Quality Assurance Plan

HR Management Application

4th September 2023

1 Introduction

1.1 Purpose

Use the Quality Assurance Plan document to describe the testing strategy and overall approach that will drive the testing of the project.

1.2 Project Overview

The project involves the validation of the HR Management application, accessible via the following URL: HR Management Application. The HR Management application is a critical component for HR professionals to manage employee profiles, review leaves and attendance reports, and approve or reject timesheets. It is essential that the application allows HR personnel to log in successfully using valid credentials and perform these tasks accurately.

The HR Management application is an integral part of the organization's human resources management system, serving as the central platform for HR-related activities. It plays a crucial role in managing employee data, leave requests, attendance reports, and timesheet approvals. Ensuring its reliability and functionality is of utmost importance to the organization.

This Quality Assurance Test Plan will outline the testing processes, risks, and mitigation strategies to ensure that the HR Management application performs as expected and meets the business requirements effectively. The testing approach will be based on IEEE (Institute of Electrical and Electronics Engineers) standards, which are widely recognized and respected in the field of software quality assurance.

2 Scope

2.1 In-Scope

The testing scope for the HR Management application includes the following:

1. **User Authentication:**
    a. Verification of the ability for HR personnel to log in with a valid username and password successfully.
2. **Employee Profiles:**
    a. Testing the functionality to view and edit employee profiles.
3. **Leave and Attendance Reports:**
    a. Validating the ability to generate and review leave and attendance reports for employees.
4. **Timesheet Approval/Rejection:**
    a. Confirming the system's capability to approve or reject timesheets submitted by employees.

2.2 Out-of-Scope

The following features and combinations of features are out of scope for this testing phase:

1. **User Registration:**
    a. The process of creating new HR user accounts will not be tested as it is not a part of this specific requirement. The focus is on HR users who are already registered and attempting to log in.
2. **Employee Self-Service:**
    a. Features related to employees updating their own profiles or submitting timesheets will not be tested as it falls outside the scope of HR personnel's actions.
3. **Integration Testing:**
    a. Integration of the HR Management application with other systems or applications is out of scope. This test plan focuses solely on the functionality of the HR Management application.
4. **Performance and Load Testing:**
    a. The plan does not cover performance or load testing, which evaluates the application's response to high user loads or extended usage over time. This plan focuses on functional testing.
5. **Security Testing:**
    a. Detailed security testing, including penetration testing or vulnerability assessment, is not within the scope of this document. Security testing should be conducted separately to ensure the application's security.

3 Testing Strategy

3.1 Product/Application/Solution Risks

The following product risks have been identified along with their criticality and mitigation strategies:

Risks	Criticality	Mitigation Strategy
1. User Authentication Failures	High	Implement thorough validation checks to ensure that only valid usernames and passwords are accepted for login. Ensure appropriate error handling for authentication failures.
2. Inaccurate Employee Data Updates	Medium	Implement user confirmation prompts for critical updates to employee profiles. Perform data validation checks to prevent inaccurate data entry.
3. Inconsistent Leave and Attendance Reports	Medium	Validate report generation and data accuracy. Implement reconciliation processes to resolve discrepancies.
4. Timesheet Approval/Rejection Errors	High	Implement a clear and secure workflow for timesheet approval/rejection. Enable HR personnel to review timesheets before taking action.
5. Performance Bottlenecks	Low	While performance testing is out of scope, regular monitoring and optimization will be performed during functional testing to identify potential bottlenecks.

6. Security Vulnerabilities	Medium	Conduct a separate security testing phase to identify and mitigate security vulnerabilities.

3.2 Level of Testing

Test Type	Description
Functional Testing	Functional testing will ensure that the HR Management application's functions perform correctly according to the specified business requirements. This includes verifying user authentication, employee profile management, leave and attendance report generation, and timesheet approval/rejection.
Regression Testing	Regression testing will be performed to ensure that any changes or updates to the application do not adversely affect its existing functionality. This will help identify and address any unintended side effects of code modifications.
Non-Functional Testing	Non-functional testing includes various aspects of the application's performance, usability, and security, which are not part of traditional functional testing. It involves evaluating the application's response times, user interface, and security features.

Details of Each Testing Type:

3.2.1 Functional Testing:

Functional testing will be carried out to verify that the application's features perform as expected. Test cases will be created to assess each of the following functionalities:

- User Authentication Testing: Verify successful login with valid credentials and authentication failure with invalid credentials.
- Employee Profile Management: Test the ability to view and edit employee profiles.
- Leave and Attendance Reports: Verify the generation and review of leave and attendance reports.
- Timesheet Approval/Rejection: Test the approval and rejection process for employee timesheets.

3.2.2 Regression Testing:

Regression testing will be conducted throughout the testing phase to ensure that any updates or changes do not negatively impact existing functionality. This includes retesting of critical functional areas to confirm that they continue to work as expected.

3.2.3 Non-Functional Testing:

Non-functional testing will be performed to assess the application's performance, usability, and security. This phase will include:

- Performance Testing (Out of Scope): Performance testing will be conducted separately to evaluate the application's response times and scalability.
- Usability Testing: Usability testing will assess the application's user interface, including ease of navigation and user-friendliness.
- Security Testing (Out of Scope): Security testing will be conducted separately to identify and mitigate security vulnerabilities.

4. Test Approach

4.1 Test Design Approach

The test design approach for validating the HR Management application will employ a combination of various testing techniques, including but not limited to:

- **Equivalence Partitioning**: We will categorize inputs into groups that should be processed identically, helping us efficiently test different scenarios.
- **Boundary Value Analysis**: We will test the application's boundary conditions to identify any issues at the edges of valid input ranges.
- **Error Guessing**: Testers will use their domain knowledge to anticipate potential errors and validate the application accordingly.
- **Positive and Negative Testing**: We will validate expected behavior (positive testing) and also test scenarios that should result in errors or exceptions (negative testing).

The testing team will create detailed test cases, ensuring that they cover the specific business requirements described in the scope. These test cases will be reviewed and approved by the Project Team before test execution. Additionally, exploratory testing may be performed to discover potential issues not covered by formal test cases.

4.2 Execution Strategy

4.3.1 Entry Criteria

Entry Criteria	Conditions	Comments
Test environment(s) is available	✓	The test environment, including servers and databases, must be set up and accessible for testing.
Test data is available		Test data, including employee profiles, timesheets, and leave data, must be prepared for test execution.
Code has been merged successfully		The application code should be integrated and merged successfully to reflect the latest changes.
Development has completed unit testing		Unit testing by the development team must be completed and validated.
Test cases and scripts are completed, reviewed and approved by the Project Team		All test cases and scripts should be developed, reviewed, and approved by the Project Team.

3.2.2 Exit criteria

Exit Criteria	Conditions	Comments
100% Test Scripts executed	✓	All planned test scripts must be executed as per the test plan.
90% pass rate of Test Scripts		At least 90% of the test scripts should pass, indicating that most of the test cases have been successful.
No open Critical and High severity defects		All critical and high severity defects should be resolved and retested.
All remaining defects are either cancelled or documented as Change Requests for a future release		Any low or medium severity defects that are not fixed will be documented for consideration in future releases.
All expected and actual results are captured and documented with the test script		Comprehensive documentation of expected and actual results is crucial for traceability and analysis.

All test metrics collected based on reports from daily and Weekly Status reports		Test metrics, including test execution progress, defects, and test coverage, should be consistently reported.
All defects logged in -Defect Tracker/Spreadsheet		All defects should be recorded and tracked in a dedicated defect tracking system or spreadsheet.
Test environment cleanup completed and a new back up of the environment		The test environment should be cleaned up, and a backup taken for future reference.

3.3 Defect Management

Defect management in this project will follow a structured process to ensure that identified issues are effectively addressed. Here's how defects should be managed in the HR Management application project:

1. Defect Identification:
    - Testers will be responsible for identifying defects during the testing process. Defects can be related to functionality, usability, or any aspect of the application that does not meet the specified requirements.
2. Defect Logging:
    - Testers will log defects in a dedicated Defect Tracker or Spreadsheet. Each defect report should include the following details:
        - Defect description: A clear and concise description of the issue.
        - Steps to reproduce: Detailed steps to recreate the problem.
        - Severity: Categorize the severity of the defect based on its impact.
        - Impact: Describe the impact of the defect on the application and testing.
        - Expected behavior: Define what the expected behavior should be.
        - Actual behavior: Describe what is observed in the application.
        - Test case reference: Reference to the test case that identified the defect.
        - Attach screenshots or additional files, if relevant.
3. Defect Categorization:
    - Defects should be categorized based on their severity and impact. Use a standard classification, such as:
        - Severity 1 (Critical): Functionality is blocked, and no testing can proceed. The application/program/feature is unusable in the current state.
        - Severity 2 (High): Functionality is not usable, and there is no workaround, but testing can proceed.
        - Severity 3 (Medium): Functionality has issues, but there is a workaround for achieving the desired functionality.
        - Severity 4 (Low): Unclear error message or cosmetic errors with minimal impact on product use.

4. Defect Prioritization:
   - The project team, including developers, testers, and project managers, should prioritize defects based on their severity and impact. Critical and high severity defects should be given top priority.
5. Defect Assignment:
   - Defects should be assigned to developers or development teams responsible for addressing and resolving the issues. Clear ownership should be established.
6. Defect Resolution:
   - Developers will work on resolving the defects by making the necessary code changes. They should also perform unit testing to verify that the fixes are effective.
7. Defect Verification:
   - Testers should retest the defects to ensure that they have been successfully resolved and that no new issues have been introduced as a result of the fixes.
8. Defect Closure:
   - Once a defect is verified and deemed resolved, it should be marked as closed in the Defect Tracker or Spreadsheet, indicating that it is no longer an active issue.
9. Change Requests (CRs):
   - If low or medium severity defects are not addressed immediately, they may be documented as Change Requests for consideration in future releases. CRs should be reviewed and prioritized for implementation in subsequent iterations.
10. Defect Tracking and Reporting:
    - Detailed records of defects, their status, and resolutions should be maintained throughout the defect management process. These records are essential for tracking progress and reporting to project stakeholders.

  - *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* ▪ *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	1

5.2 Roles and Responsibilities

QA Manager:

- The QA Manager will oversee the entire testing process, including test planning, strategy development, and execution.
- Responsibilities include resource allocation, budget management, and ensuring that testing aligns with project objectives and quality standards.
- The QA Manager will also be responsible for communicating the testing progress and results to project stakeholders.

QA Leads:

- QA Leads will assist the QA Manager in defining the testing strategy and planning.
- They will be responsible for test case design and creation, as well as ensuring that test execution aligns with the test plan.
- QA Leads will coordinate the testing efforts of the QA Engineers and Senior QA Engineers.

Senior QA Engineers:

- Senior QA Engineers will have a deep understanding of the application's functionality and business requirements.
- They will be responsible for developing detailed test cases, executing test scripts, and identifying defects.
- Senior QA Engineers will mentor and provide guidance to QA Engineers, ensuring effective testing.

QA Engineers:

- QA Engineers will execute test scripts, report defects, and participate in test case design.
- They will actively contribute to the test execution and defect identification processes.
- QA Engineers will collaborate with Senior QA Engineers to ensure thorough testing of the application.

6. Test Schedule

Test Phase: Test Planning and Preparation

- Start Date: December 1, 2023
- End Date: December 15, 2023

Activities:

- Define the test strategy and approach.
- Develop test plans and test cases.
- Prepare the test environment and test data.

Test Phase: Functional Testing

- Start Date: December 16, 2023
- End Date: December 30, 2023

Activities:

- Execute functional test cases.
- Identify and report defects.
- Retest and verify defect fixes.

Test Phase: Regression Testing

- Start Date: January 1, 2024
- End Date: January 15, 2024

Activities:

- Execute regression test cases.
- Ensure that changes and updates do not negatively impact existing functionality.
- Verify defect resolutions.

Test Phase: Non-Functional Testing

- Start Date: January 16, 2024
- End Date: January 31, 2024

Activities:

- Perform non-functional testing, including usability testing and other relevant tests.
- Collect performance metrics.
- Security testing is performed in a separate phase.

Test Phase: Test Reporting and Defect Closure

- Start Date: February 1, 2024
- End Date: February 15, 2024

Activities:

- Generate test reports and document test results.
- Close defects as they are resolved.
- Review the test process and results with the project team.

7. Test Reporting

7.1.Test Reporting Approach

#	Report Name	Owner	Audience	Frequency
1	TEST PROGRESS REPORT	QA Manager	Project Managers, Development Team, QA Team, Stakeholders	Weekly
2	TEST PROGRESS REPORT	QA Leads	Project Managers, Development Team, QA Team, Stakeholders	Weekly

7.2. Quality Matrices

The following test matrices will be used and measured:

7.2.1 Defect Metrics:

- Defect Density: The number of defects identified per unit of size (e.g., per 1,000 lines of code).
- Defect Severity Distribution: Categorization of defects based on their severity (Critical, High, Medium, Low).
- Defect Closure Rate: The rate at which defects are resolved and closed.

7.2.2 Test Coverage Metrics:

- Test Case Coverage: The percentage of test cases executed relative to the total test cases defined.
- Requirement Coverage: The percentage of requirements covered by test cases.
- Code Coverage (if applicable): The percentage of code covered by executed test cases.

7.2.3 Test Execution Metrics:

- Test Execution Progress: The percentage of test cases executed over time.
- Defect Detection Rate: The rate at which defects are identified during testing.
- Defect Resolution Time: The time taken to resolve defects from identification to closure.

7.2.4 Test Efficiency Metrics:

- Test Execution Time: The time taken to complete test cycles.
- Test Case Preparation Time: The time taken to create, review, and approve test cases.
- Test Environment Setup Time: The time required to prepare the test environment.

8. Test Environment Requirements

- Hardware: The test environment should replicate the hardware setup that mirrors the production environment to ensure accurate testing. This includes servers, storage, and network infrastructure.
- Software: The test environment should include the necessary software components, including the operating system, web server, database server, and any other applications required for the HR Management application.
- Test Data: Sufficient and realistic test data should be available in the test environment to simulate real-world scenarios. This includes employee profiles, leave records, attendance data, and timesheet information.
- Browsers and Devices: The test environment should cover a variety of browsers and devices to ensure cross-browser and cross-platform compatibility. This includes popular browsers like Chrome, Firefox, Safari, and Edge, as well as different devices such as desktops, tablets, and mobile phones.
- Access Control: The test environment should replicate the access control mechanisms, ensuring that roles and permissions are configured as they are in the production environment. This includes the setup of HR user accounts with appropriate roles and privileges.
- Network Configuration: The test environment's network configuration should mimic the production network to validate application behavior under real network conditions, including firewall settings and network latency.
- Test Management Tools: Test management tools such as test case management and defect tracking software should be available to support test execution and defect management.

9. Dependencies and Assumptions

- Test Item Availability: It is assumed that the HR Management application and its components will be available and accessible for testing, including necessary configurations and test data.
- Testing Resource Availability: The availability of the test team, including QA Managers, QA Leads, Senior QA Engineers, and QA Engineers, is assumed as per the team structure mentioned earlier.
- Development Milestones: Testing is dependent on development milestones, including code integration and availability of code for testing. The assumption is that development will be on schedule.
- Test Environment Availability: It is assumed that the test environment will be set up and accessible for testing, including the necessary hardware, software, and test data.
- Defect Resolution Turnaround: The timely resolution of defects by the development team is essential for the testing process to progress smoothly.
- Stakeholder Availability: The assumption is that project stakeholders, including project managers and development leads, will be available for test reporting and issue resolution as needed.

- Deadlines: The testing schedule depends on the project's overall schedule and deadlines. Assumptions include adherence to project timelines and milestones.