Quality Assurance Plan

actiTIME

<4th November 2023>

1 Introduction

1.1 Purpose

The purpose of this Quality Assurance Plan (QAP) is to describe the testing strategy and overall approach that will drive the testing of the actiTIME application. This QAP will provide a roadmap for testing activities and ensure that the actiTIME application meets all quality requirements.

1.2 Project Overview

actiTIME is a time tracking and project management application that helps businesses of all sizes track employee time, manage projects, and collaborate with team members. actiTIME is available as a cloud-based or on-premise solution.

The actiTIME application is a complex system with a wide range of features. The testing strategy for this project will focus on the following areas:

- Functionality: Test all features of the actiTIME application to ensure that they work as expected.
- Usability: Test the user interface and user experience of the actiTIME application to ensure that it is easy to use and navigate.
- Performance: Test the performance of the actiTIME application under load to ensure that it can meet the needs of users.
- Security: Test the security of the actiTIME application to ensure that it is protected from unauthorized access and malicious attacks.

The testing team will use a variety of testing methods to test the actiTIME application, including unit testing, integration testing, system testing, and acceptance testing. The testing team will also use test automation tools to help automate repetitive testing tasks.

The testing team will work closely with the development team to identify and fix defects. The testing team will also produce a test report that summarizes the results of the testing activities and provides recommendations for improvement.

2 Scope

2.1 In-Scope

The following features of the actiTIME application are in-scope for testing:

- Time tracking: Track employee time spent on projects, tasks, and breaks.
- Project management: Create and manage projects, tasks, and deadlines.
- Team collaboration: Share projects, tasks, and files with team members.
- Reporting: Generate reports on employee time, project progress, and more.

2.2 Out-of-Scope

The following features of the actiTIME application are out-of-scope for testing:

- Customizations: Any customizations that have been made to the actiTIME application by the customer.
- Third-party integrations: Any integrations with third-party applications.

- Performance under extreme load: The actiTIME application is designed to handle a large number of users and concurrent transactions. However, the testing team will not test the performance of the actiTIME application under extreme load conditions.
- Security under attack: The actiTIME application is designed to be secure from unauthorized access and malicious attacks. However, the testing team will not test the security of the actiTIME application under attack conditions.

The following are the justifications for the items that are out-of-scope:

- Customizations: Customizations are made to the actiTIME application by the customer to meet their specific needs. Therefore, it is the responsibility of the customer to test the customizations to ensure that they work as expected.
- Third-party integrations: The actiTIME application can be integrated with a variety of third-party applications. However, the testing team will not test all possible third-party integrations. The customer is responsible for testing any third-party integrations that they are using.
- Performance under extreme load: The actiTIME application is designed to handle a large number of users and concurrent transactions. However, the testing team will not test the performance of the actiTIME application under extreme load conditions. This is because it is difficult and expensive to simulate extreme load conditions in a testing environment.
- Security under attack: The actiTIME application is designed to be secure from unauthorized access and malicious attacks. However, the testing team will not test the security of the actiTIME application under attack conditions. This is because it is difficult and expensive to simulate all possible attacks in a testing environment.

If the customer requires testing of any of the out-of-scope items, this can be discussed with the testing team and a separate test plan can be created.

3 Testing Strategy

3.1 Product/Application/Solution Risks

Risks	Criticality	Mitigation Strategy
Performance issues	High	The testing team will perform performance testing to identify and fix any performance issues.
Security vulnerabilities	High	The testing team will perform security testing to identify and fix any security vulnerabilities.
Usability issues	Medium	The testing team will conduct usability testing to identify and fix any usability issues.
Data loss	Medium	The testing team will implement a data backup and recovery plan to prevent data loss.

Compatibility issues	Low	The testing team will test the actiTIME application on different browsers and operating systems to identify and fix any compatibility issues.

3.2 Level of Testing

Test Type	Description
Functional Testing	Functional testing ensures that all features of the actiTIME application work as expected.
Regression Testing	Regression testing ensures that changes made to the actiTIME application do not break existing functionality.
Non-Functional Testing	Non-functional testing ensures that the actiTIME application meets non-functional requirements such as performance, security, and usability.

3.2.1 Functional Testing

Functional testing will be performed to ensure that all features of the actiTIME application work as expected. The following types of functional testing will be performed:

- Unit testing: Unit testing will be performed to test individual units of code.
- Integration testing: Integration testing will be performed to test how different units of code work together.
- System testing: System testing will be performed to test the entire actiTIME application as a whole.
- Acceptance testing: Acceptance testing will be performed to ensure that the actiTIME application meets the customer's requirements.

3.2.2 Regression Testing

Regression testing will be performed to ensure that changes made to the actiTIME application do not break existing functionality. Regression testing will be performed after each code change.

3.2.3 Non-Functional Testing

The following types of non-functional testing will be performed:

- Performance testing: Performance testing will be performed to ensure that the actiTIME application can meet the performance requirements of the customer.

- Security testing: Security testing will be performed to identify and fix any security vulnerabilities in the actiTIME application.
- Usability testing: Usability testing will be performed to ensure that the actiTIME application is easy to use and navigate.

4. Test Approach

4.1 Test Design Approach

The test design approach for the actiTIME application will focus on the following areas:

- Functional testing: Test all features of the actiTIME application to ensure that they work as expected.
- Non-functional testing: Test the performance, security, and usability of the actiTIME application.

The following test design techniques will be used:

- Equivalence partitioning: Divide the input data into equivalence classes and test each equivalence class.
- Boundary value analysis: Test the minimum and maximum values for each input field.
- Error guessing: Test the actiTIME application with invalid inputs to identify any errors.
- Use case testing: Test the actiTIME application based on the use cases that have been developed.

4.2 Execution Strategy

4.2.1 Entry Criteria

The following entry criteria must be met before test execution can begin:

- The test environment(s) must be available.
- Test data must be available.
- Code must have been merged successfully.
- Development must have completed unit testing.
- Test cases and scripts must be completed, reviewed, and approved by the Project Team.

4.2.2 Exit criteria

The following exit criteria must be met before testing can be considered complete:

- 100% of test scripts must be executed.
- The pass rate of test scripts must be at least 90%.
- There must be no open critical or high severity defects.
- All remaining defects must be either canceled or documented as Change Requests for a future release.
- All expected and actual results must be captured and documented with the test script.
- All test metrics must be collected based on reports from daily and weekly status reports.
- All defects must be logged in the Defect Tracker/Spreadsheet.
- The test environment must be cleaned up and a new backup of the environment must be created.

4.3 Defect Management

Defects will be managed using a defect tracking system. The following defect management lifecycle will be used:

1. Defect discovery: The tester discovers a defect during testing.
2. Defect logging: The tester logs the defect in the defect tracking system.
3. Defect assignment: The project manager assigns the defect to a developer.
4. Defect fixing: The developer fixes the defect and checks in the fix.
5. Defect retesting: The tester retests the defect to verify that it has been fixed correctly.
6. Defect closure: The tester closes the defect if it has been fixed correctly.

The severity and impact of each defect will be classified as follows:

- Severity:
  - Critical: Functionality is blocked and no testing can proceed.
  - High: Functionality is not usable and there is no workaround but testing can proceed.
  - Medium: Functionality issues but there is a workaround for achieving the desired functionality.
  - Low: Unclear error message or cosmetic error which has minimum impact on product use.
- Impact:
  - High: The defect will have a significant impact on the users of the actiTIME application.
  - Medium: The defect will have some impact on the users of the actiTIME application.
  - Low: The defect will have little or no impact on the users of the actiTIME application.

5. Test Team Structure

5.1 Team Structure

#	Role	Resource Count
1	QA Manager	1
2	QA Leads	2
3	Senior QA Engineers	3
4	QA Engineers	4

5.2 Roles and Responsibilities

QA Manager

The QA Manager is responsible for the overall success of the testing effort. The QA Manager will:

- Develop and maintain the QA plan
- Manage the test team and its resources
- Coordinate with the development team to ensure that the actiTIME application is tested thoroughly
- Report on the progress of testing to the project manager

QA Leads

The QA Leads are responsible for leading and mentoring the QA team. The QA Leads will:

- Develop and maintain test cases and scripts
- Assign and track test cases to QA Engineers
- Review and approve test cases and scripts
- Provide support and guidance to QA Engineers
- Report on the progress of testing to the QA Manager

Senior QA Engineers

Senior QA Engineers are experienced QA professionals who have a deep understanding of testing principles and practices. Senior QA Engineers will:

- Develop and maintain test cases and scripts
- Execute test cases and scripts
- Report defects to the development team
- Work with the QA Leads to review and approve test cases and scripts
- Mentor QA Engineers

QA Engineers

QA Engineers are responsible for executing test cases and scripts, reporting defects to the development team, and working with the QA Leads to review and approve test cases and scripts.

In addition to the above roles and responsibilities, all members of the test team are responsible for the following:

- Communicating effectively with each other and with the development team
- Working together to achieve the goals of the QA plan
- Continuously improving the testing process

6. Test Schedule

The following test schedule is for the actiTIME application:

Test Phase	Duration	Start Date	End Date
Unit Testing	1 week	2023-11-06	2023-11-12
Integration Testing	2 weeks	2023-11-13	2023-11-25
System Testing	3 weeks	2023-11-26	2023-12-16

Acceptance Testing	2 weeks	2023-12-17	2023-12-29

This test schedule is based on the following assumptions:

- The development team will complete their work on time and deliver a working version of the actiTIME application to the test team by the start of each test phase.
- The test team will be able to execute all of the test cases and scripts in each test phase.
- There will be no major unforeseen problems during testing.

If any of these assumptions change, the test schedule may need to be adjusted.

The test schedule will be reviewed and updated weekly by the QA Manager in consultation with the QA Leads and the development team.

7. Test Reporting

7.1.Test Reporting Approach

The following test reports will be produced during the testing of the actiTIME application:

#	Report Name	Owner	Audience	Frequency
1	Test Progress Report	QA Manager	Project Manager, QA Leads, Development Team	Weekly
2	Test Summary Report	QA Manager	Project Manager, QA Leads, Development Team	At the end of each test phase
3	Final Test Report	QA Manager	Project Manager, Stakeholders	At the end of testing

1. The Test Progress Report will summarize the progress of testing, including the number of test cases executed, the number of defects found, and the status of open defects.
2. The Test Summary Report will provide a detailed overview of the test results, including the pass rate, fail rate, and blocked rate for each test phase. The Test Summary Report will also include information about the severity and impact of any defects that were found.
3. The Final Test Report will summarize the overall results of testing and provide recommendations for any further testing or remediation that is required.

7.2. Quality Matrices

The following quality matrices will be used to measure the quality of the actiTIME application:

- Test Coverage Matrix: This matrix will be used to track the percentage of test cases that have been executed for each feature of the actiTIME application.
- Defect Density Matrix: This matrix will be used to track the number of defects found per unit of code.
- Defect Severity Matrix: This matrix will be used to track the number of defects found for each severity level (critical, high, medium, and low).

- Defect Impact Matrix: This matrix will be used to track the number of defects found for each impact level (high, medium, and low).

The quality matrices will be used to identify any areas of the actiTIME application that need additional testing or remediation.

8. Test Environment Requirements

The following test environment requirements are required for testing the actiTIME application:

Hardware

- Server: 4-core CPU, 16GB RAM, 500GB disk space
- Database server: 4-core CPU, 16GB RAM, 500GB disk space
- Test client machines: 2-core CPU, 4GB RAM, 100GB disk space

Software

- Operating system: Windows 10 or 11, Ubuntu 20.04 or 22.04
- Database: MySQL 8.0 or higher, PostgreSQL 14 or higher
- Web browser: Chrome latest, Firefox latest, Edge latest
- Test automation framework: Selenium, Cypress, or TestCafe
- Defect tracking system: Jira, Bugzilla, or TestRail

Network

- 100 Mbps or faster internet connection
- Access to the production environment (optional)

Other

- Test data for all features of the actiTIME application

The test environment should be configured to be as similar as possible to the production environment. This will help to ensure that the actiTIME application is tested in a realistic environment.

The test environment should also be isolated from the production environment. This will help to prevent any changes made to the test environment from affecting the production environment.

The test environment should be well-documented. This will help to ensure that the test environment is used and maintained correctly.

9. Dependencies and Assumptions

The following dependencies and assumptions apply to the testing of the actiTIME application:

Dependencies

- The development team must deliver a working version of the actiTIME application to the test team by the start of each test phase.
- The test team must have access to the test environment and test data in order to execute the test cases.
- The test team must have the support of the development team to resolve any defects that are found.

Assumptions

- The development team will complete their work on time and deliver a working version of the actiTIME application to the test team by the start of each test phase.
- The test team will be able to execute all of the test cases and scripts in each test phase.
- There will be no major unforeseen problems during testing.

The QA Manager will be responsible for tracking the dependencies and assumptions and ensuring that they are mitigated. If any of the dependencies or assumptions change, the QA Manager will update the QA plan and notify the project manager.

Additional Dependencies and Assumptions

- Testing resources (e.g., testers, test environment, test data) will be available as needed.
- Deadlines will be realistic and achievable.
- Stakeholders will be supportive of the testing process.

Mitigation Strategies

- The QA Manager will work with the development team to create a realistic and achievable test schedule.
- The QA Manager will ensure that the test team has the necessary resources to execute the test cases.
- The QA Manager will communicate regularly with the project manager and stakeholders to keep them informed of the progress of testing and any potential risks.