Quality Assurance Test Plan

Apache OpenMeetings

Trieu Huynh Ba Nguyen

Pouya Amiri

Arlis Puidet

Delia Fliscu

Eduard-Gabriel Ailincai

Table of Contents

Document Revisions

Date	Version	Description	Author
06.07.2023	1.0	The decision to choose Apache's Open Meetings as the subject for this Quality Assurance Test Plan was made. The document was created.	Eduard Ailincai, Delia Fliscu, Pouya Amiri, Trieu Huynh Ba Nguyen
11.07.2023	1.1	The outline of the document was created based on the document model provided. Each team member received parts to deliver.	Eduard Ailincai, Delia Fliscu, Pouya Amiri, Trieu Huynh Ba Nguyen, Arlis Puidet
21.07.2023	1.2	Schedule (3), Team (6), Environment (7) were added.	Trieu Huynh Ba Nguyen
25.07.2023	1.3	The document introduction (1) has been updated.	Eduard Ailincai
25.07.2023	1.4	Documentation and writing of Scope Testing (2) and Test Design (4)	Arlis Arto Puidet, Pouya Amiri
28.07.2023	1.5.1	Test Execution (5) was added	Delia Fliscu
30.07.2023	1.5.2	Test Report was created	Delia Fliscu, Eduard Ailincai

Document Approval

Signature (*Signed)*	Signature (*Signed)*
Date: 31.07.2023	Date: 31.07.2023
Name: Eduard-Gabriel Ailincai	Name: Pouya Amiri

1. Introduction

This Quality Assurance (QA) Test Plan's main goal is to create an organized approach to evaluate Apache OpenMeetings' functionality, performance, and security. The strategies, procedures, and timetables for assuring the quality and reliability of the OpenMeetings platform are explained in this comprehensive document. This software is used for user desktop sharing, online training, web conferencing, collaborative whiteboard sketching, and document editing.

1.1 Purpose

We seek to identify and address potential flaws early in the development lifecycle through standardized testing procedures.

1.2 Scope of the Document

The scope of the Quality Assurance Test Plan document is meant to outline the general testing strategy and methodology for the program. It defines the goals, techniques, materials, and timetables required to guarantee the accuracy and dependability of the software.

1.3 Overview

Introduction: The document begins with an introduction that describes the goal of the QA Test Plan and the importance of ensuring the quality of Apache OpenMeetings. It gives a quick overview of the software and emphasizes how important it is for promoting remote communication and online teamwork.

Functional Requirements: This includes testing Apache OpenMeetings' essential features to make sure they all operate as planned and conform to the specifications specified in the project documentation. *Non-Functional Requirements represent non-functional factors, including performance, scalability, security, usability, and compatibility, which will be evaluated.* These tests evaluate how well the system operates in various situations and environments.

Test Deliverables and Test Schedule: The document covers every QA deliverable that will be created throughout the testing process, including the test plan, test cases, test data, test reports, and other paperwork.

Test Design and Execution: This section of the paper describes the testing methodology, techniques, and approach that will be employed, as well as how test cases will be carried out, flaws noted, and outcomes reported.

Test Team: The document defines the members' tasks, roles, and areas of competence. It describes the coordination and cooperation needed to successfully complete the testing activities.

The test environment is described in this part, together with the necessary hardware, software, network setups, and other resources. It makes sure a reliable and accurate testing environment is set up.

2. Scope of Testing

2.1 Product Overview

Apache OpenMeetings is a free open-source web conferencing and collaboration application. It allows users to conduct online meetings, share screens, use an interactive whiteboard, record sessions, and engage with chat and polls. Apache OpenMeetings is useful for remote teams, virtual classrooms, and webinars.

2.2 Product Risks

Product risks with Apache OpenMeetings include security vulnerabilities, compatibility issues, performance issues, accessibility concerns, data loss or corruption, user experience challenges, integration difficulties, and scalability limitations. Proactive monitoring and mitigation strategies are essential to ensure the software's reliability and success.

2.3 Test Coverage

Test Type	**Applicable**	**Description**
Smoke testing	Yes	Ensure basic functionalities are working in each QA build.
Functional testing	Yes	Functions like audio, video, being able to make a meeting, and
Usability testing	Yes	Intuitiveness of UI/UX design
Security testing	No	Ensuring that connections are secure and that users don’t have access to things they aren’t authorized to

Performance testing	No	Handling of many users, data transfer limits, and the stability of the program
Accessibility testing	Yes	Ensure accessibility to people with disabilities
Localization testing	Yes	Ensuring that the application can be accessed with different languages
Authentication testing	Yes	User should be able to log in and out

2.4 Functional Requirements

Use Case No.	**Functional Requirement**	**Iteration**
1	Log in and log out	Iteration 1
2	Ability to make meetings	Iteration 1
3	Ability to join meetings	Iteration 1
4	Ability to use video while in a meeting	Iteration 1
5	Ability to use audio while in a meeting	Iteration 1
6	Ability to use screen sharing	Iteration 1
7	Ability to send and see sent messages	Iteration 1
8	Ability to record meetings	Iteration 1
9	Ability to share files	Iteration 1
10	Ability to modify user profile	Iteration 2
11	Testing of user roles and if they perform as expected	Iteration 2

12	Functional in at least 2 or more languages	Iteration 2
13	Testing of UI/UX, and if users can navigate the application	Iteration 2
14	Ability to schedule meetings	Iteration 2
15	Testing of notifications and reminders	Iteration 2
16	Testing to see if the program can be used and navigated through with only a keyboard	Iteration 2

2.5 Non-functional Requirements

No.	Non-Functional Requirement	Criticality
1	Performance: the application should support at least 10 users in a meeting	High
2	Reliability: maintain 99% uptime	High
3	Security: existence of a strong encryption	High
4	Compatibility: multi- and cross-platform compatibility	High
5	Usability: Intuitive user interface	High
6	Accessibility: Is in order with Web Content Accessibility Guidelines 2.0	High
7	Connection efficiency: optimization of the speed and usage of connections (bandwidth related)	Medium
8	Data backup and recovery: Efficient and reliable systems should be in place	Medium
9	Reporting and analytics: get metrics to improve the application and inform the user	Low
10	GDPR compliance: make sure that data collection and storage is compliant	Critical
11	Documentation: good documentation regulation to help keep track of changes	Low

2.6 Out of Scope

No.	Out of scope	Reason
1	Performance load testing	Given limited resources, extensive load testing with many users is not possible.

2	Complete review of security	Conducting a security review will not be possible due to limited resources and lack of domain knowledge.
3	Extensive platform compatibility testing	Due to not being able to get the system to work on different platforms it is impossible for us to test this.
4	Detailed connection efficiency analysis	Not having enough variation in our systems and the required number of users/systems to test this in the first place.
5	Data backup and recovery testing	While comprehensive testing is ideal, we are not able to ensure that basic data backup and recovery mechanisms are tested effectively or at all.
6	In-depth reporting and analytics	Since we have no access to the data collection, we are not able to effectively test this.
7	Comprehensive GDPR compliance audit	We do not have access to the true data collection that the application does, and we do not have the knowledge required about the domain to be able to give sufficient and accurate information.
8	Extensive documentation review	Limited access to the documentation related to the application. Since this is open source, the documentation might not be accurate and it's difficult to extract to analyze the whole thing.
9	Multilingual support for all languages	We are not able to test the application with a representative sample of languages based on user preferences and demographics. Due to not having the statistics required for this.
10	In-depth accessibility testing	While comprehensive accessibility testing is crucial, we can only ensure that the application meets basic accessibility requirements, and we can't test all of them fully in line with WCAG.

3. Test Deliverables and Schedule

Deliverable	Description	Date
Test Data Request	Describes a diverse set of data required to complete functional testing successfully	3 days before each test execution cycle
Test Execution Report	Detailed report of test execution activities and results.	After each test cycle
Test Status Report	High-level status report summarizing test progress and results.	Weekly
Defect Management	Management of defects found during testing.	Throughout the project
Report Document Revision	Updating this report	Weekly
Project Report	Final submission of report	Last day of July

4. Test Design

Test design is an important part of the testing process, where we formulate test cases to evaluate the quality and functionality of the Apache OpenMeetings. The aim is to ensure comprehensive and complete coverage of different functionalities, scenarios, and performance aspects of the project.

In this project we will use manual testing to ensure comprehensive and efficient testing. All the test cases have undergone a review phase. To accomplish this, after all the test cases were ready, we had an online meeting with all the team members in which we presented the test cases, and we went through them one by one. In the review session, we established a clear evaluation criterion for the test cases, and we checked if the test cases were in line with those criteria. After a thorough discussion with the team about each test case, we updated them and made the necessary changes to bring them in line with our vision. After the revision, we checked the test cases again and when everything was fine, we approved and finalized them.

Since in practice, it is crucial to apply different test design techniques together for more sophisticated specification, here we will go through the most popular test design techniques and identify how suitable they are for the Apache OpenMeetings project and whether we should use them or not:

4.1 Equivalence Class Testing

Equivalence class testing involves dividing the input data into separate groups or classes that should have similar behavior. This technique is useful when inputs can be categorized into distinct groups. This technique is quite suitable for Apache OpenMeetings as there are inputs like meeting parameters, user credentials, and configuration settings that can be categorized into equivalence classes, making this technique appropriate for testing.

4.2 State Transition Testing

State transition testing is a black box testing method, where the behavior of the application in response to changes in input conditions that can cause state changes or output changes are evaluated. This technique is useful for applications that have different behaviors based on their current state or previous inputs. This testing technique is not the most suitable one for Apache OpenMeetings. Even though Apache OpenMeetings may have certain states during its operation (like meeting started, meeting paused, meeting ended, etc.), these states are not crucial in terms of input conditions leading to different output behaviors.

4.3 Exploratory Testing

Exploratory testing is a dynamic kind of approach, where testers are actively exploring the application, learning about its functionalities while using it, and creating and executing test cases based on the observations. Testers only rely on their experience, skills, and creativity to uncover defects that might not be found in other testing methods. This method is quite suitable for Apache OpenMeetings, especially when it is complemented by other testing methods to help identify user experience and usability issues.

4.4 Boundary Value Analysis

Boundary value analysis involves testing the application at the boundaries of input ranges. Test cases are designed using values at the minimum, just above the minimum, just below the maximum, and at the maximum limit of the input domain. This technique is valuable for exposing issues that often occur at the edges of input ranges. This method

is also suitable for Apache OpenMeetings as testing input fields and functionalities using boundary values is very important to ensure the reliability of the software.

4.5 Pairwise Test Design

Pairwise test design (All-pairs testing) is another black box technique in which test cases are designed to test all combinations of each pair of input parameters. It is highly effective when there are multiple input parameters, and testing all combinations individually would be impractical. This method is also effective for testing Apache OpenMeetings. Given the complexity of configurations and input parameters, pairwise testing can efficiently cover various combinations in a manageable number of test cases.

4.6 Error Guessing Test Design

Error guessing is an informal test design technique where testers use their experience and intuition to predict potential errors in the system. Testers base their test cases on past experiences with similar systems or common error-prone areas. This method can be effective but may not be necessary as the other methods will cover the most key areas. But it can be used as an additional informal way to uncover defects and errors.

4.7 Use Case Testing

Use Case Testing is a black box testing technique that focuses on designing test cases based on the specific interactions or scenarios that end-users are likely to encounter while using the application. In Use Case Testing, test cases are derived from the use cases or user stories, and the primary goal is to verify if the application behaves as expected for these user-driven scenarios. Use Case Testing is highly suitable for Apache OpenMeetings. As an online collaboration tool, the application's success depends on providing a seamless user experience for different user interactions. Use Case Testing allows us to validate how well the application caters to common scenarios encountered by users during meetings and collaborations.

5. Test Execution

5.1 Approach

Black box testing is a software testing technique where the tester examines the functionality of the software without having access to its internal code or implementation details. Even if Apache's OpenMeetings is an open-source software, and we do have access to the code, the testing team focuses on testing the main external functionalities.

5.1.1 Smoke Testing

Smoke testing is a particular kind of black box testing that looks to see if the key components of the software are operating as intended. To make the program reliable enough for more extensive testing, it entails running a small number of test cases that focus on key functionality.

5.1.1.1 Objective

We use smoke testing to quickly confirm that Apache OpenMeetings' essential features are operating as intended for versions 7.1.0 and 7.2 (Demo).

5.1.1.2 Entry Criteria

The testing environment uses the most recent build or version of Apache OpenMeetings (7.2). The testing environment has been set up and is prepared.

5.1.1.3 Exit Criteria

- The identified smoke test scenarios have all been put into practice.
- Apache OpenMeetings' core features pass the smoke tests without any significant errors.

5.2 Test Data Requirements

- Test user accounts in the regular user role.
- Meetings that have been planned with multiple setups (different time zones, media settings, etc.).
- Simulated conversation messages and multimedia materials

5.3 Test Reporting

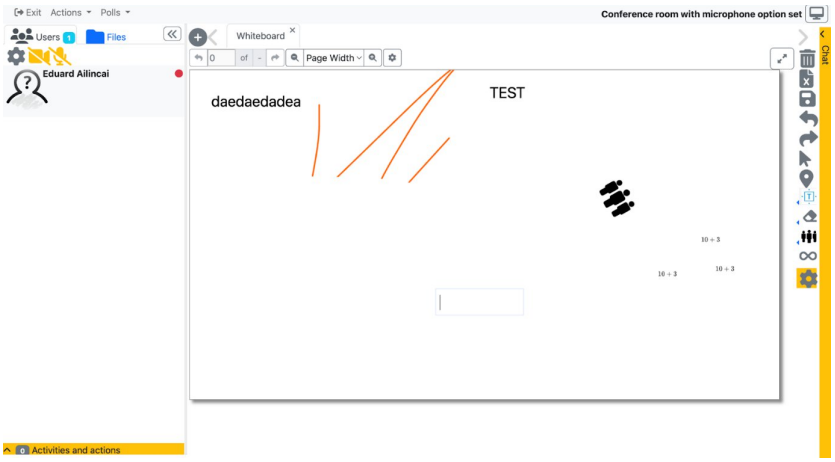
As this document serves as a vital project submission for the Software Quality Management course, the responsibility for the test execution reports lies with the dedicated smoke testing team members (Delia and Eduard). The intended audience includes the course instructors and fellow peers who will evaluate and assess this document's contents.

Report	**Objective/Content**	**Owner**	**Audience**	**Frequency**
Test Execution Report	Check to see if users can log in using legitimate credentials and that the login page is reachable.	Testing Team	Teachers, Project members	Regularly, after each test case
Test Execution Report	See whether you can successfully plan a new meeting.	Testing Team	Teachers, Project members	Regularly, after each test case
Test Execution Report	Check that users can access audio/video features and join a planned meeting.	Testing Team	Teachers, Project members	Regularly, after each test case
Test Execution Report	Make sure the meeting's basic chat capability functions.	Testing Team	Teachers, Project members	Regularly, after each test case
Test Execution Report	Verify that the process of recording a meeting goes smoothly.	Testing Team	Teachers, Project members	Regularly, after each test case
Test Execution Report	Check to see if users can log in	Testing Team	Teachers, Project members	Regularly, after each test case

	using Facebook and Google accounts			
Test Execution Report	Upload the profile picture to the account	Testing Team	Teachers, Project members	Regularly, after some test cases
Test Execution Report	Delete the profile picture of the account	Testing Team	Teachers, Project members	Regularly, after some test cases
Test Execution Report	Network testing tool (tests ping, jitter, upload and download speed)	Testing Team	Teachers, Project members	Regularly, after some test cases
Test Execution Report	Testing the functionalities of the calendar tool	Testing Team	Teachers, Project members	Regularly, after some test cases
Test Execution Report	Testing the functionalities of the public conference rooms	Testing Team	Teachers, Project members	Regularly, after some test cases

5.4 Defect Management

Test execution report	**Description**
Calendar tool	It seems that the button “Calendar” from the dashboard, as well as the button from the top menu does not work after uploading the profile picture / deleting the profile picture unless the page is refreshed.



6. Test Team

6.1 QA Team Organization

The team consists of 1 Tech Lead and 4 Testers.

6.2 Roles and Responsibilities

Role	Name(s)	Responsibilities
Test Lead	Eduard-Gabriel Ailincai	Plans and manages the entire testing process, including defining test objectives and strategies. Coordinates with the development team to gather requirements and ensure test coverage. Creates the Test Plan and assigns testing tasks to individual testers. Reviews and approves test cases and test results. Monitors the testing progress and ensures that testing activities align with the project schedule. Reports testing status, issues, and risks to project members. Conducts regular meetings with the QA team to provide guidance and support.
Tester	Pouya Amiri	Participates in the Test Plan preparation and test design process. Executes test cases based on the assigned testing tasks.

		Logs defects in the Defect Management tool (e.g., JIRA) with detailed information and steps to reproduce. Performs regression testing to ensure defects are fixed correctly. Assists in the preparation of test data and maintains test environments. Collaborates with the Test Lead to address any testing-related queries.
Tester	Delia Fliscu	Participates in test case design and execution. Conducts smoke testing for each QA build to ensure basic functionalities are working. Executes performance testing and analyzes system performance metrics. Logs and tracks defects, retests resolved defects, and verifies defect fixes. Collaborates with the development team to ensure timely resolution of issues.
Tester	Arlis Arto Puidet	Executes functional and non-functional test cases. Conducts usability testing to ensure the user interface is intuitive and user-friendly. Reports and prioritizes issues found during testing. Assists in test data preparation and maintenance. Collaborates with the Test Lead to improve the overall testing process.
Tester	Trieu Huynh Ba Nguyen	Executes test cases and records test results accurately. Performs compatibility testing on different browsers and devices.

		Conducts security testing and identifies potential vulnerabilities. Reports and communicates testing progress and issues to the Test Lead and other team members. Assists in the maintenance of test documentation and test artifacts.

7. Test Environment

7.1 Software and Hardware

No.	Machine	Purpose	Software Configuration	Hardware Configuration	Note
1	Dell Inspiron 15	Test environment	Windows 10	AMD Ryzen 5 5625U AMD Radeon Graphics 16 GB DDR4-2666 MHz RAM (2 x 8 GB) 512 GB PCIe® NVMe™ M.2 SSD	
2	HP Pavilion 15	Test environment	Windows 11	Intel® Core™ i7-1355U Intel® Iris® Xᵉ Graphics 16 GB DDR4-3200 MHz RAM (2 x 8 GB) 512 GB PCIe® NVMe™ M.2 SSD	
3	HP ZBook Firefly 14 G9	Test environment, Project management	Linux Mint 21.2 'Victoria'	Intel® Core™ i7-1255U NVIDIA Quadro T500 32 GB DDR4-3200 MHz RAM (2 x 16 GB) 512 GB PCIe® NVMe™ M.2 SSD	Apache OpenMeetings could not be installed
4	Apple MacBook Air	Test environment, Project management	MacOS Ventura	Apple M2 Pro (CPU and GPU) 16 GB DDR4-6400 MHz RAM	Apache OpenMeetings could not be installed.
5	Acer Nitro 5	Test environment, Project management	Windows 11	Intel® Core™ i5 11400H Nvidia RTX 3050Ti 16 GB RAM	

7.2 Tools

No.	Purpose	Tool
1	Meetings	Microsoft Teams
2	Scheduling and emails	Microsoft Outlook
3	Instant messaging	WhatsApp
4	File storage	Microsoft OneDrive
5	File editing	Microsoft Word
6	Version control	Git

8. References

openmeetings.apache.org. (n.d.). *Apache OpenMeetings Project – Home*. [online] Available at: https://openmeetings.apache.org/ [Accessed 28 Jul. 2023].

Copeland, L. (2004). *A Practitioner's Guide to Software Test Design*. Artech House.

Beizer, B. (2003). *Software testing techniques*. New Delhi: Dreamtech.

Sommerville, I. (2016). *Software engineering*. 10th ed. Boston, Mass. Amsterdam Cape Town Pearson Education Limited.