# Mini HealthCare Helper

A project report submitted for

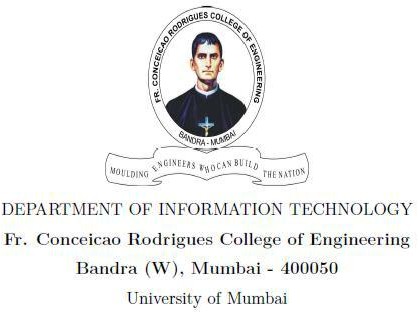
# Android Apps Development Lab (Semester VII)

by

Student 1 (Roll No) Student 2 (Roll No) Student 3 (Roll No)

Under the guidance of Prof. Nilesh Patil

(sign with date)



# APPROVAL SHEET

**Project Report Approval**

This project report entitled as <Project Title> by **<Student 1 (Roll No.), Student 2 (Roll No.), Student 3 (Roll No.)>** is approved as project for **Android Apps Development Lab** in Fourth year Engineering (Sem - VII), Information Technology.

Examiners 1.——————————

2.——————————

Date:

Place:

# ABSTRACT

* This project is healthcare helper created for this COVID like crisis.
* Helps patient to schedule an appointment
* Doctor can prepare as per the scheduled appointment
* Blood donors can register and Admin can connect them in need.
* **Motivation**: Patients can avoid queue in hospitals. Lessen public gathering during COVID crisis

**PRE-REQUISITES FOR ANDROID DEVELOPMENT**

**Android Studio**

You can use Android studio for development and Emulator

Android Studio need android SDK. Lates version <> was installed

**Database**

FireBase

plugin: 'com.android.application'

dependencies {

// ...

implementation 'com.google.firebase:firebase-core:9.6.1'

}

// ADD THIS AT THE BOTTOM

apply plugin: 'com.google.gms.google-services'

# DATABASE – Firebase

Diagram

Description automatically generated

**SQL Queries**

* **Find doctors**

Select doctorId from DoctorEnt where doctorPhone ? and doctorPassword? limit 1

* **Find doctors with given Id**

Select \* from DoctorEnt where doctorId=:doctorId limit 1

* **Find doctors with given Id**

Select patientId from PatientEnt where patientPhone =:patientPhone and patientPassword=:patientPassword

* **Find doctors with given Id**

Select \* from DoctorEnt Group By doctorId

* **Find doctors with given Id**

Select \* from AvailabilityEnt where doctorId=:doctorId and availableDate=:date

* **Find doctors with given Id**

Select \* from AppointmentEnt where doctorId=:doctorId and dateOfAppointment=:date

* **FInd doctors with given Id**

SELECT \* FROM ((SELECT \* FROM AppointmentEnt as app where app.patientId=:patientId and (dateOfAppointment > :currentDate OR (dateOfAppointment = :currentDate AND timeOfAppointment > :currentTime))) as PatientResult) INNER JOIN DoctorEnt as doc ON doc.doctorId=PatientResult.doctorId

* **FInd doctors with given Id**

SELECT \* FROM ((SELECT \* FROM AppointmentEnt as app where app.patientId=:patientId and (dateOfAppointment < :currentDate OR (dateOfAppointment = :currentDate AND timeOfAppointment < :currentTime))) as PatientResult) INNER JOIN DoctorEnt as doc ON doc.doctorId=PatientResult.doctorId

**CODE & SCREENSHOT**

**(All .java and .xml files)**

1. **AndroidManifest.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.healthcare.helper">  
  
 <application  
 android:name="application.ApplicationClass"  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher\_bk"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round\_bk"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MiniHealthcareHelper"  
 tools:replace="android:icon">  
 <activity  
 android:name="com.healthcare.helper.SplashScreenActivity"  
 android:theme="@style/SplashScreenTheme">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <activity  
 android:name="patient.PatientLoginSignUpActivity"  
 android:theme="@style/PatientLoginSignUpTheme"  
 android:windowSoftInputMode="adjustResize" />  
  
 *<!-- Patient search Activity Starts Here -->* <activity android:name="patient.PatientSearchResultActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.SEARCH" />  
 </intent-filter>  
 </activity>  
 <activity android:name="patient.PatientMainActivity">  
 <meta-data  
 android:name="android.app.searchable"  
 android:resource="@xml/searchable" />  
 </activity> *<!-- Patient search Activity Ends Here -->* <activity android:name="SampleTestingProject.Sample.Sample1" />  
 <activity  
 android:name="SampleTestingProject.Constraintlayout3"  
 android:parentActivityName="SampleTestingProject.Constraintlayout2">  
 <meta-data  
 android:name="android.support.PARENT\_PARENT\_ACTIVITY"  
 android:value="SampleTestingProject.Constraintlayout2" />  
 </activity>  
 <activity android:name="SampleTestingProject.Constraintlayout2" />  
 <activity android:name="SampleTestingProject.ConstraintLayoutActivity" />  
 <activity android:name="patient.PatientAppointmentHistoryActivity" />  
 <activity android:name="patient.PatientBookAppointmentActivity" />  
 <activity android:name="doctor.DoctorProfileActivity" />  
 <activity  
 android:name="doctorPatientCommon.DoctorLoginActivity"  
 android:theme="@style/DoctorLoginTheme"  
 android:windowSoftInputMode="adjustResize" />  
 <activity  
 android:name="admin.RegisterRoleActivity"  
 android:parentActivityName="admin.ChooseRoleActivity"  
 android:windowSoftInputMode="adjustResize" />  
 <activity android:name="admin.ChooseRoleActivity" />  
 <activity android:name="com.healthcare.helper.MainActivity" />  
 </application>  
  
</manifest>

2. **Splash Screen Activity Screen**

class SplashScreenActivity : AppCompatActivity() {  
 private lateinit var backgroundExecutor: ScheduledExecutorService  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_splash\_screen*)  
  
 backgroundExecutor = Executors.newSingleThreadScheduledExecutor()  
 backgroundExecutor.schedule(**{** *// 1) For Patient* val patientCredential = this.*getPatientCredentials*()  
  
 *// 2) For Doctor* val doctorCredential = this.*getDoctorCredentials*()  
  
 when {  
 patientCredential != 0.toLong() -> {  
 startActivity(Intent(this, PatientMainActivity::class.*java*))  
 finish()  
 }  
 doctorCredential != 0.toLong() -> {  
 startActivity(Intent(this, DoctorProfileActivity::class.*java*))  
 finish()  
 }  
 else -> {  
 startActivity(Intent(this, PatientLoginSignUpActivity::class.*java*))  
 finish()  
 }  
 }  
 **}**, 3, TimeUnit.*SECONDS*)  
 }  
  
 override fun onDestroy() {  
 super.onDestroy()  
 backgroundExecutor.shutdown()  
 }  
}

1. **Views**

2.1 Filtering the doctors

class DoctorFilterDialog(private val activity: Activity) {  
 private lateinit var alertDialog: AlertDialog  
  
 fun startLoadingDialog(viewGroup: ViewGroup) {  
 val builder = AlertDialog.Builder(activity)  
 val layoutInflater = activity.*layoutInflater* builder.setView(layoutInflater.inflate(R.layout.*doctor\_filter\_dialog*, viewGroup, false))  
  
 alertDialog = builder.create()  
 alertDialog.show()  
 }  
  
 fun stopLoadingDialog() {  
 alertDialog.dismiss()  
 }  
}

1. **Functionality**

**3.2 If doctors exists then Login else sighnup and add to the database on click of the button**

class RegisterRoleActivity : AppCompatActivity() {

…

private fun handleClickListeners() {  
 binding.RoleSignUpBtn.setOnClickListener **{** val doctorUsername = binding.RoleSignUpUsernameEt.*text*.*toString*()  
 val doctorPhone = binding.RoleSignUpPhoneEt.*text*.*toString*()  
 val doctorPassword = binding.RoleSignUpPasswordEt.*text*.*toString*()  
 val doctorDegreeName = binding.AutoCompleteDegreeDropDown.*text*.toString()  
 val graduatedIn = binding.DoctorDegreeExpiryDateBtn.*text*.toString()  
  
 *// string.xml variables* val expString = *resources*.getString(R.string.*select\_date\_of\_doctor\_degree\_expiry*)  
  
 if (doctorUsername.*isEmpty*() || doctorPassword.*isEmpty*() || doctorPhone.*isEmpty*() || doctorDegreeName.*isEmpty*() || graduatedIn.*equals*(  
 expString, true  
 )  
 ) {  
 *getToast*(  
 this,  
 "All Fields Are Mandatory",  
 Toast.*LENGTH\_SHORT* ).show()  
 if (doctorPhone.*isEmpty*())  
 binding.RoleSignUpPhoneEt.*error* = "Phone Number is Mandatory"  
 if (doctorPassword.*isEmpty*())  
 binding.RoleSignUpPasswordEt.*error* =  
 "8 Characters Password is Mandatory"  
 if (doctorUsername.*isEmpty*())  
 binding.RoleSignUpUsernameEt.*error* = "Username is Mandatory"  
 if (graduatedIn.*equals*(expString, false))  
 binding.DoctorDegreeExpiryDateBtn.*error* =  
 "Degree Expiry Date has to be selected"  
 } else {  
 addDoctorDataToDatabase(  
 doctorUsername,  
 doctorPhone,  
 doctorPassword,  
 doctorDegreeName,  
 graduatedIn  
 )  
 }  
 **}**

**…}**

3.3 Doctor’s Login Activity

class DoctorLoginActivity : AppCompatActivity() {  
  
 private lateinit var binding: ActivityDoctorPatientLoginBinding  
 private lateinit var backgroundExecutor: ScheduledExecutorService  
 private val doctorDao by *lazy* **{** (this.*application* as ApplicationClass).doctorDao **}** private val context: Context by *lazy* **{** this **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityDoctorPatientLoginBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 *// show Custom Dialog after clicking Login Button* val loadingDialog = LoadingDialog(this)  
  
 *// Request Focus For Phone Number Field Automatically* binding.DoctorLoginPhoneEt.requestFocus()  
  
 *// BackGround and Main Executor* val mainExecutor: Executor = ContextCompat.getMainExecutor(this)  
 backgroundExecutor = Executors.newSingleThreadScheduledExecutor()  
  
 binding.DoctorLoginBtn.setOnClickListener **{** val phone = binding.DoctorLoginPhoneEt.*editText*!!.*text*.toString()  
 val password = binding.DoctorLoginPasswordEt.*editText*!!.*text*.toString()  
 if (TextUtils.isEmpty(phone) || TextUtils.isEmpty(password)) {  
 *getSnackBar*(  
 binding.SnackBarLinearLayout,  
 getString(R.string.*empty\_field\_error*)  
 ).show()  
 showEtError(phone, password)  
 } else {  
 loadingDialog.startLoadingDialog(binding.SnackBarLinearLayout)  
*// val check = Database.checkDoctorLogin(phone, password)  
 lifecycleScope*.*launch* **{** val check = doctorDao.doctorLogin(phone, password)  
 loadingDialog.stopLoadingDialog()  
 when {  
 check.size > 1 || check.isEmpty() -> mainExecutor.execute **{** *getSnackBar*(  
 binding.SnackBarLinearLayout,  
 "Invalid Login Found !!!"  
 ).show()  
 **}** check.size == 1 -> {  
 *// Save Data To Doctor Shared Preferences* val sharedPreferences = *getDoctorSharedPreferences*(context)  
 val editor = sharedPreferences.edit()  
 editor.putLong(*DOCTOR\_CREDENTIAL*, check[0])  
 editor.apply()  
 context.*openActivity*(DoctorProfileActivity::class.*java*)  
 finish()  
 }  
 }  
 **}** }  
 **}** }

3.4 Filtering the doctors

1. **Models**

@Dao  
interface DoctorDao {  
 @Insert  
 suspend fun insertDoctor(doctor: DoctorEnt): Long  
  
 @Query("Select doctorId from DoctorEnt where doctorPhone =:doctorPhone and doctorPassword=:doctorPassword limit 1")  
 suspend fun doctorLogin(doctorPhone: String, doctorPassword: String): List<Long>  
  
 @Query("Select \* from DoctorEnt where doctorId=:doctorId limit 1")  
 fun getDoctorById(doctorId: Long): LiveData<List<DoctorEnt>>  
  
 @Transaction  
 @Insert(onConflict = OnConflictStrategy.*IGNORE*)  
 suspend fun insertCertifications(certifications: List<CertificationEnt>): List<Long>  
  
 @Insert  
 suspend fun insertAvailabilities(availabilities: List<AvailabilityEnt>): List<Long>  
}

**4. Models**

**CONCLUSION**

**REFERENCES**

1. <https://www.tutorialspoint.com/android/android_user_interface_layouts.htm>
2. <https://developer.android.com/guide/topics/media/camera>
3. <https://developer.android.com/studio>
4. <https://www.tutorialspoint.com/android/android_location_based_services.htm>
5. <https://drawio-app.com/entity-relationship-diagrams-with-draw-io/>
6. <https://en.wikipedia.org/wiki/Database_normalization> - 2NF