credit-limit-check

Steps/Commands to Test the Assignment

To analyse the implementation of Credit Limit Check assignment, the following commands needs to be executed in the terminal, at the path where the code has been extracted/checked-out:

`npm install`

npm Dependencies

- csv-parse

Dev Dependencies

- chai
- mocha

This command will install required node modules and add the required libraries in node_module folder.

`npm test`

This command run unit tests for main methods used for various operations in the Credit Limit Check.

Note: It is recommended to install mocha globally by running `npm i --global mocha` command, in case it is not already installed globally on your machine. It is required to run `npm test` command successfully.

`npm start csv/sample.csv`

This command will help to test against the **sample.csv** file. This **csv/sample.csv** file is provided in the csv folder along with the Code.

sample.csv file includes the data that is provided in the problem statement and verifies the code against the provided use case. When this command is executed, the following steps are done:

- Parse the data from **sample.csv** file and add it in an array.
- Convert the array Data into required Parent Child JSON/Tree Structure.
- Create the required entity names for related elements.
- Calculate the combined utilisation of entities at each level and Compare it against entity limit.
- Prints the Parent Entity Names and Limits Breached message in **output/report.txt** file as below.

```
 Entities: A/B/C/D:
No limit breaches

Entities: E/F:
Limit breach at E (limit = 200, direct utilisation = 150, combined utilisation = 230).
```

`npm start csv/subentity.csv`

Another csv file **subentity.csv** is added in csv folder to test the **Optional/Bonus Use Case**.

subentity.csv file includes the data to test if the combined sub-entity utilisation also exceeds the limits. After doing the similar steps as for previous csv file, it prints the following output for sub-entities in **output/report.txt** file.

```
 Entities: A/B/C/D:
Limit breach at A (limit = 100, direct utilisation = 0, combined utilisation = 200).
Limit breach at B (limit = 90, direct utilisation = 100, combined utilisation = 200).
Limit breach at C (limit = 40, direct utilisation = 50, combined utilisation = 50).
Limit breach at D (limit = 40, direct utilisation = 50, combined utilisation = 50).

Entities: E/F:
Limit breach at E (limit = 200, direct utilisation = 150, combined utilisation = 230).
```

Error Handling

Few common error scenarios are also been handled and it can be verified as following:

`npm start` // no csv file name passed

If csv file name is not passed along with the command, it will display following error message as *console.log* output.

```
Error : CSV File name not passed while executing the code
```

`npm start csv/noname.csv` // csv passed with wrong name.

If wrong name of the csv file is passed along with the command, it will display following error message as *console.log* output.

```
ENOENT: no such file or directory, open 'file path and name'
```

`Number values are entered wrongly in CSV, as not a number value .`

It will display following error message as *console.log* output.

```
Error: Utilization data in csv is Not a Number'
```

`Number values are missing for some elements in CSV .`

Number values are entered as **0**, If they are missing in CSV