Pomodoro Timer

Kwok Chau Lam

Contents

[Introduction 2](#_Toc526368968)

[Requirement 3](#_Toc526368969)

[Covered 3](#_Toc526368970)

[Not covered 3](#_Toc526368971)

[Design 4](#_Toc526368972)

[HTML Layout 4](#_Toc526368973)

[AJAX 4](#_Toc526368974)

[Node / Express API 4](#_Toc526368975)

[Implementation 6](#_Toc526368976)

[Deployment 6](#_Toc526368977)

[Test and Screenshots 6](#_Toc526368978)

[Troubleshooting 11](#_Toc526368979)

# Introduction

This is a homework assignment for DFS position. Please refer to [requirement](#_Requirement) section for what to build. Fancy (the recruiter) told me that DFS is using Go.Js. That’s why I choose Go.Js for UI presentations. For api layer, I choose to use Node.Js running in server side. With Node.Js, you will see clear separation between UI and API.

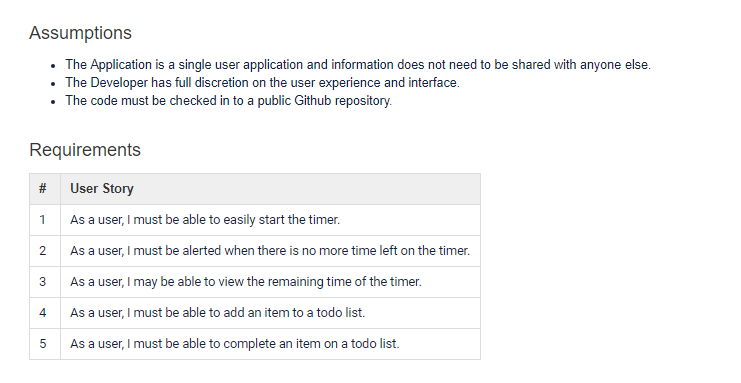
Here is summary of technologies used in this homework assignment:

* HTML
* CSS
* Go.Js
* JavaScript
* React
* Axios
* Node.Js
* Express.Js
* HTTP

# Requirement

## Covered

Here is the requirement coming from Fancy (cut and paste from her email):



## Not covered

The following features, for example, are not part of the requirement and so you won’t find the implementation in this project:

* Update a task in to-do list
* Delete a task from to-do list
* Delete all tasks
* Etc

For testing, please follow test case in [test](#_Test_and_Screenshots) section. Since it is a homework assignment, I only cover functionality and basic error handling test cases in test section. No performance, security, localization and etc testing is covered.

# Design

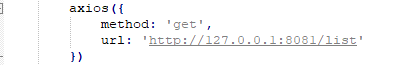
## HTML Layout

The UI has 3 sections:

* Task
  + Add a task
  + Error message when error
* Go.JS List (graph)
  + Display a list of task
  + Color indication for complete or not
* Timer
  + Start timer
  + Display remaining time
  + Alert
  + Error message when error

## AXIOS

It is the technology I choose to connect from UI to Node Js server-side api layer. Here is example of Ajax syntax:



* method defined GET, POST and son
* Url point to localhost with route api

## Node / Express API

Here are apis:

|  |  |
| --- | --- |
| API | Note |
| app.get('/', function (req, res) {}) | Display main html |
| app.get('/list', function (req, res) {}) | Return a list of tasks |
| app.get('/exist', function (req, res) {}) | Check the task existence from the list |
| app.get('/first', function (req, res) {}) | Get the first non complete task from the list |
| app.post('/task', function (req, res) {}) | Add a task in the list |
| app.post('/complete', function (req, res) {}) | Mark the task completed in the list |

# Implementation

The implementation is located at <https://github.com/autlam/pomodoro>\_react.

Here are files:

|  |  |
| --- | --- |
| Files | Note |
| .babelrc | File for React |
| App.js | React components |
| Index.html | Start page |
| Main.css | Styling |
| Main.js | Start JavaScript |
| Package.json | File for React |
| Package-lock.json | File for React |
| Webpack.config.js | Config for React |
| Pomodoro\_timer.docx | Design doc |

# Deployment

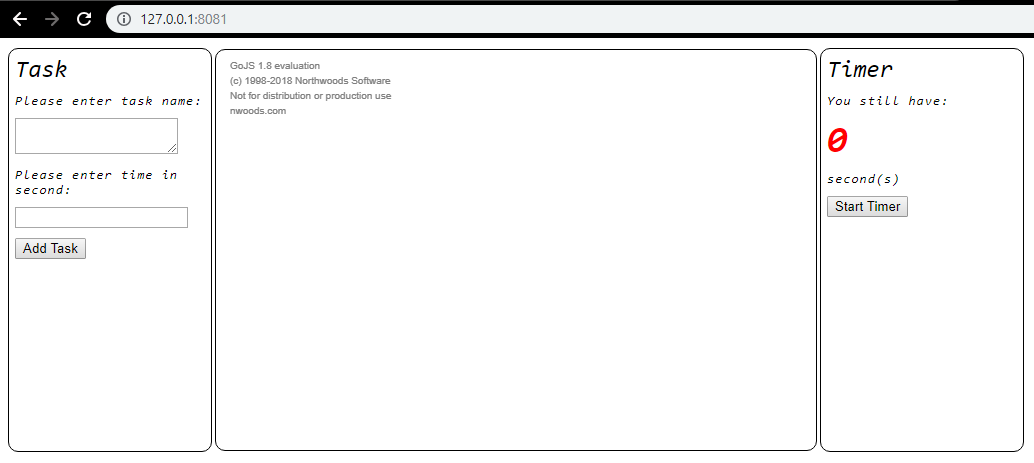
Here are steps to try this prototype:

1. Download NodeJS from <https://nodejs.org/en/download/>
2. Create a folder with name = pomodoro\_react in C:\Program Files\nodejs
3. Go to C:\Program Files\nodejs\pomodoro\_react folder and then follow instruction in [here](https://www.tutorialspoint.com/reactjs/reactjs_environment_setup.htm) to setup all react modules
4. Download and replace all files from <https://github.com/autlam/pomodoro>\_react into C:\Program Files\nodejs\pomodoro\_react folder
5. Open a command prompt as administrator and then type below to install needed modules
   1. npm install express
   2. npm install babel-loader@7 --save
   3. npm install babel-cli babel-preset-es2015
   4. npm install axios --save
   5. npm install react-gojs --save
6. Open node.exe as administrator
7. Type “require(“./pomodoro\_react/server.js”)”
8. In command prompt, go to :\Program Files\nodejs\pomodoro\_react and then type “npm start”
9. ping [autlam@gmail.com](mailto:autlam@gmail.com) if you experience any problem

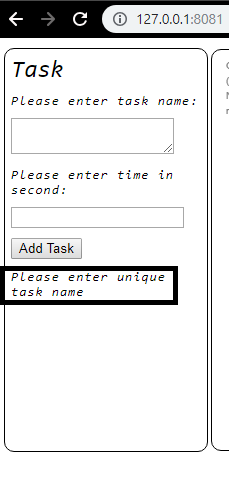
# Test and Screenshots

Here is the test case for functionality and some basic error checking:

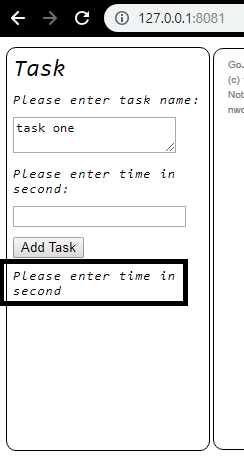
1. Following [deployment](#_Deployment) steps and verify you see below:



1. Click “Add Task” without enter anything and verify you see below error message:



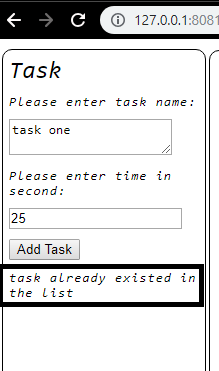
1. Enter “task one” in text area under “please enter task name”, click “Add Task” and verify you see below error message:



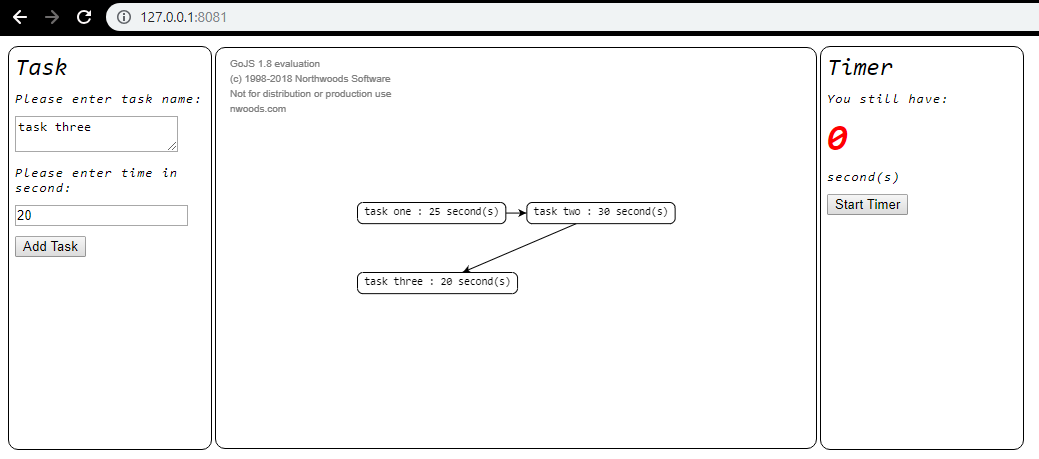
1. Enter seconds i.e. 25, click add task and verify new task text block is added into Go.Js diagram as shown below:



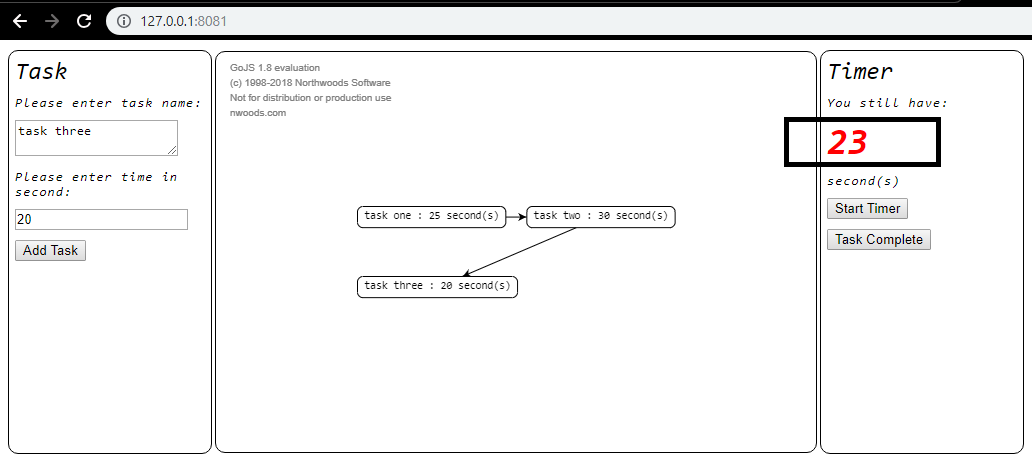
1. Click add task again, verify below error message and no change in Go.Js diagram:



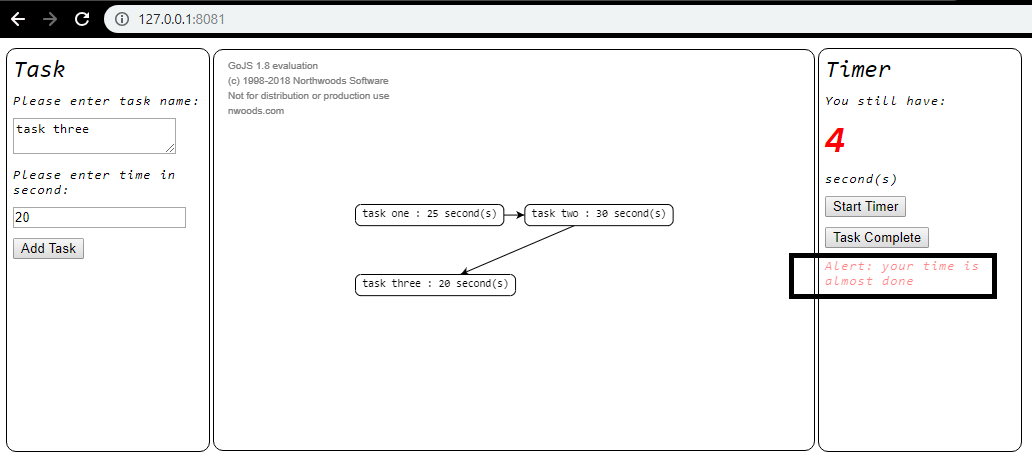
1. Add below tasks
   1. Name: task two, second: 30
   2. Name: task three, second: 20
2. Verify you have below 3 tasks in Go.Js diagram:



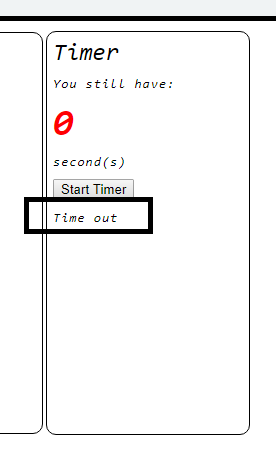
1. Click start timer to track the first task, verify 25 second displayed and start counting down as shown below:



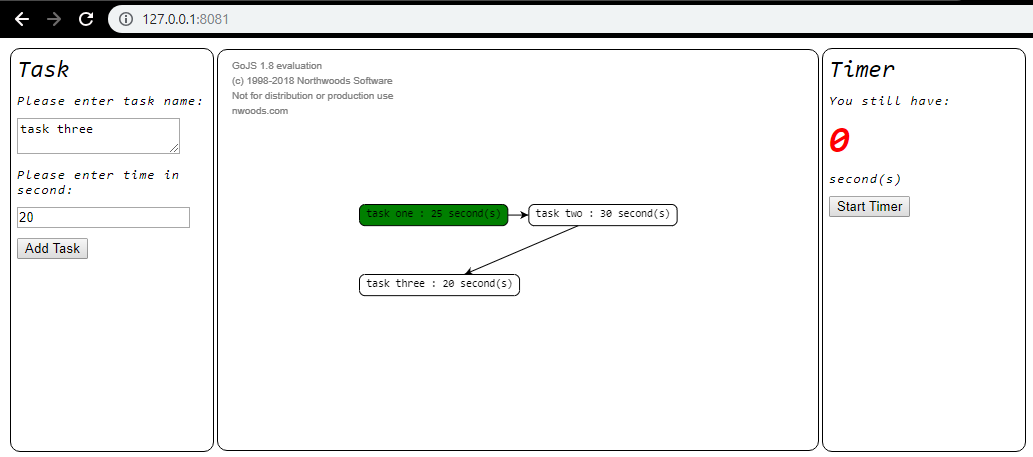
1. Verify below “blinking” alerts on UI when time is less than 5 seconds as shown below:



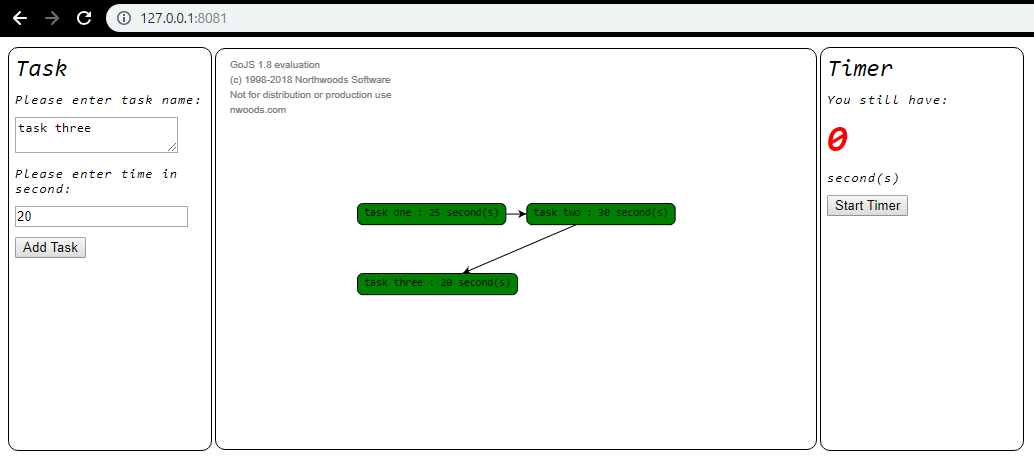
1. Wait all the time is gone and verify below error message:



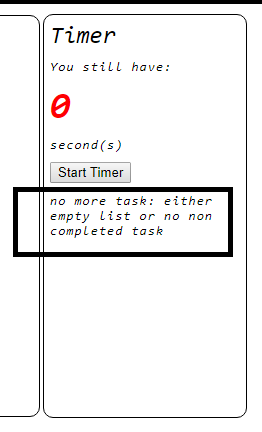
1. Click “Start Timer” again, wait 10 seconds and then click “Task Complete”. Verify the task is marked completed in Go.Js diagram with “green” background as indicator as shown below:



1. Repeat step 11 until all tasks are completed as shown below. Verify timer starts with seconds as inputted before. For example, task two should starts with 30 seconds and task three should start with 20 seconds.



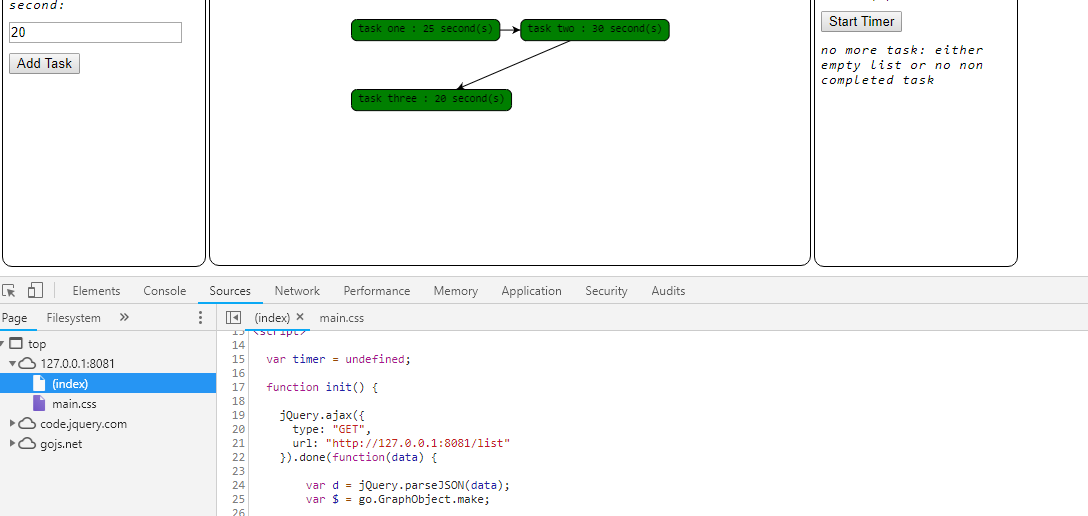
1. Click “Start Timer” again and verify below error message:



# Troubleshooting

Here are tips to troubleshoot:

For client side, use F12 debugger as show below:



For Node.Js Api, read below article and debug in chrome

<https://medium.com/the-node-js-collection/debugging-node-js-with-google-chrome-4965b5f910f4>

you should see below for debugging api via chrome devtools:

