**DOCUMENTATION**

**The Prequesties to run the Node.js application**

**1. Ubuntu 16.04 lts**

**2. Install Node.js**

**step 1:**

sudo apt-get update

**step 2:**

sudo apt-get install nodejs

**step 3:**

node -v

**3. Install NPM**

**step 1 :**

sudo apt-get install npm

**step 2:**

npm -version

**4. Install Postgres**

**step 1:**

sudo add-apt-repository "deb http://apt.postgresql.org/pub/repos/apt/ xenial-pgdg main"

**step 2:**

wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

**step3:**

sudo apt update

**step 4 :**

sudo apt install postgresql-9.6 postgresql-contrib-9.6

**step 5:**

psql –version

**step 6:**

sudo -u postgres createuser -P amudalab

sudo -u postgres createdb -O amudalab smartbuilding

**step 7:**

psql -h localhost -U amudalab smartbuilding

\q

**5. Install Postgis**

**step 1:**

sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable

sudo apt update

**step 2:**

sudo apt install postgis postgresql-9.6-postgis-2.3

**step 3:**

sudo -u postgres psql -c "CREATE EXTENSION postgis; CREATE EXTENSION postgis\_topology;" gisdata

**6. Install PHP**

**7. Create database – Smartbuilding**

**The required tables are**

**1. main\_person**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Identity** | **Description** |
| Mac | Varchar | Primary key, foreign key -- person\_details(mac) | Mac id of the android device |
| X | Double |  | X coordinate |
| Y | Double |  | Y coordinate |
| Time | Timestamp without timezone | Primary key | Recorded time |
| Geom | Geometry |  | Geometric value of coordinates x and y |

**2. person\_details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Identity** | **Description** |
| Person\_ID | Varchar | Primary key | Unique id of the person |
| Mac | Varachar | foreign key -- main\_person(mac) | Mac id of the android device |
| Name | Varchar |  | Name of the person |
| Person\_Category | Varachar |  | Person Category |
| Person\_status | Varchar |  | Current status of the person |
| Temporal\_granularity | Varchar |  | Temporal granularity of the person |
| Spatial\_granularity | Varachar |  | Spatial granularity of the person |

**3. main\_asset**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Identity** | **Description** |
| EPC | Varchar | Primary key, foreign key -- asset\_details(EPC) | EPC id of an asset |
| X | Double |  | X coordinate |
| Y | Double |  | Y coordinate |
| Time | Timestamp without timezone | Primary key | Recorded time |
| Geom | Geometry |  | Geometric value of coordinates x and y |

**4. asset\_details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Identity** | **Description** |
| Asset\_ID | Varchar | Primary key | Unique id of the asset |
| EPC | Varachar | foreign key -- main\_asset(EPC) | EPC id of the asset’s tag |
| Name | Varchar |  | Name of the asset |
| Asset\_Category | Varachar |  | Asset Category |
| Asset\_status | Varchar |  | Current status of the asset |
| Temporal\_granularity | Varchar |  | Temporal granularity of the asset |
| Spatial\_granularity | Varachar |  | Spatial granularity of the asset |

**5. room**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Identity** | **Description** |
| Room\_id | Varchar | Primary key | Unique id of the room |
| Room\_name | Varachar |  | Name of the room |
| Geom | Varchar |  | Geometry of the room |
| Floor\_id | Varachar |  | Floor number of the room located |

**File Structure**

**Location of the file :**

System id : 5036

/home/researcher/Query-Interface

**Smart Hospital**

**-- css**

-- bootstrap.css

-- bootstrap.min.css

-- bootstrap-datetimepicker.css

-- bootstrap-datetimepicker.min.css

-- bootstrap-slider.css

-- halfslider.css

-- jquery-ui.css

-- L.icon.Pulse.css

-- style.css

-- wickedpicker.css

**-- floor1**

-- bld\_blockIII\_floor1\_columns.json

-- bld\_blockIII\_floor1\_full\_perm\_walls.json

-- bld\_blockIII\_floor1\_perm\_walls.json

-- bld\_blockIII\_floor1\_rooms.json

-- bld\_blockIII\_floor1\_stairss.json

-- bld\_blockIII\_floor1\_temp\_walls.json

-- bld\_blockIII\_floor1\_windows.json

**-- floor2**

-- bld\_blockIII\_floor2\_columns.json

-- bld\_blockIII\_floor2\_full\_perm\_walls.json

-- bld\_blockIII\_floor2\_perm\_walls.json

-- bld\_blockIII\_floor2\_rooms.json

-- bld\_blockIII\_floor2\_stairss.json

-- bld\_blockIII\_floor2\_temp\_walls.json

-- bld\_blockIII\_floor2\_windows.json

**-- floor3**

-- bld\_blockIII\_floor3\_columns.json

-- bld\_blockIII\_floor3\_full\_perm\_walls.json

-- bld\_blockIII\_floor3\_perm\_walls.json

-- bld\_blockIII\_floor3\_rooms.json

-- bld\_blockIII\_floor3\_stairss.json

-- bld\_blockIII\_floor3\_temp\_walls.json

-- bld\_blockIII\_floor3\_windows.json

**-- fontawesome**

**--css**

-- fontawesome.css

-- fontawesome.min.css

**-- fonts**

-- FontAwesome.otf

-- fontawesome-webfont.otf

-- fontawesome-webfont.svg

-- fontawesome-webfont.ttf

-- fontawesome-webfont.woff

-- fontawesome-webfont.woff2

**-- fonts**

-- fontello.eot

-- fontello.svg

-- fontello.ttf

-- fontello.woff

-- glyphicons-halflings-regular.eot

-- glyphicons-halflings-regular.svg

-- glyphicons-halflings-regular.ttf

-- glyphicons-halflings-regular.woff

-- glyphicons-halflings-regular.woff2

**-- images**

-- leaflet.ajax.min.js

-- markers-matte.png

-- markers-matte@2x.png

-- markers-plain.png

-- markers-shadow.png

-- markers-shadow@2x.png

-- markers-soft.png

-- markers-soft@2x.png

**-- js**

**-- leaflet-0.7.2**

-- leaflet.ajax.min.js

**-- locales**

-- collection of bootstrap js files

-- collection of bootstrap js files

**-- node\_modules**

-- collection of node\_modules js and css

**-- pg**

-- collection of postgres dependencies

-- combined.css

-- function.js

-- leaflet.awesome-markers.css

-- leaflet.awesome-markers.js

-- package-lock.json

-- people\_live.php

-- server.js

**Node.js App**

**FILES:**

**1. people\_live.php**

Contains the visualization code of the live tracking of people and assets. It also contains the live queries for the people and assets.

**2. Server.js**

It is the main node.js server page, which consists of the code for the following

**database connectivity**

var pg = require('pg');

var connectionString = 'postgres://amudalab3:amudalab@127.0.0.1:5432/smartbuilding'; //host:port/dbname

var client = new pg.Client(connectionString);

client.connect();

**queries for the visualization**

**adding the shapefiles from directory**

the following node.js route is used

app.get('/floor/:floorid', function (req, res)

**3. function.js**

Contains the functions for all live and history based querying the people and assets for the visualization.

**the location of the files**

/home/researcher/Query\_Interface/SmartHospital/people\_live.php

/home/researcher/Query\_Interface/SmartHospital/history.php

/home/researcher/Query\_Interface/SmartHospital/server.js

/home/researcher/Query\_Interface/SmartHospital/function.js

**Data generation**

**1. Synthetic data generation**

synthetic data for the people and asset are generated using the files

1. populate\_people.py

2. populate\_asset.py

**the location of the files**

/home/researcher/Query\_Interface/Populate/populate\_people.py

/home/researcher/Query\_Interface/Populate/populate\_asset.py

**2. Handling the Real time data**

data in real time for the people and asset are handled by the rabbitmq server.

The data is consumed from the wifi device and rfid sdk using the following python scripts.

1. people\_realtime.py

2. asset\_realtime.py

**the location of the files**

/home/researcher/Query\_Interface/Realtime/people\_realtime.py

/home/researcher/Query\_Interface/Realtime/asset\_realtime.py

**Adding a new button in the Node.js app and also to add a shape file**

**HTML code**

<input type="button" name="loadmap" id = "loadmap" class="btn btn-success" onclick="loadmap()">

**Javascript code**

**//**javascript function to get the floor id and to add it to the leaflet container

function loadmap() {

var currentFloor = 1; // the id may be 1, 2,3,4 based on the required floor

$.getJSON('http://172.17.137.160:3333/floor/' + currentFloor, function(data) {

console.log(data);

for (i in data.files) {

addJSONDataToMap(data.files[i]);

}

});

}

function addJSONDataToMap(dataURL) {

$.getJSON(dataURL, function(jsondata) {

let layer = L.geoJson(jsondata).addTo(map);

map.fitBounds(layer.getBounds());

});

}

**Node.js code**

//corressponding node.js code for the above javascript code, fetches the json files from the folder in the system. The folder should be inside the main folder of the application.

app.get('/floor/:floorid', function (req, res) {

"use strict";

fs.readdir('./floor' + req.params.floorid + '/', (err, files) => {

let fileList = [];

if (req.params.floorid) {

files.forEach(file => {

fileList.push('/floor' + req.params.floorid + '/' + file);

});

res.json({

files: fileList

});

} else {

res.json({

files: []

});

}

});

});

**Adding a dropdown list and button in the Node.js app to track entities**

**HTML Code**

//the html code to insert an dropdown list and a button in the webpage

<select id="track" >

<option selected="selected" value = "all">--Asset/People--</option>

<option value="People">People</option>

<option value="Asset">Asset</option>

</select>

<input type="button" name="track" id = "track" class="btn btn-success" onclick="findallpeople()">

**Javascript code**

//the javascript code to have markers on the map and update the position of the marker based on the current position of the data.

var markers = [];

var totalTrackingObjects = 0;

var allpeople = function () {

markers = [];

var pc = $("#track option:selected").val();

$.getJSON('http://172.17.137.160:3333/live/' +pc, function (data) {

totalTrackingObjects = data.length;

for (var i = 0; i < data.length; i++) {

var cat = data[i].person\_category;

console.log(cat);

if(cat == "PATIENT"){

var marker = L.marker([0,0], { icon: L.AwesomeMarkers.icon({icon: 'wheelchair', prefix: 'fa', markerColor: 'red'})}).addTo(map);

markers.push(marker);

//icn = 'wheelchair';

}

else if(cat == "DOCTOR")

{

var marker = L.marker([0,0], { icon: L.AwesomeMarkers.icon({icon: 'user-md', prefix: 'fa', markerColor: 'green'})}).addTo(map);

markers.push(marker);

//icn = 'user-md';

//icon = 'medkit';

}

else if(cat == "NURSE")

{

var marker = L.marker([0,0], { icon: L.AwesomeMarkers.icon({icon: 'medkit', prefix: 'fa', markerColor: 'blue'})}).addTo(map);

markers.push(marker);

}

}

});

updateallpeople();

};

var updateallpeople = function () {

var pc = $("#track option:selected").val();

$.getJSON('http://172.17.137.160:3333/live/' +pc, function (data) {

//totalTrackingObjects = data.length;

if (data.length == 0) {

alert("No Person Available");

window.location.reload();

console.log("Detected 0");

for (var i = 0; i < totalTrackingObjects; i++) {

map.removeLayer(markers[i]);

//window.location.reload();

}

} else if (data.length != totalTrackingObjects) {

for (var i = 0; i < totalTrackingObjects; i++) {

map.removeLayer(markers[i]);

}

allpeople();

}

for (var i = 0; i < data.length; i++) {

var x = data[i].x;

var y = data[i].y;

var mac = data[i].mac;

var cat = data[i].person\_category;

markers[i].setLatLng([x, y]);

markers[i].bindPopup('x:'+x+'<br/>y:'+y+'<br/>NAME:'+mac+'<br/>Category:'+cat);

}

});

};

//this function is used to recursively call the above function after the timeout.

var findallpeopl = function () {

setTimeout(function () {

updateallpeople();

findallpeopl();

}, 1000); // 1000 milliseconds to refresh

};

**Node.js code**

//the corresponding node.js code to the above javascript function.it is used to fetch the data from the database using the folowing queries.

app.get('/live/:pc', function (req, res) {

"use strict";

var results = [];

var m = req.params.pc;

if(m == "Asset"){

var query = client.query("SELECT epc as mac,time,x,y FROM main\_asset t1 WHERE time = (SELECT MAX(time) FROM main\_asset t2 WHERE t1.epc = t2.epc)");

query.on('row', (row) => {

results.push(row);

});

query.on('end', () => {

res.end(JSON.stringify(results));

});

}

else if(m == "People"){

var query = client.query("SELECT person\_category,t2.name as mac,time,x,y FROM main\_person t1,person\_details t2 WHERE t1.mac in (select mac from person\_details where t1.mac=t2.mac) and time = (SELECT MAX(time) FROM main\_person t2 WHERE t1.mac = t2.mac)");

query.on('row', (row) => {

results.push(row);

});

query.on('end', () => {

res.end(JSON.stringify(results));

});

}

});