Quality Assurance Plan

Actitime

<4th September 2023>

1 Introduction

1.1 Purpose

This Quality Assurance Plan (QAP) outlines the testing strategy and overall approach that will drive the validation of the HR Management application hosted at https://demo.actitime.com/. The QAP is designed to ensure that the application meets its functional requirements reliably and efficiently, providing HR personnel with the ability to manage employee profiles, review leave and attendance reports, and approve or reject timesheets.

1.2 Project Overview

The project involves testing the HR Management application to verify that its features function correctly and that it provides an intuitive and efficient experience for HR staff. The application should securely handle sensitive employee data and allow HR managers to perform their tasks with accuracy and ease. The core functions include secure authentication, employee profile management, leave and attendance tracking, and timesheet approvals.

2 Scope

2.1 In-Scope

Testing will encompass the following functionalities and features of the HR Management application:

1. Verification of secure login with valid credentials.
2. Confirmation of access control: only authorized HR personnel can view and manage employee profiles.
3. Validation of the employee profile section for completeness and editing capabilities.
4. Evaluation of leave and attendance reporting, including filtering, sorting, and exporting capabilities.
5. Assessment of the timesheet submission process and the subsequent approval/rejection functionality.

These functionalities are essential for the HR department to effectively manage employee data and are, therefore, the focus of our testing efforts.

2.2 Out-of-Scope

The following features and areas will not be tested within the scope of this project:

1. Performance testing under high load is out of scope due to the limitation of the testing environment which does not replicate the production scale.
2. Penetration testing and deep security analysis are not covered in this phase as they require specialized security testing tools and expertise.
3. Localization and internationalization testing are not included, as the current requirement is for the application to support English language only.
4. Testing of third-party integrations (e.g., payroll systems, external HR databases) is excluded as these integrations are not yet implemented in the demo application.

The decision to exclude these areas from testing is based on current priority and resource allocation, as well as the scope of the demo application which may not fully replicate all production features.

3 Testing Strategy

3.1 Product/Application/Solution Risks

Risks	Criticality	Mitigation Strategy
Unauthorized access to sensitive employee data	High	Implement strong authentication and authorization checks. Regularly audit security systems and update them. Use multi-factor authentication for added security.
Timesheet approval/rejection process errors	Medium	Test the workflow extensively with different scenarios. Implement checks to verify the status of timesheets before and after actions are performed.
Miscommunication of requirements	Medium	Ensure clear and regular communication between the

		development team and stakeholders. Use requirement management tools to track changes and updates.
Inadequate test coverage	Medium	Utilize code analysis tools to assess test coverage. Increase coverage through additional test cases, exploring gray box testing approaches to enhance test scenarios.
Software compatibility issues	Low	Validate the application against different web browsers and devices it is expected to be used on. Keep the software updated in terms of compatibility.

3.2 Level of Testing

Test Type	Description

Unit Testing	Tests individual components or pieces of code for the application to ensure that each function is correct.
Integration Testing	Checks the interaction between integrated units/modules to detect interface defects.
System Testing	A complete, integrated system is tested to evaluate the system's compliance with its specified requirements.
User Acceptance Testing (UAT)	Conducted to ensure the system meets the business needs. It is usually performed by the end-users to validate end-to-end business flow.
Security Testing	Involves the testing of the application for its ability to protect data and maintain functionality as intended from potential threats and vulnerabilities.
Performance Testing	Tests the application's performance under specific conditions, particularly its speed, responsiveness, and stability under a heavy load.
Compatibility Testing	Ensures the application performs as expected across different browsers, devices, and operating systems.
Regression Testing	Performed after changes (enhancements or defect fixes) to the system to ensure

	that the unchanged areas of the application have not been affected.
Exploratory Testing	Unscripted testing to explore the application's functionality and look for defects in areas that structured testing might not cover.
Load Testing	Tests the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
Stress Testing	Involves testing an application under extreme workloads to see how it handles high traffic or data processing. The goal is to identify the breaking point of an application.
Usability Testing	Evaluating the application by testing it with real users. It focuses on the application's ease of use, flexibility, and the ability to correct mistakes.

3.2.1 Unit Testing

Unit Testing is typically a white box testing method and involves testing individual components or units of the HR Management application's codebase. It is usually performed by developers as they work on the code to ensure that each unit performs as designed:

- Functionality Checks: Each function or method is tested to verify that it correctly performs its intended operation.
- Boundary Conditions: Test cases that check how the unit behaves at the edge of its operational parameters (e.g., handling of maximum, minimum, or unexpected input values).
- Error Path Testing: Ensures that error conditions within the unit are handled properly.
- Mock Objects: Use mock objects and data to simulate parts of the system that the unit interacts with but are not part of the unit itself.

3.2.2 Intergration Testing

Integration Testing is the phase in software testing in which individual software modules are combined and tested as a group. For the HR Management application, integration testing will ensure that different components of the application work together as expected:

- Module Integration: Testing the interaction between different modules or services, such as the login process, profile management, and report generation to ensure data flows correctly between them.
- Interface Testing: Verifying that the interfaces between different software modules are working properly, including APIs, web services, and other protocols.
- End-to-End Testing: Simulate real user scenarios to ensure the system works from start to finish. For example, the process of an HR user logging in, reviewing, and approving timesheets should be seamless.
- Regression Integration: Ensuring that newly integrated modules do not disrupt the existing integrated components' operations.

- Data Integration: Ensuring that data integrity is maintained when data is passed between various modules, such as transferring leave balances from the leave management module to the report generation module for end-of-year processing.

3.3.3 System Testing

System Testing is a level of software testing where a complete, integrated system/software is tested. The purpose is to evaluate the system's compliance with the specified requirements. In the context of the HR Management application, system testing will validate the fully integrated application and its behavior against the overall requirements.

- System Functionality: Test the complete and integrated software to verify that it meets all requirements, including functional, performance, and security aspects.
- Performance Testing: Evaluate the system's performance under expected and peak load conditions to ensure it meets performance criteria including responsiveness, stability, and scalability.

3.3.4 User Acceptance Testing

User Acceptance Testing, often the last step before the application is released to the market or handed off to the client, involves verifying that the HR Management application meets the business requirements and is ready for real-world use. This testing is conducted with the actual end-users to ensure the system meets their needs and is user-friendly.

- Business Processes: UAT testers, who are usually actual users or clients, verify that the application supports all the necessary HR business processes effectively. This includes logging in, managing employee profiles, reviewing and approving leave and attendance, and processing timesheets.
- Real-World Usage: Simulate real-life scenarios where the end-users employ the system under normal working conditions to ensure the application behaves as expected.

- Acceptance Criteria: Testing is based on the agreed-upon criteria and requirements outlined by stakeholders, which the application must meet to be accepted.

3.3.5 Security Testing

Security Testing will be critical in verifying that the HR Management application is robust against unauthorized access and data breaches:

- Authentication: Test the system's ability to secure user login and handle password protection mechanisms correctly.
- Authorization: Ensure that users can only access features and data that they have permissions for and cannot exploit any access control issues.
- Data Encryption: Verify that sensitive data, such as personal employee details, is encrypted in transit and at rest.
- Vulnerability Scanning: Use automated tools to scan for common vulnerabilities and assess the application against known security risks.

3.3.6 Performance Testing

This testing ensures that the application performs well under various conditions and does not suffer from latency or crashes:

- Load Testing: Assess the application's performance under expected user loads.
- Stress Testing: Determine the application's breaking point by subjecting it to extreme workloads.
- Endurance Testing: Check if the application can handle the expected load over an extended period.

3.3.7 Compatibility Testing

Compatibility Testing will confirm that the application operates across different user environments:

- Browser Compatibility: Ensure the application works on various web browsers like Chrome, Firefox, Edge, and Safari.
- Operating System Compatibility: Test the application on different operating systems such as Windows, macOS, and Linux.
- Mobile Responsiveness: Verify that the application is responsive and functional on various mobile devices and screen sizes.

3.3.8 Regression Testing

After any update or bug fix, Regression Testing is essential to ensure that new code has not disrupted existing features:

- Automated Regression Suite: Use automated tests to quickly validate core functionalities.
- Full Regression: In the case of major updates, a full regression may be warranted to cover all aspects of the application.
- Smoke Testing: After each release, perform basic checks to confirm that the system's most crucial functions are still operational.

3.3.9 Exploratory Testing

Exploratory Testing is less structured and allows testers to explore the application's capabilities:

- Ad-hoc Testing: Testers randomly test the application without specific test cases to find defects that may not be caught in structured testing.
- Session-based Testing: Testers document their findings in time-boxed sessions, allowing for a structured approach to unscripted testing.

3.3.10 Load Testing

Load Testing specifically examines performance under various loads:

- Concurrent Users: Simulate multiple users accessing the system simultaneously to see how it handles concurrent operations.
- Data Volume: Test the system's handling of large amounts of data, which is common in HR environments with extensive employee records.

3.3.11 Stress Testing

Stress Testing determines the robustness of the application under extreme conditions:

- Limits: Gradually increase the load until the system breaks to understand its upper limits.
- Recovery: Evaluate the system's ability to recover from crashes or failures due to overload.

3.3.12 Usability Testing

Usability Testing is an essential part of testing the HR Management application, focusing on the user's ease of using the application, the flexibility of the application to handle controls, and the application's ability to meet its objectives. In Usability Testing, the application is evaluated by testing it with real users who are part of the intended target audience or user base.

- Ease of Navigation: Determine how easily users can navigate through the application to complete typical HR tasks, such as finding an employee profile or submitting a time-off request.
- Clarity of Interface: Assess if the application's user interface elements, such as buttons, forms, and menus, are clear and understandable to the end-users.
- Task Completion: Evaluate the ability of users to complete tasks efficiently and to their satisfaction without assistance.

4. Test Approach

4.1 Test Design Approach

Our test design strategy for validating the HR Management application will utilize a combination of black box, white box, and gray box testing methodologies, along with exploratory testing, to ensure comprehensive coverage of the application's functionality, performance, and security. Each technique will be applied as follows:

Black Box Testing:

We will employ equivalence partitioning and boundary value analysis to validate all functional requirements of the application, such as logging in, viewing employee profiles, reviewing leaves, and attendance reports, and the approval/rejection of timesheets.
Decision table testing will be used to understand the software's behavior with different combinations of inputs for complex business rules related to timesheet approvals.
State transition testing will help us to verify the application's behavior in different states, for example, an employee's timesheet status changing from 'submitted' to 'approved' or 'rejected'.

White Box Testing:

We will perform code coverage analysis to ensure that all the possible paths within the code have been tested. This will include statement coverage, branch coverage, and path coverage techniques.

Loop testing will be particularly focused on the code where loops are used for generating reports and managing data entries to ensure that the loops perform correctly for all iterations.

Control flow testing will be applied to assess the logical flow of the application and ensure that the sequence of processing follows the intended paths.

Gray Box Testing:

For gray box testing, we will combine our knowledge of the internal data structures with black-box testing techniques to design test cases. This will be especially useful for testing the application's interaction with the database and the proper handling of transactions.

We will focus on integration points where modules interact with each other to ensure that partial knowledge of internal workings is used effectively to test these key areas.

Exploratory Testing:

We will conduct time-boxed exploratory testing sessions, particularly in areas where the specification documents may be incomplete or subject to interpretation. Testers will explore the application, designing and executing tests on the fly based on their understanding and findings.

Exploratory testing will also be employed for usability testing, as it allows testers to simulate the behavior of real users and uncover issues that might not be apparent in scripted testing.

Experience-Based Techniques:

Experienced testers will use error guessing to identify potential problem areas where the application may not handle certain inputs or conditions well, based on their experience with similar applications.
We will involve domain experts in the testing process to utilize their in-depth knowledge for creating realistic and critical test scenarios, especially for business workflow tests.
The application of these test design techniques will be documented in detailed test cases and test scripts that outline the expected inputs, execution conditions, and expected outcomes. We will also use automated test case generation tools where possible to increase the efficiency of the test case creation process. Regular reviews and walkthroughs of the test cases will be conducted to ensure their validity and effectiveness.

Through this multi-faceted test design approach, we aim to cover all aspects of the HR Management application, ensuring a robust and user-friendly product that meets the business and technical requirements set forth by stakeholders.

4.2 Execution Strategy

4.3.1 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	✓	The test environment closely mimics the production setup for accurate results.
Test data is available	✓	Test data includes a variety of scenarios, including edge cases for comprehensive testing.
Code has been merged successfully	✓	Code integration has been completed with no merge conflicts.
Development has completed unit testing	✓	Unit tests have been passed, ensuring that the basic building blocks of the application are stable.
Test cases and scripts are completed, reviewed and approved by the Project Team	✓	All test cases and scripts have been created based on the test strategy and reviewed for quality and completeness.

3.2.2 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	✓	All test scripts have been run to ensure complete coverage of the application's features.
90% pass rate of Test Scripts	✓	This high pass rate indicates a stable application; issues found are documented for fixing.
No open Critical and High severity defects	✓	Critical and High severity defects have been addressed to ensure they don't impact production.
All remaining defects are either cancelled or documented as Change Requests for a future release	✓	Non-critical defects are tracked for future prioritization, ensuring the application's continuous improvement.
All expected and actual results are captured and documented with the test script	✓	Ensures traceability of tests and supports quality control measures.
All test metrics collected based on reports from daily and Weekly Status reports	✓	Facilitates ongoing monitoring and management of the testing process and outcomes.
All defects logged in -Defect Tracker/Spreadsheet	✓	Defect logging is crucial for traceability, accountability, and future reference.

Test environment cleanup completed and a new back up of the environment	✓	Critical for maintaining a clean state for future testing and rollback scenarios.

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.*

Process Overview:

Execution of Test Scripts:

Testers must follow the testing plan to execute all designated test scripts in the specified cycles.

Defect Tracking:

All identified defects are to be tracked rigorously, using either a dedicated Defect Tracker tool or a structured Spreadsheet.

Tester's Responsibility:

Testers are responsible for the end-to-end defect management process, which entails:

- Opening Defects: Logging new defects as they are discovered during testing.
- Retesting: Ensuring each defect is retested post-resolution to verify the effectiveness of the fix.
- Closing Defects: Officially closing defects in the tracking system once they have been verified as resolved.

Defect Lifecycle Stages:

- New: Detected defects are logged with all pertinent details.
- Assigned: The defect is assigned to the respective developer or team for remediation.
- Open: The assigned party acknowledges the defect and commences work on the resolution.
- Fixed/Resolved: Post-fix, the developer updates the status, indicating that the defect should be retested.
- Pending Retest: The defect awaits confirmation testing by the testing team.
- Retest: The tester conducts targeted testing to confirm the defect has been resolved.
- Verified: Upon successful retest, the tester marks the defect as verified.
- Closed: The defect's status is updated to closed once verification is complete.

Special Cases:

- Reopened: If a defect persists post-fix, it is reopened for further investigation.
- Duplicate, Rejected, Deferred, Could Not Reproduce: Special statuses for edge cases during defect management.

Expectations:

Adherence to the testing plan is imperative, and testers are to ensure thoroughness in defect identification, retesting, and closure to uphold the quality standards of the HR software.

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 - 6

Roles and Responsibilities:

- QA Manager: Oversee the entire testing process, ensure compliance with standards, manage team and resources, and act as a liaison with other project stakeholders.

- QA Leads: Coordinate testing activities, mentor QA Engineers, develop test strategies, and ensure the quality execution of test cases.
- Senior QA Engineers: Lead complex test design, execution, and troubleshooting while providing technical guidance.
- QA Engineers: Execute test cases, report defects, and retest resolved issues.

5.2 Roles and Responsibilities

QA Manager:

- Strategic Oversight: Defines the test strategy and ensures alignment with project objectives.
- Resource Management: Allocates and manages testing resources, including personnel and testing environments.
- Stakeholder Communication: Acts as the primary point of contact for test-related communications with project stakeholders.
- Risk Management: Assesses testing risks and implements mitigation plans.
- Quality Assurance: Oversees the maintenance of quality standards throughout the testing process.

QA Leads:

- Test Planning: Develops detailed test plans, outlining the scope, approach, resources, and schedule of intended test activities.
- Team Coordination: Coordinates among testing team members and ensures adherence to the test schedule and strategy.
- Mentoring: Provides guidance and support to QA Engineers and helps in resolving complex issues.
- Reporting: Monitors test progress and reports the status to the QA Manager and other stakeholders.

Senior QA Engineers:

- Test Design: Designs complex test cases and scenarios that align with user requirements and risk assessments.
- Technical Leadership: Provides technical expertise in test automation, performance testing, and other specialized areas.
- Problem Solving: Analyzes defects and identifies systemic root causes to improve the overall quality.
- Process Improvement: Suggests improvements to the testing process and helps implement best practices.

QA Engineers:

- Test Execution: Executes test cases manually or via automated tools and records the outcomes.
- Defect Reporting: Logs defects in the tracking system with detailed, reproducible steps and severity ratings.
- Verification: Retests fixed defects to confirm resolution and performs regression testing to ensure no new issues are introduced.
- Documentation: Maintains clear and accurate documentation for test cases, test results, and defect findings.

6. Test Schedule

Week 1: Test Planning

Day 1-2: Review of HR software requirements with a focus on login and security features.

Day 3: Test strategy finalization and resource allocation.

Day 4-5: Creation of test cases for the login module.

Week 2: Test Environment Setup and Test Design

Day 1-3: Setup of the test environment and verification of access controls for testers.

Day 4-5: Design of detailed test cases for employee record and leave management modules.

Week 3: Test Execution - Round 1

Day 1-3: Execution of login functionality test cases - includes positive, negative, and edge cases.

Day 4-5: Execution of test cases for basic employee record management functionalities.

Week 4: Continued Test Execution and Defect Management

Day 1-3: Continue testing for employee record management and initiate leave management testing.

Day 4: Review and address defects found in the login module.

Day 5: Retesting fixed defects and regression testing for the login functionality.

Week 5: Test Execution - Round 2 and Non-Functional Testing

Day 1-2: Complete testing for leave management module.

Day 3: Performance testing for login and employee record management under load.

Day 4: Security testing focusing on login module.

Day 5: Accessibility and usability testing for the overall HR software with emphasis on login ease-of-use.

Week 6: Final Testing and Closure

Day 1-2: Final round of regression testing for all modules.

Day 3: Validation of the entire application and final defect fixes.

Day 4: Preparation of final test report and metrics.

Day 5: Test closure activities and archiving of test artifacts.

7. Test Reporting

7.1.Test Reporting Approach

#	Report Name	Owner	Audience	Frequency
1	Login Module Test Report	Senior QA Engineer	QA Manager, Development Team	Bi-Weekly
2	Overall Test Progress Report	QA Manager	Project Stakeholders	End of each phase

Details:

Login Module Test Report:

Owner: Senior QA Engineer responsible for the login module.

Audience: QA Manager for oversight, Development Team for defect resolution.

Frequency: Bi-weekly to align with sprints, ensuring timely feedback for the iterative development and testing of the login functionality.

Overall Test Progress Report:

Owner: QA Manager oversees the overall testing progress.

Audience: Project Stakeholders including Project Managers, Product Owners, and possibly Client Representatives.

Frequency: At the end of each testing phase to summarize findings, overall progress, and key metrics.

7.2. Quality Matrices

Our approach to evaluating HR software quality involves distinct test matrices. A critical metric is the Requirement Traceability Matrix, linking test cases back to specific project requirements. This ensures comprehensive coverage and alignment with project objectives. Additionally, the Performance Matrix will gauge the system's responsiveness under varying loads, providing insights into its scalability. Lastly, the Customer Satisfaction Matrix will capture user feedback, enabling us to gauge the software's real-world impact and user experience effectively.

8. Test Environment Requirements

Hardware Requirements:

Server: Minimum Quad-core Processor, 16GB RAM, 500GB SSD.

Client machines: Dual-core Processor, 8GB RAM, 256GB SSD.

Network: Gigabit Ethernet, to support client-server communication.

Software Requirements:

Operating System: Windows 10 for client machines, Windows Server 2019 for the server.

Web Server: Apache or IIS compatible with the HR software.

Database: SQL Server 2019, with initial test data seeded.

Browsers: Latest versions of Chrome, Firefox, Edge, and Safari for cross-browser testing.

Test Automation Tools: Selenium WebDriver for automated UI tests.

Network Configuration:

The test environment should mimic the production network topology, including similar firewall rules and network latency.

Secure VPN access for remote testers.

Access Requirements:

Required system accesses for testing personnel to perform tests on the application, database, and server.

Access to logging and monitoring tools for test analysis.

Data Requirements:

Test data that represents the production data in terms of data volume and data patterns to ensure realistic testing scenarios.

Anonymization of sensitive data to comply with privacy regulations.

Tools and Licenses:

All necessary software licenses for the test environment must be acquired.

Version control system for managing test scripts and documentation.

Backup and Restore:

Capabilities to regularly backup and restore the test environment to ensure testing can resume quickly after any critical failure or to reset the environment for new test cycles.

Security:

Implementation of security protocols to protect the test environment from unauthorized access and to ensure the safety of sensitive test data.

Monitoring and Logging:

Tools to monitor the performance of the application and to log errors and system behavior for analysis.

9. Dependencies and Assumptions

Dependancies

Test Item Availability:

Finalized and stable build of the HR software is available for testing by the proposed start date.
All third-party integrations and services that the HR software depends on are operational and accessible.

Testing Resource Availability:

Availability of the test team members as per the roles and responsibilities outlined in the test plan.
Access to the testing environment, including all necessary hardware and software, is ready and reserved for the duration of the testing phase.

Data Dependency:

Provision of adequate test data sets, including both normal and edge case scenarios, especially for the login functionality.
Sufficient anonymized production data available for performance and load testing.

Documentation:

Availability of up-to-date requirement documentation, user stories, and acceptance criteria to ensure test cases align with expected outcomes.

Stakeholder Availability:

Key stakeholders and decision-makers are available for the critical milestones such as kick-off, review meetings, and sign-off.

Assumptions

Testing Delays:

There will be no significant delays in the delivery of the application builds for testing.
Stable Environment:

The test environment will remain stable and accessible 24/7 during the testing period unless scheduled maintenance is agreed upon in advance.

Bug Fixing:

Defects identified will be fixed and patches will be available for retesting within the agreed-upon timeframes.

Support and Communication:

There will be continuous support and clear communication from the development team and project management throughout the testing phase.

Tools and Access:

All necessary testing tools, systems, and accesses required by the testing team will be provided without delay.

External Factors:

No major external factors (such as third-party service outages) will impact the ability to test.

Change Management:

Any changes to the requirements will be communicated and agreed upon before the impact on the testing timeline or scope.