Technical

MANUAL

USER'S MANUAL

TABLE OF CONTENTS

Page #

1.0 GENERAL INFORMATION

A. GENERAL INFORMATION

1.1 System Overview

An intelligent aid for impaired individuals:

- A software system is a cloud based library for android mobile devices.
- Efficient data storage, query and Dynamic GUI generation solutions.
- User Doctors/patients/specialists
- System name or title: Med-e-form mobile application
- System category:
  - *Major application:*generates dynamic GUI corresponding to archetype files, provide facility of querying data, support for single patient and multi patient queries, storing user data submitted through.
- Operational status:
  - Partially Operational
  - Under development for large number of users and huge data

1.2 Project References

References that were used in preparation of this document in order of importance to the end user.

2.0 SYSTEM SUMMARY

B. SYSTEM SUMMARY

This section provides a general overview of the system written in non-technical terminology. The summary should outline the uses of the system in supporting the activities of the user and staff.

2.1 System Configuration

System is designed to provide a cloud based application which can generate dynamic and interactive GUI for mobile devices, persist and query data in cloud based databases. So, we developed a clinical mobile application and a library which binds multiple features of clinical process, i.e. investigations, observation, examinations, clerking, history and assessment.

2.2 Data Flows

Once user includes this library into their project, they can create GUI according to them. Developers can use our library function to make their work easy.

2.3 User Access Levels

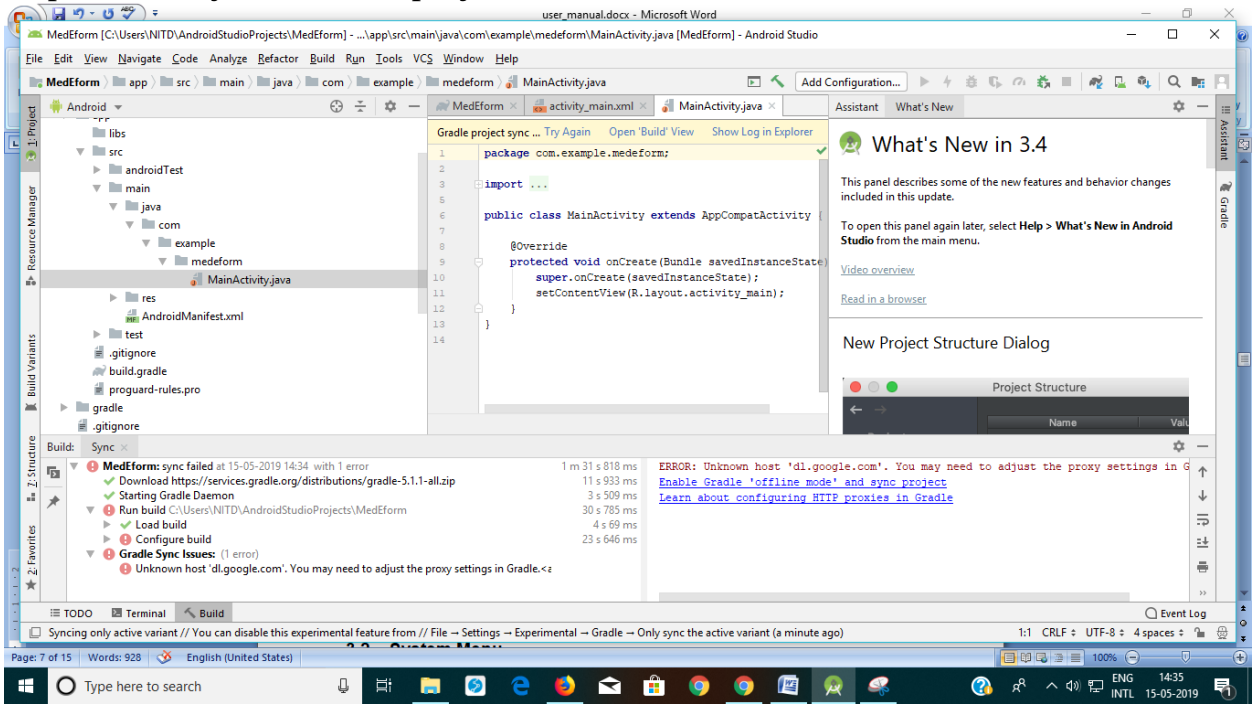
The Primary user and family member and authorized user may be able to add data and information, can see results of data fetching queries and get access to e-forms.

3.0 GETTING STARTED

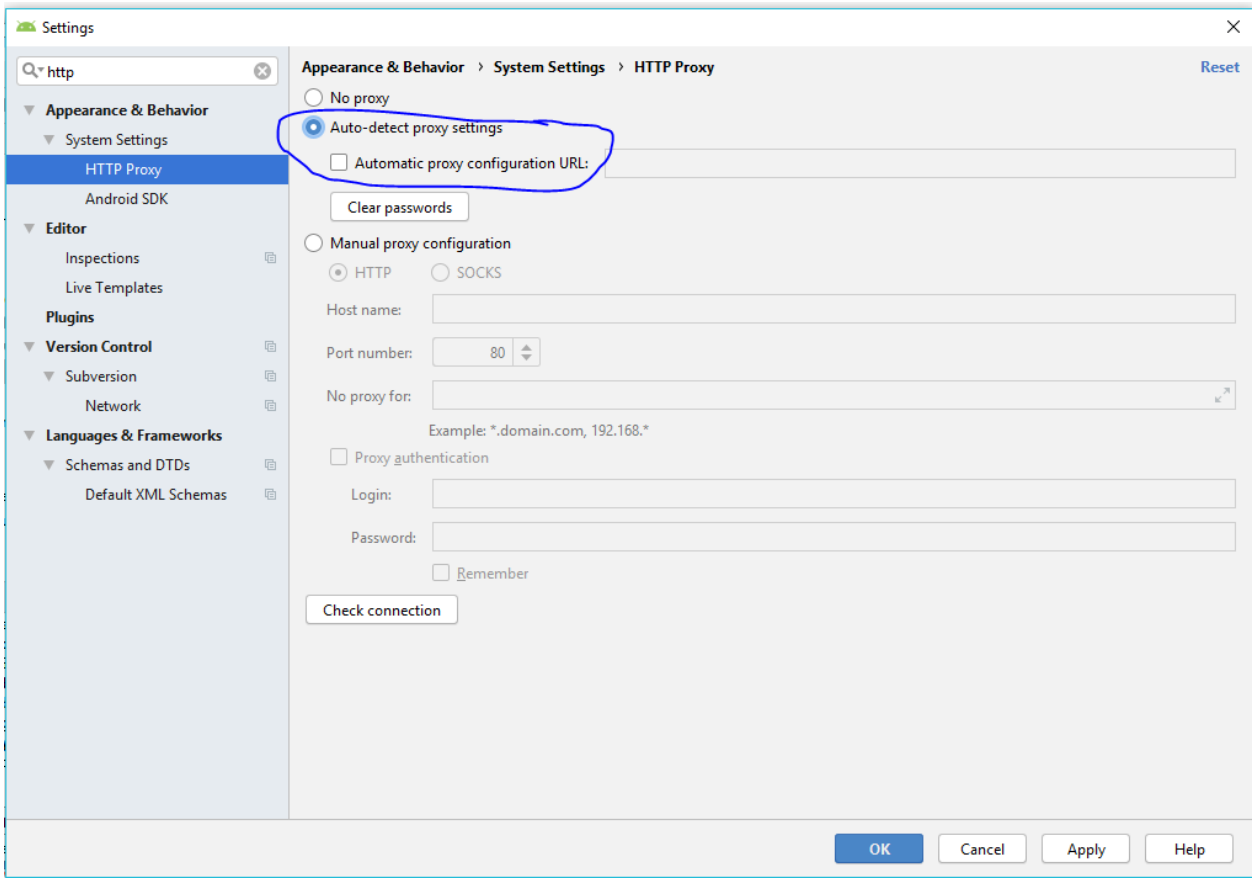
C. GETTING STARTED

This section provides a general walkthrough of the system from initiation through exit. The logical arrangement of the information shall enable the functional personnel to understand the sequence and flow of the system.

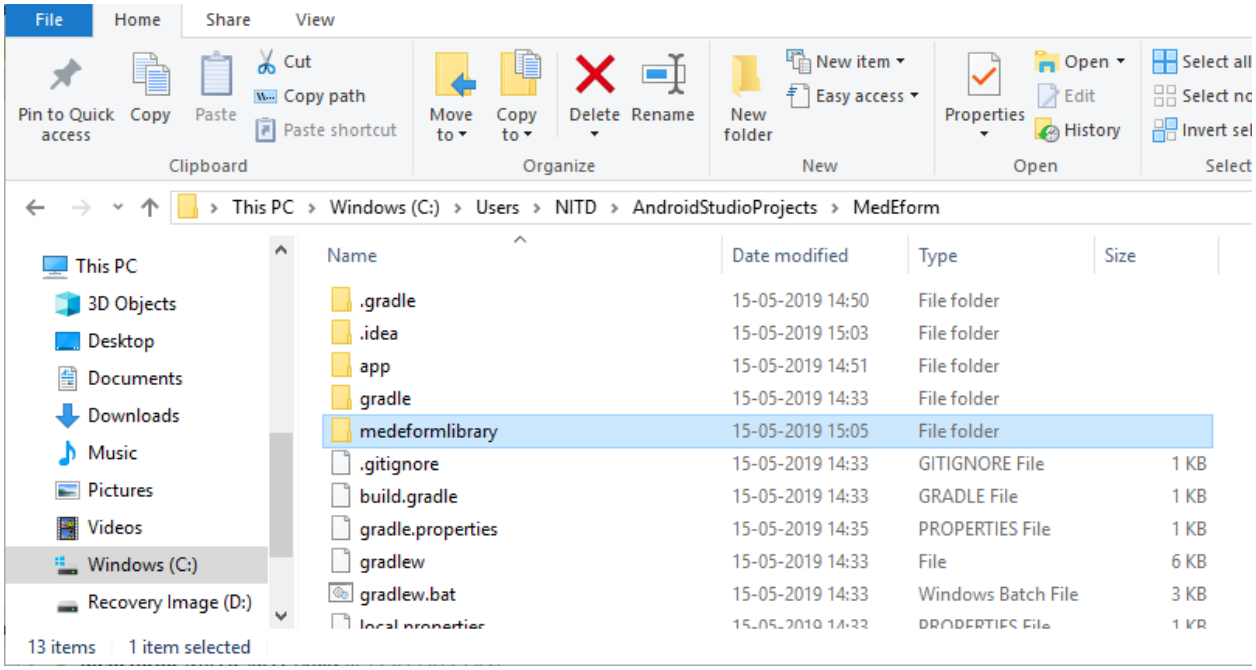
Step1. Download android studio. https://developer.android.com/studio/
Step2. Download JDK version 8.*.*
Step3. Create your android project first.

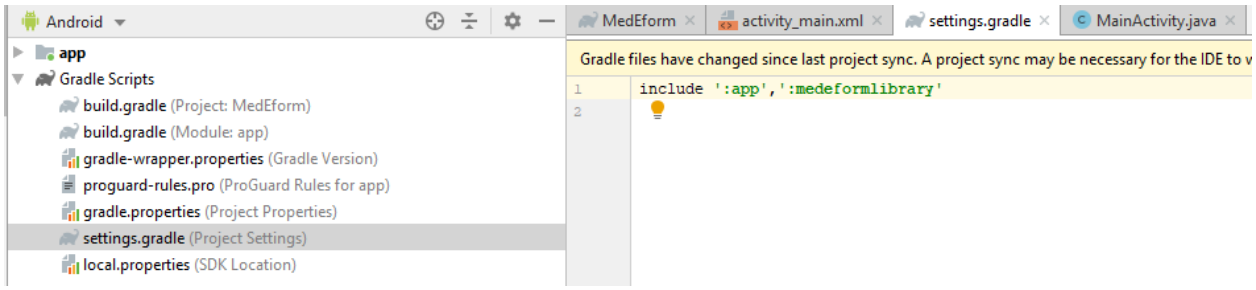


In order to resolve

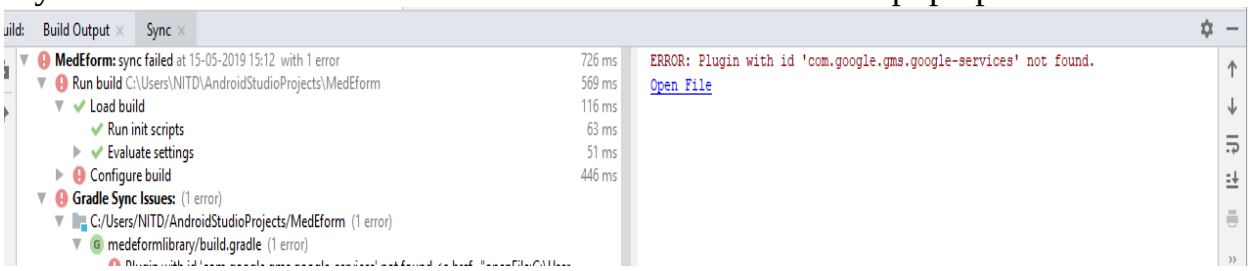


Step 4. To include 'MedEForm Library' from github, download
https://github.com/tanujyadav97/MedEFormLibrary
Unzip the folder and copy this (medeformlibrary) inside root directory of your project folder.
(In case you want to create your own firebase account)





If you have not created firebase account then this error will pop up



OR

If you want to persist data on cloud-based database (Firebase account created earlier), add the snippet shown below in your root 'build.gradle' at the end of repositories.

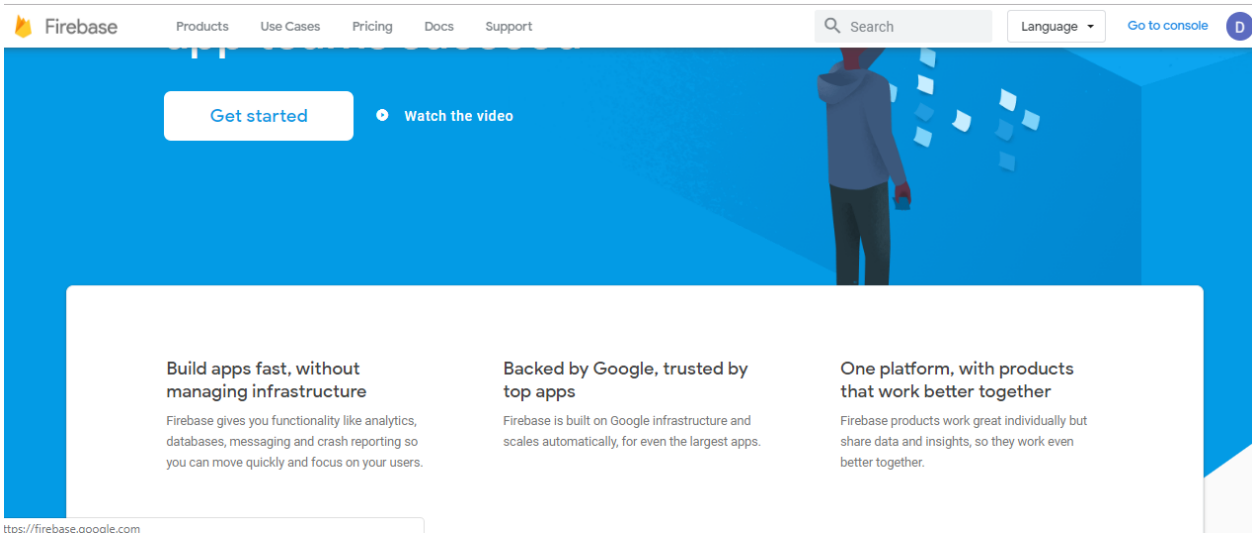
```
allprojects {
                repositories {
                        ...
                        maven { url 'https://jitpack.io' }
                }
        }


dependencies {
                implementation 'com.github.tanujyadav97:MedEFormLibrary:1.3'
        }
```

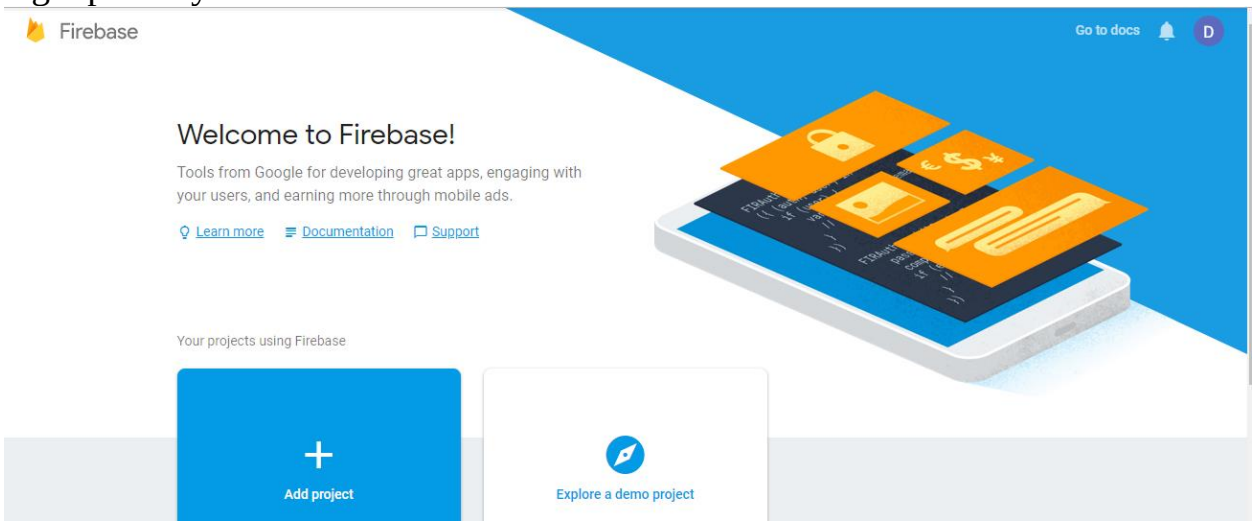
Step 5. Build your project again. Now you can see MedEForm library in your project.

Step6. Creating Database and connecting it with 'MedEForm' library.

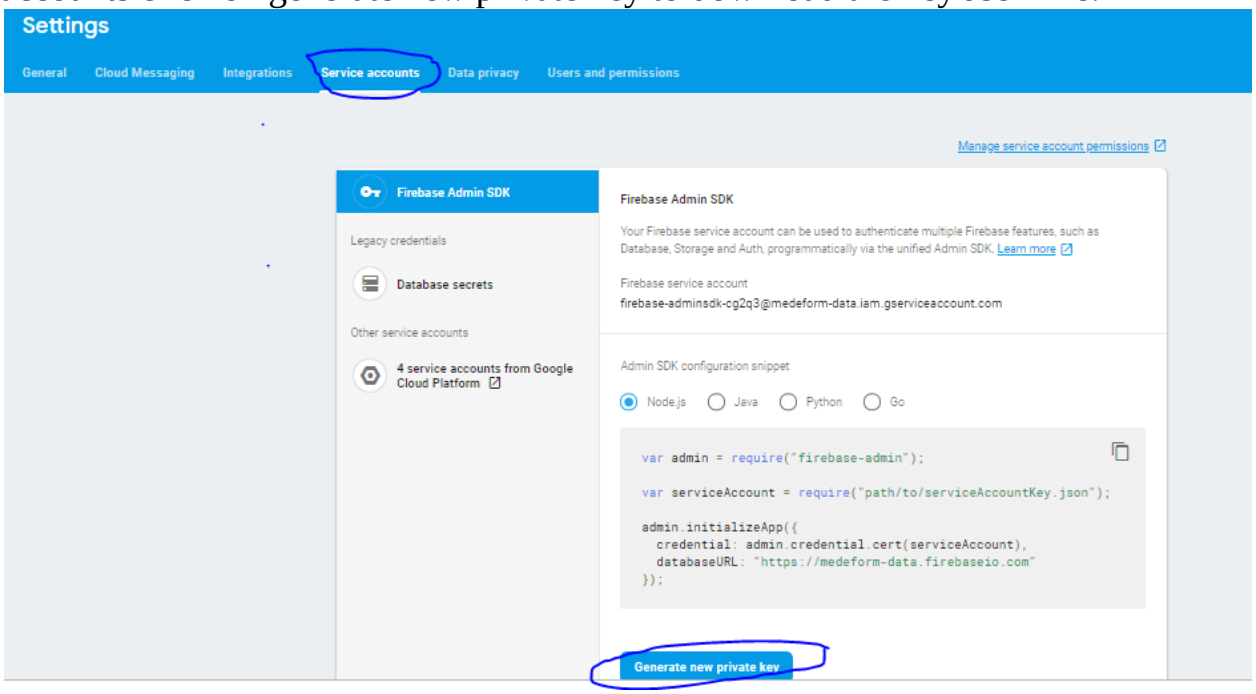
- Go to firebase console.



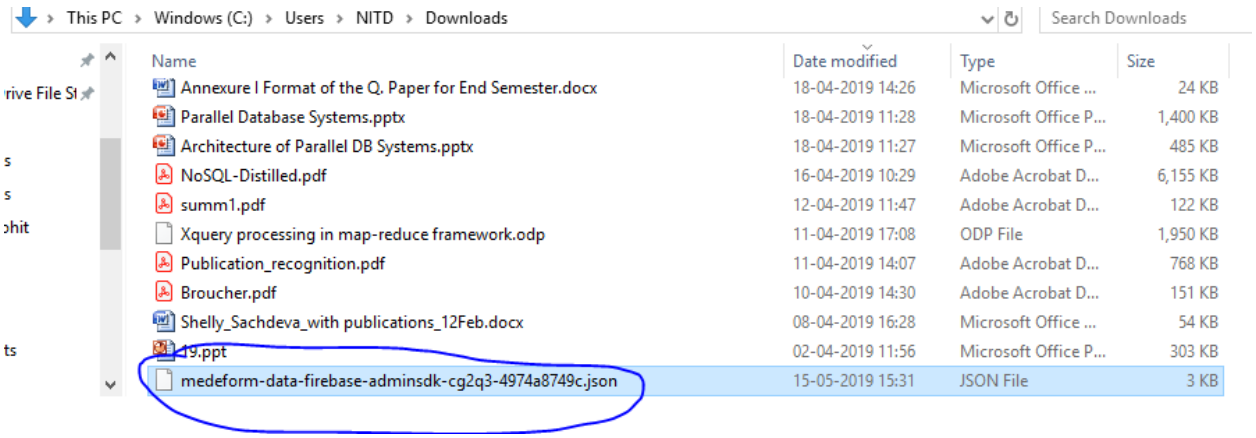
- Signup with your Gmail

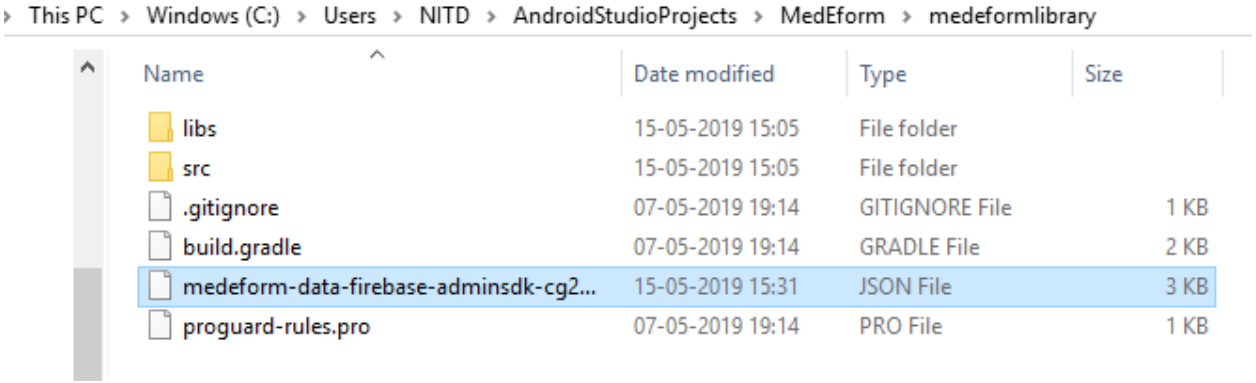


- Create a project in this and then go to project setting and inside service accounts click on generate new private key to download the key Json file.

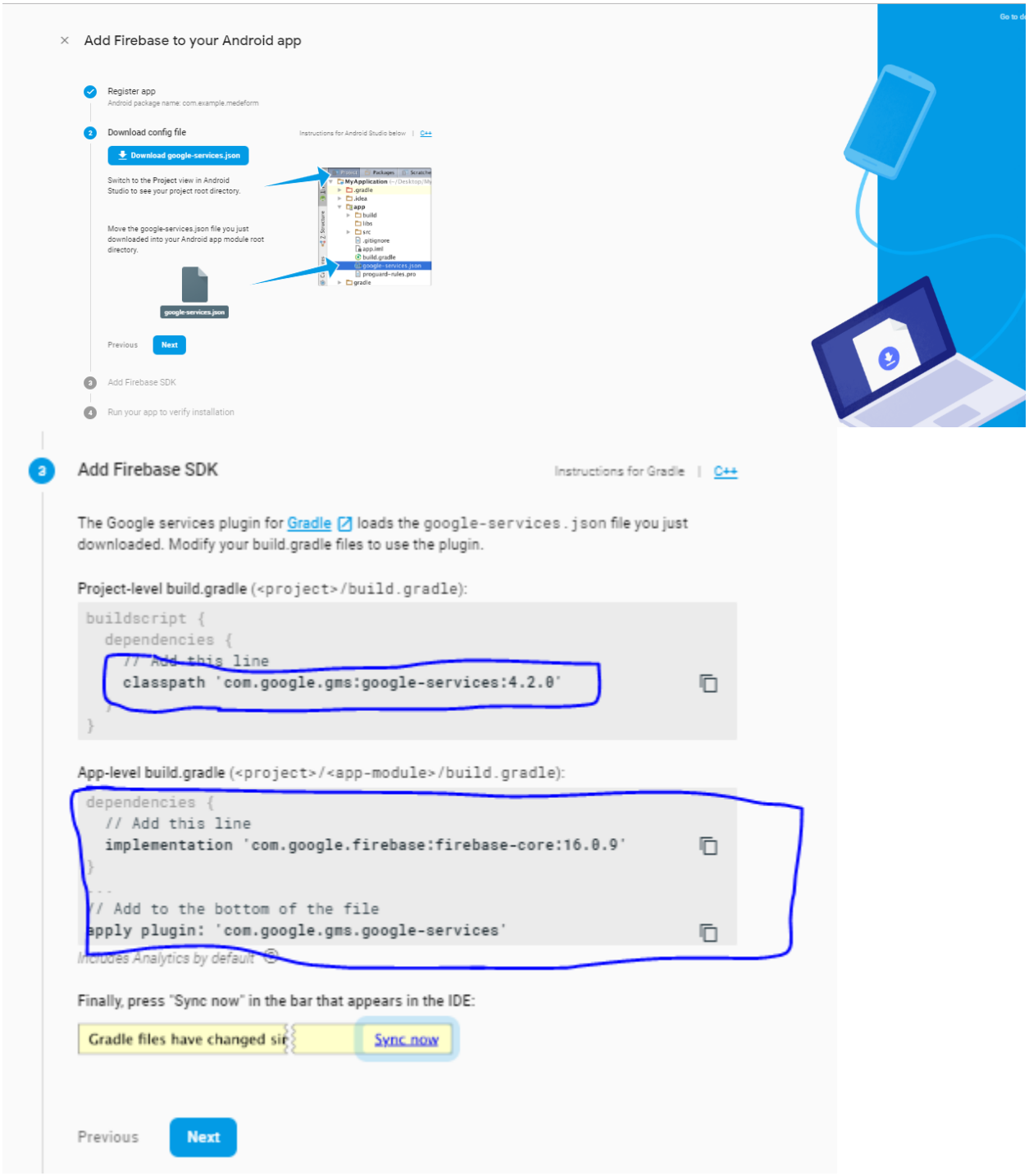


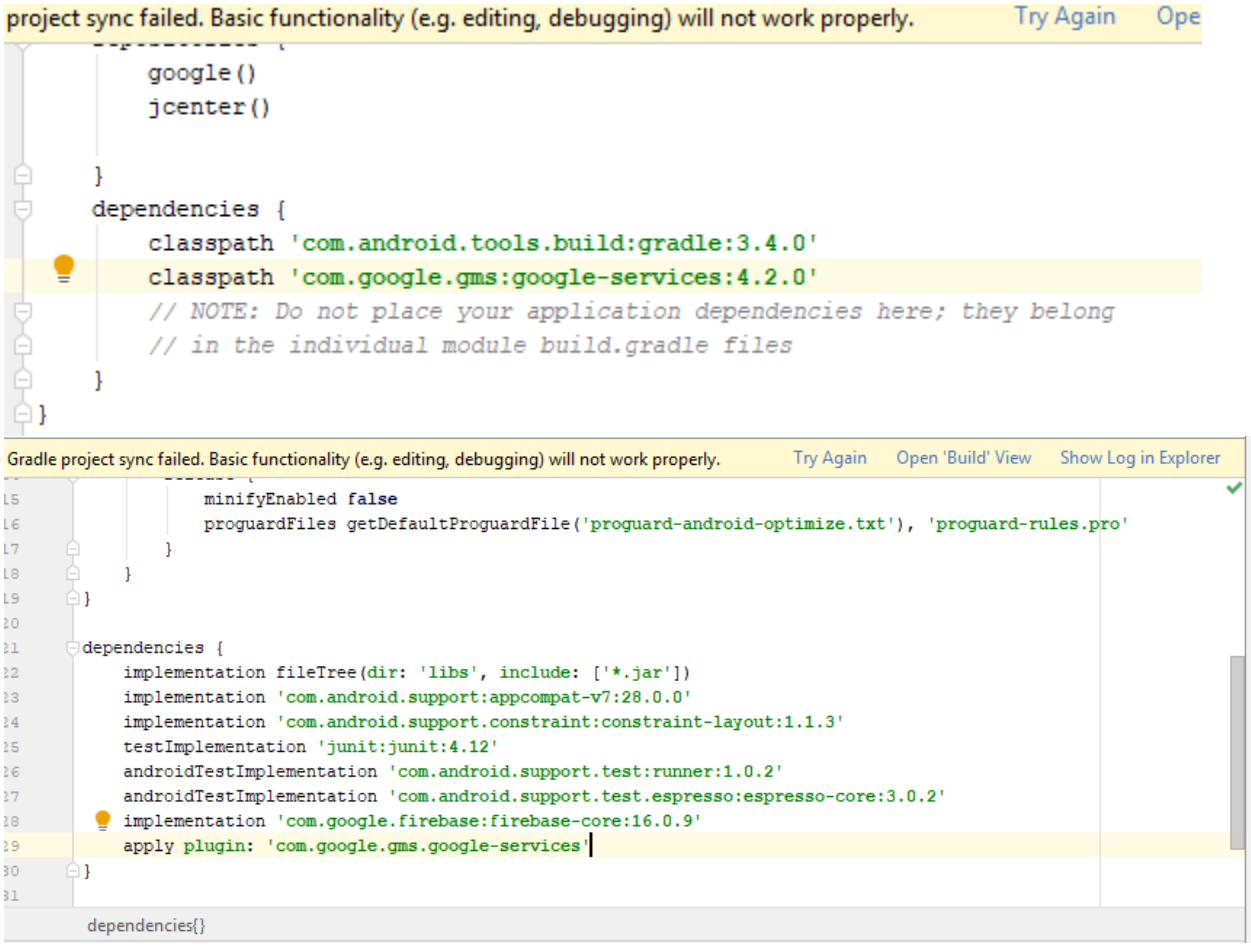
- Include this file into MedEForm folder.





- And connect your android studio project with Firebase.


Add Firebase to your Android app
Register app
Android package name: com.example.medeform
Download config file
Instructions for Android Studio below | C++
Download google-services.json
Switch to the Project view in Android Studio to see your project root directory.
Move the google-services.json file you just downloaded into your Android app module root directory.
google-services.json
Previous
Next
Add Firebase SDK
Run your app to verify installation
3
Add Firebase SDK
Instructions for Gradle | C++
The Google services plugin for Gradle loads the google-services.json file you just downloaded. Modify your build.gradle files to use the plugin.
Project-level build.gradle (<project>/build.gradle):
buildscript {
dependencies {
// Add this line
classpath 'com.google.gms:google-services:4.2.0'
}
}
App-level build.gradle (<project>/<app-module>/build.gradle):
dependencies {
// Add this line
implementation 'com.google.firebase:firebase-core:16.0.9'
}
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
Includes Analytics by default
Finally, press "Sync now" in the bar that appears in the IDE:
Gradle files have changed sir
Sync now
Previous
Next



- And now data will be stored in both the schemas automatically.

3.2 System Menu

3.2.1. Dynamic Form Generation

To use this functionality user has to create instance of Generate Form class like this and call make form function for that archetype.
Steps:

- Copy your archetype file to local storage of your phone.
- Create an instance of GenerateForm class.
- Create a linear layout in XML.
- Arguments are passed in this order, current activity, and path to archetype file, name of archetype file and Type of archetype, Id of linear layout, patient Id for which user will store data.

```
GenerateForm gf = new GenerateForm(MainActivity.this, this,
    "Android/data/com.android.hackslash.openehr",
    "openEHR-EHR-OBSERVATION.body_temperature.v2.adl", "OBSERVATION",
R.id.llroot, "patientid");

gf.makeForm();
```

After the form will be generated user can fill the data and can click on submit button and data will be submitted to fire-store.

3.3. Query Interface

To use query interface you need to follow following procedure

- First need to select type of query i.e. it is patient specific or archetype specific.
- If it is patient specific query then you need to enter patient ID, if it is archetype specific then you need to enter archetype names.
- Next you need to enter the fields that you want to fetch(For example see in snapshots below)
- We provided 6 selection conditions:
    - WHERE_EQUAL_TO
    - WHERE_LESS_THAN
    - WHERE_GREATER_THAN
    - WHERE_LESS_THAN_OR_EQUAL_TO
    - WHERE_GREATER_THAN_OR_EQUAL_TO
    - WHERE_ARRAY_CONTAINS
- Click on submit button to get the result of query. Which will be shown below submit button.

For developers to create User interface. They can use same library and create their own query interface and our cloud based query manger can be accessed easily. Following is the procedure by which developers can access our library function.

Steps to use Query interface as a library.

- Creating Database and connecting it with library.
    - Go to firebase console.
    - Signup with your Gmail.
    - Create a project in this and then go to project setting and inside service accounts click on generate new private key to download the key Json file.
    - Include this file into MedEForm folder.
    - And connect you android studio with Firebase.
    - And now data will be stored in both the schemas automatically.

To use library for query managing

1. **create instance of Query class**

   Queries qry = new Queries();

2. **create an array of type QueryObj to hold the condition**

```
ArrayList<QueryObj> queries = new ArrayList<>();
queries.add(new QueryObj(queryType.WHERE_GREATER_THAN, "Hb Count", 0));
queries.add(new QueryObj(queryType.WHERE_LESS_THAN, "Hb Count", 300));
```

//if it is patient specific query

```
qry.selectPatientQuery(patientID.getText().toString(), queries,
                    new QueryCallback() {
        Public void onCallback(HashMap<String, HashMap<String, Map<String, Object>>> data,
String exception) {
                        if (data == null) {
                           Log.e(TAG, exception);
                           output.setText(exception);
                        } else {
                           Log.d(TAG, data.toString());
                           output.setText(data.toString());
                        }
                     }
});
```

//for archetype specific queries

```
qry.getArchetypeTimestampQuery(archetypeName.getText().toString(),
                    startTimestamp.getText().toString(), endTimestamp.getText().toString(), new
QueryCallback() {  @Override
                      public void onCallback(HashMap<String, HashMap<String, Map<String,
Object>>> data, String exception) {
                        if (data == null) { Log.e(TAG, exception);
                           output.setText(exception);
                        } else {Log.d(TAG, data.toString());
                        output.setText(data.toString());
      }
}});
```

3.4 Exit System

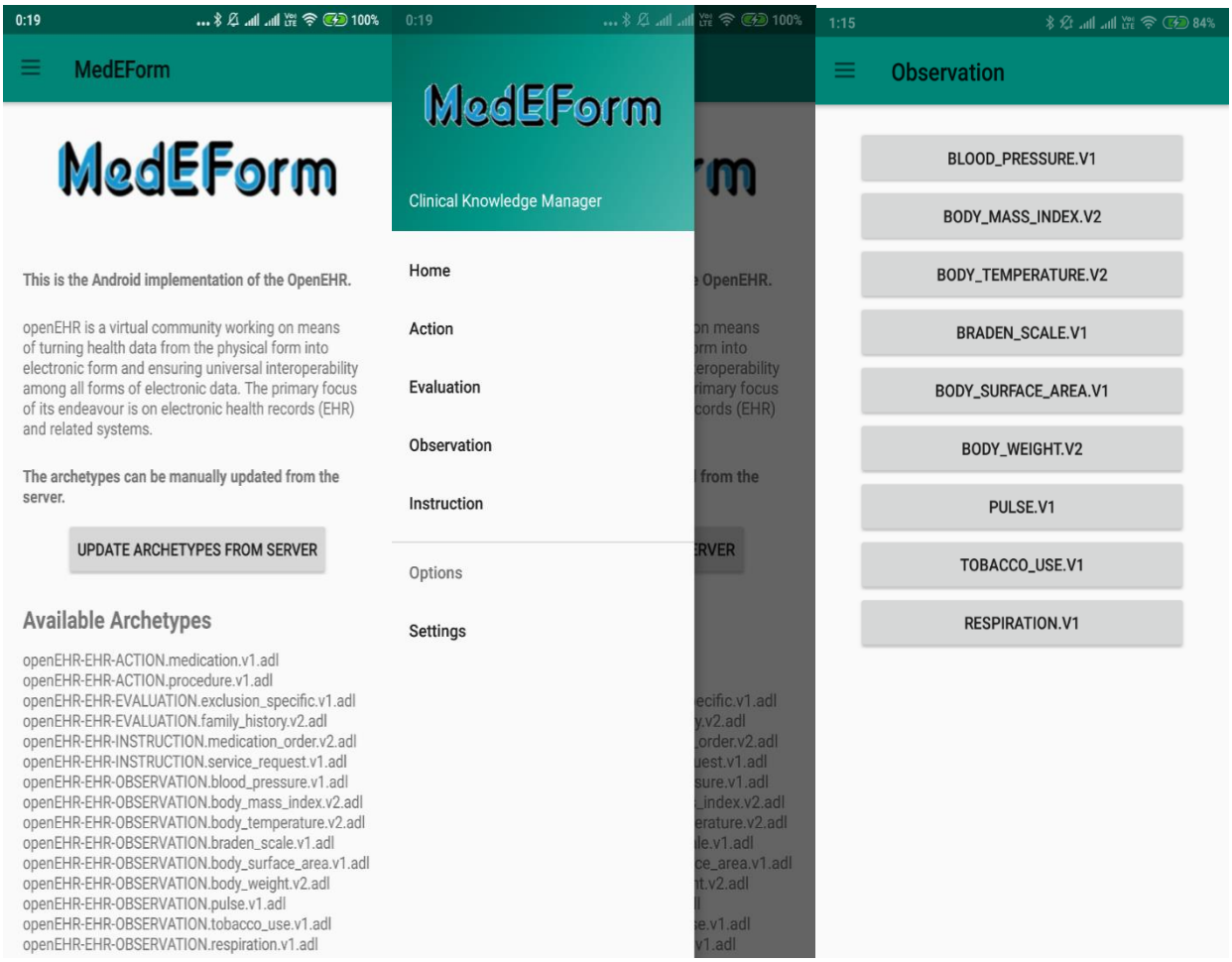
Click on logout button and you will be out of the system.

10.0 APPENDIX

A. APPENDIX

Screen Shots:

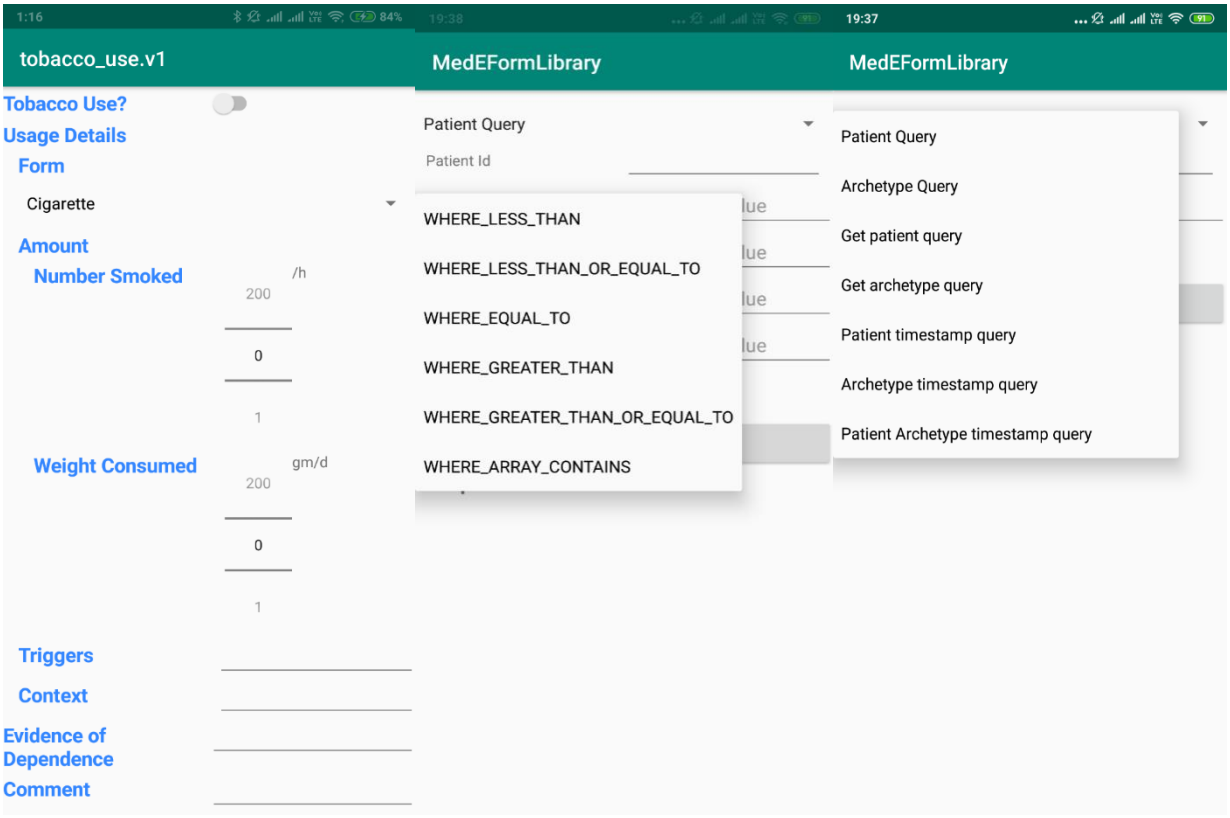
Location Top Menu *Main Menu Screen* *Observation tab*



Top Menu Hierarchy:

Screen Shots:

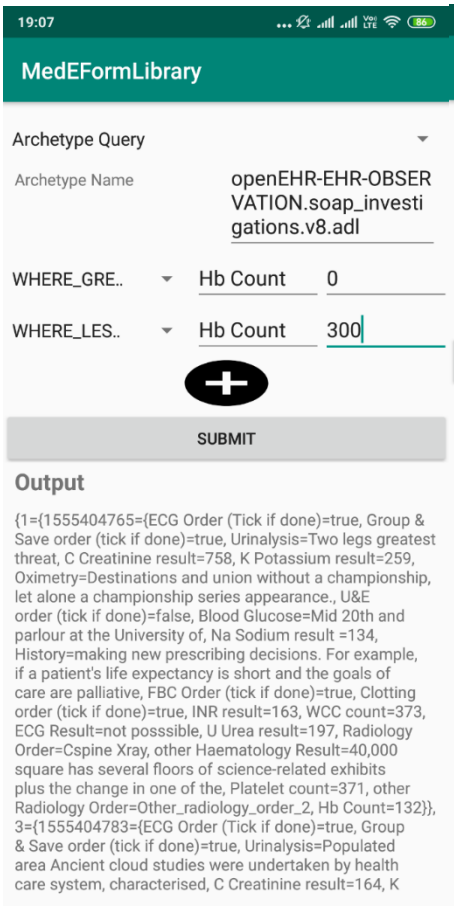
Generated form Query InterfaceQuery Interface



Medical Sub-Systems:

Screen Shots:

Query interface and results fetched



results fetched