Capstone: Workflow Outline

Liam Wynn

2/19/2019

Version 2

Purpose Of This Document

This document will more or less lay out two things: writing code/working on the project, and what to do about documentation. I'd like to take this opportunity to note that this is the first version of this document! It will be a work in progress, whose final form should only manifest when we actually start writing code, which will be the Spring Term of 2019.

This document is split up into several sections that specify the rules for handling various processes. I will also attempt to justify why those rules are in place. If I miss anything, or you want to discuss this document in any capacity PLEASE DO SO. This is a learning experience for me, as this is my first foray into DevOps and whatnot. Do not be afraid to be brutally honest with me. I may not act like I want to hear something, but in the long run it will be good for me. It's like vegetables: you don't think you want them, and they can suck, but it will make you happier in the long run.

One more thing: this version is not laid out in any particular order. As I put it together, I will hopefully be able to better organize it.

Where will our project live?

This is perhaps the most important question we can ask. The answer will be of course a git repo, which shall be hosted on Github. The link to this repo is here: LINK GOES HERE.

How is this repository organized?

I can't give a clear answer to this entirely. Each team member may organize their branches as they see fit. With this said, I will have one requirement: THE MASTER BRANCH IS WHERE PRODUCTION CODE GOES. Let me repeat that:
THE MASTER BRANCH IS WHERE PRODUCTION CODE GOES.
At some point, we will have a place to put the "final" version of our code. That will be the master branch.
Essentially the repository will be organized like this:

- The branch where your code will first live will be the "developer" branch. This is where experiments, dummy code, works in progress, and other "messy" code is.
- Next there will be an "integration" branch. This is where unit tested code that hasn't been fully integrated goes. We will do integration testing on this branch.
- Finally when our code is integrated, we can put it in master.

I forgot to mention this but **THE MASTER BRANCH IS WHERE PRODUCTION CODE GOES.**

When can I put my code in the master branch?

At some point, we'll have to put something in there. I think my requirements for this will be as follows:

- The code you wish to put in master fully implements a feature.
- Unit tests cover at least 80% of the code.
- At least one other member of the team has reviewed your code.

Once this is checked off, we can put it in integration. Once integration testing is done, we can put it in master.

How should documentation be handled?

Generally speaking, there are two kinds of documentation: code specific, and non-code specific. Code specific obviously can come in the form of comments. If you find that comments are insufficient for explaining how something works, please use the Git repo wiki pages. The Wiki pages are great for explaining

higher level code specific stuff. For example, if you want to explain how a system works over all, that's a great place for it. Then comments for individual components of that system.

If you want to write some documentation on how Spring Boot works, put that somewhere else. That "somewhere else" I think will be a collection of documents on Google Drive I think. As of writing this, I've not set that up yet.

If you are unsure where a document ought to go, feel free to ask me.

What if I don't like this?

That's fine. I realize not everyone will agree with the rules here. I don't think they need to be set in stone until the next term, when we start implementing the project. If you don't like something, or want to make a comment, PLEASE DO SO. I cannot emphasize this enough. As I said above, this is a learning process for everyone involved.

Conclusion

I hope this document lays out the basics for getting this project done. The summary of everything is as follows:

- The master branch is for production code only.
- As a corollary, your code will be in a seperate branch.
- To put your code in integration, it must meet three requirements:
  - Unit tests cover 80% of your code.
  - Your code fully implements some kind of feature.
  - Your code is reviewed by at least one other member of the team.
  - The above three put your code in integration. Once the code is fully integrated and as bug free as we can make it, we can put it in master.
- Documentation that is code specific should either be in the comments or a wiki page.
- Documentation that is not code specific should be in a shared Google drive.

Please feel free to send me any questions or comments you have.