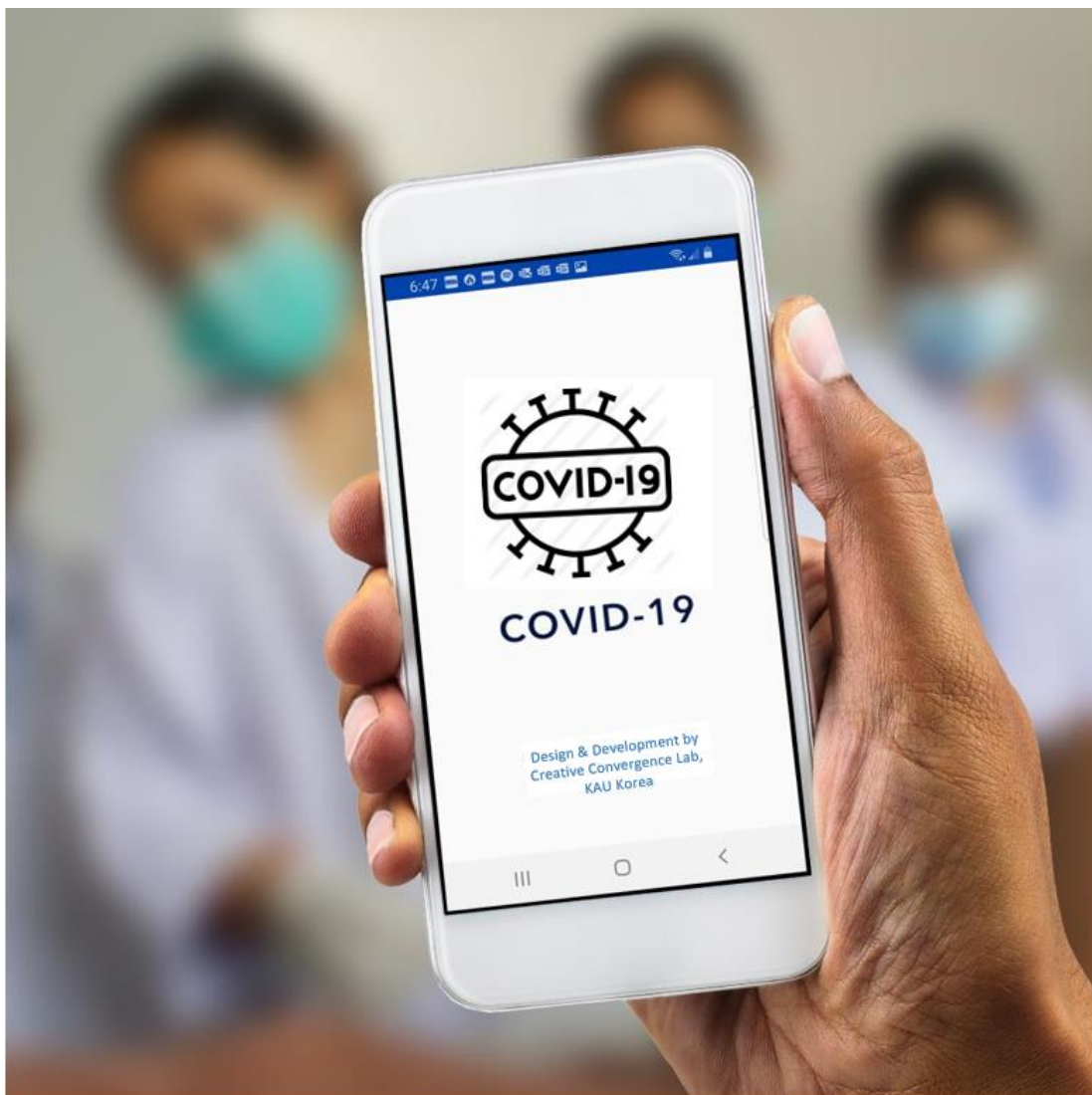


# Implementation MANUAL

## *KS Project Mobile App*



## Table of Contents

1.0 General Information .....	3
1.1 Application Overview .....	3
1.2 Organization of the Manual .....	3
2.0 User Registration .....	3
2.1 User Login .....	4
2.2 OTP Verification.....	4
2.3 Forget Email .....	4
3.0 Main Screen .....	5
3.1 Main Screen GUI .....	5
3.2 App Functionalities .....	6
3.2.1 Multiple sensors data collection.....	7
3.2.2 Covid Contact Tracing .....	8
3.2.3 Hospital Dashboard .....	9
3.2.4 Alert generation on user device .....	10
3.2.5 Geo-Fencing .....	11
3.2.6 Red alert area marking on map .....	12
3.2.7 Tree Construction .....	13
3.2.8 Network Construction .....	7
3.2.9 Access Point (AP) .....	9

## 1.0 General Information

COVID-19 is an infectious disease caused by a newly discovered coronavirus and has spread around the world in a deadly pandemic. To Avoid the spread of COVID-19, needs a such kind of system that is helpful in the reduction of COVID-19 spread. For the reduction of COVID-19 spread, first needs to trace contact of COVID-19 patient. There are multiple ways of tracing contacts of COVID-19 patients. Previously manual collecting contacts of COVID-19 patient using some general information. But in that case, not able to control the spread of COVID-19.

Coronavirus mobile app are programs that governments, hospitals, colleges, universities, and other groups are using to aid in the public response to COVID-19. These applications and dashboards can track Coronavirus patients, provide the latest data about the spread of the virus, help us limit contact, and much more.

KS mobile app is helpful in the tracing contacts of COVID-19 patients, using Bluetooth and GPS technology. If someone is in the contact of COVID-19 patients, we send an alert message to that specific person for COVID-19 testing.

### 1.1 Application Overview

KS mobile app is developed for the contact tracing of COVID-19 patient to avoid the spread of COVID-19. Health departments use contact tracing apps to find people who may have meet someone with COVID-19. KS app help to capture data (contacts of COVID-19 Patient) and watch the movement of people to make the process faster and more useful. KS Project has several features:

**Contact Tracing:** When someone is in your Bluetooth range, app automatically scan that app user and Storing information first locally in a device and then store on the server. When someone is COVID positive, with the help of their contact information, app send alert message to all those who are in contact of that person.

**Alert:** Sending alert message to app users that are in contact of COVID-19 positive person.

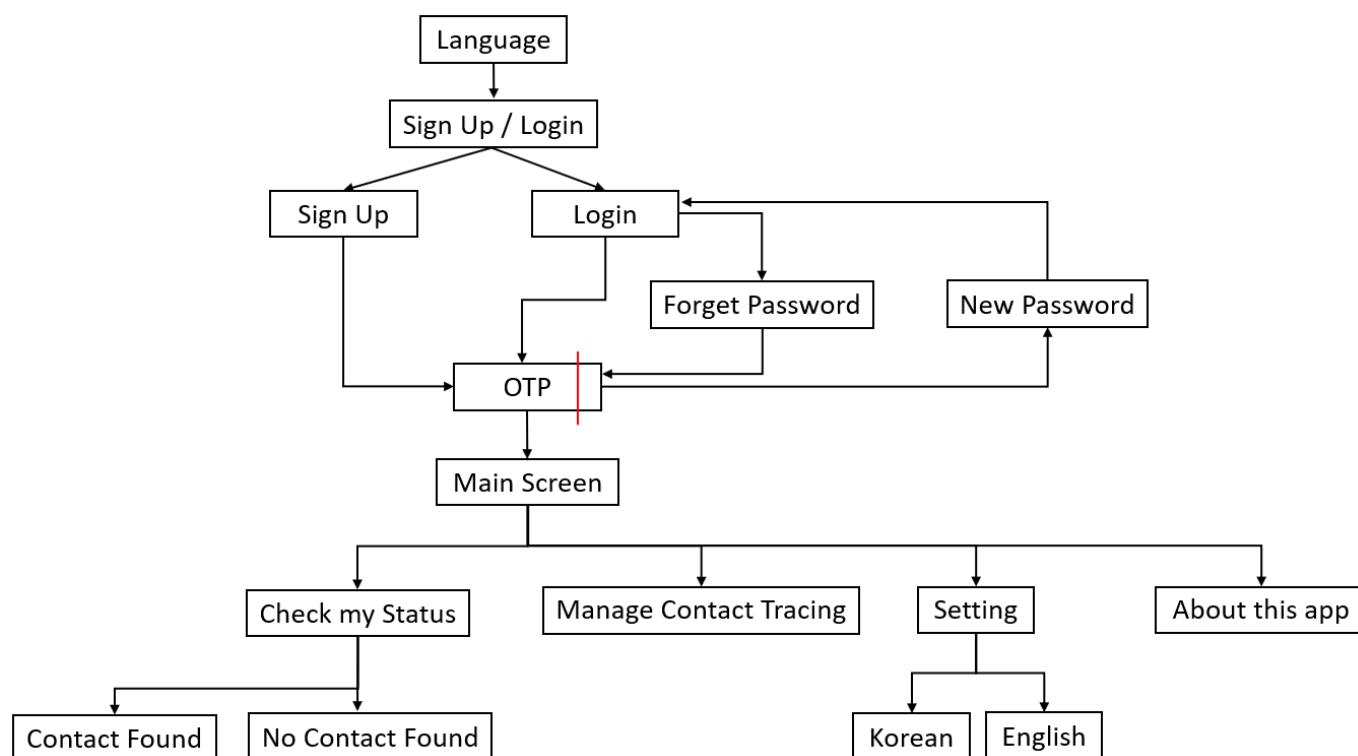
**Geo-Fencing:** When someone is going to the area that is red alert area (area in which COVID-19 patients present), app send alert message that please careful you are going to enter red alert area.

**Contact Tracing Tree:** When someone is COVID positive or in contact of COVID positive

patient, with the help of contact tree, the system identifies the overall contacts of that person.

**Access Point:** Deploying access points in the stores, public places, schools, universities and offices.

Available in English, and Korea Language.



## 1.2 Organization of the Manual

The user manual consists of the following four sections

1. User Registration
2. Login
3. Allow GPS, Storage and Bluetooth
4. Functionalities of KS app

**User Registration** section explains in general terms of the KS application registration.

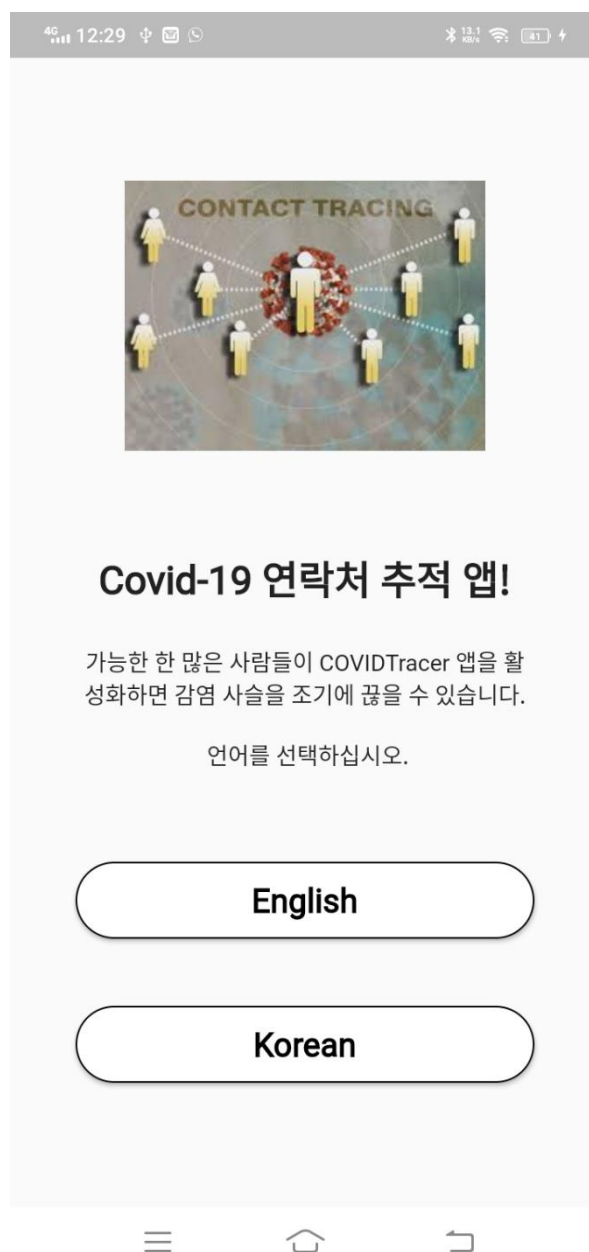
**Login** section explains in general terms of the KS application login.

**Allow GPS, Storage and Bluetooth** section explains about the permission required to KS app.

**Functionalities of KS app** section provides a detailed description of the functionalities of the KS application.

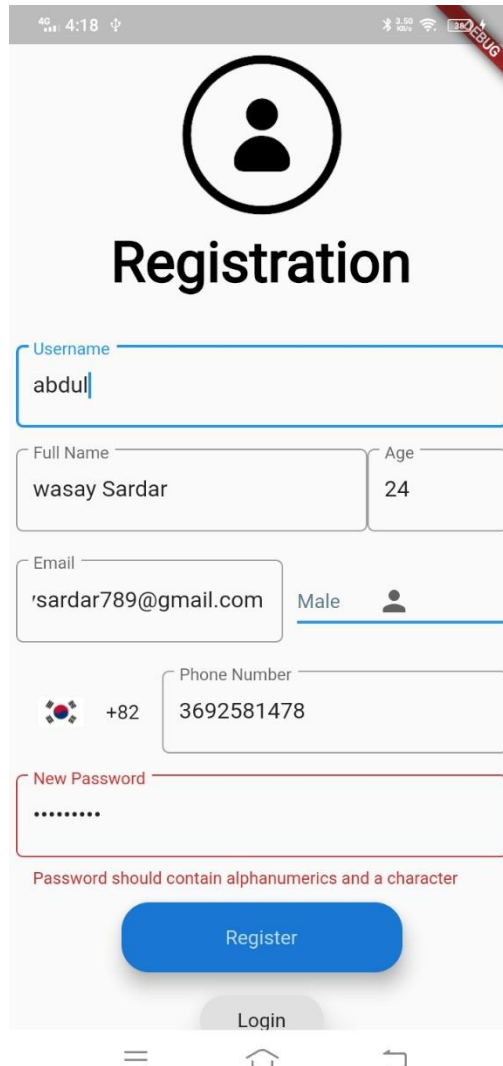
## 2.0 User Registration:

First you need to install KS mobile application in your mobile phone. After installation you need to select your Language:



*Figure 1: Language Screen of KS mobile application*

“**Registration**” screen can also be accessed which is meant for registration of a new user. The GUI would be composed of textboxes e.g., ‘Username’, ‘Full name’, ‘Email’, ‘Gender’, ‘Phone Number’ and ‘New Password’.

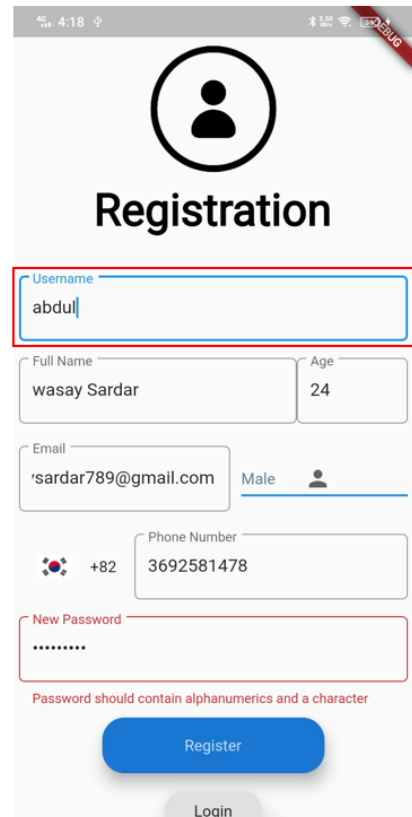


The image shows a mobile application registration screen. At the top, there is a status bar with '4G', '4:18', and battery level. Below it is a large circular icon with a person silhouette. The title 'Registration' is centered. The form includes: a 'Username' field with 'abdul'; a 'Full Name' field with 'wasay Sardar' and an 'Age' field with '24'; an 'Email' field with 'sardar789@gmail.com' and a gender selection with 'Male' and a person icon; a 'Phone Number' field with a South Korean flag icon, '+82', and '3692581478'; and a 'New Password' field with masked characters. A red error message below the password field states: 'Password should contain alphanumerics and a character'. At the bottom are 'Register' and 'Login' buttons. The bottom navigation bar shows a hamburger menu, a home icon, and a back arrow.

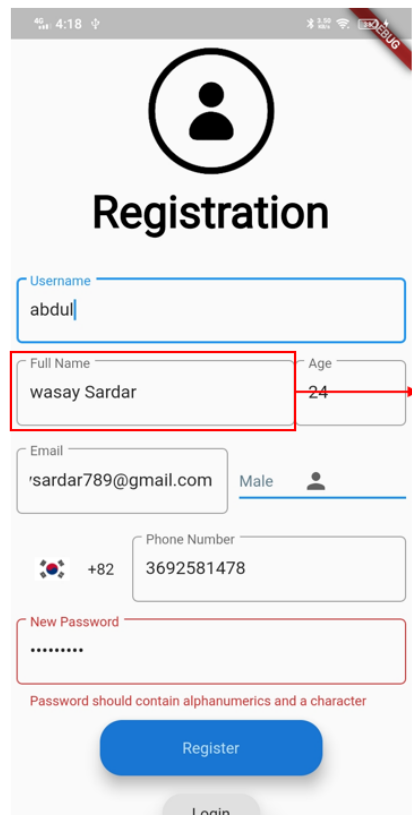
*Figure 2: Registration Screen of KS mobile application*

Several constraints are put upon these text boxes, so that when 'Register' button is pressed, so that user can give correct information:

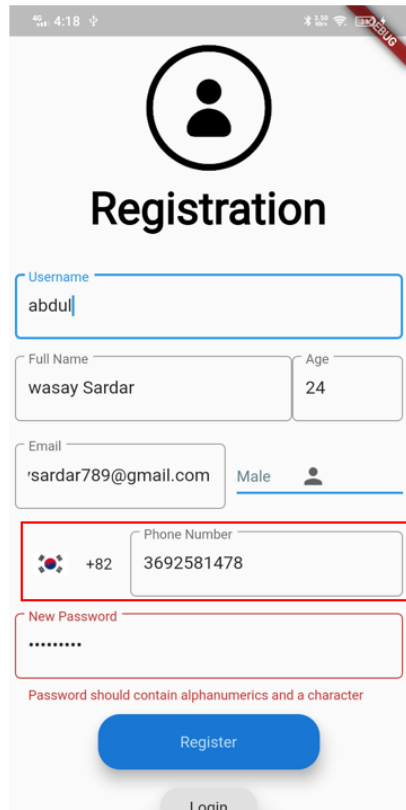
- All fields must be filled
- 'Username' must be alphanumeric
- 'Full Name' must be composed of only alphabets
- 'Email' should have correct format
- 'Age' should be numeric
- 'Phone Number' should be numeric and 10 digits long
- 'New Password' should contain small and capital alphabets alongside with numeric and special characters
- Username, Phone Number and Email should not exist in database

A screenshot of a mobile registration form. The form has a header with a person icon and the title 'Registration'. Below the title, there are several input fields: 'Username' (containing 'abdul'), 'Full Name' (containing 'wasay Sardar'), 'Age' (containing '24'), 'Email' (containing 'sardar789@gmail.com'), 'Phone Number' (containing '+82 3692581478'), and 'New Password' (containing '\*\*\*\*\*'). A red box highlights the 'Username' field, and a red arrow points from it to a separate text box on the right. The text box contains the text 'Username is alphanumeric e.g ccrlab123'. Below the 'New Password' field, there is a red error message: 'Password should contain alphanumerics and a character'. At the bottom of the form, there are two buttons: 'Register' (blue) and 'Login' (grey).

*Figure 3: Constrains on the username*

A screenshot of the same mobile registration form as in Figure 3. In this version, a red box highlights the 'Full Name' field, which contains the text 'wasay Sardar'. A red arrow points from this field to a separate text box on the right. The text box contains the text 'Full Name Only Contains Alphabets e.g Creative Convergence Lab'. The 'Username' field still contains 'abdul'. The 'Age' field contains '24'. The 'Email' field contains 'sardar789@gmail.com'. The 'Phone Number' field contains '+82 3692581478'. The 'New Password' field contains '\*\*\*\*\*'. The red error message 'Password should contain alphanumerics and a character' is still present. The 'Register' and 'Login' buttons are at the bottom.

*Figure 4: Constrains on the Full name.*



Registration

Username: abdul

Full Name: wasay Sardar Age: 24

Email: 'sardar789@gmail.com' Male

Phone Number: +82 3692581478

New Password: .....

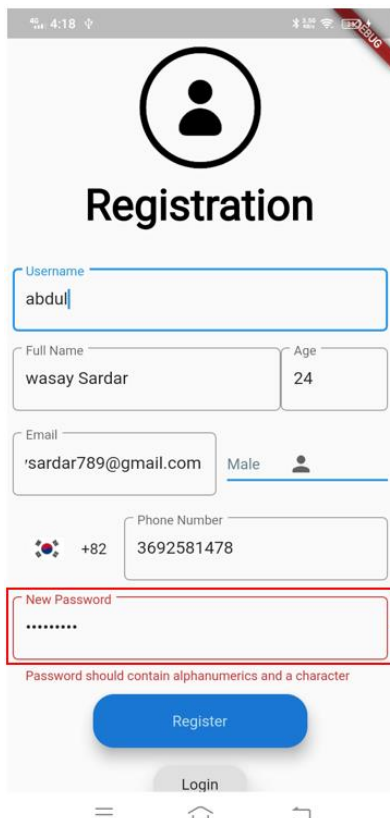
Password should contain alphanumerics and a character

Register

Login

Phone Number Must Contains only numbers and length is equal to 10 e.g. 1021880131

Figure 5: Constrains on the phone number.



Registration

Username: abdul

Full Name: wasay Sardar Age: 24

Email: 'sardar789@gmail.com' Male

Phone Number: +82 3692581478

New Password: .....

Password should contain alphanumerics and a character

Register

Login

Password Contains Capital, small alphabets, numbers and special characters e.g Ccrlab@123

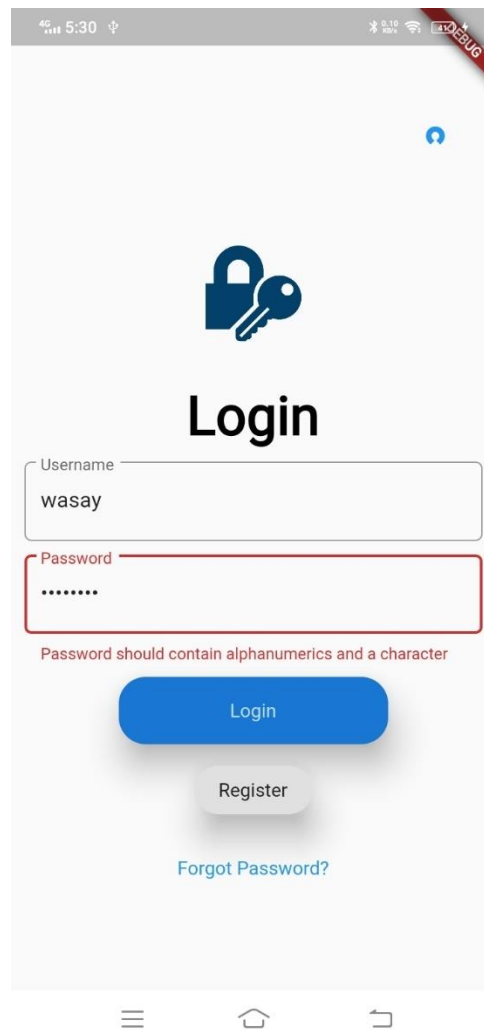
Figure 6: Constrains on the registration password.



Furthermore, in order to enforce users to make a unique account on a single advice, a backend functionality is also implemented where when a user tries to make a new account on the safe device, after uninstalling and then installing the app, it gets detected on the “Registration” screen and an error dialog box is shown device is already register.

## 2.1 User Login:

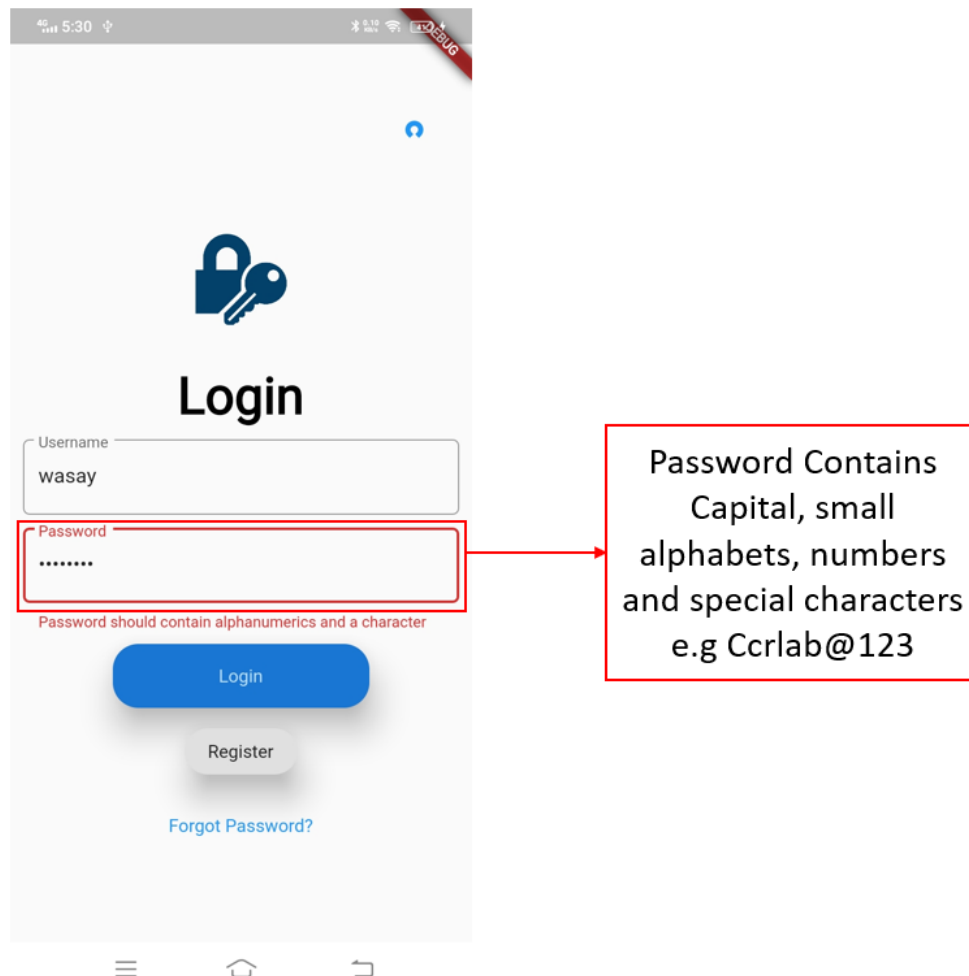
A user can be prompted to “Login” screen, where GUI would consist of ‘Username’ and ‘Password’ textboxes.



*Figure 7: login screen of KS Mobile Application.*

It is necessary that these textboxes follow certain constraints, like username must be alphanumeric or password must contain small and capital alphabets with numeric and special characters, otherwise corresponding error highlights or prompts would be shown. For example, in following figure, as password did not follow the guideline, corresponding textbox got highlighted with correction guideline.

Additionally, an error dialog box would also be shown when username or password does not exist in database.

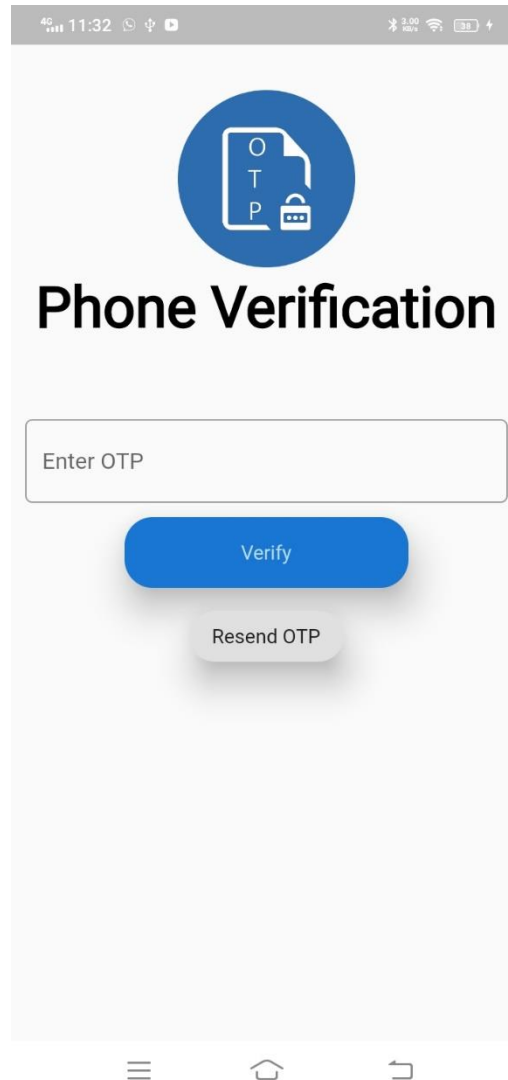


*Figure 8: Constrains on login password.*

## 2.2 OTP Verification:

Whether a user logs in or registers, user is always prompted towards 'Phone Verification' screen where a onetime password (OTP) is sent to email of user, as a security measure to confirm the authenticity of user. Upon correct OTP entering, the user can proceed towards main GUI of the app.

An exemplary view of GUI of ‘Phone Verification’ screen is shown below in following figure.

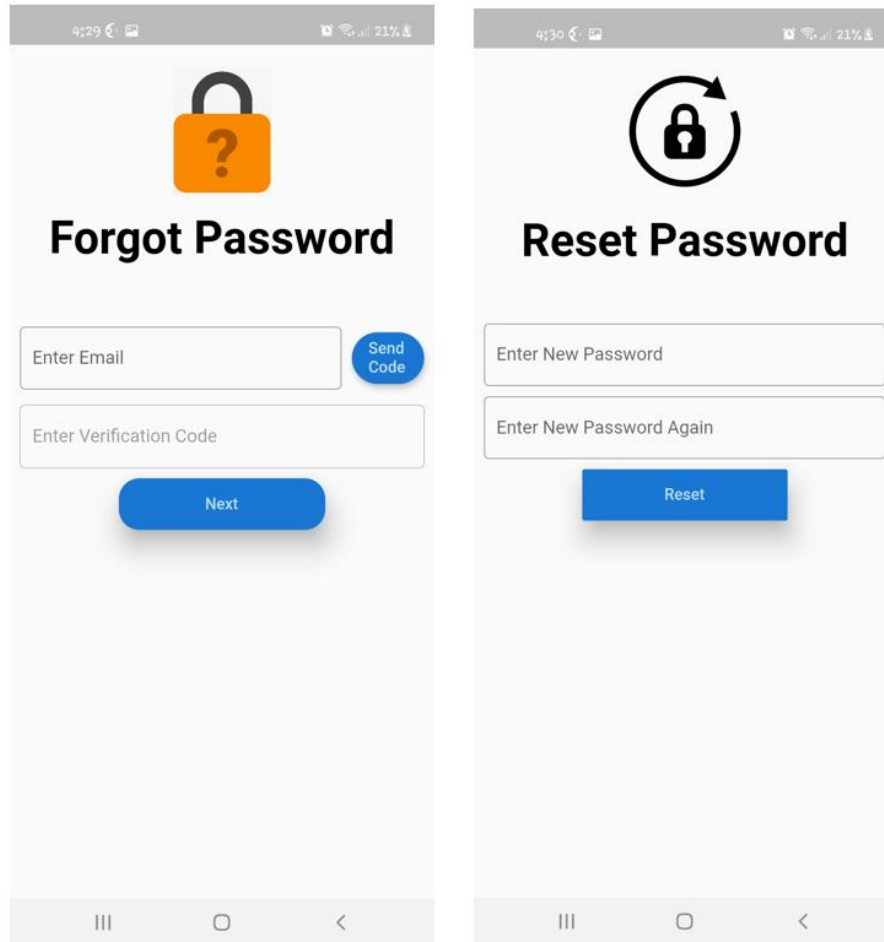


*Figure 9: OTP verification screen.*

## 2.3 Forget Email:

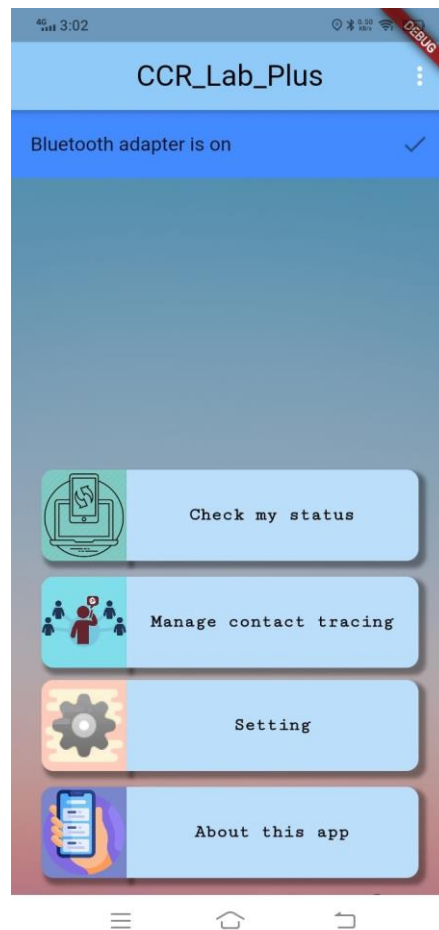
In case a user forgets his/her password, a user can easily access the “Password-Reset” functionality through “Login” screen. Here a user needs to first enter his/her email in ‘Forgot Password’ screen, upon which, a verification code would send to user’s email. Upon entering the correct verification code, the user can be prompted towards ‘Reset Password’ screen, where the user can enter new password and confirm that password, upon pressing the ‘Reset’ button. For the sake of visualization, the GUI of both ‘Forgot Password’ and ‘Reset Password’ screens are shown in following figure.

Additionally, similar constraints to that of “Login” and “Registration Screen” are also added in this functionality screens.

The image displays two side-by-side mobile application screens. The left screen, titled 'Forgot Password', features an orange padlock icon with a question mark. It includes an input field for 'Enter Email', a blue 'Send Code' button, another input field for 'Enter Verification Code', and a blue 'Next' button. The right screen, titled 'Reset Password', features a black padlock icon with a circular arrow. It includes an input field for 'Enter New Password', another input field for 'Enter New Password Again', and a blue 'Reset' button. Both screens have a status bar at the top showing the time (4:29 and 4:30 respectively) and battery level (21%). The bottom of each screen shows a standard Android navigation bar with three icons: a square, a circle, and a triangle.

*Figure 10: Forget password screen.*

### 3.1 Main Screen GUI:



*Figure 11: Main screen of the KS Mobile Application.*

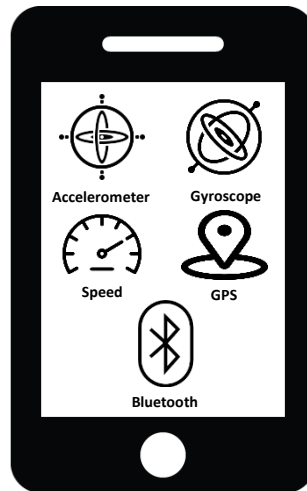
### 3.2 App Functionalities

There are following functionalities of our app.

- Multiple sensors data collection
- Covid Contact Tracing
- Hospital Dashboard
- Alert generations on user device
- Geo- Fencing
- Red alert area marking on map
- Tree Construction
- Network Construction
- Access Point (AP)

### 3.2.1 Multiple sensors data collection:

Multiple sensors including GPS, Speed, Accelerometer, and Gyroscope data is collected. With the help of GPS sensor, we trace the covid positive person locations where he/she visit in last 7 days. Also, GPS sensor helpful in the plotting of map. Accelerometer and Gyroscope sensor helpful in the activity's recognition, monitoring and classification in normal/epidemic condition for epidemic spread prevention and control. Bluetooth is also used in the contact tracing of covid positive person.



*Figure 12: Multiple sensors.*

All sensors' data is stored (in a CSV format) first locally in a mobile device and then store all sensor data in a server (PostgreSQL).

	A	B	C	D	E	F	G	H	I	J	K
1	User_Id	Date	Time	Latitude	Longitude	Altitude	Speed	ax	ay	az	gx
2	farman123	7/11/2021	2:19:44	37.6007734	126.864617	41.4016984	0.02786	-0.1	0	9.8	0
3	farman123	7/11/2021	2:19:45	37.6007734	126.864617	41.4016984	0.02786	-0.1	0	9.8	0
4	farman123	7/11/2021	2:19:46	37.6007604	126.864609	42.1381283	0	-0.1	0	9.8	0
5	farman123	7/11/2021	2:19:47	37.6007604	126.864609	42.1381283	0	-0.1	0	9.8	0
6	farman123	7/11/2021	2:19:48	37.6007604	126.864609	42.1381283	0	-0.1	0	9.8	0
7	farman123	7/11/2021	2:19:49	37.6007604	126.864609	42.1381283	0	-0.1	0	9.8	0
8	farman123	7/11/2021	2:19:50	37.6007604	126.864609	42.1381283	0	-0.1	0	9.8	0
9	farman123	7/11/2021	2:19:51	37.6007604	126.864609	42.1381283	0	-0.1	0	9.8	0
10	farman123	7/11/2021	2:19:52	37.6007608	126.864609	41.8172043	0.004216	-0.1	0	9.8	0
11	farman123	7/11/2021	2:19:53	37.6007608	126.864609	41.8172043	0.004216	-0.1	0	9.8	0
12	farman123	7/11/2021	2:19:54	37.6007608	126.864609	41.8172043	0.004216	-0.1	0	9.8	0
13	farman123	7/11/2021	2:19:55	37.6007608	126.864609	41.8172043	0.004216	-0.1	0	9.8	0
14	farman123	7/11/2021	2:19:56	37.6007608	126.864609	41.8172043	0.004216	-0.1	0	9.8	0
15	farman123	7/11/2021	2:19:57	37.6007547	126.86461	41.6716792	0.117803	-0.1	0	9.8	0
16	farman123	7/11/2021	2:19:58	37.6007547	126.86461	41.6716792	0.117803	-0.1	0	9.8	0
17	farman123	7/11/2021	2:19:59	37.6007547	126.86461	41.6716792	0.117803	-0.1	0	9.8	0
18	farman123	7/11/2021	2:20:00	37.6007547	126.86461	41.6716792	0.117803	-0.1	0	9.8	0
19	farman123	7/11/2021	2:20:02	37.6007551	126.864616	41.7453085	0.100501	-0.1	0	9.8	0
20	farman123	7/11/2021	2:20:03	37.6007551	126.864616	41.7453085	0.100501	-0.1	0	9.8	0
21	farman123	7/11/2021	2:20:04	37.6007551	126.864616	41.7453085	0.100501	-0.1	0	9.8	0
22	farman123	7/11/2021	2:20:05	37.6007551	126.864616	41.7453085	0.100501	-0.1	0	9.8	0
23	farman123	7/11/2021	2:20:06	37.6007574	126.864612	41.661952	0.062939	-0.1	0	9.8	0
24	farman123	7/11/2021	2:20:07	37.6007574	126.864612	41.661952	0.062939	-0.1	0	9.8	0
25	farman123	7/11/2021	2:20:08	37.6007574	126.864612	41.661952	0.062939	-0.1	0	9.8	0
26	farman123	7/11/2021	2:20:09	37.6007574	126.864612	41.661952	0.062939	-0.1	0	9.8	0
27	farman123	7/11/2021	2:20:10	37.6007574	126.864612	41.661952	0.062939	-0.1	0	9.8	0
28	farman123	7/11/2021	2:20:11	37.6007577	126.864614	41.6148856	0.023605	-0.1	0	9.8	0
29	farman123	7/11/2021	2:20:12	37.6007577	126.864614	41.6148856	0.023605	-0.1	0	9.8	0
30	farman123	7/11/2021	2:20:13	37.6007577	126.864614	41.6148856	0.023605	-0.1	0	9.8	0
31	farman123	7/11/2021	2:20:14	37.6007577	126.864614	41.6148856	0.023605	-0.1	0	9.8	0
32	farman123	7/11/2021	2:20:15	37.6007577	126.864614	41.6148856	0.023605	-0.1	0	9.8	0
33	farman123	7/11/2021	2:20:16	37.6007611	126.864612	41.5416143	0.065629	-0.1	0	9.8	0
34	farman123	7/11/2021	2:20:17	37.6007611	126.864612	41.5416143	0.065629	-0.1	0	9.8	0

Figure 12: Sensor data csv in a local device.

test\_db/ubuntu@postgresql

Query Editor

Query History

Scratch Pad

1

SELECT \* FROM public.sensor\_data

2

Data Output

Explain

Messages

Notifications

	uname text	date text	time text	lat text	long text	altitude text	velocity text	ax text	ay text	az text	gx text	gy text	gz text
1	wasay123	2021-07-11	23:56:57	37.6007602	126.8646142	41.68063974109031	0.0000000	-0.1	-0.0	9.8	0.0	0.0	-0.0
2	wasay123	2021-07-11	23:56:58	37.6007602	126.8646142	41.68063974109031	0.0000000	-0.1	-0.1	9.8	-0.0	-0.0	0.0
3	wasay123	2021-07-11	23:56:59	37.6007602	126.8646142	41.68063974109031	0.0000000	-0.1	-0.0	9.8	0.0	-0.0	-0.0
4	wasay123	2021-07-11	23:57:00	37.6007602	126.8646142	41.68063974109031	0.0000000	-0.1	-0.0	9.8	0.0	0.0	0.0
5	wasay123	2021-07-11	23:57:01	37.6007602	126.8646142	41.68063974109031	0.0000000	-0.1	-0.0	9.8	-0.0	-0.0	0.0
6	wasay123	2021-07-11	23:57:02	37.6007602	126.8646142	41.68063974109031	0.0000000	-0.1	-0.0	9.8	-0.0	0.0	-0.0
7	wasay123	2021-07-11	23:57:03	37.6007602	126.8646142	41.68063974109031	0.0000000	-0.1	-0.1	9.8	0.0	0.0	-0.0
8	wasay123	2021-07-11	23:57:04	37.6007673	126.8646177	41.558588955657946	0.0849141	-0.1	-0.0	9.8	-0.0	0.0	-0.0
9	wasay123	2021-07-11	23:57:05	37.6007673	126.8646177	41.558588955657946	0.0849141	-0.1	-0.0	9.8	0.0	0.0	0.0
10	wasay123	2021-07-11	23:57:06	37.6007673	126.8646177	41.558588955657946	0.0849141	-0.1	-0.0	9.8	0.0	0.0	0.0
11	wasay123	2021-07-11	23:57:07	37.6007673	126.8646177	41.558588955657946	0.0849141	-0.1	-0.1	9.8	-0.0	0.0	-0.0
12	wasay123	2021-07-11	23:57:08	37.6007679	126.8646133	41.49656762415453	0.0660006	-0.1	-0.1	9.8	-0.0	-0.0	0.0
13	wasay123	2021-07-11	23:57:09	37.6007679	126.8646133	41.49656762415453	0.0660006	-0.1	-0.1	9.8	0.0	0.0	0.0
14	wasay123	2021-07-11	23:57:10	37.6007679	126.8646133	41.49656762415453	0.0660006	-0.1	-0.1	9.8	-0.0	-0.0	-0.0
15	wasay123	2021-07-11	23:57:11	37.6007679	126.8646133	41.49656762415453	0.0660006	-0.1	-0.1	9.8	-0.0	0.0	-0.0
16	wasay123	2021-07-11	23:57:12	37.6007679	126.8646133	41.49656762415453	0.0660006	-0.1	-0.0	9.8	0.0	-0.0	0.0
17	wasay123	2021-07-11	23:57:13	37.6007679	126.8646133	41.49656762415453	0.0660006	-0.1	-0.1	9.8	-0.0	0.0	0.0
18	wasay123	2021-07-11	23:57:14	37.6007656	126.8646293	41.52957650658271	0.2084670	-0.1	-0.1	9.8	0.0	-0.0	0.0
19	wasay123	2021-07-11	23:57:15	37.6007656	126.8646293	41.52957650658271	0.2084670	-0.1	-0.0	9.8	-0.0	0.0	-0.0
20	wasay123	2021-07-11	23:57:16	37.6007656	126.8646293	41.52957650658271	0.2084670	-0.1	-0.0	9.8	-0.0	0.0	0.0
21	wasay123	2021-07-11	23:57:17	37.6007656	126.8646293	41.52957650658271	0.2084670	-0.1	-0.1	9.8	-0.0	0.0	0.0
22	wasay123	2021-07-11	23:57:18	37.600762	126.8646273	41.440167641421084	0.0766245	-0.1	-0.0	9.8	0.0	-0.0	0.0
23	wasay123	2021-07-11	23:57:19	37.600762	126.8646273	41.440167641421084	0.0766245	-0.1	-0.0	9.8	-0.0	-0.0	0.0
24	wasay123	2021-07-11	23:57:20	37.600762	126.8646273	41.440167641421084	0.0766245	-0.1	-0.0	9.8	0.0	0.0	-0.0
25	wasay123	2021-07-11	23:57:21	37.600762	126.8646273	41.440167641421084	0.0766245	-0.1	-0.0	9.8	0.0	0.0	-0.0

Figure 13: Sensor data table (Server).

### 3.2.2 Covid Contact Tracing:

In the KS mobile application for the contact tracing used a beacon (Bluetooth) technology. When user is register, system assign that specific user a unique beacon UUID. This unique beacon id is the identifier of each user. Previously, mobile applications used MAC address for the identification of each user, but each device has their own MAC address and MAC address easily identify by anyone. In this app, our focus is on privacy, so instead of using MAC address used a beacon (unique UUID) technology.

In order to invoke privacy preserving in Bluetooth based contact tracing, a mechanism involving both smartphone and server is implemented, to associate a unique and intractable identifier to each smartphone, which would be only transmitted, while the corresponding Bluetooth MAC address remains hidden.



On client side, when user is register system assign that user their Unique ID. With the help of Bluetooth technology app transmit and scan each user beacon ID continuously. Both transmission and scanning of beacon id is done simultaneously. We store each user beacon id first in a mobile device and then send to server. On server-side system as also an information about the covid infected persons. When someone is covid positive system check their contact automatically and send alert message to all those persons that are in the contact of covid positive person.

test\_db/ubuntu@postgresql

Query Editor

Query History

Scratch Pad

1 SELECT \* FROM public.beacon\_scan

2

Data Output

Explain

Messages

Notifications

	uname text	beaconid_others text	date text	time time without time zone	distance text	u_beaconid text
1	harry12	62636364366137302d653234642d3131	2021-07-11	23:14:19	0.026261434835529902	34366562376636302d653232382d3131
2	wasay123	34366562376636302d653232382d3131	2021-07-11	23:15:02	0.7083182641128207	66626334396232302d653232372d3131
3	harry12	62636364366137302d653234642d3131	2021-07-11	23:16:05	0.15611358027757272	34366562376636302d653232382d3131
4	jamshaid12	34366562376636302d653232382d3131	2021-07-11	23:16:30	0.4124938537103828	62636364366137302d653234642d3131
5	jamshaid12	34366562376636302d653232382d3131	2021-07-11	23:18:55	0.4124938537103828	62636364366137302d653234642d3131
6	harry12	62636364366137302d653234642d3131	2021-07-11	23:18:53	0.000020034561445840098	34366562376636302d653232382d3131
7	wasay123	62636364366137302d653234642d3131	2021-07-11	23:20:17	1.7955643585953094	66626334396232302d653232372d3131
8	jamshaid12	34366562376636302d653232382d3131	2021-07-11	23:22:23	0.004038955367298437	62636364366137302d653234642d3131
9	harry12	66626334396232302d653232372d3131	2021-07-11	23:22:15	1.2744385809290257	34366562376636302d653232382d3131
10	jamshaid12	34366562376636302d653232382d3131	2021-07-11	23:25:55	1.4298239596645876	62636364366137302d653234642d3131
11	wasay123	62636364366137302d653234642d3131	2021-07-11	23:26:03	0.0004028260290140457	66626334396232302d653232372d3131
12	jamshaid12	66626334396232302d653232372d3131	2021-07-11	23:28:56	0.0005795090158611954	62636364366137302d653234642d3131
13	wasay123	62636364366137302d653234642d3131	2021-07-11	23:29:36	0.0008231137613537373	66626334396232302d653232372d3131
14	wasay123	33386566643137302d653264652d3131	2021-07-12	16:09:12	1.6029664746240697	66626334396232302d653232372d3131
15	wasay123	33386566643137302d653264652d3131	2021-07-12	16:35:24	4.756481003459447	66626334396232302d653232372d3131
16	wasay123	33386566643137302d653264652d3131	2021-07-12	16:41:04	2.5090972430581693	66626334396232302d653232372d3131
17	wasay123	33386566643137302d653264652d3131	2021-07-12	16:42:44	2.5090972430581693	66626334396232302d653232372d3131
18	wasay123	33386566643137302d653264652d3131	2021-07-12	16:42:44	2.5090972430581693	66626334396232302d653232372d3131
19	wasay123	33386566643137302d653264652d3131	2021-07-12	16:42:44	2.5090972430581693	66626334396232302d653232372d3131
20	wasay123	33386566643137302d653264652d3131	2021-07-12	16:42:44	2.5090972430581693	66626334396232302d653232372d3131
21	wasay123	33386566643137302d653264652d3131	2021-07-12	16:42:45	2.5090972430581693	66626334396232302d653232372d3131
22	wasay123	33386566643137302d653264652d3131	2021-07-12	16:42:45	2.5090972430581693	66626334396232302d653232372d3131
23	wasay123	33386566643137302d653264652d3131	2021-07-12	16:42:45	2.5090972430581693	66626334396232302d653232372d3131
24	wasay123	33386566643137302d653264652d3131	2021-07-12	16:42:45	2.5090972430581693	66626334396232302d653232372d3131

Figure 14: Beacon table

### 3.2.3 Hospital Dashboard:

Our system needs the information about the covid positive persons. For the collection of covid positive persons information, hospital dashboard is designed. In a hospital dashboard there are two kinds of user's admin and hospital. Admin is the authority that approve each hospital with the help of some general information.

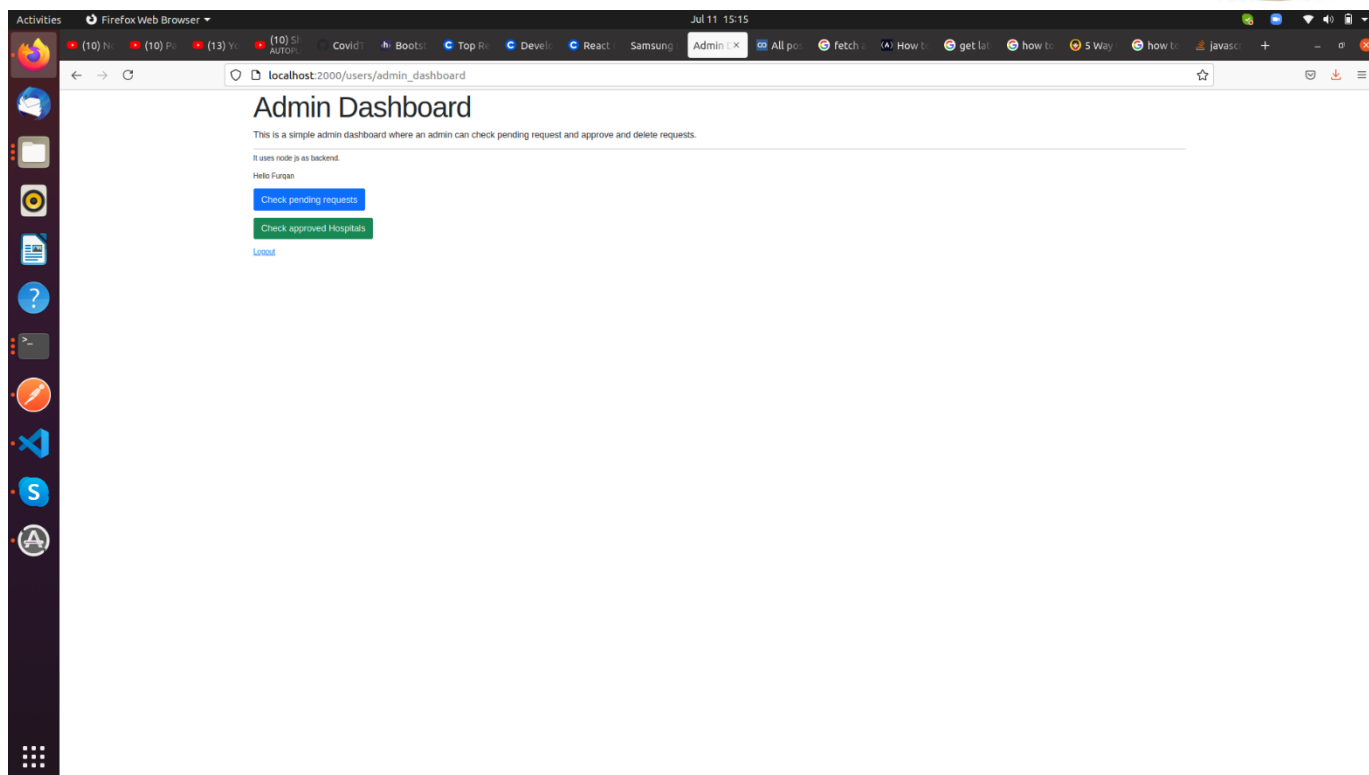


Figure 15: Hospital dashboard

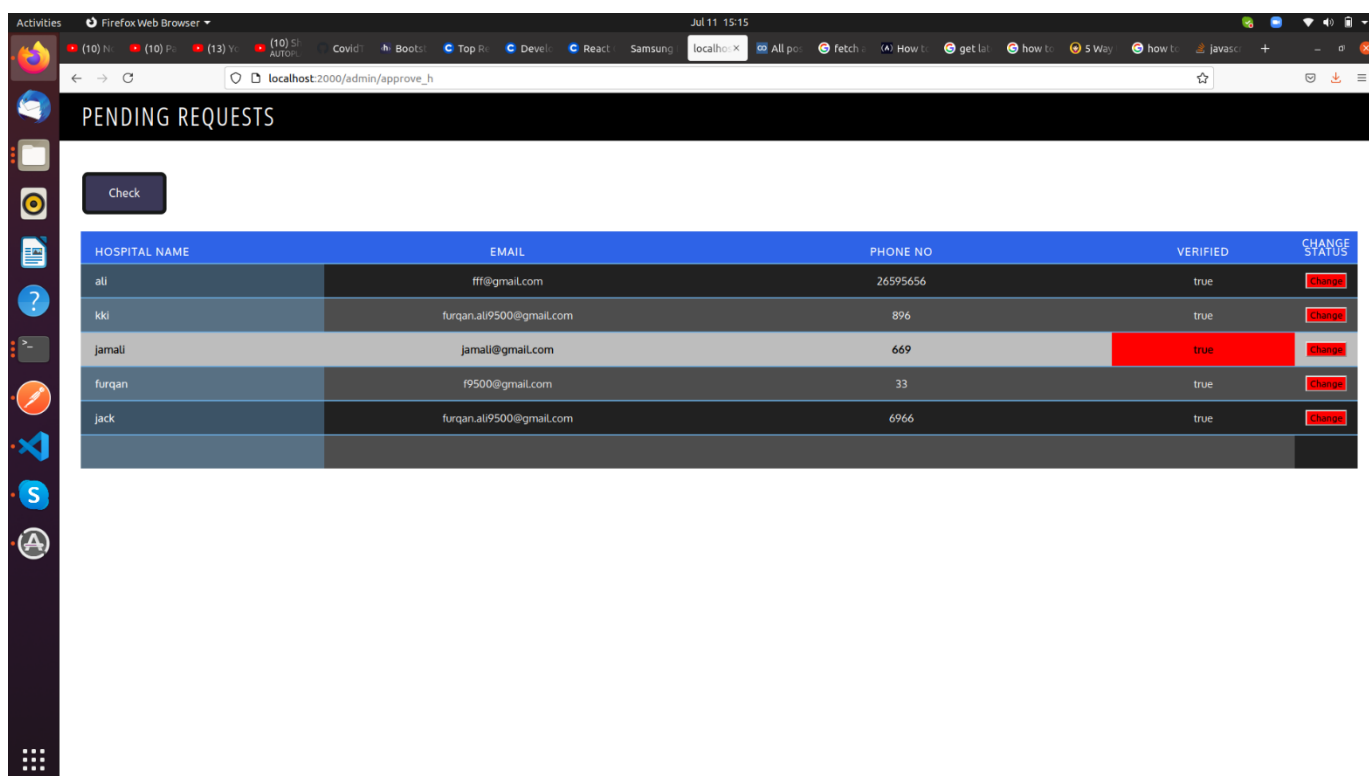
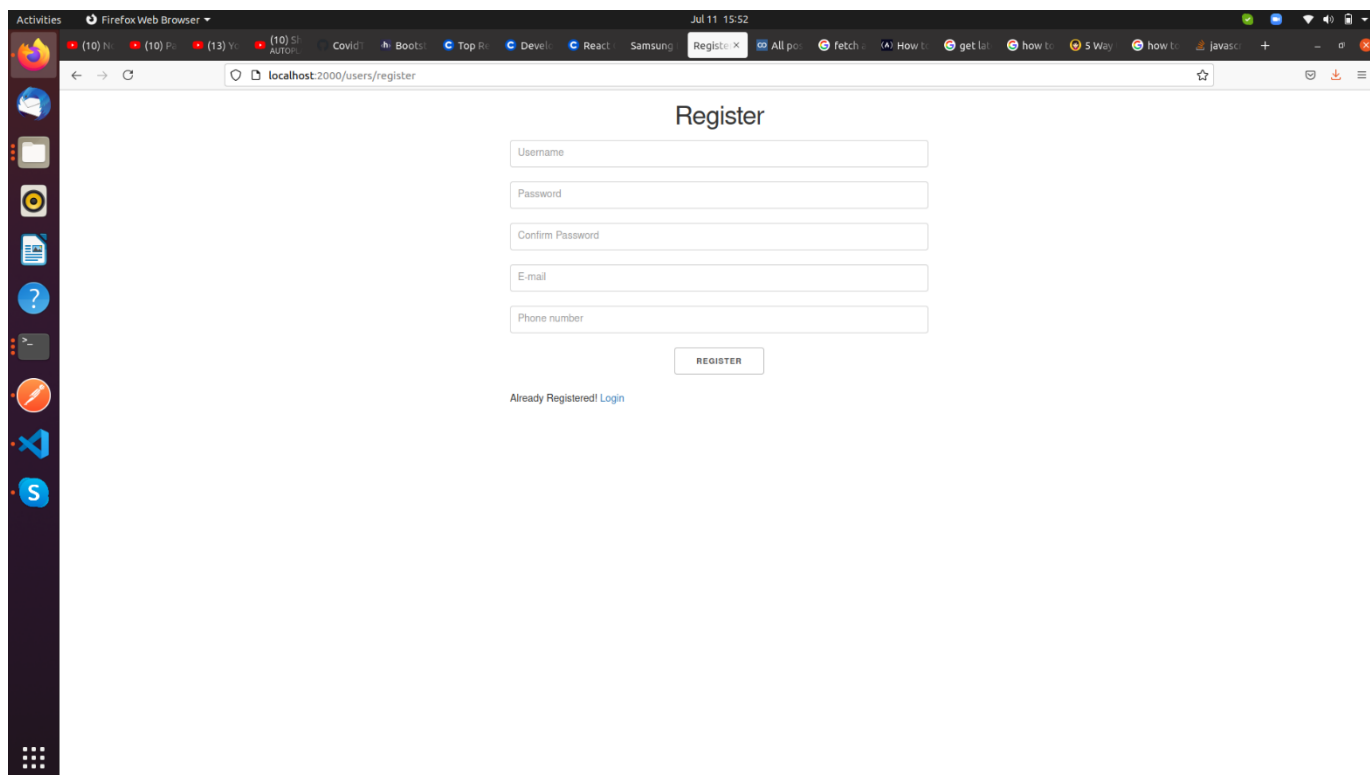


Figure 16: Admin hospital pending requests page

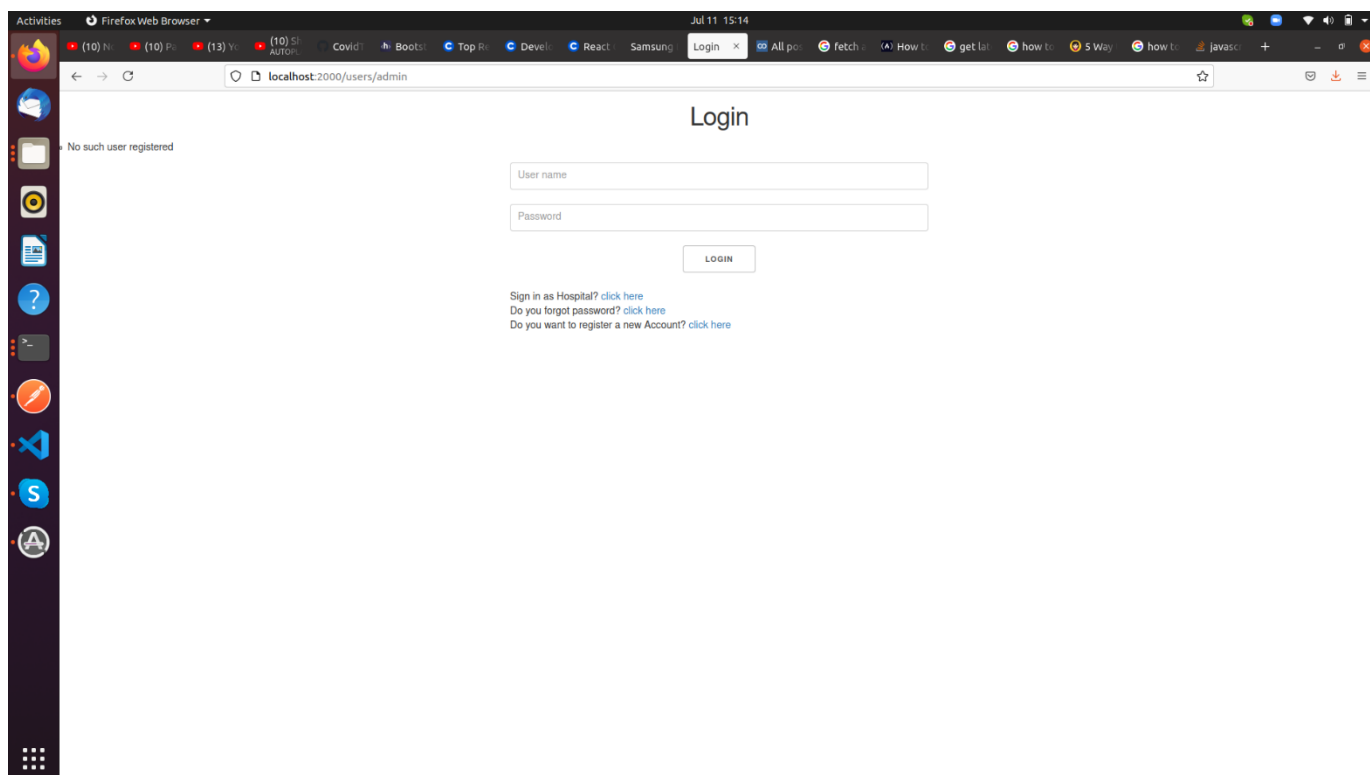
Hospital open webapp application and register hospital. After the registration hospital needs some approval from the administrator (admin). When hospital is approved then hospital login. Now hospital just enter each

patient's username and all information about that user is fetched from server automatically. Hospital needs to verify their information and change their status.



The screenshot shows a web browser window with the URL `localhost:2000/users/register`. The page title is "Register". It contains a registration form with the following fields: Username, Password, Confirm Password, E-mail, and Phone number. Below the form is a "REGISTER" button. At the bottom of the form, there is a link: "Already Registered? [Login](#)".

*Figure 17: Hospital registration page*



The screenshot shows a web browser window with the URL `localhost:2000/users/admin`. The page title is "Login". It contains a login form with the following fields: User name and Password. Below the form is a "LOGIN" button. At the bottom of the form, there are three links: "Sign in as Hospital? [click here](#)", "Do you forgot password? [click here](#)", and "Do you want to register a new Account? [click here](#)".

*Figure 18: Hospital login page*

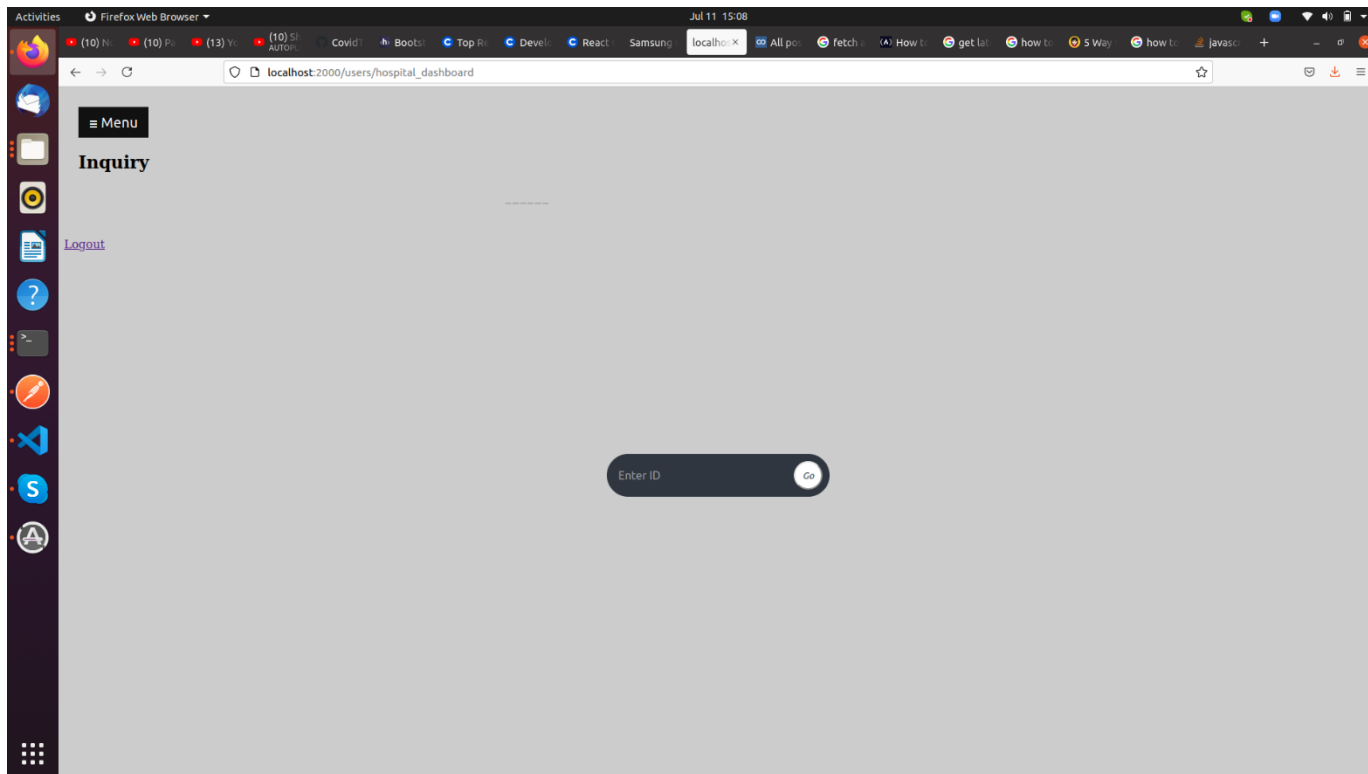


Figure 19: Search user id (hospital main page).

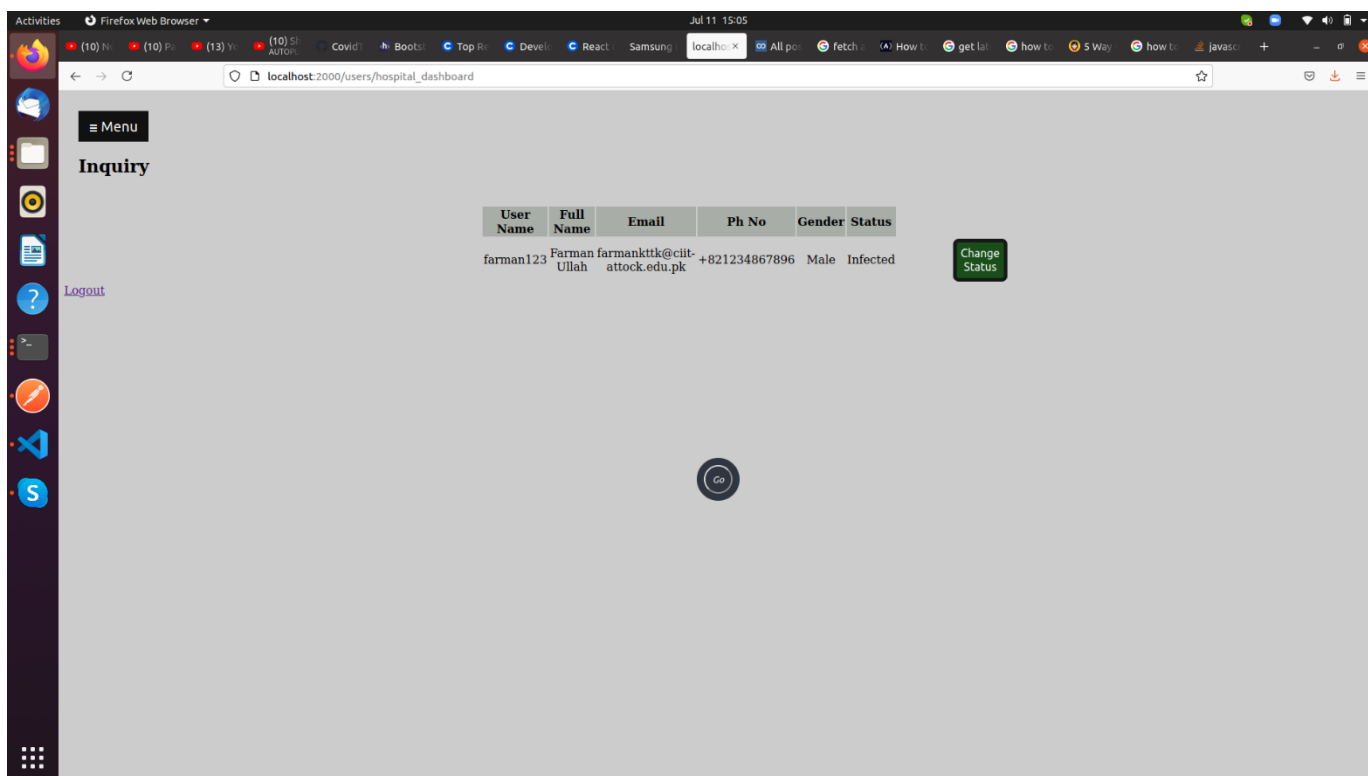


Figure 20: Changing the status of user.

test\_db/ubuntu@postgresql

Query Editor

Query History

Scratch Pad

1

SELECT \* FROM public.user\_status

2

Data Output

Explain

Messages

Notifications

	uname text	date date	time time without time zone	patient_beacon text	hospital_uid text	status text
1	TTrTriali	2021-07-08	07:45:43	511689569823659363	0	Normal
2	rafael	2021-07-08	08:19:26	33343065356430302d646663352d313165622d393233342d3166636564613031	0	Normal
3	alex123	2021-07-08	08:28:58	38386436306336302d646663362d313165622d393233342d3835316239356138	0	Normal
4	farman123	2021-07-09	15:13:20	33313736346535302d653063382d313165622d393233342d3262316336646232	0	Normal
5	farman123	2021-07-11	01:09:06	33313736346535302d653063382d313165622d393233342d3262316336646232	ali	Infected
6	wasay123	2021-07-11	09:11:33	66626334396232302d653232372d3131	0	Normal
7	harry12	2021-07-11	09:13:36	34366562376636302d653232382d3131	0	Normal
8	wasay123	2021-07-11	18:33:19	66626334396232302d653232372d3131	ali	Infected
9	wasay123	2021-07-11	19:50:52	66626334396232302d653232372d3131	ali	Normal

Figure 21: Each user status table (Normal or Infected)

test\_db/ubuntu@postgresql

Query Editor

Query History

Scratch Pad

1

SELECT \* FROM public.hospital

2

ORDER BY uname ASC

Data Output

Explain

Messages

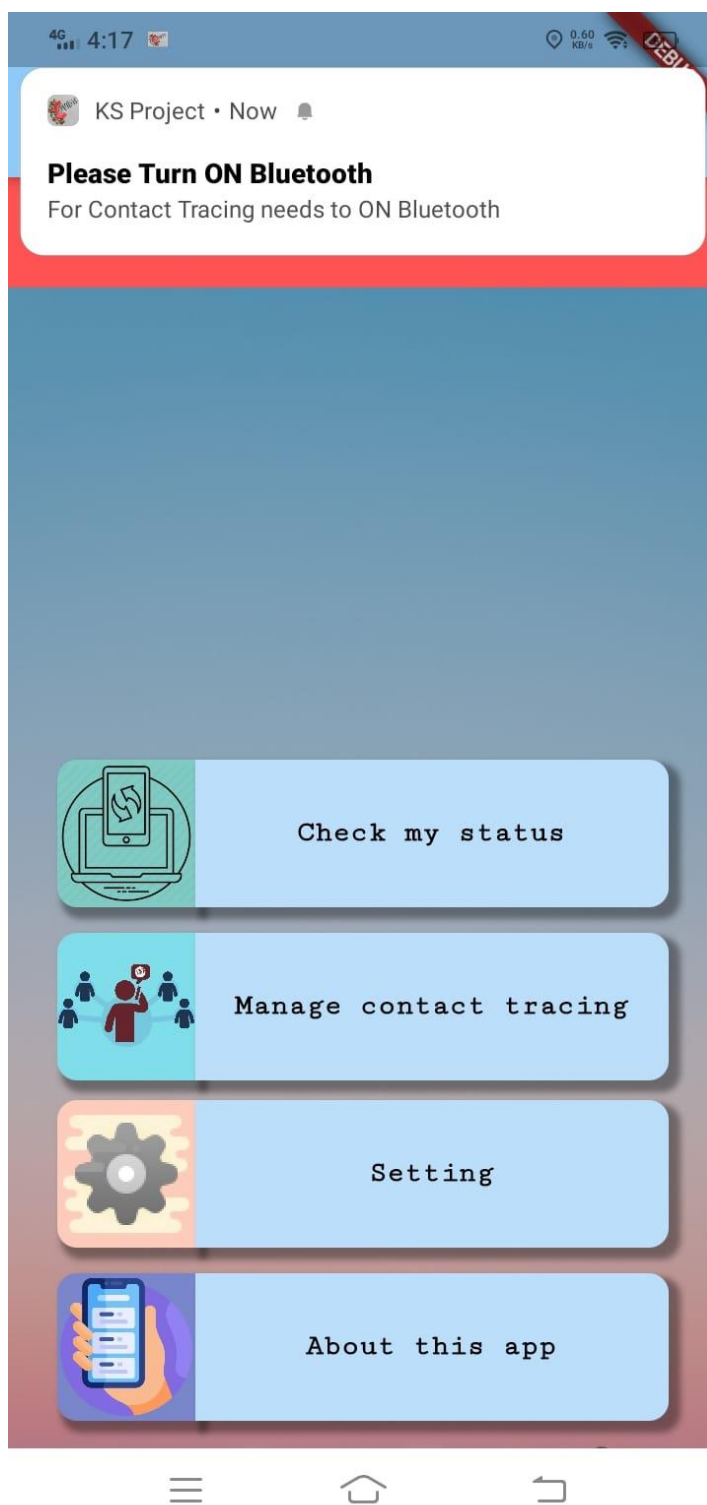
Notifications

	uname [PK] text	password text	email text	ph_no text	otp integer	verified boolean
1	ali	\$2b\$10\$YDs7mebqTvpUBbntdEKShuSPAm6KgJEnv/odyvVpqqdZP0qWb9USXG	fff@gmail.com	26595656	94329	true
2	furqan	\$2b\$10\$Mdkdj20.9XiLLvyJ1wtOOJBnsxywOZuDKFzTsPNyibfDgzcze/ZDm	f9500@gmail.com	33	71260	true
3	jack	\$2b\$10\$mxVONPJghKGnCO6uJul3xeg2N/ERxmzxJ0RFy/fXzWM2YPy./nX9e	furqan.ali9500@gmail.com	6966	23034	true
4	jamali	\$2b\$10\$ISV5btM2uidJmVpU7oWfw.klgKHE6/YRDht/Pujne6IDKBWd1MhHW	jamali@gmail.com	669	22254	false
5	kki	\$2b\$10\$BKLMQvpeC66DtkWnJ8wTCu5P4F19BO6.LJosJx3m4i5N3dEmmpUkO	furqan.ali9500@gmail.com	896	20451	true

Figure 22: Register Hospitals table.

### 3.2.4 Alert generation on user device:

KS mobile application generate alerts in case of someone contact found and app not working. App is not working perfectly in case of Bluetooth is off. App check continuously, is Bluetooth enable or not in case of not enable app to generate a notification that please on the Bluetooth. As, all the contact tracing process is done with the help of Bluetooth technology, so device Bluetooth must be enabled. When someone is covid positive, system send an alert message to all devices that are in the contact of covid positive person. There are two types of alerts messages one is automatically generating alert and second one is manually with the help of check my status button. In the manual case user open and press the check my status button and app notify user contact found or not.



*Figure 23: Notification generation when bluetooth is off.*

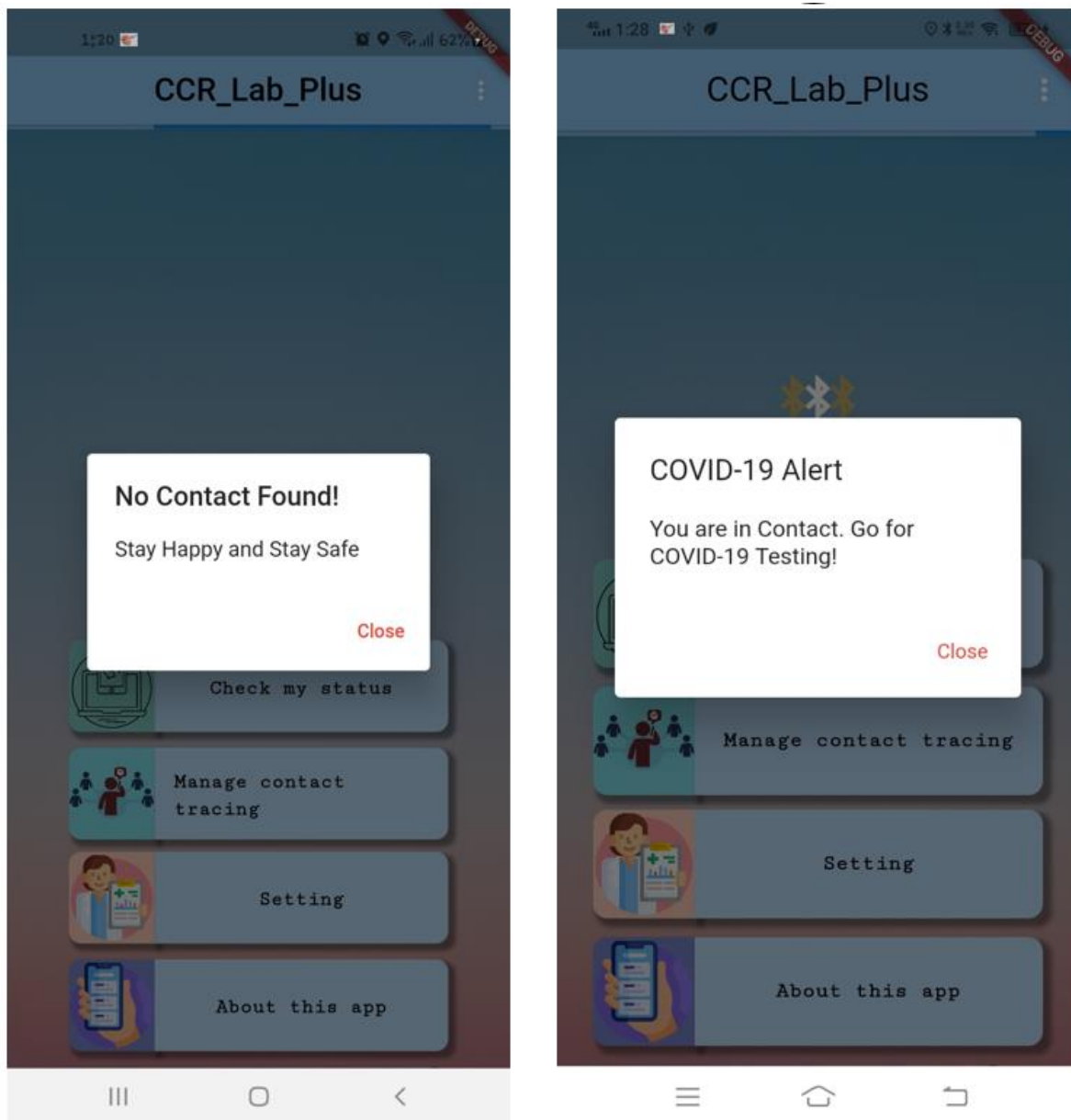


Figure 24: Alert notification generation when someone press check my status.

### 3.2.5 Geo-Fencing:

The data of GPS is storing on the server in a table named sensor data. That data includes the latitude and longitude information for each unique user. Using the AP's, the data is taken from the server for each user separately. First, all the markers are plot on map for just one user. With a unique color. Then it is also important to trace the user, so the line is plot form the first position to the last. Similarly, the same process is do for each user. The algorithm collects all the location from the server the user visited. May be there will be other user's data in his/her data, but the algorithm will collect only for this user. Then do for all the users.



### 3.2.6 Red alert area marking on map:

**Alert based on Patient:** If there is patient in the data so the region around that user will become the red alert region. The other normal user distance will be calculated based on Haversine formula. If there are any users whose distance less than 1.5km the alert is generated.

**Alert based on Red Alert region:** If there is any restaurant and the AP find any patient come to here so that region will become the red alert region. Now it is important to find the distance of each user who come close to this region in 1.5km. Also, the alert is generated to those who visited that region.

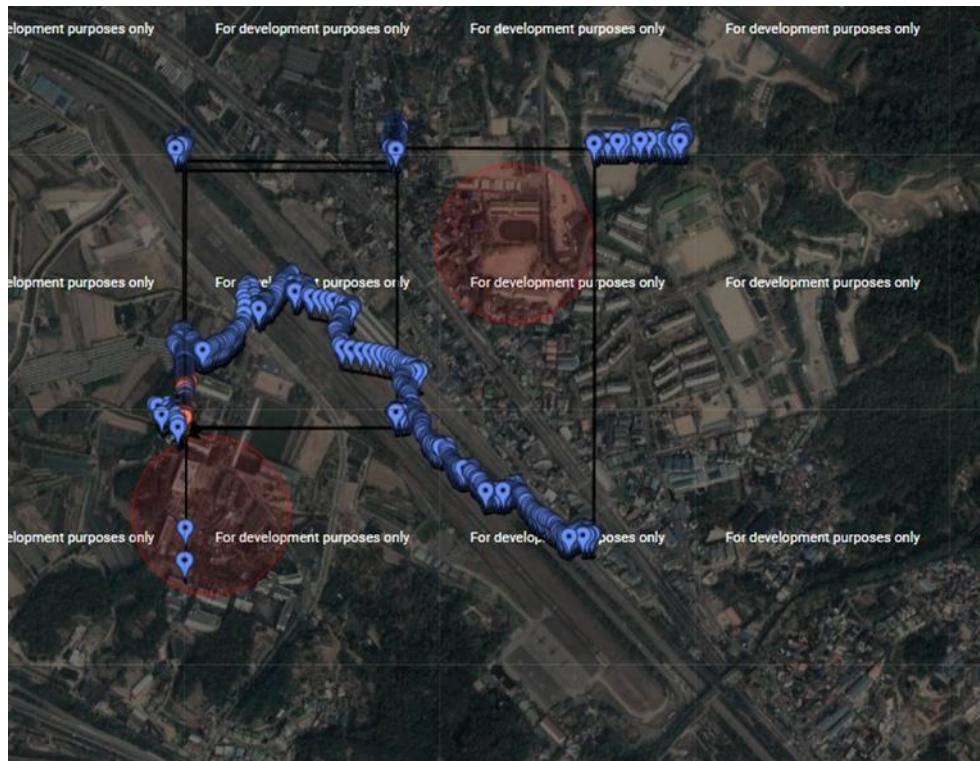


Figure 25: User traces on the map base on GPS data.

### 3.2.7 Tree Construction:

For the tree construction, the user interaction table will be used. But for this case the tree will be plot based on patient. Because first the patient UUID will come and then will check according to that id. First, all the patient ids will be taken from the table that include all the statuses of the user so every person having latest status with infected as patient will collect all those ids. Then took the first patient id and will check how many other users relate to this id. That will be plot and this is called the first layer of the tree for the first patient. If there are two user U1 and U2 interacted with patient P1 it will be plot. Now to find the other user who are in contact with U1 will find and will be plot under the U1. Similarly, for U2 and this one will be second layer of the tree. The processes will go on. But this will stop either the data is iterated completely, or the user decided number of layer (depth).



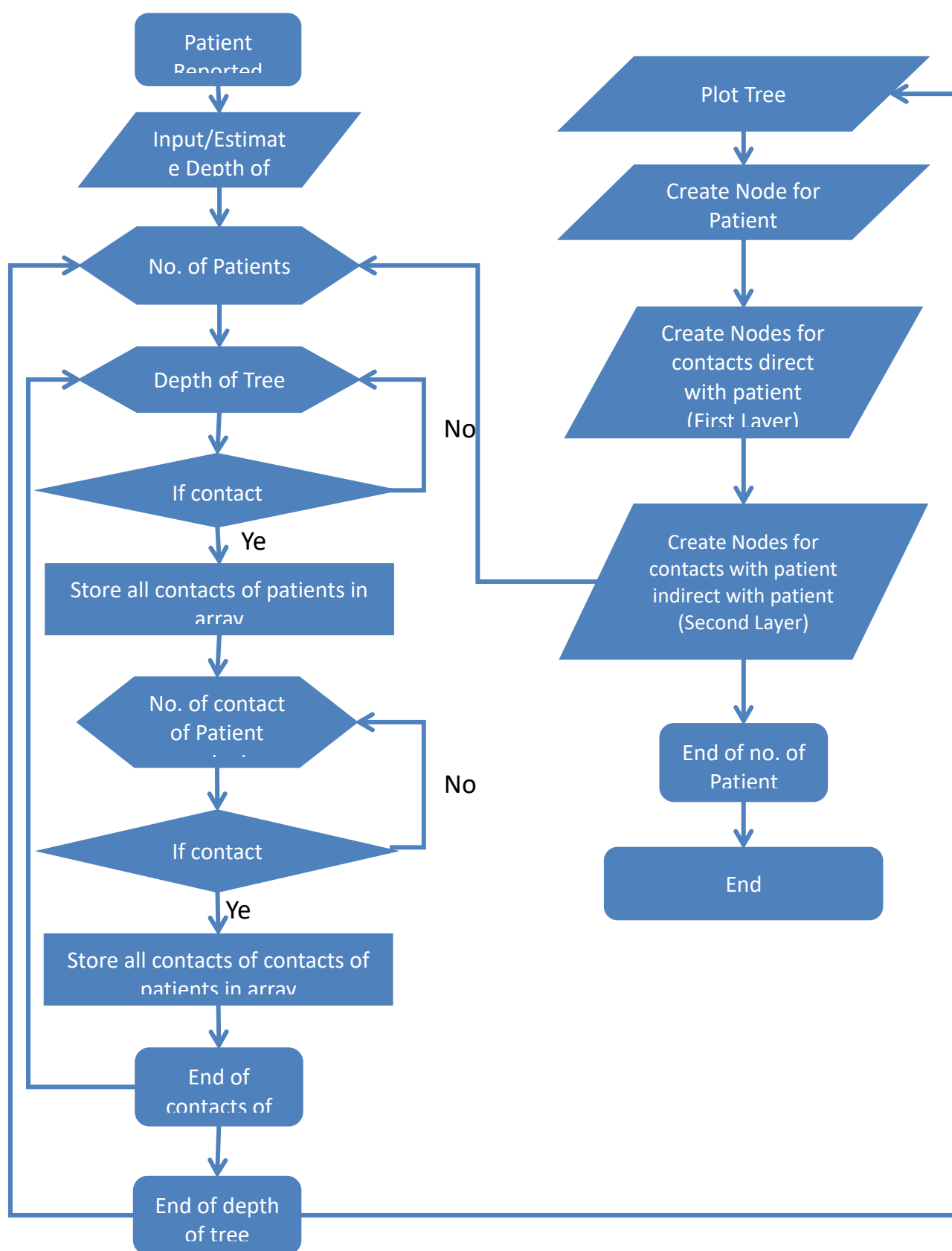
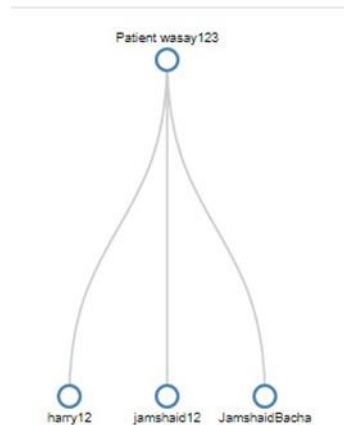


Figure 26: Flow chart of the tree construction.



*Figure 27: Tree on the basses of patient contact*

### 3.2.8 Network Construction:

This field will explain about how the network link is create when there is interaction between users. First, when the users install our mobile app, they will register through form and the user data will be stored on the server securely. When the user is registering during this time, a unique key is generated for the user which is called UUID. That unique id also stored with the user information. On the server there is one table which we can say interaction table. The data come to this table will be used for network creation. When all the users start moving around the city or the whole country the UUID's will store in a csv file in the user internal storage when they come in contact. That data will be stored on the server in the interaction table.

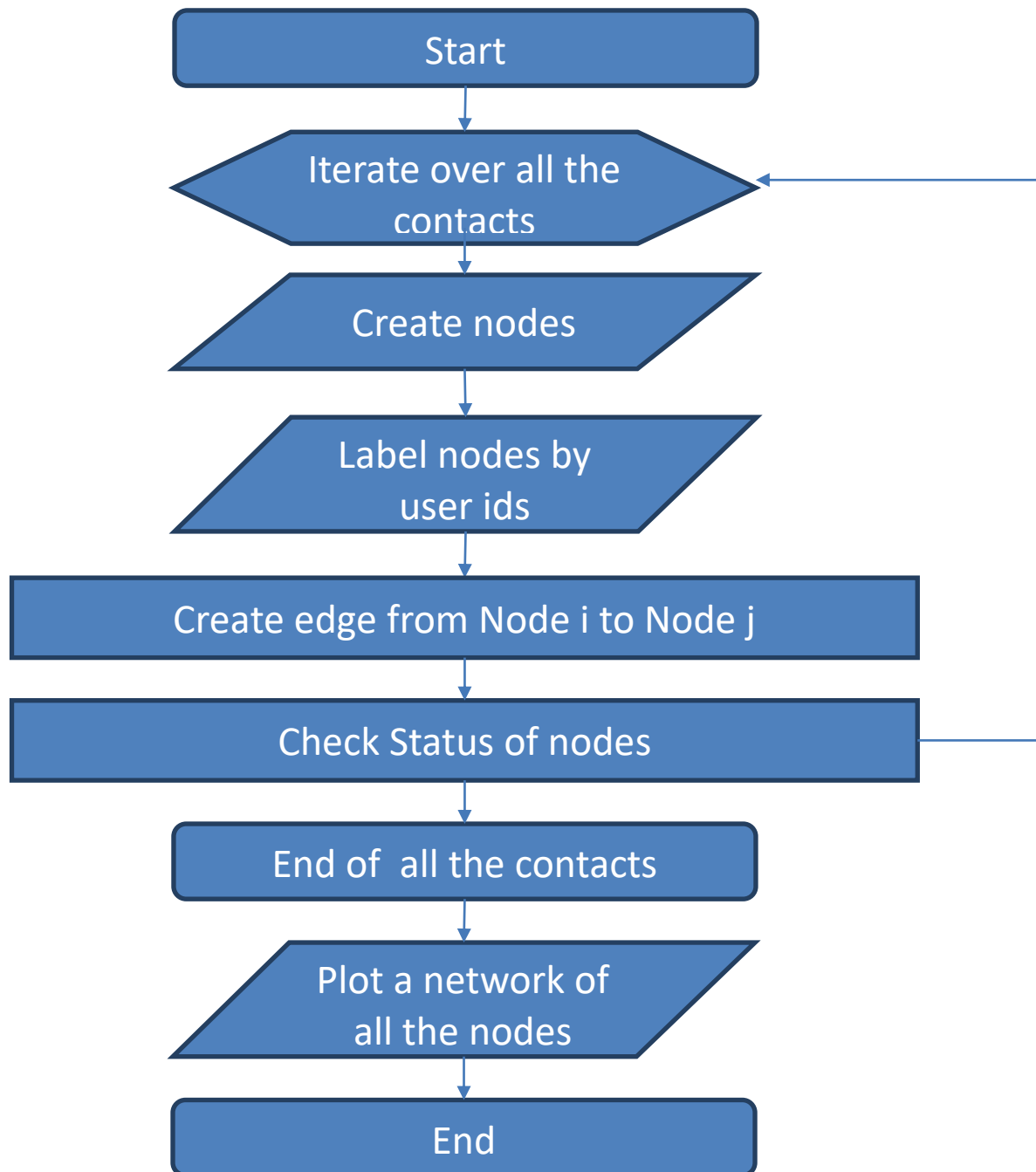
So, for the network plotting on the website side will use that interaction table. From the interaction table first, it will find who relates to whom. The nodes will create for each user UUID first. The node by default color normal user (Not a Patient). The node will label according to the user UUID username. Will create a link between all the nodes who relate to each other.

Now, it is also important that the user is normal, suspected or patient. There are three cases so for each different color is used. For normal user the node will be green, for suspected the node will be yellow and for patient the node will be red. The status information will be fetching from the server table name status table.

For those who are close enough to the user who are infected the status will be find based on TIE strength. The tie strength will be calculated from the distance they both were interacted. The relationship between tie strength and distance is inverse. The less distance the more will be the tie strength. The second scenario for the tie strength is based on time for how long they were in contact. If the time is

more between the normal user and the patient the tie strength will be increase.

When the user who was infected so if he/she recover so the status will change from the hospital according to that the network will update and the color of the node will be changed.



*Figure 28: Flow chart of the network construction.*

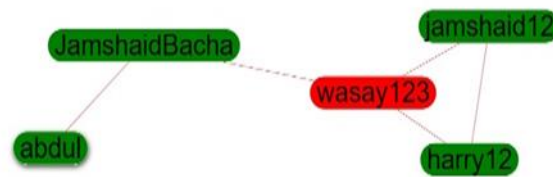


Figure 29: Network of user's interaction

### 3.2.9 Access Point:

Access point (AP) is the device that helpful in the collection of covid contacts from a specific venue. AP is the python-based web application implemented on Raspberry PI. AP is work same as mobile application, scanning and transmission beacon id and store data locally in a Raspberry PI and send data to server. Overall view of the AP on the bases of their working is shown in figure 30.

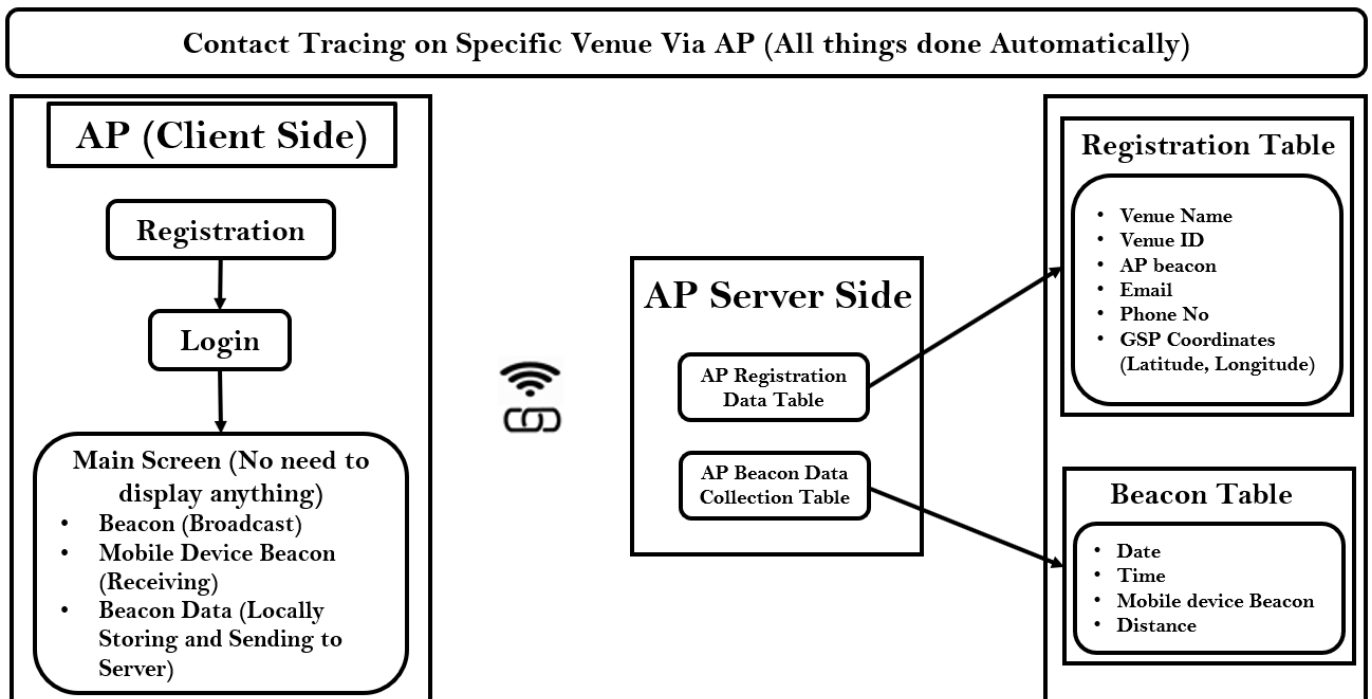


Figure 30: Overall view of the AP working

On AP side, first need to register AP. When register AP, system collect information about AP: AP Location, Venue id, AP register username, User email address, Phone number and Password.

localhost:8501

Menu  
Signup

## AP KS Project

Venue Address

Venue ID

Full Name

Email

Phone Number

Password

o": "823068800261", "lat": "37.5112", "long": "126.97489999999999", "otp": "27535", "status": 200}'

Figure 31: AP Registration Page

Menu  
Login

## AP KS Project

Venue ID

Password

Figure 32: AP Login Page

test\_db/ubuntu@postgres

Query Editor

Query History

Scratch Pad

1 SELECT \* FROM public.venue\_register

2 ORDER BY id ASC

Data Output

Explain

Messages

Notifications

	id	ven_f_name	ven_id	password	ven_beacon	pname	email	ph_no	lat	long	otp
	[PK] big	text	text	text	text	text	text	text	text	text	bigint
46	46	CCR Lab KAU Korea	CCR1234	\$2b\$10\$MCMdT9...	0d85902f-ed29-11eb-ae97-20cf30326da4	Creative Convergence Lab	wasay@kau.kr	823068800261	37.5112	126.97409999999999	27535

Figure 33: AP registration table on server

Overall functionality of AP is shown in the figure 34.

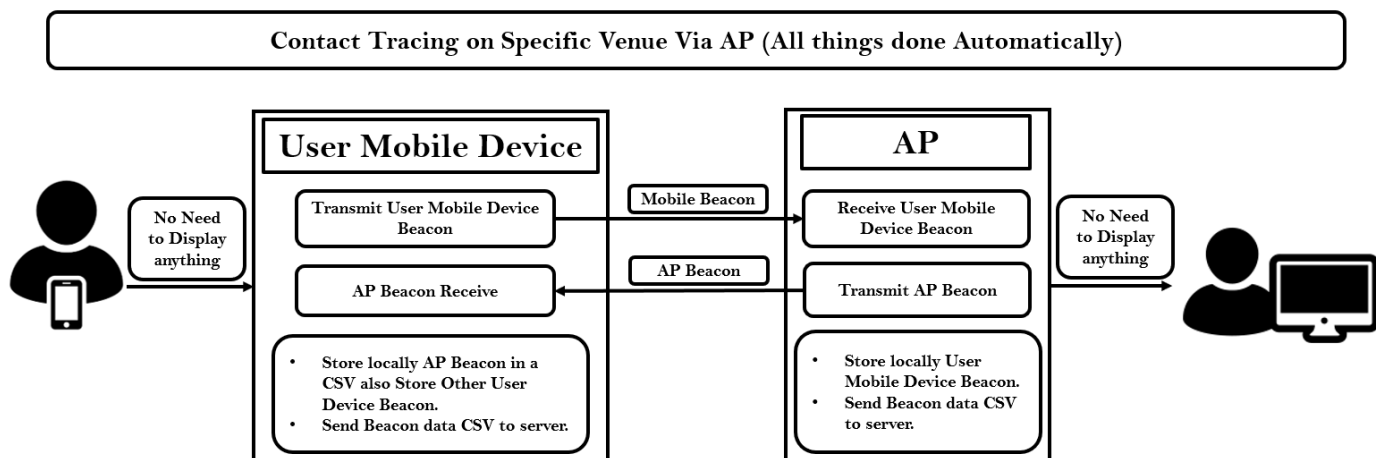


Figure 34: Overall functionality(working) of AP