

---

# Quality Assurance Plan

actiTIME

**4<sup>th</sup> September 2023**

# **1. Introduction**

## **1.1 PURPOSE**

The purpose of this Quality Assurance Plan is to outline the testing approach and strategy for actiTIME, focusing on the functionalities relevant to HR users within organizations. This plan aims to ensure that the software operates flawlessly, meeting user expectations and requirements.

## **1.2 PROJECT OVERVIEW**

actiTIME is a user-friendly time-tracking software that enables users to log work hours, time off, and sick leave effortlessly. It offers a robust reporting feature with numerous detailed reports suitable for various management and accounting purposes. The software also provides colorful charts for effective management and team analysis. Trusted by over 10,000 companies including industry giants like DHL, Huawei, Philips, and Xerox, as well as academic institutions like the University of Bristol, actiTIME offers a centralized solution for efficient time and project management.

# **2. Scope**

## **2.1 IN-SCOPE**

The scope of this QA plan will cover the following features relevant to a user playing the role of an HR within an organization using actiTIME.

- Log in with username and password.
- Manage employee profiles.
- Review employee leaves.
- Review attendance reports.
- Approve/reject timesheets.

## **2.2 OUT-OF-SCOPE**

The following features will not be tested as per this QA plan.

- Managing projects and assignments
- Billing and invoicing.
- Template management.
- API integration.
- Alert/Notifications.
- Customization

### 3. Testing Strategy

#### 3.1 PRODUCT/APPLICATION/SOLUTION RISKS

Risks	Criticality	Mitigation Strategy
Incorrect access control policies	Medium	Multi-factor authentication for all critical user roles, regular testing, testing of all possible edge cases, and continuous monitoring.
Performance issues under high user load	High	Perform load testing using simulated user loads to identify bottlenecks and optimize system performance. Optimize database queries, and use caching mechanisms.
Data entry errors	Medium	Implement user guides, simple interfaces, required form validation, conduct usability testing.
Inaccurate attendance data	Medium	Implement time-tracking validation and error correction features.
Delays in timesheet approval	Medium	Set clear approval workflows and notifications.
SQL injections	High	Penetration testing and code reviews, input validations, and regular updates of dependencies and libraries
Compatibility with different browsers	Medium	Thorough testing on all (or most popular) browsers and devices in the market.
Inadequate data backup and recovery mechanisms	High	Scheduled data backup. Regularly simulate system failures and test relevant recovery procedures.
Inaccurate absence data analysis	High	Test accuracy of reports for multiple scenarios, continuous monitoring.

### 3.2 LEVEL OF TESTING

Test Type	Description
Unit Testing	Involves testing individual components or modules of the software in isolation. This form of testing is typically implemented in all software development projects during development.
User Acceptance Testing (UAT)	Performed by end users to validate whether the application meets its requirements before it goes live. Clients/users can be involved in designing and executing these test cases.
Regression Testing	Involves re-running existing test cases to ensure that new code changes do not adversely affect the existing functionalities. A suite of regression tests covering core functionalities of actiTime can be maintained and executed after each feature addition or bug fix.
System Testing	Testing the entire application as a whole to ensure that it meets the specified requirements and behaves as expected. These tests should validate the end-to-end workflow and integration points within the application.
User Interface (UI) Testing	Validating the graphical user interface elements of the application to ensure they function correctly and provide a positive user experience. Test scripts to interact with UI components, validate form submissions, and ensure that user interactions and form inputs work as expected can be implemented and automated.
Load Testing	Evaluates the application's performance under a specific load, ensuring it can handle concurrent users without performance issues. User load can be simulated and test scenarios can be executed validating response time, resource utilization, and overall performance.
Volume Testing	Assesses the application's ability to handle a large volume of data, ensuring it performs well as the data size increases. A large number of timesheets, user profiles and leave records can be populated and the responsiveness and database performance can be evaluated.

### 3.2.1 Functional Testing

Software testing that focuses on confirming that an application's features and functionalities operate as intended and meet the requirements is known as functional testing. These tests verify the functionality of the system, data processing, user interactions, and software behavior. They include unit, integration, system, smoke, and other types of testing.

Functional tests are crucial to actiTIME because they guarantee that the application's key functions operate as intended and meet user expectations. They also aid in locating bugs and other issues with functionality and enhance the overall reliability of the application.

### 3.2.2 Regression Testing

Regression tests are a kind of software testing used to make sure that recent code modifications (bug patches, additions, or enhancements) don't negatively impact the application's current functions. It involves rerunning test cases that were previously executed. ActiTime is a sophisticated software program with an increasing amount of features, therefore regression testing will assist in identifying any unintentional bugs in current functionalities.

### 3.3.3 Non-Functional Testing

Non-functional tests are a sort of software testing that analyzes aspects of a system's performance, usability, dependability, and other non-functional properties to ensure that the intended level of standards is reached. Non-functional tests assess the system's reliability and user-friendliness as well as its performance under particular scenarios, in contrast to functional tests, which concentrate on particular features. These tests include performance testing, security testing, usability testing, scalability testing, and so on. ActiTime can ensure that its product not only functions properly but also operates efficiently, is safe, dependable, and complies with regulatory requirements by carrying out non-functional testing.

## 4. Test Approach

### 4.1 TEST DESIGN APPROACH

The test design strategy for the application will follow a combination of directed and regression-averse testing.

Analytical Test Design Strategy includes thoroughly evaluating the HR process requirements to fully understand what is expected of the application. Regression testing should be prioritized to promptly detect and fix issues caused by recent changes without affecting already-existing HR functionalities. The use of a combination of black-box and white-box testing techniques can be employed. Some of the black box testing techniques to be employed are,





- Equivalence partitioning
  - To test user login functionality, the set of valid and incorrect credentials will be divided into equivalence classes. Then to validate a representative sample from each class, test cases can be created.
  - Leave requests can be organized into categories such as half-day and full-day leaves. To validate user input such as hours worked and other data, sample test cases for each type of leave can be chosen with the help of equivalence partitioning.
- Decision table
  - A decision table will be generated with columns reflecting input criteria (valid username, valid password, and so on) and corresponding actions (successful login, error message displayed, and so on). To make sure all possible outcomes are covered, test cases can then be created based on combinations of these conditions.
- Boundary value analysis
  - A username with the minimum and maximum allowed characters will be evaluated to ensure that the application handles these scenarios correctly. In a similar vein, invalid usernames and passwords that are somewhat longer than permitted will also be examined.
  - To ensure proper attendance report production, boundary values for date ranges, such as the first and final day of the month, will be evaluated.

- State transition testing
  - To ensure the timesheet progresses through the proper approval workflow, state transition diagrams illustrating the various stages of a timesheet (such as submitted, approved, and rejected) will be made, and test cases will be constructed to validate transitions between these states.
  - Test cases will also be created to verify state transitions, like changing an employee's status from active to inactive, and make sure the application handles these changes appropriately.

## 4.2 EXECUTION STRATEGY

### 4.2.1 Entry Criteria

- The entry criteria refer to the desirable conditions to start test execution.
- Entry criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions, and provide a recommendation.

Entry Criteria	Conditions	Comments
Test environment(s) is available		This includes all criteria mentioned in section 9 below.
Test data is available and sanitized for testing purposes.		Test data provided by actiTIME to its users can be used.
Code has been merged successfully from development branches.		-
Developers have completed unit testing		-
Test cases and scripts are completed, reviewed, and approved by the Project Team		-

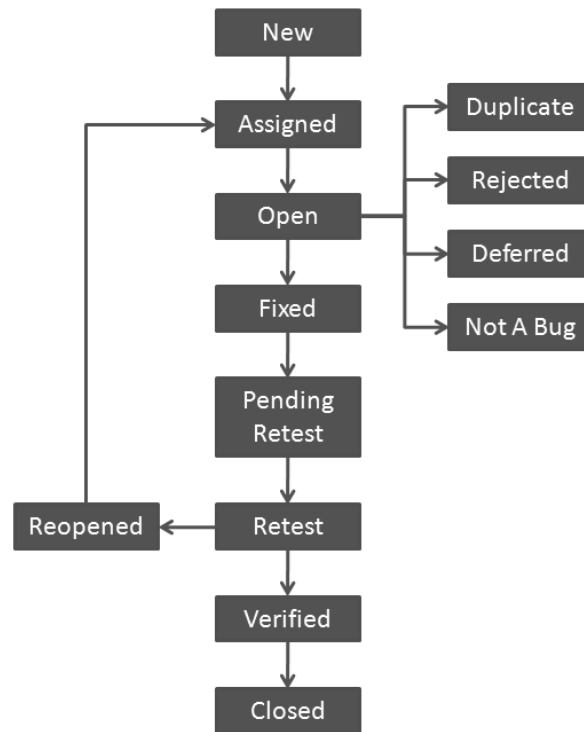
#### 4.2.2 Exit criteria

- The exit criteria are the desirable conditions that need to be met to proceed with the implementation.
- Exit criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions, and provide a recommendation.

Exit Criteria	Conditions	Comments
100% Test Scripts executed		
90% pass rate of Test Scripts		
No open Critical and High-severity defects		
All remaining defects are either canceled or documented as Change Requests for a future release		
All expected and actual results are captured and documented with the test script		
All test metrics collected based on reports from daily and Weekly Status reports		
All defects logged in Defect Tracker/Spreadsheet		
Test environment cleanup completed and a new backup of the environment		



## 5. Defect Management



- It is expected that the testers execute all the scripts in each of the cycles described above.
- The defects will be tracked through the Defect Tracker or Spreadsheet.
- It is the tester's responsibility to open the defects, retest, and close them.
- Defects found during the Testing should be categorized as below:

Severity	Impact
1 (Critical)	Functionality is blocked and no testing can proceed. The application/program/feature is unusable in the current state.
2 (High)	Functionality is not usable and there is no workaround but testing can proceed
3 (Medium)	Functionality issues but there is a workaround for achieving the desired functionality
4 (Low)	Unclear error message or cosmetic error which has minimum impact on product use.

## 6. Test Team Structure

### 6.1 TEAM STRUCTURE

#	Role	Resource Count
1	QA Manager	1
2	QA Leads	2
3	Senior QA Engineers	3
4	QA Engineers	6

### 6.2 ROLES AND RESPONSIBILITIES

#### QA Manager

QA Manager is in charge of managing the complete QA procedure. He/she creates the quality assurance plan, assigns resources, establishes the QA strategy, and ensures that testing activities complement the overarching business goals. The QA Manager works together with other teams to set standards and guidelines for quality.

Some of the responsibilities include

- Develop QA strategies and policies.
- Allocate resources and assign tasks to QA leads and engineers.
- Collaborate with stakeholders to understand business requirements.
- Monitor overall QA progress and report to upper management.
- Ensure QA processes align with industry standards and best practices.

#### QA Leads

QA Leads serve as team leaders, coordinating testing activities, maintaining test plans, and communicating with QA engineers, developers, and the QA Manager. They are in charge of the QA team's daily operations.

Some of their responsibilities include

- Report test progress to the QA Manager.
- Assign tasks to QA engineers based on project priorities.

- Coordinate with developers and other stakeholders to clarify requirements.
- Conduct regular team meetings to track progress and address challenges.
- Review and approve test cases and test results.

### Senior QA Engineers

Senior QA Engineers are experts who concentrate on complex testing situations as well as crucial features. They also frequently serve as mentors for junior QA engineers. They are in charge of executing test cases, identifying and reporting defects, and guaranteeing the reliability and quality of the application.

Some of their responsibilities include

- Perform exploratory testing to identify unforeseen issues.
- Collaborate with QA leads to create comprehensive test scenarios.
- Mentor junior QA engineers and guide testing best practices.
- Report and track defects, ensuring they are addressed by developers.
- Contribute to test automation efforts.

### QA Engineers

QA engineers are in charge of executing test cases, finding and reporting issues, and ensuring the software's usability and functionality. They collaborate closely with senior QA engineers and execute functional, regression, and usability testing according to test plans.

Some of their responsibilities include

- Execute test cases as per the defined test plans.
- Document test results and report defects accurately.
- Participate in team meetings and provide updates on testing progress.
- Assist in creating and maintaining test documentation.
- Acquire and prepare test data to be used in tests.

## 7. Test Schedule

Test Activity	Month					
	1	2	3	4	5	6
Test Planning	■	■				
Test Environment Setup	■	■	■	■		
Test Case Design		■	■	■		
Test Automation Development		■	■	■	■	
Test Execution				■	■	■
Test Summary and Reporting			■	■	■	■

## 8. Test Reporting

### 8.1. QUALITY MATRICES

- **Percentage of planned work done in test case preparation.**
  - Helps assess the team's progress in creating necessary test scenarios and identify delays or gaps in test coverage.
- **Percentage of planned work done in test environment preparation.**
  - Ensures that the necessary infrastructure is in place for testing activities to proceed as planned.
- **Pass/Fail Rate (percentage of test cases that pass and fail during testing).**
  - Gives an overview of the overall test success and areas that may require improvement.
- **Defect Density (number of defects discovered per unit of code).**
  - Helps to assess the code quality and the effectiveness of defect detection during testing.
- **User Satisfaction**
  - Provides a qualitative measure of user satisfaction and helps identify areas for improvement.
- **Test Automation Coverage (for automated testing)**
  - Ensures that essential scenarios are covered by automated tests, providing faster feedback.

## **9. Test Environment Requirements**

### **9.1. HARDWARE**

A separate environment that is identical to having the same specifications (server type, RAM, CPU, containers) as the production environment should be created for testing. During performance testing, this will ensure that the results obtained are accurate and equal to the performance of the application that is deployed to end-users.

The team should also have access to devices with different screen sizes (smartphones, tablets) to test the interface of actiTIME in each of them. This helps ensure the application's responsiveness and usability on mobile devices, enhancing the user experience for users accessing actiTIME on the go.

### **9.2. SOFTWARE**

Some features or functionalities might behave differently on different operating systems. For example, file handling, system notifications, and keyboard shortcuts. Testing actiTIME on multiple platforms helps identify and address any such OS-specific issues. Therefore, members of the QA team must have separate virtual/physical machines with different Operating Systems (Windows, macOS, Linux, Android, iOS) to test the application's behavior in each environment. In addition to this, a separate database should be prepared for testing purposes and it should be populated with somewhat realistic data sufficient for testing different scenarios.

Different browsers interpret web pages differently. Testers should also have different web browsers (such as Chrome, Firefox, Safari, and Edge) installed to ensure that the application functions correctly and appears consistently across each of them. It is also important to test the application for different versions available for each type of browser.

### **9.3. BACKUP AND RECOVERY**

Regular backup procedures to prevent data loss in case of system failures or unexpected errors during testing should be implemented. A reliable backup and recovery mechanism ensures that test data can be restored to a known state when needed.

### **9.4. NETWORK**

Testers should have stable network connectivity with adequate bandwidth to support concurrent user access and data transmission between clients and the server.

## **10. Dependencies and Assumptions**

### **10.1. DEPENDENCIES**

- The completion of development milestones is required before testing can begin. The testing timeline may be affected by requirements changes or software development delays.

### **10.2. ASSUMPTIONS**

- The project plan assumes that essential resources, such as test management tools and testing professionals, would be allocated.
- Sample data provided by the actiTIME application to its new users is sufficient and can be used for all the planned testing activities.
- There is only one team involved in the testing of actiTIME and the team includes the members mentioned in 6.1 above.
- The duration allocated for the testing is 6 months.