Developer Behaviors and Team Processes

Release 3 Evaluation, December 7, 2018

|  |
| --- |
| **1. Testing with unit tests and other tools.**  Our process for unit testing is that we will be developing unit tests alongside our code as it is written. Any merge requests to develop for new features should include unit testing. Any time the develop branch is merged into the master branch, the merge must contain code that has been fully tested and analyzed unless approved by the Development Manager.  As of now, we have written a camera control API that allows control of DSLR and other compatible cameras as well as make filesystem changes to save and manipulate the photos taken. We have unit tests that ensure any code that does not directly make a gphoto2 call runs correctly. We have also implemented  **Documentation:** |
| **2. Always have a working build of the product with tests running.**  Our working build of the product will always available on the Spreetail GitLab repository on the “Master” branch. We have a “Develop” branch to test any new updates and changes to the code before pushing and merging it to the “Master” branch. Changes are merged by the Development Manager after he has reviewed, tested, and approved it. We create branches off of “Develop” and name it as “feature/name-here” when we create new features or have bug fixes. Those branches are also tested, reviewed, and approved by the Development Manager before merging it to the “Develop” branch. We will be using the GitLab CI as well as docker to build our code. Docker is what spreetail uses and has requested that we use to build our project components.  As of now, we are working on the CI process in GitLab. We are running into issues where docker is able to build our project on our machines, but not in the GitLab CI. Clayton has knowledge with docker and using CI, and he will be helping us through the debugging process. When we get the builds up and running in the GitLab CI, we can run unit tests alongside the build to ensure that everything is building and running correctly.  **Documentation:**    Here is an example of our branches that we have on GitLab and the naming conventions.      Here is an example of the built in CI in gitlab. Notice that all of the builds are failing as we continue development alongside the debugging process. |
| **3. Run static analysis tools to remove errors and warnings.**  We are developing the web server and camera control code in python. For python, we are using Pycharm, an IDE offered by Jetbrains. Pycharm has a built in analysis tool that finds warnings, errors, spacing, and other issues in the code. Below is an example of the analysis tool at work.  Front end development is being done in javascript and html using an open source editor called Atom. Atom has plugins that can easily be installed to provide code analysis. The library that we are using is called linter-eslint by AtomLinter. Below is an example of ESLint at work.  **Documentation:**  Pycharm**:**    The tool is giving an error for non-existent modules or modules that aren’t yet installed. It also grays out modules that are imported but not yet used. It also has its own spacing rules, and it is expecting at least 2 empty lines before the “def run(port=5000): line.  Atom with ESLint package:  This tool gives us warnings for components that aren’t be used at the moment. It will also display errors and warnings on the bottom left-hand side, if any. |
| **4. Maintain a story map using the Senior Design story format.**  We use storiesonboard to maintain an up-to-date story map for this project. Below is a screenshot of our next semester's story map.  **Documentation:** |
| **5. Maintain up to date definition of done and backlogs for each milestones and sprint.**  For the definition of done, we are using Zenhub as well as the github repository wiki. On Zenhub, we add a “Definition of Done” section to every story that we create. This allows us to look back on the story after development to make sure that the story is actually done. We also use the github wiki to keep track of the project level definition of done. This is mainly to keep in mind the end goal and the actions that the user should be able to perform.  **Documentation:** Zenhub: <https://app.zenhub.com/workspaces/spreetail-5bbceb414b5806bc2bec3047/boards?repos=145894767>  Github wiki: <https://github.com/cseseniordesign/spreetail/wiki> |
| **6. Estimate stories and assign points in the context of design.**  We have sprint planning every 2 weeks at the beginning of each sprint. Story time estimates are assigned by the group as a whole during this planning. Each member writes down how long they think the story would take, and then the group decides on an estimate for that story. We also implement the T-shirt method for the entire backlog, so that we can have an idea of the size of stories in the backlog without having to go into more detail. The t-shirt sizing is a label that is assigned to each story.  **Documentation:**  Zenhub: <https://app.zenhub.com/workspaces/spreetail-5bbceb414b5806bc2bec3047/boards?repos=145894767> |
| **7. Release milestones to the sponsor.**  We just finished release 3. This was the first release that we demonstrated a prototype that included code for the project, and that was a big step for us. We also received some very helpful advice from our sponsor’s guest at the presentation. He was very nice about letting us know where we can improve and how the presentation looked from his point of view. An overall idea was that we need to keep in mind that our presentations should include what our plan was for the milestone before going into what we were able to complete. Some viewers of our presentation also might not have a lot of the background knowledge known by some of the other attendees. Another piece of advice is that we should focus on the definition of done for our milestones and stories. This will allow for easier understanding of project progress.  **Documentation:**  *Include links and/or screenshots here.* |
| **8. Have a security plan for the highest security risk areas of project.**  The area of our project with the highest security risk is access to Spreetail’s GitLab. Their GitLab contains all of the code for their company, and if anyone were to get access to our computers or login info then they have the potential to access all of Spreetail’s source code. Usernames and passwords for the GitLab should never be shared, and when done working in GitLab, the developers should log out of their account and logout of their computer for extra security.  **Documentation:** |
| **9. Measure performance and use telemetry to inform decisions.**  We have determined the requirement to downsize images before they are submitted over request. The time it took to send a full-sized 6000x4000 image taken by the camera was too long for a user to wait for. For front-end preview experience, we currently downsized it by 8, resulting in 750x500, which takes less than a second to transfer compared to the previous twenty seconds.  **Documentation:**  *Include links and/or screenshots here.* |
| **10. Maintain cadence across iterations.**  We hold retrospectives and sprint planning at the end of each sprint so that we can keep track of what is working well and what we should work on. We also have daily standups at team time to make sure that everyone has something to do and progress is being made. We also keep track of all stories assigned to the current sprint, so that developers can easily move from one story to the next on completion.  **Documentation:**  Zenhub: <https://app.zenhub.com/workspaces/spreetail-5bbceb414b5806bc2bec3047/boards?repos=145894767> |