Assumptions/constraints in your execution

1. I have assumed the Device Id, Device type and location to be strings whereas the timestamp to be in Date-time format.
2. I have assumed the user to have email = name + @GPSTracker.com

Technology concepts used

1. To design the BackEnd in node JS, I have used several open source frameworks like, express js, sequelize, body-parser, bcryptJs, chai, express-validator, jwt, mocha mysql, postman, MySQL Workbench.
2. To design the frontEnd in react JS, I have used several open source frameworks like, react fonts, axios, bootstrap table, react-dom, react-scripts.

Possible enhancements to improve user experience.

1. The UI design could be better.
2. When the data is huge, the pagination execution part should be shifted to BackEnd as it would reduce load.

Thoughts on scalability of the solution.

1. The Solution can be deployed to a real time tracker but with some minute changes as I suggested above.
2. It can easily handle small - mid range loads but to make it efficient and scalable, It needs to be deployed in a Database for real.

BackEnd documentation

The overall flow of any API call in my backend design is

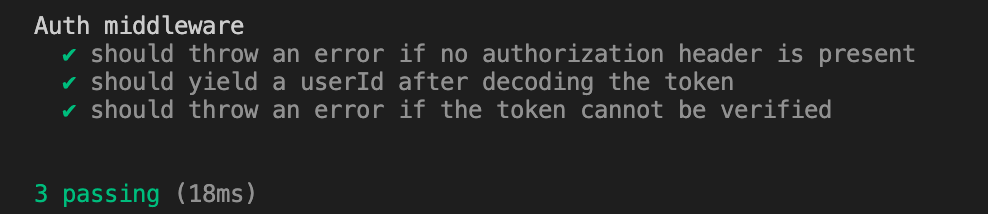
Server -> middleware -> routes -> controller -> model -> config

To start the server write, npm start in the terminal.

1. The signup API ('/signup') : It is a post API request which needs the username, name and password to be set for the particular user. I have applied some basic checks from express-validator like, isEmail and is Password(with min and max length) before the call. The validation is also done in the FrontEnd side so, one can cancel the validations in the backend server. Next, I performed some basic checks and if the user already existed I passed an error message 'UserName Taken' with status code 403. If the user does not exist after that, signup is successful and the table in mysql is updated with the username, name and b-crypted password.
2. The login API ('/login'): It is a post API request which needs the username and password for login. The first check is for the user that, if it does not exist the server returns a message ‘Email not found’ with status code 404. The given password is matched using b-crypt and then the jwt token is created and sent to the frontEnd which is valid for 1 hr.
3. The Get Data API (‘/getData’): It is a get API request which supports pagination and sorting(optional, as I have implemented in FrontEnd directly). I have ensured readability of data after login. To do the pagination on the server(In case when data is very huge), we can pass the params as page and page number as its value which will display the results of that page within limit = 5. If any page is not passed to the server, it takes page as 1 by default. The frontend can also pass sortDeviceId to be true in params to sort the data according to device id. Or, it can pass sortDeviceType to be true in params to sort the data according to device type. The last line of the returned json file shows a page marker, which returns 1-5 of 7 which means displaying 1 to 5 items of total 7 entries. However, For ease I have done pagination and sorting in the frontEnd itself but this is useful for huge data.
4. The Search Data API (‘/searchData’): The frontEnd can also run search Queries on the columns directly but if one wants to implement it in the backEnd, I have created an API for that too. It is a get API request which needs either a device Id or device type in params which the user needs to search for. The API will return all data corresponding to that device Id or type.

UNIT TESTS

1. First, I am testing the is-auth middleware by not providing any authorization header, the test should pass if not authenticated is returned from the function.
2. Second, I am sending a dummy token and the test should pass if it yields a user Id after decoding the token.
3. Third, I am sending a token and verifying it, the test should not pass if the token cannot be verified.



FrontEnd documentation

The overall flow is,

Index -> App -> Login/register -> displayTable

1. I have created a toggle Form to switch between register, login and GPS summary page. It is used to switch between Login and register. If a user is not logged it just switches to register and once it is registered, it switches to login.
2. In login.js, I just give a call to backEnd and check the Email and password and if it is verified correctly, the frontEnd gets a jwt token to work further. I have also applied different error messages for different reasons in case of login failure.
3. In register.js, I can just click on the “Don’t have an account register here” button in the login page to go to the register page. Some necessary checks are applied which when passed stores the user data directly in the database.
4. In the display Table, I have implemented a bootstrap Table which fetches data and displays the response. I have implemented sorting, pagination here(entries per page can be changed in the option given below). One can easily sort the “device Id” and “device type” field by clicking on the column name itself. I have also implemented pagination in the table which can be seen below the table. The data can be made editable by just uncommenting some codes in the display Table.js.