Applied Internet Technology

Final Project

Final Project

Final Project, **Final Milestone Due ~~4/26~~** 4/29

Earlier milestones due throughout April

Overview

Create a **small** web application using Express and MongoDB. Build the application incrementally over the course of 4 weeks.

Project Requirements

Requirements

- You must use Express and MongoDB (or other server-side framework and database with permission)
- You must write your own code, with annotations/references added for any code sourced from books, online tutorials, etc.

Grading Rubric

Completing the milestones leading up to the due date is required! Milestones 1 through 3 are worth over half of your final project grade.

- (20) Milestone #1 - requirements, draft data model, and a skeleton application
- (20) Milestone #2 - deployment attempt and a single working form (**You cannot change *your idea* for your final project after this**, but you can still make minor modifications)
- (20 points) Milestone #3 - two working forms and proof of work on research topics
- (40 points total) Completed project
  - (12 points) minimum 3 x forms or ajax interactions (**excluding login**)
  - (6 points) minimum 3 x any of the following (can be the same):
    - es6 classes that you've written yourself (using the `class` keyword)
    - Object.create (where prototype matters)
    - original higher order functions that you've written yourself
    - or use any of these built-in higher order functions found in `Array.prototype`: `map`, `reduce`, `filter`
  - (2 points) minimum 2 x mongoose schemas
  - (8 points) stability / security
    - simple validation on user input to prevent application from crashing
    - doesn't allow user input to be displayed unescaped directly on page
    - pages that require authentication cannot be accessed without authentication
    - data specified as private to a user cannot be viewed by another user
    - etc.
  - (4 points) *originality*
    - is not mostly based on existing homework
    - majority of code is not from online tutorial
  - (8 points) worth of research topics; see below

Additional Requirements, Your Choice

Choose at least **8 points** worth of these following topics (research and implementation). **This list may change slightly (added items, adjustments to points) as project ideas come in.**

- (3 points) Unit testing with JavaScript
  - Jasmine (http://jasmine.github.io/)
  - Mocha (https://github.com/mochajs/mocha)
  - Any others found through research

    - Minimally 4 tests
    - You'll have to link to testing code in repository
    - … and show a screen capture of tests
- (5 points) Automated functional testing for all of your routes using any of the following:
    - Selenium (http://www.seleniumhq.org/)
    - Headless Chrome (https://developers.google.com/web/updates/2017/06/headless-karma-mocha-chai) - headless browser testing
    - Minimally 4 tests
    - You'll have to link to testing code in repository
    - … and show a screen capture of tests
- (3 points) Configuration management
    - nconf (https://github.com/flatiron/nconf)
    - Node convict (https://github.com/mozilla/node-convict)
    - Any others found through research
- (3 points) Use grunt (http://gruntjs.com/), gulp (http://gulpjs.com/), webpack or even make (!) to automate any of the following … must be used in combination with one or more of the other requirements, such as:
    - (2 points) Integrate ESLint / JSHint / JSLint into your workflow
        - Must be used **with build tool** (see above requirement on Grunt or Gulp
        - Must have have configuration file in repository
        - Must run on entire codebase **outside of** `node_modules` **automatically on file change** (*watch* for changes to the file system)
        - Must link to relevant lines in build configuration and lint configuration
        - Must show screen capture / animated gif of running on save
    - (2 points) Concatenation and minification of CSS and JavaScript files
        - Must be used **with build tool** (see above requirement on Grunt or Gulp
        - (Only client side files!)
        - Only minify and concatenate client side JavaScript
        - Must link to relevant lines in build configuration and mark-up (to show included css)
        - Must run automatically on fille change
        - Must show screen capture / animated gif of running on save
    - (2 points) Use a CSS preprocesser
        - Sass (http://sass-lang.com/)
        - Less (http://lesscss.org/)
        - Myth (http://www.myth.io/)
        - Must link to relevant lines in build configuration and directory of *unprocessed* CSS source
        - Must run automatically on fille change
        - Must show screen capture / animated gif of running on save
- (3 points) Perform client side form validation using custom JavaScript or JavaScript **library**
    - errors must be integrated into the DOM
    - the following will not receive full credit:
- (2 points) Use a CSS framework throughout your site, use a reasonable of customization of the framework (don't just use stock Bootstrap - minimally configure a theme):
    - Bootstrap (http://getbootstrap.com/)
    - Foundation (http://foundation.zurb.com/)
- (1 - 6 points) Use a **server-side** JavaScript library or module that we did not cover in class (not including any from other requirements)
    - assign a point value to the library or module that you're using based on amount of effort and/or code required for integration
    - Must link to source code relevant to implementation and evidence of working implementation on site
- (1 - 6 points) Use a **client-side** JavaScript library or module that we did not cover in class (not including any from other requirements)
    - assign a point value to the library or module that you're using based on amount of effort and/or code required for integration
    - for example, angular 2 or d3 might be 6 points, while google maps might be 1 point
    - Must link to source code relevant to implementation and evidence of working implementation on site
- (1 - 6 points) Per external API used
    - assign a point value to the library or module that you're using based on amount of effort and/or code required for integration
    - for example, angular 2 might be 6 points, while google maps might be 1 point
    - Must link to source code relevant to implementation and API documentation

Milestones

Due 4/5 at 11PM - Milestone 1 - Requirements / Specifications, Draft Data Model, Skeleton Application (20 points)

Check out sample documentation (https://github.com/nyu-csci-ua-0480-008-spring-2017/final-project-example)

- Documentation
  - Submit electronically through a supplied GitHub repository
  - Write everything up in your README.md
    - Drop the images into your repository (either under a separate branch or in a folder called documentation)
    - Link to it based on this SO article (http://stackoverflow.com/questions/10189356/how-to-add-screenshot-to-readmes-in-github-repository)
  - A one-paragraph description of your project
  - Requirements
    - Sample documents (JSON / JavaScript literal objects will be fine, or your actualy Schemas) that you will store in your database, and a description of what each document represents
    - Enumerate any references from one document to another
  - Wireframes (like this one (http://upload.wikimedia.org/wikipedia/commons/4/47/Profilewireframe.png))
    - a great article on wireframes (http://www.onextrapixel.com/2011/03/28/creating-web-design-wireframes-tools-resources-and-best-practices/)
    - some possible tools
      - Hand-drawn
      - Balsamiq
      - Google drawings
      - Omnigraffle
      - Adobe tools such Photoshop (psds), Illustrator (ai) etc.
  - A Site Map (see examples) (http://creately.com/diagram-community/popular/t/site-map)
  - One of the following to represent what your application will actually do:
    - A list of user stories (simply a list of sentences in the form of *as a <type of user>, I want <some goal> so that <some reason>* (http://en.wikipedia.org/wiki/User_story#Format))
    - A list/spreadsheet of use cases (see the end of this article) (http://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases/)
    - A Use Case Diagram (https://www.andrew.cmu.edu/course/90-754/umlucdfaq.html)
  - **Which modules / concept will you research?**
    - List of research topics
    - A brief description of concept (3 or 4 sentence each)
      - What is it?
      - Why use it?
      - List of possible candidate modules or solutions
      - Points for research topic (based on specifications above)
- Code
  - A skeleton express app
    - Start populating your package.json with required modules
    - **It's ok to just have boilerplate code and no route handlers!**
  - A 1st draft mongoose schema

~~Due Date 4/12 at 11pm~~ **Due Date 4/13 at 11pm** - Milestone 2 - Initial Deployment and First Form (20 points)

1. your server and port name can be accessed through a link in a piazza post for milestone #2
2. attempt to deploy your code to Courant's servers by following instructions (homework/deploy.html)
3. use this form to submit your deployed site (https://docs.google.com/forms/d/e/1FAIpQLSeDK0kCfwzhpZbEW1XSWq1sT1WIinZdZJ8Q8KjQiB5vTFKh9Q/viewform)
4. your submission won't be graded unless the form above is sent with urls to your deployed site
5. your deployed site should contain the following progress:
   - **one working form (that is not login or registration)**
     - …that should allow data to be modified or deleted
     - the results of submitting this form should be apparent/viewable
   - show *some* progress on at least 1 of your research topics, **it doesn't have to be fully working… stub code from documentation or pseudocode is adequate**
     - consequently, a link to the github repository / line no that shows any proof of work is sufficient
     - or, if it's something that's already visible, a link to the a page on your site that's deployed to the server

Due Date 4/20 at 11pm - Milestone 3 - 2nd Form and More Progress on Research (20 points)

1. make at least 3 additional commits to add:
    - your 2nd form / ajax interaction
    - make more progress on your research topics
2. **update youre README.md to reflect any changes to what your final project is** (you cannot change your project idea after this point, but you can make modifications to your research topics)
3. redeploy your code to Courant's server by running git pull and restarting forever (**do not do this until you receive your milestone #2 grades**)
    1. `ssh` into linserv1 or linserv2 (remember, you have to go to access.cims.nyu.edu first)
    2. `cd` into your project directory (should be in `~/opt/NETID-final-project`)
    3. run `forever stopall` and `forever start bin/www`
        - you'll have to use the full path to forever, likely `~/usr/local/node_modules/bin/forever`
        - and perhaps the full bath to `bin/www`
4. **fill out form to submit assignment (https://docs.google.com/forms/d/e/1FAIpQLSd720a6Hdlc8Ok_raL4VAk2p5C9sfQVXHZmcaSbq30QJ8u67w/viewform)**; it will contain:
    - **both working forms or ajax interactions**
    - a link to show code changes since milestone #2:
        - start with the url to your repository: `https://github.com/nyu-csci-ua-0480-008-spring-2019/NETID-final-project/`
        - and append the following to the url: `compare/master@%7B04-20-19%7D...master`
        - for example: `https://github.com/nyu-csci-ua-0480-008-spring-2019/NETID-final-project/compare/master@%7B04-13-19%7D...master`

Due Date ~~4/26~~ 4/29 **at 11PM** - Final Project Complete and Code is fully *Deployed* (40 points)

- **all commits must be in by Monday, April 29th**
- **project must be deployed** on cims servers (or other platform, such as Heroku, gomix, zeit, etc.)
    - if your application needs to be restarted while being graded; I will contact you
    - you will not receive a penalty for restarting after the due date
- **the final project form submission (https://docs.google.com/forms/d/e/1FAIpQLSfpl94OS-dQkjFpPgVuYyJm2Yo7-kmcUPEX6qF97PmH25f_Ww/viewform) must be filled out** (if a form is not submitted, you will receive a 0 for your project)
- **Research Topic Notes**
    - if you require a **specific user to be logged in**, please add the username and password in the form submission above
    - if you used **unit testing** or **functional testing**, upload a screen shot or an animated gif of your tests running to the documentation folder of your project; link to it in your form submission
    - if you used **grunt**, **gulp**, or **webpack** … or some if you used a pre-processor like babel, sass, etc. … link to the relevant configuration file in your form submission
    - if you are using **facebook login**, and your application is in *testing mode*, add this user: `Eef Aqua` so that graders can test your application