Module 10
Agile Teams

CS 169A: Software Engineering

1 Overview

Becoming a proficient software engineer is itself a continuous, lifelong feat. However, what sometimes is more challenging is learning to work with other people. Whether its a group project, a sports team, or a student organization, chances are, you've already run into the nuances that come with working with people!

Module 10 is completely dedicate to the processes and workflows that make the human component of the Agile workflow successful. Specifically, we will look at processes and tools for debugging, checking in code, and deploying to production. Just remember that the core to great team dynamics is ensuring *everyone feels psychologically safe* to foster an environment where anyone can speak their mind without fear of failure or humiliation.

2 Version Control

From prior classes, hopefully you are familiar with version control as a tool for taking snapshots of your work and backing them up. *Branches* are a feature that allow you to simultaneously work collaboratively on the same codebase with multiple engineers! Answer the questions below about how to resolve different scenarios in Git version control:

2.1

Lay out the steps and commands that you would use to resolve a merge conflict in Git.

2.2

What is the difference between Git and Github?

2.3

What is the difference between a fork, a branch, and a clone of a Git repository?

2.4

What is the difference between `rebase` and `merge` in Git?

2.5

True or false: If you attempt `git push` and it fails with a message such as `Non-fast-forward (error): failed to push some refs`, this means some file contains a merge conflict between your repo's version and the origin repo's version.

3 Five R's of Bug Fixes

Debugging is an inevitable part of software development, but luckily, there's a team workflow in place for resolving them when they come up. The 5 R's are:

- Reporting a bug
- Reproducing the problem, or else Reclassifying it as "not a bug" or "won't be fixed"
- Creating a Regression test that demonstrates the bug
- Repairing the bug
- Releasing the repaired code

3.1

Assume you a part of a software engineering team building a ride-sharing application. Determine and justify if and why the following backlog items should be classified as bugs:

- A beta tester of a limited edition of the app reports a button that is not redirecting properly when clicked. It has been decided that the limited edition version of the app will not be deployed.
- A user with an iPhone 4s and an outdated version of iOS reports that certain UI elements are not rendering properly.
- A duplicate of a previous issue that has been fixed in the upcoming to be released version of the app.
- A user does not like that the app doesn't include information about the driver and requests it to be displayed.
- When a user requests a ride from an airport, the designated driver receives an incorrect pick up location.

3.2

Why do you think “bug fix” stories are worth zero points in Tracker even though they follow the same lifecycle as regular user stories?

3.3

True or false: a bug that is triggered by interacting with another service (for example, authentication via Twitter) cannot be captured in a regression test because the necessary conditions would require us to control Twitter’s behavior.

3.4

True or false: a bug in which the browser renders the wrong content or layout due to JavaScript problems might be reproducible manually by a human being, but it cannot be captured in an automated regression test.