Course project: Implementation and Documentation

Overview

Make progress on the implementation of your project and solidify the documentation for end-users and developers.

Set up

Work with your project group to (1) make progress towards the final release, (2) provide a comprehensive user documentation, and (3) provide a comprehensive developer documentation.

Instructions

1. Make progress in your public GitHub repository

* Each group member must contribute to the code base.
* Each group member must demonstrate proper use of version control and CI.
* Each contribution (commit/pull request) must be tested, commented, and code reviewed.

Review and follow the [Git conflict resolution tutorial](https://rawgit.com/mernst/git-conflict-tutorial/master/git-conflict-resolution.html). When merge conflicts come up later in the quarter, you will be happy that you did.

2. Write user documentation

Your public repository must contain a complete user manual. Anyone looking at your repository should be able to easily find the user manual. **The user manual is focused solely on people who want to use your project**.

The user manual should describe the functionality of your project as you expect it to be at the end of the quarter. For this assignment, **indicate missing functionality as work in progress**.

The user documentation should include at least the following information:

* **A high-level description.** What does the system do and why would a user want to use it.
* **How to install the software.** If your system has prerequisites (e.g., tools, libraries, emulators, third-party applications, etc.), your instructions should list all of them and indicate how to install and configure them. Make sure to indicate what specific version requirements these prerequisites must satisfy. If running the system requires the installation of, e.g., a virtual machine, a database, or an emulator, make sure to provide clear step-by-step instructions.
* **How to run the software.** How to start up the system?
* **How to use the software.** You can assume that your user is familiar with your particular platform (e.g., use of a Web browser, desktop applications, or mobile applications). For missing functionality, your documentation should simply indicate that this functionality is work in progress.
* **How to report a bug.** This should include not just the mechanics (a pointer to your issue tracker), but also what information is needed. You can set up a bug-report template in your issue tracker, or you can reference a resource about how to write a good bug report. Here is an [example for bug reporting guidelines](https://developer.mozilla.org/en-US/docs/Mozilla/QA/Bug_writing_guidelines).
* **Known bugs.** Known bugs or limitations should be documented in the bug tracker. A user testing the implemented use case(s) should not encounter trivial bugs (e.g., NPEs) or a large number of bugs that are unlisted in your bug tracker.

3. Developer documentation

Your public repository must contain developer guidelines. Anyone looking at your repository should be able to easily find these guidelines. **The developer guidelines are focused solely on people who want to contribute to your project**.

The developer documentation should include at least the following information:

* **How to obtain the source code.** If your system uses multiple repositories or submodules, provide clear instructions for how to obtain all relevant sources.
* **The layout of your directory structure.** What do the various directories (folders) contain, and where to find source files, tests, documentation, data files, etc.
* **How to build the software.** Provide clear instructions for how to use your project’s build system to build all system components.
* **How to test the software.** Provide clear instructions for how to run the system’s test cases. In some cases, the instructions may need to include information such as how to access data sources or how to interact with external systems. You may reference the user documentation (e.g., prerequisites) to avoid duplication.
* **How to add new tests.** Are there any naming conventions/patterns to follow when naming test files? Is there a particular test harness to use?
* **How to build a release of the software.** Describe any tasks that are not automated. For example, should a developer update a version number (in code and documentation) prior to invoking the build system? Are there any sanity checks a developer should perform after building a release?

Assignment Submission:

Email link to your team GitHub Project Repository to our course TA.

The subject of the email should be: [Your StudentID, Assignment Name]

Clarifications

*What if info needed in my user or developer documentation already exists in my living document?*

The documentation in your repository should be largely self-contained, and anyone looking for documentation should find it, or a link to it, in the repository.

You may link to sections of your living document from your user/developer documentation, or you may move these sections to the repository and put a link to it in your living document.

Do not maintain two parallel versions of any documentation.