- **Project Overview:**
  The purpose of the project is to visualize data from the database to the user interface, I used NodeJS as a backend to handle requests and retrieve data from MySQL database, reactJS with TypeScript in the frontend, and react-query to fetch data and handle state management.

- **Why JavaScript Ecosystem?**
  Since the project description did not include any tech stack to make use of, I used JavaScript to build backend and App,
  NodeJS and the JavaScript ecosystem were chosen for the following reasons:

  1. Asynchronous Nature of NodeJS:
     NodeJS is well-suited for handling asynchronous operations, making it efficient for data-intensive apps, also because of the nature of event-driven, non-blocking I/O architecture.

  2. Single Language Across the Stack.

  4. ReactJS for Declarative UI:
     simplifies the development process, enhances code readability, and promotes reusability.

  5. React Query for Data Fetching:
     React Query is a powerful library for managing state and fetching data in React applications. Its simplicity and efficiency make it an excellent choice for handling data fetching and management in the frontend.

- **What You will need to run the application?**
  1- NodeJS (^16.10) must be installed.

- **How to run:**
  1. Run backend:
     A- using the terminal enter the backend folder
     B- run (yarn) command to install any missing dependencies
     C- run (yarn start or yarn run debug) to start backend

  2. Run App:
     A- using the terminal enter the application folder
     B- run (yarn) command to install any missing dependencies
     C- run (yarn start) to start react application

- **Environment Variables:**
  1- Make sure the .env file in the backend contains these data:

```
PORT=
CORS_ALLOW_ORIGIN=
DB_HOST=
```

```
DB_NAME=
DB_USER=
DB_PASS=
DB_PORT=
```

2- make sure .env file in frontend(react app) contains these variables

```
REACT_APP_PLATFORM_URI=
```

- ## Navigating Backend:

Backend API Endpoints:

1. Thermometer Log Endpoint Endpoint:
   GET /thermometerLog Description: Retrieve thermometer log data.
2. Daily Account Balance Endpoint Endpoint:
   GET /dailyAccountBalance Description: Retrieve daily account balance data.
3. Daily Email Log Endpoint Endpoint:
   GET /dailyEmailLog Description: Retrieve daily email log data.

Connection Pool with MySQL Database: file(db/connectionPool)

The backend establishes a connection pool with the MySQL database to efficiently handle database connections. The connection details are specified in the environment variables.

Controller Logic (data.controller.js):

The data.controller.js file contains the logic for handling requests to the above endpoints. It interacts with the MySQL database using the established connection pool.

- ## Navigating ReactApp:

Home Component:(src/pages/home/Home.tsx)

There is a dropdown list that allows the user to select what they want to view. The default selection is AC_Thermometer_Log, The user can select Daily_Account_Balance or Daily_Email_Log. Upon the user's selection, a specific component will be loaded alongside react-query will fetch all the data needed by that component

Table Component:(src/pages/home/component/Table.tsx)

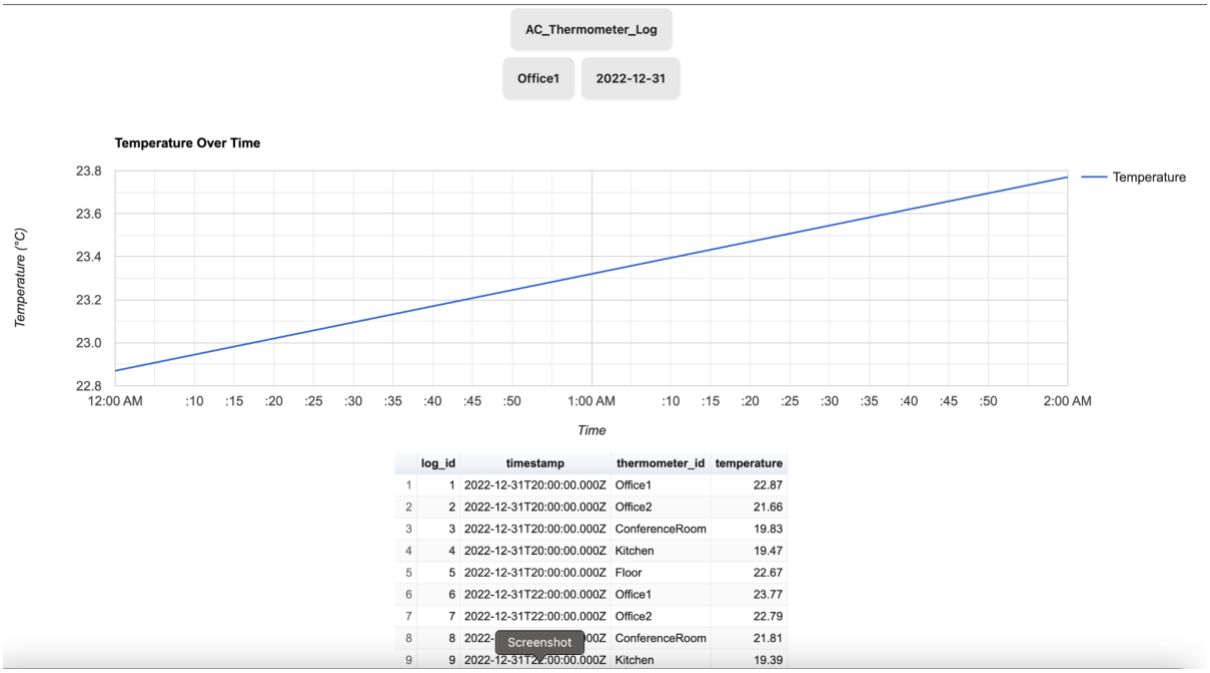
The table component uses Google charts to visualize selected data into a table and can handle different data structures.

ThermometerLog Component:(src/pages/home/component/TempretureLog.tsx)

Visulaize ThermometerLog Data

AccountBalance Component:(src/pages/home/component/AccountBalance.tsx)
Visualize AccountBalance data

DailyEmailLog Component: (src/pages/home/component/DailyEmailLog.tsx)
Visualize AccountBalance data



	log_id	timestamp	thermometer_id	temperature
1	1	2022-12-31T20:00:00.000Z	Office1	22.87
2	2	2022-12-31T20:00:00.000Z	Office2	21.66
3	3	2022-12-31T20:00:00.000Z	ConferenceRoom	19.83
4	4	2022-12-31T20:00:00.000Z	Kitchen	19.47
5	5	2022-12-31T20:00:00.000Z	Floor	22.67
6	6	2022-12-31T22:00:00.000Z	Office1	23.77
7	7	2022-12-31T22:00:00.000Z	Office2	22.79
8	8	2022-	00Z	ConferenceRoom	21.81
9	9	2022-12-31T22:00:00.000Z	Kitchen	19.39

- **What To do next:**
  Things I was up to but didn't find enough time to do:

  1- Implementing JWT Authentication and Castl Authorization to control access to the database.

  3- Implementing error handling in backend.

  2- Refactor the react app to enhance code readability and maintain good practice.

  3- integrate the calendar to filter data in a better way.