SOURCE CODES

ANDROID STUDIO

Main Activity.java(USER AUTHENTICATION)

```
package com.example.hospital;

import android.content.Intent;

import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
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;


public class MainActivity extends AppCompatActivity  {

    Button button,button1;
    ActionBar abr;
    EditText username,password1;
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);  //Setting environment in java XML

        abr=getSupportActionBar();
        abr.setBackgroundDrawable(new ColorDrawable(Color.parseColor("#f07c1d")));
        mAuth = FirebaseAuth.getInstance();
        button=findViewById(R.id.button);
        button1=findViewById(R.id.button1);

        username=findViewById(R.id.editText);
        password1= findViewById(R.id.editText1);



        button.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {


                String email=username.getText().toString().trim();
                String password=password1.getText().toString().trim();


                if(TextUtils.isEmpty(email))
                {
                    Toast.makeText(MainActivity.this,"Enter login
id",Toast.LENGTH_SHORT).show();
                    return;
                }

                if(TextUtils.isEmpty(password))
                {
                    Toast.makeText(MainActivity.this,"Enter
password",Toast.LENGTH_SHORT).show();
                    return;
                }



                mAuth.signInWithEmailAndPassword(email, password)
                        .addOnCompleteListener(MainActivity.this, new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull Task<AuthResult> task) {
                                if (task.isSuccessful())
                                {
                                    Toast.makeText(MainActivity.this, "AMBULANCE FOUND",
Toast.LENGTH_SHORT).show();
                                    Intent intent = new
Intent(MainActivity.this,Ambulance.class);
                                    startActivity(intent);
                                }
                                else{
                                    try {
                                        throw task.getException();
                                    } catch (FirebaseAuthWeakPasswordException e) {
                                        Toast.makeText(MainActivity.this, e.toString(),
Toast.LENGTH_SHORT).show();
                                    } catch (FirebaseAuthInvalidCredentialsException e) {
                                        Toast.makeText(MainActivity.this, e.toString(),
Toast.LENGTH_SHORT).show();
                                    } catch (FirebaseAuthUserCollisionException e) {
                                        Toast.makeText(MainActivity.this, e.toString(),
Toast.LENGTH_SHORT).show();
                                    } catch (Exception e) {
                                        Toast.makeText(MainActivity.this, e.toString(),
Toast.LENGTH_SHORT).show();
                                    }
                                }
                            }
                        }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Log.i("Fail", "onFailure: "+e);
                    }
                });
//                if(username.getText().toString().equals("admin") &&
password1.getText().toString().equals("admin"))
//                {
//                    Toast.makeText(MainActivity.this, "AMBULANCE FOUND",
Toast.LENGTH_SHORT).show();
//                    Intent intent = new Intent(MainActivity.this,Ambulance.class);
```

```
//                      startActivity(intent);
//                  }
//                  else
//                  {
//                      Toast.makeText(MainActivity.this, "INVALID LOGIN",
Toast.LENGTH_SHORT).show();
//                  }

              }
          });


          button1.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View view) {

                  Toast.makeText(MainActivity.this,"LOGIN NORMAL EMERGENCY
DETECTED",Toast.LENGTH_SHORT).show();
                    Intent intent1=new Intent(MainActivity.this,Hello.class);
                    startActivity(intent1);
              }
          });

        }



}
```

Main Activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    android:gravity="center"
    android:orientation="vertical"
    android:padding="20dp"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="AMBULANCE LOGIN"
        android:textAlignment="center"
        android:textSize="25dp" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="LOGIN"
        android:inputType="textEmailAddress" />
```

```
    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="PASSWORD"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimaryDark1"
        android:text="OK"
        android:textAppearance="@style/TextAppearance.AppCompat.Large" />

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:translationY="@android:dimen/notification_large_icon_width"
        android:background="@color/Red"
        android:text="EMERGENCY"
        android:textAppearance="@style/TextAppearance.AppCompat.Large" />


</LinearLayout>
```

Activity Hello.java(USER REGISTRATION AND VERIFICATION)

```
package com.example.hospital;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthProvider;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.concurrent.TimeUnit;
```

```
public class Hello extends AppCompatActivity {

    Button button,buttonvcd;
    EditText name,mobno,vcd;
    FirebaseDatabase firebase;
    FirebaseAuth mAuth;
    String s1,mob,nm,codesent;
    ActionBar abr;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello);
        firebase=FirebaseDatabase.getInstance();
        mAuth= FirebaseAuth.getInstance();
        abr=getSupportActionBar();
        abr.setBackgroundDrawable(new ColorDrawable(Color.parseColor("#f07c1d")));

        button=findViewById(R.id.button2);
        buttonvcd=findViewById(R.id.buttonvcd);
        name=findViewById(R.id.editText2);
        mobno=findViewById(R.id.editText3);
        vcd=findViewById(R.id.editTextvcd);









        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String name1=name.getText().toString().trim();
                String mobno1=mobno.getText().toString().trim();

                mob=mobno1;
                nm=name1;


                if(TextUtils.isEmpty(name1))
                {
                    Toast.makeText(Hello.this,"Enter Name",Toast.LENGTH_SHORT).show();
                    return;
                }
                if(TextUtils.isEmpty(mobno1) )
                {
                    Toast.makeText(Hello.this,"Enter Mobile no",Toast.LENGTH_SHORT).show();
                    return;
                }

                if(mobno1.length()!=10)
                {
                    Toast.makeText(Hello.this,"Invalid Mobile no",Toast.LENGTH_SHORT).show();
                    return;
                }


                Toast.makeText(Hello.this, "1", Toast.LENGTH_SHORT).show();

               sendVerification();

                //Toast.makeText(Hello.this, "5", Toast.LENGTH_SHORT).show();

               /*try {
```

```
                    s1=nm+"/"+mob;
                    DatabaseReference data = firebase.getReference("DataUsr");
                    data.push().setValue(s1);
                    s1="";
                }catch (Exception e){

                }

                Toast.makeText(Hello.this, "Data Send", Toast.LENGTH_SHORT).show();


                Intent intent=new Intent(Hello.this,Ambulance.class);

                startActivity(intent);
                */

            }
        });



        buttonvcd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {



                verifySignin();



            }
        });
    }


    private void verifySignin()
    {
        String code;
        code=vcd.getText().toString();
        Toast.makeText(Hello.this, "4", Toast.LENGTH_SHORT).show();
        PhoneAuthCredential credential = PhoneAuthProvider.getCredential(codesent, code);
        signInWithPhoneAuthCredential(credential);
    }

    private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {
         mAuth.signInWithCredential(credential)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful() ){
                            try {
                                s1 = nm + "/" + mob;
                                DatabaseReference data = firebase.getReference("DataUsr");
                                data.push().setValue(s1);
                                s1 = "";
                            } catch (Exception e) {

                            }

                            Toast.makeText(Hello.this, "Verification Completed",
Toast.LENGTH_SHORT).show();


                            Intent intent = new Intent(Hello.this, Ambulance.class);
                            startActivity(intent);
```

```
                        }

                        else
                            {
                            Toast.makeText(Hello.this, "INVALID VERIFICATION CODE",
Toast.LENGTH_SHORT).show();
                        }

                    }
                });

    }
    private  void sendVerification()
    {
        PhoneAuthProvider.getInstance().verifyPhoneNumber(
                "+91"+ mob,        // Phone number to verify
                60,                 // Timeout duration
                TimeUnit.SECONDS,   // Unit of timeout
                this,               // Activity (for callback binding)
                mCallbacks);        // OnVerificationStateChangedCallbacks

        Toast.makeText(Hello.this, "2", Toast.LENGTH_SHORT).show();

    }

    PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks=new
PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
        @Override
        public void onVerificationCompleted(PhoneAuthCredential phoneAuthCredential) {


            //Toast.makeText(Hello.this, "31", Toast.LENGTH_SHORT).show();

        }

        @Override
        public void onVerificationFailed(FirebaseException e) {

            //Toast.makeText(Hello.this, "32", Toast.LENGTH_SHORT).show();


        }

        @Override
        public void onCodeSent(String s, PhoneAuthProvider.ForceResendingToken
forceResendingToken) {
            super.onCodeSent(s, forceResendingToken);

           // System.out.println(s);
            //Toast.makeText(Hello.this, "33", Toast.LENGTH_SHORT).show();
            codesent=s;

        }
    };
}
```

Main Hello.xml(USER REGISTRATION AND VERIFICATION)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="bottom"
    android:background="@color/White">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="76dp"
        android:layout_gravity="center"
        android:background="@color/Blue"
        android:text="QUICKLY PROVIDE US WITH YOUR DETAILS "
        android:textAlignment="center"
        android:gravity="center"
        android:textSize="20dp" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint=" YOUR NAME"
        android:inputType="textPersonName"
        android:translationY="80sp" />

    <EditText
        android:id="@+id/editText3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint=" PHONE NO 10 DIGIT"
        android:inputType="phone"
        android:translationY="90sp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/Red"
        android:text="SUBMIT"
        android:translationY="100sp" />


    <EditText
        android:id="@+id/editTextvcd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"

        android:ems="10"
        android:hint=" VERIFICATION CODE"
        android:inputType="phone"
        android:translationY="150sp" />


  <Button
        android:id="@+id/buttonvcd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/Green"
        android:text="CONFIRM"
        android:translationY="200sp" />
</LinearLayout>
```

Activity Hospital.java(SELECTION AND VERIFICATION OF USER)

```
package com.example.hospital;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.Handler;
import android.provider.Settings;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

//import static android.icu.text.Normalizer.YES;

public class Ambulance extends AppCompatActivity {
     Button button,getLbutton;
     TextView showlocation;
     CheckBox t1,t2,t3,t4,t5,t6,t7,t8,t9,t10;
     String s1,lo,lal;
     int f1,f2,f3,f4,f5,f6,f7,f8,f9,f10;

     LocationManager locationManager;
     String latitude,longitude;
     private  static  final  int REQUEST_LOCATION=1;

     FirebaseDatabase firebase;
     DatabaseReference ref;
    ActionBar abr;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ambulance);

        abr=getSupportActionBar();
        abr.setBackgroundDrawable(new ColorDrawable(Color.parseColor("#f07c1d")));

        firebase=FirebaseDatabase.getInstance();

        t1=findViewById(R.id.switch11);
        t2=findViewById(R.id.switch12);
        t3=findViewById(R.id.switch13);
        t4=findViewById(R.id.switch14);
        t5=findViewById(R.id.switch15);
        t6=findViewById(R.id.switch16);
```

```
        t7=findViewById(R.id.switch17);
        t8=findViewById(R.id.switch18);
        t9=findViewById(R.id.switch19);
        t10=findViewById(R.id.switch20);

        f1=0;
        f2=0;
        f3=0;
        f4=0;
        f5=0;
        f6=0;
        f7=0;
        f8=0;
        f9=0;
        f10=0;

        s1="";
        lal="";
        lo="";


        ActivityCompat.requestPermissions(Ambulance.this,new String[]
                {Manifest.permission.ACCESS_FINE_LOCATION},REQUEST_LOCATION );

        showlocation=findViewById(R.id.textviewlc);
        getLbutton=findViewById(R.id.button4);
        button=findViewById(R.id.button3);

        getLbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                locationManager=(LocationManager) getSystemService(Context.LOCATION_SERVICE);

                if(!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER))
                {
                    OnGPS();
                }
                else{
                    getLocation();
                }
            }

            private void getLocation() {
                if(ActivityCompat.checkSelfPermission(Ambulance.this,Manifest.permission.ACCES
S_FINE_LOCATION)!= PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(Ambulance.this,
                        Manifest.permission.ACCESS_COARSE_LOCATION)!=
PackageManager.PERMISSION_GRANTED)
                {
                    ActivityCompat.requestPermissions(Ambulance.this,new String[]
                            {Manifest.permission.ACCESS_FINE_LOCATION},REQUEST_LOCATION );
                }
                else
                {
                    Location
LocationGPS=locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                    Location
LocationNetwork=locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                    Location
LocationPassive=locationManager.getLastKnownLocation(LocationManager.PASSIVE_PROVIDER);

                    if(LocationGPS !=null)
                    {
                        double lat=LocationGPS.getLatitude();
                        double longi=LocationGPS.getLongitude();
```

```
                        latitude=String.valueOf(lat);
                        longitude=String.valueOf(longi);

                        lo=longitude;
                        lal=latitude;

                        showlocation.setText("LOCATION"+"\n"+"LATITUDE"+latitude+"\n"+"LONGITU
DE"+longitude);
                    }
                    else if(LocationNetwork !=null)
                    {
                        double lat=LocationNetwork.getLatitude();
                        double longi=LocationNetwork.getLongitude();

                        latitude=String.valueOf(lat);
                        longitude=String.valueOf(longi);

                        lo=longitude;
                        lal=latitude;

                        showlocation.setText("LOCATION"+"\n"+"LATITUDE"+latitude+"\n"+"LONGITU
DE"+longitude);
                    }
                    else if(LocationPassive !=null)
                    {
                        double lat=LocationPassive.getLatitude();
                        double longi=LocationPassive.getLongitude();

                        latitude=String.valueOf(lat);
                        longitude=String.valueOf(longi);

                        lo=longitude;
                        lal=latitude;

                        showlocation.setText("LOCATION"+"\n"+"LATITUDE"+latitude+"\n"+"LONGITU
DE"+longitude);
                    }
                    else
                    {
                        Toast.makeText(Ambulance.this,"CANNOT
DETECT",Toast.LENGTH_SHORT).show();
                    }

                }
            }

            private void OnGPS() {
                final AlertDialog.Builder builder=new AlertDialog.Builder(Ambulance.this);
                builder.setMessage("ENABLE GPS").setCancelable(false).setPositiveButton("YES",
new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int which) {
                        startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));

                    }
                }).setNegativeButton("NO", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                            dialog.cancel();
                    }
                });
                final AlertDialog alertDialog=builder.create();
                alertDialog.show();

            }
```

```
});

t1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(t1.isChecked())
        {
            f1=1;
        }
        else
        {
            f1=0;
        }
    }
});

t1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(t1.isChecked())
        {
            f1=1;
        }
        else
        {
            f1=0;
        }
    }
});

t2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(t2.isChecked())
        {
            f2=1;
        }
        else
        {
            f2=0;
        }
    }
});

t3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(t3.isChecked())
        {
            f3=1;
        }
        else
        {
            f3=0;
        }
    }
});

t4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(t4.isChecked())
        {
            f4=1;
        }
```

```
            else
            {
                f4=0;
            }
        }
    });

    t5.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(t5.isChecked())
            {
                f5=1;
            }
            else
            {
                f5=0;
            }
        }
    });

    t6.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(t6.isChecked())
            {
                f6=1;
            }
            else
            {
                f6=0;
            }
        }
    });

    t7.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(t7.isChecked())
            {
                f7=1;
            }
            else
            {
                f7=0;
            }
        }
    });

    t8.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(t8.isChecked())
            {
                f8=1;
            }
            else
            {
                f8=0;
            }
        }
    });

    t9.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
```

```
                if(t9.isChecked())
                {
                    f9=1;
                }
                else
                {
                    f9=0;
                }
            }
        });

        t10.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(t10.isChecked())
                {
                    f10=1;
                }
                else
                {
                    f10=0;
                }
            }
        });


        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                if(lo=="" && lal=="")
                {
                    Toast.makeText(Ambulance.this,"GET LOCATION
AGAIN",Toast.LENGTH_SHORT).show();

                }


                else {
                    if (f1 == 1) {
                        s1 = s1 + "Y";
                    } else {
                        s1 = s1 + "N";
                    }


                    if (f2 == 1) {
                        s1 = s1 + "Y";
                    } else {
                        s1 = s1 + "N";
                    }


                    if (f3 == 1) {
                        s1 = s1 + "Y";
                    } else {
                        s1 = s1 + "N";
                    }

                    if (f4 == 1) {
                        s1 = s1 + "Y";
                    } else {
                        s1 = s1 + "N";
                    }

                    if (f5 == 1) {
```

```
                    s1 = s1 + "Y";
                } else {
                    s1 = s1 + "N";
                }

                if (f6 == 1) {
                    s1 = s1 + "Y";
                } else {
                    s1 = s1 + "N";
                }

                if (f7 == 1) {
                    s1 = s1 + "Y";
                } else {
                    s1 = s1 + "N";
                }

                if (f8 == 1) {
                    s1 = s1 + "Y";
                } else {
                    s1 = s1 + "N";
                }

                if (f9 == 1) {
                    s1 = s1 + "Y";
                } else {
                    s1 = s1 + "N";
                }

                if (f10 == 1) {
                    s1 = s1 + "Y";
                } else {
                    s1 = s1 + "N";
                }


                s1 = s1 + "/" + lal + "/" + lo;

                try {
                    DatabaseReference data = firebase.getReference("Data");
                    data.child("data").setValue(s1);
                    s1="";
                    new Handler().postDelayed(new Runnable() {

                        @Override
                        public void run() {
                            //do something
                        }
                    }, 2000 );//time in milisecond
                }catch (Exception e){

                }


                Toast.makeText(Ambulance.this, "Data Sent.", Toast.LENGTH_SHORT).show();



                Toast.makeText(Ambulance.this, "Data Sent...", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(Ambulance.this, Linkdel.class);
                intent.putExtra("DATA", s1);
                startActivity(intent);
            }
        }
    });
```

```
    }
}
```

Activity Hospital.xml(SELECTION AND VERIFICATION OF USER)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/White">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

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

            <TextView
                android:id="@+id/textView2"
                android:layout_width="match_parent"
                android:layout_height="60dp"
                android:text="Doctors Required"

                android:textAlignment="center"
                android:textSize="45dp"
                android:background="@color/Blue"/>

            <CheckBox
                android:id="@+id/switch11"
                android:layout_width="match_parent"
                android:layout_height="60dp"
                android:textSize="20dp"
                android:background="@color/White"
                android:padding="10sp"
                android:text="Anesthesiologists"
                android:textAlignment="center" />

            <CheckBox
                android:id="@+id/switch12"
                android:textSize="20dp"
                android:layout_width="match_parent"
                android:layout_height="60dp"
                android:background="@color/White1"
                android:text="Cardiologist"
                android:textAlignment="center" />

            <CheckBox
                android:id="@+id/switch13"
                android:textSize="20dp"
                android:layout_width="match_parent"
                android:layout_height="60dp"
                android:background="@color/White"
                android:text="Orthopedic Surgeon"
                android:textAlignment="center" />

            <CheckBox
                android:id="@+id/switch14"
                android:textSize="20dp"
                android:layout_width="match_parent"
                android:layout_height="60dp"
                android:background="@color/White1"
```

```
        android:text="Gastroenterologists"
        android:textAlignment="center" />

    <CheckBox
        android:id="@+id/switch15"
        android:textSize="20dp"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="@color/White"
        android:text="Neurologists"
        android:textAlignment="center" />

    <CheckBox
        android:id="@+id/switch16"
        android:textSize="20dp"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="@color/White1"
        android:text="Gynecologists"
        android:textAlignment="center" />

    <CheckBox
        android:id="@+id/switch17"
        android:textSize="20dp"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="@color/White"
        android:text="Ophthalmologists"
        android:textAlignment="center" />

    <CheckBox
        android:id="@+id/switch18"
        android:textSize="20dp"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="@color/White1"
        android:text="Plastic Surgeons"
        android:textAlignment="center" />

    <CheckBox
        android:id="@+id/switch19"
        android:textSize="20dp"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="@color/White"
        android:text="Urologists"
        android:textAlignment="center" />

    <CheckBox
        android:id="@+id/switch20"
        android:textSize="20dp"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="@color/White1"
        android:text="Dentist"
        android:textAlignment="center" />



    <TextView
        android:id="@+id/textviewlc"
        android:textSize="20dp"
        android:layout_width="wrap_content"
        android:layout_height="80dp"

        android:background="@color/White" />
```

```
            <Button
                android:id="@+id/button4"
                android:textSize="20dp"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="@color/Green"
                android:text="GET LOCATION"
                android:height="60dp"/>

            <Button
                android:id="@+id/button3"
                android:height="60dp"
                android:textSize="20dp"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="@android:color/holo_red_light"
                android:text="SUBMIT" />
        </LinearLayout>
    </ScrollView>

</LinearLayout>
```

Activity Linkdel.java(PROVIDING DIRECTION TO USER)

```
package com.example.hospital;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.net.Uri;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.os.Handler;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class Linkdel extends AppCompatActivity {
    ActionBar abr;
    FirebaseDatabase f1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_linkdel);
        abr=getSupportActionBar();
        abr.setBackgroundDrawable(new ColorDrawable(Color.parseColor("#f07c1d")));



        f1=FirebaseDatabase.getInstance();
        Intent intent=getIntent();
```

```
        String s=intent.getStringExtra("DATA");
        final TextView textView=findViewById(R.id.link);

        DatabaseReference mref=f1.getReference("loc");

        mref.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                // This method is called once with the initial value and again
                // whenever data at this location is updated.
                String  value="";
                for(DataSnapshot ds:dataSnapshot.getChildren()){
                    value = ds.getValue(String.class);
                }

                Toast.makeText(Linkdel.this, value, Toast.LENGTH_SHORT).show();

                //System.out.println(value);
                //textView.setText(value);


                String url = "https://www.google.com/maps/dir/?api=1&destination=" + value +
"&travelmode=driving";
                //System.out.println(url);
                Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
                startActivity(intent);


            }

            @Override
            public void onCancelled(DatabaseError error) {
                // Failed to read value

            }
        });
    }
}
```

Activity Linkdel.xml(PROVIDING DIRECTION TO USER)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView5"
        android:layout_width="match_parent"
        android:layout_height="51dp"
        android:layout_gravity="center"
        android:background="@color/Blue"
        android:text="PLEASE WAIT"
        android:textAlignment="center"
        android:gravity="center"
        />

    <TextView
        android:id="@+id/link"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
```

```
        android:autoLink="all"
        android:clickable="true"
        android:text=""
        android:textAlignment="center"
        android:gravity="center"
        />


    <TextView
        android:id="@+id/textView7"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
         />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="                WE ARE WORKING ON YOUR REQUEST" />
</LinearLayout>
```

DJANGO SOURCE CODE

VIEWS

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.contrib.auth.forms import AuthenticationForm
from users.forms import CustomUserCreationForm, PatientAdmissionForm,
PatientDischargeForm, DoctorRegistrationForm, DoctorDeleteForm, DoctorLoggInForm,
DoctorSignOffForm
from django.contrib.auth import login, logout, authenticate
from django.contrib import messages

from .models import CustomUser, Patient, Doctor
# Create your views here.


def patient_admission(request):
            if request.method == "POST":
                  form = PatientAdmissionForm(request.POST)
                  if form.is_valid():
                        patient = form.save(commit=False)
                        patient.patient_user=request.user
                        patient.save()
                        messages.success(request, f"Patient Added Successfully")
                        return redirect("users:patients")
                  else:
                        messages.error(request, f"Invalid Entry")
            form = PatientAdmissionForm()
            return render(request,"users/patient_admission.html",
                                    context={"form": form})


def patient_discharge(request):
      if request.method == "POST":
            form = PatientDischargeForm(request.POST)
            if form.is_valid():
                  #patient = form.save(commit=False)

      Patient.objects.filter(patient_id=form.cleaned_data.get('patient_id')).delete()
                  #patient.save()
                  messages.success(request, f"Patient Discharged")
                  return redirect("users:patients")
            else:
                  messages.error(request, f"Invalid Patient ID")
      form = PatientDischargeForm()
      return render(request,"users/patient_discharge.html",
                              context={"form": form})


def doctor_register(request):
      if request.method == "POST":
            form = DoctorRegistrationForm(request.POST)
            if form.is_valid():
                  doctor = form.save(commit=False)
                  doctor.doctor_user = request.user
```

```
                specialized_in = form.cleaned_data.get('specialized_in')
                doctor_id = form.cleaned_data.get('doctor_id')
                doctor_name = form.cleaned_data.get('doctor_name')
                doctor_license_no = form.cleaned_data.get('doctor_license_no')
                #doctor.specialized_in =
form.cleaned_data.get('doctor_specialized_in')
                doctor.save()
                messages.success(request, f"Doctor Registered Successfully")
                return redirect("users:doctors")
          else:
                messages.error(request, f"Invalid name Entry")

     form = DoctorRegistrationForm()
     return render(request,"users/doctor_register.html",
                        context={"form": form})



def doctor_delete(request):
     if request.method == "POST":
          form = DoctorDeleteForm(request.POST)
          if form.is_valid():
                #patient = form.save(commit=False)

     Doctor.objects.filter(doctor_id=form.cleaned_data.get('doctor_id')).delete()
                #patient.save()
                messages.success(request, f"Doctor Removed")
                return redirect("users:doctors")
          else:
                messages.error(request, f"Invalid Doctor ID")
     form = DoctorDeleteForm()
     return render(request,"users/doctor_delete.html",
                        context={"form": form})


def doctor_logg_in(request):
     if request.method == "POST":
          form = DoctorLoggInForm(request.POST)
          if form.is_valid():
                doctor_id = form.cleaned_data.get('doctor_id')
                doctor = Doctor.objects.get(doctor_id=doctor_id)
                doctor.is_active=True
                doctor.save()
                #print(Doctor.doctor_id)
                #doctor.Doctor.is_active = True
                #print(doctor.is_active)
                messages.success(request, f"Doctor
{form.cleaned_data.get('doctor_id')} signed in")
                return redirect("users:doctors")
          else:
                messages.error(request, f"Invalid Doctor ID")
     form = DoctorDeleteForm()
     return render(request,"users/doctor_logg_in.html",
                        context={"form": form})

def doctor_sign_off(request):
     if request.method == "POST":
          form = DoctorSignOffForm(request.POST)
          if form.is_valid():
```

```
                doctor_id = form.cleaned_data.get('doctor_id')
                doctor = Doctor.objects.get(doctor_id=doctor_id)
                doctor.is_active=False
                doctor.save()
                messages.success(request, f"Doctor
{form.cleaned_data.get('doctor_id')} logged off")
                return redirect("users:doctors")
          else:
                messages.error(request, f"Invalid Doctor ID")
     form = DoctorDeleteForm()
     return render(request,"users/doctor_sign_off.html",
                         context={"form": form})
'''patient_name = form.cleaned_data.get('patient_name')
                    patient_id = form.cleaned_data.get('patient_id')
                    form.save()
                    messages.success(request, f"Patient Added Successfully")
                    return redirect("users:patients")
               else:
                    #for msg in form.error_messages:
                    messages.error(request, f"Invalid Entry")
          form = PatientAdmissionForm()
          return render(request,
                              "users/patient_admission.html",
                              context={"form": form})'''

'''def patients(request):
     return render(request,
                         "users/patients.html",
                         context={"patients": Patient.objects.all})'''
def homepage(request):
     return render(request=request,
                      template_name="users\home.html",
                      context={"users": CustomUser.objects.all})

def register(request):
     if request.method == "POST":
          form = CustomUserCreationForm(request.POST)
          if form.is_valid():
               user = form.save()
               username = form.cleaned_data.get('username')
               messages.success(request, f"New Account Created: {username}")
               reg_no = form.cleaned_data.get('reg_no')
               name = form.cleaned_data.get('h_name')

               h_state = form.cleaned_data.get('h_state')

               hospital_address = form.cleaned_data.get('hospital_address')
               hospital_pincode = form.cleaned_data.get('hospital_pincode')

               g_maps_coords_lat = form.cleaned_data.get('g_maps_coords_lat')
               g_maps_coords_lon = form.cleaned_data.get('g_maps_coords_lon')

               no_of_doc = form.cleaned_data.get('no_of_doc')
               total_no_of_wards = form.cleaned_data.get('total_no_of_wards')
               total_no_of_beds = form.cleaned_data.get('total_no_of_beds')

               facilities = form.cleaned_data.get('facilities')

               type_of_doctors = form.cleaned_data.get('type_of_doctors')
```

```
                raw_password = form.cleaned_data.get('password1')
                user = authenticate(username=username, password=raw_password)
                login(request, user)
                messages.info(request, f"You are now logged in as: {username}")
                return render(request,
                              "users/user_logged_in.html",
                              context={})
          else:
                for msg in form.errors:
                      messages.error(request, f"{msg}: {form.errors[msg]}")
     form = CustomUserCreationForm()
     return render(request,
                              "users/register.html",
                              context={"form":form})

def login_request(request):
     if request.method == "POST":
          form = AuthenticationForm(request, data = request.POST)
          if form.is_valid():
                username = form.cleaned_data.get('username')
                password = form.cleaned_data.get('password')
                reg_no = form.cleaned_data.get('reg_no')
                name = form.cleaned_data.get('h_name')

                hospital_address = form.cleaned_data.get('hospital_address')
                hospital_pincode = form.cleaned_data.get('hospital_pincode')

                g_maps_coords_lat = form.cleaned_data.get('g_maps_coords_lat')
                g_maps_coords_lon = form.cleaned_data.get('g_maps_coords_lon')

                no_of_doc = form.cleaned_data.get('no_of_doc')
                total_no_of_wards = form.cleaned_data.get('total_no_of_wards')
                total_no_of_beds = form.cleaned_data.get('total_no_of_beds')

                facilities = form.cleaned_data.get('facilities')

                type_of_doctors = form.cleaned_data.get('type_of_doctors')

                user = authenticate(username = username, password = password)
                if user is not None:
                      login(request, user)
                      messages.info(request, f"You are now logged in as:
{username}")
                      return redirect("users:user_logged_in")
                else:
                      messages.error(request, "Invalid username or password")

          else:
                messages.error(request, "Invalid username or password")

     form = AuthenticationForm()
     return render(request,
                       "users/login.html",
                       {"form":form})


def logout_request(request):
```

```
      logout(request)
      messages.info(request, "Logged out successfully!")
      return redirect("users:homepage")

def user_logged_in(request):
      return render(request=request,
                        template_name="users/user_logged_in.html",
                        context={"users": CustomUser.objects.all})

def patients(request):
      users = request.user
      patient_list = Patient.objects.filter(patient_user = users)
      '''i = 0
                        for p in Patient.objects.all():
                              if p.patient_user == user:
                                    patient_list[i] = p
                                    i = i + 1'''
      return render(request,
                              "users/patients.html",
                              context={"patients": patient_list})


def doctors(request):
      users = request.user
      doctor_list = Doctor.objects.filter(doctor_user = users).order_by('-is_active')
      '''i = 0
                        for p in Patient.objects.all():
                              if p.patient_user == user:
                                    patient_list[i] = p
                                    i = i + 1'''
      return render(request,
                              "users/doctors.html",
                              context={"doctors": doctor_list})
```

==========

URL's

```
"""new_project URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/2.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.urls import path, include
from . import views

app_name = "users"

urlpatterns = [
```

```
      path('', views.homepage, name='homepage'),
      path('register/', views.register, name='register'),
      path('login/', views.login_request, name="login"),
      path('logout/', views.logout_request, name='logout'),
    path('patient_admission/', views.patient_admission, name='patient_admission'),
    path('patients/', views.patients, name='patients'),
    path('user_logged_in/', views.user_logged_in, name='user_logged_in'),
    path('patient_discharge/', views.patient_discharge, name='patient_discharge'),
    path('doctor_register/', views.doctor_register, name='doctor_register'),
    path('doctors/', views.doctors, name='doctors'),
    path('doctor_delete/', views.doctor_delete, name='doctor_delete' ),
    path('doctor_logg_in', views.doctor_logg_in, name='doctor_logg_in'),
    path('doctor_sign_off', views.doctor_sign_off, name='doctor_sign_off')
```

MODELS

```
from django.contrib.auth.models import AbstractUser, UserManager
from django.db import models
from django.core.validators import RegexValidator


SELECT_SPECIALITY = (
            ('Anesthesiologist','Anesthesiologist'),
            ('Anesthesiologist','Cardiologist'),
            ('Orthopedic Surgeon','Orthopedic Surgeon'),
            ('Gastroenterologist','Gastroenterologist'),
            ('Neurologist','Neurologist'),
            ('Gynecologist','Gynecologist'),
            ('Opthalmologist','Opthalmologist'),
            ('Urologist','Urologist'),
            ('Dentist','Dentist'),
            ('Plastic Surgeon','Plastic Surgeon'),
            )

alpha = RegexValidator(r'^[a-zA-Z ]*$', 'Only characters are allowed.')
beta = RegexValidator(r'^[a-zA-Z0-9 ]*$', 'Only characters are allowed.')

class CustomUserManager(UserManager):
      pass

class CustomUser(AbstractUser):

      h_name = models.CharField(default='NULL', max_length=50, validators=[beta])
      reg_no = models.CharField(max_length=10, blank=True, unique=True)
      h_state = models.CharField(max_length=30)
      h_multi = models.CharField(max_length=450)

      hospital_address = models.CharField(max_length=500, default='NULL')
      hospital_pincode = models.PositiveIntegerField(default=0)

      hosptal_telephone = models.PositiveIntegerField(default=0)
      g_maps_coords_lat = models.DecimalField(max_digits=8, decimal_places=6,
default=0)
      g_maps_coords_lon = models.DecimalField(max_digits=9, decimal_places=6,
default=0)

      no_of_doc = models.PositiveIntegerField(default=0)
```

```
      total_no_of_beds = models.PositiveIntegerField(default=0)
      total_no_of_wards = models.PositiveIntegerField(default=0)

      facilities = models.CharField(max_length=1000, default='NONE')

      type_of_doctors = models.CharField(max_length=1000, default='NONE')



      objects = CustomUserManager()

      def __str__(self):
            return self.username

# Create your models here.

class Patient(models.Model):


      patient_name = models.CharField(max_length=30, validators=[alpha])
      patient_id = models.CharField(max_length=100, unique=True)
      patient_user = models.ForeignKey(CustomUser, verbose_name="User",
on_delete=models.CASCADE)

      class Meta:
            verbose_name_plural = "Patients"

      def __str__(self):
            return self.patient_id


class Doctor(models.Model):

      doctor_id = models.CharField(max_length=100, unique=True)
      doctor_name = models.CharField(max_length=30, validators=[alpha])
      doctor_license_no = models.CharField(max_length=10, unique=True)
      specialized_in = models.CharField(max_length=1000, default='NONE')
      is_active = models.BooleanField(default=False)

      doctor_user = models.ForeignKey(CustomUser, verbose_name="User",
on_delete=models.CASCADE)

      class Meta:
            verbose_name_plural = "Doctors"

      def __str__(self):
            return self.doctor_id
```

FORMS

```
from django import forms
from django.contrib.auth.forms import UserCreationForm, UserChangeForm
from .models import CustomUser, Patient, Doctor
```

```
class CustomUserCreationForm(UserCreationForm):


      SELECT_STATE = (
('Maharashtra', 'Maharashtra'),
('Gujarat', 'Gujarat'),
('Madhya Pradesh', 'Madhya Pradesh'),
)

      SELECT_FACILITIES = (
            ('1','ICU'),
            ('2','OPD'),
            ('3','OT'),
            ('4','Laboratory'),
            ('5','Pharmacy'),
            ('6','Blood Bank'),
            ('7','Ambulance'),
            )

      SELECT_DOCTORS = (
            ('1','Anesthesiologists'),
            ('2','Cardiologists'),
            ('3','Orthopedic Surgeon'),
            ('4','Gastroenterologists'),
            ('5','Neurologists'),
            ('6','Gynecologists'),
            ('7','Opthalmologists'),
            ('8','Urologists'),
            ('9','Dentist'),
            ('10','Plastic Surgeons'),
            )


      h_name = forms.CharField(required=True, max_length=50)
      email = forms.EmailField(required=True, max_length=254)
      #h_state = forms.ChoiceField(choices=SELECT_STATE)
      reg_no = forms.CharField(max_length=10)
      #h_multi = forms.MultipleChoiceField(widget=forms.CheckboxSelectMultiple,
choices=SELECT_STATE)


      hospital_address = forms.CharField(required=True, max_length=500)
      hospital_pincode = forms.CharField(required=True, max_length=6)

      g_maps_coords_lat = forms.DecimalField(required=True, max_digits=8,
decimal_places=6)
      g_maps_coords_lon = forms.DecimalField(required=True,max_digits=9,
decimal_places=6)

      no_of_doc = forms.CharField(required=True, max_length=4)
      total_no_of_beds = forms.CharField(required=True, max_length=4)
      total_no_of_wards = forms.CharField(required=True, max_length=4)

      facilities = forms.MultipleChoiceField(widget=forms.CheckboxSelectMultiple,
choices=SELECT_FACILITIES)
```

```
      type_of_doctors =
forms.MultipleChoiceField(widget=forms.CheckboxSelectMultiple,
choices=SELECT_DOCTORS)

      class Meta:
            model = CustomUser
            fields = ('username', 'email', 'h_name', 'reg_no',
                  #'h_state', #'h_multi',
                  'hospital_address', 'hospital_pincode',
                  'g_maps_coords_lat', 'g_maps_coords_lon',
                  'no_of_doc', 'total_no_of_wards', 'total_no_of_beds',
                  'facilities','type_of_doctors',
                  'password1', 'password2')
            #fields_required = ['email', 'username', 'reg_no', 'h_state', 'h_multi']
            '''fieldsets = (
                  (('User'), {'fields': ('username', 'email', 'reg_no')}),
                  (('Address'), {'fields': ('h_state', 'h_multi')}),

            )'''


      def save(self, commit=True):
            user = super(CustomUserCreationForm, self).save(commit=False)

            user.h_name = self.cleaned_data['h_name']
            user.email = self.cleaned_data['email']
            #user.h_state = self.cleaned_data['h_state']
            #user.h_multi = self.cleaned_data['h_multi']

            user.hospital_address = self.cleaned_data['hospital_address']
            user.hospital_pincode = self.cleaned_data['hospital_pincode']

            user.g_maps_coords_lat = self.cleaned_data['g_maps_coords_lat']
            user.g_maps_coords_lon = self.cleaned_data['g_maps_coords_lon']

            user.no_of_doc = self.cleaned_data['no_of_doc']
            user.total_no_of_beds = self.cleaned_data['total_no_of_beds']
            user.total_no_of_wards = self.cleaned_data['total_no_of_wards']

            user.facilities = self.cleaned_data['facilities']

            user.type_of_doctors = self.cleaned_data['type_of_doctors']

            user.save()
            return user


class CustomUserChangeForm(UserChangeForm):

      class Meta:
            model = CustomUser
            fields = ('username', 'email', 'h_name', 'reg_no',
              'h_state', 'h_multi',
              'hospital_address', 'hospital_pincode',
               'g_maps_coords_lat', 'g_maps_coords_lon',
               'no_of_doc', 'total_no_of_wards', 'total_no_of_beds',
               'facilities','type_of_doctors',
               )
```

```
            add_fieldsets = (
             (
                 None,
                 {
                     "classes": ("wide",),
                     "fields": ("username", "h_name", "email", "reg_no",
                     "h_state", "h_multi",
                     "hospital_address", "hospital_pincode",
                     "g_maps_coords_lat", "g_maps_coords_lon",
                             "no_of_doc", "total_no_of_wards", "total_no_of_beds",
                             "facilities", "type_of_doctors",
                     "password1", "password2"),
                 },
             ),
        )


class PatientAdmissionForm(forms.ModelForm):

      #patient_id = forms.CharField(max_length=100)
      #atient_name = forms.CharField(max_length=30, required=True,
widget=forms.TextInput(attrs={'class':'form-control' ,
'autocomplete':'off','pattern':'[A-Za-z]+','title':'Enter Characters only'}))

      class Meta:
            model = Patient
            fields = ['patient_id','patient_name',]
            fields_required = ['patient_id','patient_name',]

      '''
      def save(self, commit=True):
            patient = super(PatientAdmissionForm, self).save(commit=False)
            #username = CustomUser.objetcs.filter()
            #for user in CustomUser.objects.all():
                  #if user.is_authenticated:
            patient.patient_name = self.cleaned_data['patient_name']
            patient.patient_id = self.cleaned_data['patient_id']
            #patient.patient_user = patient.patient_user
            patient.save()
            return patient
'''


class PatientDischargeForm(forms.Form):

      patient_id = forms.CharField(max_length=100)

      class Meta:
            #model = Patient
            fields = ['patient_id']
            fields_required = ['patient_id']


class DoctorRegistrationForm(forms.ModelForm):


      '''SELECT_SPECIALITY = (
```

```
            ('Anesthesiologist','Anesthesiologist'),
            ('Cardiologist','Cardiologist'),
            ('Orthopedic Surgeon','Orthopedic Surgeon'),
            ('Gastroenterologist','Gastroenterologist'),
            ('Neurologist','Neurologist'),
            ('Gynecologist','Gynecologist'),
            ('Opthalmologist','Opthalmologist'),
            ('Urologist','Urologist'),
            ('Dentist','Dentist'),
            ('Plastic Surgeon','Plastic Surgeon'),
            )'''

      SELECT_SPECIALITY = (
            ('1','Anesthesiologist'),
            ('2','Cardiologist'),
            ('3','Orthopedic Surgeon'),
            ('4','Gastroenterologist'),
            ('5','Neurologist'),
            ('6','Gynecologist'),
            ('7','Opthalmologist'),
            ('8','Urologist'),
            ('9','Dentist'),
            ('10','Plastic Surgeon'),
            )

      #doctor_id = forms.CharField(max_length=100)
      #doctor_name = forms.CharField(max_length=30)
      #doctor_license_no = forms.CharField(max_length=10)
      #specialized_in = forms.CharField(max_length=1000, help_text="(Example:
Anesthesiologist, Cardiologist, Neurologist, etc.)")
      specialized_in = forms.ChoiceField(choices=SELECT_SPECIALITY)

      class Meta:
            model = Doctor
            fields = ['doctor_id', 'doctor_name', 'doctor_license_no',
'specialized_in',]
            fields_required = ['doctor_id', 'doctor_name', 'doctor_license_no',
'specialized_in',]

      '''def save(self, commit=True):
            doctor = super(DoctorRegistrationForm, self).save(commit=False)

            doctor.doctor_id = self.cleaned_data['doctor_id']
            doctor.doctor_name = self.cleaned_data['doctor_name']
            #user.h_state = self.cleaned_data['h_state']
            #user.h_multi = self.cleaned_data['h_multi']
            doctor.doctor_license_no = self.cleaned_data['doctor_license_no']
            doctor.specialized_in = self.cleaned_data['specialized_in']

            doctor.save()
            return doctor'''



class DoctorDeleteForm(forms.Form):

      doctor_id = forms.CharField(max_length=100)
```

```
      class Meta:
            #model = Patient
            fields = ['doctor_id']
            fields_required = ['doctor_id']

class DoctorLoggInForm(forms.Form):

      doctor_id = forms.CharField(max_length=100)

      class Meta:
            #model = Patient
            fields = ['doctor_id']
            fields_required = ['doctor_id']

class DoctorSignOffForm(forms.Form):

      doctor_id = forms.CharField(max_length=100)

      class Meta:
            #model = Patient
            fields = ['doctor_id']
            fields_required = ['doctor_id']
      '''def save(self, commit=True):
                                          doctor = super(DoctorRegistrationForm,
self).save(commit=False)

                                          doctor.doctor_id =
self.cleaned_data['doctor_id']
                                          doctor.doctor_name =
self.cleaned_data['doctor_name']
                                          doctor.doctor_license_no =
self.cleaned_data['doctor_license_no']
                                          doctor.specialized_in =
self.cleaned_data['doctor_specialized_in']
                                                      #doctor.doctor_user =
self.request.user

                                          doctor.save()
                                          return doctor'''
```

ADMIN

```
from django.contrib import admin

from tinymce.widgets import TinyMCE

from django.contrib.auth import get_user_model
from django.contrib.auth.admin import UserAdmin

from .forms import CustomUserCreationForm, CustomUserChangeForm
from .models import CustomUser, Patient, Doctor

from django.db import models


class CustomUserAdmin(UserAdmin):
```

```
      add_form = CustomUserCreationForm
      form = CustomUserChangeForm
      model = CustomUser
      list_display = ['username','email', 'h_name', 'reg_no',
      'h_state', 'h_multi',
      'hospital_address', 'hospital_pincode',
      'g_maps_coords_lat', 'g_maps_coords_lon',
      'no_of_doc', 'total_no_of_wards', 'total_no_of_beds',
      'facilities', 'type_of_doctors']

      #fields_required = ['email', 'username', "h_name" 'reg_no', 'h_state',
'h_multi', 'hospital_address', 'hospital_pincode']

      fieldsets = (
                  (('User'), {'fields': ('username', 'email', 'h_name', 'reg_no',
'password')}),
                  (('Permissions'), {'fields': ('is_active', 'is_staff')}),
                  (('Address'), {'fields': ('hospital_address', 'hospital_pincode')}),
                  (('Location'), {'fields': ('g_maps_coords_lat',
'g_maps_coords_lon')}),
                  (('General Info'), {'fields': ('no_of_doc', 'total_no_of_wards',
'total_no_of_beds')}),
                  (('Facilities'), {'fields': ('facilities', 'type_of_doctors')}),

            )

      add_fieldsets = (
            (
                None,
                {
                    "classes": ("wide",),
                    "fields": ("username", "h_name", "email", "reg_no",
                    #"h_state", "h_multi",
                    "hospital_address", "hospital_pincode",
                    "g_maps_coords_lat", "g_maps_coords_lon",
                            "no_of_doc", "total_no_of_wards", "total_no_of_beds",
                            "facilities", "type_of_doctors",
                    "password1", "password2"),
                },
            ),
        )

      formfield_overrides = {
            models.TextField: {'widget': TinyMCE()}
      }

admin.site.register(Patient)

admin.site.register(Doctor)

admin.site.register(CustomUser, CustomUserAdmin)
```

HTML FILES

Header

```
<head>
      {% load static %}
      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">

      <title></title>
</head>

<body>
      {% include "users/includes/navbar.html" %}

      {% include "users/includes/messages.html" %}
      <div class="container">
            <br>
            {% block content %}
            {% endblock %}
      </div>
</body>
```

Home

```
{% extends "users/header.html"%}

{% block content %}
<div>
    <h1><center>IMES</center></h1>
</div>
<div class="container">
      <div class="container">
            <div class="container">
            <div class="col s12 m6 l4">
                  <center>
                  <div class="card w-50">
                        <div class="card-body">
                              <center>
                                    <h5 class="card-title">---Welcome to the IMES---
</h5>
                              <h7 class="card-text">
                                    <br>
                                    <br>
                                    <br>
                                    <br>
                                    <p>
                                    New User??<br>
                                    <a href="/register"><button class="btn btn-
light">Register</button></a><br><hr>
                                    Already a user!!<br>
                                    <a href="/login"><button class="btn btn-
light"\>Login</button></a>
                                    </p>
                                    </h7>
                              </center>
                        </div>
                  </div>
            </center>
            </div>
      </div>
```

```
      </div>
</div>
      <!--
      {% if user.is_authenticated %}
            <p>{{user.CustomUser.objects.all}}</p>
            <p>{{user.h_name}}</p>
            <p>{{user.reg_no}}</p>
      {% endif %}
-->
{% endblock %}
<!--

<center>
      <div>
            <a class="btn btn-primary" href="/register"><button>Patient
Admission</button></a>
            <br>
            <br>
            <a class="btn btn-primary" href="/login"><button>Patient Discharge
</button></a>

      </div>
</center>


-->
```

User Login

```
{% extends "users/header.html"%}

{% block content %}
<div class="container">
    <h2><center>{{user.h_name}}</center></h2>
</div>
<div class="container">
      <center>
      <a href="/patients"><button class="btn btn-light">Patients</button></a>
      <a href="/doctors"><button class="btn btn-light">Doctors</button></a>
      </center>
      {% if user.is_authenticated %}
            <p>{{user.CustomUser.objects.all}}</p>
            <div clss="row">
                  <div class="col s12 m6 l4">
                        <div class="card">
                              <div class="card-body">
                                    <div class="card-title">Id: {{user.username}}</div>
                              </div>
                        </div>
                  </div>
                  <div class="col s12 m6 l4">
                        <div class="card">
                              <div class="card-body">
                                    <div class="card-title">Reg No:
{{user.reg_no}}</div>
                              </div>
                        </div>
```

```
                  </div>
                  <div class="col s12 m6 l4">
                        <div class="card">
                              <div class="card-body">
                                    <div class="card-title">No. of Doctors:
{{user.no_of_doc}}</div>
                              </div>
                        </div>
                  </div>
                  <div class="col s12 m6 l4">
                        <div class="card">
                              <div class="card-body">
                                    <div class="card-title">No. of Beds:
{{user.total_no_of_beds}}</div>
                              </div>
                        </div>
                  </div>
                  <div class="col s12 m6 l4">
                        <div class="card">
                              <div class="card-body">
                                    <div class="card-title">No. of Wards:
{{user.total_no_of_wards}}</div>
                              </div>
                        </div>
                  </div>
            </div>
      {% endif %}
</div>
{% endblock %}
<!--

<center>
      <div>
            <a class="btn btn-primary" href="/register"><button>Patient
Admission</button></a>
            <br>
            <br>
            <a class="btn btn-primary" href="/login"><button>Patient Discharge
</button></a>

      </div>
</center>
```

R STUDIO

```
# libraries for downloading json file
library(RJSONIO)
library(RCurl)

#downloading data from firebase
raw_data<-getURL("https://hospital3-e1eda.firebaseio.com/.json")
raw_data

# extracting important data
dataJson<-fromJSON(raw_data)
dataJson
#spliting to get y/n series and co-ordinates
data_list<-as.list(unlist(strsplit(dataJson$Data,"/")))
data_list
lat_min<-as.numeric(data_list[2])
lat_min
long_min<-as.numeric(data_list[3])
long_min
#converts data_list[1] into character
temp<-as.character(data_list[[1]])
temp
#splits each letter of temp
series_yn<-as.list(unlist(strsplit(temp,"")))
series_yn
pos_y<-c()
#counts no of Y and N
n<-length(series_yn)
j<-1
#loop to find positions of Y
for (i in 1:n){
  if(series_yn[[i]]=="Y")
  {
```

```
    pos_y[[j]]<-i
    j<-j+1
  }
}
pos_y
radians<-function(rad){
  value<-((rad*3.14)/180)
  return (value)
}
#function to calculate distance of hospital
cal_distance<-function(lat1,lon1,lat2,lon2){
  R<-6371
  fi1<-as.numeric(radians(lat1))
  fi2<-as.numeric(radians(lat2))
  dfi<-as.numeric(radians(lat2-lat1))
  dlm<-as.numeric(radians(lon2-lon1))
  a<-as.numeric(sin(dfi/2)*sin(dfi/2)+cos(fi1)*cos(fi2)*sin(dlm/2)*sin(dlm/2))
  c<-as.numeric(2*atan2(sqrt(a),sqrt(1-a)))
  d<-as.numeric(R*c)
  return(d)
}
#function for finding intersection of
countit<-function(pos_y,list_spl){
  value <-length(intersect(pos_y,list_spl))
  print(value)
  store_len<-length(pos_y)
  if(all(pos_y%in%list_spl)){
    return(1)
  }
  else if(value>store_len-1){

    return(2)
  }
  else{
    return(-1)
  }
```

```
}
#=================================================**************************==========
=========================================

#hospital database connectivity

library("DBI")

library("odbc")

library("RSQLite")

#connection between database and R server

con <- DBI::dbConnect(RSQLite::SQLite(),
"C:\\Users\\shahy_mxyzd8u\\AppData\\Local\\Programs\\Python\\Python37-
32\\Scripts\\new_project\\db.sqlite3")

#extracting count of distinct usernames from database

hos_cnt<-dbGetQuery(con,"select count(Distinct username) from users_customuser")

#count of no. of hospitals in database converted to numeric data

hos_cnt<-as.numeric(hos_cnt)

#extracting longi. and latit. of hospitals from database

hos_coord<-dbGetQuery(con,"select g_maps_coords_lat,g_maps_coords_lon from
users_customuser")

hos_coord

#extracting usernames  from database

hos_name<-dbGetQuery(con,"select username from users_customuser")

#counter variable for storing positions for hospitals in range


#=========================================================**********distance
sorting******************==========================================================

o<-1

list_hos<-c( )

for (i in 1:hos_cnt){

  dist<-cal_distance(lat_min,long_min,hos_coord[i,1],hos_coord[i,2]) #calculates distance
of hospital from incident place

  dist<10

  print(dist)

  if(dist<10){                                                    #loop for storing
positions of hospitals with distance <10km

    print(dist)

    list_hos[[o]]<-i

    o<-o+1
```

```
  }
}
list_hos
#=================================================================**********finding
hospital with all or two
facilities**********************=================================================
m<-1
x<-1
y<-1
hos_avail<-c()
list_num_allfac<-c()
list_num_2fac<-c()
match_no<-0
for(k in list_hos){
  hos_fac<-paste("select type_of_doctors from users_customuser where id=",k,"")
  hos_avail[[m]]<-dbGetQuery(con,hos_fac)
  Tempp<-as.character(split(hos_avail[[m]],f=""))
  Tempp<-as.list(unlist(strsplit(Tempp,"")))
  temp2<-as.character(1:10)
  list_spl<-c()
  h<-1
  for(i in temp2){
    for(j in Tempp){
      if(identical(i,j)){
        if(as.list(as.numeric(j))==1 & h > 1){
          j<-10
        }
        list_spl[h]<-as.list(as.numeric(j))
        h<-h+1
      }
    }

  }
  match_no<-countit(pos_y,list_spl)
  print(match_no)
  if(match_no == 1){
```

```
    list_num_allfac[[x]]<-k
    x<-x+1
  }
  else if(match_no==2){
    list_num_2fac[[y]]<-k
    y<-y+1
  }
  else {
    print("Finding next location")
  }

  m<-m+1
}
list_spl
hos_avail
list_num_allfac
list_num_2fac
#================================================***************checking***************=
=========================================================================
actual_hosp_sorted<-c()
if(is.null(list_num_allfac)){
  actual_hosp_sorted<-list_num_2fac
}
if(!is.null(list_num_allfac)){
  actual_hosp_sorted<-list_num_allfac
}
if(is.null(list_num_allfac)&is.null(list_num_2fac)){
  o<-1
  list_hos2<-c( )
  for (i in 1:hos_cnt){
    dist<-cal_distance(lat_min,long_min,hos_coord[i,1],hos_coord[i,2]) #calculates
distance of hospital from incident place
    if(dist<15){                                                    #loop for storing
positions of hospitals with distance <10km
      print(dist)
      list_hos2[[o]]<-i
      o<-o+1
```

```
  }
}
m<-1
x<-1
y<-1
hos_avail<-c()
list_num_allfac<-c()
list_num_2fac<-c()
match_no<-0
for(k in list_hos2){
  hos_fac<-paste("select type_of_doctors from users_customuser where id=",k,"")
  hos_avail[[m]]<-dbGetQuery(con,hos_fac)
  Tempp<-as.character(split(hos_avail[[m]],f=""))
  Tempp<-as.list(unlist(strsplit(Tempp,"")))
  temp2<-as.character(1:10)
  Tempp
  list_spl<-c()
  h<-1
  for(i in temp2){
    for(j in Tempp){
      if(identical(i,j)){
        if(as.list(as.numeric(j))==1 & h > 1){
          j<-10
        }
        list_spl[h]<-as.list(as.numeric(j))
        h<-h+1
      }
    }

  }
  list_spl
  match_no<-countit(pos_y,list_spl)
  print(match_no)
  if(match_no == 1){
    list_num_allfac[[x]]<-k
    x<-x+1
```

```
    }
    else if(match_no==2){
      list_num_2fac[[y]]<-k
      y<-y+1
    }
    else {
      print("Finding next location")
    }

    m<-m+1
  }
  if(is.null(list_num_allfac)){
    actual_hosp_sorted<-list_num_2fac
  }
  if(!is.null(list_num_allfac)){
    actual_hosp_sorted<-list_num_allfac
  }
}
dbGetQuery(con,"select type_of_doctors from users_customuser where id=2")
actual_hosp_sorted
#=================================================================**********************
*****==================================================================
final_hosp<-c()
temp_doc_info<-c()
b<-1
for(h in actual_hosp_sorted){
  que1<-paste("select total_no_of_beds from users_customuser where id =",h," ")
  no_of_beds<-dbGetQuery(con,que1)
  no_of_beds
  que2<-paste("select count(patient_user_id) from users_patient where patient_user_id
=",h," ")
  no_of_patients<-dbGetQuery(con,que2)
  avail_bed<-no_of_beds-no_of_patients
  if(avail_bed!=0){
    final_hosp[[b]]<-h
    b<-b+1
```

```
  }

}

#=====================================================%%%%%%%%%%%%%%%%%%%%+==========
=================================================================

final_hosp

dbExecute(con,"create view temp_doc_id_finals as select doctor_user_id , is_active ,
specialized_in from users_doctor")

e<-1

final_hospital<-c()

for(n in actual_hosp_sorted){

  query11<-paste("select doctor_user_id from temp_doc_id_finals where
specialized_in=",pos_y, " and doctor_user_id=",n," and is_active=1"," ")

  datatp<-dbGetQuery(con,query11)

  if(!is.null(datatp)){

    final_hospital[[e]]<-datatp

    e<-e+1

  }

}

dbGetQuery(con,"select * from temp_doc_id_finals")

dbExecute(con,"drop view temp_doc_id_finals")

#=============================================================================**********
**********==========================================================

final_hospital[[1]]

g<-1

data_later<-c()

length(final_hospital)

if(length(final_hospital)>1){

  for(m in final_hospital) {

    print(m)

    distance<-cal_distance(lat_min,long_min,hos_coord[m,1],hos_coord[m,2])

    print(distance)

    data_later[[g]]<-c(m,distance)

    print(data_later[[g]])

    g<-g+1

  }

  data_later<-data.frame(data_later)

  print(data_later)
```

```
  final_data<-order(data_later)
  print(final_data)
  for_cords<-final_data[[1]]
  final_hospital[[1]]<-for_cords[[1]]
}
hos_details<-hos_coord[final_hospital[[1]],]
hos_details
dataJson$link
Link<-cbind(hos_details[[1]],hos_details[[2]])
Link
dataJson$link<-Link
dbDisconnect(con)
dataJson$link<-paste(dataJson$link,collapse = ",")
dataJson$link
library("fireData")
upload(x = dataJson$link, projectURL = "https://hospital3-e1eda.firebaseio.com/",
directory = "loc")
```