Team 29: Pomato

CI/CD Pipeline - Phase 1

We have implemented a pipeline to streamline our development process which is broken up into two stages. Stage one is run any time someone pushes to a remote branch. It consists of a linting workflow using ESLint with the Airbnb style guide, followed by a unit testing workflow using Jest, a popular Javascript testing framework. Stage two is run any time a pull request is opened for a branch into the main branch. It consists of a manual code review via human, where another developer will review the changes and either approve them or suggest changes if there are mistakes. If the pull request is reviewed and approved then we have a final JSDocs workflow which will automatically update our documentation before merging the branch into main.

The first step in stage one is to run ESLint. The styling that we are using for Javascript is the Airbnb style guide. GitHub now has built in ESLint support in their workflows, so we wrote a workflow yaml file that installs all needed files to run ESLint on the GitHub server. We also included an eslintrc.js file where we include the Airbnb extensions and any additional constraints we choose to impose on the linter. If the linter catches any errors in the code it will show the file and the lines where the errors occur and will display a failing icon, notifying the developer that their code is not properly styled. The errors listed out are examples of the errors we have already seen the linter point out to us or even fixed for us.

The second step in stage one is to run our unit tests via Jest. This is something that we don't currently have working, but are already in production of finishing. The main reason we didn't have it working was because we wanted some code written to know what tests we should be running. Now that we have more code written we are just trying to finalize the actions. We have pseudo code written and we have done lots of research about GitHub actions. The final step

is to write it and see if it works. This process works best if we modularise the testing process by writing unit tests for each individual function we implement. The goal is if each method is working as intended and this is verified by all test passing, then when we have a functional product, the website should work as intended. If both steps in phase one are working the programmer can either continue to make changes in the branch or, if the intended task for the branch has been completed, they can open a pull request to the main branch.

Phase two of our pipeline is initiated off of pull requests. The first step is to have one of the other programmers review the branch which we are attempting to merge, looking through all new commits, checks, and modified files. If the reviewing developer believes everything is up to par, they can approve the pull request and leave any additional comments. If there are any errors or maybe more comments in the code that need to be implemented the reviewer can deny the pull request. In this latter case the reviewer will add comments saying what needs to be fixed before the pull request will be approved. This part of our pipeline is already in place, as GitHub has a native feature allowing for branch locking and approval requirements.

Step two for phase two is to use JSDocs to automatically generate any updated documentation and update the overall docs with this new documentation. We will make a workflow that uses the JSDocs commenting convention adopted by our developers to automatically generate documentation files based on comments left within the codebase. These documentation files state the purpose of each class or function we implement and their intended uses. This is not currently running, but we hope in the next week or two we can have this up and running. The first sprint for us focused on linting and human code review, whereas for our second sprint we plan to implement the testing and documentation workflows. In a (potential) third sprint we may implement CD as well. We think this is adequate for our application.

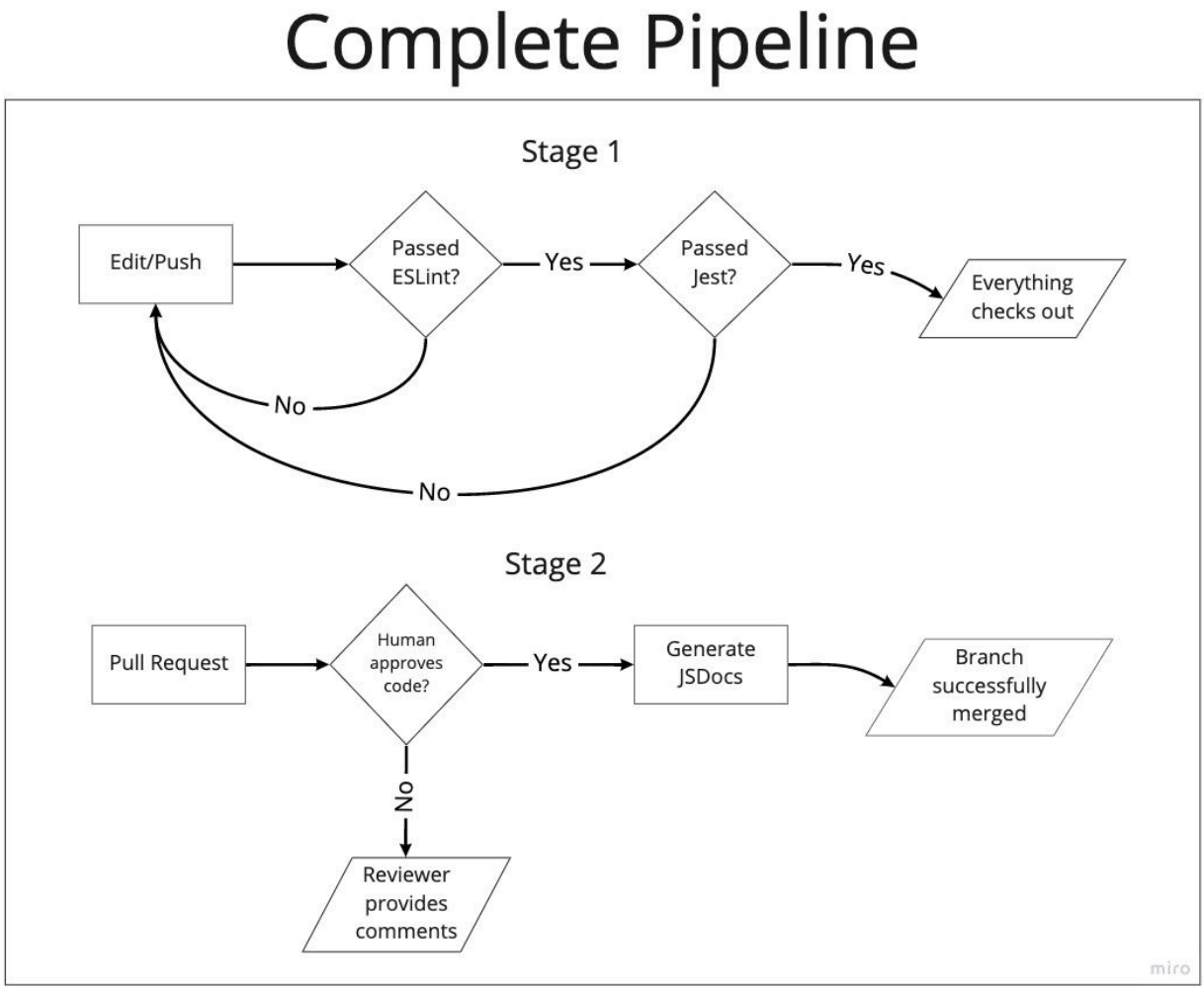

Figure 1. Complete pipeline (our first sprint will cover ESLint and human code approval)