

# **ECE Asynchronous Server Technologies**

## **Final Project 2016**

### **About this work**

This final project is the outcome of the work you had to do continuously for this class. Your git's history will be taken into account for the final grade. The project should be done in pairs.

### **Introduction**

Up until now, we have seen the way NodeJS works through modules and packages and the various steps and technologies used to develop an application:

- NodeJS to run a simple HTTP server
- NPM to manage our application and modules
- A framework, ExpressJS, to enhance our simple server and handle all the necessary aspects of a server :
  - Handle routing
  - Manage user auth and sessions with middlewares
  - Expose static content to display in a browser
  - ...
- Transpilers (coffee-script, jade and stylus) to ease writing and reading code

And also a set of best-practices and other tools to enhance the developer's experience.

### **TODO**

Using the code base that you worked on up until now, you should prepare a project meeting the following requirements:

- 1) It should use the transpilers presented in class
  - 1) Coffeescript instead of javascript (code in "src/", transpiles to JS in "lib/")
  - 2) Pug instead of HTML
  - 3) Stylus instead of CSS
- 2) It should have an ExpressJS skeleton with:
  - 1) the following middlewares :
    - body-parser
    - express-session
    - serve-static
    - errorhandler
    - morgan
    - stylus
  - 2) a LevelDB database using the packages "level", "level-session-store", "level-ws", "levelup"

- 3) static files served from a "public/" directory
- 4) pug views rendered on the "/" route
- 5) CRUD modules to:
  - 1) get, save and remove a metric batch in DB, a metric batch should be associated to a user
  - 2) get, save and remove a user in DB
- 6) user authentication and authorization
  - 1) a user should be able to signup/signin/signout
  - 2) a user should be able to access his metrics
  - 3) a user should only access his own metrics and none other
- 3) It should have front views to:
  - 1) signin/signup on the app
  - 2) query the API, display the returned metrics in a graph with d3.js and signout
- 4) It should have unit tests on the CRUD modules
- 5) It should have scripts to start the server, populate the databases and run the tests

The whole code should be updated on github with all the files necessary to meet the open source's best practices. You will upload on campus a .txt file with the url to your git named « name1\_name2.txt » and send it to me by mail at [cesar@adaltas.com](mailto:cesar@adaltas.com) with the object « [ECE] final handling <name1> <name2> ».

Your code should be commented (not too much!).

To be able to test your code, I will need:

1. A script populating the database with dummy users and metrics
2. Instructions on how to test it

**Deadline is Sunday 18 dec. at 23:55PM.**

## Bonus

1. Do the front views to add / edit / remove metric batches

## Notes

- You can use a different database than LevelDB if you want, however you have to keep the same model (user / metrics) and have the class example implemented in your git history before switching for another DB
- You can use a different graph library than d3.js such as chart.js
- You can use a JavaScript framework for the front such as Vue.JS, however you have to have the vue implemented manually such as demonstrated in class in you git history before adding the JS framework

# Marking grid

If there are no instructions, I won't look at your code

The following criteria will be used for notation :

Functional populate DB scripts	4
Functional sever with exposed front views	4
User authentication & authorization	4
Front implementation: signin / signup / signout	4
Front implementation: select a metric and display it in a graph	4

## Malus

No use of transpilers	-5
No use of github	-5
Missing best practices files presented in class (per file)	-2
.docx / .pdf file (per file)	-2
Uncommented code	-1

## Resources

All the slides : <http://github.com/adaltas/ece-nodejs> -> exports/

NPM documentation: <https://docs.npmjs.com/>

Express documentation: <http://expressjs.com/en/4x/api.html>

Coffeescript documentation: <http://coffeescript.org/>

Jade documentation: <https://pugjs.org>

Stylus documentation: <http://stylus-lang.com>

ShouldJS documentation: <http://shouldjs.github.io/>

MochaJS documentation: <http://mochajs.org/>

D3.JS documentation: <http://d3js.org/>

JQuery documentation: [code.jquery.com](http://code.jquery.com)

Bootstrap documentation: <http://getbootstrap.com/>

Level-up documentation: <https://github.com/Level/levelup>