1. Purpose of the document

This document explains the various testing activities performed on Augur's new frontend and Getting Started Document.

2. Application Overview

Augur is a Flask web application, Python library, and REST server that presents metrics on open source software development project health and sustainability. Also, Augur is the first CHAOSS Software Project to incorporate community reports that aggregate and synthesize atomic CHAOSS metrics into actionable information.

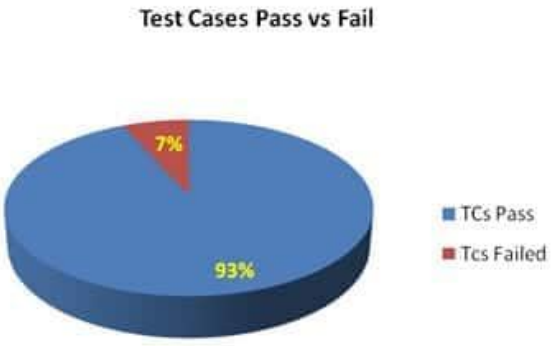
3. Testing Scope

We completed functional testing in a limited time frame to ensure that new features worked properly. Afterward, we would like to try more comprehensive testing methods and tools to optimize the quality of the code.

- In-Scope: Functional testing for the following modules is within the scope of testing.
    - New Page
    - Repos Grid Page
    - Collection Status Page
    - Getting Started Document
- Out of Scope: Unit and integration tests were not done for this application.
- Items were not tested:
    - Deploy the project to Jenkins for continuous integration and delivery.
    - Use SonarQube to check the code smells of the project.
    - Use defect reporting tools and tracking tools.

4. Metrics

Test cases planned	80
Test cases executed	75
TCs Pass	70
TCs Failed	5



5. Types of testing performed

- Manual Testing:
    - Check that Augur's new frontend displays new features correctly.
    - Use the browser's DevTools to check the error messages in the console.
    - Check that the results displayed on Augur's new frontend are consistent with the data from Augur's API.
    - Ensure that new features do not affect existing functions.
    - Check if it is displayed correctly on a variety of devices, including smartphones, tablets, and computers.
    - Check if it is displayed correctly on various browsers, such as Chrome, Firefox, Safari, Edge, etc.
    - Peer review to ensure that the development approach, including feature usage, code reuse, and coding style is consistent with the original project.
- Documentation Testing:
    - Understand all the contents of the original document.
    - Check the spelling and grammar to review any ambiguity or inconsistency between what functionality it performs and what it is supposed to do.
    - Follow the steps in the new document to actually install Augur once and check if it can be installed successfully.
    - Show the new document to people outside of our team members to ensure they can understand it.
    - Examples will be needed in case of any problem that occurs to the user, particularly novice users who may check the documentation for any confusion.

6. Test Environment & Tools

- Backend: http://ec2-3-138-116-248.us-east-2.compute.amazonaws.com:5000
- Frontend: http://ec2-3-138-116-248.us-east-2.compute.amazonaws.com:8000
- Currently serving: http://augur.chaoss.io/api/unstable
- Physical server: Linux Ubuntu 20.04
- Browsers: Chrome, Firefox, Safari, Edge

7. Lessons Learned

Issues faced	Solutions
Checking frontend displays was required to be executed manually each time.	Used Frontend Automation Testing Tools.
Checking API data was required to be executed manually each time.	Wrote a javascript function to automatically check the data.

Issues faced	Solutions
New features could easily affect existing functions.	Used version control tools such as GitHub, GitLab, and Bitbucket.
Testers may not have a variety of devices, such as tablets.	Used the device toolbar of Chrome's DevTool.
It was easy to make grammatical errors when writing updates to the getting started document.	Used automatic grammar checkers such as Grammarly.

8. Recommendations

Scrum is just a simple framework and the actual development of software still requires continuous integration, unit testing, refactoring, test-driven development, and design patterns. Also, using Mob programming and Kanban methods will make development more efficient.

9. Best Practices

- A repetitive task that is time-consuming to complete manually each time. This task is automated by adding a script and running it each time, thus saving time and resources.
- Critical scenarios are tested individually throughout the application, which is essential to prove that they work properly.

10. Exit Criteria

- All test cases should be executed.
- All critical, major, and moderate severity defects should be verified and closed.
- For any non-critical open defects, develop an action plan with an expected end date.

11. Conclusion

Since the exit criteria mentioned in Section 10 are met, the test team member recommends that the application "Go Live". Appropriate user acceptance testing should be performed prior to "Go Live".

12. Definitions, Acronyms, and Abbreviations

- About Augur: https://github.com/chaoss/augur
- About CHAOSS: https://chaoss.community/software
- Getting Started: https://oss-augur.readthedocs.io/en/main
- Our GitHub Repository: https://github.com/peteryu54089/augur_view