Infinite Endurance Technologies

The main function of the website is to provide detailed information regarding various parameters of a machine fed from the Database in a User-friendly way to the Consumer.

The Website is divided into 2 Sections –

* The Client Side (In Client Directory)
* The Server Side (In Root Directory of the Repository).

**The Client Side**- The Client Side or (Frontend) is created using React JavaScript library with CSS for styling. Additional Libraries are also used to enhance the performance and look of the website. Some of the additional libraries are: -

* Chart.Js-For Plotting of Various Graphs (Line, Bar).
* React-Gauge-Chart-For displaying Gauge meter in the Dashboard.
* React-Router-Dom-For Displaying a given Component on a specific route only.
* Query-strings-For parsing information from the URLs.
* Axios-For connecting the Frontend to the Backend of the website via different Post and Get requests.

The Frontend is further divided into different Components: -

1. **App.js-**The root Component of the website where all different Components (sections) of the website are rendered conditionally on the basis route of the website and the Authenticity of the User is verified from the Backend.
2. **Login.js-**The Login Component as the name suggests is used to authenticate the user on the basis of the username and password.

Which is the sent to the Backend as a POST request and receives a user\_id from the Backend and stored in the state to uniquely identify a user along with a token with expiration period of 1 hour. (To prevent the session from expiring on refresh of the Page).

1. **Home.js-**The Home Component Houses the Home Page of the Website which is first welcomed by the title of the Company.

The Home Component acts as a container to the following Components which comprises the complete homepage: -

1. **Machines-** The Machine Components calls the backend with the user\_id as header to retrieve the list of Machines of that particular user with the location of sensors at each

machine as a nested object (as shown in /register routes) to iterate over it and display the dashboard corresponding to it

1. **Dashboard-**The Dashboard contains all the parameters, graphs and gauges required to display functioning of the machines
   * 1. Total Uptime viii. Current Issues
     2. Oil Monitoring System
     3. Past Report
     4. Temperature
     5. Power Consumption
     6. Realtime Vibrational Analysis

The Dashboard also calls the backend with the user\_id and the machine\_id (received from the machines component) as the headers of the get request with the response as the parameters and plot points of the parameters and the data.

After Receiving the data each respective values are compared with their minimum and maximum level and issue is raised if it falls above or below the range.

All the respective grids either contain the values or Graphs and Meters derived from the libraries listed above as a separate component in the Dashboard sub-folder

**Except** the Realtime Vibrational Analysis Grid which has a

Card.js component linking the graphs of the Vibrational Analysis to their respective location in the machine with the help of location\_id (received from the machines component)

1. **Acc\_time.js,** **Freq\_amp.js,** **Acc\_time.js-** These 3 Components are used to display the Realtime Vibrational Analysis Graphs (FFT, Phase, Time History) of

each location of the specific machine. The Components receive the machine\_id and the location\_id from the URL parameters of the address and calls the Backend to receive the X and Y coordinates of the graph as well as Bearing Ratios (BPFO,BPFI,FTF,BSF) and all amplitudes which are >=10\*avg are filtered and their ratio corresponding to the first frequency which meets this condition is calculated and then these ratios are compared with (BPFO,BPFI,FTF,BSF) where margin of (±0.05) and also checks for standard ratios like 4 and 8 with margin of

(±0.1) and changes the property imbalance if present which is displayed in the issues section.

**The Server Side-** The Server Side or (Backend) is create using Node.js, Express (Middleware) with PostgreSQL as the database of the website. Additional Libraries used-

* Bcrypt-For encrypting the passwords.
* Jsonwebtoken-For creating a session of the user.
* Pg-For connecting the backend to the database

The Backend is further divided into different files: -

1. **Server.js-** It is the root file of the backend which integrates all the files, requests and connection to the database.

The file is sub divided into sections-

*Configuration*-All the required packages are imported into the file.

*Middleware’s*-Intercepts all the requests to perform necessary checks

*Routes*-Contains all the end-points of the routes that are called from the front end to receive or send data

In the get and post requests the application uses the database cursor “pool” for all the necessary queries from the backend to the database

Difffrent Routes:-

(/get-data)->Receives the location id,machine id,user\_id and bearing\_number(if provided) and sends the data of the graphs

from the database

(/get-machine)->Receives the user\_id ,machine\_id sends the information of machines to the Machines Component

(/get-machine-params)->Recieves the user\_id and sends the information of machine parameters , daily and hourly avg

data of the machines.

(/update-downtime)->Whenever the power goes down below a certain threshold this route is called to update the downtime of the machine as current time

(/get-downtime)->When the power is above the threshold this route is called with user\_id and machine\_id to send the previous stored downtime to calculate the running time by subtracting the current time from recorded time.

(/record-machine-params)->It receives all the machine dashboard parameters when an issue is raised in the issues section and the user clicks on the “Record Issue” button to record the current machine parameters for our future analysis.

(/register)->This route registers the user along with his/her machine configurations and factory name.

It also receives master username and master password to allow admin only access to this route.

The data format of this Route is as below:

**{**

**"master\_username":"admins@emailid",**

**"master\_password":"admins\_password",**

**"email":"user@gmail.com",**

**"password":"5678",**

**"machines":[{"1":"NTN 16001","2":"","3":"NTN 16009"},{"1":"NTN 16001","2":"","3":"NTN 16009"},{"1":"NTN 16001","2":"","3":"NTN 16009"}],**

**"factory\_name":"ABC"**

**}**

(/login)->This route authenticates the user by checking their username and password from the database and generates a token which is stored in the browser to keep the user signed in for 1 hour.

1. **Db.js-**It configures and sets up the connection between the database and the backend with the help of “pg” package via the connection string provided as a security key provided by the database host.
2. **Jwtgenerator.js-** After successfully validating a user the function in this file with the help of “jsonwebtoken” package helps to create a signed token with a expiration time which is verified every time the page is refreshed
3. **Daily\_Avg.js and Hourly\_Avg.js-** These two files contain a function which is ran periodically by the Heroku scheduler to compute the daily average and hourly average of the machine parameters and store it in a separate table