IOT FINAL PROJECT

DESCRIPTION OF CLIENT SIDE AND SERVER SIDE WITH ARCHITECTURE

PROFESSORS : DAVIDE ANCONA , GIORGIO DELZANNO

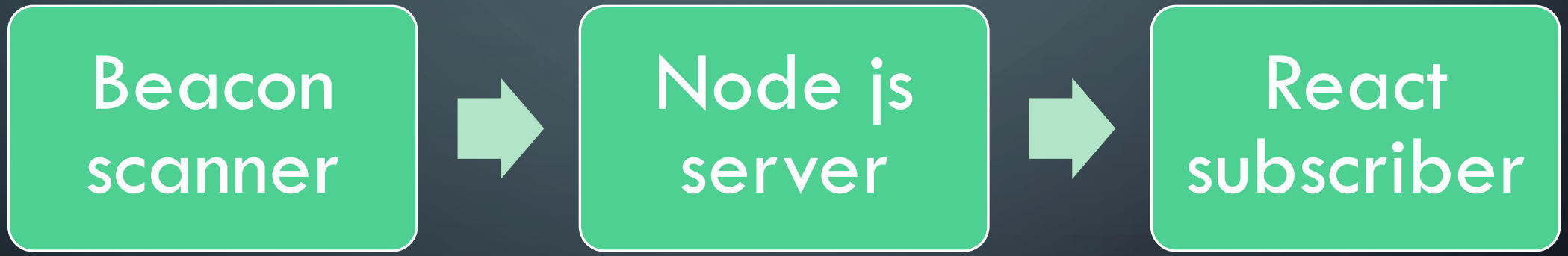
SEYED ERFAN DAVOODI

S4842444

INTRODUCTION AND INTENTION OF ARCHITECTURE

- I thought of implementing the project in a way in order to learn more and do it close to reality to be useful in the future if it was possible ,Therefor I implemented an android app which can be installed on any android device for detecting beacons. Assuming that we have each of these devices in different floors/rooms of the building for detecting beacons however as I am working on a prototype the detection is not automatic (as my beacon detector is my phone and it is a battery consuming process) .After detecting beacons by our android device the data is passed to the server which is implemented by node js. Also there is an admin dashboard implemented with react-native to monitor the data passed to it.

THE SCHEMA OF THE ARCHITECTURE

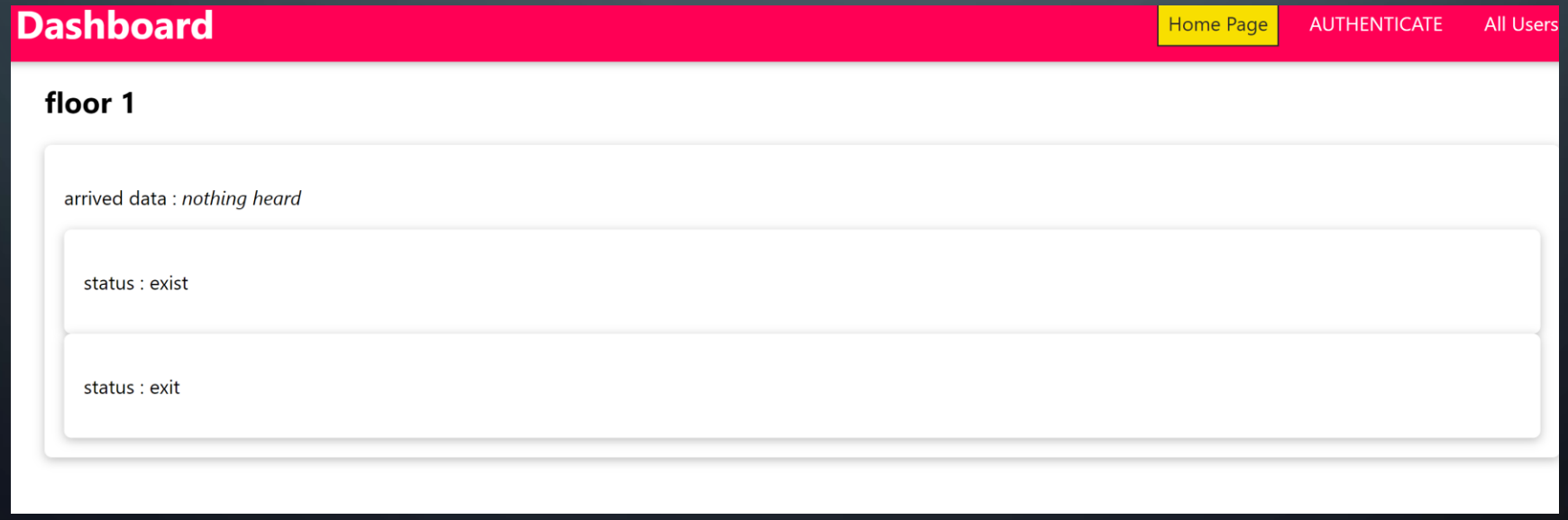


CLIENT SIDE OF THE PROJECT

- Earlier we discussed about the android beacon scanner that detects alt beacons . Accurately not only it detects beacons but monitors them if they are inside the zone or they left the zone and if they are in the zone how long is the beacon distance to beacon scanner.
- -- insert the image of beacon scanner --

CLIENT SIDE OF THE PROJECT

- The react subscriber in the home page shows the message passed to it and also shows list of beacons whether they exist in the area or they exited the area in the homepage of the application.

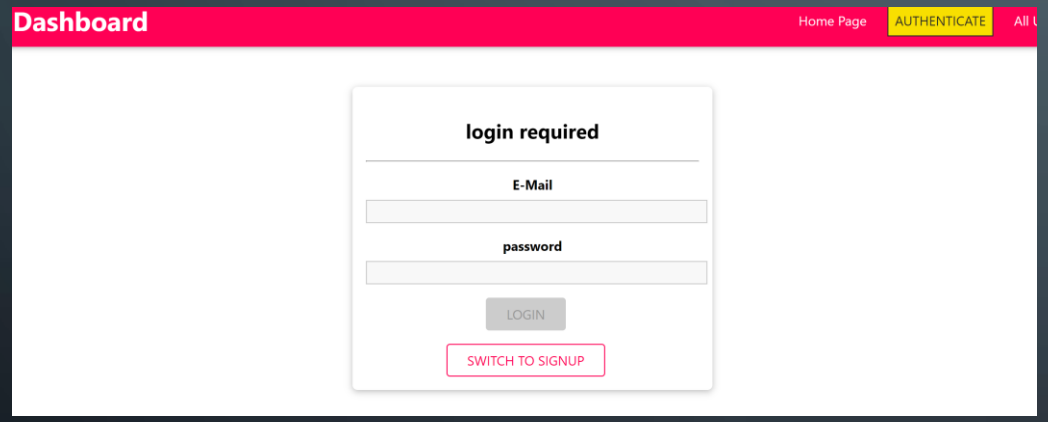


CLIENT SIDE OF THE PROJECT

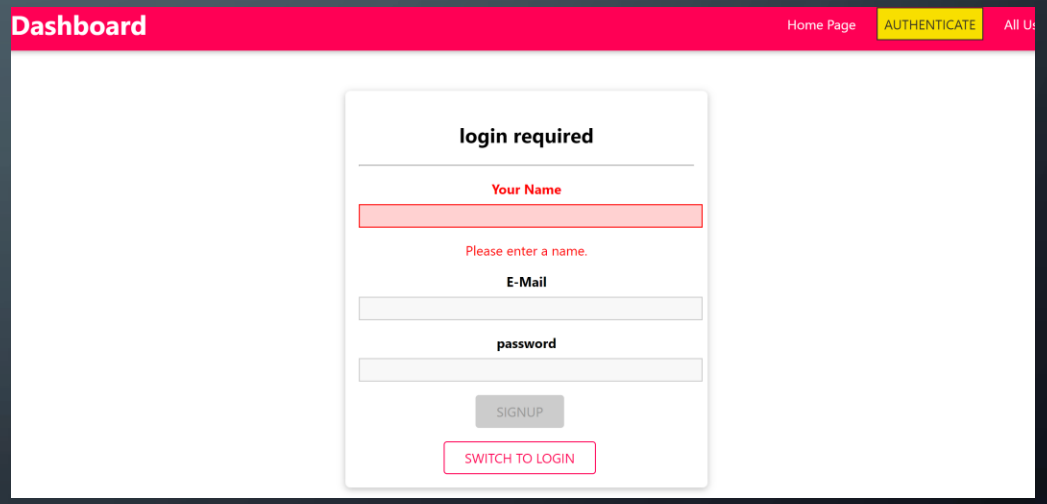
- Apart from what we have seen I provided authentication in order to avoid any access to important features like adding users to the system or monitoring beacons charts therefor you should first authenticate to access other parts of the system or you are just able to monitor real time data on screen.
- It has two parts login and register also I implemented form validation for not leaving it empty or enter email address in the correct format. Also if the validation is not ok the button for sending request will not be activated

CLIENT SIDE OF THE PROJECT

ADMIN LOGIN PORTAL



ADMIN REGISTER PORTAL

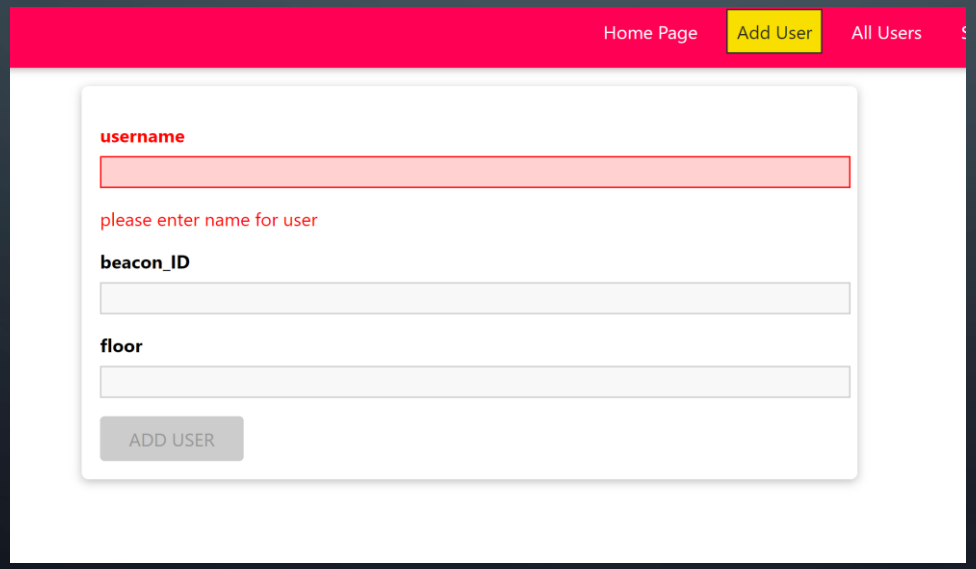


CLIENT SIDE OF THE PROJECT

- When you authenticate and you have access to the system you can register new users , Get the list of users , and also fetch sensor data in order to see the sensors charts based on date.

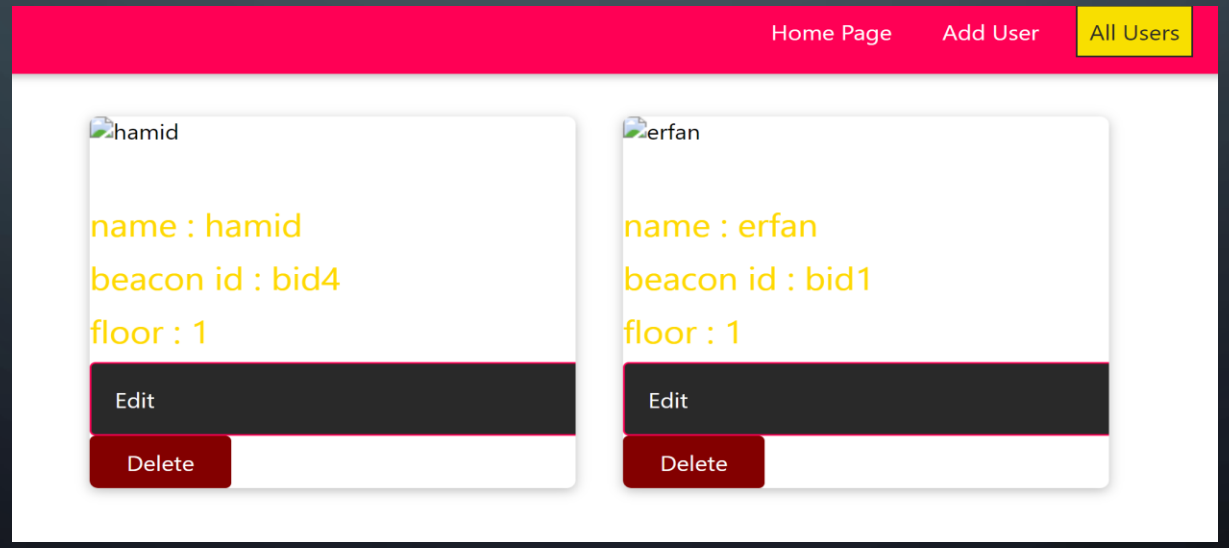
ADD USER

- Here I developed a form that you can enter the users data for example the floor they work inside and the beacon-id of the user with the user name later you can check the users based on beacon-ids to see when they left or entered the zone. Also this form has validation in order to enter correct data.



USER LIST

- Here you can get the list of users and also edit their data or delete them from the list of users . As the purpose of this project was not focused on restful services I did not implement the update part and get data part for the users but it is not going to be hard as the application is coded with expandable modularity.

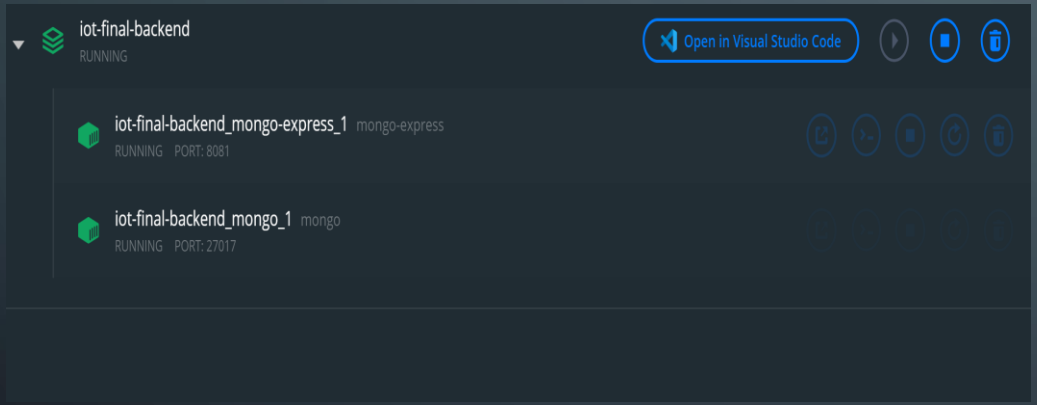


CHARTS OF BEACONS

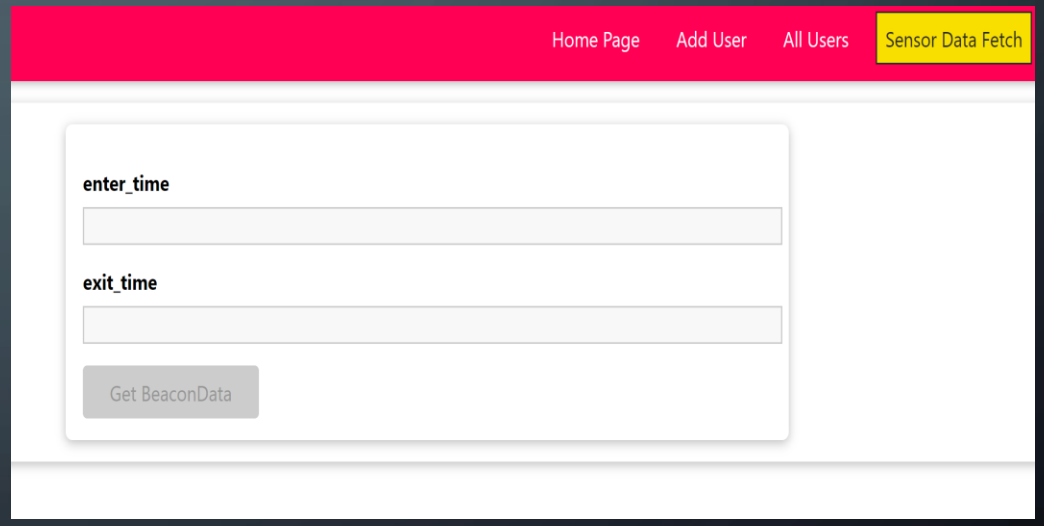
- This part was one of the most challenging parts apart from the mqtt part and learning react to code the dashboard because the performance in this part of the application was so important. Later in the summerized video I will explain the code with more details . It gets an enter time and end time and then fetches the data based on the information we saved to mongo database in the mqtt real time system about the existance and exiting the enviorment of the beacon scanner . Also the mongo database is based on docker system.

SEARCH CHART FILTER

DATABASE SYSTEM

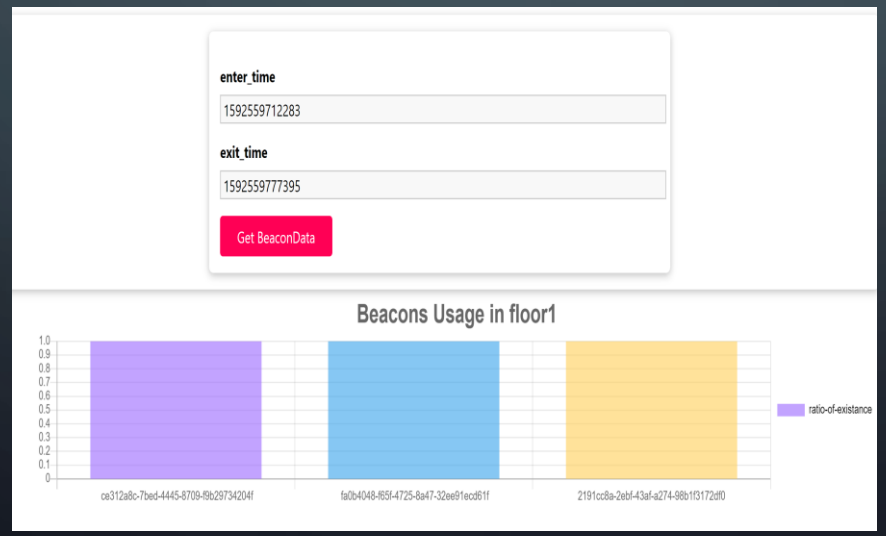


THE TIME FILTER FORM FOR SEARCH

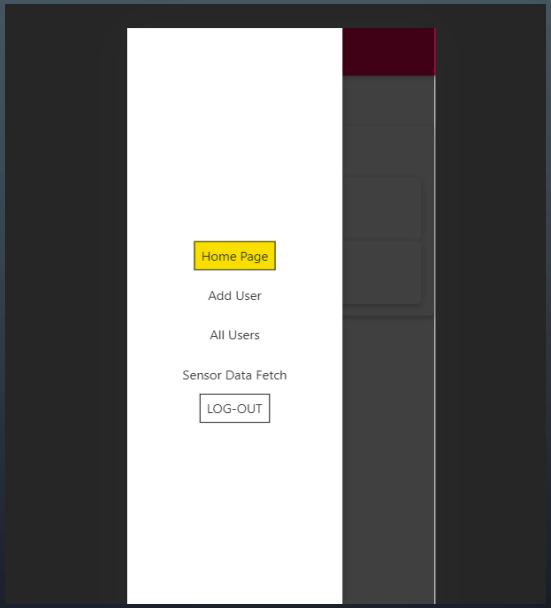


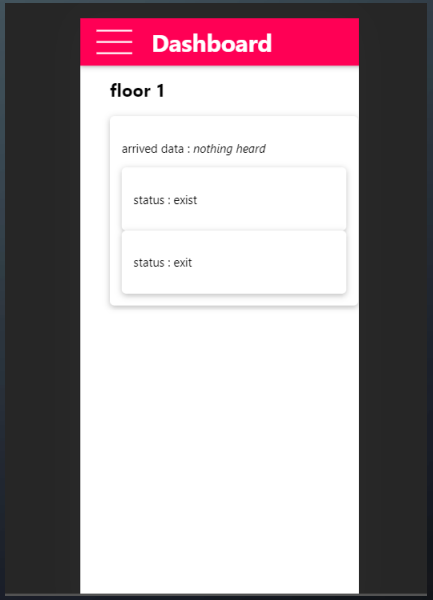
SENSORS CHARTS

- This part of the system all the process is on the server side that returns data in different arrays that indicates the number of entrances inside the enviorment and based on that shows the charts using the chart js library for react native.In the same page of the time filter it does a refresh and shows the data.



- Also there are some datils like this web application of react native is compatible with phone device sizes so it can be turned into an application. There are some animations implemented to make it more real and fluent like an application





SERVER SIDE OF THE APPLICATION

- For the server side of the application as I mentioned before I used node.js for coding it is made of two main parts.
- The rest services for crud and chart system of the application and mqtt part for showing and passing the real time data .
- In this application mongo database is used based on the docker system
- The server side of the application I will express the features inside the videos more but to give an overview I will describe some parts of the code

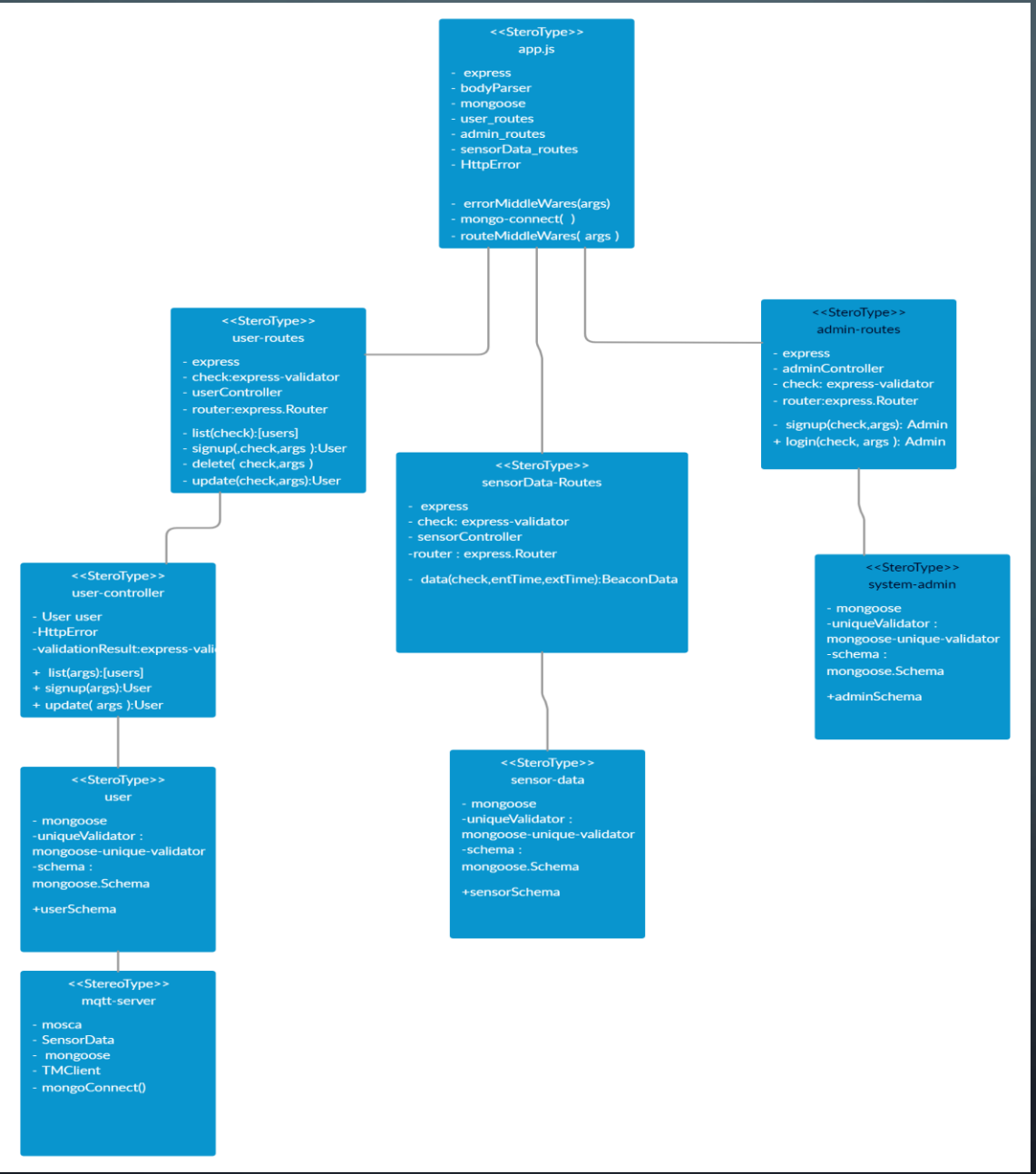
SEVER SIDE OF THE APPLICATION (REST SERVICES)

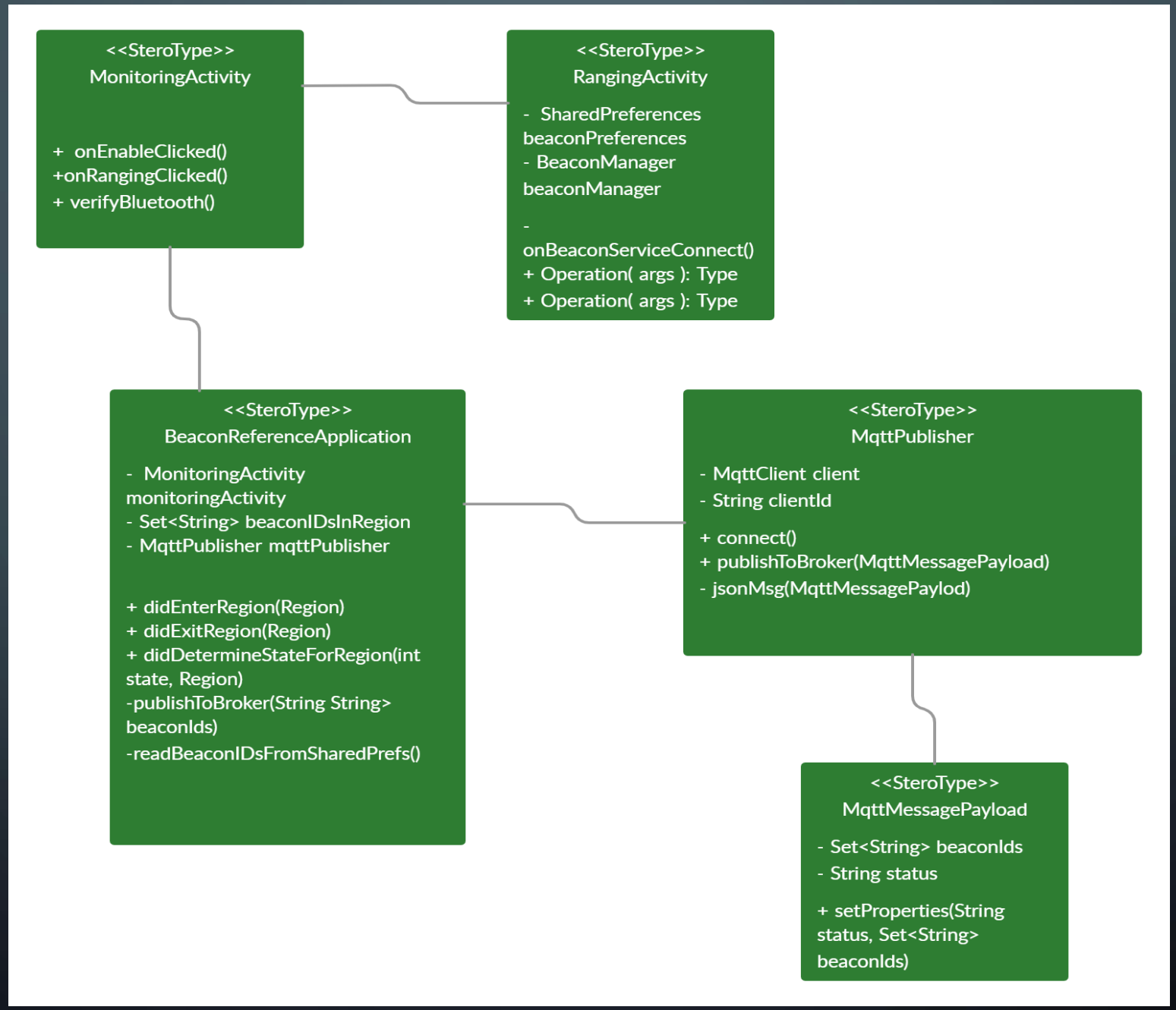
- In the server side I tired to code everything clean in order to learn more about the patterns of coding for server side in node js therefor the rest services part is implemented in three different parts that are connected to each other.
- We have app.js module which starts the server and connects to mongo database
- Each module has it's own router : 1)admin 2)user 3)sensors
- Each router has a controller that implements the logic behind each route so tracing the code will be much easier.
- Utils : For now in the utils part I only implemented a http class for forwarding errors more clean and more readable.
- Models : Also we have models for defining the schemas for the data base collections.

SERVER SIDE OF THE APPLICATION (MQTT PART)

- In the mqtt part of the application we log everything and in the publisher we save the necessary data inside the mongo database . Also there are some logics implemented to avoid saving repetitive data inside database as we can have so much data passed by messages to the mosca mqtt server.
- This part of the system will be explained in details by the videos I provided.


Node.js
React.js
Android
beacons in the zone
beacon app
Add
monitoring beacons
start ranging beacons
there is a beacon with <<id >> 1.5 meters away
using mqtt publisher on andoird
send data to server
mqtt using mosca library
node server gets beacon data
publish data to subscriber
save beacon data to mongo
using mqtt library
Add
react subscriber gets data
existing : <<beacon-id>> exited: <<beacon-id>>
data on home page
admin authentication
login or register admin
access to other parts
register a new user in system
validate data and add user to mongo db
get the list of users
validate data and send data
form provided with validation
here you can delete or edit any user if you are logged in as admin
fetch the sensor data
enter day and time for searching data
search data in database and give response
showing data with chart js
it is a form provided with validation


<<SteroType>>
app.js
- express
- bodyParser
- mongoose
- user_routes
- admin_routes
- sensorData_routes
- HttpError
- errorMiddleWares(args)
- mongo-connect( )
- routeMiddleWares(args)
<<SteroType>>
user-routes
- express
- check:express-validator
- userController
- router:express.Router
- list(check):[users]
- signup(,check,args):User
- delete(check,args)
- update(check,args):User
<<SteroType>>
admin-routes
- express
- adminController
- check: express-validator
- router:express.Router
- signup(check,args): Admin
+ login(check, args): Admin
<<SteroType>>
sensorData-Routes
- express
- check: express-validator
- sensorController
-router : express.Router
- data(check,entTime,extTime):BeaconData
<<SteroType>>
user-controller
- User user
-HttpError
-validationResult:express-vali
+ list(args):[users]
+ signup(args):User
+ update(args):User
<<SteroType>>
system-admin
- mongoose
-uniqueValidator :
mongoose-unique-validator
-schema :
mongoose.Schema
+adminSchema
<<SteroType>>
user
- mongoose
-uniqueValidator :
mongoose-unique-validator
-schema :
mongoose.Schema
+userSchema
<<SteroType>>
sensor-data
- mongoose
-uniqueValidator :
mongoose-unique-validator
-schema :
mongoose.Schema
+sensorSchema
<<StereoType>>
mqtt-server
- mosca
- SensorData
- mongoose
- TMClient
- mongoConnect()


<<SteroType>>
MonitoringActivity
+ onEnableClicked()
+onRangingClicked()
+ verifyBluetooth()
<<SteroType>>
RangingActivity
- SharedPreferences beaconPreferences
- BeaconManager beaconManager
- onBeaconServiceConnect()
+ Operation(args): Type
+ Operation(args): Type
<<SteroType>>
BeaconReferenceApplication
- MonitoringActivity monitoringActivity
- Set<String> beaconIDsInRegion
- MqttPublisher mqttPublisher
+ didEnterRegion(Region)
+ didExitRegion(Region)
+ didDetermineStateForRegion(int state, Region)
-publishToBroker(String String> beaconIds)
-readBeaconIDsFromSharedPrefs()
<<SteroType>>
MqttPublisher
- MqttClient client
- String clientId
+ connect()
+ publishToBroker(MqttMessagePayload)
- jsonMsg(MqttMessagePaylod)
<<SteroType>>
MqttMessagePayload
- Set<String> beaconIds
- String status
+ setProperties(String status, Set<String> beaconIds)