Quality Assurance Plan

ActiTime - HR Management Application

4th September 2023

19020262 - S.C Gallage

1 Introduction

1.1 Purpose

The purpose of the quality assurance plan is to document the evaluation process of the HR feature of the actiTIME application.

1.2 Project Overview

ActiTIME is a web-based application for project management and time monitoring that assists businesses and organizations in monitoring work hours, managing projects, and assessing team performance. It consists of features such as reporting, task and project management, and time monitoring.

2 Scope

2.1 In-Scope

1. Login functionality for HR personnel
2. View employee profiles
3. Review leaves and attendance reports
4. Reviewing for approval/rejection of timesheets

2.2 Out-of-Scope

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, performance reports

3 Testing Strategy

3.1 Product/Application/Solution Risks

Risks	Criticality	Mitigation Strategy
Unauthorized access	High	Implement robust user authentication mechanisms such as multi-factor authentication.

Data breach	High	Implement strict access control mechanisms and encrypt sensitive information.
System downtime	High	Regularly observe the well-being of the system and establish backup measures for essential elements.
Poor system performance under load	High	Conduct load testing and optimize system performance
User interface errors, errors in user data entry.	Low	Provide user-friendly interfaces and implement validation checks.
Incorrect attendance records.	Medium	Introduce mechanisms for verifying time-tracking accuracy and error correction mechanism.
Timesheet approval delays	Medium	Establish transparent authorization processes and alerts.
Incomplete data	Medium	Enforce mandatory fields and provide data validation.
Security Risks	High	Regularly implement updates and patches for both the application and infrastructure.
Compliance violations	High	Periodically examine and revise human resources policies and procedures to ensure adherence to regulations.

3.2 Level of Testing

Test Type	Description
Functional Testing	Ensuring that the application's capabilities and attributes align with the specified requirements and operate as planned.
Regression Testing	Testing is conducted to ensure that new changes do not negatively impact existing functionality.

Non-Functional Testing	Assessing non-functional aspects, such as reliability, scalability, performance, and security, to ensure the app meets quality standards.

3.2.1 Functional Testing

Test Type	Description
Unit Testing	Testing individual components (functions, methods) to ensure they work correctly in isolation.
Integration Testing	Testing to verify that different modules or components within the app interact correctly when integrated.
System Testing	Testing the entire system as a whole to validate that all components work together as expected.

3.2.2 Regression Testing

Test Type	Description
Selective Regression Testing	Prioritizing testing on susceptible software sections to ensure the stability of critical functions, minimizing time and resource consumption.
Retest-All Regression Testing	Re-executing all tests to ensure no new issues, maintaining application integrity postcode changes.
Partial Regression Testing	Focuses on affected segments to guarantee their proper functionality while conserving testing resources by bypassing unchanged areas.

3.3.3 Non-Functional Testing

Test Type	Description
Load Testing	Evaluating the system's performance under expected load conditions to ensure it can handle typical user activity.
Stress Testing	Testing the app's resilience by subjecting it to extreme conditions beyond normal usage to identify potential failure points.

Compatibility Testing	Ensuring that the app functions correctly on various browsers, operating systems, and devices.
Security Testing	Evaluating the app's security measures to identify and address vulnerabilities and protect against unauthorized access.

4. Test Approach

4.1 Test Design Approach

Combining analytical and directed test design strategies for testing actiTIME's features provides a complete and targeted validation approach. The features of actiTIME could be thoroughly and effectively tested by QA teams using a combination of analytical understanding and focused attention, resulting in a solid and dependable application.

Test design techniques that can be employed:

- Boundary Value Analysis
    - For the login functionality, boundary value analysis involves testing the system's behavior at the edges of valid and invalid input ranges. This includes testing usernames and passwords at their minimum and maximum lengths.
- Decision Table
    - Decision tables map different input conditions (valid username, valid password, etc.) to corresponding actions (successful login, error message displayed, etc.). Test cases can be generated based on combinations of these conditions, ensuring comprehensive coverage of login scenarios.
- State Transition
    - State transition testing involves creating a diagram representing different states of timesheets (e.g., submitted, approved, rejected). Test cases are designed to validate transitions between these states, ensuring that timesheets move through the correct approval workflow.
- Equivalence Partitions
    - Equivalence partitioning involves dividing inputs into different classes, such as timesheets with 8 hours, 4 hours, or other specific durations to approve or reject. Test cases are designed for representative values from each partition.

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	✓	Mentioned in section 9
Test data is available	✓	Sample data provided by actiTIME
Code has been merged successfully	✓	-
Development has completed unit testing	✓	-
Test cases and scripts are completed, reviewed and approved by the Project Team		-

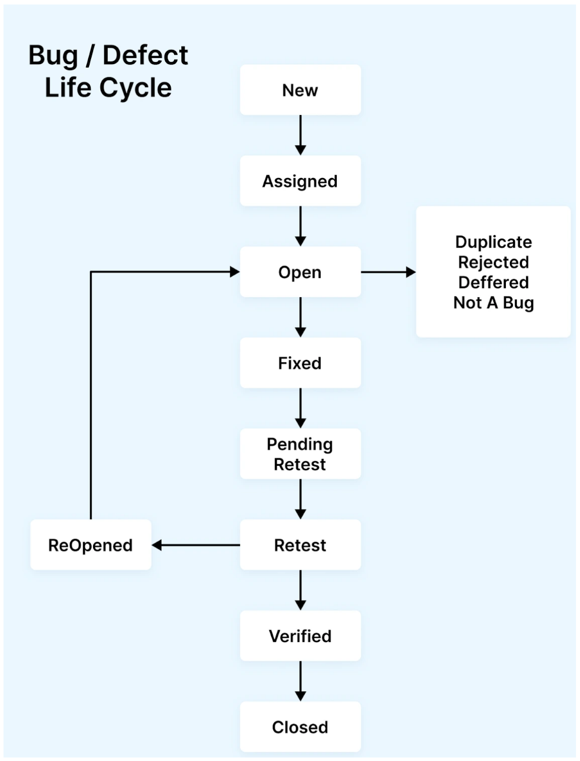
4.2.2 Exit criteria

- 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		

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 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	1
2	QA Leads	2
3	Senior QA Engineers	4
4	QA Engineers	7

5.2 Roles and Responsibilities

QA Manager

- Overall Test Strategy: Define the overall test strategy, including objectives, scope, and resources required for testing the HR process of actiTIME.
- Resource Allocation: Allocate resources and personnel to specific testing areas and coordinate test activities across the team.
- Risk Management: Identify and assess potential risks in the testing process and develop mitigation strategies.

QA Leads

- Test Planning: Develop detailed test plans and strategies for specific testing levels and areas within the HR process.
- Test Case Design: Oversee the creation of test cases, ensuring that they are comprehensive and align with requirements.
- Test Execution: Coordinate and monitor test execution, making sure that test cases are executed as planned.
- Defect Management: Oversee the defect tracking and resolution process, ensuring that identified issues are documented and addressed.

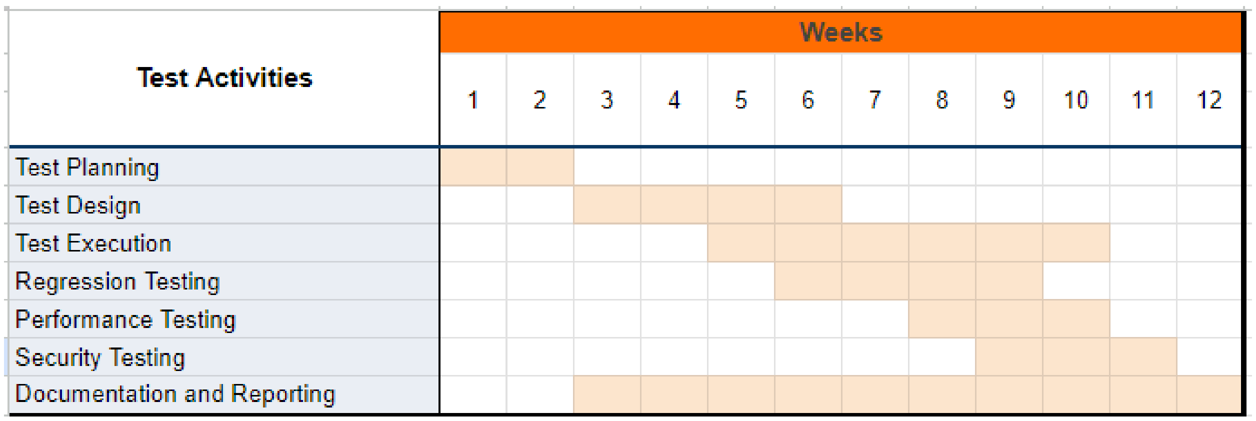
Senior QA Engineers

- Test Case Design: Create detailed test cases based on the test plans and requirements, covering various scenarios.
- Test Execution: Execute test cases, report defects, and verify defect resolutions.
- Test Automation: Develop and maintain test automation scripts where applicable to increase testing efficiency.
- Test Data Management: Ensure that appropriate test data is available and utilized for test cases.

QA Engineers

- Test Execution: Execute test cases according to the test plans, report defects, and verify defect resolutions.
- Test Data Management: Assist in preparing and managing test data required for test cases.
- Documentation: Maintain clear and accurate test documentation, including test execution records.
- Regression Testing: Participate in regression testing to identify and verify issues resulting from new changes.

6. Test Schedule



Test Activities	Weeks											
	1	2	3	4	5	6	7	8	9	10	11	12
Test Planning	▨	▨										
Test Design			▨	▨	▨	▨						
Test Execution					▨	▨	▨	▨	▨	▨		
Regression Testing						▨	▨	▨	▨			
Performance Testing								▨	▨	▨		
Security Testing									▨	▨	▨	
Documentation and Reporting			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

7. Test Reporting

7.1. Quality Matrices

- Defect Density: Measure the number of defects per unit of code.
- Test Coverage: Assess the extent of testing (e.g., code, requirements, functions).
- Pass/Fail Rate: Track the percentage of test cases passing and failing.
- Test Execution Time: Monitor the time taken to execute test cases.
- Defect Severity Distribution: Categorize defects by severity.
- User Satisfaction: Gather qualitative feedback from end-users during UAT.
- Test Automation Coverage: Measure the percentage of automated test cases.

8. Test Environment Requirements

Hardware

A distinct testing environment mirroring the production setup, including server specifications, RAM, CPU, and containers, should be established. This ensures accurate performance testing results, aligning with the application's real-world usage. Additionally, the team should possess various devices like smartphones and tablets with different screen sizes to assess actiTIME's interface. This diverse testing approach ensures the application's responsiveness and usability on mobile devices, enhancing the user experience for individuals accessing actiTIME while on the move.

Software

Certain features or functions within actiTIME may behave differently across various operating systems due to differences in file handling, system notifications, and keyboard shortcuts. To pinpoint and resolve these OS-specific issues, it's essential to conduct testing on multiple platforms. Therefore, QA team members must have access to separate virtual or physical machines running different Operating Systems like Windows, macOS, Linux, Android, and iOS. This approach allows thorough testing of the application's behavior in each environment.

Additionally, for testing purposes, a distinct database should be set up and populated with reasonably realistic data, enabling the testing of various scenarios. Web pages can be interpreted differently based on the browser being used. To ensure consistent functionality and appearance across different browsers, testers should have a range of web browsers installed, including Chrome, Firefox, Safari, and Edge. Furthermore, it's crucial to test the application compatibility with different versions of each browser type.

9. Dependencies and Assumptions

Dependencies:

- Test Data: Testing relies on the availability of accurate and relevant HR data.
- Resource Availability: Timely access to testing resources is crucial for staying on schedule.
- Development Milestones: Testing depends on the completion of development phases.

Assumptions:

- Timely Environment Setup: The test environment is set up promptly.
- Stakeholder Availability: End-users and experts are available for testing tasks.
- Resource Allocation: Adequate resources are allocated as planned.
- Assuming the system development, code merging, and unit testing have been completed.
- Demo data provided by the actiTIME application will be considered test data.
- A period of 12 weeks is allocated for testing.