
Quality Assurance Plan

<Validating the HR Management application>

<31st October 2023>

1 Introduction

1.1 PURPOSE

This Quality Assurance Plan (QAP) outlines the methodology and testing plan for the validation of HR Management application. It outlines the main components of quality assurance, such as the goals, deliverables, scope, and general framework that will direct the testing operations.

1.2 PROJECT OVERVIEW

The application for HR Management is an essential part of the company's HR procedures. HR staff may use it as a platform to manage employee profiles, examine attendance and leave records, and accept or reject timesheets. In order to guarantee the effectiveness of HR operations and compliance with corporate rules and regulatory requirements, this program must run successfully without errors. As part of this project, the application must be fully tested to ensure that it satisfies all requirements and is prepared for deployment.

2 Scope

2.1 IN-SCOPE

In-scope defines what will be tested as part of this QA effort. For the validation of the HR Management application, the following aspects are considered as in-scope:

- Log in with a valid username and password: Verify that HR personnel can successfully log in with their credentials.
- Accessing and viewing employee profiles: Ensure that HR users can access and view employee profiles without encountering errors or issues.
- Reviewing leave and attendance reports: Verify the accuracy and reliability of leave and attendance reports.
- Approval and rejection of timesheets: Confirm that HR personnel can perform the necessary actions on timesheets.

2.2 OUT-OF-SCOPE

Out-of-scope items specify what will not be tested as part of this QA effort. The justification for excluding these elements should be provided. For the validation of the HR Management application, the following items are out of scope:

- Testing of invalid or incorrect login credentials: The focus is on testing valid logins. Testing invalid logins is considered a separate test scenario.

- Performance and load testing: Performance and load testing will not be conducted as part of this plan but may be addressed in a separate performance testing plan.
- Compatibility testing on all possible devices and browsers: The primary focus is on a specific set of browsers and devices based on the organization's standard. A separate compatibility testing plan may be developed for additional coverage.
- Testing of functionalities not related to HR management: Features unrelated to HR management, such as accounting or project management, are not within the scope of this plan.

3 Testing Strategy

3.1 PRODUCT/APPLICATION/SOLUTION RISKS

Risks	Criticality	Mitigation Strategy
Unauthorized Access	High	Implement strong authentication mechanisms to prevent unauthorized access. Regularly audit access controls and user permissions.
Data Security Breach	High	Employ encryption and secure data storage practices. Conduct security assessments, including vulnerability scanning and penetration testing.
Data Loss or Corruption	Medium	Regularly backup data and implement data validation checks. Train users on data entry best practices.
User Interface Inconsistencies	Low	Conduct usability testing to ensure a consistent and user-friendly interface. Implement design standards and guidelines.

3.2 LEVEL OF TESTING

Test Type	Description
Functional Testing	Validate that all functions/features work according to specifications.
Integration Testing	Ensure seamless integration with other systems and services.
Performance Testing	Evaluate the system's responsiveness, scalability, and resource utilization.
Regression Testing	Verify that new updates or changes do not introduce new defects.
User Acceptance Testing	Involve end-users to validate the application against real-world scenarios.

4. Test Approach

4.1 TEST DESIGN APPROACH

The test design approach for validating the HR Management application involves a comprehensive strategy that encompasses various testing techniques to ensure the application's functionality, reliability, and security. The following test design techniques will be applied:

Black Box Testing: This technique will be used to verify the functional behavior of the application without examining its internal code. Test cases will be designed to cover different usage scenarios, including valid and invalid inputs, boundary conditions, and error handling.

Usability Testing: Human-centered design principles will be applied to assess the application's user-friendliness. A group of representative end-users will evaluate the interface for intuitiveness, efficiency, and overall user satisfaction.

Data-Driven Testing: A data-driven approach will be adopted to test the application's handling of various types of data, including data validation, storage, and retrieval. Test cases will include boundary values, large data sets, and edge cases.

Integration Testing: The integration points with other systems and services will be rigorously tested to ensure seamless data flow and interoperability. Test cases will focus on data exchange, communication protocols, and error handling.



Performance Testing: Performance testing, including load testing, will be performed to assess the application's responsiveness, scalability, and resource utilization under different workloads. Performance metrics and bottlenecks will be identified and addressed.

Regression Testing: Continuous regression testing will be conducted to ensure that new updates and changes do not introduce new defects or adversely impact existing functionality. Test cases will be automated to facilitate efficient regression testing.

User Acceptance Testing (UAT): End-users will be actively involved in UAT to validate the application against real-world scenarios and ensure that it meets their specific business needs and requirements.

4.2 EXECUTION STRATEGY

4.3.1 Entry Criteria

Entry Criteria	Conditions	Comments
<i>Test environment(s) is available</i>		The necessary test environments are set up and ready for testing.
<i>Test data is available</i>		Test data required for testing scenarios is prepared and accessible.
<i>Code has been merged successfully</i>		The code from development is successfully integrated and merged.
<i>Development has completed unit testing</i>		Development has confirmed that unit testing of code is completed successfully.
<i>Test cases and scripts are completed, reviewed and approved by the Project Team</i>		All test cases and scripts are prepared, reviewed, and approved for execution.

3.2.2 Exit criteria

Exit Criteria	Conditions	Comments
---------------	------------	----------

<i>100% Test Scripts executed</i>		All planned test scripts have been executed as per the test plan.
<i>90% pass rate of Test Scripts</i>		A minimum of 90% of executed test scripts must result in a pass.
<i>No open Critical and High severity defects</i>		There should be no unresolved critical or high-severity defects.
<i>All remaining defects are either cancelled or documented as Change Requests for a future release</i>		Any remaining defects must be appropriately managed and documented for future consideration.
<i>All expected and actual results are captured and documented with the test script</i>		Ensure that all test results, both expected and actual, are properly documented and associated with corresponding test scripts.
<i>All test metrics collected based on reports from daily and Weekly Status reports</i>		Test metrics and performance indicators are regularly collected and evaluated based on daily and weekly status reports.
<i>All defects logged in -Defect Tracker/Spreadsheet</i>		Every defect identified during testing is documented in the designated defect tracking system or spreadsheet.
<i>Test environment cleanup completed and a new back up of the environment</i>		The test environment has been cleaned up, and a backup has been taken for reference and potential future testing or troubleshooting.

3.3 DEFECT MANAGEMENT

Defect management is a crucial aspect of the testing process. It ensures that any issues or anomalies identified during testing are adequately documented, tracked, and resolved. The following details the process for managing defects in the testing of the HR Management application:

Defect Tracking Tool: Defects found during testing will be tracked and managed using a dedicated Defect Tracker tool or a spreadsheet. This tool will serve as the central repository for recording, monitoring, and managing defects.

Defect Lifecycle: The management of defects will follow a defined lifecycle, which typically includes the following stages:

Defect Identification: Testers will identify and report defects, providing detailed information about the issue, its impact, and the steps to reproduce it.

Defect Prioritization: Defects will be assigned a priority based on their impact on the system. Prioritization will follow the defined severity and impact criteria.

Assignment: Defects will be assigned to the responsible team or individual for analysis and resolution.

Resolution: Developers will work on resolving the defects, making necessary code changes, and implementing fixes.

Retesting: Testers will verify the defect fixes to ensure that the issue has been resolved and does not introduce new problems.

Approval: After successful retesting, defects will be marked as approved for closure.

Closure: Closed defects will be documented, and relevant stakeholders will be notified of the resolution.

Reporting: Regular defect reports will be generated and shared with project stakeholders to provide visibility into the defect status.

Defect Categorization: Defects found during testing will be categorized based on severity and impact. The following classification will be used:

Severity	Impact
1 (Critical)	<ul style="list-style-type: none">▪ <i>Functionality is blocked and no testing can proceed</i>▪ <i>Application/program/feature is unusable in the current state</i>
2 (High)	<ul style="list-style-type: none">▪ <i>Functionality is not usable and there is no workaround but testing can proceed</i>
3 (Medium)	<ul style="list-style-type: none">▪ <i>Functionality issues but there is a workaround for achieving the desired functionality</i>
4 (Low)	<ul style="list-style-type: none">▪ <i>Unclear error message or cosmetic error which has minimum impact on product use.</i>

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	varies

5.2 ROLES AND RESPONSIBILITIES

QA Manager

The QA Manager is responsible for overall test strategy, planning, and execution.

They oversee the entire testing process and ensure it aligns with project objectives and quality standards.

The QA Manager coordinates with stakeholders, reports progress, and manages the test team.

QA Leads

QA Leads are responsible for leading and managing specific test teams or areas within the project.

They develop test strategies, create test plans, and supervise test execution.

QA Leads also mentor junior team members and ensure that testing aligns with project requirements.

Senior QA Engineers

Senior QA Engineers have a wealth of experience in testing and serve as subject matter experts.

They create and execute complex test cases, analyze results, and identify defects.

Senior QA Engineers may also assist in test planning and mentor junior team members.

QA Engineers

QA Engineers execute test cases, report defects, and verify defect fixes.

They actively participate in test case design and documentation.

QA Engineers work closely with leads to ensure comprehensive test coverage and support the testing process.

6. Test Schedule

Phase	Duration
Test Planning	1 week
Test Environment Setup	1 week
Test Case Design	2 weeks
Test Execution	4 weeks
Defect Management	Ongoing
Test Reporting	Ongoing
User Acceptance Testing	2 weeks
Test Closure	1 week
Project Handover	1 week

7. Test Reporting

7.1. TEST REPORTING APPROACH

#	Report Name	Owner	Audience	Frequency
1	TEST PROGRESS REPORT	QA Manager	Project Team, Management	Weekly
2	TEST PROGRESS REPORT	QA Lead(s), QA Manager	Development Team, Project Team, Management	As needed

7.2. QUALITY MATRICES

Defect Density: This matrix measures the number of defects identified per unit of code (e.g., lines of code, function points). It helps assess the quality of the code and its overall stability. A lower defect density is generally indicative of higher code quality.

Test Coverage: Test coverage matrices assess the extent to which the application's functionality and code are tested. This includes:

Code Coverage: Measures the percentage of code executed by test cases.

Requirements Coverage: Evaluates how well test cases cover the specified requirements.

Functional Coverage: Assesses the coverage of various functional areas within the application.

Defect Age: This matrix tracks the amount of time defects remain open, helping to identify bottlenecks in the defect resolution process. It's measured in terms of the number of days or cycles a defect is open before closure.

Pass/Fail Ratio: This matrix calculates the ratio of passed test cases to failed test cases. It provides insights into the overall test result and application stability. A high pass/fail ratio indicates better quality.

Regression Test Effectiveness: This matrix assesses the ability of regression tests to identify new defects. It measures the number of newly discovered defects compared to the total number of regression test cases executed.

Testing Effort vs. Test Coverage: This matrix evaluates the relationship between testing effort (measured in person-hours or person-days) and the achieved test coverage. It helps assess the efficiency and effectiveness of testing.

User Satisfaction and Feedback: While not a traditional metric, capturing user feedback and satisfaction is essential for assessing the application's quality. It can include surveys, user acceptance testing results, and feedback collected from real users.

8. Test Environment Requirements

The test environment plays a critical role in ensuring the successful execution of testing activities. The following are the test environment requirements for the HR Management application:

Hardware Requirements:

Test Servers: A dedicated server or servers that replicate the production environment, including hardware specifications like CPU, RAM, and storage.

Test Workstations: Workstations for test team members to execute tests and access the application.

Networking Equipment: Routers, switches, and firewalls to simulate network conditions and configurations.

Software Requirements:

Operating Systems: Supported operating systems for server and client machines.

Web Browsers: Supported web browsers for compatibility testing.

Database Systems: Database systems for data storage and retrieval.

Test Automation Tools: If test automation is planned, the necessary automation tools and frameworks.

Virtualization Software: If virtualized test environments are used.

Data Requirements:

Test Data: Realistic and representative data sets for testing various scenarios, including user profiles, leave records, and attendance data.

Backup and Restore Mechanism: A mechanism to back up and restore the test data and environment to ensure data integrity.

Access Rights and Permissions:

Testers require appropriate access rights and permissions to configure, execute, and monitor the application under test.

Configuration Management: A version control system to manage test scripts, test data, and related artifacts.

9. Dependencies and Assumptions

Successful testing is dependent on various factors, and assumptions may need to be made. The following dependencies and assumptions are identified:

Availability of Test Items: Availability of the HR Management application in the test environment.
Availability of relevant documentation, such as requirements, design documents, and user manuals.

Testing Resource Availability: Availability of an adequately staffed test team with the necessary skills and expertise. Availability of the necessary hardware, software, and testing tools.

Data Privacy and Security: Assumption that appropriate data privacy measures are in place, and test data is anonymized or de-identified as required.

Development Progress: The testing schedule assumes that development is on track, with code available for testing according to the planned timelines.

Deadlines: Assumption that project deadlines for testing phases, including test planning, execution, and reporting, are adhered to.

Change Management: Assumption that changes to the application during testing will be managed through a defined change control process.

Stakeholder Cooperation: Dependency on cooperation and timely feedback from stakeholders, including development, project management, and end-users.

Test Environment Readiness: The test environment will be ready and configured according to requirements before test execution begins.

Test Data Availability: Assumption that the required test data will be available, with backup and restore mechanisms in place to maintain data integrity.