# Assignment T3: First Iteration

Published | Assign To | Edit | ⋮

This is a team assignment. See submission instructions at the end.

Implement a **demoable** version of your server and make sure all the implementation code has been committed to the main branch of your github repository (its ok to have some other code in branches but it will not be considered part of your submission). This version of your server does not need to include all the functionality you proposed, but some API entry points must be operational in order to demo. In your github repository, **tag** (https://stackoverflow.com/questions/18216991/create-a-tag-in-a-github-repository) the set of file revisions considered part of the the first iteration.

Implement and run unit tests for all the major "units" (methods, functions, procedures, subroutines, etc.) in your code. You should use a mocking framework, where applicable. Unit tests should be grouped so related code (e.g., classes, modules, files, etc.) share setup, teardown and mocks when appropriate. Your entire collection of unit tests should run "push-button" via some test runner appropriate for your language and platform. Try to fix most of the bugs found by the tests.

Implement and run system tests that exercise your server's initial API, e.g., using some API testing tool. Your tests should exercise the multiple clients and persistent data aspects of your server. Try to fix most of the bugs found by the tests.

Make sure all your test code and scripts have been committed to the main branch of your repository (its ok to have some other test code in branches but it will not be considered part of your submission).

**Important Note: Your tests do not need to achieve high coverage, yet, that will be part of the second iteration. You will also use a static analysis bug finder and continuous integration during the second iteration, as well as add integration testing and a sample client. You can do any or all of these during the first iteration if you want, but it is not required until the second iteration.**

All commit messages, except for the initial repo-setup commit(s), should say something meaningful about what was added/removed/changed.

All non-trivial code, including test cases, should be self-documenting with mnemonic identifiers and header comments and, where appropriate, in-line comments. (Trivial means things like getters, setters, small helper functions, and other self-evident code. There is no point in commenting **int addone (x) { return x+1 }**.)

Your entire codebase, including test cases and even trivial code, should be compliant with a style checker appropriate for your language/platform. Include some reports from your style checker in your repository. If it does not generate files, use redirection of output to a file, screenshots, or some other means to produce files.

Include in your repository any configuration files needed to build, run and test your server.

Write a README.md that describes in simple terms how to build, run and test your server.

The README.md should also describe all the operational entry points to your server or link to some external documentation, e.g., **Swagger** (https://swagger.io/) . As noted above, this might be a subset of your proposed API. This is your "API documentation". If certain entry points must be called in certain orders or never called in certain orders, make sure to say so.

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

**Submission instructions:** One member of your team should submit a url pointing to your codebase repository. We encourage you to make your repository public, but if you prefer to keep it private then make sure the instructor and your Mentor 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 |
| --- | --- | --- | --- |
| Oct 24, 2022 | Everyone | Sep 5, 2022 at 12am | Nov 1, 2022 at 11:59pm |

**First Iteration Rubric** 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. (0 points)<br>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 |
| Documented API (0 to 10 points)<br>Does the team repo's README document their API or link to external documentation of the API? Is it really a code-calls-code-over-network API, not a user-facing CLI or other kind of UI? Does it cover the inputs, outputs including status codes and any exceptions, and ordering (if relevant) of all implemented entry points? | This area will be used by the assessor to leave comments related to this criterion. | 10 pts |
| System Tests Corresponding to API (0 to 15 points)<br>Is there evidence in the team's codebase, an external tool like postman, and/or the team's demo that the team conducted code-calls-code-over-network system-level tests exercising every API entry point? For each given API entry point, were there tests forcing every possible status code (if applicable)? Did the set of system-level tests exercise the multiple clients and persistent data aspects of the service? | This area will be used by the assessor to leave comments related to this criterion. | 15 pts |
| Unit Tests (0 to 10 points)<br>Are there unit tests for most major methods that directly call the unit (not over the network)? Are the unit tests organized to all run automatically from a test runner? Do the unit tests or test classes include setup/teardown code (where applicable)? Do they use mocks (where applicable)? | This area will be used by the assessor to leave comments related to this criterion. | 10 pts |
| Commit Messages (0 to 3 points)<br>Check a few of the commit messages, do they say something meaningful about what changed? | This area will be used by the assessor to leave comments related to this criterion. | 3 pts |
| Self-Documenting Code (0 to 3 points)<br>Check a few of the longer functions or methods, do they have reasonable comments and meaningful function names and parameter names? | This area will be used by the assessor to leave comments related to this criterion. | 3 pts |
| Style Compliant (0 to 3 points)<br>Can you find style checker reports in their repository and does it look like most of the problems (if any) were fixed? | This area will be used by the assessor to leave comments related to this criterion. | 3 pts |
| Build, run, test instructions (0 to 3 points)<br>Does the README include instructions explaining how to build, run and test their service and does the repository contain all configuration files needed for build, run and test? | This area will be used by the assessor to leave comments related to this criterion. | 3 pts |
| Tagged Revisions (0 to 3 points)<br>Are the file revisions that contributed to the "first iteration" all tagged as such? | This area will be used by the assessor to leave comments related to this criterion. | 3 pts |
| | | Total Points: 50 |