

IoT Dashboard Project

IoT dashboard project created using MEAN stack.

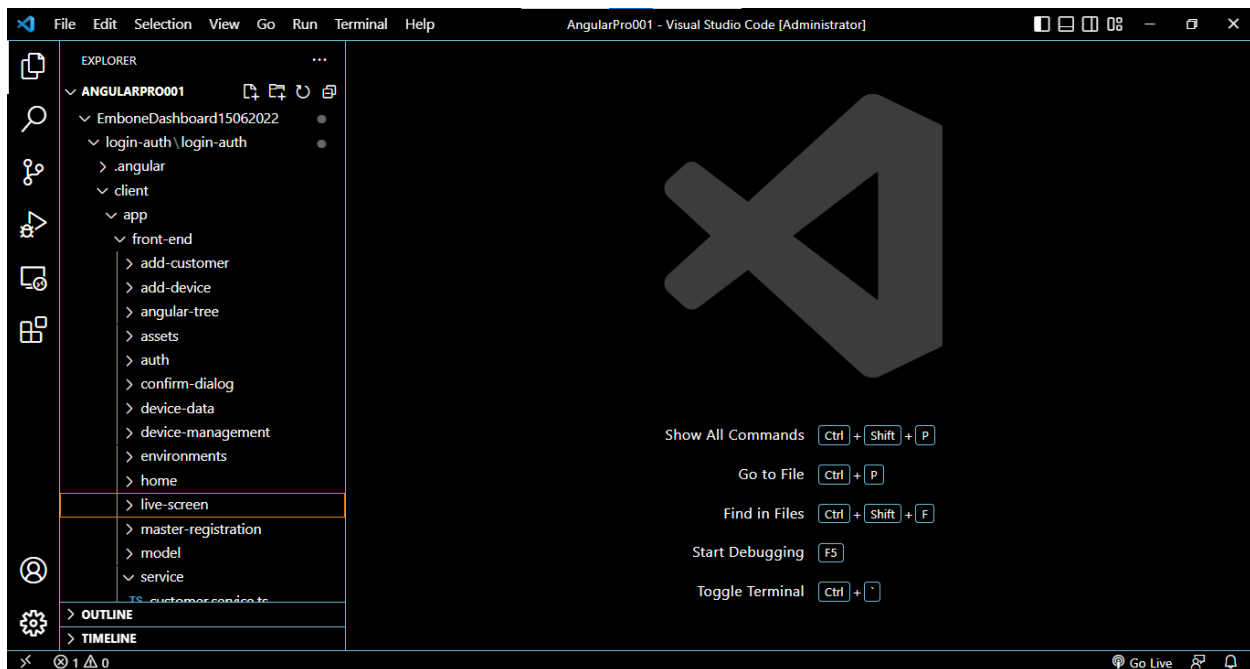
It is based on Angular. Command for adding component in angular :

```
ng g c component_name
```

command for adding service in angular :

```
ng g s service_name
```

Dashboard project Structure's Screenshot :



All components are in app folder.

All component names list :

In front end part component names -

addCustomer, addDevice, angularTree, assets, auth, confirmDialog, deviceData, deviceManagement, environment, home, liveScreen, masterRegistration, model, signIn, signUp, user, userProfile.

In services part services names -

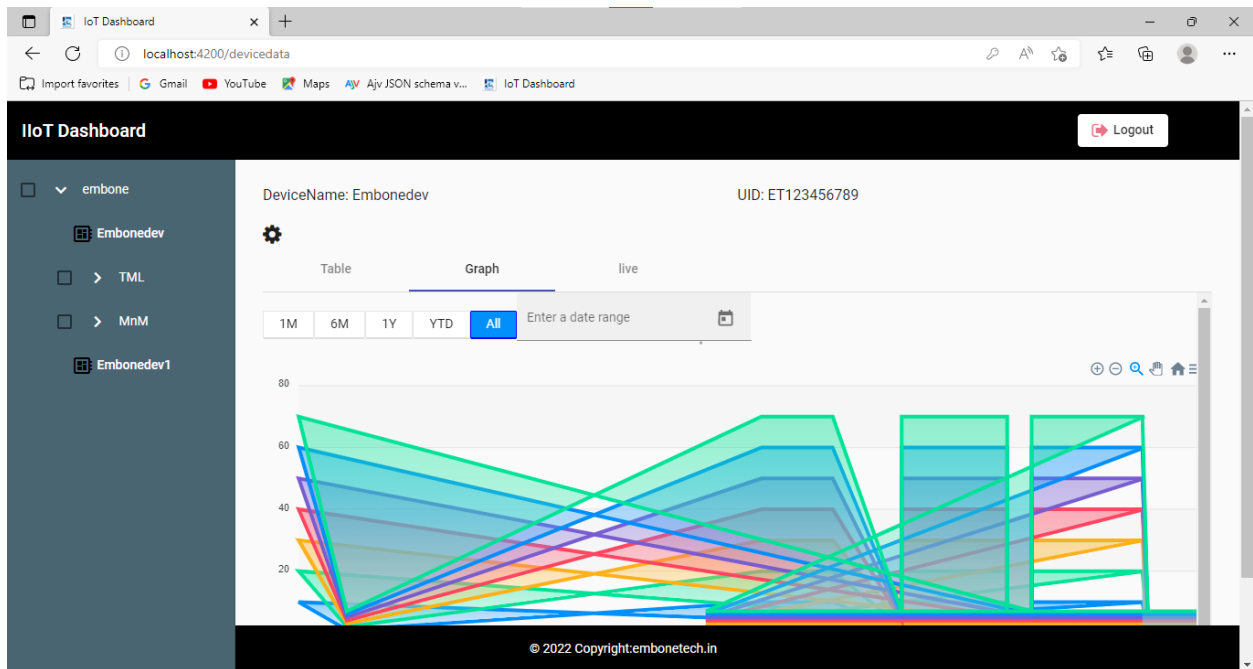
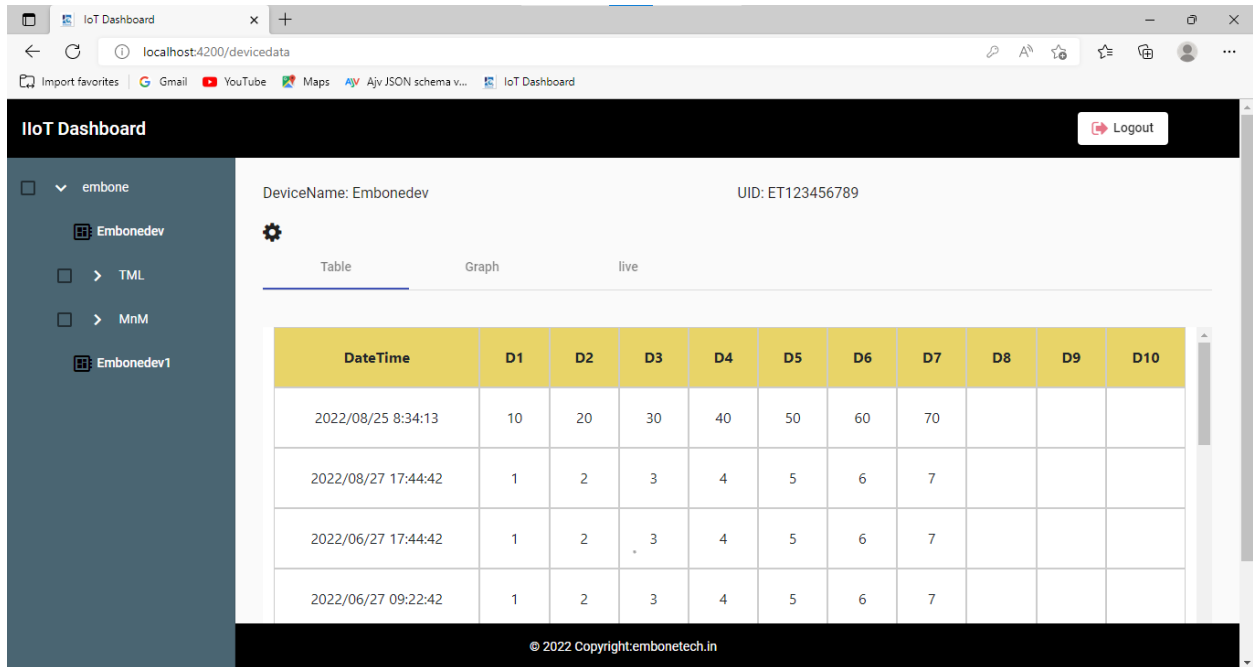
customerService, treeService, deviceManagementService, deviceDataService, liveScreenService.

In backend part controller, models and routes are written.

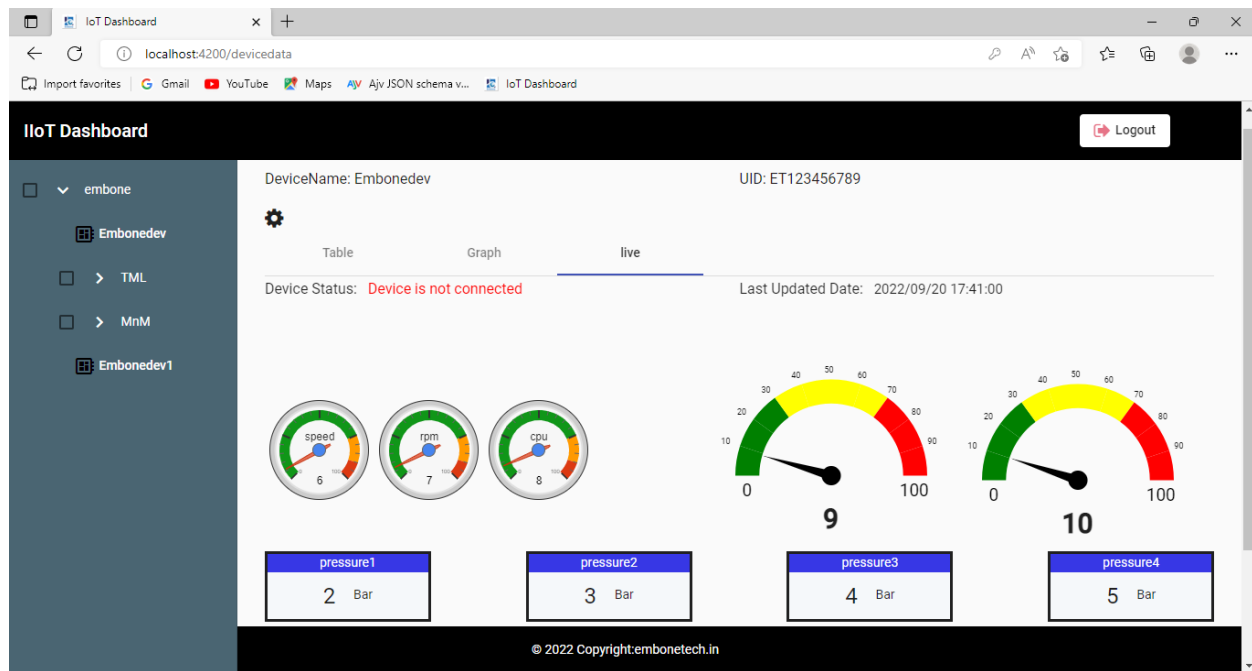
Front End Portion –

I have created deviceData, deviceManagement and liveScreen component. In deviceData component, table and graph is shown. In liveScreen component, real time data display shown. In deviceManagement component configuration table shown.

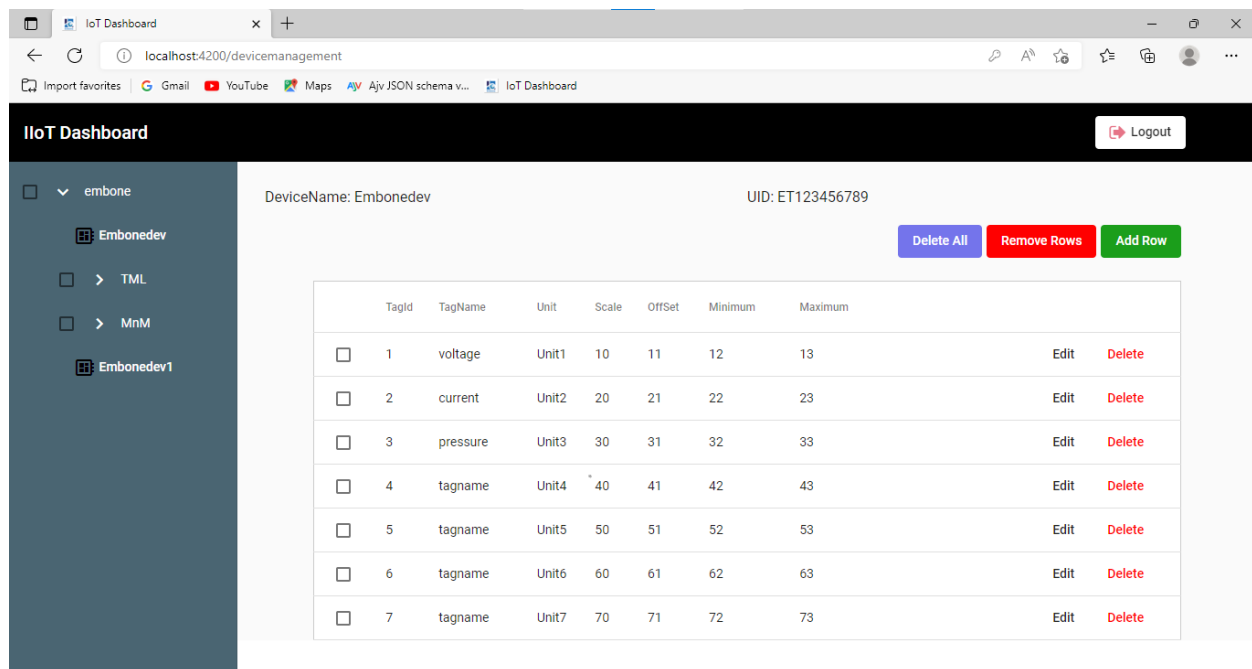
deviceData component's dashboard screenshot :



liveScreen component's dashboard screenshot :



DeviceManagement component's dashboard screenshot :



All angular tree related services written in treeService.

Treeservice code screenshot :

```
77 /** The selection for checklist */
78 checklistSelection = new SelectionModel<ExampleFlatNode>(true);
79
80 treeControl = new FlatTreeControl<ExampleFlatNode>(
81   node => node.level, node => node.expandable); //user can expand and cc
82
83 treeFlattener = new MatTreeFlattener(
84   this.transformer, node => node.level, node => node.expandable, node => node.children
85 );
86
87 dataSource = new MatTreeFlatDataSource(this.treeControl, this.treeFlattener);
88
89 ngOnInitTemp(): void {
90   console.log(" In Treeservice");
91   const token = this.userService.getToken();
92   var decoded = jwt_decode(token);
93   console.log("decoded['path']", decoded['path']);
94   this.login_path = decoded['path'];
95   this.token.push(token);
96   console.log("this.login_path is", this.login_path);
97   this.customerService.postCustomerPath(this.token).subscribe(
98     // this.customerService.getCustomer().subscribe(
99     res => {
100       console.log("response is", res)
101       this.createTreeData(res);
102       this.dataSource.data = TREE_DATA_Temp;
103     },
104   );
105 }
```

Back End Portion –

All backend files are written in javaScript language. Rest API’S are used to communicate between client and server. Rest API’S are written in device.routes.js file.

API’S code screenshot :

```
12 //const deviceManagement = require('...');
13 module.exports = function(app) {
14   app.use(function(req, res, next) {
15     res.header(
16       "Access-Control-Allow-Headers",
17       "x-access-token, Origin, Content-Type, Accept"
18     );
19     next();
20   });
21
22   app.post('/deviceData', validate({body: jsonSchemaValidationObject.deviceDataJsonObject}), c
23   app.post('/addDeviceName', validate({body: jsonSchemaValidationObject.deviceNameJsonObject}), c
24   app.post('/getDeviceNamePath', validate({body: jsonSchemaValidationObject.getDeviceName
25   app.post('/deviceLiveData', deviceDataCTL.deviceLiveData);
26   app.post('/createDeviceManagementData', validate({body: jsonSchemaValidationObject.crea
27   app.put('/updateDeviceManagementData/:id', validate({body: jsonSchemaValidationObject.u
28   app.delete('/deleteDeviceManagementDataById/:id', deviceManagementCTL.deleteDeviceManag
29   app.post('/deleteAllDeviceManagementDataByUID', deviceManagementCTL.deleteAllDeviceManag
30   app.post('/getUID', validate({body: jsonSchemaValidationObject.getUIDJsonObject}), devi
31   /**
32    * Error handler middleware for validation errors.
33    */
34   app.use((error, request, response, next) => {
35     // Check the error is a validation error
36     if (error instanceof ValidationErrors) {
37       // Handle the error
38     }
39   });
40 }
```

To run the front end portion use following command :

ng serve

To run the back end portion use following command :

node app.js

For Backup

For Updating Angular Version I have used below reference link

<https://dalenguyen.medium.com/migrate-angular-v12-to-angular-v13-71ea3b0180b3>

npx is used for locally installation of angular.

Available IoT Dashboard :

<https://cumulocity.com/guides/users-guide/device-management/>

<https://relevant.software/cases/sensor-innovation/>

<https://relevant.software/cases/airthings/>

<https://www.airthings.com/dashboard>

Readings:

<https://lanars.com/blog/web-application-architecture-best-practices>