

SPRINT - 4

Date	13 NOV 2022
Team ID	PNT2022TMID05358
Project Name	Smart Waste Management System for Metropolitan Cities

1, Simulate python code in Python IDE software to transmit data to IBM Watson IOT platform

Python code:

smartbin.py:

```
#Project: Smart Waste Management System for Metropolitan cities  
#Team ID: PNT2022TMID05358
```

```
#Installing necessary libraries
```

```
import wiotp.sdk.device
```

```
import time
```

```
import random
```

```
import requests
```

```
import math
```

```
#Configuration details for connecting python script to IBM Watson IOT Platform
```

```
myConfig = {
```

```
"identity": {
```

```
"orgId": "mldk59",
```

```
"typeId": "pythoncode",
```

```
"deviceId": "252525"
```

```
},
```

```
"auth": {
```

```
"token": "QZqODYo6U*Q6b+IpuC"
```

```
} }
```

```
def myCommandCallback(cmd):
```

```
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
```

```
    m=cmd.data['command']
```

#Connecting the client to ibm watson iot platform

```
client = wiotp.sdk.device.DeviceClient(config=myConfig,logHandlers=None)
client.connect()
```

#Generate Random values for latitude, longitude in a circular distribution from the current location and

#alert the garbage collector to go to the particular location where the bin level and bin weight exceeds the threshold

while True:

```
    res = requests.get('https://ipinfo.io/')
    data = res.json()
    loc = data['loc'].split(',')
    theta = random.uniform(0,2*math.pi)
    area = (0.05**2)*math.pi
    radius = math.sqrt(random.uniform(0,area/math.pi))
    latitude,longitude = [float(loc[0])+radius*math.cos(theta), float(loc[1])+radius*
    math.sin(theta)]

    binlevel=random.randint(10,100)
    binweight = random.randint(50,1500)

    if binweight>=1000 and binlevel>80:
        myData={'latitude':latitude, 'longitude':longitude,'binlevel':binlevel,
                'binweight':binweight}
        client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=
0, onPublish=None)
        ##print("Published data Successfully: %s", myData)
        print("BIN IS FULL..TIME TO EMPTY IT!!!!\n",myData)
        client.commandCallback = myCommandCallback
        time.sleep(2)
        #break

    else :
        print("BIN IS IN NORMAL LEVEL...")
        time.sleep(2)
```

```
#Disconnect the client connection
client.disconnect()
```

Python IDE output:

```
C:\Users\prady\Desktop> cd C:\Users\prady\Desktop\Scripts && python smartbin.py (33/3)
File Edit Format Run Options Window Help

#Project: Smart Waste Management System for Metropolitan cities
#Team ID: FNT2022FMTD53567

#Installing necessary libraries
import wiotp.sdk.device

import time
import random
import requests
import math

#Configuration details for connecting python script to IBM Watson IoT Platform
myConfig = {
    "identityId": {},
    "orgId": "mldk559",
    "typeId": "pythoncode",
    "deviceId": "252525"
},
    "auth": {
        "token": "QGqoPto6U*Q6bIpoC"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data('command'))
    n=cmd.data['command']

#Connecting the client to ibm watson iot platform
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

#Generate Random values for latitude, longitude in a circular distribution from the current location and
#Alert the garbage collector to go to the particular location where the bin level and bin weight exceeds th

while True:
    res = requests.get('https://ipinfo.io/')
    data = res.json()
    loc = data['loc'].split(',')
    theta = random.uniform(0, 2*math.pi)
    area = (0.05+2)*math.pi
    radius = math.sqrt(random.uniform(0,area/math.pi))
    latitude,longitude = [float(loc[0])+radius*math.cos(theta), float(loc[1])+radius*math.sin(theta)]

    binlevel=random.randint(10,100)
    binweight = random.randint(50,1500)
```

2. Data is transferred to IBM Watson IoT platform.

IBM Platform output:

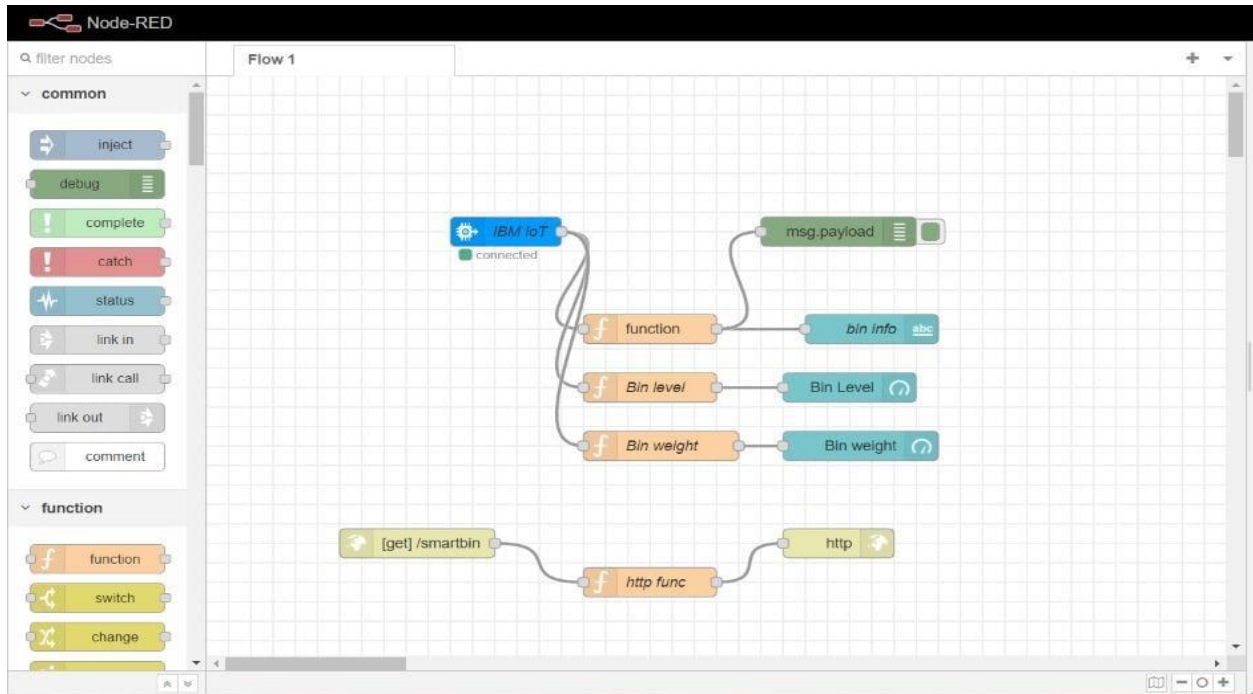
The screenshot displays the IBM Watson IoT Platform interface. At the top, the browser address bar shows the URL: `mldk59.internetofthings.ibmcloud.com/dashboard/devices/browse`. The page header includes the IBM Watson IoT Platform logo and a user profile for `2019ec0032@svce.ac.in` with ID `mldk59`. A navigation bar contains links for `Browse`, `Action`, `Device Types`, and `Interfaces`, along with an `Add Device` button.

The main content area shows a list of devices. The selected device is `252525`, which is `Connected` and uses the `pythoncode` device type. It was added on `Nov 5, 2022 8:24 PM`. Below the device list, the `Recent Events` tab is active, showing a stream of data events. The events table has columns for `Event`, `Value`, `Format`, and `Last Received`.

Event	Value	Format	Last Received
status	<code>{"latitude":93.38291147072071,"longitude":85....</code>	json	a few seconds ago
status	<code>{"latitude":123.34633147794314,"longitude":7...</code>	json	a few seconds ago
status	<code>{"latitude":120.69034946242466,"longitude":9...</code>	json	2 minutes ago
status	<code>{"latitude":82.92484862339958,"longitude":93....</code>	json	2 minutes ago
status	<code>{"latitude":44.682952261624024,"longitude":9...</code>	json	2 minutes ago

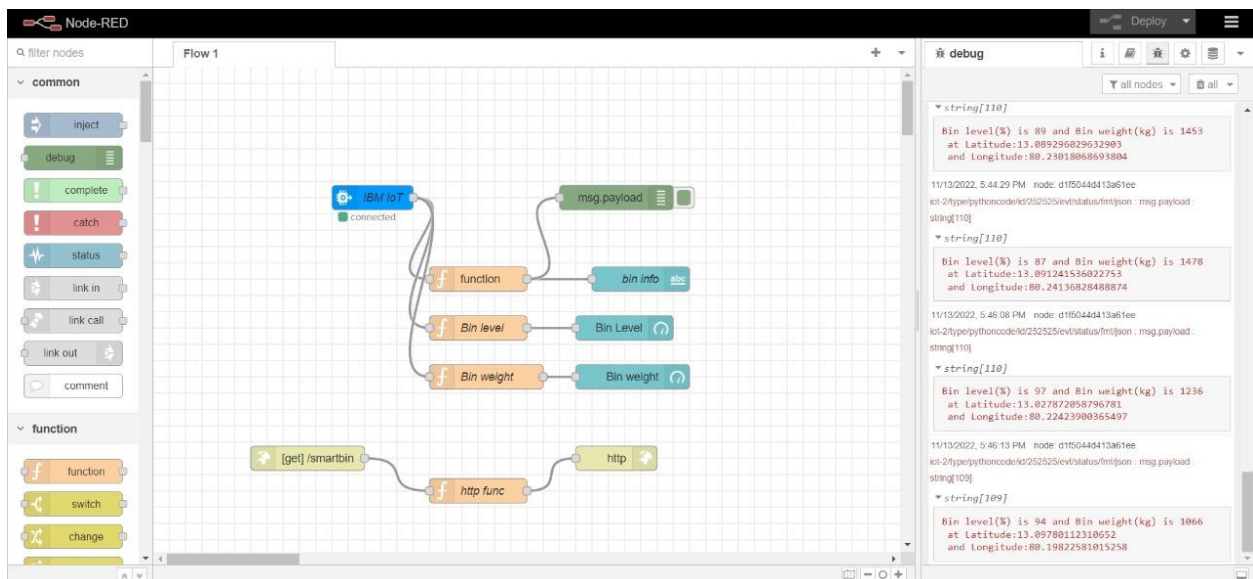
3. Data transfer from IBM Watson IOT platform and Python IDE to Node RED.

Node-RED:

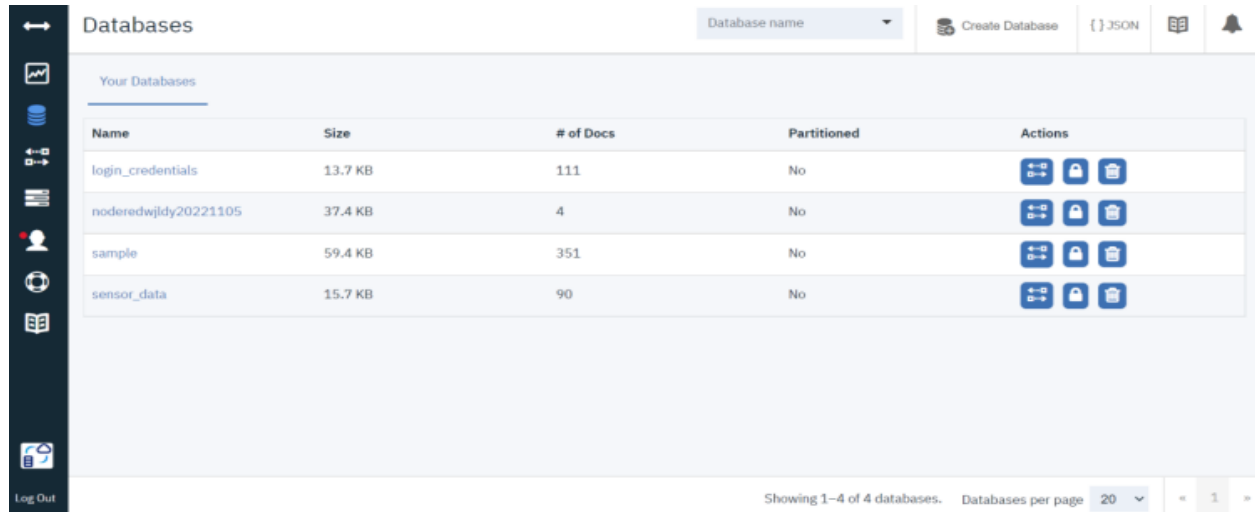


4. Node-RED Connection setup for data transmission from IBM Watson IoT platform to Node-RED dashboard and viewing in Web UI .

Node-RED:

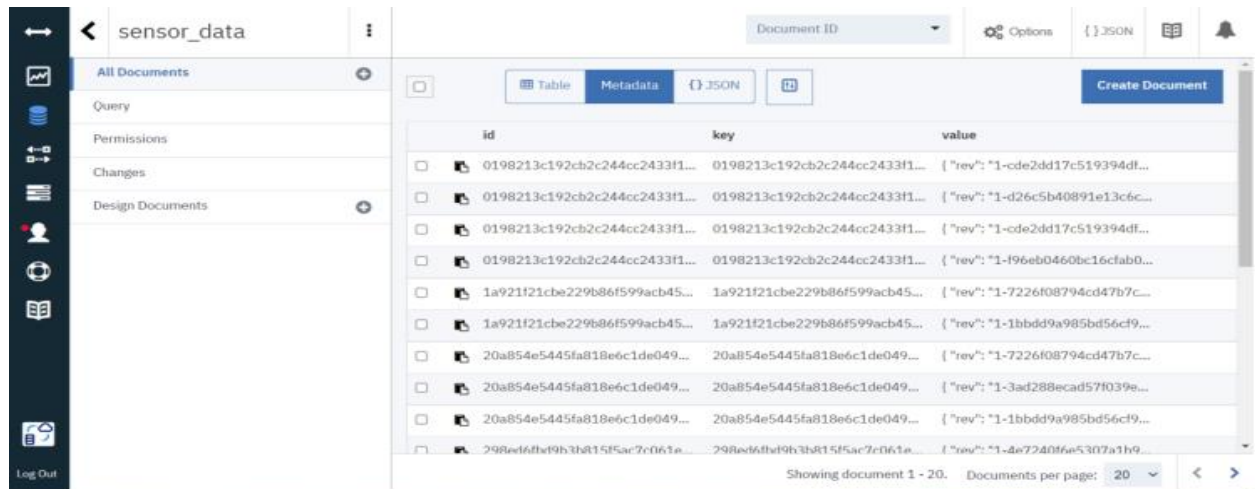


5. Storing database in IBM Cloudant DB



The screenshot shows the IBM Cloudant 'Databases' page. At the top, there's a search bar for 'Database name' and buttons for 'Create Database', 'JSON', and a notification bell. Below this is a table titled 'Your Databases' with columns: Name, Size, # of Docs, Partitioned, and Actions. The table lists four databases: 'login_credentials' (13.7 KB, 111 docs), 'noderedwjldy20221105' (37.4 KB, 4 docs), 'sample' (59.4 KB, 351 docs), and 'sensor_data' (15.7 KB, 90 docs). Each database has three action icons: a plus/minus icon, a lock icon, and a trash icon. At the bottom, it says 'Showing 1-4 of 4 databases. Databases per page: 20'.

Name	Size	# of Docs	Partitioned	Actions
login_credentials	13.7 KB	111	No	[+/-] [lock] [trash]
noderedwjldy20221105	37.4 KB	4	No	[+/-] [lock] [trash]
sample	59.4 KB	351	No	[+/-] [lock] [trash]
sensor_data	15.7 KB	90	No	[+/-] [lock] [trash]



The screenshot shows the 'sensor_data' database view in IBM Cloudant. The left sidebar has a 'Query' section with 'All Documents' selected. The main area shows a table with columns 'id', 'key', and 'value'. It lists 20 documents, each with a unique ID and a JSON value. At the bottom, it says 'Showing document 1 - 20. Documents per page: 20'.

id	key	value
0198213c192cb2c244cc2433f1...	0198213c192cb2c244cc2433f1...	{ "_rev": "1-cde2dd17c519394df..." }
0198213c192cb2c244cc2433f1...	0198213c192cb2c244cc2433f1...	{ "_rev": "1-d26c5b40891e13c6c..." }
0198213c192cb2c244cc2433f1...	0198213c192cb2c244cc2433f1...	{ "_rev": "1-cde2dd17c519394df..." }
0198213c192cb2c244cc2433f1...	0198213c192cb2c244cc2433f1...	{ "_rev": "1-f96eb0460bc16cfab0..." }
1a921f21cbe229b86f599acb45...	1a921f21cbe229b86f599acb45...	{ "_rev": "1-7226f08794cd47b7c..." }
1a921f21cbe229b86f599acb45...	1a921f21cbe229b86f599acb45...	{ "_rev": "1-1bbdd9a985bd56cf9..." }
20a854e5445fa818e6c1de049...	20a854e5445fa818e6c1de049...	{ "_rev": "1-7226f08794cd47b7c..." }
20a854e5445fa818e6c1de049...	20a854e5445fa818e6c1de049...	{ "_rev": "1-3ad288ecad57f039e..." }
20a854e5445fa818e6c1de049...	20a854e5445fa818e6c1de049...	{ "_rev": "1-1bbdd9a985bd56cf9..." }
298ed6f9b3b815f5ac7c061a...	298ed6f9b3b815f5ac7c061a...	{ "_rev": "1-4e7240f6e5307a1b9..." }

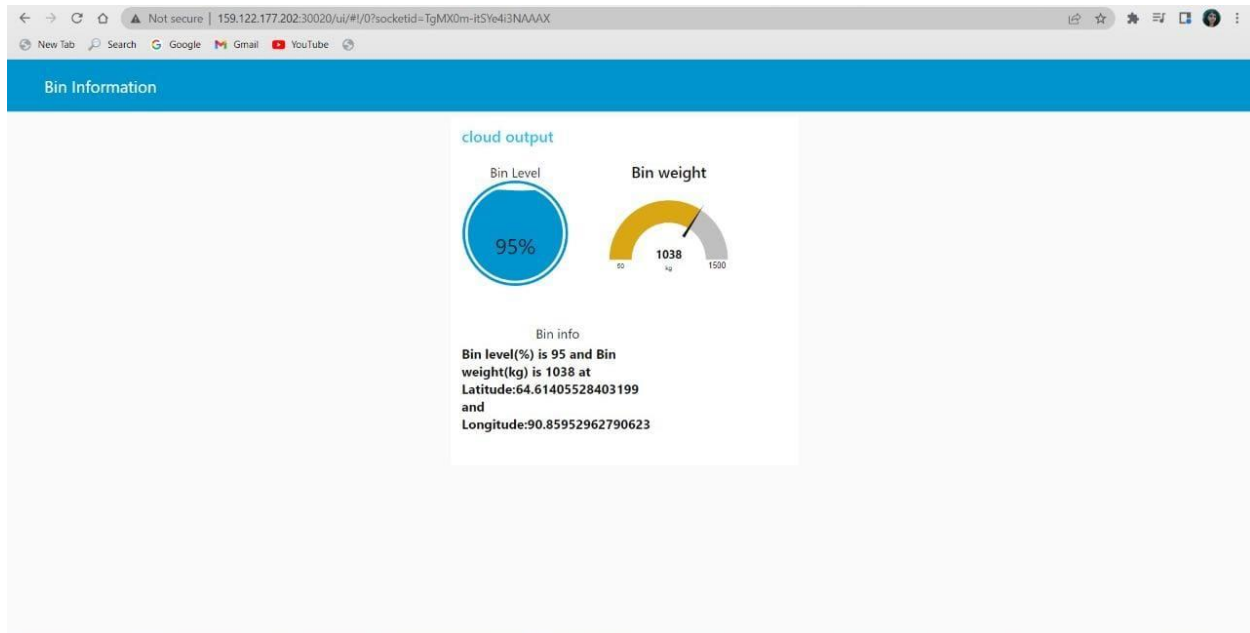
6. Data is stored in JSON format



The screenshot shows the 'sensor_data' database view in IBM Cloudant, displaying a single document in JSON format. The document has the following structure:

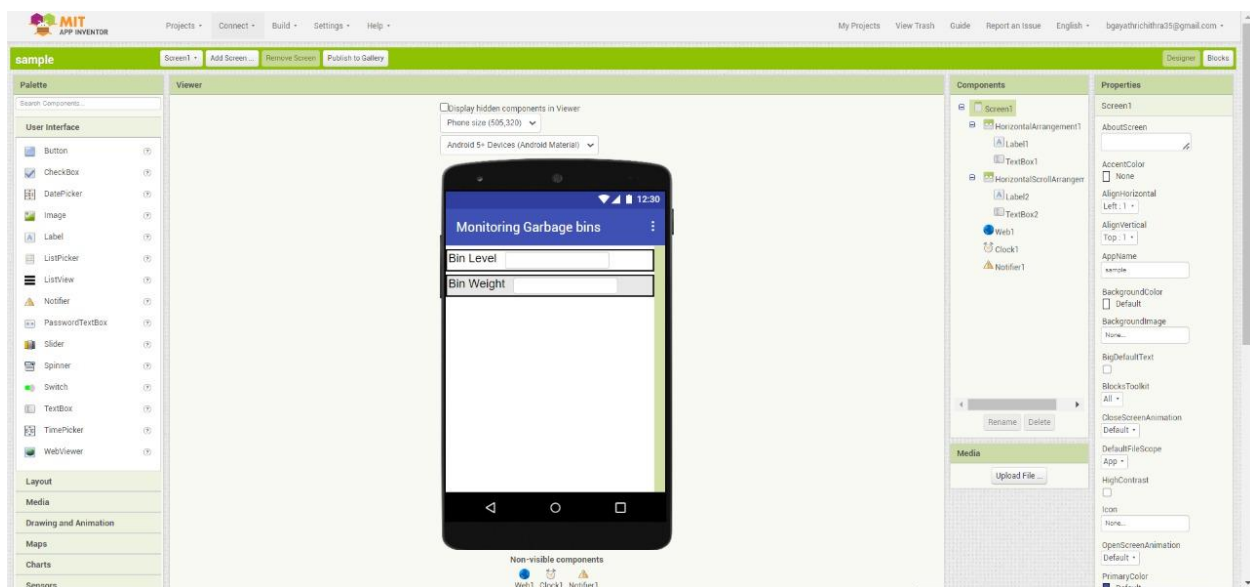
```
1 {
2   "_id": "0198213c192cb2c244cc2433f1802b91",
3   "_rev": "1-cde2dd17c519394df774730c495f8b",
4   "topic": "iot-2/type/SWMSGC/id/ibmproject/evt/data/fmt/json",
5   "payload": {
6     "Warning!!": "244.971e1f"
7   },
8   "deviceId": "ibmproject",
9   "deviceType": "SWMSGC",
10  "eventType": "data",
11  "format": "json"
12 }
```

Web UI:

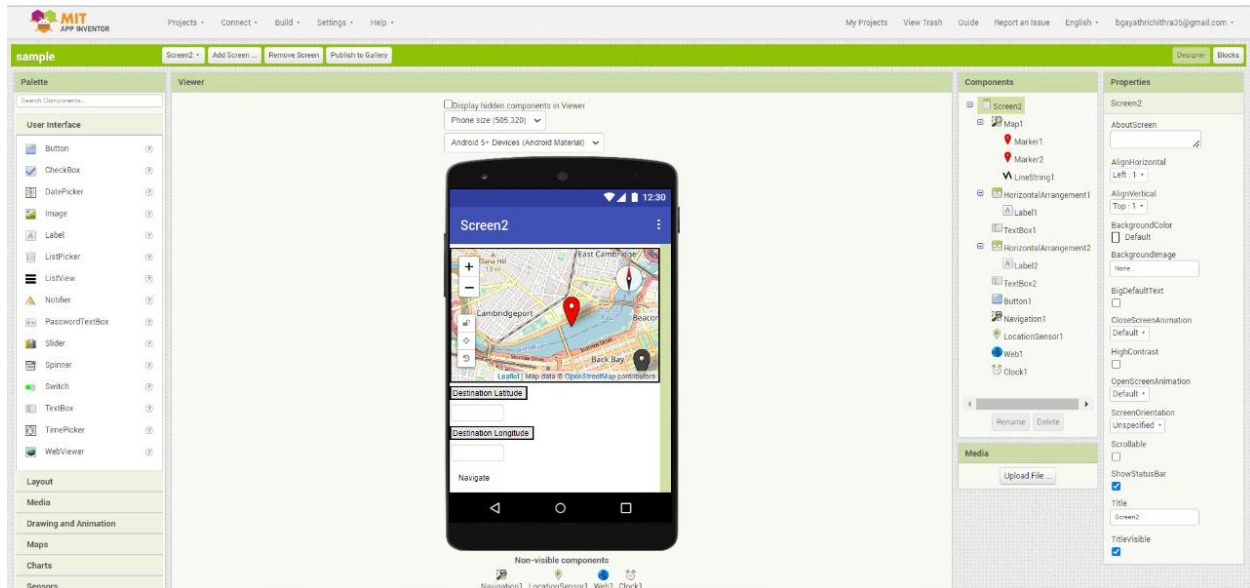


7. App is created using MIT App inventor

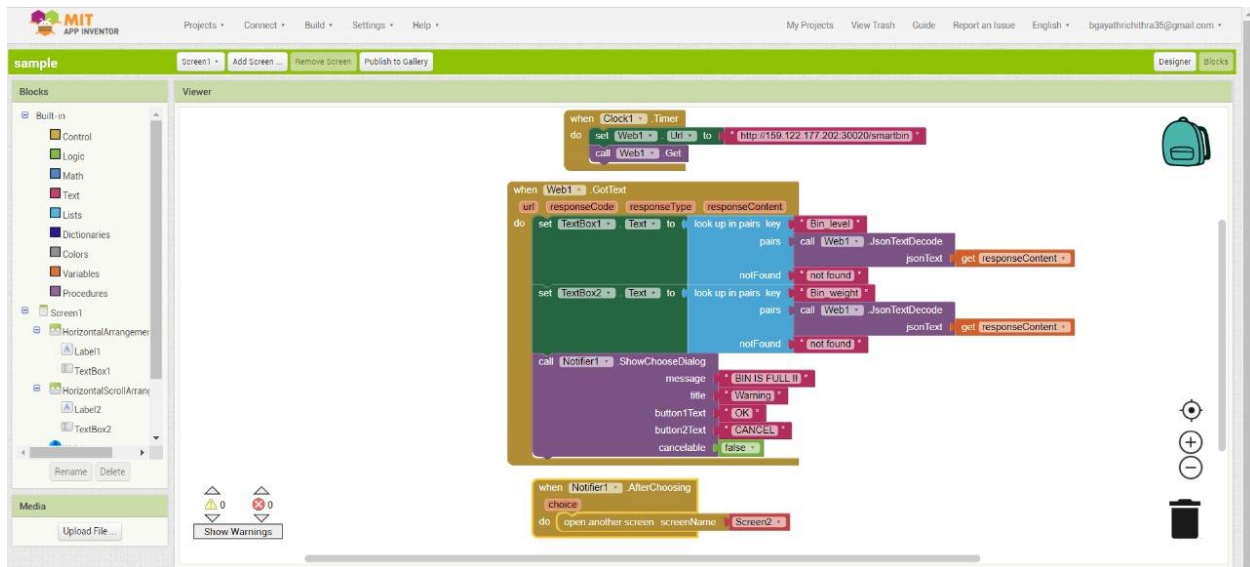
Screen 1:



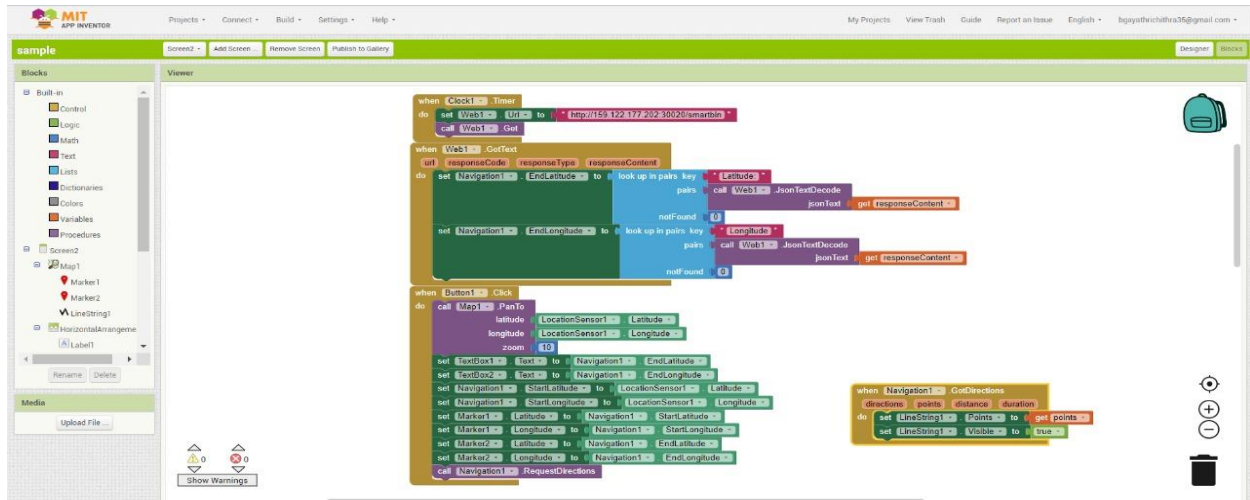
Screen 2:



Screen 1 blocks:



Screen 2 blocks:



8. Install MIT AI2 Companion in phone and scan the QR code showed in AI connect

