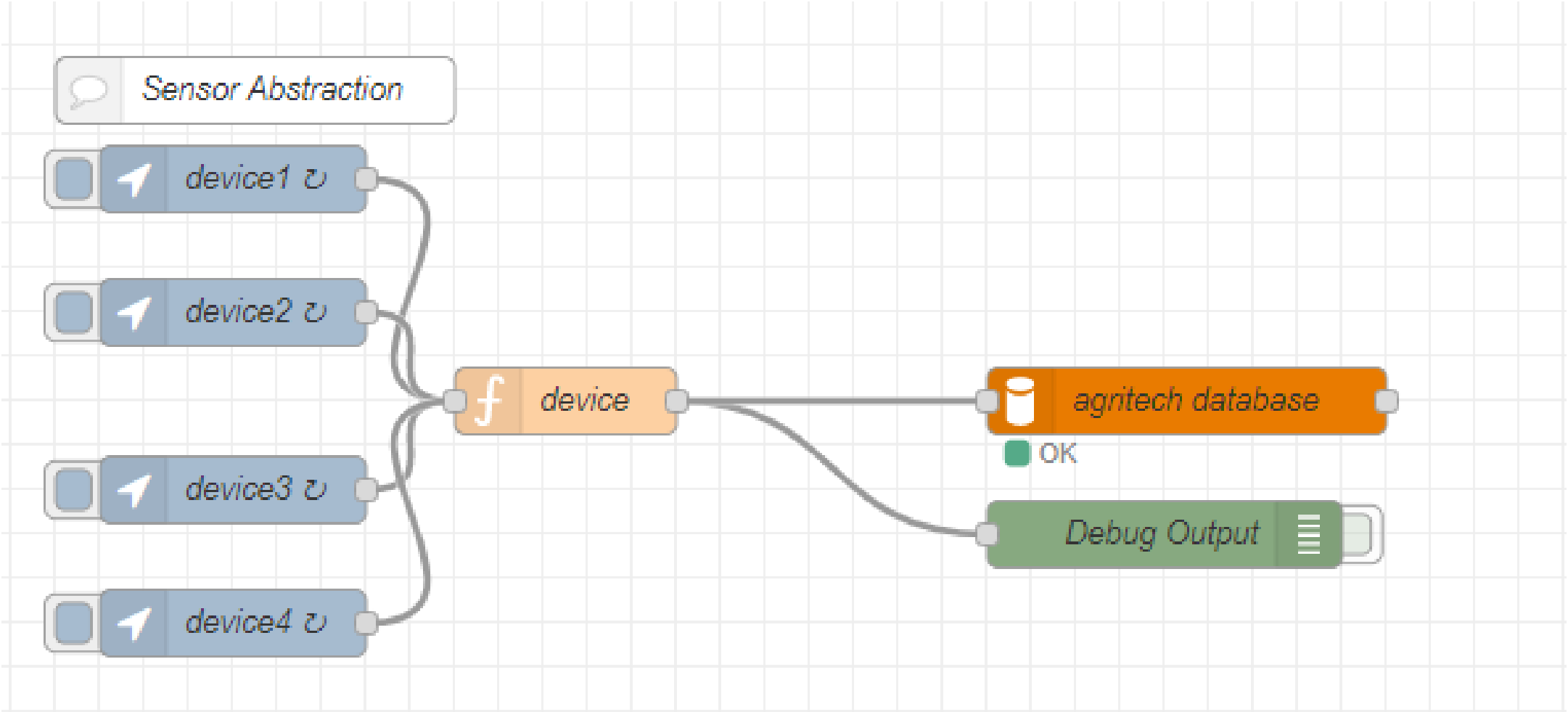
# Sensor Ball Tracking Map

Muhammad Shaafay Saqib

# Agenda

- Research, figure out and show how to:
  - Get Sensor Data from Node-Red to Database
  - Get data from Database to Server using Node.js
  - GET data from server to Google Maps Output

# Sensor Data on Node-RED



# Insert Sensor data into database

Edit function node

Delete | Cancel | Done

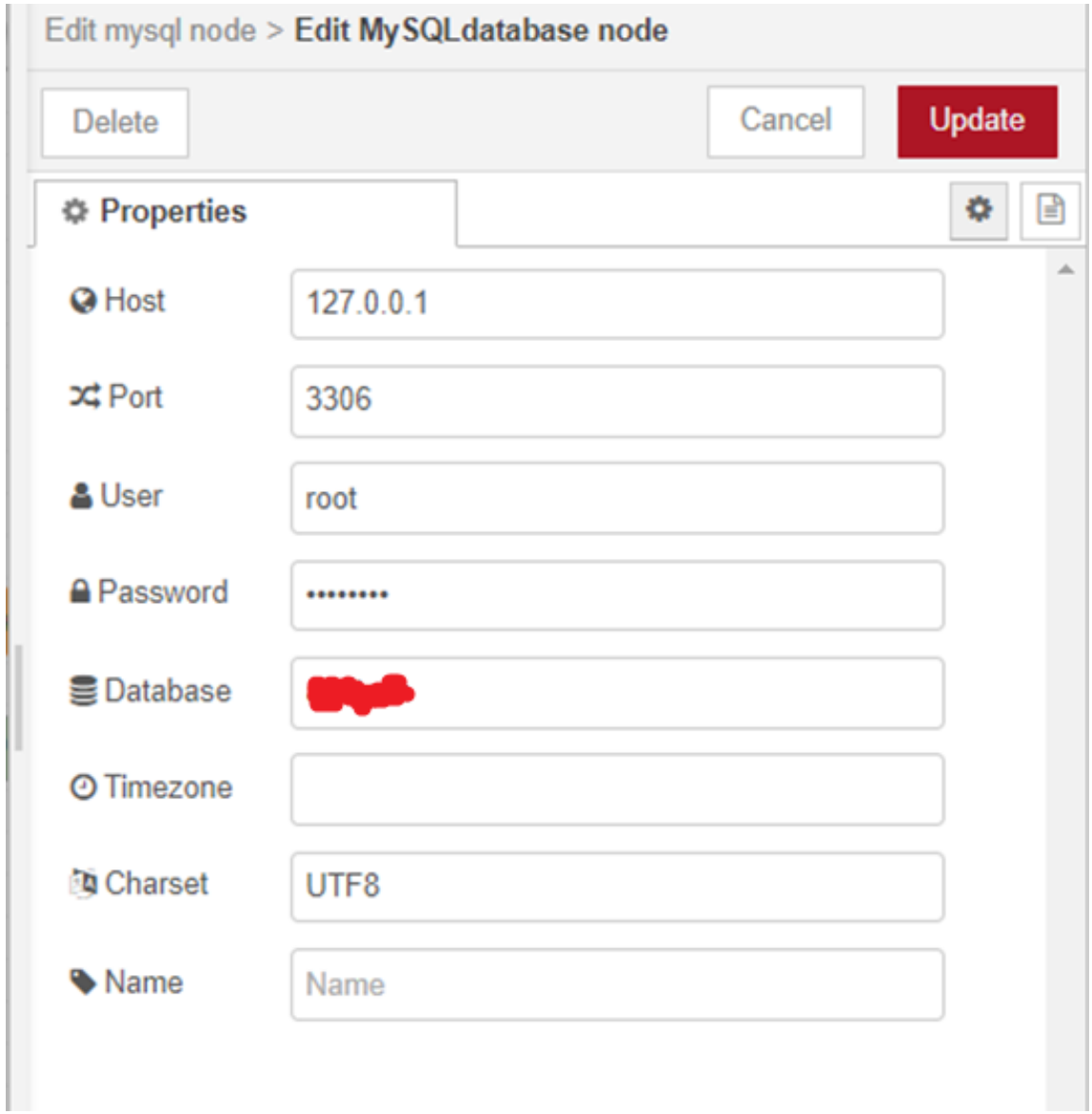
Properties

Name: device

Setup | On Start | **On Message** | On Stop

```
device = msg.device;

/* --- ABSTRACTION :: get new lat and lng values from sensor*/
original_lat = context.get('lat') || msg.original_lat;
original_lng = context.get('lng') || msg.original_lng;
d_lat = Math.random() * 0.0001 * (Math.random() < 0.5 ? 1 : -1);
d_lng = Math.random() * 0.0001 * (Math.random() < 0.5 ? 1 : -1);
lat = parseFloat((original_lat + d_lat).toFixed(4));
lng = parseFloat((original_lng + d_lng).toFixed(4));
context.set('lat', lat);
context.set('lng', lng);
/* --- END ABSTRACTION */

msg.payload = [lat, lng, device, lat, lng]
msg.topic = "INSERT INTO positions VALUES (?, ?, ?) ON DUPLICATE KEY UPDATE `lat`=?, `lng`=?;";

return msg;
```

# Insert Sensor data into database



# Get data from Database to Server using Node.js

- Must use Node.js to create a server
- Set up a GET request to have the server retrieve from the database
- Send back to the client

We will need to change values to connect to the real database

# Connect to database

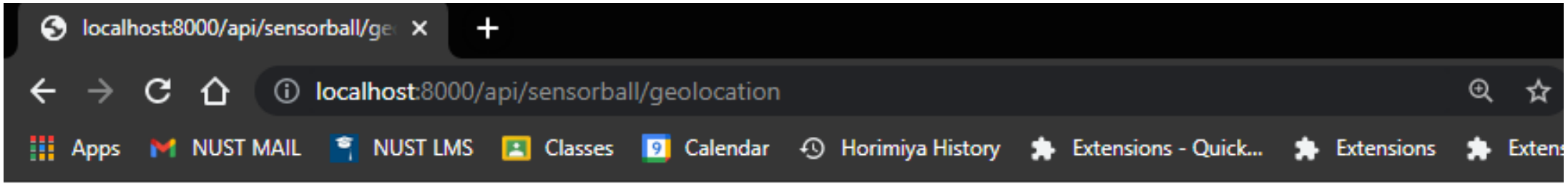
```
"use strict";
//creating a mysql pool for the a shared instance of db con
nection in whole application
var mysql = require('mysql');
var fs = require('fs')

var HOST = "127.0.0.1"
var USER = "root"
var PASSWORD = "[redacted]"
var DB = "[redacted]"
var PORT = "3306"

var connection = mysql.createPool({
   host: HOST,
   user: USER,
   password: PASSWORD,
   database: DB,
   port: PORT,
   connectionLimit: 100,
   connectTimeout: 60 * 60 * 1000,
   acquireTimeout: 60 * 60 * 1000,
   supportBigNumbers: true,
   bigNumberStrings: true,
   charset: 'utf8mb4_unicode_ci',
   multipleStatements: true,
});

module.exports = connection;
```

Set up get request, which will retrieve from database and return JSON

# Index.js

```
const express = require('express')
const mysql = require("mysql")
const app = express();
const port = 8000;
const connection = require('./dbretrieve')

app.use(express.static(__dirname));

app.get('/api/sensorball/geolocation', function(req,res) {
    new Promise(function (resolve, reject) {
        var sql = "SELECT * from ??;";
        var inserts = ['positions'];
        sql = mysql.format(sql, inserts);

        connection.query(sql, function (err, result) {
            if (err) return reject(err);

            const content = JSON.stringify(result);
            res.send(content);

            resolve(result);
         });
    })

});

app.get('/', function(req, res) {
  res.sendFile(__dirname + "/dashboardSB1.html");
})

app.listen(port, () => {
  console.log(`Cool app listening on port ${port}!`);
});
```

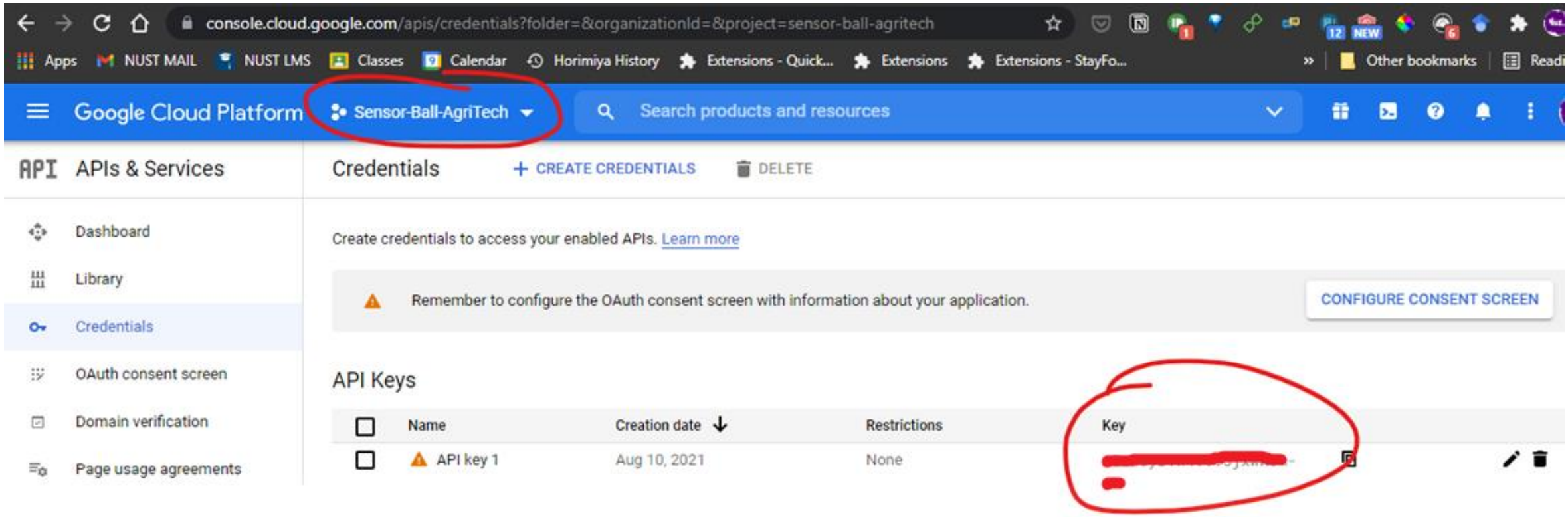
# GET Request




[{"lat":33.6482,"lng":72.9943,"dname":"device1"},
{"lat":33.6482,"lng":72.9944,"dname":"device2"},
{"lat":33.6482,"lng":72.9945,"dname":"device3"},
{"lat":33.6482,"lng":72.9944,"dname":"device4"}]

# GET data from server to Google Maps Output

First set up the Google Maps API and get the API key



# Add Google Maps JS API

```
<body>
        <div class="container ">
            <h1 class = ".h1"><b>
Sensor Ball Location Tracker</b></h1>
            <div id="map-canvas" style="
width:900px;height:500px"></div>
        </div>

        <!-- initalisation with markers etc-->
        <script src = "initialisation.js"></script>

<!-- Google Maps API with my API Key and initialise() as th
e call back-->
        <script src=
"https://maps.googleapis.com/maps/api/js?v=3.exp&key=[redacted]
[redacted]&callback=initialize"
></script>

<!-- handling change of marker positions with events -->
        <script src="eventHandling.js"></script>

    </body>
```

# initialisation.js

```
/* VARIABLES */

// Map Starting Location (i.e NUST)
STARTING_LATITUDE = 33.6425;
STARTING_LONGITUDE = 72.9930;

// Array of all markers with lng, lat and an empty marker a
ttribute.
MARKER_OBJECTS_JSON_FILE_PATH =
`/api/sensorball/geolocation`

MAP_DIV = 'map-canvas';
// The id of the div in which the map will be displayed
ZOOM = 20; // Initial zoom of the map
```

# initialisation.js

```
// The callback function that will be called by the Googl
e Maps API
var initialize = function() {


// Initialise the Map in the div, with specified lat, lng a
nd zoom
    map = new
 google.maps.Map(document.getElementById(MAP_DIV), {cente
r: {lat: lat, lng: lng}, zoom: ZOOM});

    $.getJSON(markerObjectsJSON, function(markerObjects) {

        // Initialise the markers
        for (var i = 0; i < markerObjects.length; ++i) {
            console.log(markerObjects[i]);
            allmarkers[i] = new google.maps.Marker(
                {
                    position:{ lat: markerObjects[i].lat, l
ng: markerObjects[i].lng},
                    label: markerObjects[i].dname,
                    map: map
                }
            );
        }
    });


// set call back function to be part of the window so tha
t API can call it?
    window.initialize = initialize;
};

// Input: Marker, Payload with attributes lat and lng
// Output: Void
// Effect: Map updates Marker with new lat and lng coordina
tes from Payload
var redraw = function(marker, payload) {
    lat = payload.lat;
    lng = payload.lng;
    marker.setPosition({lat:lat, lng:lng, alt:0});
};
```

# eventHandling.js

```
// function to update markers from JSON file
function updateMarkersWithJSONFile() {
    $.getJSON(markerObjectsJSON, function(markerObjects) {
        for (var i = 0; i < allmarkers.length; ++i) {
            console.log(markerObjects[i])
            redraw(allmarkers[i], {lat: markerObjects[i].la
t, lng: markerObjects[i].lng})
        }
    });
}

// Run call every 1 second
setInterval(updateMarkersWithJSONFile, 1000);
```

# Next Steps

- Making the code a bit more sophisticated (e.g using Routes)
- Improving design
- Integrating with final website