COVID-19 PATIENT ALERT SYSTEM

Approach:

The project essentially consists of four parts-

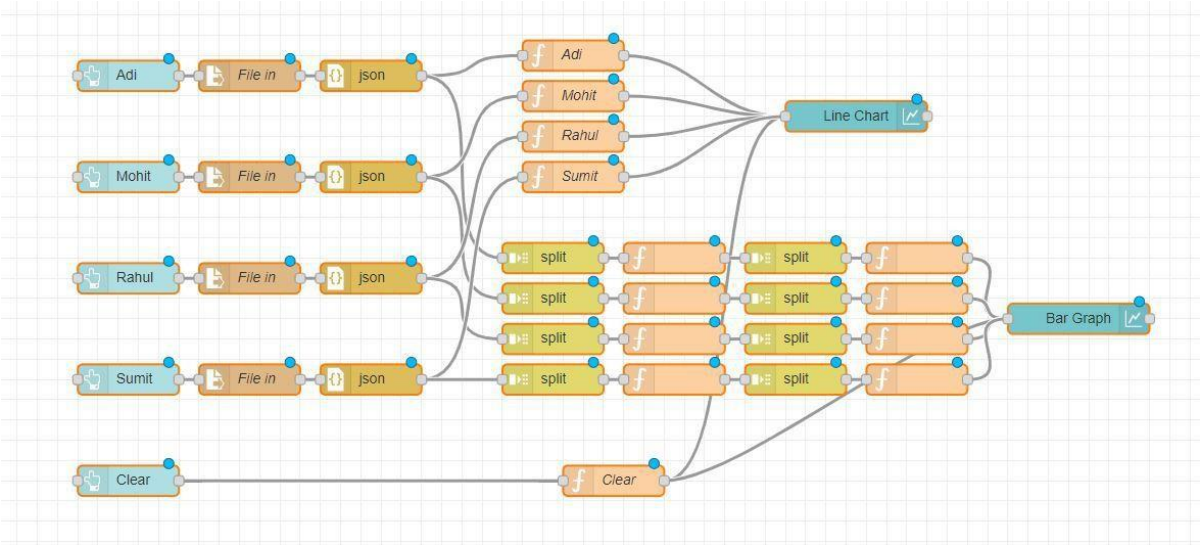
1. Designing a node-red flow to plot the graph from the provided database

The initial JSON file contained four folders each consisting of the temperature data of the four friends respectively. However, there was a trouble accessing the file in Node-red. So we copied and concatenated all of the them into one single file separated by commas. We have attached the drive link of the concatenated JSON file.

https://drive.google.com/file/d/1jJUR99Bu-_t29sf8R0glwk0BaezEvTDi/view

In the flow we insert the file using the file in node and then separate the four different objects using split node. Furthermore, we split individual temperature data at specific hours using another split node and feed it to the chart nodes which then plots the chart on the node-red dashboard.

1.a. NODE RED FLOW



1.b. Dashboard

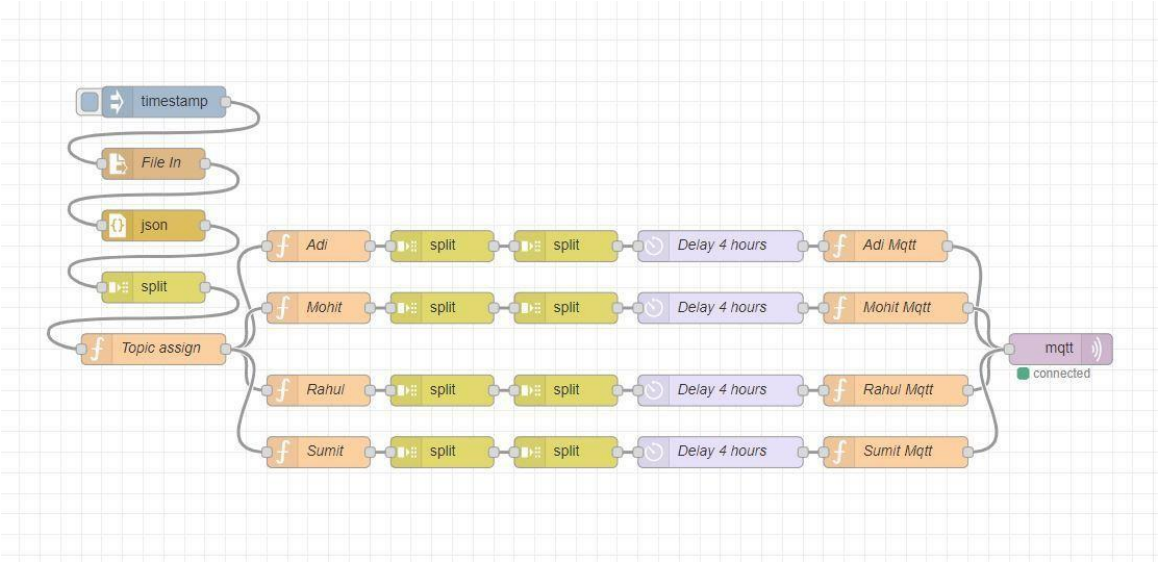
1. Uploading the data to AWS IoT Core through AWS MQTT broker

In a separate flow we split the data in similar way to access individual temperature data which was then uploaded in different topics using AWS MQTT broker.

1.a. NODE RED FLOW

We have added four function nodes (by the names Adi, Mohit, Rahul and Sumit) which will filter out the data of the one of the friends from the JSON file which contains all their temperature values. Again we used four more function nodes (by the names Adi Mqtt, Mohit Mqtt, Rahul Mqtt and Sumit Mqtt) to set different topics for each of their datas to be uploaded on to the AWS MQTT Broker.

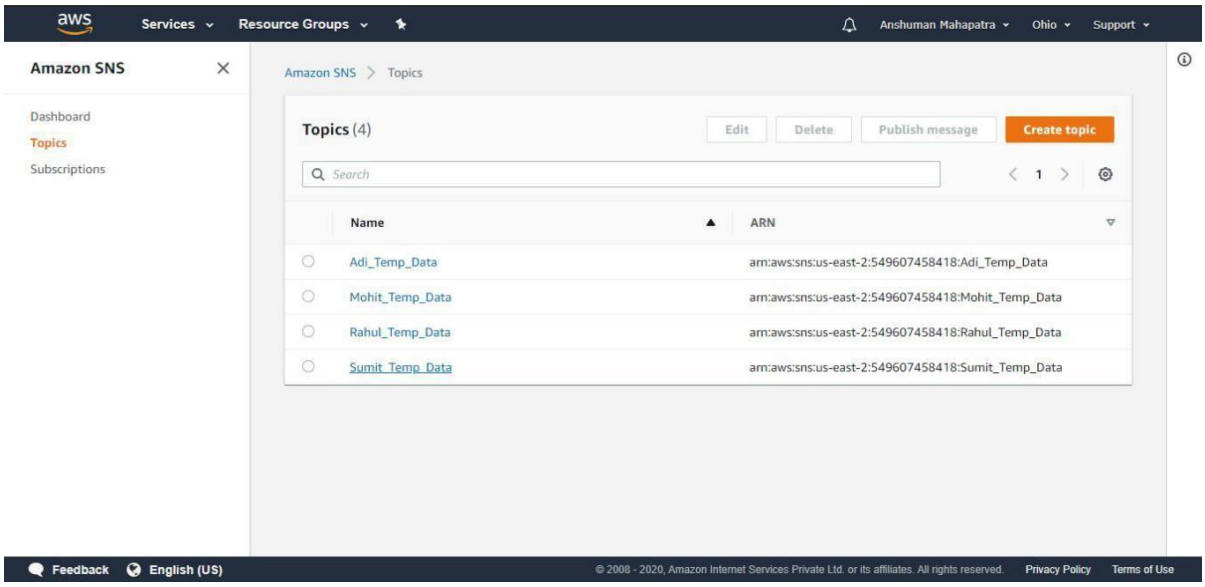
We have also inserted four delay nodes. These nodes prevent whole of the data to be uploaded at a time on to the MQTT Broker. The delay nodes put a time gap of four hours between two consecutive temperature.

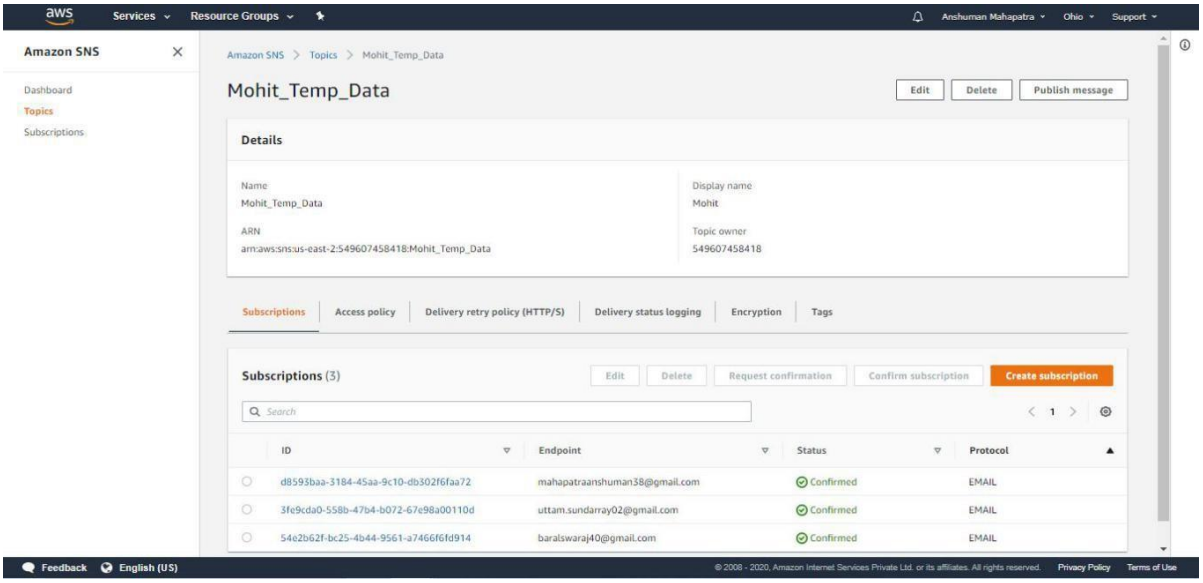


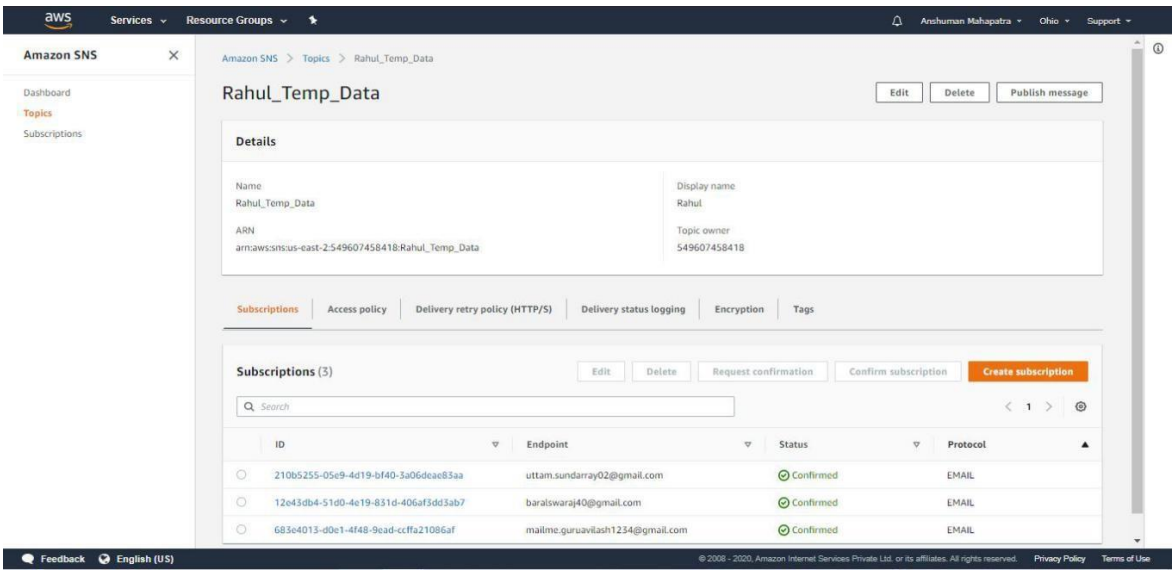
2. Sending alert mails when one of the four friends has a fever

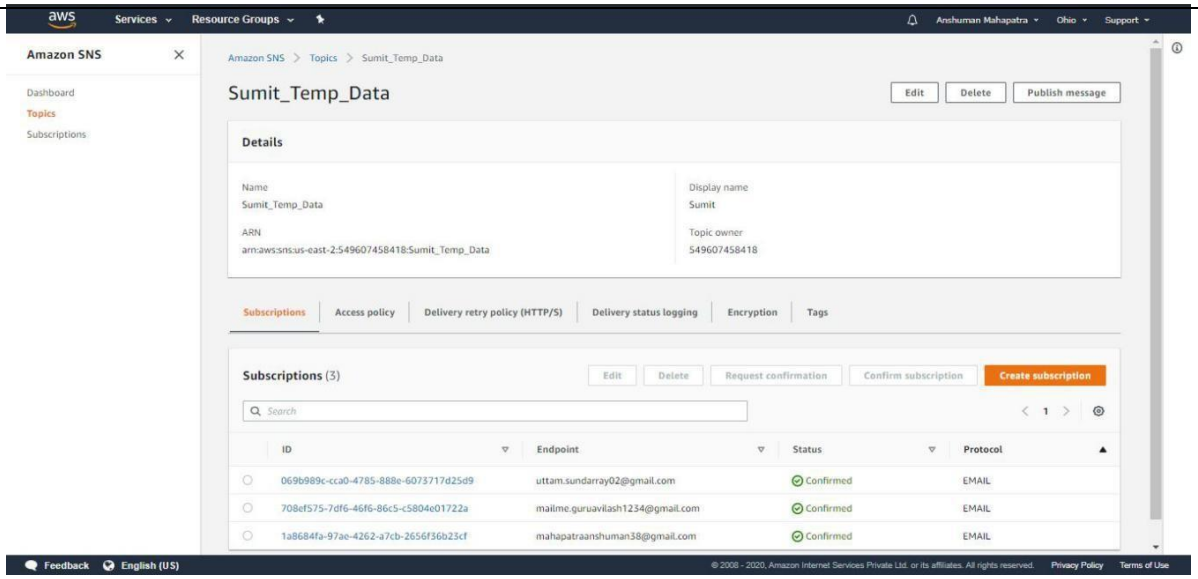
The task in this section was to send an alert mail to the other three friends whenever anyone of them had a high temperature. So we created four topics on Amazon SNS which were named after the four friends respectively. Each topic has to be subscribed by the other three friends.

2.a. AWS SNS Screenshots


Services
Resource Groups
Anshuman Mahapatra
Ohio
Support
Amazon SNS
Dashboard
Topics
Subscriptions
Amazon SNS > Topics
Topics (4)
Edit
Delete
Publish message
Create topic
Search
Name
ARN
Adi_Temp_Data
arn:aws:sns:us-east-2:549607458418:Adi_Temp_Data
Mohit_Temp_Data
arn:aws:sns:us-east-2:549607458418:Mohit_Temp_Data
Rahul_Temp_Data
arn:aws:sns:us-east-2:549607458418:Rahul_Temp_Data
Sumit_Temp_Data
arn:aws:sns:us-east-2:549607458418:Sumit_Temp_Data
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

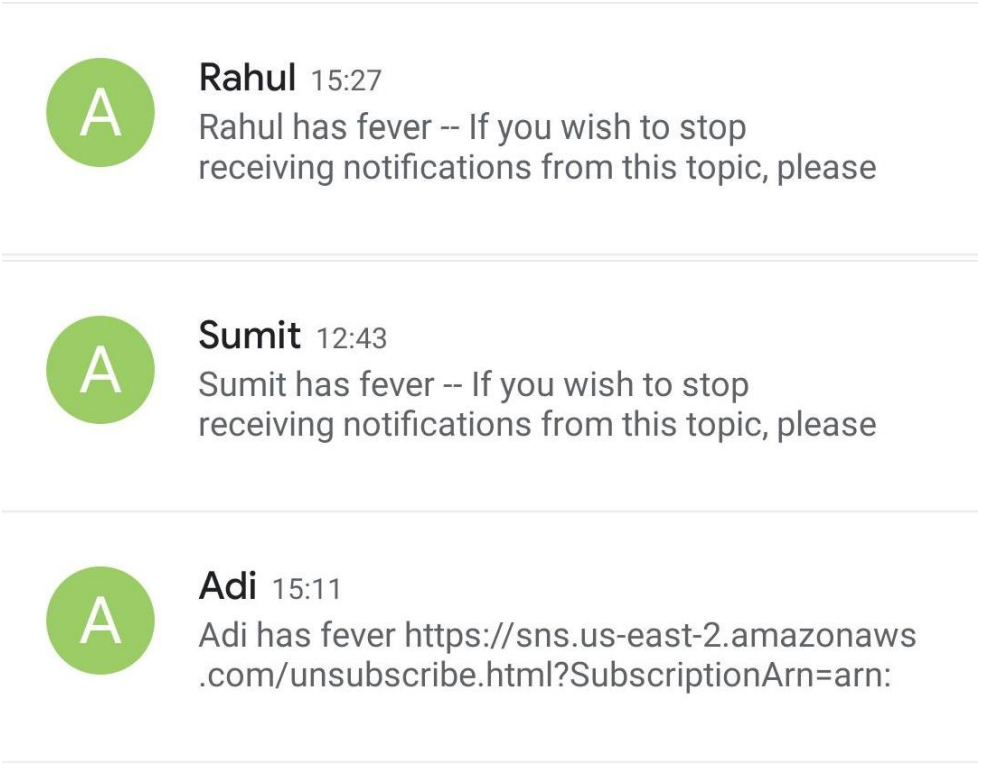

Services
Resource Groups
Anshuman Mahapatra
Ohio
Support
Amazon SNS
Dashboard
Topics
Subscriptions
Amazon SNS > Topics > Mohit_Temp_Data
Mohit_Temp_Data
Edit
Delete
Publish message
Details
Name
Mohit_Temp_Data
Display name
Mohit
ARN
arn:aws:sns:us-east-2:549607458418:Mohit_Temp_Data
Topic owner
549607458418
Subscriptions
Access policy
Delivery retry policy (HTTP/S)
Delivery status logging
Encryption
Tags
Subscriptions (3)
Edit
Delete
Request confirmation
Confirm subscription
Create subscription
Search
ID
Endpoint
Status
Protocol
d8593baa-3184-45aa-9c10-db302f6faa72
mahapatraanshuman38@gmail.com
Confirmed
EMAIL
3fe9cda0-558b-47b4-b072-67e98a00110d
uttam.sundarray02@gmail.com
Confirmed
EMAIL
54e2b62f-bc25-4b44-9561-a7466f6fd914
baralswaraj40@gmail.com
Confirmed
EMAIL
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


Services
Resource Groups
Anshuman Mahapatra
Ohio
Support
Amazon SNS
Dashboard
Topics
Subscriptions
Amazon SNS > Topics > Rahul_Temp_Data
Rahul_Temp_Data
Edit
Delete
Publish message
Details
Name
Rahul_Temp_Data
Display name
Rahul
ARN
arn:aws:sns:us-east-2:549607458418:Rahul_Temp_Data
Topic owner
549607458418
Subscriptions
Access policy
Delivery retry policy (HTTP/S)
Delivery status logging
Encryption
Tags
Subscriptions (3)
Edit
Delete
Request confirmation
Confirm subscription
Create subscription
Search
ID
Endpoint
Status
Protocol
210b5255-05e9-4d19-bf40-3a06deac83aa
uttam.sundarray02@gmail.com
Confirmed
EMAIL
12e43db4-51d0-4e19-831d-406af3dd3ab7
baralswaraj40@gmail.com
Confirmed
EMAIL
683e4013-d0e1-4f48-9ead-ccffa21086af
mailme.guruavilash1234@gmail.com
Confirmed
EMAIL
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use



So as to automate the mail system we wrote four python programs, which were to run in the background on each of the four friends' devices. The program subscribes to the MQTT topic of the user of the device. The program then compares the fetched temperature value of the user to the threshold(100°F). In case the fetched value exceeded the threshold an alert mail was sent to the subscribers of the user's SNS topic.

2.b. Email Confirmation Screenshots



3. Designing an app using MIT App Inventor to continuously save the user's data in a local database

3.a. LocSen (Location Sensor)

We created a simple android application using MIT App inventor that sends out the longitude and latitude of the user to a database. The chosen database here is a simple Google sheet file. The UI of the app is fairly simple consisting of a Current Geographical Co-ordinate display along with address and a list view that shows all the previously visited locations serially. The user can even choose to hide or show the list view according to his/her will.

1. The Layout : LocSen consists only a single screen that displays all the relevant data. The home screen contains a text label displaying the **current latitude** , **current longitude** and relevant **address** . Below the address label, there are two buttons. One to **Show** the location list and another to **Hide** the location list. Finally the screen ends with a **Listview**. The screen also contains some hidden components such as a **Location Sensor,** two **Web** nodes. One web node is used to write the data into the database and another is used to read the data from the database and feed it into the Listview.

2. The Code Blocks : On the coding side of the application, we have used various block to make sure the app functions as per the question demanded. The explanation of each block used are as follows :-

A) **when Screen1.Initialize** - We configured this block to direct the app to fetch data from the database the moment the app was turned on. We achieved this by joining a **set Web.Url to** block. This block was linked to the Google sheet database that fetched the data. The previous block was executed using a **call Web.Get** block.

B) **when LocationSensor1.LocationChanged** - The application of this block is pretty obvious from the name itself. This block tells application to perform certain tasks when the location sensor senses a different location using the user's phone's GPS. We programmed it to feed the location, latitude and address data to their respective text labels. Also, this node performed the most important task of the entire application, it sends the location data to the database. It does so by utilization of the Web_2_For_Sending_Data node. Without this node, the application looses its meaning.

C) **When Web_1_For_Reading.GotText** - This block is linked to the first web

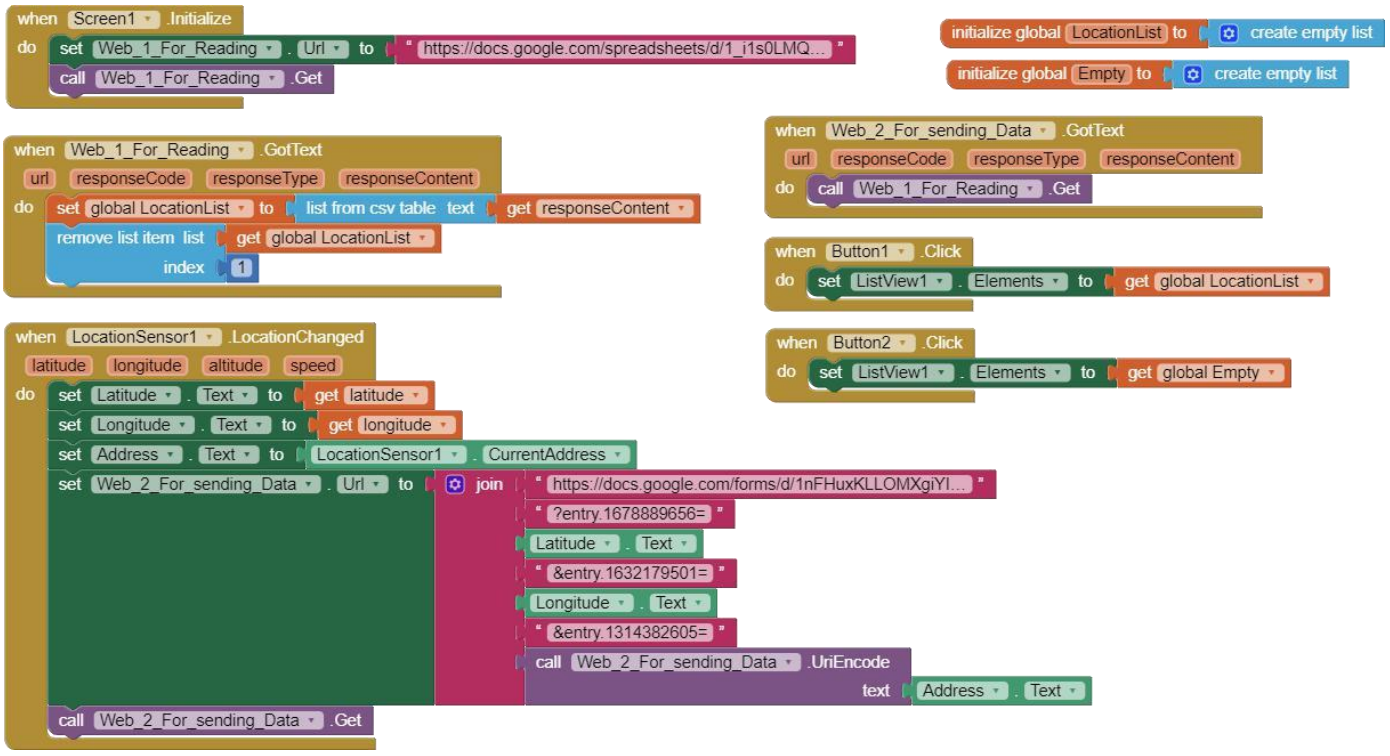
node, I.e, Web_1_For_Reading. By using this block we derived the data from the URL, I.e, the data from the Google sheet, and programmed it to feed the data into the Listview of the application. We also configured the node to remove the first column of the spread sheet that contains the field values.

D) **when Web_2_For_Sending_Data.GotText** - When the data is entered into the database, simultaneously the second URL is also executed to read the data and feed it into the List View. This is the only for using the when Web_2_For_Sending_Data.GotText node. It enables this function and lets us see the location in the list view and upload it to the database at the same time.

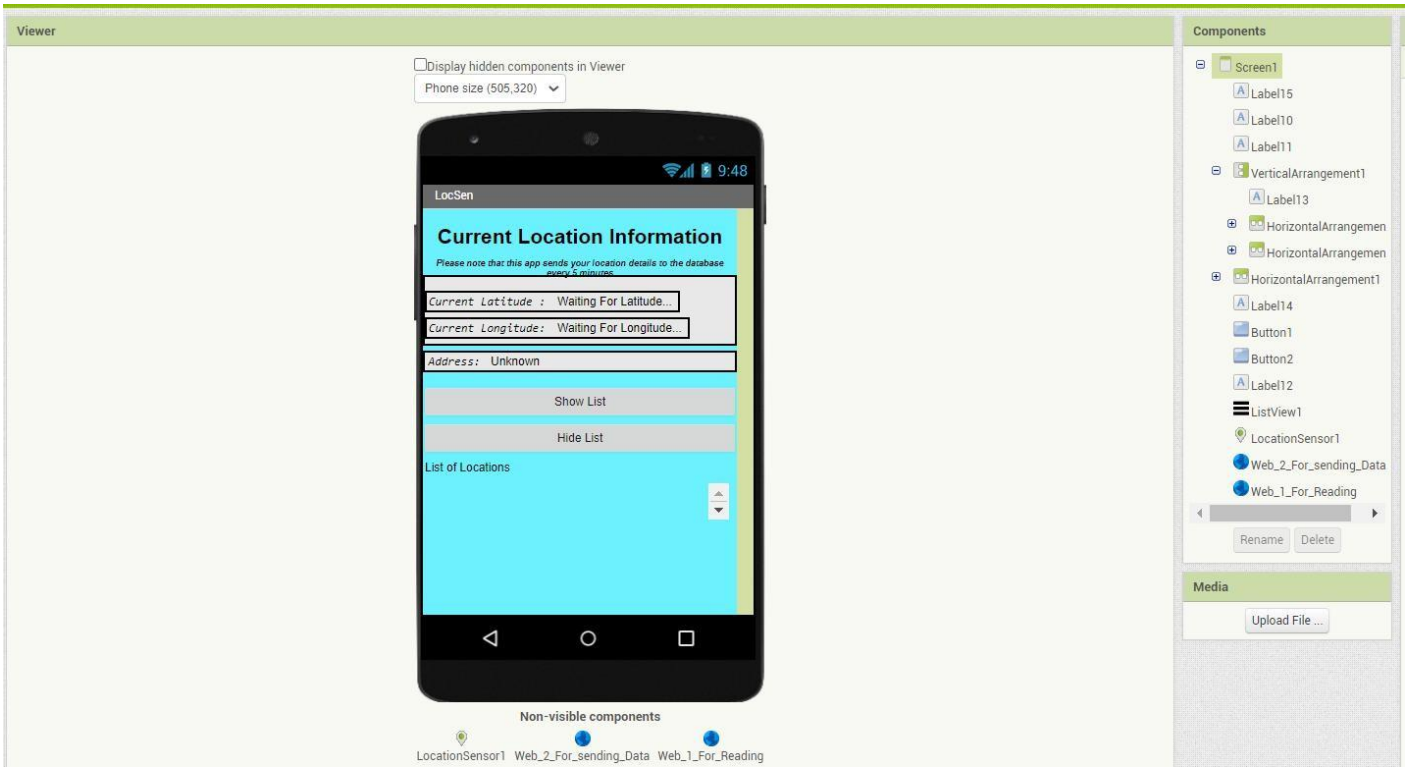
E) **Global Variables** - We have also used two Global variable nodes. The **LocationList** variable stores the incoming data from the Google sheet in it making it easier to perform actions on the data.The second variable **Empty** is used to store an empty list that will be used to hide the existing Location List.

F) Buttons - Finally we have used two button nodes to program the **Show List** and **Hide List** functions.

3.b. Blocks

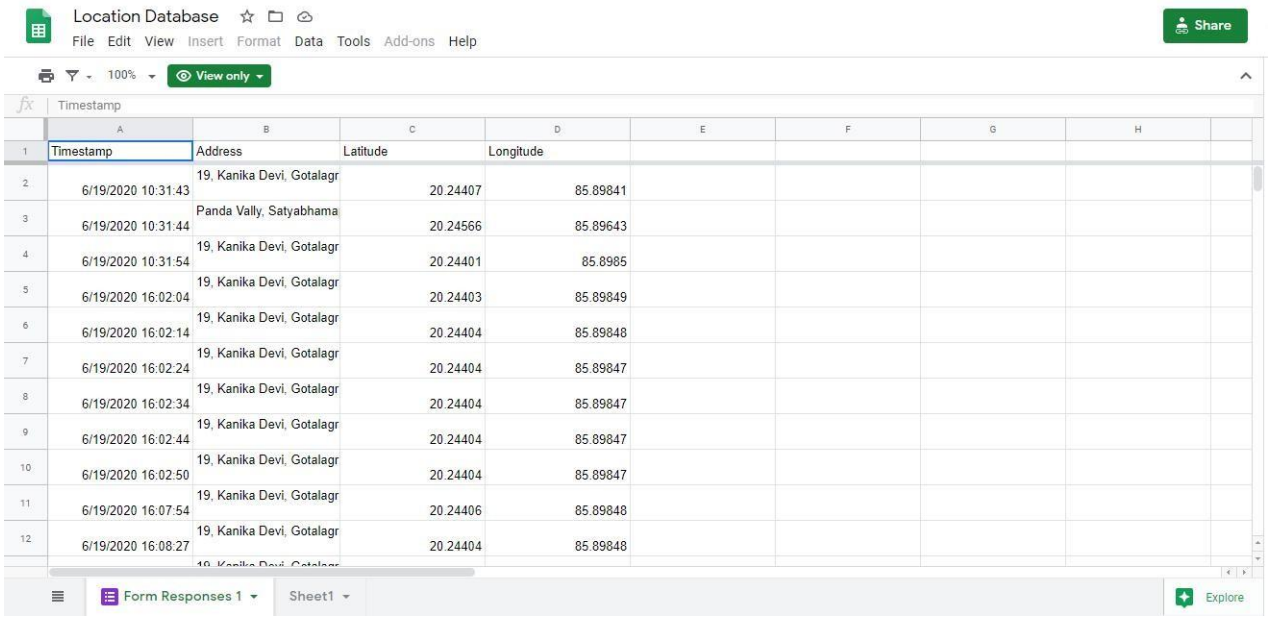


3.c. MIT App Interface Screenshot



3.d. Location Database- Screenshot and Link

The following is the link to where we are storing the location data in Google Sheets- https://docs.google.com/spreadsheets/d/1_i1s0LMQRLkj3eZzyfRPnuMki4yQYAqmvPaEg8E79yM/edit#gid=1 728332482



Timestamp	Address	Latitude	Longitude
6/19/2020 10:31:43	19, Kanika Devi, Gotalagr	20.24407	85.89841
6/19/2020 10:31:44	Panda Vally, Satyabhama	20.24566	85.89643
6/19/2020 10:31:54	19, Kanika Devi, Gotalagr	20.24401	85.8985
6/19/2020 16:02:04	19, Kanika Devi, Gotalagr	20.24403	85.89849
6/19/2020 16:02:14	19, Kanika Devi, Gotalagr	20.24404	85.89848
6/19/2020 16:02:24	19, Kanika Devi, Gotalagr	20.24404	85.89847
6/19/2020 16:02:34	19, Kanika Devi, Gotalagr	20.24404	85.89847
6/19/2020 16:02:44	19, Kanika Devi, Gotalagr	20.24404	85.89847
6/19/2020 16:02:50	19, Kanika Devi, Gotalagr	20.24404	85.89847
6/19/2020 16:07:54	19, Kanika Devi, Gotalagr	20.24406	85.89848
6/19/2020 16:08:27	19, Kanika Devi, Gotalagr	20.24404	85.89848