SDK	:	Software development kit
GUI	:	Graphic user interface
PaaS	:	Platform as a service
UI	:	User interface
UX	:	User experience
HTTPS	:	Hypertext transfer protocol secure
GPS	:	Global Positioning system
MD-5	:	Message digest-5
GNU	:	GNU's Not Unix
GPL	:	General public license
JSON	:	JavaScript object notation

NOTATIONS

$\bullet$	:	Initial state
$\odot$	:	Final state
$\dashrightarrow$	:	Dependency
$\rightarrow$	:	Association

Chapter 1

Details of the organization

1.1 Humble

Humble is a team of talented software developers. Humble believes in not only building high quality softwares, but solutions that help businesses and delivers value to our customers. Its said highest comes to humblest. At humble excellence in work and humbleness in attitude is way for us to grow individually and as team. Figure 1.1

Figure 1.1: Humble, software development as it should be.

1.2 Services

Humble has expertise in following areas:

- Developing web applications. The company expertise in developing rich web applications with great UX, responsiveness and capability to handle large data volumes. The technologies we use for web development are ReactJS, NodeJS, MongoDB, Postgres.
- Developing mobile applications using react native.
- CI and managing code deployment. [1]

1.3 Customers

Following customers are associated with humble for its services. [1]

- Atlassian
- Audioflea
- Atlassian
- SL Engineering Company
- Xebia
- Ephesoft
- Altavoz
- uthored
- IPAOO
- Sizung
- GoStudent

1.4 Mission

Humble believes in making software solution for its customer's business understanding, their need and using the technology that is most suitable for their complex use case. The company aims to provide not just a software application, but a solution that adds value to its customer's business.

1.5 Work Culture and Company Values

The Work culture at Humble is fast paced, challenging, and a whole lot of fun. Whether at your workstation, in a virtual meeting with another part of the world, or taking a coffee break with friends, you will always find yourself collaborating, sharing new ideas. Growth is valued and high paced as the size of the company is small and the opportunities ample. At Humble, the following values have been observed and followed by the employees:

- Integrity and Honesty
- Passion
- Accountability
- Big Challenges
- Open and Respectful
- Self-critical

Chapter 2

Introduction

With the country turning smarter every day, it's only fair that its citizen turn even smarter. In the time where carrying documents has become almost obsolete, when it comes to carrying prescriptions it becomes even more annoying. "CARDIGRAM" aims to give its users a wide range of functionality in the medical world, making it their most utility companion. The platform conferences between the doctors and their patients from the very start to ensure full time assist for those in need. The main objective of "CARDIGRAM" is to reduce the communication distance in the medical world where even the slightest of seconds count.

2.1 Document Purpose

The intended readers of this document are the developers of the system, the testers and the consumers or clients that will use the tool on a day to day basis. Any changes incorporated in the document will be added in the last section for the reference of the developers and its users.

2.2 Product Scope

The client will get the following functionalities from the product:

- In hand portable database of patients for the reference of a large database of doctors.
- Personalized hub for each patient with records of his/her visitations and all the prescriptions and reports.Figure 2.1
- Setting up appointments with doctors.
- Scalable charts and graph representation for test values of the patients to facilitate easier knowledge outlook and more robust overview.
- Real time patient-doctor, doctor-doctor, group chat.

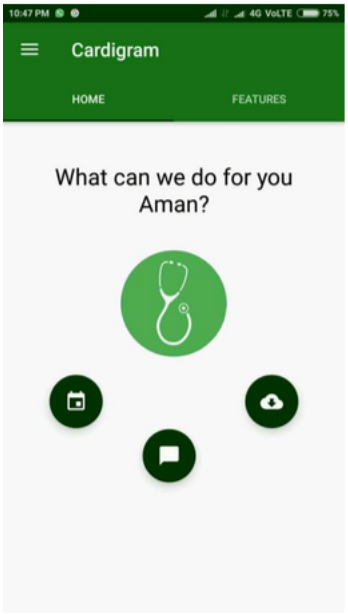

Figure 2.1: User's home page.

2.3 Intended Audience and Document Overview

The set of stakeholders include the team members, the overseeing instructors and potential user of the tool.

- Team members can use this document for detailed understanding of the requirements that are needed to be met by the application from both

user centric and design centric viewpoints.

- Instructors can use this document to access the level of detail the group is working from to guide development.
- The document can be used as an agreement form for the tool being developed.

2.4 Libraries Used

- Android Design Library 25.1.1
- Firebase Realtime Database
- Firebase Cloud Storage
- Firebase Authentication
- Firebase Analytics
- MP Android Charts
- Glide and Picasso For Image Rendering
- Google Places API

Chapter 3

Overall description of the application

3.1 Product Perspective

This product is an extension to already existing primitive Hospital Management Systems in the market. It is not a component of a larger system. The highlights added in the product are the first of its types to have been included in an integrated platform.

The idea behind the product is to build a tool that helps accessing patient's documentations on the go along with creating a one stop platform for the doctors and the patients to create an automated world for themselves which is always in there touch space. The product will help the hospitals, clinics and working professionals in this field stay connected with their patients and also the patients to stay updated about their physical well being without misinterpreting the data they usually find in their reports and prescriptions.

The product with its existing features and most importantly the vast database on the cloud i.e Firebase will be able to revolutionize the medical world, with its users asking for nothing less. After its completion; your prescriptions, reports, doctors, References and others will just be a click away on your phone.

So no more hogging on internet for hours to find the best possible treatment for your disease, with Cardigram finding yourself the best ever you deserve will never be easier.

Cardigram uses cloud storage and server space from Google's Firebase which intern facilitates the chat module integrated within. The system will also interact with the profile of each individual user and give information on the recent activities performed using the tool and the past records. It will also give user a statistical and subjective analysis of the reports uploaded in the patient's personalized dashboard in the form of scalable charts which will help in providing a more robust overview of the reports to its owner.

Cardigram is very user-friendly, 'quick to understand' and reliable application for the above purpose and it's simple.

3.2 Product Functionality

- The user can sign up on Cardigram using his/her name, email, password and display picture. Figure 3.1
- Email verification has been added as means of verifying the user which would be causal to registering the user onto Cardigram.
- The user can either login by giving their email and password.
- The users can upload their medical data i.e. reports, prescriptions etc onto their profile which would in turn be available on the go using Google Firebase cloud storage.
- Patients can use the application to set appointment with doctors which would be shown as an upcoming event in their dashboard.
- The users can use the integrated chat module to chat with doctors, patients in an individual chat or a group chat which is an amalgamation of

all the patients registered under Cardigram's database. The chat module uses Google's Firebase cloud server to facilitate the chat servers.

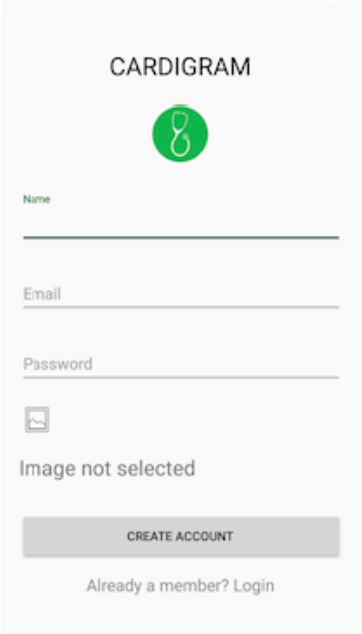

Figure 3.1: Application Sign-up page

3.3 Users and Characteristics

The users of the product will be ones in the medical industry along with the patients. A registered user would be able to create an account which will contain the information about his/her visitations, the prescriptions, lab reports, medicines. Along with this the registered users can take advantage of all the product functionalities mentioned in clause 3.2.

The application is made for the general mass, keeping simplicity in mind. The user doesn't need to have any prior technical knowledge for operating the application. The user has to sign up for the application by providing their credentials and he/she is good to go. The user credentials are encrypted and stored on the remote database securely. The layout of the application is very simple and follows the WYSIWYG paradigm.

Since it is reasonable to assume that the potential user has basic knowledge about the working of an online commercial service, we assume that our users will already be informed about basic functionality of the product. Also clear documentation and tutorials about the product feature will be provided.

3.4 Operating Environment

The application is designed for the Android operating system and supports all the android devices with minimum SDK version 15. Thus, it supports all the android versions from 'Ice Cream Sandwich' to 'Nougat'. It requires a minimum of around 100 MB of storage on the disk drive after installation. The app functions smoothly on any dual core processor device with a minimum of 1GB RAM and require a constant high speed internet connection, as the data is retrieved from the cloud. For the app to run properly, the mobile device should have GPS capabilities, as it retrieves the location of the hospital from a map view.

The program uses a custom made back-end for getting requested information about the patients, doctors etc. from its database stored on the cloud.

3.5 Design and Implementation Constraint

The application is cloud based, so it requires a constant high speed internet connection for its proper functioning. This is one of the major limiting factors of the application. This also limits the application from working in offline mode. Even, if the internet connection is slow, the application won't function properly and would take longer time to retrieve data from the database.

3.6 User Documentation

At the time of uploading of the application onto the store a specific write up would be provided acting as a user manual for better understanding of the application. Cardigram will have a user friendly GUI, and the user will easily be able to use this application, and understand each option and navigate around.

3.7 Assumptions and Dependencies

The application uses Google Firebase (PaaS) as the backend service provider. Two of the Firebase modules are used in the application, namely, the authentication module, and the real time database module. The authentication module is used for storing the user's credentials in an encrypted format in the cloud. The real time database is used to store the user profile details, event details and the group messages of an event.
The application also uses an external font library for custom fonts. The success of the system will depend on:

- Existence of internet service.
- Users must be comfortable with android apps and have enough conation to work with the application.

Chapter 4

Material Design

The entire front end of the application is made following the design paradigms of material design Library.

4.1 What is Material Design

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices. It is a unified system that combines theory, resources, and tools for crafting digital experiences. Android now includes support for material design apps. [2] To use material design in your Android apps, follow the guidelines defined in the material design specification and use the new components and functionality available in Android 5.0 (API level 21) and above. Android provides the following elements for you to build material design apps:

- A new theme
- New widgets for complex views
- New APIs for custom shadows and animations

4.2 Goal

The goal of using this library in the application is to create a visual language that synthesizes classic principles of good design with the innovation and possibility of technology and science.

It helps in developing a single underlying system that allows for a unified experience across device sizes figure 4.1. Mobile precepts are fundamental, but touch, mouse, and keyboard are all ïňĄrst-class input methods.

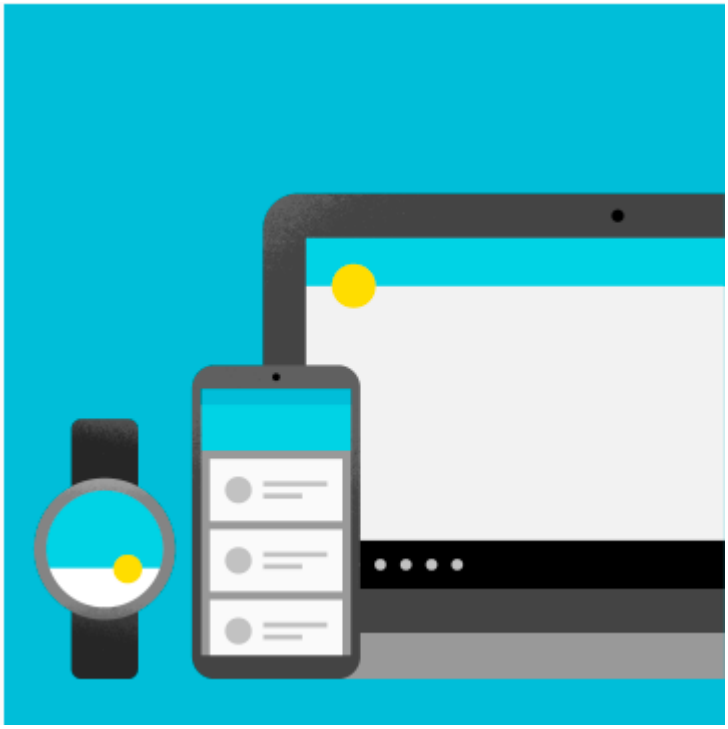

Figure 4.1: Material Design Unified experience across platforms

4.3 Principles

Material design is guided by print-based design elements such as typography, grids, space, scale, color, and imagery to create hierarchy, meaning, and focus that immerse the user in the experience. These layouts scale to fit any screen size, which simplifies the process of creating scalable apps.

Table 4.1 explains the principles of Material Design.

Table 4.1: Principles of Material Design

Principles		
Material is the metaphor	Bold, graphic, intentional	Motion provides meaning
A material metaphor is the unifying theory of a rationalized space and a system of motion. The material is grounded in tactile reality, inspired by the study of paper and ink, yet technologically advanced and open to imagination and magic.	The foundational elements of print-based design typography, grids, space, scale, color, and use of imagery guide visual treatments. These elements do far more than please the eye.	Motion respects and reinforces the user as the prime mover. Primary user actions are inflection points that initiate motion, transforming the whole design.

Chapter 5

Firebase

Firebase is a mobile platform that helps to quickly develop high-quality apps and grow the user base. Firebase is made up of complementary features that can be mix-and-match to fit the application's needs, with Google Analytics for Firebase at the core. [3] Firebase services can be explored and integrated in the app directly from Android Studio using the Assistant window.
Firebase gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users. Firebase is built on Google infrastructure and scales automatically, for even the largest apps. Firebase products work great individually but share data and insights, so they work even better together.It helps in giving users a simple, secure way to sign into the app, then monitor the onboarding process and find ways to improve it. It can also be used to implement a user-friendly chat feature so that the users can chat with each other in realtime without leaving the app. The backend of the application primarily uses Firebase to store all the data and provide bunch of functionalities to users using the following services.

- **Firebase Authentication**: To store user name and passwords.
- **Firebase Storage**: To store patients confidential data.
- **Firebase Real Time Database**: For chatting purpose.

Chapter 6

Requirements

6.1 External Interface Requirements

6.1.1 User Interfaces

- The application shall start with a splash activity which waits for the app to load and will prompt user for confirming all the main operations like termination, invalid log-in, etc.
- The UI/UX incorporates shared preferences which saves the state of the user and help avoid the same user to log in again and again.
- It will follow the basic GUI and error display standards.
- The first start includes an introduction to the application and tells the user what the application is meant for. The UI is consistent across the application.

6.1.2 Hardware Interfaces

The hardware requirements include a compatible Android device running Android Lollipop or above. There are no constraints on the screen size. However, it is required to have 1GB or above of RAM for smooth functioning of the

application. The application requires constant access to the internet since the data gets updated in real time.

6.1.3 Software Interfaces

The product is designed for Android operating system for smart phones. It uses the standard libraries provided by Google for Android. We have additionally used a library for introduction cards and fonts. The messages transferred through and forth the product are data items pertaining to the details of users and events and the chat messages between users. We have used the API provided by Firebase for communication between client and server. No data is shared between other applications of the system. The Glide library which is an image loading and caching library for Android focused on smooth scrolling [4] and Picasso Library which allows for hassle-free image loading in the application often in one line of code [5] has been used for rendering the images uploaded by the users. The MP Android chart library facilitates the generation of charts and graphs in the application in the analysis module of the patients reports. [6]

6.1.4 Communication Interfaces

The product requires constant access to the internet, since it access the back-end database - Firebase. The data is updated in real time. The data is transferred encrypted and uses HTTPS for communication. It is recommended to have a high speed connection.

6.2 Functional Requirements

6.2.1 System Initialization

Description and priority

This shows the user the front page of the Cardigram which is a splash activity after which is an activity which includes the create account for patients. The home screen consists of a typewriter text welcoming the user with a customized personalized text.

Priority: High

Stimulus and Response sequences

The action is triggered on the starting of the application. The user can select either to log in if he/she is already registered or create new account.

Functional Requirements

- The system should be working.

6.2.2 Create Account

Description and priority

Upon clicking the create account button the user is directed to a page where he/she will have to register by entering a unique username, password, email and other registration fields accordingly.

Priority: High

Stimulus and Response sequences

The action is triggered on the starting of the application. The user can select either to log in if he/she is already registered or create new account.

Functional Requirements

- The username should be unique and cannot be left empty.
- The password should not be left empty and should be of at least 8 characters.

- The email provided should be valid.
- The display picture provided by the user should be in valid format.

6.2.3 Set Appointments

Description and priority

This functionality can be used by user to setup an appointment with the doctor. The functionality makes use of the Google Places API.

Priority: Medium

Stimulus and Response sequences

The action is performed when the user clicks on the floating button corresponding to setting up appointment on the home screen or clicks on the appointment event in the Navigation Bar The window for setting up appointment is opened up and after setting up the appointment the event is shown in the users list of upcoming events.

Functional Requirements

- The user must select a doctor.
- The user should get a location from the google maps window prompt(Active internet connection and GPS must be enabled).
- The user must select a date and time for the appointment.

6.2.4 Uploading Reports

Description and priority

The users can upload all their medical data i.e. the reports, medicine prescriptions, and other visitation details on the application's cloud storage. The functionality uses glide and Picasso for dynamic rendering of the images uploaded by the user.

Priority: Medium

Stimulus and Response sequences

This functionality is rendered upon choosing the floating button corresponding to the documents upload or by clicking upon the upload event in the navigation bar. User can choose from Taking A Picture or Choose from Gallery options. The user can input the title and description and save the upload. The image is not compressed to maintain the quality. The images are stored in the firebase storage for higher quality. The image URL is stored for each user. There is a download card view for each uploaded document.

Functional Requirements

- The documents uploaded must be in valid format.
- The size of the documents uploaded must be within permissible limits.

6.2.5 Chat Interface For Patients

Description and priority

A chat interface for each individual patient to chat with other patients or doctors individually or in a group chat. The chat server utilizes the online cloud server of Google Firebase. The user can send images and select images from gallery.

Priority: Medium

Stimulus and Response sequences

This functionality appears on pressing the chat float button on the home screen or by selecting the chat option from the navigation bar. The option results in opening of a chat window with the display picture and name of the user on the top.

The images to be sent by the user are selected from the gallery converted to bitmap, compressed and scaled. The compressed and scaled bitmap is then converted to a string which is stored in the database, while on the receiver side the string is decoded and rendered to bitmap.

Functional Requirements

- The user must be connected to internet at all times.
- System must be able to detect that the user has joined the specific event and display the chat button accordingly.
- System must coordinate all other users' updates and display them in a sequential order that is common for everyone involved.

6.2.6 Generate Charts

Description and priority

This function will generate the charts and graphs taking in the values entered by the user. The type of charts can be chosen from a variety of charts and can be saved into gallery also.

Priority: Medium

Stimulus and Response sequences

Whenever a generate chart floating button is pressed or the generate chart event is selected from the navigation bar, the charts window is pop up. The window prompts the user to select from a variety of graphs and the user can also zoom in and out of the generated charts. The charts generated are in or user interaction as the width, height and other parameters can by varied by the user.

Functional Requirements

- The user must be a registered user.
- The values entered must be within the permissible limits i.e. the threshold.

6.3 Behaviour Requirements

6.3.1 Use Case View

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Figure 6.1 shows the use case view of the application.

6.3.2 Class Diagram

A class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Figure 6.2 shows the class diagram of the application.

6.3.3 Swimlane Diagram

It is a visual element used in process flow diagrams, or flowcharts, that visually distinguishes job sharing and responsibilities for sub-processes of a business process. Swim lanes may be arranged either horizontally or vertically. Figure 6.3 shows the swimlane diagram of the application.

6.4 Performance Requirements

The system must be interactive and the delays involved must be less .So in every action-response of the system, there are no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds, In case of opening databases, sorting questions and evaluation there are minor delays based on the amount of data that needs to be fetched and stored.

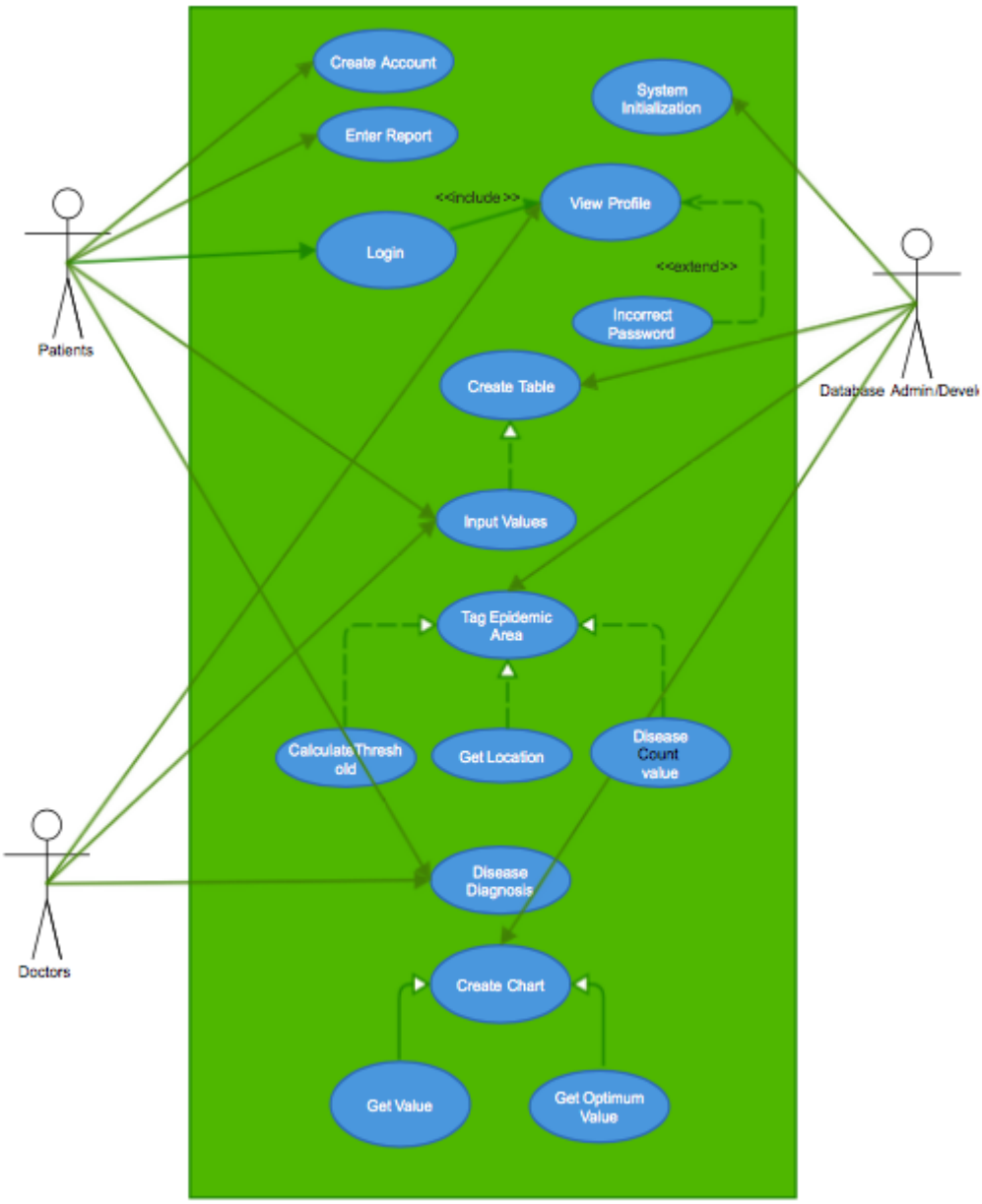

Figure 6.1: Use Case View

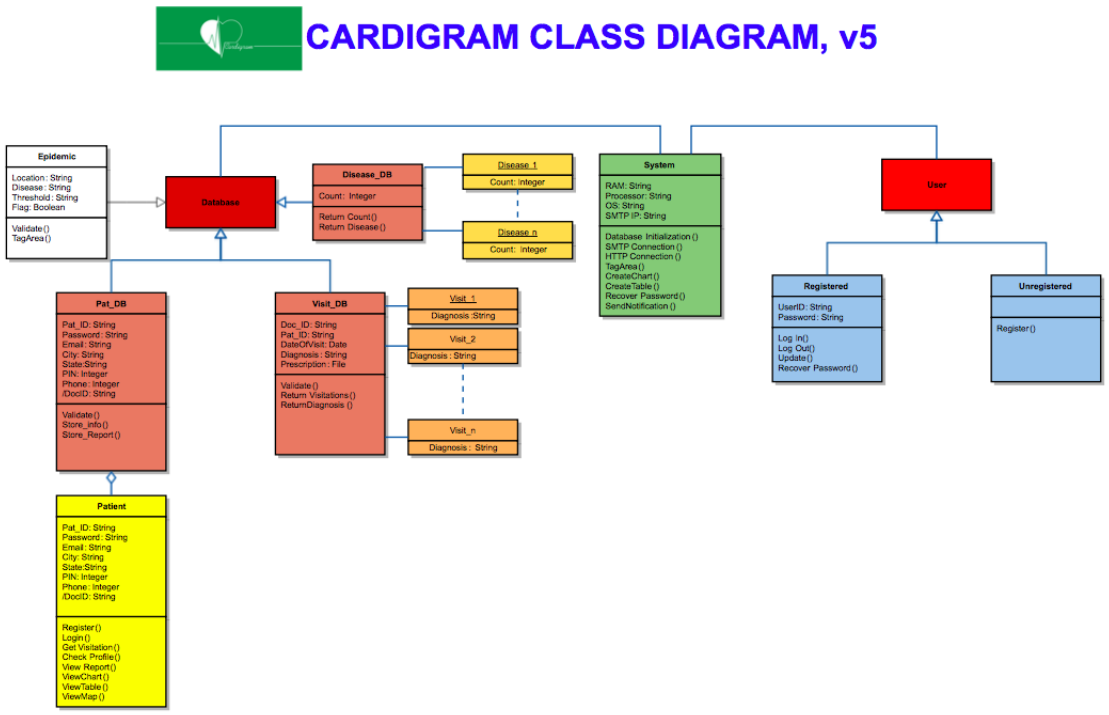

Figure 6.2: Class Diagram

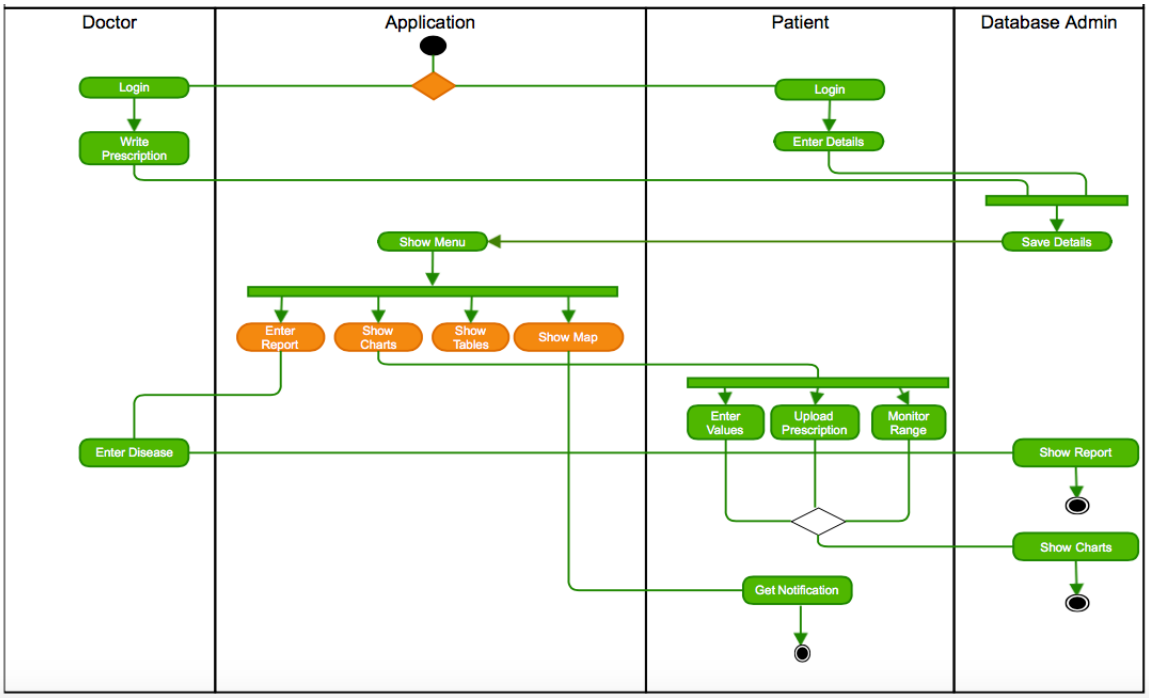

Figure 6.3: Swimlane Diagram

System must be able to fetch chat message updates within an order of 100ms. This is important is important due to the real-time nature of the chat interface. System must be able to fetch events in map view within an order of 2s. Such a big value is given to allow for a lot of events being taken place in the user's vicinity. System must be able to fetch the user profile within an order of 1s. Further changes made to the profile should propagate back to Firebase within 100ms in case the user kills the activity. All the views should be rendered almost instantaneously and further loading of items should be done later on.

6.5 Safety and Security requirements

- Information transmission should be securely transmitted to cloud without any changes in information.
- The passwords are not directly stored in the database, rather MD-5 (Message- Digest 5 algorithm) is used for generation of hashes thus ensuring the safety of the stored passwords and the privacy of the user.
- Keeping in mind the Doctor-Patient confidentiality terms, the patients records can be seen only by the patient himself and the appointed doctor.
- Unauthenticated use of this application is not to be permitted.
- User authentication should be done using firebase's authentication feature such as private user info and passwords are not visible to system administrators.

6.6 Software Quality Attributes

The cardigram's tools are reusable and can be extended to other such applications. The testing of this product requires traversing of all forms and ensuring any arbitrary error does not occur or is properly handled.

Availability: The system should provide 99% uptime as it's dependent only on Firebase and Google Maps, which are very reliable services.

Correctness: Firebase data should at no point be at an inconsistent state.

Learnability: App interface should be easy to use and learn.

Scalability: The app should be build to be scalable, i.e, able to handle large amount of people and events, and minimize data traffic of users to permit future growth. Scalability factor of at least login is preferred.

Affordability: The app designed is meant to be Open Source and completely free to use.

6.7 Database Requirements

Firebase real time database forms a hierarchical structure of nodes. It basically forms a JSON tree of objects. So, to insert values into the database, a child of a specific node is created and the instance of a class is stored in it. Even, while retrieving the values, an object is retrieved from the JSON tree, by specifying the class of the object, and hence the required object is retrieved from the data snapshot.

6.8 Reuse Objectives

The project is hosted on GitHub and can be reused as per the GNU GPL 3.0 license.

Chapter 7

Conclusion

Android application development is an interesting field where the concepts of java and android framework are tested. This application helped me learn a lot of things about material design library which helped in creating beautiful Front-end design for the application. It also helped me dive deeper into Google's firebase and learn how these services can be efficiently integrated within the android application to develop them quickly. This application is a new idea to make a patient's life easier. With is handy application, user can go to doctors at geographically varied locations without having to take care about their previous prescriptions as it is saved securely in our application. The user interface is extremely friendly and a bunch of other useful functionalities like chatting, appointment reminders and visual analysis of textual data have been provided. This internship was a great learning experience and I have used the knowledge gained to integrate new ideas and implement them in Android.

Appendices

Appendix A

Code

A.1 Login

```
public class Login extends AppCompatActivity {

    private static final String TAG = "LoginActivity";

    private static final int REQUEST_SIGNUP = 0;

    private FirebaseAuth mAuth;

    private DatabaseReference mdatabase;

    private FirebaseAuth.AuthStateListener mAuthListener;

    public int flag=0;

    EditText _emailText;

    EditText _passwordText;

    Button _loginButton;

    TextView _signupLink;


    public static int a = 0;

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_login);

        _emailText = (EditText)findViewById(R.id.input_email);
```

```
_passwordText = (EditText)findViewById(R.id.input_password);
_loginButton = (Button)findViewById(R.id.btn_login);
_signupLink = (TextView)findViewById(R.id.link_signup);


mAuth = FirebaseAuth.getInstance();
mAuthListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth
        firebaseAuth) {
        FirebaseUser user = firebaseAuth.getCurrentUser();
        if (user != null) {
            // User is signed in
            Log.d(TAG, "onAuthStateChanged:signed_in:" +
                user.getUid());
        } else {
            // User is signed out
            Log.d(TAG, "onAuthStateChanged:signed_out");
        }

    }
};
_loginButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        String email = _emailText.getText().toString();
        String password = _passwordText.getText().toString();
        signIn(email,password);
    }
```

```
    });


    _signupLink.setOnClickListener(new View.OnClickListener() {


        @Override
        public void onClick(View v) {
            finish();
            Intent intent = new Intent(getApplicationContext(),
                Signup.class);
            startActivityForResult(intent, REQUEST_SIGNUP);
        }
    });
}
private void signIn(String email, String password) {
    Log.d(TAG, "signIn:" + email);
    if (!validate()) {
        return;
    }
    if(!isOnline())
    {


        AlertDialog.Builder builder = new
            AlertDialog.Builder(this,R.style.AppCompatAlertDialogStyle);
        builder.setTitle("Connectivity Error.");
        builder.setMessage("Check your internet connection.");
        builder.setPositiveButton("OK", null);
        builder.show();
        return;
    }
```

```
final ProgressDialog progressDialog = new
    ProgressDialog(Login.this,
        R.style.AppTheme_Dark_Dialog);
progressDialog.setCanceledOnTouchOutside(false);
progressDialog.setMessage("Authenticating...");
progressDialog.show();
long delayInMillis = 8000;
Timer timer = new Timer();
timer.schedule(new TimerTask() {
    @Override
    public void run() {
        progressDialog.dismiss();
        //toast("Process is taking too long");
        Login.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if(flag==0)
                {
                    AlertDialog.Builder builder = new
                        AlertDialog.Builder(getApplicationContext(),R.style.App
                    builder.setTitle("Connectivity Error.");
                    builder.setMessage("Check your internet
                        connection.");
                    builder.setPositiveButton("OK", null);
                    builder.show();
                    return;
                }
            }
        });
    }
```

```
        }, delayInMillis);
        final AlertDialog.Builder builder1 =
                new AlertDialog.Builder(this,
                    R.style.AppCompatAlertDialogStyle);
        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new
                    OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
                        task) {
                        if(task.isSuccessful()) {

                            Log.d(TAG, "signInWithEmail:onComplete:" +
                                task.isSuccessful());
                            flag=1;

                                progressDialog.dismiss();
                                changeActivity();


//                Intent intent = new Intent(this, MainActivity.class);
//                startActivity(intent);
                        }
                        if (!task.isSuccessful()) {
                            flag = 1;
                            Log.w(TAG, "signInWithEmail:failed",
                                task.getException());
                            progressDialog.dismiss();
                            builder1.setTitle("Error.");
                            builder1.setMessage(task.getException().getMessage());
```

```
                    builder1.setPositiveButton("OK", null);
                    builder1.show();
                    return;
                }


            }
        });

}
private void changeActivity()
{
    finish();
    Intent intent = new Intent(this, MainActivity.class);
    startActivity(intent);
}
public boolean validate() {
    boolean valid = true;

    String email = _emailText.getText().toString();
    String password = _passwordText.getText().toString();
    TextInputLayout til = (TextInputLayout)
        findViewById(R.id.text_input_layout);

    if (email.isEmpty() ||
        !android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches())
        {
        til.setErrorEnabled(true);
        til.setError("Enter a valid email address.");
```

```
            valid = false;
        } else {
            til.setErrorEnabled(false);
        }

        TextInputLayout till = (TextInputLayout)
            findViewById(R.id.password_input_layout);
        if (password.isEmpty() || password.length() < 4 ||
            password.length() > 10) {
            till.setErrorEnabled(true);
            till.setError("4-10 alphanumeric characters.");
            valid = false;
        } else {
            till.setErrorEnabled(false);
        }

        return valid;
    }
    private boolean isOnline() {
        ConnectivityManager connectivityManager =
            (ConnectivityManager)
            getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetworkInfo =
            connectivityManager.getActiveNetworkInfo();
        return activeNetworkInfo != null &&
            activeNetworkInfo.isConnected();}
}
```

A.2 Charts

```
public class Charts extends ActionBarActivity {
   int flag = 0;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_charts);
      final BarChart chart = (BarChart) findViewById(R.id.chart);
      BarData data = new BarData(getXAxisValues(), getDataSet());
      chart.setData(data);
      chart.setDescription("Haemoglobin");
      int maxCapacity = 13;
      LimitLine ll = new LimitLine(maxCapacity, "Max Capacity");
      chart.getAxisLeft().addLimitLine(ll);
      chart.animateXY(2000, 2000);
      Button b = (Button)findViewById(R.id.save) ;
      b.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(final View view) {
            chart.saveToGallery("mychart.jpg", 85);
//          AlertDialog.Builder builder = new
    AlertDialog.Builder(getApplication(),R.style.AppCompatAlertDialogStyle);
//          builder.setTitle("Saved!");
//          builder.setMessage("Chart saved Successfully");
            flag=1;
         }
      });
      if(flag==1)
      {
```

```
        AlertDialog.Builder builder = new
            AlertDialog.Builder(getApplication(),R.style.AppCompatAlertDialogStyle);
            builder.setTitle("Saved!");
            builder.setMessage("Chart saved Successfully");

            builder.setPositiveButton("OK", null);
            builder.show();
    }
    chart.invalidate();
}

private ArrayList<BarDataSet> getDataSet() {
    ArrayList<BarDataSet> dataSets = null;
    ArrayList<BarEntry> valueSet1 = new ArrayList<>();
    BarEntry v1e1 = new BarEntry(13, 0); // Jan
    valueSet1.add(v1e1);
    BarEntry v1e2 = new BarEntry(13, 1); // Feb
    valueSet1.add(v1e2);
    BarEntry v1e3 = new BarEntry(13, 2); // Mar
    valueSet1.add(v1e3);
    BarEntry v1e4 = new BarEntry(13, 3); // Apr
    valueSet1.add(v1e4);
    BarEntry v1e5 = new BarEntry(13, 4); // May
    valueSet1.add(v1e5);

    ArrayList<BarEntry> valueSet2 = new ArrayList<>();
    BarEntry v2e1 = new BarEntry(12, 0); // Jan
    valueSet2.add(v2e1);
    BarEntry v2e2 = new BarEntry(13, 1); // Feb
    valueSet2.add(v2e2);
```

```
        BarEntry v2e3 = new BarEntry(14, 2); // Mar
        valueSet2.add(v2e3);
        BarEntry v2e4 = new BarEntry(11, 3); // Apr
        valueSet2.add(v2e4);
        BarEntry v2e5 = new BarEntry(12, 4); // May
        valueSet2.add(v2e5);
        BarDataSet barDataSet1 = new BarDataSet(valueSet1, "Normal");
        barDataSet1.setColor(Color.rgb(0, 155, 0));
        BarDataSet barDataSet2 = new BarDataSet(valueSet2, "Observed");
        barDataSet2.setColors(ColorTemplate.COLORFUL_COLORS);
        dataSets = new ArrayList<>();
        dataSets.add(barDataSet1);
        dataSets.add(barDataSet2);
        return dataSets;
    }

    private ArrayList<String> getXAxisValues() {
        ArrayList<String> xAxis = new ArrayList<>();
        xAxis.add("FIRST");
        xAxis.add("SECOND");
        xAxis.add("THIRD");
        xAxis.add("FOURTH");
        xAxis.add("FIFTH");
        return xAxis;
    }
}
```

Full code can be found at `https://github.com/Aman-Chopra/Cardiapp`

Appendix B

Application's screenshots

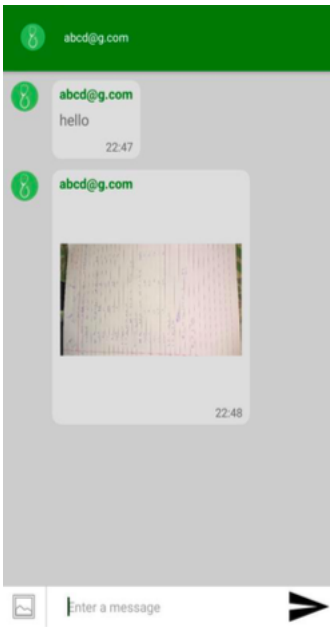

Figure B.1: Chat interface with Text and Images

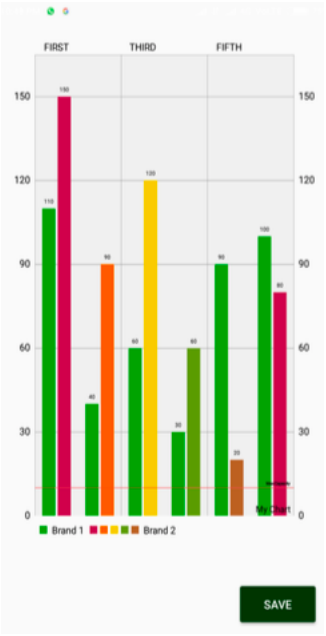

Figure B.2: Animated charts of patients data

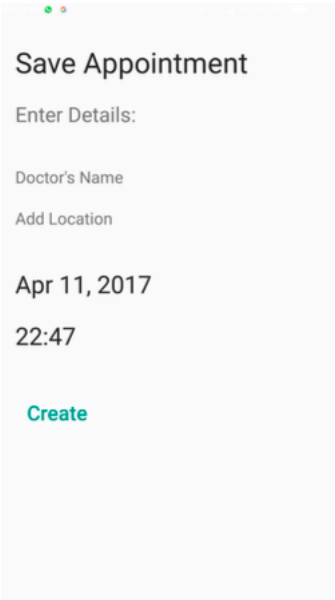

Figure B.3: Booking an appointment

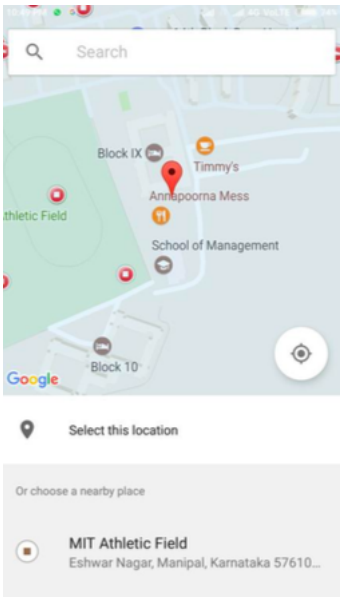

Figure B.4: Using Google map API to choose location

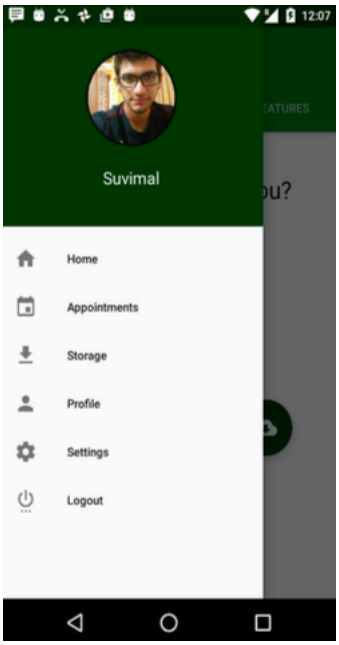

Figure B.5: User's Navigation pane

References

[1] Humble. [Online]. Available: http://humble.net.in/

[2] Material design. [Online]. Available: https://material.io/guidelines/material-design/

[3] Firebase. [Online]. Available: https://firebase.google.com/

[4] Glide. [Online]. Available: https://github.com/bumptech/glide

[5] Picasso. [Online]. Available: http://square.github.io/picasso/

[6] Mp android charts. [Online]. Available: http://theandroidlibrary.com/control/mp-android-chart