# Quality Assurance Plan

## actiTime HR Management Application

3rd November 2023

**H P N H Pathirana**
**19020546**
IS 4102 - Advanced Software Quality Assurance

# 1 Introduction

## 1.1 Purpose

A Quality Assurance (QA) Plan for the actiTIME HR management demo application serves the purpose of ensuring that the software is developed, tested, and delivered with high quality and reliability.

The document covers the,

- Project overview
- Testing Scope
- Test Strategy
- Test Approach
- Test Schedule
- Test Reporting

## 1.2 Project Overview

actiTIME is an online software solution created to assist businesses and institutions in effectively handling their projects, monitoring employee work hours, and evaluating team productivity. It provides various functions, including time monitoring, task coordination, project administration, and data analysis.

# 2 Scope

## 2.1 In-Scope

1. HR login with valid username and password
2. View employee profiles
3. Review leaves and attendance reports
4. Reviewing for approval/rejection of timesheets

## 2.2 Out-of-Scope

The below features will not be tested within the scope of this Quality Assurance plan given that the testing will be solely focused on the core functionality HR management of this application.

1. Customer and project management feature
2. Creation and management of tasks
3. Lock time tracking feature
4. New user (employee) addition feature
5. Managing work assignments feature
6. Financial, perf reports

# 3 Testing Strategy

## 3.1 Product/Application/Solution Risks

| Risks | Criticality | Mitigation Strategy |
|---|---|---|
| Unauthorized access by others | High | Enhance user authentication by implementing a strong and secure method, such as multi-factor authentication. |
| Data leakage or privacy breach | High | Secure confidential information through encryption and enforce rigorous access restrictions. |
| Poor performance under increased load | High | Perform load testing and enhance system efficiency. |
| Incompatible browsers/devices | Medium | Ensure cross-browser and cross-device compatibility during development. |
| Compliance violations | High | Implement regular compliance training for the testing team and conduct thorough compliance reviews during testing phases. |
| User errors in data entry | Medium | Provide user-friendly interfaces and validation checks. |

| | | |
|---|---|---|
| System downtime or outages | High | Implement redundancy measures by deploying backup servers and data centers to ensure continuous availability.<br>Regular monitoring and performance testing conducted to proactively identify and address potential issues before they lead to significant downtime. |
| Data inaccuracy and incompletion | Medium | Implement data validation checks at the input stage to prevent incorrect or incomplete data from being entered.<br>Conduct thorough data reconciliation and verification procedures during testing to identify and rectify any inconsistencies or errors<br>comprehensive training provided to users and testers to ensure accurate data entry practices and ongoing data integrity. |
| Inconsistent reporting | Medium | Implement standardized reporting templates and guidelines to ensure uniformity and consistency in report generation.<br>Employ automated reporting tools to reduce manual data entry errors and streamline the reporting process. |
| Security vulnerabilities | High | Implement regular security assessments and code reviews to identify and address potential vulnerabilities in the software, with a focus on areas like data handling, authentication, and authorization. Ensure that the application and its underlying infrastructure are kept up-to-date. |

## 3.2 Level of Testing

| Test Type | Description |
|---|---|
| Functional Testing | Functional testing is conducted to evaluate whether the software application's features and functions perform as expected, based on the specified requirements. It verifies that the application meets its intended purpose. |
| Non-functional Testing | Non-functional testing assesses the performance, reliability, scalability, and security aspects of a software system, ensuring it meets quality standards beyond basic functionality. |
| Regression Testing | Regression testing is conducted to verify whether recent code changes have impacted the existing functionality of an application by retesting previously tested features to ensure they still work as expected. |

### 3.2.1 Functional Testing

| | |
|---|---|
| Unit Testing | Unit Testing for the application involves testing individual components or functions in isolation to ensure they work correctly.<br>For example, testing the "HR login with valid username and password" functionality would involve verifying that the login mechanism correctly authenticates HR users. Unit tests could check if the correct username and password combination grants access while incorrect ones are denied. |
| Integration Testing | Integration Testing focuses on verifying the interactions between different components or modules of the application. For instance, when "Viewing employee profiles," integration testing ensures that the employee profile information is correctly retrieved from a database, displayed in the user interface, and that the data matches what is expected. It would test that data is seamlessly integrated and presented accurately. |
| System Testing | System Testing evaluates the application as a whole and checks if it meets the specified requirements. |

### 3.2.2 Regression Testing

**HR Login with Valid Username and Password:** Regression testing in this scenario involves verifying that after any software updates or changes, the login functionality for HR users continues to work correctly.
Example: After an application update, HR personnel should be able to log in with their valid usernames and passwords without encountering any errors or issues.

**View Employee Profiles:** This regression test ensures that HR can still access and view employee profiles after system modifications.
Example: After a software upgrade, HR staff should be able to view employee profiles to check their personal and professional information, ensuring that no data is missing or displayed incorrectly.

**Review Leaves and Attendance Reports:** Regression testing for this function validates that HR can generate and review leave and attendance reports without any unintended changes to the reporting mechanism.
Example: Post an application update, HR should be able to generate monthly attendance reports for all employees and verify that the data is accurate.

**Reviewing for Approval/Rejection of Timesheets:** This test ensures that the functionality allowing HR to review and approve/reject timesheets remains intact and error-free after any system changes.
Example: Following a software modification, HR must be able to review timesheets submitted by employees, approve or reject them, and verify that the approval workflow functions as expected without any issues.

### 3.3.3 Non-Functional Testing

| | |
|---|---|
| Load Testing | Load testing evaluates how well the actiTime HR management application can handle an expected volume of users and data. It aims to determine its performance under typical usage conditions.<br><br>Example: Simulate a scenario where 100 HR personnel log in simultaneously, access employee profiles, review leaves and attendance reports, and approve/reject timesheets to ensure the application maintains acceptable response times and doesn't crash under this load. |

| | |
|---|---|
| Stress Testing | Stress testing goes beyond load testing by pushing the application to its limits to identify the breaking point. It helps understand the application's stability under extreme conditions.<br><br>Example: Increase the user load well beyond what's typical, such as having 1000 users attempting to log in simultaneously with multiple requests for various actions, including viewing reports and timesheet approvals. The goal is to identify system vulnerabilities or failures under stress. |
| Usability Testing | Usability testing assesses the user-friendliness and ease of interaction with the actiTime HR management application, focusing on how well users can accomplish tasks efficiently and intuitively.<br><br>Have a group of HR professionals interact with the application to perform tasks like HR login, accessing employee profiles, reviewing reports, and timesheet approval. Collect feedback on their experience, ease of navigation, and any issues encountered to improve user satisfaction. |

# 4. Test Approach

## 4.1 Test Design Approach

The test design strategy for the HR process of actiTIME aims to ensure the reliability, security, and performance of the HR features. It will employ a combination of test design techniques to thoroughly evaluate the functionality and quality of the application.

A suitable test strategy for the actiTime HR management application can be a combination of several strategies to ensure comprehensive and effective testing. A combination of the "Analytical" and "Process-compliant" test strategies is recommended

Analytical Test Strategy:
This strategy involves a deep analysis of the application's requirements and design to determine the testing approach. It's particularly important for a complex application like HR management software.

Process-compliant Test Strategy:
HR management applications often involve handling sensitive employee data and compliance with regulations like GDPR or labor laws. A process-compliant strategy ensures that the testing aligns with these regulations and follows industry best practices.

Analytical testing will help identify critical functionalities and potential risks within the HR application. Testers can analyze the requirements thoroughly to create test cases that cover essential features, ensuring comprehensive coverage.
Process-compliant testing will ensure that the testing process adheres to regulatory and industry standards. This is crucial to maintain data security and compliance, which are paramount in HR software

### 4.1.1 Test Design Techniques

| Test Type | Description | Application |
|---|---|---|
| **Boundary Value Analysis** | This technique involves testing the extreme boundaries of valid and invalid login credentials. Test cases are designed to cover scenarios where the input data is at the edge of what's acceptable. | For example, for the HR login, you would test with the minimum and maximum password length, as well as incorrect or missing credentials. |
| **Decision Table** | Decision tables are used to systematically test different combinations of inputs and conditions. | In the context of HR login, this would involve creating a table that lists various possible scenarios, including valid and invalid credentials, and the expected outcomes for each combination. |
| **State Transition** | State transition testing is employed to assess how an application responds to different states or conditions | In the case of timesheet approval and rejection, you would identify the various states (e.g., pending, approved, rejected) and test how the application transitions between these states based on user interactions and system rules. |
| **Error Guessing** | Error guessing is an informal technique where testers use their | In this case, testers would explore the application, especially the |

| | | |
|---|---|---|
| | experience and intuition to identify potential issues. | access to leave and attendance reports, to guess and identify potential issues or vulnerabilities based on their knowledge of common problems or user behaviors. |
| **Equivalence Partitioning** | Equivalence partitioning involves dividing input data into partitions or groups based on similar characteristics. | For HR input regarding timesheet approval and rejection, you'd identify different categories or equivalence partitions, such as valid input, invalid input, and boundary cases. Testing is then focused on these partitions to ensure that various scenarios are covered effectively. |

## 4.2 Execution Strategy

### 4.2.1 Entry Criteria

- *The entry criteria refer to the desirable conditions in order 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 | ✓ | |
| Test data is available | ✓ | |
| Code has been merged successfully | ✓ | |
| Development has 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 in order 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 back up of the environment | | |

## 3.3 Defect Management

The defect management lifecycle chart involves a circular process, as defects may be identified at any stage of software development. The goal is to address defects promptly and effectively, ensuring that the software meets quality standards and user expectations.

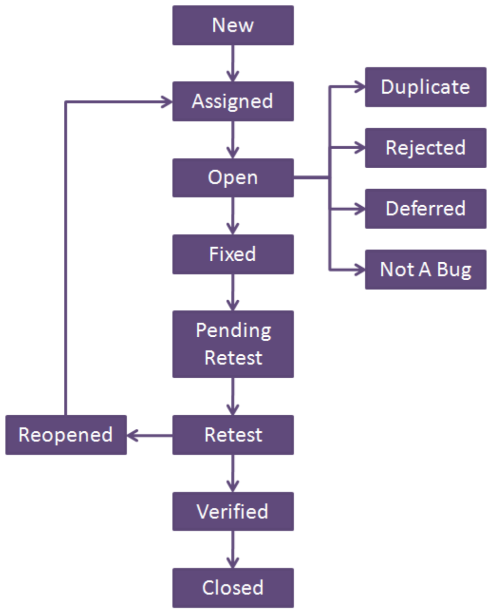


*Figure1 : Defect Management Lifecycle*

- It is expected that the testers execute all the scripts in each of the cycles described above.
- The defects will be tracked through 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<br>▪ 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. |

## 5. Test Team Structure

### 5.1 Team Structure

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

### 5.2 Roles and Responsibilities

**QA Manager**

Role: The QA Manager is responsible for the overall quality assurance and testing process within the organization.

Responsibilities:

- Strategy and Planning: Develop and implement the overall QA strategy, including test plans, test cases, and testing methodologies.
- Resource Management: Allocate resources, manage the QA team, and ensure proper staffing.

- Quality Standards: Establish and enforce quality standards and best practices.
- Budget Management: Manage the QA budget, ensuring cost-effective testing processes.
- Stakeholder Communication: Communicate with stakeholders about the quality and testing progress.

**QA Leads**

Role: QA Leads are responsible for leading and managing smaller QA teams or specific testing projects.

Responsibilities:
- Test Planning: Develop test plans, strategies, and schedules.
- Team Leadership: Lead and manage the QA team, providing guidance and support.
- Test Execution: Oversee and participate in test execution, including the creation of test cases.
- Mentoring: Mentor and train junior QA engineers.

**Senior QA Engineers**

Role: Senior QA Engineers are experienced testing professionals responsible for test execution and providing expertise in various testing areas.

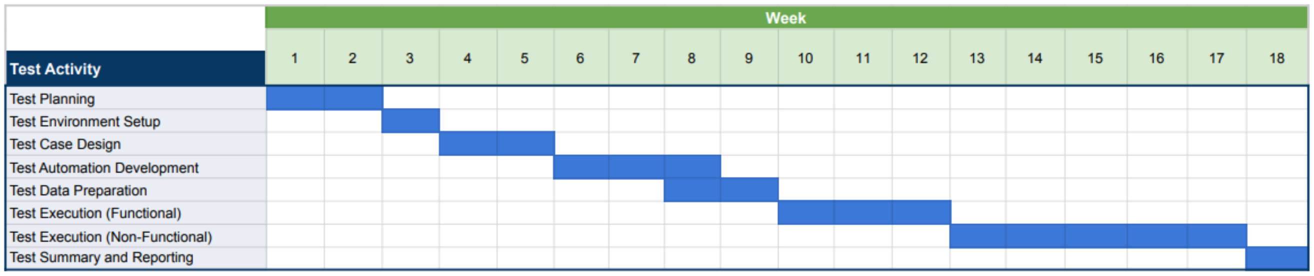
Responsibilities:
- Test Design: Create detailed test cases and test scripts based on requirements and specifications.
- Test Execution: Execute test cases and record results.
- Test Automation: Develop and maintain test automation scripts and frameworks.
- Regression Testing: Conduct regression testing to ensure existing functionality is not impacted by changes.
- Exploratory Testing: Perform exploratory testing to find hidden defects.
- Defect Reporting: Document and report defects with detailed information.

**QA Engineers**

Role: QA Engineers are responsible for performing the actual testing of software applications.

Responsibilities:
- Test Execution: Execute test cases and report results.
- Regression Testing: Conduct regression testing on new software versions.
- Defect Reporting: Document and report defects with clear and concise details.
- Test Data Preparation: Prepare necessary test data sets.
- Adherence to Processes: Follow established testing processes and methodologies.
- Documentation: Maintain test documentation and records.
- Continuous Learning: Stay updated on testing techniques and tools.

## 6. Test Schedule



| Test Activity | Week 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Test Planning | 🟦 | 🟦 | | | | | | | | | | | | | | | | |
| Test Environment Setup | | | 🟦 | | | | | | | | | | | | | | | |
| Test Case Design | | | | 🟦 | 🟦 | | | | | | | | | | | | | |
| Test Automation Development | | | | | | 🟦 | 🟦 | 🟦 | | | | | | | | | | |
| Test Data Preparation | | | | | | | | 🟦 | 🟦 | | | | | | | | | |
| Test Execution (Functional) | | | | | | | | | | 🟦 | 🟦 | 🟦 | | | | | | |
| Test Execution (Non-Functional) | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | |
| Test Summary and Reporting | | | | | | | | | | | | | | | | | | 🟦 |

*Figure2 : Gantt Chart of the test schedule*

## 7. Test Reporting

### 7.1.Test Reporting Approach

| # | Report Name | Owner | Audience | Frequency |
|---|---|---|---|---|
| 1 | Test Progress Report | QA Manager/ QA Lead | Project managers, Development teams, Senior management | Weekly/Bi-Weekly |
| 2 | Test Summary Report | QA Manager/ QA Lead | Project managers, Senior management | End of a testing phase |

### 7.2. Quality Matrices

1. **Test Coverage**: Evaluate the comprehensiveness of testing efforts, which includes assessing how much of the codebase, requirements, and functions have been tested.
2. **Pass/Fail Rate**: Keep track of the percentage of test cases that have passed successfully and those that have failed to identify areas that require further attention.
3. **Defect Severity Distribution**: Categorize defects based on their severity levels to prioritize critical issues for immediate resolution and to guide quality improvement efforts.
4. **User Satisfaction**: Collect qualitative feedback from end-users during User Acceptance Testing (UAT) to gauge their contentment with the software and identify areas for enhancement.
5. **Test Execution Time**: Monitor the time taken to execute test cases, helping in assessing test efficiency and identifying potential bottlenecks in the testing process.

6. **Defect Density**: Measure the number of defects identified per unit of code, providing insights into the software's overall quality and stability.
7. **Test Automation Coverage**: Determine the percentage of test cases that have been automated, aiding in the evaluation of automation efforts and their impact on testing efficiency and coverage.

## 8. Test Environment Requirements

**Hardware Requirements:**

Server:
- Sufficient processing power and memory.
- Adequate storage capacity.

Client Machines:
- Various devices (desktops, mobile) for compatibility testing.
- Diverse operating systems and screen resolutions.

**Software Requirements:**

Operating Systems:
- Compatibility with multiple OS (Windows, macOS, Android, iOS).
- Support for various web browsers (e.g., Chrome, Firefox).

Database:
- Relational database system (e.g., MySQL, PostgreSQL).
- Compatibility with different versions of the app for regression testing.

**Network Requirements:**

Internet Connectivity:
- Stable internet connections for web-based app access.
- Secure network infrastructure to protect data.

**Additional Requirements:**

Test Data:
- Realistic and diverse HR data.
- Security, performance, and accessibility testing tools.

Test Management:
- Tools for test case management and reporting.
- Compliance and backup mechanisms for data integrity.

## 9. Dependencies and Assumptions

### Dependencies:

1. Development Milestones: Testing relies on the completion of development milestones. Delays or requirement changes can affect the testing schedule.
2. Resource Availability: Adequate testing resources, including hardware and software, are essential for comprehensive testing.
3. Test Data Availability: Effective testing depends on having realistic and representative HR data, including employee profiles, leave records, and timesheets.

### Assumptions:

1. Sufficient Test Coverage: Testing assumes that planned coverage, both functional and non-functional, will ensure product quality.
2. Compliance with Standards: The application and data handling are assumed to comply with industry standards and regulations.
3. Resource Allocation: Necessary resources, such as test management tools and personnel, will be allocated as per the project plan.
4. Timely Test Environment Setup: The test environment, including hardware, software, and network infrastructure, will be set up promptly.
5. Stakeholder Availability: Availability of stakeholders, including end-users and domain experts, is assumed for tasks like user acceptance testing and feedback.
6. Data Privacy and Security: Adequate measures are in place to protect sensitive HR data during testing.
7. Documentation and Requirements: Testing assumes access to up-to-date documentation and clear, complete, agreed-upon HR process requirements.