Medical Analytics Project

niRog App | EPICS Review 1

Table of contents

Team Members

Somardh Jaiswal
20BCE10880

Aaditya Sreenivasan
20BCE10738

Akshat Mehrotra
20BCE10246

Yaksh Goyani
20BCE10483

Ankit Deb
20BCE10495

Shivam Dubey
20BCE10133

Introduction - niRog App

The application is named “niRog”. It is mainly focused on uplifting Health and Medical services to citizens of our nation. It is focused on collecting insights and recommendations based on real-time data, prescriptions, etc.

Throughout our daily lives, we’ve learned about many incidents which have occurred and still occurring in many parts of our country where citizens face unnecessary encounters and feedback while they were trying to seek health facilities in government hospitals: be it standing in a queue, be it not being able to receive proper feedback, etc.

Motivation

Cashless, queueless, worry-less

- The application aims to eradicate the endless queues which is common among many government hospitals and dispensaries.
- Cashless payments are convenient for both the doctor and the patient.
- By introducing the above procedures, a revolutionary medical system will be established

Problem Identification

- niRog application comes forward with the helping motive to provide citizens receive sufficient Healthcare.
- Initally, the portal will lead them to register their accounts by filling the required details. Then it’d lead them towards their profile dashboard.
- The portal will also have specialized corners for Doctors and Admins. With the given permission of the user, medicine and appointment details will be stored in a reliable database to provide better analytics and services.
- niRog is constantly dependent on Google Location Services so that it can function its features properly.
- One of the features it includes has, to locate nearby Government Hospitals which are linked with niRog to work seamlessly.

Knowledge regarding Project Modules

Regarding the initial phases of Front-End(XML) part of our Project:

Using ScrollView

A view group that allows the view hierarchy placed within it to be scrolled. Scroll view may have only one direct child placed within it. To add multiple views within the scroll view, make the direct child you add a view group, for example LinearLayout, and place additional views within that LinearLayout.

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    tools:context=".RegisterActivity">
```

Using Relative Layouts and Constraint Layouts

RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements or in positions relative to the parent RelativeLayout area.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="180dp"
        android:background="#5384FF">
```

Knowledge regarding Project Modules

Regarding the initial phases of Front-End(XML) part of our Project:

```
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/button_login_LA"
    android:layout_width="180dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/RL_pwd"
    android:layout_centerHorizontal="true"
    android:layout_marginStart="70dp"
    android:layout_marginTop="30dp"
    android:background="@drawable/btnbgthree"
    android:text="@string/login"
    android:textColor="@color/white" />
```

Using AppCompatButtons instead of Default Buttons

A Button which supports compatible features on older versions of the platform, including:

Allows dynamic tint of its background via the background tint methods in androidx.core.view.ViewCompat.
Allows setting of the background tint using backgroundTint and backgroundTintMode.
Allows setting of the font family using fontFamily

```
<androidx.swiperefreshlayout.widget.SwipeRefreshLayout
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/swipeContainer"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

Using SwipeRefresh Layout

We have tried to use SwipeRefresh Layout here because whenever the patient/user joins the activity, he or she will be able to refresh the contents of the view via a vertical swipe gesture. This will let the user stay updated.

Knowledge regarding Project Modules

Moving onto the initial phases of the Back-End (Java Logic and Firebase Database) parts of our Project:

Initializing views and objects of our Project

Here, we have taken our first step in working with Logic and Connections in our Main Activity, where we initialize our views and objects. They will be further recognized here from their submitted attribute IDs through the XML codebase by using R.findViewByID attribute where R denotes to the resources in our project.

```
ublic class MainActivity extends AppCompatActivity {

    Button loginButtonMain;
    TextView registerMain;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

Importing the sufficient set of ViewModes and Modules for our MainActivity

It is highly important for our Project to have a good exposure on the modules and libraries that we will be using to enhance the performance and services of our application

```
package com.epics.medanalytics6;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

Knowledge regarding Project Modules

Moving onto the initial phases of the Back-End (Java Logic and Firebase Database) parts of our Project:

```
loginButtonMain.setOnClickListener(new View.OnClickListener()
    @Override
    public void onClick(View v) {

        Intent intent = new Intent( packageContext: MainActivity.
        startActivity(intent);
```

Setting up onClickListeners to redirect users/patients to their respective areas

Here, we have used a few onClickListeners on two of our Buttons: Login and Register Button so that users can freely choose whether they wish to register a new account or login to their original page every time they initialize installing and then open the interface of our software. We have used intents here.

```
EditText etRegFullName, etRegEmail, etRegDOB, etRegMobileNumber
ProgressBar progressBarReg;
RadioGroup RGRegGender;
RadioButton RBRegGenderAfterSelectionBtn;

FirebaseAuth mAuth;
FirebaseUser mUser;
```

Setting up Register Activity for Patients/Users

Here, moving on to the register activity page of our project, we have tried to introduce new views and objects such as the 4 kinds of edit texts which are based on details regarding the user(example- name, mobile number, etc.). We have also tried to implement a radio button in here for accepting the submissions of their gender. Then, we have introduced two important aspects of our Database registration, i.e. FirebaseAuth and FirebaseUser.

Knowledge regarding Project Modules

Moving onto the initial phases of the Back-End (Java Logic and Firebase Database) parts of our Project:

Using FirebaseUser and FirebaseAuth

The Firebaseuser object represents a user account that has signed up for our app in our project

The Firebase Authentication SDK provides methods to create and manage users that use their email addresses and passwords to sign in. Firebase Authentication also handles sending password reset emails.

```
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;
import com.google.firebase.auth.FirebaseAuthUserCollisionException;
import com.google.firebase.auth.FirebaseAuthWeakPasswordException;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.UserProfileChangeRequest;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
```

Implementing the correct set of dependencies with latest versions in the Build Gradle under Scripts

Here, we have used some important dependencies for our Project, such as Material Design and Firebase Database. We have also used AppCompat and a few other layout dependencies for the UI designs of our project.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.5.1'
    implementation 'com.google.android.material:material:1.7.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.firebase:firebase-auth:21.1.0'
    implementation 'com.google.firebase:firebase-database:20.1.0'
    implementation 'androidx.swiperefreshlayout:swiperefreshlayout:1.1.0'
```

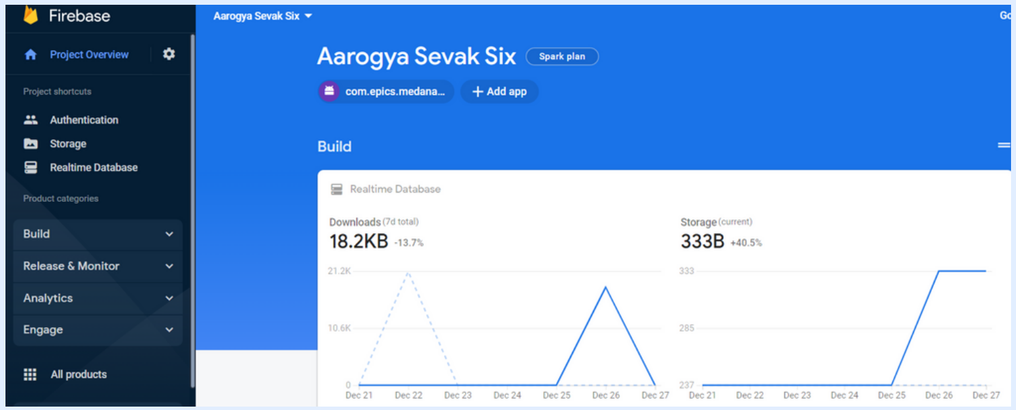
Knowledge regarding Project Modules

Moving onto the initial phases of the Back-End (Java Logic and Firebase Database) parts of our Project:

```
        etRegPwd.setError("Your password is too weak. Kindly use a mix of alphabets, numbers and special characters");
        etRegPwd.requestFocus();


    }

    catch (FirebaseAuthInvalidCredentialsException e){
        etRegPwd.setError("Your email is invalid");
        etRegPwd.requestFocus();

    }

    catch (FirebaseAuthUserCollisionException e){
        etRegPwd.setError("User is already regsitered");
        etRegPwd.requestFocus();

    }

    catch (Exception e){
```

Using try catch block to deal with credentials setup exceptions

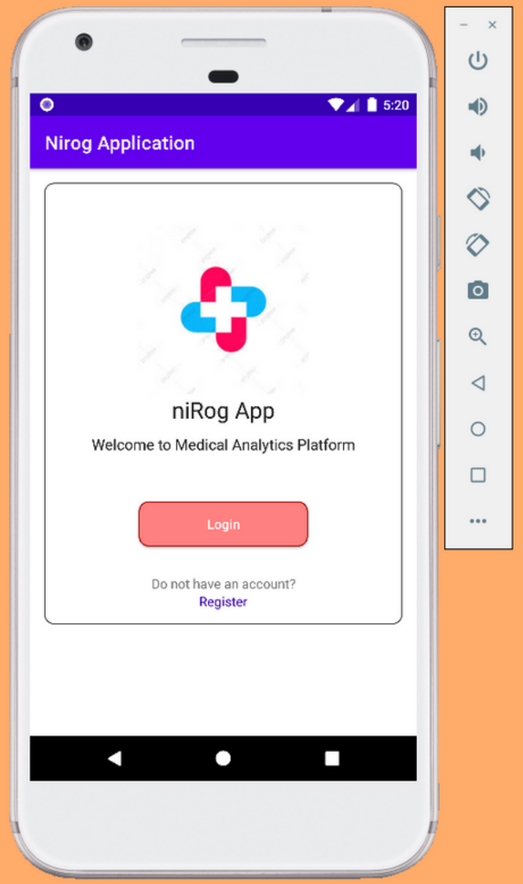
While trying to work our way around our Login and Register Activity logic parts, we are in debt to face a lot of exceptions. To try to tackle a few of them, we have introduced try catch blocks to deal with the exceptions we may encounter while our user tries to register his/her account. Example – Weak Password exception where user's password doesn't cross more than 6 letters/numbers and also doesn't have unique symbols. Another example is the Collision Exception which may occur if the user tries to register an account which is already registered with an email earlier.

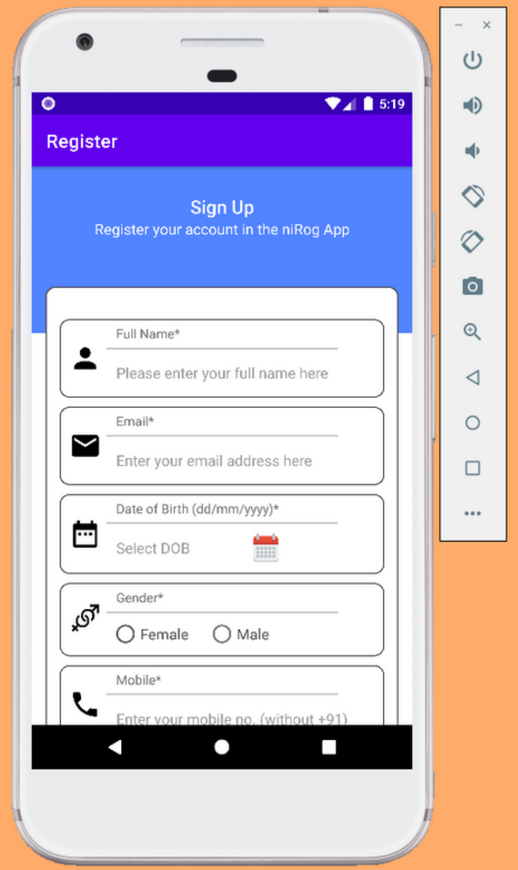


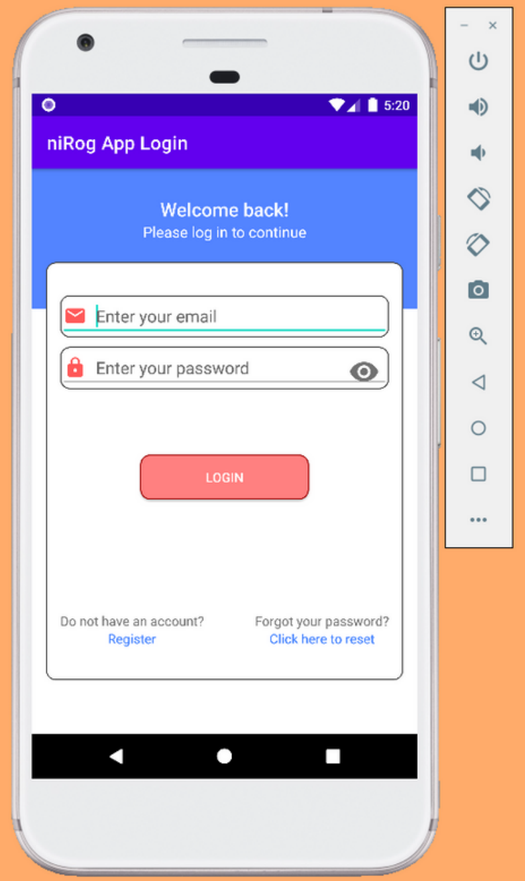
Setting up Firebase Console

Since these are the initial stages of our Project, we adhered to make attempts on starting out our Project on a full emphasis on all fields, both back-end and front-end. We have attempted to setup our Firebase Console in areas concerning topics such as Authentication, Storage and Realtime Database.

Overview of our Initial UI Designs







Timeline of our Project

A brief history of our ongoing endeavors

18TH OCTOBER

Announcement of EPICS Project circulated around the team.

20TH OCTOBER

Planned an idea regarding the project, finalized workload areas.

26TH OCTOBER

Started working on the initial parts of the Front-End part of our Project using XML.

15TH NOVEMBER

Attempted to start working on the initial parts of the Back-End areas of our Project using Java.

17TH DECEMBER

Started to learn about Firebase and initialized implementing Firebase Database in our Project.

PRESENT

Maintaining future plans to develop the software further by improving the UI and strengthening the back end part of it.

Individual Contributions

Teamwork based efforts to continue with our ongoiing Project

Somardh Jaiswal	Idea, overall planniing and functioning of Payment Gateway
Aaditya Sreenivasan	Awareness, analysis team, worked on front-end & ppt.
Akshat Mehrotra	Material Design UI Team - using XML
Ankit Deb	Back-end using Java and Firebase, ppt.
Yaksh Goyani	Front-end and resources development team
Shivam Dubey	Connections, resources development and testing team, prototypes, etc

"Health is the greatest gift, contentment the greatest wealth, faithfulness the best relationship."

-Lord Buddha

Experiential Programme Impact:

3

Cities Visited - Pune, BLR, Chennai

44

Awareness - Students

64

Continous Workdays to Project

500

Current Target Goal - Individuals

Conclusion

- The platform that we are adhering to develop is to be capable of providing more than just Medical Services to individuals who seek Healthcare Information, Support and access to care.
- It is also going to be a gateway for individuals to manage their health and well-being sufficiently, connect with other healthcare providers, and access medical services remotely.
- However, it is important to carefully consider the accuracy, reliability and security of the information of our users.

Contact us

VIT Bhopal University,
Bhopal-Indore Highway
Kothrikalan, Madhya Pradesh - 466114
Mail - info@vitbhopal.ac.in

Discord Link -
https://discord.gg/vKZUP9RGRP

Thank You