Assignment T5: Second Iteration

Published | Edit | ⋮

Implement the **final** version of your service. Document your service's API in your README (or include a clearly marked link in the README to where your API documentation resides).

Develop a "client" for your service. Your client could be an app, with a UI, or another service, with its own API. Your client should be demoable, but please do not spend a lot of time making a fancy UI! This is a software engineering class, not a user interfaces class. Explain in your README how to build and run your client (ideally building the client would be included in CI - see below - but that may not be practical). It's ok for the client to run on "localhost". Ideally your client code would be in the same repository as your service (**Google stores billions of lines of code in a single repository** (https://cacm.acm.org/magazines/2016/7/204032-why-google-stores-billions-of-lines-of-code-in-a-single-repository/fulltext)), but it's ok to put the client code in a separate repository as long as the README for your main repository provides a link to this second repository and you have added the teaching staff as collaborators to the new repository.

Add continuous integration (CI) to your main github repository, so that all your build, analysis and testing tools are automatically run for every commit. Integrate CI with your repository, e.g., using **github actions** (https://github.com/marketplace?category=continuous-integration&type=actions) or **github apps** (https://github.com/marketplace?category=continuous-integration&type=apps). Include the CI reports in your repository and explain where to find them in your README (or include a clearly marked link in the README to where your CI reports reside). If the CI reports include any or all of the other reports mentioned below, you do not need to also include them separately, but please explain what is and is not included with the CI reports in your README.

Implement service tests that exercise your service's final API. Your tests should run automatically during CI (**this explains how for Postman** (https://learning.postman.com/docs/integrations/ci-integrations/)). Include your API test reports in your repository.

Implement unit tests for all the major "units" in your code and group your unit tests into test classes. You should use a mocking tool, where applicable, and a coverage tool together with your unit testing tool. Try to achieve at least 85% branch coverage. Your unit testing should run automatically during CI (**this explains how for your favorite language and Travis CI** (https://docs.travis-ci.com/user/languages)). Include your unit testing and coverage reports in your repository.

Also run a style checker and static analysis bug finder, e.g., **SonarCloud** (https://sonarcloud.io/documentation), during CI. Include their reports in your repository.

Make sure all your implementation, test and client code, and any configuration files, scripts, etc. has been committed to the main branch of your github repository. (If your client code is in a separate

repository, then it should be in the main branch.)

If any third-party code is included in your codebase, also document in your README exactly which code this is, where it resides in your codebase, and where you got it from (e.g., download url).

Submission instructions: One member of your team should submit a url pointing to your main codebase repository. Make sure your IA mentor and the instructor have collaborator access! This should normally be the same repository from your revised proposal, but we ask you to submit again for this assignment to make it easier to use Coursework's "SpeedGrader" tool.

Points 50

Submitting a website url

Due	For	Available from	Until
Dec 11, 2021	Everyone	Nov 14, 2021 at 12:01am	Dec 17, 2021 at 11:59pm

T5 Second Iteration

You've already rated students with this rubric. Any major changes could affect their assessment results.

Criteria	Ratings	Pts
Ignore imported code for all concerns. There do not need to be any test cases, comments, style compliance, etc. for any code documented in the README as third-party code.	This area will be used by the assessor to leave comments related to this criterion.	0 pts
Client Did they develop a client that reasonably exercises their API, is the client code in the repo, and is it documented in the README? They were not required to run testing or checking tools on the client.	This area will be used by the assessor to leave comments related to this criterion.	15 pts
Continuous Integration Are they using CI on github commit to the main branch, to automatically build their service and run all its unit tests and other checking tools (coverage, bug finder, style checker)? Can you find CI reports in their repo? (They do not have to use CI for system tests or for their client.)	This area will be used by the assessor to leave comments related to this criterion.	15 pts
Coverage Did they use a coverage checking tool together with unit testing their service? Can you find coverage reports in their repo? (This might be included in CI reports.) They did not need to integrate coverage with system testing and they were not required to test their client.	This area will be used by the assessor to leave comments related to this criterion.	10 pts
Bug Finder Did they use a bug finder tool on their service? Make sure its really looking for bugs, not just style checking. Can you find its reports in their repo? (This might be included in CI reports.) They did not need to apply to their client.	This area will be used by the assessor to leave comments related to this criterion.	10 pts
Partial credit Need to determine how to handle partial credit for all relevant items.	This area will be used by the assessor to leave comments related to this criterion.	0 pts
		Total Points: 50