Quality Assurance Plan

<Validating the HR Management application>

<3rd November 2023>

Created By:

19020742

Y.N. Samaradivakara

1 Introduction

1.1 Purpose

This Quality Assurance Plan (QAP) is established to outline the testing strategy, methodologies, and criteria that will guide the testing phase. This document serves as a guideline for the activities necessary to ensure that the basic HR Management application meets all specified requirements and quality standards.

1.2 Project Overview

The HR Management application is designed to enable HR personnel to securely log in, view and manage employee profiles, review and make decisions on leave and attendance reports, and approve or reject timesheets.

2 Scope

2.1 In-Scope

- User authentication and authorization
- Employee profile accessibility and data accuracy
- Leave and attendance reviewing functions
- Timesheet approval and rejection processes

3 Testing Strategy

3.1 Product/Application/Solution Risks

Risks	Criticality	Mitigation Strategy
Unauthorized access to employee data (Security risk)	High	● Implement role-based access control (RBAC) to ensure users can only access data necessary for their role. ● Conduct penetration testing to uncover potential security vulnerabilities. ● Utilize multi-factor authentication (MFA) for all users accessing sensitive information. ● Regularly update and patch the system to protect against known security threats. ● Conduct security awareness training for employees to recognize and avoid phishing attempts and other common threats. ● Encrypt sensitive data both at rest and in transit.
Timesheet approval workflow errors	High	● Map out all possible scenarios in the workflow and create comprehensive test cases to cover each path.

		● Utilize workflow automation tools to manage and monitor the approval process, ensuring that timesheets move through the system as expected. ● Set up a fallback procedure for manual approval in case the automated system fails.
Incorrect leave and attendance reports	Medium	● Incorporate automated checks to validate data upon entry for correctness and completeness. ● Implement unit testing and integration testing to ensure that all system components work directly together to produce accurate reports. ● Use test-driven development (TDD) approaches for critical report generation features to ensure they meet business requirements. ● Schedule regular audits of the reports generated by the system against manual checks or secondary systems to ensure accuracy.
System downtime	Medium	● Establish a robust disaster recovery and business continuity plan that

		includes regular backups and clear procedures for restoring the system in case of failure. ● Implement high-availability solutions to minimize the risk of downtime, such as database replication, failover clustering, or load balancing. ● Perform stress testing and load testing to ensure the system can handle high traffic volumes without failure.
Data Loss or Corruption	High	● Implement comprehensive input validation to prevent malicious data from entering the system, which can lead to data corruption. ● Schedule regular data backups and ensure they are stored securely in a separate location or cloud environment to prevent total data loss. ● Use database management systems that include transaction logging and rollback features to recover data in the event of corruption.
Inconsistent User Interface (UI) Across Different Browsers	Low	● Implement responsive design principles to ensure

		the UI is functional and appears consistent across various screen sizes and devices. ● Use a cross-browser testing tool to verify that the application behaves as expected on different web browsers. ● Develop a style guide and UI component library to maintain consistency in the look and feel across the application. ● Regularly update the application to comply with the latest web standards and browser updates. ● Engage in user acceptance testing (UAT) with a diverse group of users to gather feedback on UI consistency and usability

3.2 Level of Testing

3.2.1 Functional Testing	To verify all application functionalities against requirements
3.2.2 Regression Testing	To confirm that recent changes have not adversely affected existing functionalities
3.3.3 Non-Functional Testing	To assess the application's usability, security, and compliance with non-functional requirements

3.2.1 Functional Testing

- **Login Functionality**: Test the login process with both valid and invalid credentials. (https://online.actitime.com/ysam1/login.do)
- **Profile Access**: Ensure HR can view employee profiles after successful login. (https://online.actitime.com/ysam1/administration/userlist.do#page-1)
- **Leaves and Attendance Review**: Verify HR can access and review leaves and attendance reports. (https://online.actitime.com/ysam1/administration/userlist.do#page-1)
- **Timesheet Approval/Rejection**: Check the functionality for approving and rejecting timesheets. (https://online.actitime.com/ysam1/administration/approve_tt.do**)**

3.2.2 Regression Testing

- Ensure that new code changes do not affect the existing functionality of login, profile viewing, report reviewing, and timesheet processing.

3.3.3 Non-Functional Testing

- **Security Testing**: Confirm that user data is protected and access control is in place for sensitive HR functions.
- **Performance Testing**: Assess the application's performance, especially when multiple HR users are logged in and performing operations simultaneously.
- **Usability Testing**: Ensure the interface for HR tasks is intuitive and user-friendly.
- **Compatibility Testing**: Check the application's compatibility with various browsers and devices commonly used by the HR staff.

4. Test Approach

4.1 Test Design Approach

Our test design strategy will utilize a combination of black-box and white-box testing techniques to cover the breadth and depth of the HR Management application:

- Equivalence Partitioning: To reduce the number of test cases to a manageable level while covering all possible scenarios for input fields.
- Boundary Value Analysis: Focusing on the boundary conditions for inputs, such as the maximum and minimum values for numerical inputs and character limits for text fields.
- Decision Table Testing: For complex business rules, especially in the leave and attendance module and timesheet approval workflows.
- State Transition Testing: Especially for the login session management and state changes from submitted to approved/rejected for timesheets.
- Exploratory Testing: Encourage testers to use their creativity to explore the application's functionality beyond pre-written test cases.

4.2 Execution Strategy

4.3.1 Entry Criteria

Entry Criteria	Conditions	Comments
Test environment(s) is available	The test environment is set up and configured	Ensure it mirrors the production environment
Test data is available	Data for testing purposes has been created or obtained	Ensure it is anonymized if using real user data

Code has been merged successfully	The latest code base is deployed to the test environment	Confirm with development team
Development has completed unit testing	Developers have tested the unit components	Obtain unit test reports
Test cases and scripts are completed, reviewed and approved by the Project Team	Test plans and cases are ready for execution	Approval from QA leads and project managers

3.2.2 Exit criteria

Exit Criteria	Conditions	Comments
100% Test Scripts executed	All planned tests have been run	-
90% pass rate of Test Scripts	Acceptable pass rate achieved	Investigate failures to assess impact
No open Critical and High severity defects	Critical issues resolved	Prioritize fixes before release

All remaining defects are either cancelled or documented as Change Requests for a future release	Non-critical issues documented	Plan for future resolution
All expected and actual results are captured and documented with the test script	Test evidence is available	For audit and review purposes
All test metrics collected based on reports from daily and Weekly Status reports	Tracking progress and quality	-
All defects logged in Defect Tracker/Spreadsheet	Defects are recorded and tracked	Facilitates defect lifecycle management
Test environment cleanup completed and a new backup of the environment	Prepare environment for future tests or release	Confirm with IT operations

3.3 Defect Management

Defects will be managed through a Defect Tracker, with a lifecycle from discovery to verification of the fix. Critical and high-severity defects must be addressed immediately, with medium and low-severity defects scheduled for resolution based on priority and impact.

5. Test Team Structure

5.1 Team Structure

#	Role	Resource Count
1	QA Manager	1
2	QA Leads	2
3	Senior QA Engineers	2
4	QA Engineers	3+

5.2 Roles and Responsibilities

- **QA Manager**: Oversee the QA process, ensure quality standards, and communicate with stakeholders.
- **QA Leads**: Coordinate testing efforts, manage the test team, and ensure adherence to the test plan.
- **Senior QA Engineers**: Design test cases, perform complex testing, and mentor QA Engineers.
- **QA Engineers**: Execute test cases, report defects, and verify bug fixes.

6. Test Schedule

Week	Activity	Deliverable	Responsible Person
Week 1	Test Planning	Test Plan Document	QA Manager
Week 2	Environment Setup	Configured Test Environment	IT Operations Team
Week 3	Test Case Development	Test Cases for Authentication, Profile Review, Leave/Attendance, Timesheet Approval/Rejection	QA Engineers
Week 4	Test Case Review	Reviewed and Approved Test Cases	QA Leads
Week 5	Test Execution	Initial Test Results, Defect Logs for Authentication and Profile Access	QA Engineers
Week 6	Test Execution	Test Results for Leave/Attendance and	QA Engineers

		Timesheets, Defect Retest	
Week 7	Regression Testing	Regression Test Results	Senior QA Engineers
Week 8	Performance & Security Testing	Performance and Security Test Reports	Senior QA Engineers
Week 9	UAT Coordination & Testing	UAT Setup, Feedback, Defect Reports	QA Leads
Week 10	Final Test Cycle & Bug Fix Verification	Final Test Summary, Verified Bug Fixes	QA Team
Week 11	Test Closure Activities	Test Closure Report, Test Metrics, Recommendations for Future Releases	QA Manager

7. Test Reporting

7.1.Test Reporting Approach

#	Report Name	Owner	Audience	Frequency
1	Test Progress Report	QA Leads	Project Team	Weekly
2	Defect Report	QA Engineers	QA Leads	As Occurs

7.2. Quality Matrices

Feature Completion Rate: The percentage of required features (login, profile access, report review, timesheet approval/rejection) that have been implemented and tested.

- **Goal:** 100% of the features implemented and tested by the end of the testing cycle.

Authentication Success Rate: The percentage of successful logins using valid credentials during testing.

- **Goal:** 99% success rate, considering possible edge cases.

Report Accuracy: The percentage of generated leave and attendance reports that match the expected outcomes.

- **Goal:** 100% accuracy for all reports generated.

Timesheet Workflow Success Rate: The percentage of timesheet approvals and rejections that are processed correctly without system errors.

- **Goal:** 100% successful processing of timesheet workflows.

User Profile Access Rate: The ability to access and review employee profiles without errors.

- **Goal:** 100% success in accessing employee profiles without system errors.

Defect Density: The number of defects found per functional area (login, profile access, etc.) divided by the size of the application component.

- **Goal:** Zero defects per functional area by the release date.

System Availability: The percentage of time the application is available and operational during testing.

- **Goal:** 99.9% uptime during testing periods.

Response Time: The average time taken for the system to respond to a user action, such as logging in, accessing profiles, generating reports, or processing timesheet approvals/rejections.

- **Goal:** Average response time of less than 2 seconds for user actions.

Note: Numbers are taken based of assumptions

8. Test Environment Requirements

Requirement	Description
Web Servers	Same configuration as production to host the HR Management application.
Database Servers	Mirrored production database setup with sanitized test data.
Network Configuration	Simulated network conditions including bandwidth and latency.
Security Settings	Firewalls and other security mechanisms as per the production setup.
Browsers	Latest versions of Chrome, Firefox, Edge, Safari for cross-browser testing.
Test Data	Anonymized real-world data to test application functionality.
Monitoring Tools	Tools to monitor application performance and server health.

Error Logging	Systems to capture and analyze logs for debugging purposes.

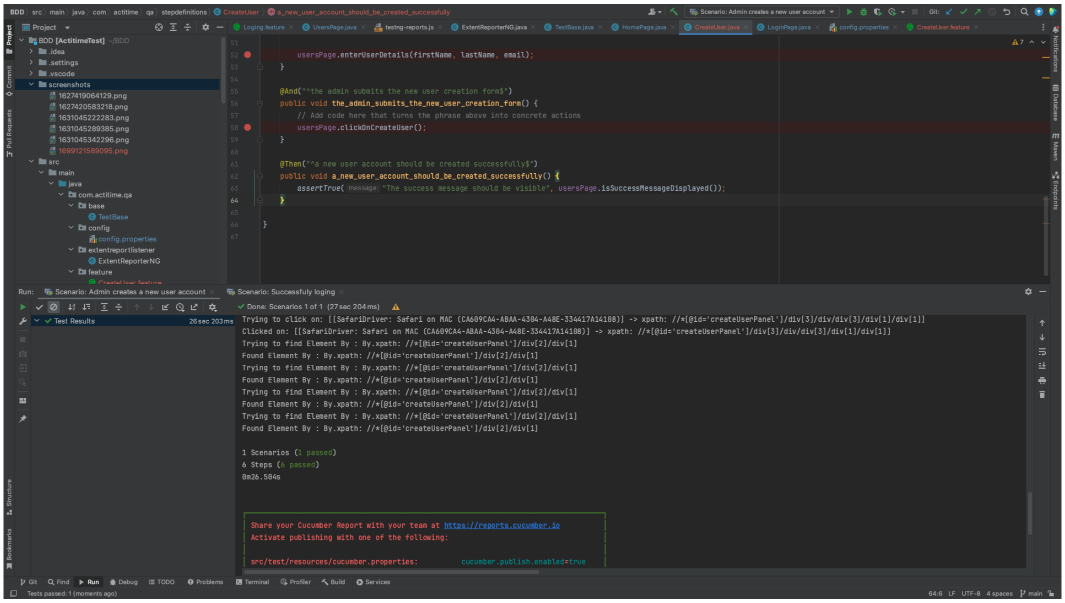
9. Dependencies and Assumptions

- Dependencies:
    - Availability of the test environment and access for the QA team.
    - Timely delivery of application builds for testing.
    - Access to real-world datasets for testing or creation of synthetic test data.

- Assumptions:
    - The test team will have the required access and permissions.
    - Test item availability will align with the project schedule.
    - Resources for testing (both hardware and personnel) will be available as scheduled.

Question 3

https://github.com/YasithSam/BDD



```java
        usersPage.enterUserDetails(firstName, lastName, email);
    }

    @And("^the admin submits the new user creation form$")
    public void the_admin_submits_the_new_user_creation_form() {
        // Add code here that turns the phrase above into concrete actions
        usersPage.clickOnCreateUser();
    }

    @Then("^a new user account should be created successfully$")
    public void a_new_user_account_should_be_created_successfully() {
        assertTrue( message: "The success message should be visible", usersPage.isSuccessMessageDisplayed());
    }

}
```

```
Trying to click on: [[SafariDriver: Safari on MAC (CA609CA4-ABAA-4304-A48E-334417A14108)] -> xpath: //*[@id='createUserPanel']/div[3]/div/div[3]/div[1]/div[1]]
Clicked on: [[SafariDriver: Safari on MAC (CA609CA4-ABAA-4304-A48E-334417A14108)] -> xpath: //*[@id='createUserPanel']/div[3]/div/div[3]/div[1]/div[1]]
Trying to find Element By : By.xpath: //*[@id='createUserPanel']/div[2]/div[1]
Found Element By : By.xpath: //*[@id='createUserPanel']/div[2]/div[1]
Trying to find Element By : By.xpath: //*[@id='createUserPanel']/div[2]/div[1]
Found Element By : By.xpath: //*[@id='createUserPanel']/div[2]/div[1]
Trying to find Element By : By.xpath: //*[@id='createUserPanel']/div[2]/div[1]
Found Element By : By.xpath: //*[@id='createUserPanel']/div[2]/div[1]
Trying to find Element By : By.xpath: //*[@id='createUserPanel']/div[2]/div[1]
Found Element By : By.xpath: //*[@id='createUserPanel']/div[2]/div[1]

1 Scenarios (1 passed)
6 Steps (6 passed)
0m26.504s


│ Share your Cucumber Report with your team at https://reports.cucumber.io
│ Activate publishing with one of the following:
│
│ src/test/resources/cucumber.properties:          cucumber.publish.enabled=true
```