

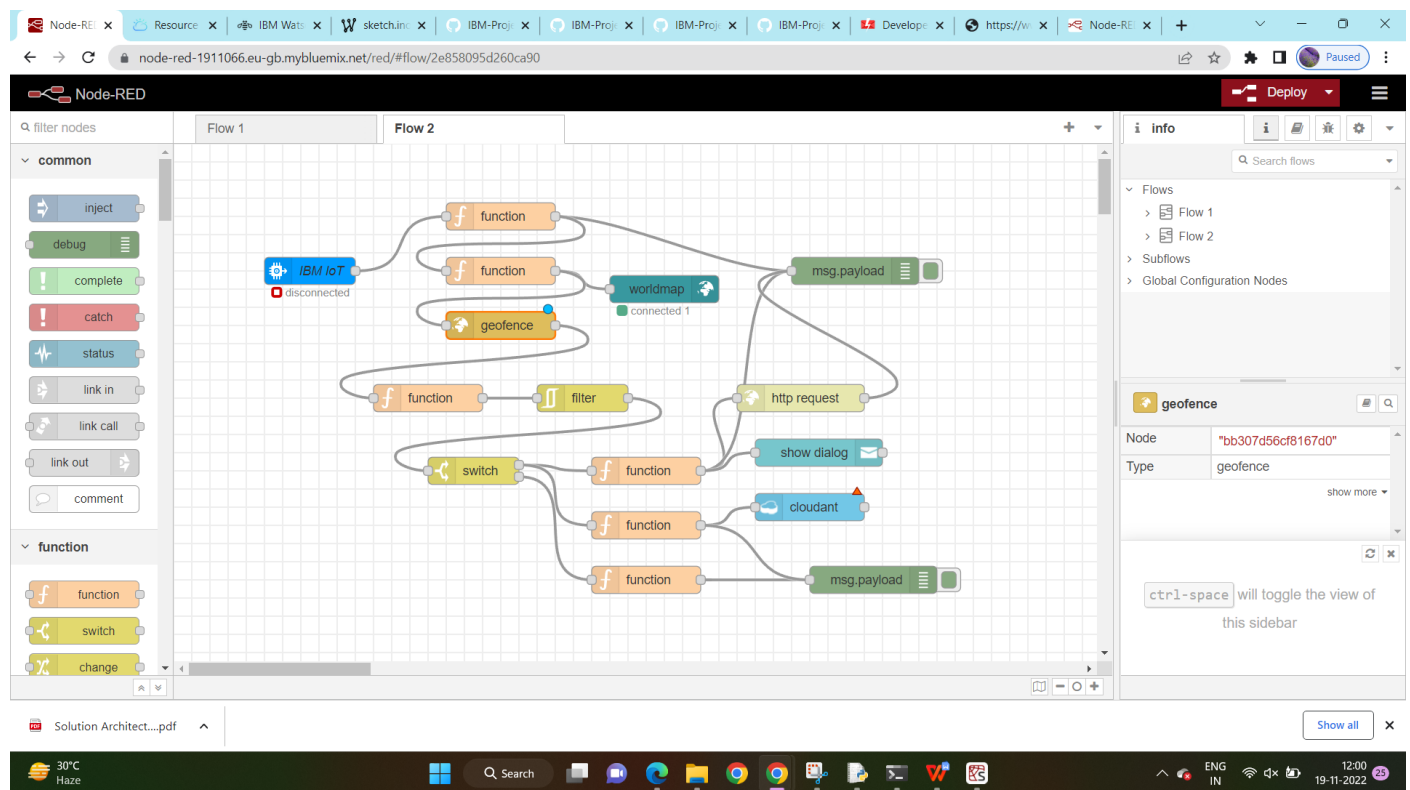
Project Development - Sprint 1

Iot Based Safety For Child Safety Monitoring & Notification

Team ID: PNT2022TMID20307

Creating Node - Red Service and Connecting with IBM cloud

Creating Node-Red Service :



Codes in Each Node:

This screenshot shows the Node-RED web interface in a browser. The main workspace displays a flow with an 'IBM IoT' node connected to several 'function' nodes, a 'worldmap' node, a 'geofence' node, a 'filter' node, a 'switch' node, and an 'http request' node. The right-hand panel is titled 'Edit ibmiot in node' and contains the following settings:

- Authentication:** API Key
- API Key:** child_iot
- Input Type:** Device Event
- Device Type:** IOT_CHILD
- Device Id:** 16022002
- Event:** All or +
- Format:** json
- QoS:** 0
- Name:** IBM IoT
- Service:** registered

The bottom of the screen shows a Windows taskbar with various application icons and a system tray displaying the date and time as 12:48 on 19-11-2022.

This screenshot shows the same Node-RED interface, but the right-hand panel is now titled 'Edit function node'. It displays the JavaScript code for a function node:

```
1 var name = msg.payload.name
2 var lat = msg.payload.lat
3 var lon = msg.payload.lon
4 global.set('latitude',lat)
5 global.set('longitude',lon)
6 global.set('name',name)
7 return msg;
```

The 'Properties' section on the right shows the 'Name' field set to 'Name'. The 'Setup' tab is selected, and the 'On Message' tab is also visible. The bottom of the screen shows the same Windows taskbar with the date and time as 12:48 on 19-11-2022.

Node-RED interface showing a flow diagram with nodes like IBM IoT, function, worldmap, geofence, filter, switch, and http request. The right panel displays the 'Edit function node' configuration with a JavaScript payload:

```
1- msg.payload={
2  'name':global.get('name'),
3  'lat':global.get('latitude'),
4  'lon':global.get('longitude')
5- }
6  return msg;
```

The bottom status bar shows the system temperature as 30°C and the date as 19-11-2022.

Node-RED interface showing the same flow diagram. The right panel displays the 'Edit geofence node' configuration with a map view of Hyderabad, India, and a search bar. The bottom status bar shows the system temperature as 30°C and the date as 19-11-2022.

Node-RED interface showing a flow diagram with nodes like **IBM IoT**, **function**, **worldmap**, **geofence**, **filter**, **switch**, and **http request**. The right sidebar displays the **Edit function node** configuration with the following code:

```
1 msg.payload = msg.location.inarea
2 return msg;
```

The bottom status bar shows the system temperature as 30°C and the date as 19-11-2022.

Node-RED interface showing the same flow diagram. The right sidebar displays the **Edit switch node** configuration with the following rules:

- Property: **msg.payload**
- Rule 1: **is false** → 1
- Rule 2: **is true** → 2

The bottom status bar shows the system temperature as 30°C and the date as 19-11-2022.

Node-RED interface showing a flow diagram and the Edit function node editor.

Flow Diagram:

- Flow 1: IBM IoT (connected) → function → worldmap (connected 1) → http request → show.
- Flow 2: function → geofence → function → filter → http request → show.
- Flow 3: switch → function → function → function.

Edit function node:

```
1 var d = new Date();
2
3 var utc = d.getTime() + (d.getTimezoneOffset()*60000);
4
5 var offset = 5.5;
6
7 newDate = new Date(utc +(3600000*offset));
8
9
10 msg.payload = {
11   "message": "Exit",
12   "Time": newDate.toLocaleString(),
13   "name": global.get('name'),
14   "lat": global.get('latitude'),
15   "lon": global.get('longitude')
16 };
17
18 return msg;
```

Node-RED interface showing a flow diagram and the Edit function node editor.

Flow Diagram:

- Flow 1: IBM IoT (connected) → function → worldmap (connected 1) → http request → show.
- Flow 2: function → geofence → function → filter → http request → show.
- Flow 3: switch → function → function → function.

Edit function node:

```
1
2 var d = new Date();
3
4 var utc = d.getTime() + (d.getTimezoneOffset()*60000);
5
6 var offset = 5.5;
7
8 newDate = new Date(utc +(3600000*offset));
9
10
11 msg.payload = {
12   "message": "Entry",
13   "Time": newDate.toLocaleString(),
14   "name": global.get('name'),
15   "lat": global.get('latitude'),
16   "lon": global.get('longitude')
17 };
18
19 return msg;
```

