Test Plan

FileSender

SURF
Product Manager: William van Santen
Mentor: Rogier Spoor

Windesheim University of Applied Sciences
Supervising Teacher: Rob Kaesehagen
Student: Aaron Jonk (s1170298)

Date: 11/02/2025

Version: 0.2

Version management

Version	Date	What
0.1	07/02/2025	Start development of this file
0.2	11/02/2025	Write introduction, unit testing, end-to-end testing, and code coverage.

Distribution

Version	Date	Recipient
0.2	18/12/2025	Publish on Codeberg

Contents

1 Introduction and Context

For the development of FileSender version 4, this project will focus on creating a robust file-sharing application with enhanced testing methodologies to ensure its stability, security, and functionality. This document outlines the various testing methods that will be applied during the development process, explaining how each will be implemented and integrated into the development pipeline. Key considerations include available development time, the importance of specific functionalities, and the overall security of the system.

As part of the chosen testing approach, specific test cases have been defined for user testing. Each test case includes detailed instructions on how the test should be conducted and records the findings from these tests. At the conclusion of the testing phase, the results will be compiled into a comprehensive overview, highlighting critical bugs, impactful improvements, and suggestions for optimization. This analysis will serve as the foundation for determining which improvements can be addressed in the short and long term, allowing developers to prioritize fixes based on user feedback and system needs.

2 Unit Testing

The first category of tests outlined in this document is unit testing. Fortunately, Go's standard library includes built-in support for these tests, making it easy to implement them. Unit tests are designed to specifically test individual functions and validate their returned values. By running these tests, we can detect errors in the code and ensure that the output aligns with expected responses. Since these tests must remain up to date, a requirement will be added to the "definition of done" to ensure that unit tests are created for any newly implemented features.

Additionally, these unit tests can be automatically executed within the CI/CD pipeline. If any test fails, it will halt the pull request, ensuring that only properly tested and functional code is merged into the repository.

3 End to End Testing

After evaluating various frameworks for end-to-end testing, Playwright emerged as the best option. With Playwright, we can test the application across multiple browsers and environments, ensuring comprehensive coverage. One key requirement is that the application must function properly even when JavaScript is disabled, and Playwright allows us to simulate this scenario by turning off JavaScript during testing.

By running tests across different browsers, configurations, and environments, we can ensure the application is accessible and reliable for a diverse range of users. These end-to-end tests will also be integrated into the CI/CD pipeline, and new tests will be included as part of the "definition of done" whenever new features are developed. The pipeline will automatically run these tests as new features are added.

3.1 Code Coverage Testing

As an extension of the end-to-end tests, we will utilize Go's built-in code coverage functionality. The application can be started with a coverage flag, which tracks how much of the application's code is being executed during testing. When the application completes its run, a detailed code coverage report is generated. This report will be assessed within the CI/CD pipeline to ensure adequate test coverage. Our initial goal will be to achieve at least 80% code coverage, providing a solid baseline for effective testing.

4 User Tests

The final testing phase will involve user testing. At the end of the development process for the new FileSender, we will invite several users to follow detailed test scripts. These users will complete specific tasks without any prior insider knowledge, ensuring that the application is intuitive and user-friendly. Tasks will range from system administrators setting up the application to end users uploading files.

Test scripts and cases will be carefully crafted and executed methodically. By analyzing the results of these tests, we will identify areas that require improvement and give additional attention to any features that users find confusing or problematic. This feedback will be crucial for refining the application and enhancing the overall user experience.

4.1 Test Cases

TBD