Test Plan

Crowdsourcing Code Comments with ChatGPT (CrowdDoc)
CSE 4101 Senior Project - Fall 2026
Ahmed Alzahrani and Mohammed Zakry
Faculty Advisor / Client: Professor Marius Silaghi

1. Purpose

This test plan explains how we will check that CrowdDoc follows the Requirement Document. We include normal cases and unusual cases because the goal is to find bugs, not only to show the easy path.

The website label Reviewer means the same second AI step called Checker / Reviewer in this plan.

2. Test Environment

- A normal laptop and modern web browser.
- CrowdDoc running with a test SQLite database.
- The saved Linux v6.12 snapshot with the three current C files.
- An OpenAI API key only when testing live Writer and Checker / Reviewer calls.

3. Test Cases

TC-01 - FR-1 to FR-4: Normal: browse the saved folders, open each supported file, and open several selected functions. Unusual: try a missing path, '..', a backslash path, and a function not in the manifest. Expected: valid items open normally; unsafe or unknown paths do not escape the source tree or connect comments to the wrong function.

TC-02 - FR-5 and FR-7: Normal: run the Writer on a known function with a live API key. Unusual: remove the key or force an empty/bad model response. Expected: a live Writer result is saved when the call works; otherwise a clear error or clearly labeled stand-in is shown.

TC-03 - FR-6 and FR-7: Normal: save a Writer result, then run the Checker / Reviewer using the same function and saved Writer explanation. Unusual: give the Writer an obvious wrong claim or force the Checker / Reviewer call to fail. Expected: the Checker / Reviewer is linked to the exact Writer it checked; if the Checker / Reviewer fails, the saved Writer result remains.

TC-04 - FR-8 and FR-9: Normal: submit a human correction or addition with purpose, inputs, outputs, and side effects. Unusual: try an empty field, more than 4000 characters, wrong mode, wrong parent, or a parent from another function. Expected: only valid human explanations are saved and they stay attached to the correct function.

TC-05 - FR-10 to FR-12: Normal: vote for a valid explanation, refresh the page, and restart using the same database. Unusual: try an invalid function/explanation pair and repeated votes. Expected: valid data is saved and reloaded; wrong relationships are rejected; repeated votes are documented as a current limitation because sign-in/user identity is not part of Milestone 1.

TC-06 - FR-13 and SI-5: Normal: open the Elixir/Bootlin link for a supported Linux v6.12 function. Unusual: make the external source site unavailable. Expected: the link points to the matching v6.12 source and CrowdDoc's local page still works if the outside site is down.

TC-07 - IR-1 to IR-7: Normal: use the home page, folder pages, breadcrumbs, function page, human form, and vote controls. Unusual: use long text and a narrow browser window. Expected: the main page stays readable and Writer, Checker / Reviewer, stand-in, and human labels stay clear.

TC-08 - PR-1 and PR-2: Normal: measure normal folder/function page load time and one live Writer and Checker / Reviewer run. Unusual: repeat with a slower connection or larger supported file. Expected: record the real times and do not claim a speed target passed unless it was actually measured.

TC-09 - PR-3: Normal: have a human reviewer compare Writer-only and Writer-plus-Checker / Reviewer output with the source. Unusual: include a case where Writer is already correct and a case where the Checker / Reviewer gives a false warning. Expected: record false claims, missing details, Checker / Reviewer catches, false alarms, and new mistakes.

TC-10 - PR-4 to PR-6: Normal: add comments and votes, inspect the saved source files, and inspect browser-visible content. Unusual: force an AI error and inspect the shown error message. Expected: the source stays unchanged and no API key or private server value appears in the browser or error text.

4. Test Evidence

- Keep screenshots for important interface tests.
- Keep terminal output for automated tests, lint, and build checks.
- Restart the app when checking persistence.
- Use a simple reviewer sheet for AI accuracy testing.

5. Pass / Fail Rules

- A test passes only when the expected result is actually observed.
- Changing the saved C source because of comments or votes is a failure.
- A comment attached to the wrong function or version is a failure.
- Showing a stand-in as live AI is a failure.
- Exposing the OpenAI API key is a serious failure.
- AI agreement by itself does not prove correctness.