Asynchronous Server Technologies project

DEADLINE: SUNDAY 16TH OF DECEMBER AT 11:55PM

Instructions

This work is part of the continuous assessment of this class and will be the basis for your final project. Your final grade will be calculated based on the final project's result and your Git's history.

The project in itself is a simple web API with a dashboard that should allow you to :

- API side
  - CRUD users
  - Authenticate
  - CRUD your own metrics (make use of an authorization middleware)
- Front side
  - Log in
  - Insert/update/delete metrics once logged in
  - Retrieve the user's metrics and display it in a graph
  - Only access the user's metrics, not the other ones
- Utils
  - pre-populate the database with at least two users and their own metrics

The codebase should:

- be commented when necessary (listing the routes and parameters in the README is ok)
- be unit tested with mocha and chai
- use travis-ci
- have the necessary project files (README, package.json, package-lock.json or yarn.lock; .gitignore, ts-config.json, ...) as shown in class
- not have any file/dir in git that shouldn't be (node_modules, db files, ...)

The project shall be written in typescript, css and ejs and nothing else. You are encouraged to use the tools presented in class.

Notation process

To test your codebase, the following steps will be taken:

- Observe the repo's git history
- `git clone` the project
- Run:
  - `npm install`
  - `npm run populate`
  - `npm test`

    - `npm run dev` or `npm start`
- Play with the front UI
- Test the API to:
    - insert metrics unlogged
    - try reading metrics unlogged
    - login and try reading someone else's metrics
- Read and analyse the codebase

To facilitate the process, don't hesitate to add as much information as possible in the README.md ! And cloning the project freshly + running the instructions as I will might be a good process for final validation...

Also: **DON'T FORGET TO MAKE YOUR REPO PUBLIC**

Grading

The following reasons will instantly equal to zero:

- I detect cheating, usually a repo with full implementation and 1-2 modification commits, I might mail you to discuss it
- You started from the codebase implemented in class and provided as an example

Otherwise:

Subject	Grading
Functional populate DB scripts	+3
Functional sever with exposed front views	+4
User authentication & authorization	+4
Functional unit tests	+2
Front implementation: signin / signup / signout	+2
Front implementation: select a metric and display it in a graph	+3
Front implementation: update / delete a metric	+2
No use of transpilers	-5
No use of GitHub	-5
Missing mandatory project files (per file)	-2
.docx / .pdf file (per file)	-2
Files/dir in git that shouldn't be (per file/dir)	-2

Negative points total will be ceiled to 10 i.e. you can't get more than 10 negative points

Some useful links

- TypeScript documentation: https://www.typescriptlang.org/docs/
- Mocha documentation: https://mochajs.org/
- Chai documentation: https://www.chaijs.com/api/bdd/
- D3.JS documentation: http://d3js.org/

- JQuery documentation: https://code.jquery.com/
- Bootstrap documentation: http://getbootstrap.com/
- Level-up documentation: https://github.com/Level/levelup