CSCI 3308 Homework #2

Aparajithan Venkateswaran

Part 1

1. Command to be used: `git checkout`

```
$ git checkout my_program.py
```

2. Command to be used: `git reset`

```
$ git reset HEAD~
```

3. Reasons to use feature branches:
    a. Master branch is live production code. If we commit directly to this, we can potentially cause a bug or break the system. Using a feature branch allows us to work seamlessly without worrying about breaking the system in the process because, there will be a code review process by the team and multiple tests before the branch is merged into master
    b. It also allows developers to take on a task and work on it until it's finished. If a feature X will not be ready for the next release, then the team does not have to wait until feature X is ready. They can release the next version and postpone the delivery date for feature X. Branches allow for continuous development
4. A pull request is essentially a merge request. You request the changes you made in your branch (or fork) to be merged into another branch. And your changes are reviewed by your team (or the code owners). Then ask you to make necessary changes before merging it. Git push is specifically for a single branch where the changes you make in branch X gets pushed to the remote's branch X. Also, there is no code review process.
   They are similar in that the commits you make eventually get added to the remote repository. They are both mechanisms for "uploading" your commits to the remote.
   Git pull "pulls" commits from the remote to your local repository. A pull request "requests" your commits to be added to the remote repository (or a different branch).

Part 2

1. Challenges in Agile:
    - Agile requires team members to be highly devoted to work intensely on short sprints. This can be tiring and exhaustive for the developers
    - No one knows how the final product will look like because Agile is an iterative process. So, the team needs to spend a lot of time meeting with customers to understand their needs during each cycle. And if the customers are not clear with what they want, the team will be taken off track.
    - In some deliverables, it is very hard to estimate the amount of effort required during the beginning of the sprint. Only when you start working on it, you will start to realize the difficulty/easiness of the feature.
2. Some stories might be large deliverables. They will have smaller specifics or features that the team may overlook when looking at the big picture. And only when they start delving into the sprint, they will realize that this deliverable will take longer than initially planned. This will affect the current sprint and have a cascading effect on other dependent features and sprints.

3. The people disagreeing should discuss and see where the problem lies. A good way for that would be for the person who gave the higher point estimate to break it down into 3-4 simpler tasks and find the hole. If disagreement still exists, the scrum master should probably average the points after discarding the outliers.

Part 3

1. Medium.com
   Good design features:
   a. Narrow layout of articles makes it easier for users to read the articles without having to move their eyes across the entire screen and losing track of the next line
   b. The large font size and amount of whitespaces (spaces, newlines, section separation) also adds on to the improving the readability of the text

   Scope for improvement:

   a. On some publications, there is a fixed navbar and footer with a large height. This obstructs the text leaving a small area where the actual article is. This means the user can only read so much at a time.
   b. There should be a fixed position "Jump to Comments" button along the margin. This way, users can easily go comment about their opinions without having to scroll all the way down to comments section.

   Accessibility: I think this website is not always accessible to people with disabilities. People with low vision benefit from the large font size and whitespaces that improve readability. A closer inspection at the source code reveals that the text has the correct tags for headings, links, text, images, etc. However, the links and images do not have an alternative text explaining where the link goes to, or the content of the image. This can be confusing and frustrating to blind people who use readers to read websites.

Part 4

1. Reasons for code review:
   i. Common coding style – Indentation preferences, variable naming conventions, etc. needs to be consistent across the code base for improving readability. This also helps a new developer get familiar with the code easily.
   ii. Finding bugs, glitches, etc. with the multiple tests.
   iii. Finding a better/efficient implementation of a procedure by an experienced developer who reviews the code

   Code reviews will keep the overall code base consistent in terms of style, conventions and practices. It also keeps the production code relatively bug-free and efficient. It will also ensure that the code base is not constantly updated by an intern or an inexperienced developer, because any change to production code needs to be approved by superiors and peers.

2. I would like to receive comments about whether my coding style matches the code base. And improvements to the changes I made, whether they are good or bad, necessary or not, is there a more efficient way to accomplish the same thing. I would also like for potential bugs to be pointed

out. I would like the reviewers to conduct the testing that they usually use for the production code base to be conducted on my changes. This means that they are actually serious about integrating my code into production and makes me feel good about the changes I proposed. In the real world, reviewers would probably only do tests for the modules I changed, and not all of the tests. This is more efficient and saves time for both parties.

If any of the tests fail, or the reviewers point out a mistake in my code, I would hear their opinion and if I agree with them, I will make the changes they request. If I have a different opinion, I would voice it out and make sure that we come to a mutual conclusion after both parties have heard both sides of the argument.

3. I will look for changes in important functions and review them closely because they are important to the product and any small mistake will result in a cascading effect on the product as a whole. I will also look for differences in coding style – tabs vs. spaces; variable naming convention; etc. If they added a new function or procedure, I will inspect it closely to make sure that it is bug free and run multiple tests with edge cases to filter out the bugs.

   Ways my attitude can affect team dynamics:

   i. If I am harsh and ask them to make changes without providing an explanation, it might demotivate them or create tension between the two parties. This becomes worse if I don't hear their opinions.
   ii. If I politely explain why I request the changes and provide constructive criticism, it will motivate them to work better and create mutual respect between both parties. I should also listen to and respect their opinions. This is important and necessary in teams.