PUBLIC TRANSPORT OPTIMIZATION

Phase-4: Continue building the project by developing the real-time transit information platform. Use web development technologies to create a platform that displays real-time transit information. Design the platform to receive and display real-time location, ridership, and arrival time data from IoT sensors

Introduction:

In this document we explain about display the real-time location, arrival time data and ridership data of the Public Transport Vehicles using web development-based technologies such that **Firebase Cloud**, **MIT App Inventor, and Python Script** in public platform. In Phase-3 we showed how passenger detector and GPS tracking are works using Python script. In Phase-4 will show how data's send to the cloud and how reach the end user using **IoT Sensors.**

Connecting Wi-Fi:

```
#Connecting wifi
import network
print("Connecting to WiFi", end="")
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('Wokwi-GUEST', '')
while not sta_if.isconnected():
  print(".", end="")
  time.sleep(0.1)
print(" Connected!")
```

Above python code connect the ESP32 controller to Wi-Fi Modem to access the internet. After every event detected the controller will upload the data to the Cloud. (**Note**: Ensure **network** package available in controller)

Connect to Firebase Cloud:

```
#Importing firebase tools
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
```

These packages used to access **C**reate **R**ead **U**pdate **D**elete(**CRUD)** Operations on **Firebase Real Time Database.**

- **Initialize Firebase using Python:**

```
# Initialize Firebase Admin SDK
cred = credentials.Certificate('mitiot1-firebase-adminsdk-g6nw3-d317fcae89.json')
firebase_admin.initialize_app(cred,{'databaseURL':'https://mitiot1-default-rtdb.firebaseio.com/'})
bus_id=db.reference('gps_details/bus_gps/91V')
```

db.reference method create or access the tags in cloud.

- **Read or Get the data from Firebase:**

```
#get user gps data from firebase
la=db.reference('gps_details/user_gps/latitude_user').get()
lo=db.reference('gps_details/user_gps/longitude_user').get()
lat1 = float(la)
lon1 = float(lo)
```

Above code get the user gps coordinates in desired tags and store to the variables using **get()** method. The tag value (value) is in string form. So the **float()** method covert string into float number for further calculations.

- **Send or Update data to Firebase:**

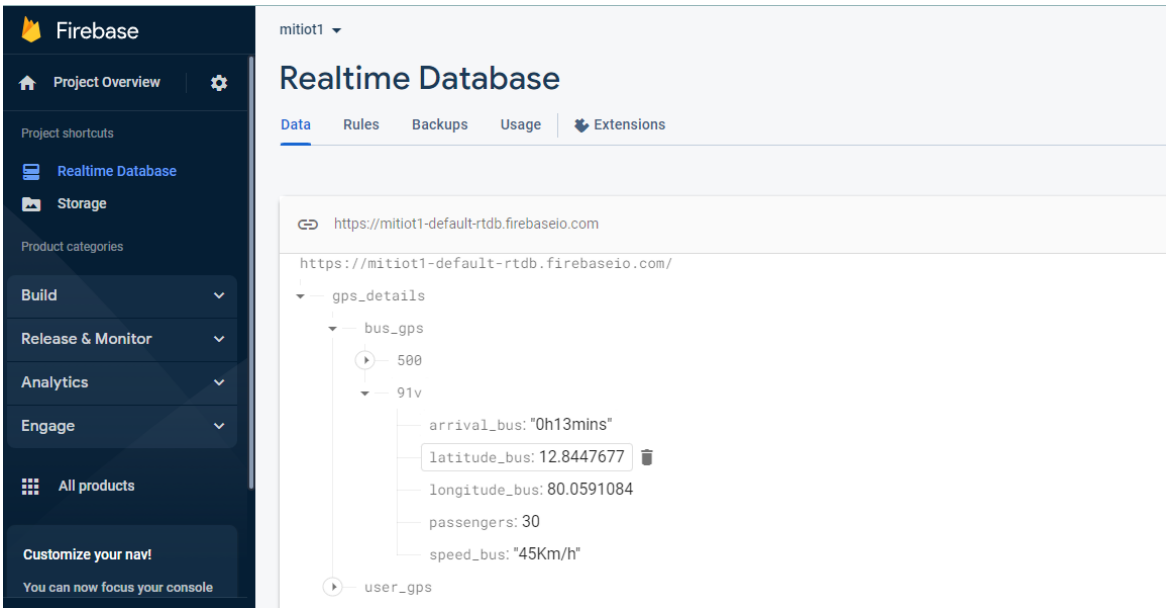
```
bus_id=db.reference('gps_details/bus_gps/91V')
bus_id.update()

data = {
     'latitude_bus' gps.latitude,
     'longitude_bus' :gps.longitude ,
     'speed_bus' : bus_speed,
     'arrival_bus': arrival_bus_time
    }

bus_id.update(data)
```

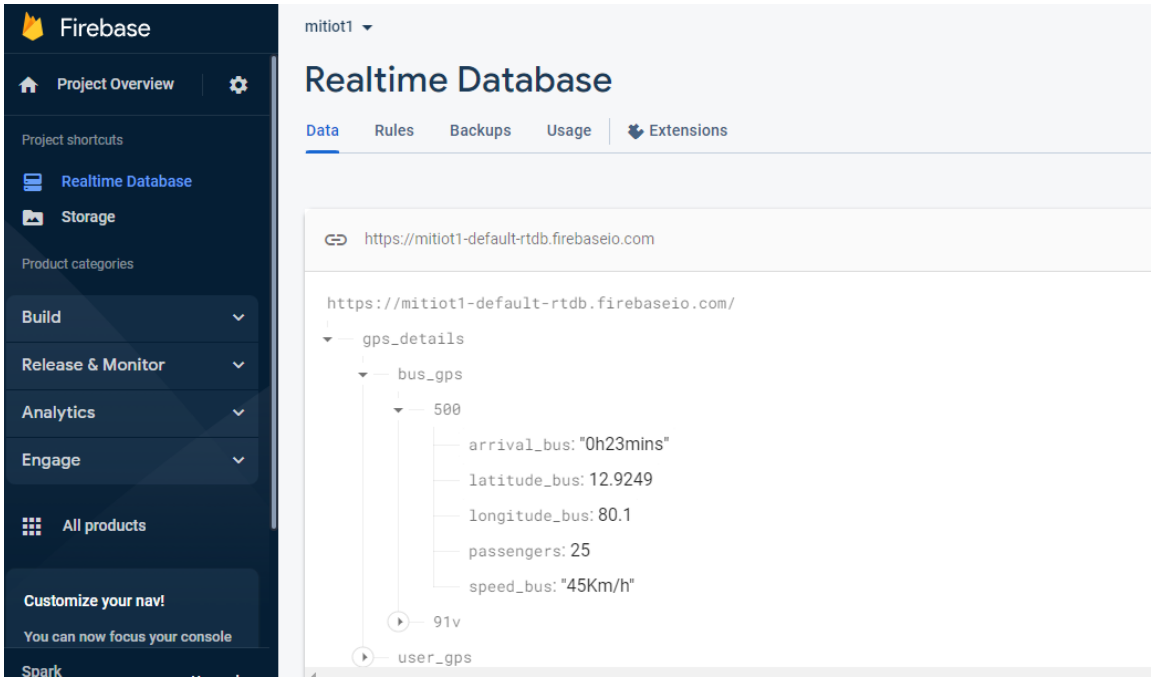
```
data={'passengers': Total}
bus_id.update(data)
```

The **update() or set()** methods used to write the data on Firebase cloud. The above Fig. show uploading GPS details and Passengers data's in public transport.

Firebase Collections:

In this Fig. shows example of how to store data's in cloud. We take two bus id (**91V,500)** for example. The firebase stores the data in the form of **{key : value}** or **{tag : value}** pair.

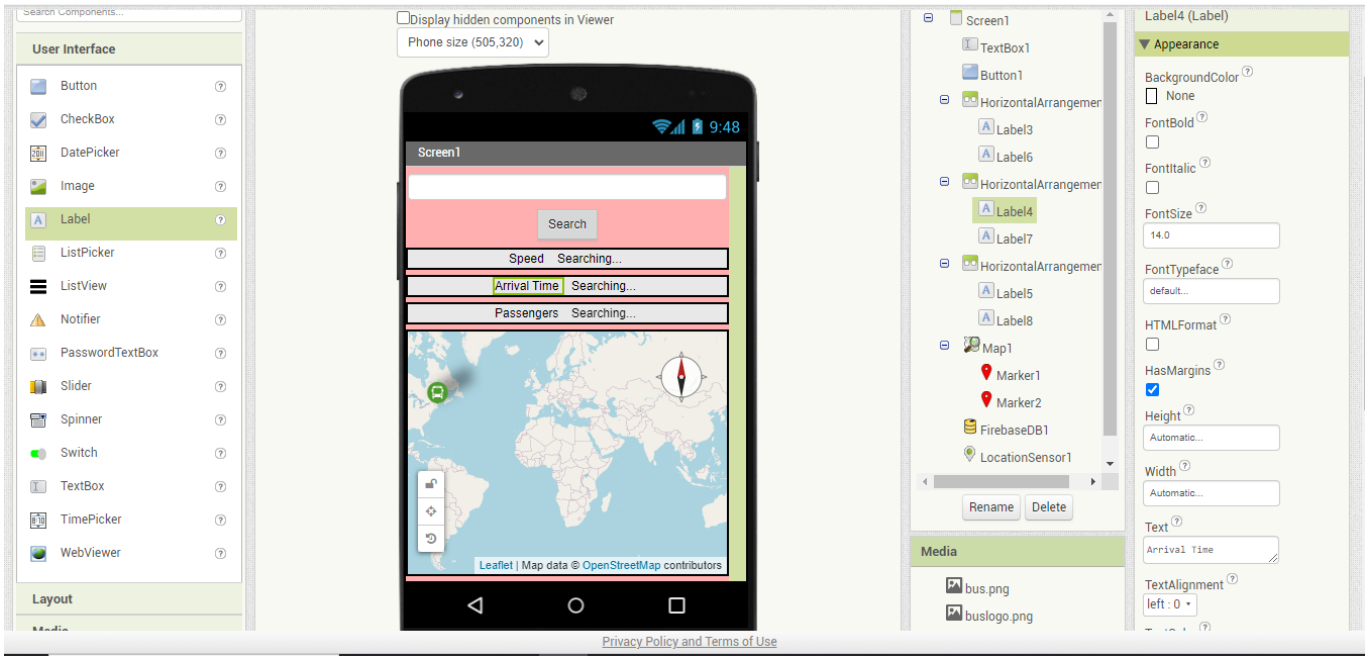




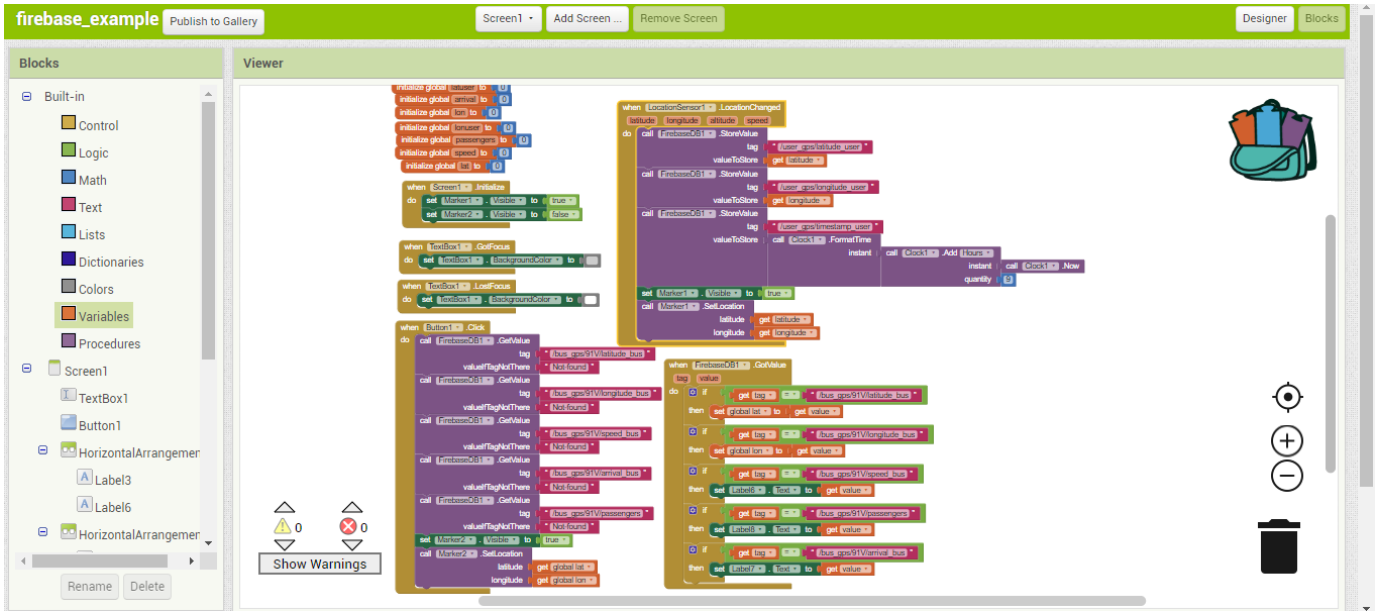
Public Platform Creation:

We used **MIT App Inventor** platform to create Bus Tracking mobile app. It is free platform to create apps without coding required.

- **Initialize the Screen (Design):**



- **Assign Functions to each block:**

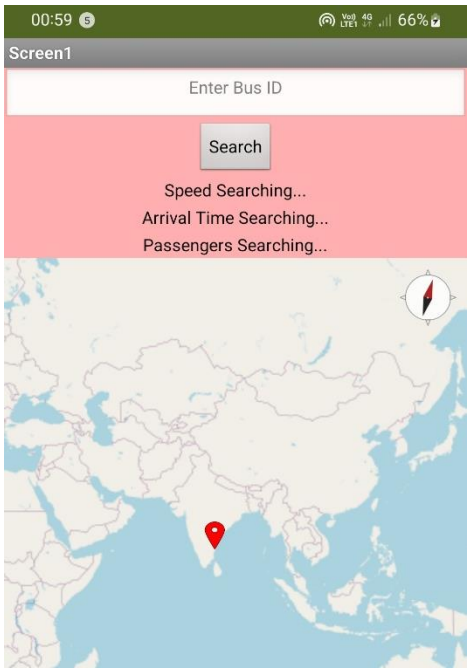


(We take Bus ID 91V as an example to show how output will project)

Output in User Mobile:

- **App initialize:**

  When app initialize only user location is shows in the screen.



- **After Enter the Bus ID and Press Search:**

  After enter the Bus ID real-time details of Bus will be project.