

Senior Design Project Report

CSE/EEE/ETE 499B

COVID-19 TRACKER BD

APPLICATION BASED ON ANDROID



Submitted By

161 1205 042 Aaquib Javed

161 1231 042 Md. Sayem Mahmud

133 0056 043 Shumaiya Karim

152 0082 043 Zahir Mohammad Mishaal

Supervisor

Syed Athar Bin Amir – SAA3

Lecturer, Department of Electrical & Computer Engineering (ECE)

Faculty Sponsor, NSU ACM Student Chapter

ELECTRICAL AND COMPUTER ENGINEERING

NORTH SOUTH UNIVERSITY

FALL 2020

Agreement Form

We take great pleasure in submitting our senior design project report on COVID-19 Tracker BD Application Based On Android. This report is prepared as a requirement of the Capstone Design Project CSE/EEE/ETE 499 A & B which is a two semester long senior design course. This course involves multidisciplinary teams of students who build and test custom designed systems, components or engineering processes. We would like to request you to accept this report as a partial fulfillment of Bachelors of Science degree under Electrical and Computer Engineering Department of North South University.

Declared By:

.....
Name: Aaqib Javed
ID: 161 1205 042

.....
Name: Md. Sayem Mahmud
ID: 161 1231 042

.....
Name: Shumaiya Karim
ID: 133 0056 043

.....
Name: Zahir Mohammad Mishaal
ID: 152 0082 043

Approved By:

.....
Supervisor
Syed Athar Bin Amir
Lecturer, Department of Electrical and Computer Engineering
North South University, Dhaka, Bangladesh

.....
Dr. Rezaul Bari
Chair, Department of Electrical and Computer Engineering
North South University, Dhaka, Bangladesh

COVID-19 TRACKER BD **APPLICATION BASED ON ANDROID**

Abstract

To understand the exact coronavirus outbreak and the way it quickly surges worldwide, many countries are adopting non-therapeutic preventive measures, which include travel bans, remote work, complete country lock down, regular update of this pandemic, and most importantly, social distancing. However, these measures face challenges in Bangladesh known as the world's densest populations. This war like situation, a lower-middle-income economy put this country and the government in a big challenge to implement the mitigation facilities. As a densest population country it is quite a challenge to maintain social distancing in many areas of the country. At a local level mobile sanitization, temporary quarantine sites and healthcare facilities could help mitigate the impact of the pandemic, and health experts, along with international assistance can enable the country to minimize the impact of the pandemic. To minimize the impact and to spread more awareness in the country we need to take more steps for spreading the awareness, so in this modern era the manuscript authors taking a new steps that is **android COVID-19 tracking system**, it could be very helpful for the people and make them more aware of this pandemic situation and also this could be helpful for the government and UN to understand and get the exact data of COVID-19.

GitHub repository: https://github.com/psych094/499_covid_tracker

Table of Contents

SL:		Page No
	Chapter 1: Introduction	5
1.1	Project Details	6
1.2	Background and Motivation	7 - 9
1.3	Project Goal	9
	Chapter 2: Technical Details	10
2.1	Existing Solution	11
2.2	Proposed Solution	12
2.3	Solution Assessment	13
2.4	Design Alternative	14
2.5	Technical Design: Module Level	15 - 20
2.6	Technical Design: System Level	21 - 24
2.7	Required Skill	24
	Chapter 3: Tools Used	25
3.1	Description of Components	26 - 30
3.2	Test Equipment	31
	Chapter 4: Working Sheets	31
4.1	Work Breakdown Structure	33 - 35
4.2	Financial Plan and Costs	36 -38
	Chapter 5: Project Summary	39
5.1	Result and Discussion	40
5.2	Feasibility Study	40
5.3	Problem Faced and Solutions	41
5.4	Future Development	42
5.5	Conclusion	42 - 43
5.6	Brochure	43
5.7	Poster	44
5.8	IEEE Format paper	45 - 49
	Appendix	50
A	Reference	50 - 52
B	Code	53 - 141

CHAPTER 1

INTRODUCTION

1.1 Project Details:

What is a covid-19 tracker?

In any scenario, a tracker is a method or device that can determine where someone or something is. As the novel coronavirus, also known as covid-19, became a global pandemic, it became essential to back trace where and how one has come in contact with this virus. It is said that it is better to be safe than sorry therefore, we strategize to find a way to eliminate the element of surprise after contracting the virus. We brainstormed on how we can take preventive measures against it. This brings us to discuss our problem domain, where we analyzed what the issues were and using our collective knowledge and understanding of our studies, we implemented it into a viable solution, which was building an application that will detect any covid-19 infected patients in an area and notify nearby users if they fall in the danger zone or are coming into contact. Many countries are adopting preventive measures, e.g., remote office activities, international travel bans, mandatory lockdowns, and social distancing. Bangladesh, a lower-middle-income country and one of the world's most densely populated areas, is struggling to combat the spread of the disease. In this write-up project details, we briefly articulate the current scenario of COVID-19 in Bangladesh and provide some recommendations with new technological solutions on how the country can combat this pandemic. During this pandemic, everything, outside the comfort of our own homes is unpredictable and this, in turn, has instilled a fear within us especially when we are going out for necessities or if a relative comes over. To overcome this state of uncertainty, I would like to propose an application that lets everyone use it to search for possible infected people who are nearer. An app named after covid-19 GPS Tracking System which is based on android and installed in the phone and running a background service where the GPS location and time is stored and sent to a central server. Any individual who is reported as covid-19 positive is to be checked where he/she is in the real time and providing exact COVID-19 positive user number within a certain range to all covid-19 non positive users while they are roaming around before infecting. In this application it will also store the data of every user which will help the government and the UN organization for making a good survey and also will help them both to take the necessary steps to prevent this situation and spread awareness. Also by the help of the government and their covid-19 data this application can work more efficiently.

1.2 Background and Motivation:

Background:

During COVID-19 pandemic situation almost every country shared their every COVID case to the UN organization. Through UN organizations people can get every covid-19 information in detail through a web-based dashboard. People can easily get the information by visiting the website dashboard but the dashboard only provides information based on how many new covid-19 cases found with how many people get cured and also the death case. The dashboard also provides the visitors in what way they can identify covid-19 symptoms and possible safety rules. Till the middle of the 2020 everyone got updates regarding covid-19 based on WHO web-based dashboard but after the middle of 2020 many of the software farms and entrepreneurs started to build applications based on covid-19. Here some of the covid-19 applications details are given with their advantages and disadvantages which also help us and give our project group a great motivation to build an application based on this highly demanded topic. [11]

1. **WHO:** The world health organization web-based dashboard is the first source of information which provides how many people were affected with covid-19 virus during the initial period of 2019. This website provides people the symptoms regarding covid-19 virus and its initial possible remedies. But only the information can't make that much awareness among the people all over the world also with basic information of symptoms and its safety rules cause people failed to know who's got affected next to him/her. So, basically it was the primary step of awareness among the people in the beginning but is it enough to take the precaution and get other people in the safety zone? [11] NO, so there should be another option which will help man to man information who are affected and someone next to me who is affected by covid-19.

2. **Coronavirus Contact Tracing Apps:** Health departments use contact tracing apps to find people who may have come into contact with someone with COVID-19. Apps help to capture data and watch the movement of people to make the process faster and more useful. Some could identify people who might have been exposed to the virus so they know to isolate themselves and watch for symptoms. Tech giants Apple and Google have teamed up on a platform that uses Bluetooth and a phone's operating system for contact tracing. Some people are wary of an app that tracks where they go and whom they meet. Instead of storing data on a central server that

may be vulnerable to hackers, Google and Apple say their apps won't be able to read the raw data themselves. Instead, the information will be available only to health agencies through what's called an application-programming interface (API). [12] But is this all sufficient? Cause the person who is roaming around the city, country or else is affected or not is not confirmed. So, there is also a huge amount of data and information working but in a different manner and we can use it to work in an appropriate manner where every person covid-19 case are stored and give them a unique number using a mobile application so that it will help people around the globe to move safely.

3. COVID-19 Plasma Finder: In this application it is only to help people to find covid-19 virus survival persons who can donate their plasma blood to the new covid-19 patient to get cured. But on the other side it won't help people to take themselves in a safe zone. This application is mainly built in Bangladesh by some new entrepreneurs and many similar applications are made regularly but no applications have such effective features to help the users effectively.

4. Canadian Covid-19 Alert: Now this is the application which is truly helping its users by its own features to know covid-19 and its every situation with helpful information along with the tracking the nearest covid-19 patient who were affected. Recently in 2021 Canada launched their application for their own regional people. This application is not publicly launched all over the world but only for their own country. [13] So, basically the world needs a common application where they can find each and every helpful information which is helping them perfectly.

Motivation:

During this pandemic, everything, outside the comfort of our own homes is unpredictable and this, in turn, has instilled a fear within us especially when we are going out for necessities or if a relative comes over. The main motivation for this app was to overcome this state of uncertainty and be able to maintain some sense of security, awareness and the severity of our current state of affairs. This app is an idea in trying to help people in better understanding their surroundings as simply taking the basic, and usually easy, precautions of using a face mask and carrying sanitizers isn't enough. The number of individuals, infected or not, in the area also have to take into account the chances of getting infected; number of known infectees also affect how possible unknown infectees there are too, being the case of designing this app.

1.3 Project Goal:

Keeping our goal in mind, we researched what steps to follow and how to progress with building this application. We analyzed our requirements and over the course of eight months, worked towards achieving those key features. We made a list of features that would be user friendly and informative at the same time. One of the major requirements for our application was its efficiency. Hence, we modeled a draft design for our outlook and developed on implementing those on the user and server interface. Below are the list of features we developed and the model design we created in our mission to attain it.

- Splash Screen
- Sign Up/Log in
- Mobile Phone Verification OTP(Using Firebase)
- Location Tracking/ Map view
- Dashboard UI
- Navigation Drawer
- Personal Data Retrieve
- Total COVID patient detection on given range
- Send notification if any Covid patient within a 5 meter range of users.

CHAPTER 2

TECHNICAL DESIGN

2.1 Existing Solution:

During this pandemic situation every country is trying their best to give the people an exact solution. UN organizations trying to produce the vaccine for this virus also are trying to spread the awareness for COVID-19. So many countries all around the world share their exact COVID-19 data to UN organizations. So, this is going to help the UN to know the virus and how it operates in the human body. In terms of solution there is no hard solution worldwide like android based GPS tracking (COVID case information). The only way that people have they can only get updates of cases through a web based dashboard. In Bangladesh there is an application by the government of Bangladesh which helps the nationwide people to know how to take corona-virus precaution and what types of medicines can help them and the hospitals details that are dealing with COVID-19 cases and provide the patients treatment. But still in Bangladesh there are no such applications which can help the people to notify the nearest COVID-19 case. The dashboard also provides the visitors in what way they can identify covid-19 symptoms and possible safety rules. Till the middle of the 2020 everyone got updates regarding covid-19 based on WHO web-based dashboard but after the middle of 2020 many of the software farms and entrepreneurs started to build applications based on covid-19. Here some of the covid-19 applications details are given with their advantages and disadvantages which also help us and give our project group a great motivation to build an application based on this highly demanded topic. [11] So, there are plenty applications are launched during this pandemic but none of them are completely that much beneficial which people actually need and those applications are WHO Dashboard, Coronavirus Contact Tracing Apps, COVID-19 Plasma Finder and Canadian Covid Alert (Regional Application) etc.

2.2 Proposed Solution:

In our application which is named COVID-19 GPS Tracking System Application based on Android will help worldwide people that who is next them are COVID-19 and the exact location of that person with the distance. This application also helps the countries government to get exact data of COVID-19 particularly and also can send them the proper treatment and support. In our application the user can simply signup by his number and set up some basic information about the user. If the application user tested COVID-19 negative then it will update his status and also if any other user next to that person are being tested COVID-19 then it will notify that other user of a certain amount of distance in a radios will be automatically notified that there is someone who is tested positive and the distance with the location. Generally this application's main focus is to spread awareness among all the people about their nearest position. In this pandemic, everything, outside the comfort of our own homes is unpredictable and this, in turn, has instilled a fear within us especially when we are going out for necessities or if a relative comes over. The main proposed solution in this situation was to overcome this state of uncertainty and be able to maintain some sense of security, awareness and the severity of our current state of affairs. This app is an idea in trying to help people in better understanding their surroundings as simply taking the basic, and usually easy, precautions of using a face mask and carrying sanitizers isn't enough. The number of individuals, infected or not, in the area also have to take into account the chances of getting infected; number of known infectees also affect how possible unknown infectees there are too, being the case of designing this app.

2.3 Solution Assessment:

Though many countries are trying to generate more solutions for this pandemic situation but still there is no such application which will help the people to get notified of the nearest information about these COVID-19 cases. The few existing solutions are only based on how to take precaution and the worldwide details. So, in terms of this our application hopefully will be more helpful for every user all around the globe. If we look at the software or applications hub there are no such applications with this kind of feature where the application user surrounding can get notified through the application that he/she is covid-19 positive case or negative case. It's an open covid case navigation system where the user gets each and every update based on covid and also can easily navigate the patient or identify in what distance someone is positive if it is really.

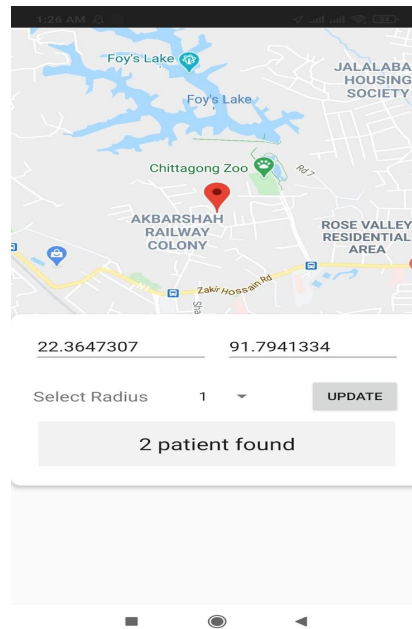


Fig 1.01: User Navigating Some Patients

2.4 Design Alternative:

If we get govt support like having covid test result data then we are planning to take user covid status update control to us. If any patient is covid victim according to Govt test data and if he/she doesn't install the application , they will get an alert message on their device to install. Once a covid victim installed the app he/she doesn't need to control their covid status there will be automatically updated syncing with the database from the beginning , the bluetooth and GPS will start work automatically until the user recovers from covid. He/she(covid patient) also can't uninstall/logout unless he/she recovered or again logged in with the same number on another device. Because by one mobile number you can register on only one device at a time .The new logged in device will work like earlier one until you recover. If users recover but the system shows them positive yet then they can send a system verification request of their covid status through giving their test result id. Then the system administrator will work following on this objection.

According to Bangladesh environment people are not willing to mark them covid infected. Because our government initially faced so much travel to make people understand about the benefit of masks. So same to our app, but our young generation shared their positive review with us. So our young generation and govt should come up together and cooperate with others , thus we could accomplish our target and reactivate the whole nation.

2.5 Technical Design: Module Level

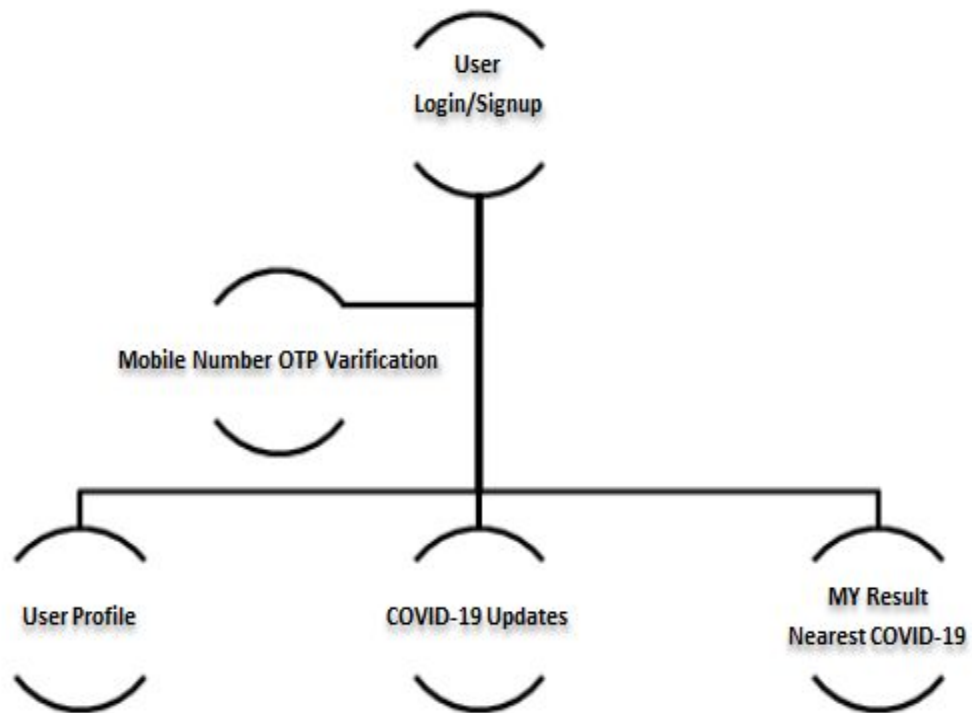


Fig 1.02: Module Diagram

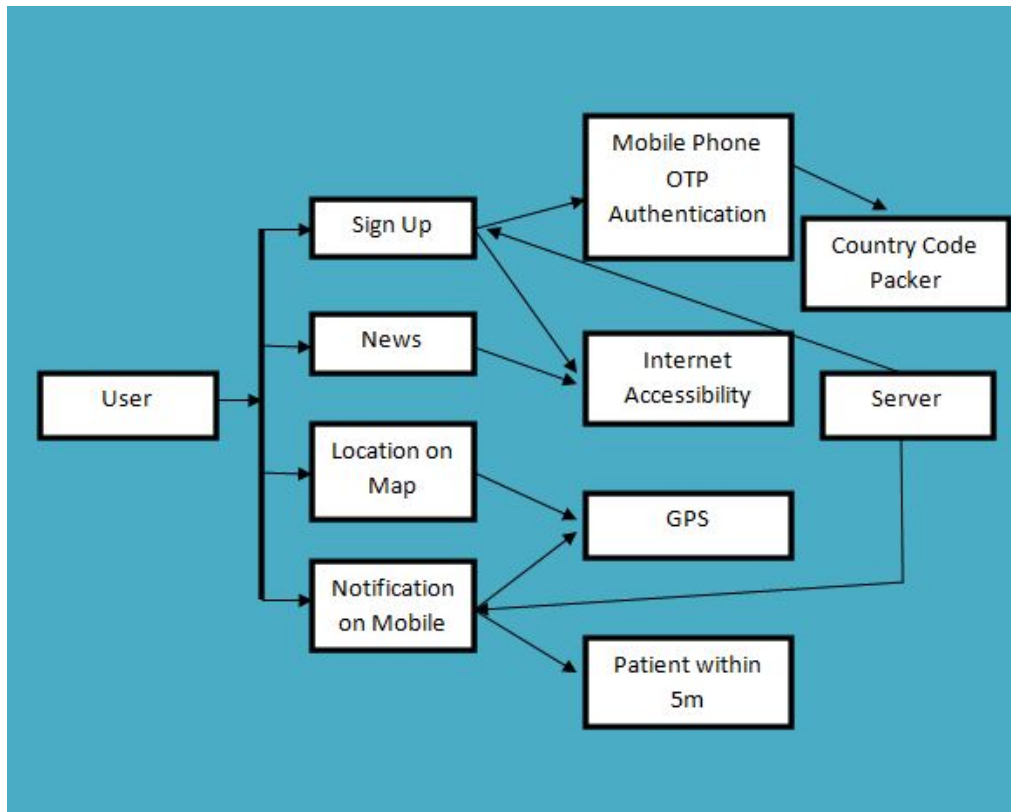


Fig 1.03: Use Case Diagram

Here we have our two active actors user and system server. After user signing up to the system there they would face a mobile phone OTP verification. After verification completion user's given will be stored on the database and the dashboard UI will open to them. Dashboard UI is full of components like searching anything etting all kinds covid related updates and news. Users can get his location by google map and can see the total numbers of affected patients nearer to them .If any patients come nearer to them within 5 meter , then the user will get an alarming notification on mobile.For this location tracking and notification system user must have GPS enabled. The whole system also required internet accessibility .

APPLICATION UI/UX

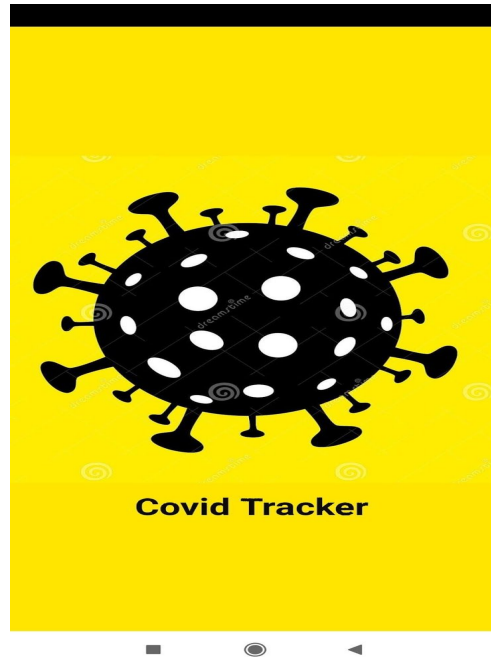


Fig 1.04: Splash Screen

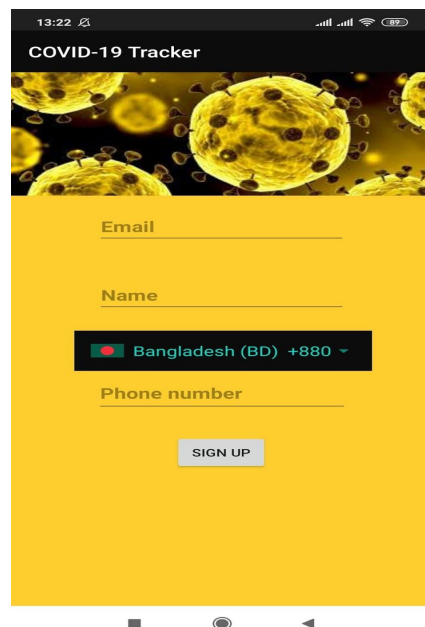
The image shows a mobile application signup page. The background is yellow. At the top, there is a black header bar with the text "COVID-19 Tracker" in white. Below the header, there is a banner image showing several yellow, spherical virus particles with black protrusions. Below the banner, there are four input fields: "Email", "Name", "Phone number", and a dropdown menu for "Bangladesh (BD) +880". Below the input fields, there is a "SIGN UP" button. At the bottom of the screen, there are three small, faint icons: a square, a circle, and a triangle.

Fig 1.05: Signup page

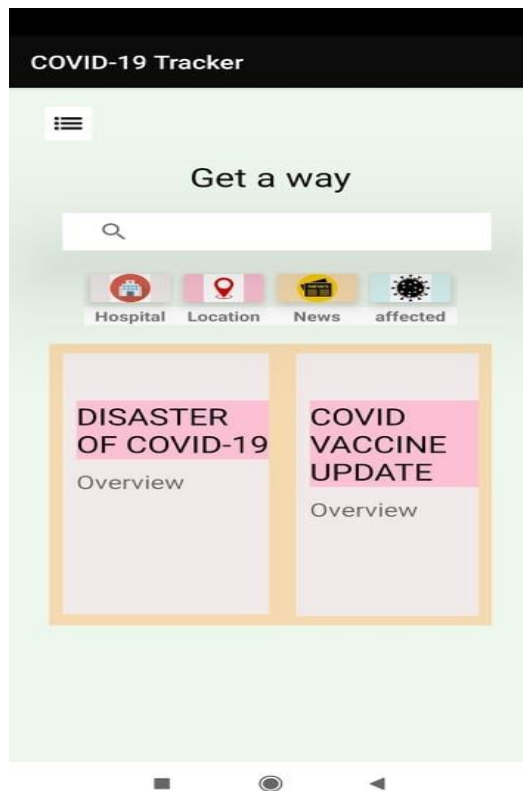


Fig 1.06: Dashboard UI

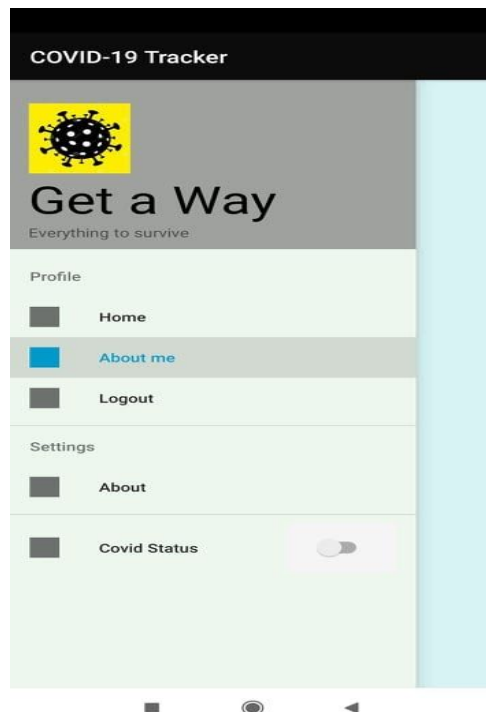


Fig 1.07: Navigation Drawer

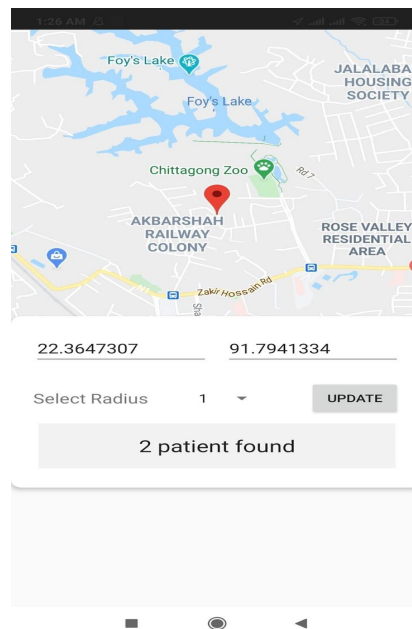


Fig 1.08: Location Tracking

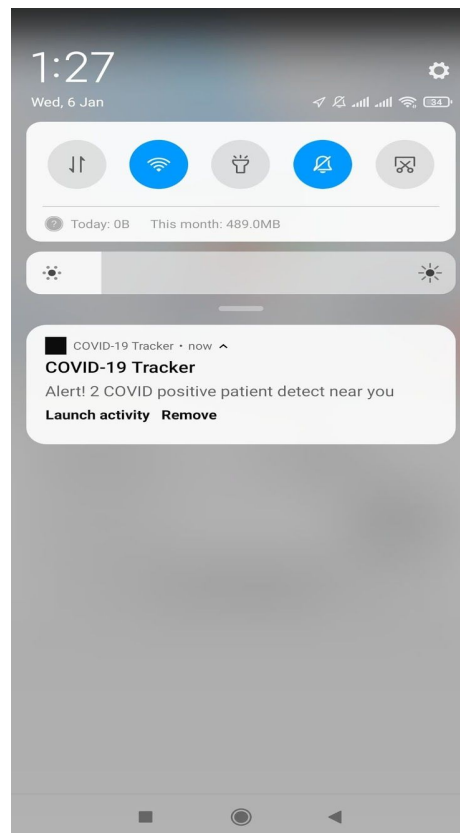


Fig 1.09: Alarming Notification(5M)

Database

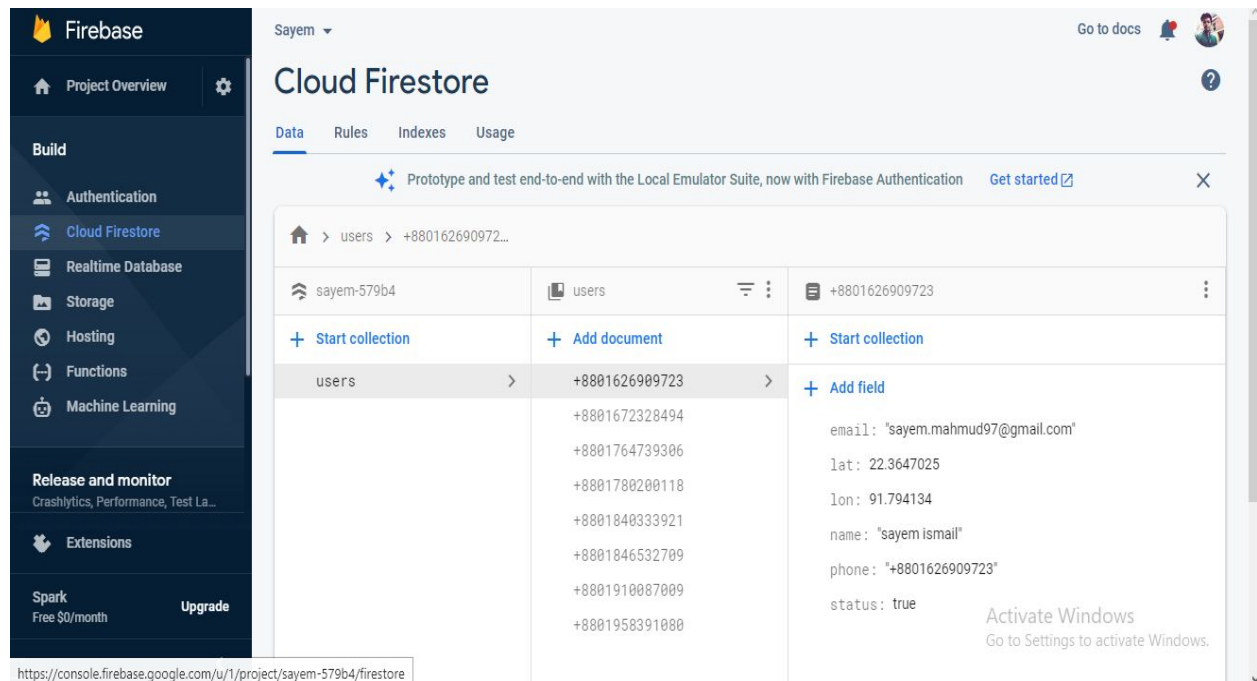


Fig 1.10: Database Functionality

On dashboard UI/UX users can search anything on the search bar which will go through Google search . And can get nearer medical support options, world wide covid storm update, Vaccine update & research related news also.

Navigation Drawer is providing some facilities like seeing personal information , how to operate this app, log out & for user's Covid status updation.

Location Map is for showing a user's current location on a map with proper latitude and longitude including total number of covid affected patients within selected range.

Notification is for alert. The cloud firebase system is for location latitude & longitude updated again and again after 3 seconds frequently so that we could get notification currently on time. Which is quite the same pace like bluetooth.

2.6 Technical Design: System Level

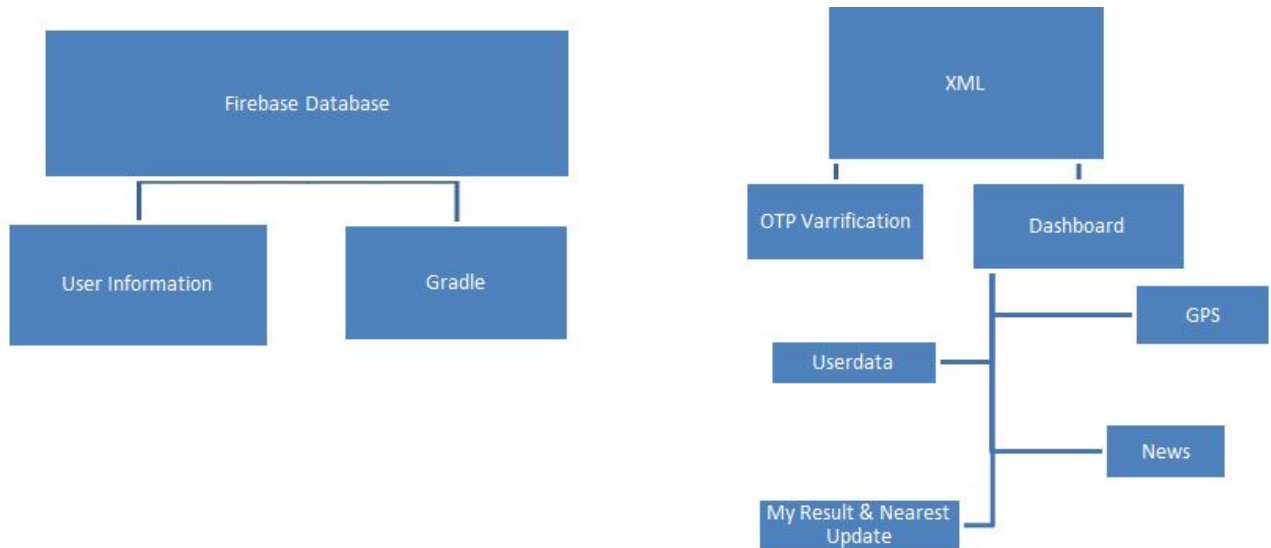


Fig 1.11: Technical Design of System Level

Calculation

Haversine formula: The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. It results in an error of up to 0.5%. The Haversine Formula method is used to determine the distance between two users' location on android. Haversine formula. Distance d is a function of two latitude and longitude coordinates (ϕ_1, γ_1) and (ϕ_2, γ_2) . Where r is the radius of the Earth. For example, $\text{haversine}(\theta) = \sin^2(\theta/2)$. The **haversine formula** is a very accurate way of computing **distances** between two points on the surface of a sphere using the latitude and longitude of the two points.

$$d(\phi_1 \gamma_1 \phi_2 \gamma_2) = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\phi_2 - \phi_1}{2} \right) + \cos(\phi_1) \cos(\phi_2) \sin^2 \left(\frac{\gamma_2 - \gamma_1}{2} \right)} \right)$$

Distance

Formula: $\Delta\psi = \ln(\tan(\pi/4 + \phi_2/2) / \tan(\pi/4 + \phi_1/2))$ ('projected' latitude difference)

$q = \Delta\phi/\Delta\psi$ (or $\cos\phi$ for E-W line)

$d = \sqrt{(\Delta\phi^2 + q^2 \cdot \Delta\lambda^2)} \cdot R$ (Pythagoras)

where ϕ is latitude, λ is longitude, $\Delta\lambda$ is taking the shortest route ($<180^\circ$), R is the earth's radius, \ln is the natural log.

- We limit the database updation range within 3 sec after and after for location latitude & longitude.

Code:

```
/**
```

```
* The desired interval for location updates. Inexact. Updates may be more or less frequent.
```

```
*/
```

```
private static final long UPDATE_INTERVAL_IN_MILLISECONDS = 5000;
```

```
/**
```

```
* The fastest rate for active location updates. Updates will never be more frequent
```

```
* than this value.
```

```
*/
```

```
private static final long FASTEST_UPDATE_INTERVAL_IN_MILLISECONDS =  
UPDATE_INTERVAL_IN_MILLISECONDS / 2;
```

- After Location updation on database system frequently search for nearer patient within 5m range as for 5000m we have taken 5 so for 5m meter the value would be .25 that's what we used for our logical operation:

Code:

```
public void notify(LatLng currentLocation){
    FirestoreUtil.getInstance().getDocumentRef().collection("users")
        .whereEqualTo("status", true)
        .get()
        .addOnCompleteListener(task -> {
            List<DocumentSnapshot> documents = task.getResult().getDocuments();
            int count = 0;
            for (DocumentSnapshot snapshot : documents){
                Double latitude = (Double) snapshot.get("lat");
                Double longitude = (Double) snapshot.get("lon");
                if (latitude != null && longitude != null){
                    LatLng latLng = new LatLng(latitude, longitude);
                    float distance = LocationUtil.getInstance().findDistance(currentLocation, latLng);
                    if (distance <= 0.25){
                        count ++;
                        Log.d(TAG, "notify: " + distance);
                    }
                }
            }

            // if count > 0 that means covid positive patient found
            // then a notification will be showing for notify the user
            if (count > 0){
                mNotificationManager.notify(NOTIFICATION_ID,
```

```
        getNotification(String.format("Alert! %s COVID positive patient detect near  
you", count)));  
    }  
});
```

source: LoctionUpdatesService.java

We have used some functionalities for time duration, database updation, shared preference & navigation drawer also .If any information needed you can check the data sheet segment of the report.

2.7 Required Skills:

In this project the required skills are Java Programming Language, Android Studio, XML Studio, Firebase account management to configure the Firebase Authentication, Firebase Database, Country Code picker, SDK, Gradle. These are the required skills and knowledge which will help to finish the project on time.

Moreover required Generating GOOGLE API key, Channel ID, Notification channel creation, location map, map marker, haversine method, fused location operation, Ibinder, frcoder, hashmap<>, Gps enability , listing, JSON file operating, SHA1 key generating, Distance between two point method, Cloud firestore operating etc which we were discussed elaborate on chapter 3 segment.

CHAPTER 3

ESSENTIAL PARTS AND DEVICES

3.1 Description of Components:

The Table provides the description of components bellow:

Tools	Description of Tools
Java programming language	Using the Java programming language for the programming of the Android app.
Android Studio	Using Android Studio to provide the Integrated Development Environment (IDE) for the development of the Android application.
XML	Using XML to design the UI for Linear Layout, Relative Layout and Constraint Layout.
Firebase	Using Firebase for the provisioning of a backend system for the Android application for easy development of user login and data storage.
Firebase Authentication	Use of the Firebase Authentication for the registration and login of users.
Firebase Database	Using the Firebase Database for the storage of information.
Cloud Firestore	Cloud Firestore is a flexible cloud database. It is used for mobile, web, and server development from Google Cloud Platform and Firebase. Like the Firebase Real-time Database, it keeps syncing our data via real-time listeners to the client app. We have used it to save our real time longitude & latitude data to database.
Country Code Picker	Using the country code picker library for mobile SMS verification. Set to automatically default to the country of the

	user's current location. SMS verification to be automatically detected when received on the same device.
SDK	Required libraries to build Android applications.
Gradle	Building system for code compilation, library importing, testing, deployment and running the app on the device which should be preinstalled on Android Studio. Updating dependencies as requirements was also there.
Google Maps API	<p>The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. You can also use API calls to add markers, polygons, and overlays to a basic map, and to change the user's view of a particular map area. These objects provide additional information for map locations, and allow user interaction with the map. The API allows you to add these graphics to a map:</p> <ul style="list-style-type: none"> ● Icons anchored to specific positions on the map. ● Sets of line segments. ● Enclosed segments. ● Bitmap graphics anchored to specific positions on the map ● Sets of images which are displayed on top of the base map tiles.
Tracking Tools	Location-based tools are based on GPS location of users. They may be used to identify people who have been in the same location as cases, to facilitate contact identification
Haversine Method	Haversine formula. Distance d is a function of two latitude and longitude coordinates (ϕ_1, λ_1) and (ϕ_2, λ_2) . Where r is the radius of the Earth.

	$d(\varphi_1 \gamma_1 \varphi_2 \gamma_2) = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\varphi_2 - \varphi_1}{2} \right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2 \left(\frac{\gamma_2 - \gamma_1}{2} \right)} \right)$
GPS	Each GPS satellite continuously broadcast a signal with information about its location as well as its current clock time. GPS receivers listen to this broadcast and through calculating the current location of the satellites via propagation delay and triangulation, its location relative to the satellites can be calculated. This location can then be transformed into a latitude and longitude coordinate on Earth.
Channel ID	For sending notification on device. For each channel, you can set the visual and auditory behavior that is applied to all notifications in that channel
Action Broadcast	When any service is running on a device then to receive data by another device it works on that.
Ibinder	Instance which works to initiate service through the localbinder method. The LocalBinder provides the getService() method for clients to retrieve the current instance of LocalService. This allows clients to call public methods in the service.
Fused location service	It's an Google API which works for receiving Android GPS sensor data .
Fused location provider client	Object which provides the location data to the client.
Update interval milliseconds	While requesting an app for location track , there is a fused location service which updates the location with this time interval again and again. Fused location service basically
Notification ID	Needed to create a notification id like notification channel.

Location Request	It's an object which requests for location service.
Notification Manager	Passes the notification through the channel.Notification channel object initiates there.
RemoveLocationUpdates()	Method which removes location update
NotificationCompat	A class structured to encapsulate a named action that can be shown as part of this notification.
getNotification()	Returns the NotificationCompat used as part of the foreground service.
HashMap<.>	HashMap is a type of Collection, that stores our data in a pair such that each element has a key associated with it. The pair of key and value is often known as Entry and these entries can have only unique keys.
Distancebetween()	MethodComputes the approximate distance in meters between two locations, and optionally the initial and final bearings of the shortest path between them. float.
Geocoder	Geocoding is the process of transforming a street address or other description of a location into a (latitude, longitude) coordinate. The Geocoder class requires a backend service that is not included in the core android framework.
List<address>	The list(object) static factory methods provide a convenient way to create immutable lists.
setRequestingLocationUpdates()	Stores the location updates state in SharedPreferences. The location update states works on requestingLocationUpdates.

checkGpsEnabled()	Method checks either the GPS connection is enabled or not.
Shared Preference	Shared Preferences is the way in which one can store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage such as String, int, float, Boolean that make up your preferences in an XML file inside the app on the device storage.
onVerificationCompleted()	Here it's used on mobile OTP verification. By this method if the verification is valid and completed
Values	The res/values folder is used to store the values for the resources that are used in many Android projects to include features of color, styles, dimensions etc.
SHA1 key	To give access to others on my database we had to import their generated sha1 key on our database. Thus we cooperated with each other.
JSON file	JSON stands for JS Notation. It is used to interchange data from the server to the desired place.
Terminal	We used this terminal to operate Github .

3.2 Test Requirements:

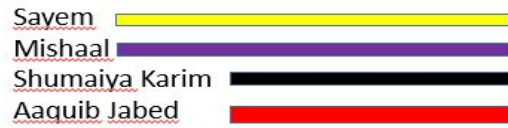
The application test requirements are given below:

Android Emulator	After the work to follow work progress in Android.
Mobile Phone(s)	Testing of the built application on physical devices.
Logcat	Logcat is a command-line tool that shows a log of system messages, including stack traces when the device throws an error and messages that you have written from your app. One can also view log messages from the Logcat window in Android Studio.
Build	It compiles app resources and source code, and packages them into APKs that you can test, deploy, sign, and distribute.

CHAPTER 4
WORKING SHEETS

4.1 Work Breakdown Structure:

Each color indicates each person's individual participation on the project .



August Work Distribution

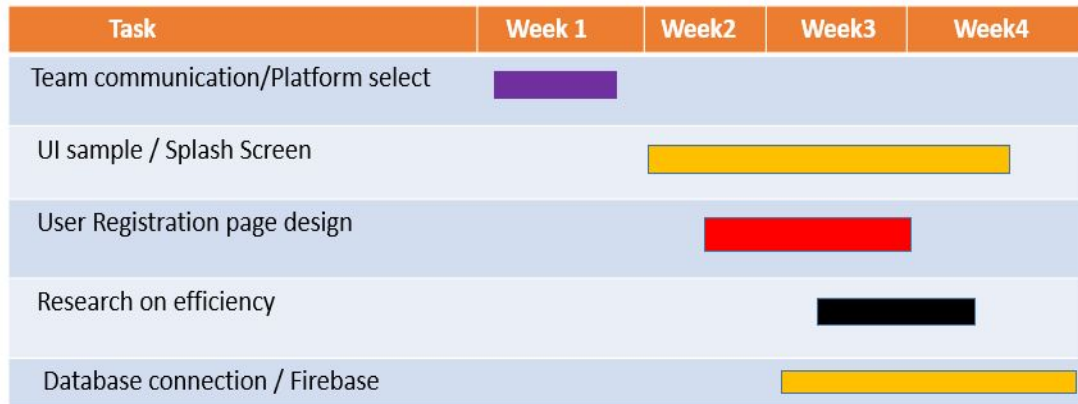


Fig 1.12: August Work Distribution

September Work Distribution

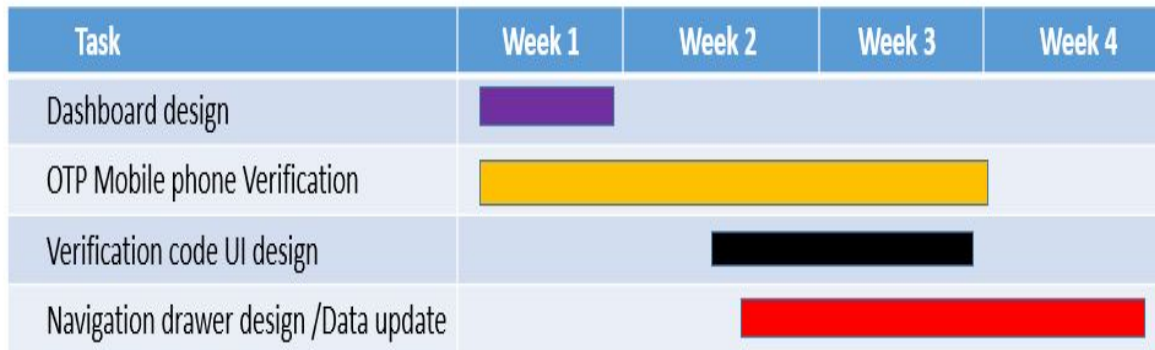


Fig 1.13: September Work Distribution

November Work Distribution

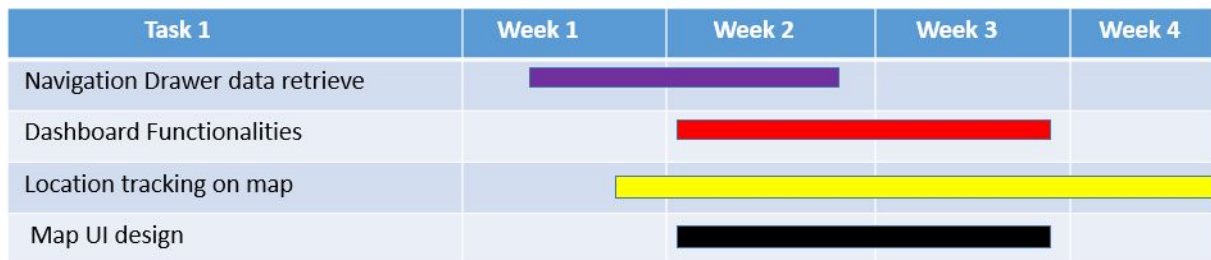


Fig 1.14: November Work Distribution

December Work Distribution



Fig 1.15: December Work Distribution

January Work Distribution

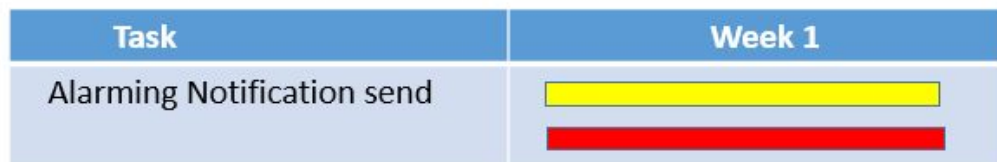


Fig 1.16: January Work Distribution

4.2 Financial Plan and Costs:

Initially we are using Free Mobile verification which is Google Firebase approved per Month 10,000 registration for free.

Then Google map offers \$200 monthly credit free for Maps, Routes and Places. We will use Google Maps Static for Android which is totally free for Android limited for 1000 requests per month. Until now our project didn't cost anything, after that before launching our App globally we will have to hire our desired packages.



Pricing			
Get \$200 in free usage for Maps, Routes, and Places every month			
MAPS	ROUTES	PLACES	
 Static Maps Display maps as images		FREE FOR MOBILE	\$2 PER 1000 REQUESTS
APIs	PRICE PER 1000 REQUESTS	USAGE	MONTHLY COST
Maps SDK for Android	FREE	Unlimited Loads	\$0
Maps Static API	\$2	Usage 	-

Fig 1.17: MAP Pricing

Deployment Cost on Playstore

For our next step we are planning to deploy our app at playstore. For experimentory and experience we want to deploy the app only in Bangladesh.Children below 13 can also use it. Firstly we will try for free version. Initially it's like a one time fee, we have to pay a sum of \$25 USDs around 2150TK BDT as it is the registration fee charged by Google. This is a one-time fee, which allows you to have a developer account and through that . We can deploy app as much as we can but we have to maintain the quality.

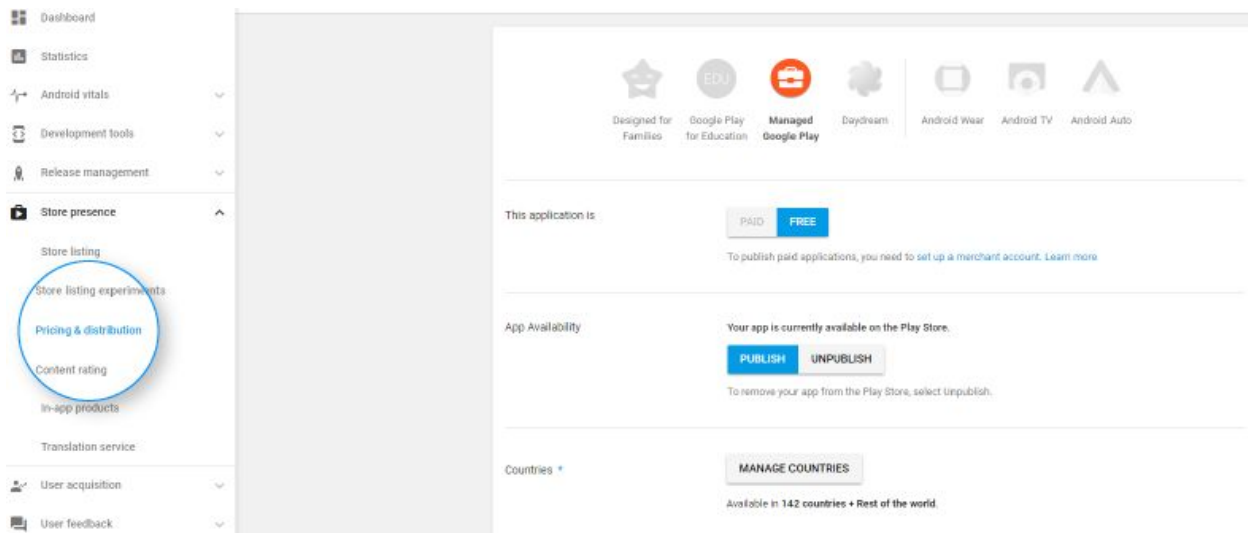


Fig 1.18: Deployment Cost

Promotion and maintenance cost

We are willing to promote our apps on social media like Facebook, Youtube, Instagram .Then we will be boosting our promotional post in a huge range so that we can reach our target audience. We have to make people concerned about this app and its advantages. There will be some requirement change and updatation . We have to efficiently maintain your application and introduce frequent updates, attending to the issues and bugs faced by users. Maintaining something is a constant task rather than a one-time thing. For that we fixed an amount around 220\$ for a year.

Sector	Cost (Yearly)
Google Mapping	500 TK BDT
OTP code send	500 TK BDT
Social Media Promotion	10000 TK BDT
Maintenance & Updation	9000 TK BDT
Developer Cost	30000 TK BDT
Total	50000 TK BDT

Fig 1.19:Yearly Cost

So here is the basic yearly cost planning for promotion & maintenance of our app. We will proceed following this in future.

CHAPTER 5

PROJECT SUMMARY

5.1 Result and Discussion:

As our project is a COVID-19 patient tracking system, the user must be registered and should mark themselves as COVID-19 positive or negative statement for the time being. If the app can be connected to the government database for the patient status, the users would not need to do a self-report. So we built our project from scratch, worked on a sign up, splash screen dashboard and logout system. The users can register themselves through the phone verification and the users can see their current location status. At the final stage they will be notified by the application that who is next to the user a covid-19 positive patient. This application provides 90% accuracy on detection as like bluetooth but according to Bangladesh environment people are not willing to mark them covid infected. Because our government initially faced so much travel to make people understand about the benefit of masks. So same to our app, but our young generation shared their positive review with us. So our young generation and govt should come up together and cooperate with others , thus we could accomplish our target and reactivate the whole nation.

5.2 Feasibility Study:

Financial perspective – Users can easily get an update of their surrounding area without any cost and as for deploying the app into the Google Play Store would have a \$25 developer fee.

Resource availability – We can collect a huge amount of COVID-19 patient data and can work on that for future research on what features could be added based on the available information.

Social importance – We can keep the members of the community alert as people are always concerned about their safety.

Safety issues – Private user information will not be publicly available to each other and with that users can not be tracked by other unauthorized parties.

5.3 Problem Faced and Solutions:

First of all we started work from scratch as a newcomer to the Android Sector. So as usual we faced lots of problems.

Syncing on Gradle caused some sync issues which were fixed after manually updating the Gradle system to 6.6.1. The emulator caused problems and with an update to the SDK version, the emulator was working.

The emulator also caused problems with a graphics card driver in a system and required an updated version of the graphics card driver but the drivers were already in the latest version. Whereas on the same system, running a Linux-based operating system the emulator didn't cause any problems.

Updated some dependencies for Firebase Authentication where we faced some problems to sync on gradle app.

Changed minimum SDK version and updated the SDK Compiler as required.

Faced problem firstly for free Google Firebase Mobile Phone Authentication without using country code picker. Then updated the country code picker library on Gradle App (dependencies) and worked on it.

We were unable to get the Bluetooth to function without having it to pair with other devices but later the application would crash when searching for other Bluetooth devices. As we were unable to get it working on time, we have decided not to add Bluetooth detection into the application.

Then faced issues on location updation on database, without clicking on location icon on dashboard the location updation didn't work on background we fixed the issues.

Faced a problem on counting patients in the 5 meter range as initially we gave the database location update limit for 10 seconds. Then we minimized the time. Then within 3 seconds after and after the database started getting updation , we came to get a more accurate result as bluetooth.

5.4 Future Development:

We have already worked on a dashboard which will be a term to view the user's current location on map, COVID-19 related news and can see nearby COVID-19 patient locations with a limited range. Users can easily update their information and statement. Consideration for the removal of the logout button for a patient who is COVID-19 positive. If we get govt support like having covid test result data then we are planning to take user covid status update control to us. If any patient is covid victim according to Govt test data and if he/she doesn't install the application, they will get an alert message on their device to install. Once a covid victim installed the app he/she doesn't need to control their covid status there will be automatically updated syncing with the database from the beginning, the bluetooth and GPS will start work automatically until the user recovers from covid. He/she(covid patient) also can't do uninstall/logout unless he/she recovered or again logged in with same number on other device. Because by one mobile number you can register on only one device at a time. The new logged in device will work like earlier one until you recover. If users recover but the system shows them positive yet then they can send a system verification request of their covid status through giving their test result id. Then the system administrator will work following on this objection.

So, overall we want the best possible accuracy level on serving & UI graphics of our projects in future thus it will be helpful not only in country but also worldwide.

5.5 Conclusion:

We were able to get started with making an app that does the basic things needed for the final state of our goals but there are still ways to make the app with more functionality. We added some important functionality in this project where the user gets notification if anyone roaming around is tested COVID-19 positive. So finally we became successful and made the application with its every initial important feature from the scratch and all the functions are working in its appropriate manner but also we need the government. support because only Bangladesh govt. organization IEDCR has a huge number of COVID-19 patient data resources. So, if the govt. shares their resources like Canadian government. then we can work it to the next level but apart from this our project can work manually with the same facilities. Not too many

functionalities were added to the app as to avoid bloating the application from main goal of having to display the user the number of COVID-19 positive patients to allow users to make better informed decisions on if they would like to continue to venture an area where a number of infectees are present. Further, with a higher number COVID-19 positive patients, the chances of there being unidentified COVID-19 infected would be exponentially higher.

5.6 Brochure:

Covid-19 Tracker

A little initiative to reactivate the world again!

App Benefits:

- User can get their exact location & total number of nearer COVID affected patients.
- User can limit the range according to their choice.

App Benefits:

Users will get a alarming notification, if any COVID affected patient come around within 5 meter of range.

Goal :

- Generally this application main focus is to spread awareness among all the people about their nearest position.
- Keeping users updated about COVID symptoms and movements all over the world.
- To get proper medial supports.
- Regenerating a safer country.

User Sevice

A user can get all in one like COVID status updatation to all kinds of COVID related news. Moreover all kinds of user friendly facilities are there.

[AVAILABLE NOW AT PLAYSTORE](#)

CONTACT US:

Block C, Bashundhara , Dhaka-1229
Bangladesh.
For more information :
Mobile: +8801626909723
Email: sayem.mahmud@northsouth.edu
Github link: https://github.com/psych094/499_covid_tracker

Fig 1.20: Brochure

5.7 Poster:

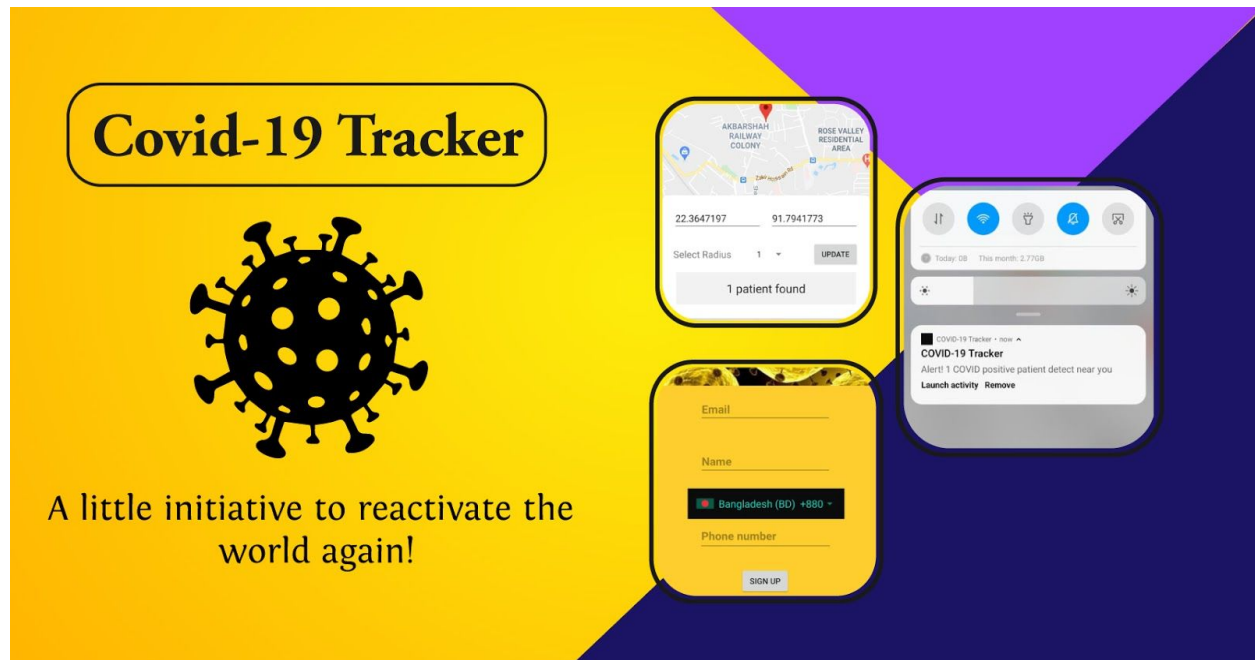


Fig 1.21 : Poster sample 1

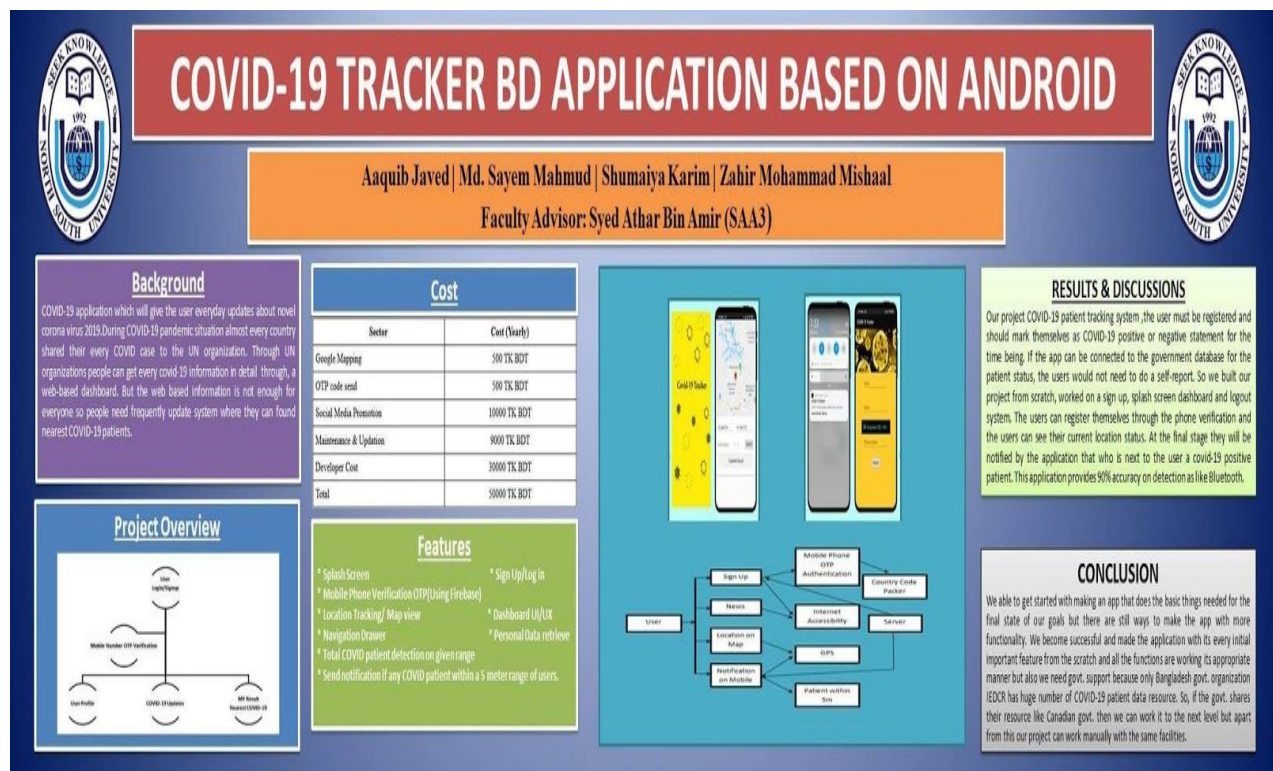


Fig 1.22: Capstone Demonstration(Poster)

5.8 IEEE format paper:

COVID-19 TRACKER BD APPLICATION BASED ON ANDROID

Aaquib Javed, Md. Sayem Mahmud, Shumaiya Karim, Zahir Mohammad Mishaal

Abstract—Background: Many countries are adopting non-therapeutic preventive measures, which include travel bans, remote work, complete country lock down, regular update of this pandemic, and most importantly, social distancing. However, these measures face challenges in Bangladesh as it has a dense population which has made mitigating the contraction of the COVID-19 virus a challenge. As it is a densely populated country it has made maintaining the preventive measures quite a challenge. At a local level mobile sanitization, temporary quarantine sites and healthcare facilities could help mitigate the impact of the pandemic, and health experts, along with international assistance can enable the country to minimize the impact of the pandemic. To minimize the impact and to spread more awareness in the country we need to take more steps for spreading the awareness, so in this modern era the manuscript authors taking a new steps that is android COVID-19 tracking system, it could be very helpful for the people and make them more aware of this pandemic situation and also this could be helpful for the government and UN to understand and get the exact data of COVID-19.

Result: Using GPS coordinates, it is possible to get information about the number of people infected with the COVID-19 virus in a certain area for individuals who are needing to commute during this pandemic. With being informed about this, users can be better suited with making better decisions.

Conclusion:

1. Introduction

We are currently living similarly to that of a post-apocalyptic state which we are calling “THE NEW NORMAL”. Here, everything, outside the comfort of our own homes is unpredictable and this, in turn, has instilled a fear within us especially when we are going out for necessities or if a relative comes over. To overcome this state of uncertainty, I would like to propose an app that lets you and all those using it to search for possible infected individuals based on people who have been infected on a certain day and where they have been commuting to in order to narrow down the populous for any other individuals who may be infected and to encourage them to have themselves checked.

This COVID-19 tracking system is an idea for people to be able to know if there are any infectees in an area. It is important to know about how many known infectees are in an area as the number of individuals in a given area who are not known to be infected would be proportional to the number of known infected.

The app named COVID-19 GPS Tracking System is installed in the phone and has a background service where the last GPS location and time is stored and sent to a central server. Any individual who is reported as COVID-19 positive is going to be known in real-time and also providing the number of positive individuals within a certain range from the user while they are on the move. This application is intended to be used with the information the government has

about the COVID-19 cases. As users of the app who are able to tell how many of the COVID-19 infected there are, they can decide on their own discretion on if they want to be in such an area.

2. Methods

Tools	Description of Tools
Java programming language	Used for the programming of the Android app.
Android Studio	The IDE for the Android application development.
XML	Used for User Interface design.
Firebase	Provisioning of a backend system for the Android application for easy development.
Firebase Authentication	To provide the backend interface for registration and login.
Firebase Database	Storage of application data.
Cloud Firestore	Flexible cloud database provided by Firebase for real-time GPS location update to the database.
Country Code Picker	Using the country code picker library to

	automatically default to the user's current location. Also used for automatic detection of SMS verification during registration.
Google Maps API	The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures.
Haversine Method	<p>Haversine formula.</p> <p>Distance d is a function of two latitude and longitude coordinates (fi1,y1) and (fi2,y2).</p> <p>Where r is the radius of the Earth.</p> $d(\varphi_1, \gamma_1, \varphi_2, \gamma_2) = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\varphi_2 - \varphi_1}{2} \right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2 \left(\frac{\gamma_2 - \gamma_1}{2} \right)} \right)$
GPS	Retrieving the coordinates of the users for determining how many users are in a location.

Fused location service	It's a Google API which works for receiving Android GPS data .
Fused location provider client	Object which provides the location data to the client.
Distancebetween() ()	Method computes the approximate distance in meters between two users. Used for determining the number of infectees from the user in a given radius.
Geocoder	Geocoding is the process of transforming a street address or other description of a location into a (longitude, latitude) coordinate.

3. Background

During COVID-19 pandemic situation almost every country shared their every COVID case to the UN organization. Through UN organizations people can get every covid-19 information in detail through a web-based dashboard. People can easily get the information by visiting the website dashboard but the dashboard only provides information based on how many new covid-19 cases found with how many people get cured and also the

death case. The dashboard also provides the visitors in what way they can identify covid-19 symptoms and possible safety rules. Till the middle of the 2020 everyone got updates regarding covid-19 based on WHO web-based dashboard but after the middle of 2020 many of the software farms and entrepreneurs started to build applications based on covid-19. Here some of the covid-19 applications details are given with their advantages and disadvantages which also help us and give our project group a great motivation to build an application based on this highly demanded topic [11].

The world health organization (WHO) web-based dashboard is the first source of information which provides how many people were affected with covid-19 virus during the initial period of 2019. This website provides people the symptoms regarding covid-19 virus and its initial possible remedies. But only the information can't make that much awareness among the people all over the world also with basic information of symptoms and its safety rules cause people failed to know who's got affected next to him/her. So, basically it was the primary step of awareness among the people in the beginning but is it enough to take the precaution and get other people in the safety zone? [11] NO, so there should be another option which will help man to man information who are affected and someone next to me who is affected by covid-19.

Health departments use contact tracing apps to find people who may have come into contact with someone with COVID-19. Apps help to capture data and watch the movement of people to make the process faster and more useful. Some could identify people who might have been exposed to the virus so they know to isolate themselves and watch for symptoms. Tech giants Apple and Google have teamed up on a platform that uses Bluetooth and a phone's operating system for contact tracing. Some people are wary of an app that tracks where they go and whom they meet. Instead of

storing data on a central server that may be vulnerable to hackers, Google and Apple say their apps won't be able to read the raw data themselves. Instead, the information will be available only to health agencies through what's called an application-programming interface (API). [12] But is this all sufficient? Cause the person who is roaming around the city, country or else is affected or not is not confirmed. So, there is also a huge amount of data and information working but in a different manner and we can use it to work in an appropriate manner where every person covid-19 case are stored and give them a unique number using a mobile application so that it will help people around the globe to move safely.

In this application it is only to help people to find covid-19 virus survival persons who can donate their plasma blood to the new covid-19 patient to get cured. But on the other side it won't help people to take themselves in a safe zone. This application is mainly built in Bangladesh by some new entrepreneurs and many similar applications are made regularly but no applications have such effective features to help the users effectively.

Now this is the application which is truly helping its users by its own features to know covid-19 and its every situation with helpful information along with the tracking the nearest covid-19 patient who were affected. Recently in 2021 Canada launched their application for their own regional people. This application is not publicly launched all over the world but only for their own country [13]. So, basically the world needs a common application where they can find each and every helpful information which is helping them perfectly.

4. Results

As our project is a COVID-19 patient tracking system, the user must be registered and should mark themselves as COVID-19 positive or negative statement for the time being. If the app can be connected to the government

database for the patient status, the users would not need to do a self-report. So we built our project from scratch, worked on a sign up, splash screen dashboard and logout system. The users can register themselves through the phone verification and the users can see their current location status.

5. Conclusion

We were able to make an app that does the basic tasks needed for the requirement of the project of our goal but there are still ways to make the app with more functionalities. Not too many functionalities were added to the app as to avoid bloating the application from main goal of having to display the user the number of COVID-19 positive patients to allow users to make better informed decisions on if they would like to continue to venture an area where a number of infectees are present. Further, with a higher number COVID-19 positive patients, the chances of there being unidentified COVID-19 infected would be exponentially higher.

References

- [1] C. in Flow, "ConstraintLayout Tutorial Part 1 - UNDERSTANDING CONSTRAINTS - Android Studio Tutorial," *YouTube*. [Online]. Available: <https://www.youtube.com/watch?v=4N4bCdYGcUc>. [Accessed: 28-Sep-2020].
- [2] C. W. Tea, "Android Navigation Drawer Menu Material Design | Android studio tutorial | Part 1," *YouTube*, 2020. [Online]. Available: <https://www.youtube.com/watch?v=HwYENW0RyY4>. [Accessed: 28-Sep-2020].
- [3] C. W. Tea, "Firebase phone authentication in android studio 2020 - City Guide app - Part 14," *YouTube*. [Online]. Available: <https://www.youtube.com/watch?v=lk4du-8giyQ>. [Accessed: 28-Sep-2020].

- [4] EasyLearn, “Create Login And Registration Screen In Android Using Firebase | App Development Tutorial,” *YouTube*, 2019. [Online]. Available: <https://www.youtube.com/watch?v=V0ZrnL-i77Q>. [Accessed: 28-Sep-2020].
- [5] hbb20, “hbb20/CountryCodePickerProject,” *GitHub*. [Online]. Available: <https://github.com/hbb20/CountryCodePickerProject/wiki/Auto-detect-country>. [Accessed: 01-Oct-2020].
- [6] M. Mansourzadeh, “Android Development for Beginners - Full Course,” *YouTube*, 2020. [Online]. Available: <https://www.youtube.com/watch?v=fis26HvvDII>. [Accessed: 28-Sep-2020].
- [7] S. Coding, “#1 Firebase Phone Authentication Android Tutorial - Sign In,” *YouTube*, 2018. [Online]. Available: <https://www.youtube.com/watch?v=N6HBStmDkMQ>. [Accessed: 28-Sep-2020].
- [8] T. C. City, “Android BroadcastReceiver Tutorial with Example,” *YouTube*, 2020. [Online]. Available: <https://www.youtube.com/watch?v=YsLLd5DaCCk>. [Accessed: 28-Sep-2020].
- [9] T. C. City, “GPS Location Tracker in Android - Full Tutorial 2020,” *YouTube*, 2020. [Online]. Available: <https://www.youtube.com/watch?v=ycBVe3iYtqQ>. [Accessed: 28-Sep-2020].
- [10] TinCoder, “How to use Country Code Picker Library in Android - [Android Libraries - #10],” *YouTube*. [Online]. Available: <https://www.youtube.com/watch?v=SOAehh31Zg>. [Accessed: 28-Sep-2020].
- [11] “Coronavirus disease (COVID-19) update,” *World Health Organization*, 12-Sep-2019. [Online]. Available: [https://www.who.int/bangladesh/emergencies/coronavirus-disease-\(covid-19\)-update](https://www.who.int/bangladesh/emergencies/coronavirus-disease-(covid-19)-update). [Accessed: 24-Oct-2020].
- [12] Nazario, Brunilda. “Mobile Apps for Coronavirus (COVID-19): See the List.” *WebMD*, WebMD, 17 Aug. 2020, www.webmd.com/lung/coronavirus-apps#1.
- [13] Canada, H., 2021. *Download COVID Alert: Canada’s Exposure Notification App*. [online] Canada.ca. Available at: https://www.canada.ca/en/public-health/services/diseases/coronavirus-disease-covid-19/covid-alert.html?utm_campaign=hc-sc-covidalertapp-20-21&utm_medium=sem&utm_source=ggl&utm_content=ad-text-en&utm_term=%2Bcovid%20%2Bapp%20%2Bcanada&adv=2021-0052&id_campaign=11822213983&id_source=115359935912&id_content=485640599456&utm_campaign=hc-sc-covidalertapp&utm_medium=sem&utm_source=ggl&utm_content=a24&id_campaign=11822213983&id_source=115359935912&id_content=485640599456 [Accessed 24 January

REFERENCES

APPENDIX A

1. C. in Flow, "ConstraintLayout Tutorial Part 1 - UNDERSTANDING CONSTRAINTS - Android Studio Tutorial," *YouTube*. [Online]. Available: <https://www.youtube.com/watch?v=4N4bCdyGcUc>. [Accessed: 28-Sep-2020].
2. C. W. Tea, "Android Navigation Drawer Menu Material Design | Android studio tutorial | Part 1," *YouTube*, 2020. [Online]. Available: <https://www.youtube.com/watch?v=HwYENW0RyY4>. [Accessed: 28-Sep-2020].
3. C. W. Tea, "Firebase phone authentication in android studio 2020 - City Guide app - Part 14," *YouTube*. [Online]. Available: <https://www.youtube.com/watch?v=lk4du-8giyQ>. [Accessed: 28-Sep-2020].
4. EasyLearn, "Create Login And Registration Screen In Android Using Firebase | App Development Tutorial," *YouTube*, 2019. [Online]. Available: <https://www.youtube.com/watch?v=V0ZrnL-i77Q>. [Accessed: 28-Sep-2020].
5. hbb20, "hbb20/CountryCodePickerProject," *GitHub*. [Online]. Available: <https://github.com/hbb20/CountryCodePickerProject/wiki/Auto-detect-country>. [Accessed: 01-Oct-2020].
6. M. Mansourzadeh, "Android Development for Beginners - Full Course," *YouTube*, 2020. [Online]. Available: <https://www.youtube.com/watch?v=fis26HvvDII>. [Accessed: 28-Sep-2020].
7. S. Coding, "#1 Firebase Phone Authentication Android Tutorial - Sign In," *YouTube*, 2018. [Online]. Available: <https://www.youtube.com/watch?v=N6HBStmDkMQ>. [Accessed: 28-Sep-2020].
8. T. C. City, "Android BroadcastReceiver Tutorial with Example," *YouTube*, 2020. [Online]. Available: <https://www.youtube.com/watch?v=YsLLd5DaCCk>. [Accessed: 28-Sep-2020].
9. T. C. City, "GPS Location Tracker in Android - Full Tutorial 2020," *YouTube*, 2020. [Online]. Available: <https://www.youtube.com/watch?v=ycBVe3iYtqQ>. [Accessed: 28-Sep-2020].
10. TinCoder, "How to use Country Code Picker Library in Android - [Android Libraries - #10]," *YouTube*. [Online]. Available: <https://www.youtube.com/watch?v=SODAehh31Zg>. [Accessed: 28-Sep-2020].

11. 11. “Coronavirus disease (COVID-19) update,” *World Health Organization*, 12-Sep-2019.
[Online]. Available:
[https://www.who.int/bangladesh/emergencies/coronavirus-disease-\(covid-19\)-update](https://www.who.int/bangladesh/emergencies/coronavirus-disease-(covid-19)-update).
[Accessed: 24-Oct-2020].
12. Nazario, Brunilda. “Mobile Apps for Coronavirus (COVID-19): See the List.”
WebMD, WebMD, 17 Aug. 2020, www.webmd.com/lung/coronavirus-apps#1.
13. Canada, H., 2021. *Download COVID Alert: Canada’S Exposure Notification App*.
[online] Canada.ca. Available at:
<https://www.canada.ca/en/public-health/services/diseases/coronavirus-disease-covid-19/covid-alert.html?utm_campaign=hc-sc-covidalertapp-20-21&utm_medium=sem&utm_source=ggl&utm_content=ad-text-en&utm_term=%2Bcovid%20%2Bapp%20%2Bcanada&adv=2021-0052&id_campaign=11822213983&id_source=115359935912&id_content=485640599456&&utm_campaign=hc-sc-covidalertapp&utm_medium=sem&utm_source=ggl&utm_content=ad-text-en&utm_term=%2Bcovid%20%2Bapp%20%2Bcanada&adv=2021-0024&id_campaign=11822213983&id_source=115359935912&id_content=485640599456> [Accessed 24 January 2021].
14. Creating a method using Haversine Formula, A., Helme, S., Vekariya, D. and alves, J., 2021. *Creating A Method Using Haversine Formula, Android V2*.
[online] Stack Overflow. Available at:
<<https://stackoverflow.com/questions/17787235/creating-a-method-using-haversine-formula-android-v2>> [Accessed 25 January 2021].
15. Medium. 2021. *How To Implement OTP-Mobile Verification Using Firebase Authentication*. [online] Available at:
<<https://abdullahaimen.medium.com/how-to-implement-otp-mobile-verification-using-firebase-authentication-6de615cc469>> [Accessed 25 January 2021].
16. 2021. [online] Available at:
<<https://developer.android.com/guide/topics/connectivity/bluetooth>> [Accessed 25 January 2021].

17. Firebase. 2021. *Add Firebase To Your Android Project*. [online] Available at:
<https://firebase.google.com/docs/android/setup?gclid=CjwKCAiA9bmABhBbEiwASb35V1hff6XpvTkNd8aPN-QhGnYzX2n5lrYvF0eaQk1lQiKMrquV1f_dGBocFIkQAvD_BwE> [Accessed 25 January 2021].
18. background?, H., 2021. *How To Make An Android App To Always Run In Background?*. [online] Stack Overflow. Available at:
<<https://stackoverflow.com/questions/34573109/how-to-make-an-android-app-to-always-run-in-background/34573169>> [Accessed 25 January 2021].
19. 2021. [online] Available at: <<https://developer.android.com/guide/topics/connectivity/bluetooth>> [Accessed 25 January 2021].

Minify your codebase and include it here. Courier New font without spaces between paragraphs with 9 font size should be used to save pages and utilize space. Reference external libraries that you have used in your project clearly in this section as well.

Project gradle file:

```
// Top-level build file where you can add configuration options common to all
sub-projects/modules.

buildscript {

    repositories {

        google()

        jcenter()

    }

    dependencies {

        classpath 'com.android.tools.build:gradle:4.0.1'

        classpath 'com.google.gms:google-services:4.3.3'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files

    }

}

allprojects {

    repositories {

        google()

        jcenter()

    }

}

task clean(type: Delete) {

    delete rootProject.buildDir

}
```

App gradle file:

apply plugin: 'com.android.application'

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

android {

 compileSdkVersion 28

 buildToolsVersion "30.0.2"

 defaultConfig {

 applicationId "com.example.sayem"

 minSdkVersion 23

 targetSdkVersion 28

 versionCode 1

 versionName "1.0"

 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

 }

 buildTypes {

 release {

 minifyEnabled false

 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
 'proguard-rules.pro'

```

    }
}

compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
}

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.google.firebase:firebase-database'
    implementation 'com.google.firebase:firebase-auth'
    implementation 'com.google.firebase:firebase-storage'
    implementation 'com.google.firebase:firebase-firestore'
    implementation 'com.google.firebase:firebase-analytics'
    implementation platform('com.google.firebase:firebase-bom:25.9.0')
    implementation 'com.hbb20:ccp:2.4.0'
    implementation 'com.google.android.gms:play-services-location:17.1.0'
    implementation 'com.google.android.gms:play-services-basement:17.5.0'

    //OTP PIN VIEW DESIGN
    implementation 'com.chaos.view:pinview:1.4.3'

```

```
implementation 'com.google.android.material:material:1.2.1'

implementation 'com.google.android.gms:play-services-maps:17.0.0'

testImplementation 'junit:junit:4.13.1'

androidTestImplementation 'androidx.test.ext:junit:1.1.2'

androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

}
```

AndroidManifest.xml :

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.example.sayem">

    <!--

        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use

        Google Maps Android API v2, but you must specify either coarse or fine

        location permissions for the "MyLocation" functionality.

    -->

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <uses-permission android:name="android.permission.INTERNET" />

    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
```



```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
```

```
<application

    android:name=".App"

    android:allowBackup="true"

    android:icon="@mipmap/ic_launcher"

    android:label="@string/app_name"

    android:roundIcon="@mipmap/ic_launcher_round"

    android:supportRtl="true"

    android:theme="@style/AppTheme">
```

```
<!--
```

The API key for Google Maps-based APIs is defined as a string resource.

(See the file "res/values/google_maps_api.xml").

Note that the API key is linked to the encryption key used to sign the APK.

You need a different API key for each encryption key, including the release key that is used to

sign the APK for publishing.

You can define the keys for the debug and release targets in src/debug/ and src/release/.

```
-->
```

```
<meta-data
```

android:name="com.google.android.geo.API_KEY"

android:value="@string/google_maps_key" />

<service

android:name=".LocationUpdatesService"

android:enabled="true"

android:exported="true" />

<activity

android:name=".MapsActivity"

android:label="@string/title_activity_maps"/>

<activity android:name=".AboutApp" />

<activity android:name=".AboutMe" />

<activity android:name=".UserDashboard" />

<activity android:name=".VerifyPhoneNo" />

<activity

android:name=".MainActivity"

android:theme="@style/Theme.AppCompat.Light.NoActionBar">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name=".SignUpActivity" />

</application>

</manifest>

MainActivity.java:

```
package com.example.sayem;
```

```
import androidx.annotation.NonNull;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Context;
```

```
import android.content.Intent;
```

```
import android.content.SharedPreferences;
```

```
import android.os.Bundle;
```

```
import android.os.Handler;
```

```
import android.util.Log;
```

```
import android.view.WindowManager;
```

```
import android.view.animation.Animation;
```

```
import android.view.animation.AnimationUtils;
```

```
import android.widget.ImageView;
```

```
import android.widget.TextView;
```

```
import com.google.android.gms.tasks.OnFailureListener;
```

```
import com.google.android.gms.tasks.OnSuccessListener;
```

```
import com.google.firebase.auth.FirebaseAuth;
```

```
import com.google.firebase.firestore.FirebaseFirestore;
```

```
import com.google.firebase.firestore.auth.User;
```

```

import java.util.HashMap;
import java.util.Map;

public class MainActivity extends AppCompatActivity {

    private static int SPLASH_SCREEN = 5000;
    Animation topAnim, bottomAnim;
    ImageView Image;
    TextView slogan;
    private static final String TAG = "APP_TAG";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.activity_main);

        topAnim = AnimationUtils.loadAnimation(this, R.anim.top_animation);
        bottomAnim = AnimationUtils.loadAnimation(this, R.anim.bottom_animation);

        Image = findViewById(R.id.imageView);
        slogan = findViewById(R.id.b);

        Image.setAnimation(topAnim);
        slogan.setAnimation(bottomAnim);

        /* Shared Preference */
        SharedPreferences sharedPref = getSharedPreferences("USER_PREFERENCE",
Context.MODE_PRIVATE);

        String phoneNo = sharedPref.getString("LOGGED_IN_USER_PHONE", null);
        if (phoneNo != null){
            Log.d(TAG, "##### Phone: " + phoneNo);
            App.getInstance().setLoggedInUserPhone(phoneNo);
            Intent intent= new Intent(getApplicationContext(),UserDashboard.class);

```

```

        startActivity(intent);
    }else {
        Log.d(TAG, "##### Value not found. Phone is null #####");
        new Handler().postDelayed(new Runnable() {
            public void run() {
                Intent intent = new Intent(MainActivity.this,
SignUpActivity.class);
                startActivity(intent);
                finish();
            }

        }, SPLASH_SCREEN);
    }
}
}

```

activity_main.xml:

```

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
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"
    tools:context="com.example.MainActivity"
    android:background="#FFE500"
    tools:ignore="LabelFor">

    <ImageView

```

```
    android:id="@+id/imageView"
    android:layout_width="414dp"
    android:layout_height="684dp"
    android:contentDescription="@string/todo"
    android:src="@drawable/download1"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.364" />
```

```
<TextView
```

```
    android:id="@+id/b"
    android:layout_width="193dp"
    android:layout_height="49dp"
    android:text="@string/covid_tracker"
    android:textColor="@color/colorPrimary"
    android:textSize="30sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/imageView"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.889" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

SignUpActivity.java:

```
package com.example.sayem;
```

```

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;
import com.hbb20.CountryCodePicker;

public class SignUpActivity extends AppCompatActivity {
    EditText emailEditText, nameEditText, mobileEditText;
    Button btnSignUp;
    CountryCodePicker countryCodePicker;
    FirebaseAuth mFirebaseAuth;

    @SuppressWarnings("WrongViewCast")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);
        mFirebaseAuth= FirebaseAuth.getInstance();
        emailEditText=findViewById(R.id.editTextTextEmailAddress);
        nameEditText=findViewById(R.id.editTextTextPassword);
        btnSignUp=findViewById(R.id.button);
        countryCodePicker=findViewById(R.id.country_code_picker);
        mobileEditText=findViewById(R.id.editTextPhone);
    }
}

```

```

btnSignUp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String email = emailEditText.getText().toString();
        String name = nameEditText.getText().toString();
        String mob = mobileEditText.getText().toString();
        String _phoneNo="+" + countryCodePicker.getSelectedCountryCode()+mob;
        String emailPattern= "[a-zA-Z0-9._-]+@[a-z]+\.[a-z]+";

        Context context = SignUpActivity.this;

        if (email.isEmpty() && name.isEmpty()){
            Toast.makeText(SignUpActivity.this,"Both fields are empty",
Toast.LENGTH_SHORT ).show();
        }
        else if (email.isEmpty()) {
            Toast.makeText(SignUpActivity.this," Email are empty",
Toast.LENGTH_SHORT ).show();
        }
        else if(!email.matches(emailPattern)){
            Toast.makeText(SignUpActivity.this," Invalid Email address",
Toast.LENGTH_SHORT ).show();
        }

        else if (name.isEmpty()) {
            Toast.makeText(SignUpActivity.this," nameEditText are empty",
Toast.LENGTH_SHORT ).show();
        }
        else if (name.length()<6) {
            Toast.makeText(SignUpActivity.this," nameEditText length at least
6 needed! ", Toast.LENGTH_SHORT ).show();
        }
        else if(mob.isEmpty()){
            Toast.makeText(SignUpActivity.this,"Mobile number is empty",
Toast.LENGTH_SHORT ).show();
        }

        else if (!email.isEmpty() && (!name.isEmpty()) &&(!mob.isEmpty()) ){

```



```

        Log.d("VerifyPhone"," phone , email , passworid " + email + name
+ mob);

        Intent intent=new Intent(context, VerifyPhoneNo.class);
        intent.putExtra("phone", _phoneNo);
        intent.putExtra("email",email );
        intent.putExtra("name", name );
        startActivity(intent);
    }
    else{
        Toast.makeText(SignUpActivity.this," Error Occurred",
Toast.LENGTH_SHORT ).show();
    }
}

}

);

}

}

```

Activity_SignUp.xml:

```

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
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:id="@+id/_conte"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#fece2f"
    tools:context=".SignUpActivity"
    tools:ignore="LabelFor">

```

```
<ImageView
    android:layout_width="438dp"
    android:layout_height="166dp"
    android:contentDescription="@string/todo"
    android:src="@drawable/download"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<EditText
    android:id="@+id/editTextTextEmailAddress"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="184dp"
    android:autofillHints=""
    android:ems="10"
    android:hint="@string/email"
    android:inputType="textEmailAddress"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<EditText
    android:id="@+id/editTextTextPassword"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="@string/name"
    android:textSize="20sp"
```

```
android:textStyle="bold"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.497"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editTextEmailAddress"
app:layout_constraintVertical_bias="0.102"
tools:ignore="Autofill" />
```

<TextView

```
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="32dp"
android:text=""
android:textSize="20sp"
android:textStyle="bold"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/button" />
```

<com.hbb20.CountryCodePicker

```
android:id="@+id/country_code_picker"
android:layout_width="277dp"
android:layout_height="54dp"
android:layout_marginTop="24dp"
android:background="@color/colorPrimary"
android:padding="7dp"
app:ccp_contentColor="@color/colorAccent"
app:ccp_showFlag="true"
app:ccp_defaultNameCode="BD"
app:ccp_autoDetectCountry="true"
app:ccp_showFullName="true"
app:ccp_showNameCode="true"
```

```
app:layout_constraintBottom_toTopOf="@+id/editTextPhone"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.508"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editTextTextPassword"
app:layout_constraintVertical_bias="0.0"
tools:ignore="MissingConstraints" />
```

<Button

```
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="164dp"
android:text="@string/sign_up"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.498"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editTextTextPassword" />
```

<EditText

```
android:id="@+id/editTextPhone"
android:layout_width="235dp"
android:layout_height="51dp"
android:autofillHints=""
android:ems="10"
android:hint="@string/phone_number"
android:inputType="phone"
android:textSize="20sp"
android:textStyle="bold"
app:layout_constraintBottom_toTopOf="@+id/button"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editTextTextPassword"
app:layout_constraintVertical_bias="0.734" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

UserDashboard.java:

```
package com.example.sayem;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SwitchCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.navigation.NavigationView;
import com.google.firebase.auth.FirebaseAuth;
```

```

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;

import java.util.HashMap;
import java.util.Map;

public class UserDashboard extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener{

    static final float END_SCALE =0.7f;
    // Button btnLogOut;

    DrawerLayout drawerLayout;
    NavigationView navigationView;
    ImageView menuIcon;
    ImageView News;
    ImageView Hospital;
    ImageView Affected;
    ImageView Location;

    private static final String TAG = "UserDashboard";

    LinearLayout contentView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.activity_user_dashboard);

        // btnLogOut= findViewById(R.id.button2);

        drawerLayout=findViewById(R.id.drawer_layout);

        navigationView=findViewById(R.id.navigation_view);

```

```

        menuIcon=findViewById(R.id.menu_icon);
        contentView=findViewById(R.id.content);
        News=findViewById(R.id.news);
        Hospital=findViewById(R.id.hospital);
        Affected=findViewById(R.id.affected);
        Location=findViewById(R.id.location);
        navigationDrawer();

        Location.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intToMap= new Intent (UserDashboard.this, MapsActivity.class);
                startActivity(intToMap);
            }
        });

        News.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                gotoUrl("https://www.who.int/bangladesh/emergencies/coronavirus-disease-(covid-19)-update");
            }
        });

        Hospital.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                gotoUrl("https://tbsnews.net/coronavirus-chronicle/covid-19-bangladesh/names-labs-contact-numbers-coronavirus-tests-67078");
            }
        });

        Affected.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                gotoUrl("https://www.worldometers.info/coronavirus/");
            }
        });

```

```

        }

    });

    View actionView =
navigationView.getMenu().findItem(R.id.nav_covid_status).getActionView();

    final SwitchCompat drawerSwitch =
actionView.findViewById(R.id.sw_covid_status);

    String phone = FirestoreUtil.getInstance().getCurrentUser().getPhoneNumber();

    if (phone != null && !phone.isEmpty()){

        final DocumentReference docRef = FirestoreUtil.getInstance()
            .getDocumentRef().collection("users").document(phone);
        docRef.addSnapshotListener(new EventListener<DocumentSnapshot>() {

            @Override
            public void onEvent(@Nullable DocumentSnapshot snapshot,
                                @Nullable FirebaseFirestoreException e) {

                if (e != null) {

                    Log.w(TAG, "Listen failed.", e);
                    return;
                }

                if (snapshot != null && snapshot.exists()) {

                    Map<String, Object> map = snapshot.getData();
                    if (map != null){

                        Boolean status =(Boolean) map.get("status");
                        if (status != null)
                            drawerSwitch.setChecked(status);

                    }
                } else {

                    Log.d(TAG, "Current data: null");
                }

            }

        });

    }
}

```



```

        drawerSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {

    @Override

    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{

        sendConditionStatus(isChecked);

    }

});

}

private void sendConditionStatus(boolean status){

    //check if user is logged in
    if (FirestoreUtil.getInstance().getCurrentUser() == null){
        return;
    }

    // user sends one's current condition of COVID situation to the db
    String phone = FirestoreUtil.getInstance().getCurrentUser().getPhoneNumber();
    if (phone != null && !phone.isEmpty()){
        FirestoreUtil.getInstance().getDocumentRef().collection("users")
            .document(phone).update("status", status)
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    Log.d(TAG, "onSuccess: ");
                }
            })
            .addOnFailureListener(new OnFailureListener(){
                @Override
                public void onFailure(@NonNull Exception e) {
                    Log.d(TAG, "onFailure: " + e.getMessage());
                }
            });
    }
}

```

```

    }
}

private void gotoUrl(String s) {
    Uri uri = Uri.parse(s);
    startActivity(new Intent(Intent.ACTION_VIEW,uri));
}

//Navigation Drawer Function
private void navigationDrawer() {
    navigationView.bringToFront();
    navigationView.setNavigationItemSelectedListener(this);
    navigationView.setCheckedItem(R.id.nav_home);
    menuIcon.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (drawerLayout.isDrawerVisible(GravityCompat.START))
                drawerLayout.closeDrawer(GravityCompat.START);
            else drawerLayout.openDrawer(GravityCompat.START);
        }
    });
    animateNavigationDrawer();
}

private void animateNavigationDrawer() {
    //Add any color or remove it to use the default one!
    //To make it transparent use Color.Transparent in side setScrimColor();
    //drawerLayout.setScrimColor(Color.TRANSPARENT);
    drawerLayout.setScrimColor(getResources().getColor(R.color.card4));
    drawerLayout.addDrawerListener(new DrawerLayout.SimpleDrawerListener() {
        @Override
        public void onDrawerSlide(View drawerView, float slideOffset) {

            // Scale the View based on current slide offset

```

```

        final float diffScaledOffset = slideOffset * (1 - END_SCALE);
        final float offsetScale = 1 - diffScaledOffset;
        contentView.setScaleX(offsetScale);
        contentView.setScaleY(offsetScale);

        // Translate the View, accounting for the scaled width
        final float xOffset = drawerView.getWidth() * slideOffset;
        final float xOffsetDiff = contentView.getWidth() * diffScaledOffset /
2;

        final float xTranslation = xOffset - xOffsetDiff;
        contentView.setTranslationX(xTranslation);
    }
}

);

}

@Override
public void onBackPressed() {
    if (drawerLayout.isDrawerVisible(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START);
    } else super.onBackPressed();
}

@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case R.id.nav_home:
            break;
        case R.id.nav_logout:
            FirebaseAuth.getInstance().signOut();
            App.getInstance().logout();
            Intent intToMain = new Intent (UserDashboard.this,
SignUpActivity.class);
            intToMain.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);

```

```

        startActivity(intToMain);

        break;

    case R.id.nav_about_app:

        Intent i = new Intent(UserDashboard.this, AboutApp.class);

        startActivity(i);

        break;

    case R.id.nav_about_me:

        Intent e = new Intent(UserDashboard.this, AboutMe.class);

        startActivity(e);

        break;

    }

    drawerLayout.closeDrawer(GravityCompat.START);

    return true;

}
}

```

activity_userdashboard.xml

```

<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="#EEF7ED"
    tools:context=".UserDashboard"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:id="@+id/drawer_layout">

```

```

<com.google.android.material.navigation.NavigationView

```

```
android:id="@+id/navigation_view"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:background="#EEF7ED"  
android:theme="@style/navigationTheme"  
android:layout_gravity="start"  
app:headerLayout="@layout/menu_header"  
app:menu="@menu/main_menu"/>
```

<LinearLayout

```
android:id="@+id/content"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:background="#EEF7ED"  
android:orientation="vertical">
```

<RelativeLayout

```
android:layout_width="match_parent"  
android:layout_height="76dp"  
android:background="#EEF7ED"  
android:padding="20dp"  
tools:ignore="MissingConstraints">
```

<ImageView

```
android:id="@+id/menu_icon"  
android:layout_width="49dp"  
android:layout_height="38dp"
```

```
        android:layout_centerVertical="true"
        android:src="@drawable/menu"
        android:contentDescription="@string/todo4" />
```

```
</RelativeLayout>
```

```
<RelativeLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
```

```
<TextView
    android:id="@+id/app_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="@string/get_a_way"
    android:textColor="#090901"
    android:textSize="28sp" />
```

```
</RelativeLayout>
```

```
<RelativeLayout
    android:background="#FFFFFF"
    android:elevation="80dp"
    android:layout_height="40dp"
    android:layout_marginBottom="20dp"
    android:layout_marginLeft="40dp"
    android:layout_marginRight="40dp"
    android:layout_marginTop="20dp"
```

```
android:layout_width="match_parent" tools:targetApi="lollipop">
```

```
<TextView
```

```
    android:layout_centerVertical="true"  
    android:layout_height="wrap_content"  
    android:layout_width="wrap_content"  
    android:text="@string/search"  
    android:layout_marginStart="15dp"  
    tools:ignore="RelativeOverlap" />
```

```
<ImageView
```

```
    android:layout_centerVertical="true"  
    android:layout_height="30dp"  
    android:layout_width="30dp"  
    android:src="@drawable/search"  
    android:layout_marginEnd="15dp"  
    android:layout_alignParentEnd="true"  
    android:contentDescription="@string/todo" />
```

```
</RelativeLayout>
```

```
<LinearLayout
```

```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center">
```

```
<LinearLayout
```

```
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:orientation="vertical">
```

```
<RelativeLayout
```

```
    android:layout_width="60dp"
    android:layout_height="30dp"
    android:layout_margin="5dp"
    android:background="@drawable/card_1"
    android:elevation="8dp" tools:targetApi="lollipop">
```

```
<ImageView
```

```
    android:id="@+id/hospital"
    android:layout_width="33dp"
    android:layout_height="33dp"
    android:layout_centerInParent="true"
    android:src="@drawable/hospital"
    android:contentDescription="@string/todo" />
```

```
</RelativeLayout>
```

```
<TextView
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FFFFFF"
    android:gravity="center_horizontal"
    android:text="@string/hospital"
    android:textStyle="bold" />
```

```
</LinearLayout>
```


<LinearLayout

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical">

<RelativeLayout

android:layout_width="60dp"
android:layout_height="30dp"
android:layout_margin="5dp"
android:background="@drawable/card_2"
android:elevation="8dp">

<ImageView

android:id="@+id/location"
android:layout_width="33dp"
android:layout_height="33dp"
android:layout_centerInParent="true"
android:src="@drawable/location"
android:contentDescription="@string/todo" />

</RelativeLayout>

<TextView

android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/location"
android:textStyle="bold"

```
        android:background="#FFFFFF"
        android:gravity="center_horizontal" />
</LinearLayout>
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RelativeLayout
        android:layout_width="60dp"
        android:layout_height="30dp"
        android:layout_margin="5dp"
        android:background="@drawable/card_3"
        android:elevation="8dp" tools:targetApi="lollipop">

        <ImageView
            android:id="@+id/news"
            android:layout_width="33dp"
            android:layout_height="33dp"
            android:layout_centerInParent="true"
            android:src="@drawable/news"
            android:contentDescription="@string/todo" />

    </RelativeLayout>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:text="@string/news"
        android:textStyle="bold"
        android:background="#FFFFFF"
        android:gravity="center_horizontal" />
</LinearLayout>
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RelativeLayout
        android:layout_width="60dp"
        android:layout_height="30dp"
        android:layout_margin="5dp"
        android:background="@drawable/card_4"
        android:elevation="8dp">

        <ImageView
            android:id="@+id/affected"
            android:layout_width="33dp"
            android:layout_height="33dp"
            android:layout_centerInParent="true"
            android:src="@drawable/affected"
            android:contentDescription="@string/todo1" />

    </RelativeLayout>

    <TextView
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/affected"
        android:textStyle="bold"
        android:background="#FFFFFF"
        android:gravity="center_horizontal" />
</LinearLayout>
```

```
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="30dp"
    android:layout_marginRight="30dp"
    android:background="@color/card3">
```

```
<LinearLayout
    android:layout_width="155dp"
    android:layout_height="278dp"
    android:layout_margin="10dp"
    android:background="@color/card1"
    android:orientation="vertical">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
android:layout_marginTop="50dp"
android:text="@string/disaster_of_covid_19"
android:textColor="#040000"
android:textSize="25sp"
android:layout_marginStart="10dp" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:textSize="20sp"
    android:text="@string/to_get_overview_of_covid" />
```

```
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="137dp"
    android:layout_height="280dp"
    android:layout_margin="10dp"
    android:background="@color/card1"
    android:orientation="vertical">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="50dp"
    android:text="@string/covid_vaccine_update"
```

```
        android:textColor="#040000"
```

```
        android:textSize="25sp" />
```

```
    <TextView
```

```
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
```

```
        android:layout_margin="10dp"
```

```
        android:layout_marginTop="200dp"
```

```
        android:textSize="20sp"
```

```
        android:text="@string/to_get_overview_of_covid" />
```

```
    </LinearLayout>
```

```
</LinearLayout>
```

```
</LinearLayout>
```

```
</androidx.drawerlayout.widget.DrawerLayout>
```

VerifyPhoneNo.java

```
package com.example.sayem;
```

```
import android.Manifest;
```

```
import android.content.Context;
```

```
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
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.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthProvider;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.TimeUnit;

import static com.google.firebase.auth.PhoneAuthProvider.ForceResendingToken;
```

```

import static
com.google.firebase.auth.PhoneAuthProvider.OnVerificationStateChangedCallbacks;

import static com.google.firebase.auth.PhoneAuthProvider.getCredential;

public class VerifyPhoneNo extends AppCompatActivity {

    public static final String TAG = "APP_TAG";

    String verificationCodeBySystem;

    Button verify_btn;

    EditText phoneNoEntryByTheUser;

    ProgressBar progressBar;

    private String mEmail;

    private String mName;

    private String mPhone;

    private double mLat;

    private double mLon;

    private FusedLocationProviderClient fusedLocationClient;

    private String APP_TAG = "VFN";

    Map<String, Object> user = new HashMap<>();

    FirebaseFirestore db = FirebaseFirestore.getInstance(); // new FirebaseFirestore()

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_verify_phone_no);

        verify_btn = findViewById(R.id.verify_btn);

        progressBar = findViewById(R.id.progress_bar);

        progressBar.setVisibility(View.GONE);

        phoneNoEntryByTheUser = findViewById(R.id.verification_code_entered_by_user);

        Log.d(APP_TAG, " startActivity has no extra value ");

        Intent startActivity = getIntent();

        if (startIntent.hasExtra("phone")) {

            mEmail = startActivity.getStringExtra("email");

```



```

        Log.d("VerifyPhone", "email number is " + mEmail);

        mName = startIntent.getStringExtra("name");

        Log.d("VerifyPhone", "Password is " + mName);

        mPhone = startIntent.getStringExtra("phone");

        Log.d("VerifyPhone", "Phone number is " + mPhone);
    } else {
        Log.d("VerifyPhone", "startIntent has no extra value ");
    }

    if(mPhone != null){
        user.put("name", mName);
        user.put("email", mEmail);
        user.put("phone", mPhone);
        sendVerificationCodeToUser(mPhone);
    } else{
        Log.d("VerifyPhone", "Phone number is null " + mPhone);
    }

    verify_btn.setOnClickListener(new View.OnClickListener(){
        public void onClick(View view){
            String code= phoneNoEntryByTheUser.getText().toString();
            if (code.isEmpty()|| code.length()<6){
                phoneNoEntryByTheUser.setError("Wrong OTP...");
                phoneNoEntryByTheUser.requestFocus();
                return;
            }
            progressBar.setVisibility(View.VISIBLE);
            verifyCode(code);
        }
    }

```

```

});

    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION)
!= PackageManager.PERMISSION_GRANTED) {

        // TODO: Consider calling
        //    ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //    public void onRequestPermissionsResult(int requestCode, String[]
permissions,
        //                                     int[] grantResults)
        // to handle the case where the user grants the permission. See the
documentation
        // for ActivityCompat#requestPermissions for more details.
        return;
    }

    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);
    fusedLocationClient.getLastLocation()

        .addOnSuccessListener(this, new OnSuccessListener<Location>() {

            @Override
            public void onSuccess(Location location) {

                // Got last known location. In some rare situations this can
be null.

                if (location != null) {

                    Log.d(APP_TAG, String.format("%f",
location.getLatitude()));

                    Log.d(APP_TAG, String.format("%f",
location.getLongitude()));

                    mLat = location.getLatitude();
                    mLon = location.getLongitude();

                }

            }

        });
}

```

```

private void sendVerificationCodeToUser(String _phoneNo) {
    Log.i("TAG", "Phone number: " + _phoneNo);
    PhoneAuthProvider.getInstance().verifyPhoneNumber(
        _phoneNo,          // Phone number to verify
        60,                 // Timeout duration
        TimeUnit.SECONDS,   // Unit of timeout
        this,              // Activity (for callback binding)
        mCallbacks);       // OnVerificationStateChangedCallbacks

    //App.getInstance().setLoggedInUserPhone(mPhone); ///passing at app.java
    setLoggedIn UserPhone

    //Log.d(TAG, "Writing to shared preference: " + mPhone);

    //SharedPreferences sharedPref = getSharedPreferences("USER_PREFERENCE",
    Context.MODE_PRIVATE);

    //SharedPreferences.Editor editor = sharedPref.edit(); //creating object of
    shared pref editor

    //editor.putString("LOGGED_IN_USER_PHONE", mPhone);
    // editor.apply();

    // Intent intent= new Intent(getApplicationContext(),UserDashboard.class);
    // startActivity(intent);
}

private OnVerificationStateChangedCallbacks mCallbacks = new
OnVerificationStateChangedCallbacks() {

    @Override

    public void onCodeSent(@NonNull String s, @NonNull ForceResendingToken
forceResendingToken) {

        super.onCodeSent(s, forceResendingToken);

        verificationCodeBySystem= s;

    }

    @Override

    public void onVerificationCompleted(@NonNull PhoneAuthCredential
phoneAuthCredential) {

```

```

String code= phoneAuthCredential.getSmsCode();

if(code!=null) {

    Log.d(TAG, "##### onVerificationCompleted #####");

    progressBar.setVisibility(View.VISIBLE);

    verifyCode(code);

}

}

@Override

public void onVerificationFailed(@NonNull FirebaseException e) {

    Log.d(TAG, "##### onVerificationFailed #####");

    Toast.makeText(VerifyPhoneNo.this, e.getMessage(),
Toast.LENGTH_SHORT).show();

}

};

private void verifyCode(String code) {

    Log.d(TAG, "##### Verify Code #####");

    PhoneAuthCredential credential = getCredential(verificationCodeBySystem,
code);

    signInTheUserByCredentials(credential);

    App.getInstance().setLoggedInUserPhone(mPhone); ///passing at app.java
setLoggedIn UserPhone

    Log.d(TAG, "Writing to shared preference: " + mPhone);

    SharedPreferences sharedPref = getSharedPreferences("USER_PREFERENCE",
Context.MODE_PRIVATE);

    SharedPreferences.Editor editor = sharedPref.edit(); //creating object of
shared pref editor

    editor.putString("LOGGED_IN_USER_PHONE", mPhone);

    editor.apply();

    Intent intent= new Intent(getApplicationContext(),UserDashboard.class);

    startActivity(intent);

```

```

// firebase data savings
user.put("lat", mLat);
user.put("lon", mLon);
Log.d(TAG, user.toString());
db.collection("users").document(mPhone).set(user)
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            Log.d(TAG, "DocumentSnapshot successfully written!");
        }
    })
    .addOnFailureListener(new OnFailureListener(){
        @Override
        public void onFailure(@NonNull Exception e) {
            Log.w(TAG, "Error writing document", e);
        }
    });
}

private void signInTheUserByCredentials(PhoneAuthCredential credential){
    FirebaseAuth firebaseAuth=FirebaseAuth.getInstance();
    firebaseAuth.signInWithCredential(credential)
        .addOnCompleteListener(VerifyPhoneNo.this, new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()){
                Intent intent= new
Intent(getApplicationContext(),UserDashboard.class);
                startActivity(intent);
            }
            else{
                Toast.makeText(VerifyPhoneNo.this,
task.getException().getMessage(),Toast.LENGTH_SHORT).show();
            }
        }
    });
}

```

```

        }
    }
});
}

}

```

Activity_verify_phone_no.xml:

```

<?xml version="1.0" encoding="utf-8"?>

<ScrollView 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:background="#F7FF00"
    android:orientation="vertical"
    tools:context=".VerifyPhoneNo">

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

        <ImageView
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:layout_gravity="center"
            android:src="@drawable/download"
            android:contentDescription="@string/todo3" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"

            android:text="@string/verify_phone_no"
            android:gravity="center_horizontal"
            android:textSize="30sp" />
    
```

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"

    android:text="@string/enter_verification_code_if_not_automatically_authenticated"
    android:gravity="center_horizontal"
    android:textSize="14sp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:orientation="vertical"
    android:padding="50dp">

    <EditText
        android:id="@+id/verification_code_entered_by_user"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/enter_otp"
        android:autofillHints=""
        android:inputType="numberPassword" />

    <Button
        android:background="#fece2f"
        android:id="@+id/verify_btn"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_width="match_parent"
        android:text="@string/verify" />

    <ProgressBar
        android:id="@+id/progress_bar"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="20dp"
        android:layout_width="wrap_content" />
    </LinearLayout>
</LinearLayout>
```

```
</ScrollView>
```

AboutApp.java

```
package com.example.sayem;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class AboutApp extends AppCompatActivity {

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

activity_about_app.xml


```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
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:background="#84FFFF"
    android:layout_height="match_parent"
    tools:context=".AboutApp">
```

```
<ImageView
    android:id="@+id/imageView3"
    android:layout_width="288dp"
    android:layout_height="166dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.495"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.076"
    app:srcCompat="@drawable/download3"
    android:contentDescription="@string/about_app" />
```

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="326dp"
    android:layout_height="435dp"
    android:text="@string/about_app"
```

```
        android:textAlignment="center"
        android:textSize="23sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView3"
        app:layout_constraintVertical_bias="0.125" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

AboutMe.java

```
package com.example.sayem;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class AboutApp extends AppCompatActivity {

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

activity_about_me.xml

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
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:background="#84FFFF"
    android:layout_height="match_parent"
    tools:context=".AboutMe">
```

```
<TextView

    android:id="@+id/nameLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name"
    android:textSize="14sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:layout_marginTop="350dp"
    android:layout_marginLeft="175dp"/>
```

```
<TextView

    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/n_a"
    android:textSize="20sp"
```

```
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/nameLabel"
android:layout_marginTop="2dp"
android:layout_marginLeft="130dp"/>
```

<TextView

```
android:id="@+id/emailLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Email"
android:textSize="14sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/name"
android:layout_marginTop="16dp"
android:layout_marginLeft="175dp"/>
```

<TextView

```
android:id="@+id/email"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/n_a"
android:textSize="20sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/emailLabel"
android:layout_marginTop="2dp"
android:layout_marginLeft="80dp"/>
```

<TextView

```
android:id="@+id/phoneLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/phone"
android:textSize="14sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/email"
android:layout_marginTop="16dp"
android:layout_marginLeft="175dp"/>
```

<TextView

```
android:id="@+id/phone"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/n_a"
android:textSize="20sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/phoneLabel"
android:layout_marginTop="2dp"
android:layout_marginLeft="127dp"/>
```

<ImageView

```
android:id="@+id/imageView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
app:layout_constraintBottom_toTopOf="@+id/name"
app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:srcCompat="@drawable/human" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

App.java

```
package com.example.sayem;
```

```
import android.app.Application;
```

```
import android.content.Context;
```

```
import android.content.SharedPreferences;
```

```
/**
```

```
 * Created by Sayem Mahmud on 6/11/20.
```

```
 */
```

```
public class App extends Application {
```

```
    private String loggedInUserPhone;
```

```
    private static App instance;
```

```
    public static App getInstance() {
```

```
        return instance;
```

```
    }
```

```

@Override

public void onCreate() {
    super.onCreate();
    instance = this;
}

public String getLoggedInUserPhone(){
    return loggedInUserPhone;
}

public void setLoggedInUserPhone(String phone){
    this.loggedInUserPhone = phone;
}

public void logOut(){
    SharedPreferences sharedPref = getSharedPreferences("USER_PREFERENCE",
Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPref.edit();

    editor.putString("LOGGED_IN_USER_PHONE", null);
    editor.apply();
}
}

```

Constant.java

```
package com.example.sayem;
```

```
/**
```

* Created by Imran Khan on 11/24/2020.

* Email : context.imran@gmail.com

*/

```
public class Constant {  
    public static final Object MIN_DISTANCE_RANGE = 5;  
}
```

Firestoreutil.java

package com.example.sayem;

import android.util.Log;

import androidx.annotation.NonNull;

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

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

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.firestore.FirebaseFirestore;

import com.google.firebase.firestore.QuerySnapshot;

/**

* Created by Imran Khan on 11/24/2020.

* Email : context.imran@gmail.com

*/

```
public class FirestoreUtil {
```



```
private static FirestoreUtil instance;
private int count = 0;

private static final String TAG = "FirestoreUtil";

public static FirestoreUtil getInstance(){
    if (instance == null){
        instance = new FirestoreUtil();
    }
    return instance;
}

public FirebaseFirestore getDocumentRef(){
    return FirebaseFirestore.getInstance();
}

public FirebaseUser getCurrentUser(){
    return FirebaseAuth.getInstance().getCurrentUser();
}
}
```

LocationUpdateService.java

```
package com.example.sayem;

import android.app.ActivityManager;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
```

```
import android.app.PendingIntent;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.content.res.Configuration;
import android.graphics.BitmapFactory;
import android.location.Location;
import android.os.Binder;
import android.os.Build;
import android.os.Handler;
import android.os.HandlerThread;
import android.os.IBinder;
import android.os.Looper;
import android.util.Log;

import androidx.annotation.Nullable;
import androidx.core.app.NotificationCompat;
import androidx.localbroadcastmanager.content.LocalBroadcastManager;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.model.LatLng;
import com.google.firebase.firestore.DocumentSnapshot;

import java.util.List;
```

```
/**
```

```
* Created by Imran Khan on 11/24/2020.
```

```
* Email : context.imran@gmail.com
```

```
*/
```

```
public class LocationUpdatesService extends Service {
```

```
    private static final String TAG = LocationUpdatesService.class.getSimpleName();
```

```
    private static final String PACKAGE_NAME = BuildConfig.APPLICATION_ID;
```

```
    private static final String CHANNEL_ID = "channel.sayem";
```

```
    public static final String ACTION_BROADCAST = PACKAGE_NAME + ".broadcast";
```

```
    public static final String EXTRA_LOCATION = PACKAGE_NAME + ".location";
```

```
    private static final String EXTRA_STARTED_FROM_NOTIFICATION =  
    PACKAGE_NAME + ".started_from_notification";
```

```
    private final IBinder mBinder = new LocalBinder();
```

/**

* The desired interval for location updates. Inexact. Updates may be more or less frequent.

*/

private static final long UPDATE_INTERVAL_IN_MILLISECONDS = 10000;

/**

* The fastest rate for active location updates. Updates will never be more frequent

* than this value.

*/

private static final long FASTEST_UPDATE_INTERVAL_IN_MILLISECONDS =
UPDATE_INTERVAL_IN_MILLISECONDS / 2;

/**

* The identifier for the notification displayed for the foreground service.

*/

private static final int NOTIFICATION_ID = 12345678;

/**

* Used to check whether the bound activity has really gone away and not unbound as part of
an

* orientation change. We create a foreground service notification only if the former takes

* place.

*/

private boolean mChangingConfiguration = false;

/**

```
    * Contains parameters used by {@link  
com.google.android.gms.location.FusedLocationProviderApi}.
```

```
    */
```

```
    private LocationRequest mLocationRequest;
```

```
/**
```

```
    * Provides access to the Fused Location Provider API.
```

```
    */
```

```
    private FusedLocationProviderClient mFusedLocationClient;
```

```
/**
```

```
    * Callback for changes in location.
```

```
    */
```

```
    private LocationCallback mLocationCallback;
```

```
    private Handler mServiceHandler;
```

```
/**
```

```
    * The current location.
```

```
    */
```

```
    private Location mLocation;
```

```
    private NotificationManager mNotificationManager;
```

```
@Override
```

```
public void onCreate() {
```

```
    super.onCreate();
```

```

mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

mLocationCallback = new LocationCallback() {
    @Override
    public void onLocationResult(LocationResult locationResult) {
        super.onLocationResult(locationResult);
        onNewLocation(locationResult.getLastLocation());
    }
};

createLocationRequest();

getLastLocation();

HandlerThread handlerThread = new HandlerThread(TAG);
handlerThread.start();

mServiceHandler = new Handler(handlerThread.getLooper());

mNotificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);

// Android O requires a Notification Channel.
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    CharSequence name = getString(R.string.app_name);
    // Create the channel for the notification

    NotificationChannel mChannel = new NotificationChannel(CHANNEL_ID, name,
NotificationManager.IMPORTANCE_DEFAULT);

    // Set the Notification Channel for the Notification Manager.
    mNotificationManager.createNotificationChannel(mChannel);
}

```

```
}  
}
```

@Override

```
public int onStartCommand(Intent intent, int flags, int startId) {  
    boolean startedFromNotification =  
intent.getBooleanExtra(EXTRA_STARTED_FROM_NOTIFICATION, false);  
  
    // We got here because the user decided to remove location updates from the notification.  
    if (startedFromNotification) {  
        removeLocationUpdates();  
        stopSelf();  
    }  
    // Tells the system to not try to recreate the service after it has been killed.  
    return START_NOT_STICKY;  
}
```

@Override

```
public void onConfigurationChanged(Configuration newConfig) {  
    super.onConfigurationChanged(newConfig);  
    mChangingConfiguration = true;  
}
```

@Override

```
public IBinder onBind(Intent intent) {  
    // Called when a client (MainActivity in case of this sample) comes to the foreground  
    // and binds with this service. The service should cease to be a foreground service  
    // when that happens.  
    stopForeground(true);  
}
```

```
mChangingConfiguration = false;
return mBinder;
}
```

@Override

```
public void onRebind(Intent intent) {
    // Called when a client (MainActivity in case of this sample) returns to the foreground
    // and binds once again with this service. The service should cease to be a foreground
    // service when that happens.
    stopForeground(true);
    mChangingConfiguration = false;
    super.onRebind(intent);
}
```

@Override

```
public boolean onUnbind(Intent intent) {
    // Called when the last client (MainActivity in case of this sample) unbinds from this
    // service. If this method is called due to a configuration change in MainActivity, we
    // do nothing. Otherwise, we make this service a foreground service.
    if (!mChangingConfiguration) {
        // If targeting O, use the following code.
        if (Build.VERSION.SDK_INT == Build.VERSION_CODES.O) {
            startForegroundService(new Intent(this, LocationUpdatesService.class));
        }
        String message = "Running location service";
        startForeground(NOTIFICATION_ID, getNotification(message));
    }
    return true; // Ensures onRebind() is called when a client re-binds.
}
```



```
}
```

```
@Override
```

```
public void onDestroy() {
```

```
    mServiceHandler.removeCallbacksAndMessages(null);
```

```
}
```

```
/**
```

```
 * Makes a request for location updates. Note that in this sample we merely log the
```

```
 * {@link SecurityException}.
```

```
 */
```

```
public void requestLocationUpdates() {
```

```
    startService(new Intent(getApplicationContext(), LocationUpdatesService.class));
```

```
    try {
```

```
        mFusedLocationClient.requestLocationUpdates(mLocationRequest, mLocationCallback,  
Looper.myLooper());
```

```
    } catch (SecurityException e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

```
/**
```

```
 * Removes location updates. Note that in this sample we merely log the
```

```
 * {@link SecurityException}.
```

```
 */
```

```
public void removeLocationUpdates() {
```

```
    try {
```

```
        mFusedLocationClient.removeLocationUpdates(mLocationCallback);
```

```
        stopSelf();
```

```

    } catch (SecurityException e) {
        e.printStackTrace();
    }
}

/**
 * Returns the {@link NotificationCompat} used as part of the foreground service.
 */
private Notification getNotification(String message) {

    Intent intent = new Intent(this, LocationUpdatesService.class);

    // Extra to help us figure out if we arrived in onStartCommand via the notification or not.
    intent.putExtra(EXTRA_STARTED_FROM_NOTIFICATION, true);

    // The PendingIntent that leads to a call to onStartCommand() in this service.
    PendingIntent servicePendingIntent = PendingIntent
        .getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);

    // The PendingIntent to launch activity.
    PendingIntent activityPendingIntent = PendingIntent.getActivity(this, 0,
        new Intent(this, MapsActivity.class), 0);

    NotificationCompat.BigTextStyle bigTextStyle = new NotificationCompat.BigTextStyle();
    bigTextStyle.bigText(message);

    NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
CHANNEL_ID)

```

```
        .addAction(R.drawable.ic_launcher_background, getString(R.string.launch_activity),  
activityPendingIntent)
```

```
        .addAction(R.drawable.ic_launcher_background,  
getString(R.string.remove_location_updates), servicePendingIntent)
```

```
        .setContentText(message)
```

```
        .setContentTitle(getString(R.string.app_name))
```

```
        .setOngoing(true)
```

```
        .setColor(getColor(R.color.colorPrimary))
```

```
        .setPriority(Notification.PRIORITY_HIGH)
```

```
        .setSmallIcon(R.mipmap.ic_launcher_round)
```

```
        .setBadgeIconType(NotificationCompat.BADGE_ICON_SMALL)
```

```
        .setLargeIcon(BitmapFactory.decodeResource(getResources(),  
R.mipmap.ic_launcher))
```

```
        .setStyle(bigTextStyle)
```

```
        .setWhen(System.currentTimeMillis());
```

```
// Set the Channel ID for Android O.
```

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
```

```
    builder.setChannelId(CHANNEL_ID); // Channel ID
```

```
}
```

```
return builder.build();
```

```
}
```

```
private void getLastLocation() {
```

```
    try {
```

```
        mFusedLocationClient.getLastLocation().addOnCompleteListener(task -> {
```

```
            if (task.isSuccessful() && task.getResult() != null) {
```

```
                mLocation = task.getResult();
```

```
            } else {
```

```

        Log.w(TAG, "Failed to get location.");
    }
});
} catch (SecurityException unlikely) {
    Log.e(TAG, "Lost location permission." + unlikely);
}
}

```

```

private void onNewLocation(Location location) {
    mLocation = location;
    // Notify anyone listening for broadcasts about the new location.
    Intent intent = new Intent(ACTION_BROADCAST);
    intent.putExtra(EXTRA_LOCATION, location);
    LocalBroadcastManager.getInstance(getApplicationContext()).sendBroadcast(intent);
    notify(new LatLng(location.getLatitude(), location.getLongitude()));
}

```

```

public void notify(LatLng currentLocation){
    FirestoreUtil.getInstance().getDocumentRef().collection("users")
        .whereEqualTo("status", true)
        .get()
        .addOnCompleteListener(task -> {
            List<DocumentSnapshot> documents = task.getResult().getDocuments();
            int count = 0;
            for (DocumentSnapshot snapshot : documents){
                Double latitude = (Double) snapshot.get("lat");
                Double longitude = (Double) snapshot.get("lon");
                if (latitude != null && longitude != null){

```

```

        LatLng latLng = new LatLng(latitude, longitude);
        float distance = LocationUtil.getInstance().findDistance(currentLocation,
latLng);

        if (distance <= 5){
            count ++;
            Log.d(TAG, "notify: " + distance);
        }
    }
}

// if count > 0 that means covid positive patient found
// then a notification will be showing for notify the user
if (count > 0){
    mNotificationManager.notify(NOTIFICATION_ID,
        getNotification(String.format("Alert! %s COVID positive patient detect near
you", count)));
    }
});
}

/**
 * Sets the location request parameters.
 */
private void createLocationRequest() {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(UPDATE_INTERVAL_IN_MILLISECONDS);

mLocationRequest.setFastestInterval(FATEST_UPDATE_INTERVAL_IN_MILLISECONDS
);

    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

```

```

    }

    /**
     * Class used for the client Binder. Since this service runs in the same process as its
     * clients, we don't need to deal with IPC.
     */
    public class LocalBinder extends Binder {
        public LocationUpdatesService getService() {
            return LocationUpdatesService.this;
        }
    }

    public boolean serviceIsRunningInForeground(Context context) {
        ActivityManager manager = (ActivityManager)
            context.getSystemService(Context.ACTIVITY_SERVICE);

        for (ActivityManager.RunningServiceInfo service :
            manager.getRunningServices(Integer.MAX_VALUE)) {
            if (getClass().getName().equals(service.service.getClassName())) {
                if (service.foreground) {
                    return true;
                }
            }
        }

        return false;
    }
}

```

LocationUtil.java

package com.example.sayem;

```
import android.content.Context;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;

import com.google.android.gms.maps.model.LatLng;

import java.util.List;
import java.util.Locale;

/**
 * Created by Imran Khan on 11/23/2020.
 * Email : context.imran@gmail.com
 */

public class LocationUtil {

    private static LocationUtil instance;

    public static LocationUtil getInstance(){
        if (instance == null) {
            instance = new LocationUtil();
        }
        return instance;
    }

    public float findDistance(LatLng source, LatLng destination) {
```

```

    if (source == null || destination == null){
        return 0f;
    }
    float[] results = new float[1];
    Location.distanceBetween (destination.latitude, destination.longitude,
        source.latitude, source.longitude, results);
    return results[0];
}

public String findLocationAddress(Context context, double latitude, double longitude) {
    String address = "";
    Geocoder geocoder = new Geocoder(context, Locale.getDefault());
    try {
        List<Address> addresses = geocoder.getFromLocation(latitude, longitude, 5);
        if (addresses != null) {
            Address returnedAddress = addresses.get(0);
            StringBuilder strReturnedAddress = new StringBuilder();
            for (int i = 0; i <= returnedAddress.getMaxAddressLineIndex(); i++) {
                strReturnedAddress.append(returnedAddress.getAddressLine(i)).append("\n");
            }
            address = strReturnedAddress.toString();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return address;
}
}

```

MapsActivity.java

package com.example.sayem;

import androidx.annotation.NonNull;

import androidx.core.app.ActivityCompat;

import androidx.fragment.app.FragmentActivity;

import androidx.localbroadcastmanager.content.LocalBroadcastManager;

import android.Manifest;

import android.content.BroadcastReceiver;

import android.content.ComponentName;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;

import android.content.ServiceConnection;

import android.content.pm.PackageManager;

import android.graphics.Color;

import android.location.Location;

import android.net.Uri;

import android.os.Bundle;

import android.os.Handler;

import android.os.IBinder;

import android.provider.Settings;

import android.util.Log;

import android.view.View;

```
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
```

```
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.firestore.DocumentSnapshot;
```

```
import java.util.List;
```

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
```

```
    private GoogleMap mMap;
```

```
    private static final String TAG = MapsActivity.class.getSimpleName();
```

```
    // Used in checking for runtime permissions.
```

```
    private static final int REQUEST_PERMISSIONS_REQUEST_CODE = 34;
```

```
    // A reference to the service used to get location updates.
```

```
    private LocationUpdatesService mService = null;
```

```
// Tracks the bound state of the service.
```

```
private boolean mBound = false;
```

```
private LatLng currentLocation = new LatLng(23.746362,90.3885771);
```

```
private EditText latEditText, lngEditText;
```

```
private TextView tvPatientCount;
```

```
private int radius = 1;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_maps);
```

```
    latEditText = findViewById(R.id.editText);
```

```
    lngEditText = findViewById(R.id.editText2);
```

```
    tvPatientCount = findViewById(R.id.tv_patient_count);
```

```
    Spinner spRadius = findViewById(R.id.sp_radius);
```

```
    spRadius.setOnItemClickListener(new AdapterView.OnItemClickListener() {
```

```
        @Override
```

```
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
```

```
            radius = Integer.parseInt(adapterView.getItemAtPosition(i).toString());
```

```
            if (mService != null){
```

```
                mService.requestLocationUpdates();
```

```
            }
```

```

        Log.d(TAG, "onItemSelected: " + radius);
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {

    }

});

// Obtain the SupportMapFragment and get notified when the map is ready to be used.
SupportMapFragment mapFragment =
    (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);

new Handler().postDelayed(() -> runOnUiThread(()
    -> mapFragment.getMapAsync(MapsActivity.this)), 1000);
}

@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
}

public void updateButtonOnClick(View view){
    String phone = FirestoreUtil.getInstance().getCurrentUser().getPhoneNumber();
    if (phone != null && !phone.isEmpty()){
        FirestoreUtil.getInstance().getDocumentRef().collection("users").document(phone)
            .update("lat", currentLocation.latitude);
        FirestoreUtil.getInstance().getDocumentRef().collection("users").document(phone)

```

```

        .update("long", currentLocation.longitude);
    }
}

public void countPatient(){
    FirestoreUtil.getInstance().getDocumentRef().collection("users")
        .whereEqualTo("status", true)
        .get()
        .addOnCompleteListener(task -> {
            List<DocumentSnapshot> documents = task.getResult().getDocuments();
            int count = 0;
            for (DocumentSnapshot snapshot : documents){
                Double latitude = (Double) snapshot.get("lat");
                Double longitude = (Double) snapshot.get("lon");
                if (latitude != null && longitude != null){
                    LatLng latLng = new LatLng(latitude, longitude);
                    float distance = LocationUtil.getInstance().findDistance(currentLocation,
latLng) / 1000;
                    if (distance < radius){
                        count++;
                        Log.d(TAG, "findPatientCount: " +
LocationUtil.getInstance().findDistance(currentLocation, latLng));
                    }
                }
            }
            tvPatientCount.setText(String.format("%s patient found", count));
        });
}

```

```

// Monitors the state of the connection to the service.
private final ServiceConnection mServiceConnection = new ServiceConnection() {

    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        LocationUpdatesService.LocalBinder binder = (LocationUpdatesService.LocalBinder)
service;
        mService = binder.getService();
        mBound = true;
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
        mService = null;
        mBound = false;
    }
};

private void callService() {
    // Check that the user hasn't revoked permissions by going to Settings.
    if (!checkPermissions()) {
        requestPermissions();
    } else {
        new Handler().postDelayed(() -> runOnUiThread(() -> {
            if (Util.checkGpsEnabled(MapsActivity.this)){
                mService.requestLocationUpdates();
            } else
                showGpsInactiveMessage();
        }), 1000);
    }
}

```

```
    }  
}
```

@Override

```
protected void onStart() {  
    super.onStart();  
}
```

@Override

```
protected void onResume() {  
    super.onResume();  
    // Bind to the service. If the service is in foreground mode, this signals to the service  
    // that since this activity is in the foreground, the service can exit foreground mode.  
    bindService(new Intent(this, LocationUpdatesService.class), mServiceConnection,  
Context.BIND_AUTO_CREATE);  
    LocalBroadcastManager.getInstance(this).registerReceiver(broadcastReceiver, new  
IntentFilter(LocationUpdatesService.ACTION_BROADCAST));  
    callService();  
}
```

@Override

```
protected void onPause() {  
    LocalBroadcastManager.getInstance(this).unregisterReceiver(broadcastReceiver);  
    super.onPause();  
}
```

@Override

```
protected void onStop() {  
    if (mBound) {
```

```

        // Unbind from the service. This signals to the service that this activity is no longer
        // in the foreground, and the service can respond by promoting itself to a foreground
        // service.
        unbindService(mServiceConnection);
        mBound = false;
    }
    super.onStop();
}

@Override
protected void onDestroy() {
    super.onDestroy();
}

/**
 * Returns the current state of the permissions needed.
 */
private boolean checkPermissions() {
    return PackageManager.PERMISSION_GRANTED ==
        ActivityCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION);
}

private void requestPermissions() {
    boolean shouldProvideRationale =
        ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.ACCESS_FINE_LOCATION);

    // Provide an additional rationale to the user. This would happen if the user denied the

```



```

// request previously, but didn't check the "Don't ask again" checkbox.
if (shouldProvideRationale) {
    Log.i(TAG, "Displaying permission rationale to provide additional context.");
    Snackbar.make(
        findViewById(R.id.rootView),
        R.string.permission_denied_explanation,
        Snackbar.LENGTH_INDEFINITE)
        .setAction(R.string.ok, new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Request permission
                ActivityCompat.requestPermissions(MapsActivity.this,
                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    REQUEST_PERMISSIONS_REQUEST_CODE);
            }
        })
        .show();
} else {
    Log.i(TAG, "Requesting permission");
    // Request permission. It's possible this can be auto answered if device policy
    // sets the permission in a given state or the user denied the permission
    // previously and checked "Never ask again".
    ActivityCompat.requestPermissions(MapsActivity.this,
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
        REQUEST_PERMISSIONS_REQUEST_CODE);
}
}

```

```

/**
 * Callback received when a permissions request has been completed.
 */

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                     @NonNull int[] grantResults) {
    Log.i(TAG, "onRequestPermissionResult");
    if (requestCode == REQUEST_PERMISSIONS_REQUEST_CODE) {
        if (grantResults.length <= 0) {
            // If user interaction was interrupted, the permission request is cancelled and you
            // receive empty arrays.
            Log.i(TAG, "User interaction was cancelled.");
        } else if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            // Permission was granted.
            if (Util.checkGpsEnabled(MapsActivity.this)){
                mService.requestLocationUpdates();
            } else
                showGpsInactiveMessage();
        } else {
            Snackbar.make(findViewById(R.id.rootView),
                R.string.permission_denied_explanation, Snackbar.LENGTH_INDEFINITE)
                .setAction(R.string.settings, view -> {
                    // Build intent that displays the App settings screen.
                    Intent intent = new Intent();
                    intent.setAction(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                    Uri uri = Uri.fromParts("package", BuildConfig.APPLICATION_ID, null);
                    intent.setData(uri);
                    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    startActivity(intent);
                });
        }
    }
}

```

```

        }).show();
    }
}
}

```

```

private final BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        Location location =
intent.getParcelableExtra(LocationUpdatesService.EXTRA_LOCATION);
        if (location != null & mMap != null) {
            currentLocation = new LatLng(location.getLatitude(), location.getLongitude());
            mMap.addMarker(new MarkerOptions().position(currentLocation));
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(currentLocation, 15));
            latEditText.setText(String.format("%s", currentLocation.latitude));
            lngEditText.setText(String.format("%s", currentLocation.longitude));
            countPatient();
        }
    }
};

```

```

public void showGpsInactiveMessage() {
    Snackbar snackbar = Snackbar
        .make(findViewById(R.id.rootView), getString(R.string.gps_required),
Snackbar.LENGTH_INDEFINITE)
        .setAction(R.string.enable, view -> startActivity(new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS)));

```

```

// Changing message text color

```

```

        snackbar.setActionTextColor(Color.RED);
        // Changing action button text color
        View sbView = snackbar.getView();
        TextView textView =
sbView.findViewById(com.google.android.material.R.id.snackbar_text);
        textView.setTextColor(Color.YELLOW);
        snackbar.show();
    }
}

```

activity_maps.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".MapsActivity"
    android:id="@+id/rootView"
    android:orientation="vertical">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="500dp"

```

```
app:layout_constraintTop_toTopOf="parent"  
/>
```

```
<androidx.cardview.widget.CardView  
    android:id="@+id/cardView"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:cardElevation="4dp"  
    app:cardCornerRadius="10dp"  
    app:layout_constraintBottom_toBottomOf="@+id/map"  
    app:layout_constraintTop_toBottomOf="@+id/map">
```

```
<androidx.constraintlayout.widget.ConstraintLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:padding="16dp">
```

```
<View  
    android:id="@+id/center_point"  
    android:layout_width="5dp"  
    android:layout_height="1dp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

```
<EditText  
    android:id="@+id/editText"
```

```
android:layout_width="0dp"
android:layout_height="wrap_content"
android:ems="10"
android:hint="@string/latitude"
android:inputType="textPersonName"
android:autofillHints=""
android:layout_margin="5dp"
app:layout_constraintEnd_toStartOf="@+id/center_point"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintStart_toStartOf="parent"/>
```

<EditText

```
android:id="@+id/editText2"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:ems="10"
android:hint="@string/longitude"
android:inputType="textPersonName"
android:autofillHints=""
android:layout_margin="5dp"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintStart_toEndOf="@+id/center_point"
app:layout_constraintEnd_toEndOf="parent"
/>
```

<Button

```
android:id="@+id/button"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:onClick="updateButtonOnClick"
android:text="@string/update"
android:layout_marginTop="20dp"
app:layout_constraintTop_toBottomOf="@+id/editText"
app:layout_constraintEnd_toEndOf="@+id/editText2" />
```

<TextView

```
android:id="@+id/tv_radius"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Select Radius"
android:textAppearance="@style/TextAppearance.AppCompat.Medium"
app:layout_constraintBottom_toBottomOf="@+id/button"
app:layout_constraintStart_toStartOf="@+id/editText"
app:layout_constraintTop_toTopOf="@+id/button" />
```

<Spinner

```
android:id="@+id/sp_radius"
android:layout_width="wrap_content"
android:layout_height="0dp"
android:entries="@array/radius"
app:layout_constraintStart_toEndOf="@+id/tv_radius"
app:layout_constraintBottom_toBottomOf="@+id/button"
app:layout_constraintEnd_toStartOf="@+id/button"
app:layout_constraintTop_toTopOf="@+id/button" />
```

<TextView

```
        android:id="@+id/tv_patient_count"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:padding="5dp"
        android:layout_margin="10dp"
        android:background="#f2f2f2"
        android:text="0 patient found"
        android:gravity="center"
        android:lines="1"
        android:ellipsize="end"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
</androidx.cardview.widget.CardView>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Util.java

```
package com.example.sayem;
```

```
import android.content.Context;
```

```
import android.content.pm.PackageManager;
```



```

import android.location.Location;
import android.location.LocationManager;
import android.preference.PreferenceManager;
import android.telephony.TelephonyManager;

import androidx.core.app.ActivityCompat;

import java.text.DateFormat;
import java.util.Date;

/**
 * Created by Imran Khan on 11/24/2020.
 * Email : context.imran@gmail.com
 */

public class Util {

    private static final String KEY_REQUESTING_LOCATION_UPDATES =
"requesting_location_updates";

    /**
     * Returns true if requesting location updates, otherwise returns false.
     *
     * @param context The {@link Context}.
     */
    public static boolean requestingLocationUpdates(Context context) {
        return PreferenceManager.getDefaultSharedPreferences(context)
            .getBoolean(KEY_REQUESTING_LOCATION_UPDATES, false);
    }
}

```

```

/**
 * Stores the location updates state in SharedPreferences.
 *
 * @param requestingLocationUpdates The location updates state.
 */
public static void setRequestingLocationUpdates(Context context, boolean
requestingLocationUpdates) {
    PreferenceManager.getDefaultSharedPreferences(context)
        .edit()
        .putBoolean(KEY_REQUESTING_LOCATION_UPDATES,
requestingLocationUpdates)
        .apply();
}

/**
 * Returns the {@code location} object as a human readable string.
 *
 * @param location The {@link Location}.
 */
public static String getLocationText(Location location) {
    return location == null ? "Unknown location" :
        "(" + location.getLatitude() + ", " + location.getLongitude() + ")";
}

public static String getLocationTitle(Context context) {
    return context.getString(R.string.location_updated,
DateFormat.getDateTimeInstance().format(new Date()));
}

```

```

    public static boolean hasOperator(Context context) {

        TelephonyManager telephonyManager = (TelephonyManager)
context.getSystemService(Context.TELEPHONY_SERVICE);

        return telephonyManager != null && telephonyManager.getSimState() ==
TelephonyManager.SIM_STATE_READY;

    }

    public static boolean checkPermissions(Context context, String permission) {

        return ActivityCompat.checkSelfPermission(context, permission) ==
PackageManager.PERMISSION_GRANTED;

    }

    public static boolean checkGpsEnabled(Context context) {

        LocationManager lm = (LocationManager)
context.getSystemService(Context.LOCATION_SERVICE);

        if (lm != null) {

            return lm.isProviderEnabled(LocationManager.GPS_PROVIDER);

        } else

            return false;

    }

}

```

menu_header.xml

```

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="200dp"

    android:background="#9EA19D"

    android:paddingLeft="15dp">

```

```
<TextView
    android:id="@+id/menu_slogan"
    android:layout_alignParentBottom="true"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:layout_width="wrap_content"
    android:text="Everything to survive" />
```

```
<TextView
    android:fontFamily="bolt"
    android:id="@+id/app_name"
    android:layout_above="@id/menu_slogan"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="Get a Way"
    android:textColor="#000000"
    android:textSize="45sp" />
```

```
<ImageView
    android:layout_width="83dp"
    android:layout_height="83dp"
    android:layout_above="@id/app_name"
    android:src="@drawable/download1" />
```

```
</RelativeLayout>
```

switch_view.xml

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:padding="16dp"

    android:background="#f5f5f5"

    xmlns:app="http://schemas.android.com/apk/res-auto">

    <androidx.appcompat.widget.SwitchCompat
        android:id="@+id/sw_covid_status"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:padding="5dp"

        app:layout_constraintStart_toStartOf="parent"

        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```
