Final Project Requirements and Overview

It's the final countdown! With two projects under your belt, you're ready to embark on something BIG.

The objective of this project is to:

- Apply knowledge and skills learned throughout the course
- Push yourself further, and try something ambitious
- Practice Behavior-Driven Development (BDD) / Test-Driven Development (TDD)

Ideally, you'll use the final week of the program to venture into new territory -- go deeper with advanced Rails or Angular topics, explore new technologies, try out libraries and APIs that you haven't used yet, utilize different testing frameworks, etc.

You may work in teams of up to three students, but we'll be expecting more challenging projects out of teams than individuals. Teams will be held to the same standards of project comprehension as in Project 2 -- each individual should be able to speak confidently and accurately about how the group's app worked, and note contributions from teammates.

If you'd like to augment a previous project (for example, your project 1 or project 2), you must build out a *different* aspect of that app, from scratch. Possible options include building an admin dashboard for an earlier project that was entirely client-facing, or incorporating a significant AngularJS feature set in a project that did not previously have it. Students who pitch extensions of earlier projects will be expected to speak to the substantial effort involved in the Project 3 version of the app.

Timeline

Projects launch on Thursday, December 11th and conclude on Friday, December 19th.

You'll meet with the instructors privately on Wednesday morning December 10th to assure that your project is in scope. You will need to have a 'pitch' for your project on December 11th to explain to the class what you would like to build.

On December 19th, you will present the project to the class, showing both the end results and code. You should plan on taking no more than 20 minutes for this, with a few minutes for questions by classmates and the instructional team.

Requirements

Make sure to do all of the following with your app. These core requirements are less restrictive than the past two project requirements, but they're still very important.

Technical requirements:

- Custom backend (Rails**) with storage to a database (MongoDB, PostgreSQL, or something else, but you must store some data in some manner!)
- Use BDD and/or TDD to produce a minimum of 10 tests
- Produce user stories and scenarios
- Track progress with Trello or Pivotal Tracker
- Deploy your app to Heroku (or another server)
- Complete all project deliverables (specified below)

** Students must get instructor approval before using other technologies/frameworks in a final project. All project pitches must involve a web application (Sorry, no iOS/Android for your final project! You may deploy to Phonegap but it will not be used as part of the evaluation of your app).

Extra Credit:

These are for extra credit! Keep in mind that we're going to be looking for your core requirements first.

- Solve a real problem (do research, have users, make something that isn't a novelty app)
- Use a technology not taught in class (Node, Ember, Meteor, etc**)
- Teach a lesson module on a skill or technique you've mastered (multiple-file uploads, kanban, etc)
- Make more than one final project (bonus: make yourself a portfolio site to show off your work!)

** If you make another app using a different stack (for example, if your main Project 3 is on Rails but you also make a Node mini-project), you are welcome to demo both.

Planning and Deliverables

Project plan deliverables:

- Due Dec 10 - Discuss your idea with the instructors
- Due Dec 11 - Pitch to the class
- Due Dec 12 - Create a GitHub repo with a README and add stories to Trello or Pivotal Tracker

Completed project deliverables:

- Due Dec 19 - Final deployment
- Due Dec 19 - Link to source code, wireframes, ERD on GitHub
- Due Dec 19 - Trello or Pivotal Tracker link

Things to Keep in Mind

Make sure that your code is:

- Tested. Be thorough and smart about your testing; write tests first and code second.
- Well-formatted. Are you indenting well? Can we find the start and end of every div, curly brace, etc?
- Well-commented/named. Will I understand what is going on in each block or function?
- Clear. Do your naming conventions make sense? Would another developer be able to look at your app and understand what everything is?
- Deployed and on GitHub. It's required for this project.

We'll also be looking at:

- Planning. Did you think through your project and try to solve a real/tangible problem?
- Documentation. Is there a README? If your app is very robust or complicated, or can be built upon by other developers, is there a wiki?
- Quality of decision-making. Can you stand behind your design and coding choices? Can you think of future improvements to your app?
- Your ability to pick up new technologies and push yourselves. What did you do that took you out of your comfort zone?