Quality Assurance Plan

HR Management Application

4th November 2023

19020554 - Thilina Vishwanath Peduruhewa

1 Introduction

1.1 Purpose

This quality assurance test plan's objective is to verify the HR Management application's dependable and secure operation, with a particular emphasis on the HR staff's authorization and authentication procedures. In order to protect sensitive HR data and guarantee that HR users can quickly access employee profiles, check leave and attendance records, and approve or reject timesheets, this plan attempts to identify software risks and provide mitigation solutions. Our goal is to comply with system standards and provide a secure and efficient user experience by means of methodical testing and risk reduction.

1.2 Project Overview

The project centers around the comprehensive evaluation of the "actiTIME" HR Management system. ActiTIME is a versatile software solution designed to assist Human Resources departments in efficiently managing employee profiles, leave and attendance reports, and timesheet approvals. This testing initiative aims to ensure the system's secure and seamless operation, with a specific focus on validating user authentication, authorization, and the functionality of all features and functions. By thoroughly examining and enhancing ActiTIME, we strive to deliver an effective HR management solution that enhances operational efficiency and data security for our organization.

2 Scope

2.1 In-Scope

The scope of the testing effort encompasses a comprehensive examination of all functions and features within the HR Management application. Specifically, this includes the evaluation of user authentication and authorization processes to ensure that only authorized HR personnel can access the system securely. Within the application, we will scrutinize the ability to view and manage employee profiles, review leave and attendance reports, and approve or reject timesheets, assessing their functionality, accuracy, and security.

Additionally, we will assess the robustness of the data access controls and user privileges, ensuring that sensitive HR information remains confidential and secure. All aspects of the application, from login to data access and transaction processing, fall within the testing scope. This holistic approach aims to guarantee that every feature and function operates seamlessly, contributing to an effective HR management solution that meets the organization's requirements and maintains data integrity and security.

2.2 Out-of-Scope

The following features and combinations of features will be considered out of scope for this testing initiative:

Third-Party Integrations: Testing of integrations with third-party systems or services is out of scope. These integrations, if any, are subject to separate testing and may require specialized testing resources and environments.

Performance Testing: Performance testing, load testing, and stress testing are excluded from this scope. These types of tests require dedicated resources and distinct testing strategies to assess system scalability and performance under various conditions.

User Experience (UX) Testing: In-depth UX testing, including user interface (UI) design and usability assessments, falls outside the current scope. While usability is essential, this scope focuses primarily on functionality and security testing.

Localization Testing: Testing the system's compatibility with different languages, currencies, or regional settings is not included. This requires specialized testing resources and may vary based on organizational needs.

3 Testing Strategy

3.1 Product/Application/Solution Risks

<List the product risks identified with mitigation.>

Risks	Criticality	Mitigation Strategy
Data Breach Risk	High	Implement robust data encryption and access controls to safeguard sensitive HR information. Regularly perform security audits, penetration testing, and security patches to mitigate vulnerabilities. Conduct employee training on data security best practices.
System Downtime Risk	High	Establish redundancy and failover mechanisms to minimize the impact of system failures. Conduct load

		testing to identify performance bottlenecks and scale resources accordingly. Develop a disaster recovery plan for rapid system restoration in case of downtime.
User Authentication Failure	Moderate	Enhance user authentication with multi-factor authentication (MFA) and account lockout policies to prevent unauthorized access. Implement user account recovery procedures. Regularly audit and monitor login activities for suspicious behavior.
Functional Defects Risk	Moderate	Implement a comprehensive testing strategy, including functional, regression, and user acceptance testing, to identify and resolve software defects. Conduct code reviews and automated testing to ensure code quality. Maintain a well-documented issue tracking and resolution process.
Compliance and Legal Risks	High	Regularly audit the system to ensure compliance with relevant labor laws, data protection regulations, and industry standards. Stay updated with legal requirements and incorporate necessary changes into the system. Document compliance measures and maintain compliance-related records.

3.2 Level of Testing

Test Type	Description
Functional Testing	Functional testing ensures that each function or feature of the HR Management application performs as expected according to the specified requirements. It involves verifying user actions, data input, and output, and system interactions to validate that the application functions correctly.
Security Testing	Security testing focuses on identifying vulnerabilities and weaknesses within the HR Management application that could be exploited by malicious actors. This type of testing includes activities such as penetration testing, code scanning, and vulnerability assessments to ensure the application's robustness against security threats.
Usability Testing	Usability testing evaluates the user-friendliness of the HR Management application. It assesses how easily users can navigate the system, perform tasks, and achieve their objectives. Usability testing often involves user feedback and observation to improve the user experience.
Performance Testing	Performance testing assesses the application's responsiveness, scalability, and stability under various conditions. This type of testing includes load testing, stress testing, and benchmarking to determine how the application performs in terms of speed, reliability, and resource utilization.

3.2.1 Functional Testing

Functional testing is a fundamental and critical testing type that ensures the HR Management application's features and functions operate as intended. It involves creating test cases that mimic real-world user interactions and verifying that the application responds correctly. Testers execute these test cases to confirm that the application meets specified requirements and delivers the expected outcomes. Functional testing encompasses a range of scenarios, from basic user actions like logging in and viewing employee profiles to complex tasks like managing leave requests and approving timesheets. This testing type plays a pivotal role in ensuring the application's core functionality is reliable and consistent, thereby providing a seamless experience for HR personnel.

3.2.2 Security Testing

Security testing is essential to protect the HR Management application from potential threats and vulnerabilities. It involves a comprehensive assessment of the application's defenses against unauthorized access, data breaches, and malicious attacks. This type of testing employs various techniques, including penetration testing to simulate real-world attacks, code scanning to identify security flaws in the application's source code, and security auditing to assess compliance with industry security standards. By conducting security testing, potential vulnerabilities can be uncovered and addressed, data confidentiality and integrity can be safeguarded, and the application can be fortified against cyber threats. As data security is paramount, ensuring that sensitive HR information remains confidential is a critical aspect of this testing type.

3.2.3 Usability Testing

Usability testing focuses on the user experience within the HR Management application. Its primary goal is to evaluate the application's user-friendliness, navigation, and overall usability. Testers gather user feedback, observe user interactions, and assess how easily users can complete common tasks within the system. Usability testing helps identify pain points in the user interface, provides insights into user preferences, and highlights areas for improvement. By enhancing usability, the application becomes more user-centric, which, in turn, leads to increased user satisfaction and productivity.

3.2.4 Performance Testing

Performance testing ensures that the HR Management application can handle different workloads and scenarios effectively. It assesses the system's responsiveness, scalability, and stability under various conditions. This type of testing includes load testing to determine how the application performs under expected user loads, stress testing to assess its behavior under extreme conditions, and scalability testing to evaluate its capacity for expansion. Benchmarking establishes performance baselines and identifies areas for improvement. Performance testing is crucial as it ensures that the application remains responsive and reliable, even during peak usage periods, ultimately contributing to user satisfaction and efficient HR operations.

4. Test Approach

4.1 Test Design Approach

Our test approach for the HR Management application is multifaceted, and designed to comprehensively evaluate its quality and reliability. We will utilize a combination of structured and exploratory testing techniques to ensure a well-rounded assessment.

- Requirements-Based Testing: We will kickstart our testing process by meticulously analyzing the application's requirements. This will guide the creation of test cases that cover every specified function and feature. Requirements-based testing serves as the foundation to verify that the software aligns with the intended functionality.

- Exploratory Testing: In parallel with structured testing, we will employ exploratory testing techniques. Testers will simulate real-world user interactions, uncover unexpected issues, and assess the application's usability and user-friendliness. This approach complements structured testing and reflects actual user behavior.

- Boundary Testing: We will apply boundary testing techniques to assess the application's limits and extremes, especially concerning data input and validation. This ensures that the software remains robust, secure, and responsive within prescribed boundaries.

- Security Testing Techniques: A key focus will be on security testing, which involves various techniques such as penetration testing, vulnerability scanning, code review, and threat modeling. These methods aim to identify and mitigate potential security threats and vulnerabilities, safeguarding sensitive HR data.

- Load and Performance Testing: Load and performance testing will be a critical component of our approach. By creating scenarios that mimic expected user loads, spikes in traffic, and stress conditions, we will assess the application's responsiveness and scalability. This ensures it remains dependable even under challenging conditions.

- Usability Testing: We will engage actual users and stakeholders in usability testing. Their feedback on the application's user interface, navigation, and overall user experience will guide improvements, making the application more intuitive and user-centric.

The application of these diverse test design techniques will be tailored to the specific requirements of each testing phase. By using these methods strategically throughout the testing process, we aim to provide a comprehensive evaluation of the HR Management application's functionality, security, performance, and user satisfaction. This approach ensures that the application meets specified requirements while maintaining data security and delivering an optimal user experience.

4.2 Execution Strategy

4.3.1 Entry Criteria

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		

3.2.2 Exit criteria

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 cancelled 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

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	3
4	QA Engineers	5

5.2 Roles and Responsibilities

QA Manager

The QA Manager is responsible for overseeing the entire quality assurance process, ensuring that the testing activities align with project goals and quality standards. They serve as the link between the QA team, project stakeholders, and upper management.

- Develop and implement the overall quality assurance strategy.
- Collaborate with project managers and stakeholders to define testing objectives and priorities.
- Allocate resources and budget for testing activities.
- Monitor and report on the progress and quality of testing efforts.
- Identify and mitigate risks that could impact the testing process.
- Establish and enforce quality assurance processes, standards, and best practices.
- Provide leadership and mentorship to QA Leads, Senior QA Engineers, and QA Engineers.
- Ensure that the testing team is well-equipped and trained for their roles.
- Act as a liaison between the QA team and other project teams, fostering effective communication and collaboration.

QA Leads

QA Leads serve as team leaders, responsible for coordinating testing activities, managing resources, and ensuring the quality of testing processes.

- Define the testing strategy and approach for specific projects or test phases.
- Assign tasks and responsibilities to Senior QA Engineers and QA Engineers.
- Coordinate and monitor test execution and defect tracking.
- Collaborate with the QA Manager to ensure alignment with the overall QA strategy.
- Provide guidance and support to the testing team, resolving issues and promoting knowledge sharing.
- Review test cases, test plans, and test reports.
- Report testing progress and results to the QA Manager and project stakeholders.
- Act as a point of contact for technical issues and escalations within the QA team.

Senior QA Engineers

Senior QA Engineers are experienced professionals responsible for designing and executing test cases, ensuring the quality of test deliverables, and mentoring junior QA Engineers.

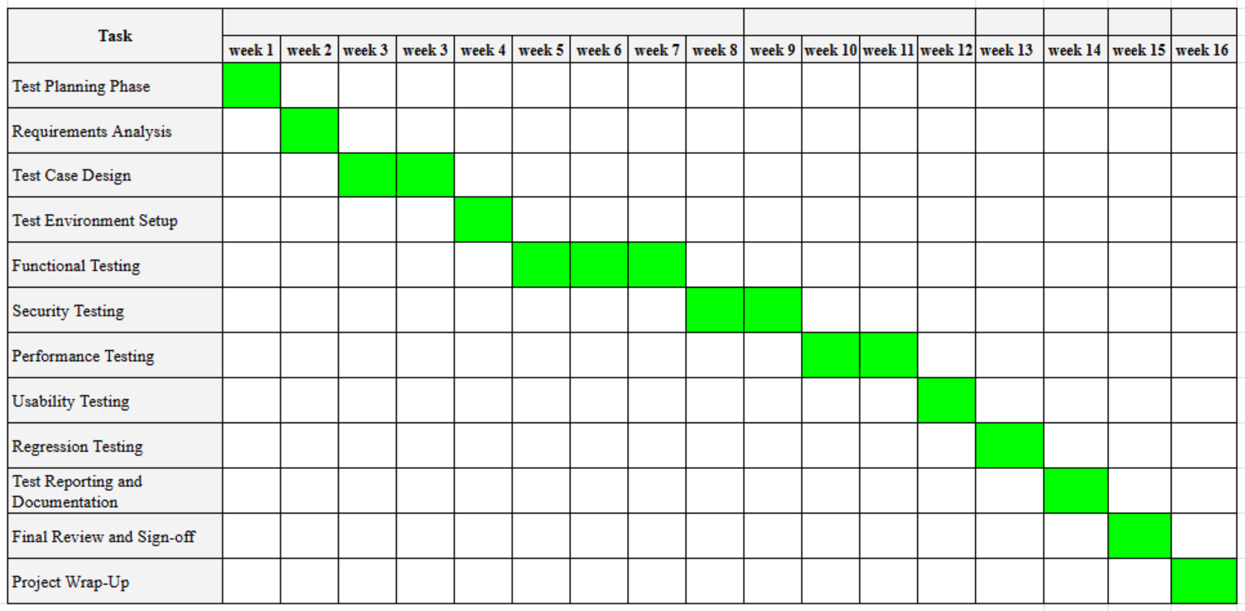
- Develop and execute test plans, test cases, and test scripts.
- Identify, record, and track defects and issues.
- Perform exploratory testing to uncover defects that may not be covered by existing test cases.
- Provide technical expertise in test automation, performance testing, or other specialized testing areas.
- Collaborate with QA Leads to understand project requirements and testing objectives.
- Mentor and assist QA Engineers in their testing activities.
- Contribute to the improvement of testing processes and best practices.
- Ensure that testing activities align with project timelines and quality standards.

QA Engineers

QA Engineers are responsible for executing test cases, performing various testing activities, and documenting results.

- Execute test cases and test scripts as per defined test plans.
- Record test results, defects, and issues.
- Participate in test case design and review.
- Collaborate with Senior QA Engineers and QA Leads to clarify requirements and resolve testing issues.
- Perform routine regression testing and report any deviations.
- Continuously update and maintain test documentation.
- Assist in test data preparation and test environment setup.
- Communicate effectively with the QA team to ensure coordinated testing efforts.
- Adhere to established testing processes and quality standards.

6. Test Schedule



Task	week 1	week 2	week 3	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14	week 15	week 16
Test Planning Phase	🟩																
Requirements Analysis		🟩															
Test Case Design			🟩	🟩													
Test Environment Setup					🟩												
Functional Testing						🟩	🟩	🟩									
Security Testing									🟩	🟩							
Performance Testing											🟩	🟩					
Usability Testing													🟩				
Regression Testing														🟩			
Test Reporting and Documentation															🟩		
Final Review and Sign-off																🟩	
Project Wrap-Up																	🟩

7. Test Reporting

7.1.Test Reporting Approach

<Add test reporting details here >

#	Report Name	Owner	Audience	Frequency
1	Test Summary Report	QA Manager	Project Stakeholders, Project Managers, Development Team	Weekly during the testing phase, and a final report at the end of testing
2	Defect Report	QA Leads	Development Team, Project Managers, QA Manager	Daily during active testing, followed by weekly summaries
3	Performance Testing Report	Senior QA Engineer	Project Stakeholders, Project Managers, Development Team	After each performance testing iteration, typically once every two weeks
4	Security Testing Report	Senior QA Engineer	Project Stakeholders, Project Managers, Security Officers	After each security testing iteration, typically once every two weeks

7.2. Quality Matrices

Test Coverage Matrix:

Measurement: Percentage of test cases executed out of the total test cases.

Purpose: Indicates how much of the application's functionality has been tested. It helps identify untested areas.

Defect Density:

Measurement: Number of defects identified per test case, test suite, or specific testing phase.

Purpose: Provides insights into the quality of the application by identifying areas with the highest defect rates.

Pass/Fail Rate:

Measurement: Percentage of test cases passed versus failed.

Purpose: Assesses the stability and readiness of the application for release.

Test Execution Progress:

Measurement: Number of test cases executed per day or per testing cycle.

Purpose: Tracks the pace of testing and helps ensure that testing aligns with project timelines.

Defect Aging Matrix:

Measurement: Age of unresolved defects (time since defect discovery).

Purpose: Assesses the efficiency of defect resolution and highlights any long-standing issues.

Test Case Execution Time:

Measurement: The time taken to execute individual test cases or test suites.

Purpose: Helps in optimizing testing efforts by identifying time-consuming test cases.

Test Environment Stability:

Measurement: Frequency of test environment-related issues and downtime.

Purpose: Ensures the reliability and availability of the test environment for testing activities.

Usability Feedback:

Measurement: Qualitative feedback and ratings from usability testing.

Purpose: Provides insights into user-friendliness and user satisfaction with the application.

Performance Metrics:

Measurement: Response times, throughput, and resource utilization during performance testing.

Purpose: Evaluate the application's performance characteristics under different loads and conditions.

8. Test Environment Requirements

Hardware and Software:

Adequate hardware and software resources for application hosting.
Compatible operating systems, web servers, and databases.
Browser compatibility for web interface testing.

Test Data:
Realistic and representative test data.
Data generation and privacy tools.

Configuration and Network:
Consistent environment configurations.
Secure network connections.
VPN or remote access.

Security Measures:
Access controls and encryption.
Regular security assessments.

Test Tools:
Functional, security, performance, and usability testing tools.
Test automation frameworks.

Scalability and Load Testing:
Scalability resources.
Load generators and simulation tools.

Usability Testing Resources:
Usability testing facilities or software.
Recording and observation tools.

Backup and Recovery:
Backup and recovery mechanisms.
Data anonymization processes.
Management and Documentation:
Environment version control.
Documentation and training resources.

9. Dependencies and Assumptions

Dependencies

Test Item Availability:
Dependency: Availability of the application build or version to be tested.
Impact: Testing cannot commence without access to the application under test.

Testing Resource Availability:
Dependency: Availability of testing resources, including hardware, software, and tools.
Impact: Lack of necessary resources can hinder testing activities and delay the testing schedule.

Development Progress:
Dependency: Completion of development activities.
Impact: Testing cannot proceed until development reaches a state where testing can effectively validate the application's functionality.

Assumptions

Timely Test Item Delivery:
Assumption: The development team will provide timely and stable builds or versions for testing.
Rationale: Timely delivery of test items is essential for adherence to the testing schedule.

Availability of Testing Resources:
Assumption: Adequate testing resources, including hardware, software, and testing tools, will be accessible.
Rationale: The availability of resources is necessary for the efficient execution of testing activities.