GitHub Gist

Test Plan

Version 1.1

Table of contents

Revision History

Date	Version	Description	Author
2023.10.18	1.0	First draft of the Test Plan	Regina Chepkunova
2023.10.22	1.1	Second version of the Test Plan	Regina Chepkunova

Objective

The objective of this test plan is to ensure the quality and reliability of the GitHub Gist feature (specifically: getting and creating gists), both in terms of API and web interface functionality.

Scope

The scope of this testing will cover the GitHub Gist feature, focusing on creating and getting Gists.

Test Levels

This test plan will include both manual and automated testing. Manual testing will help validate user experience, while automation will ensure coverage of critical functionality.

Test Types

The following test types will be performed:

- Functional Testing
- Integration Testing
- API Testing
- UI Testing
- Regression Testing

Test resources

Environment

The testing environment will be set up using Docker containers to ensure consistency.

Testing Tools

Automated testing will be implemented using the following tools and frameworks:

- Python with Playwright for UI testing
- Python with Pytest for API testing
- Docker
- Allure + pytest html for reporting
- GitHub Actions for CI/CD integration.

Tests will be conducted on various browsers (Chrome, Firefox) for UI testing.
API tests will be performed on different endpoints and with various inputs.

Test Data

Test data will include a range of scenarios, including valid and invalid inputs, edge cases, and boundary values.

Test documentation

Test Scenarios

Creating Gists (UI and/or API)

1. Positive Scenario - UI: Verify that a new Gist can be created via the web interface with valid data.
   - Input: Valid Gist content, description, and filename.
   - Expected Outcome: The Gist is successfully created, and the user is redirected to the Gist's page.

2. Negative Scenario - UI: Ensure that Gist creation fails with missing or invalid data.
   - Input: Incomplete or incorrect Gist data.
   - Expected Outcome: The system displays an error message and prevents Gist creation.

3. Positive Scenario - API: Confirm that a new Gist can be created via the API with valid data.
   - Input: Valid Gist content, description, and filename through API request.
   - Expected Outcome: The API response indicates a successful Gist creation.

4. Negative Scenario - API: Validate that Gist creation via the API fails with missing or invalid data.
   - Input: Incomplete or incorrect Gist data in the API request.
   - Expected Outcome: The API returns an error response, and the Gist is not created.

Getting Gists (UI and API)

5. Positive Scenario - UI: Verify that a user can access an existing Gist via the web interface.
   - Input: An existing Gist URL or search query.
   - Expected Outcome: The Gist is displayed, and its content is accessible.

6. Negative Scenario - UI: Ensure that Gist retrieval fails with incorrect or non-existent Gist information.
   - Input: An invalid Gist URL or search query.
   - Expected Outcome: The system provides an error message or an empty result if the Gist does not exist.

7. Positive Scenario - API: Confirm that a Gist can be retrieved via the API with valid parameters.
   - Input: A valid Gist ID or other required parameters in the API request.
   - Expected Outcome: The API responds with the Gist's details and content.

8. Negative Scenario - API: Validate that Gist retrieval via the API fails with incorrect or non-existent parameters.
   - Input: Invalid Gist ID or incorrect parameters in the API request.
   - Expected Outcome: The API returns an error response or an empty result if the Gist does not exist.

Reporting

Test results will be reported using Allure + pytest-html, providing interactive and informative reports.
Test reports will be shared with the team and stakeholders for review.

Test Automation

Automated tests will cover critical paths and frequent user interactions.
UI tests will simulate user actions such as clicking, typing, and verifying page elements.
API tests will use a combination of positive and negative test cases to validate endpoints.

Test Maintenance

Test scripts and data will be maintained and updated as the application evolves.
Regression testing will be performed with each new release.

CI/CD

CI/CD is set up with GitHub Actions and located on gist_project/actions. After every push to master branch, api + ui tests run and user can see test report via Artifacts panel in every specific run. The test report is visible through .zip file that has to be downloaded to check the results out.

Risks and solutions

Risk: Limitation of working with Rest API: after certain amount of requesting some endpoints, GitHub blocks user with provided token for some period of time
Solution: Have additional users to switch between them in the tests

Timeline

The testing phase started 18.10.2023 and completed 22.10.2023.

Review and Feedback

Continuous feedback and review of the testing process will be sought from the development and quality assurance teams.