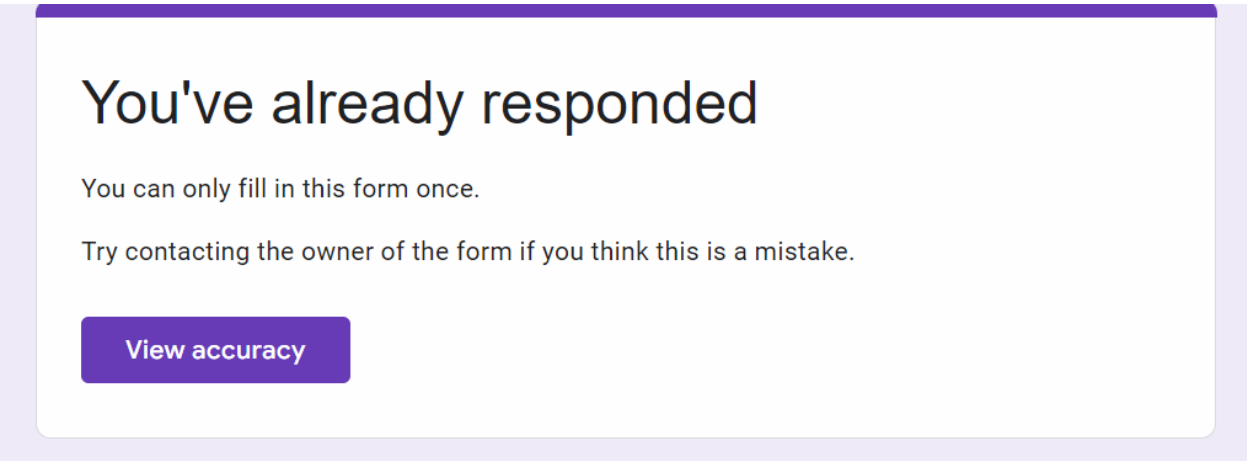
IS4102 - Advance Software Quality Assurance

Assignment 02

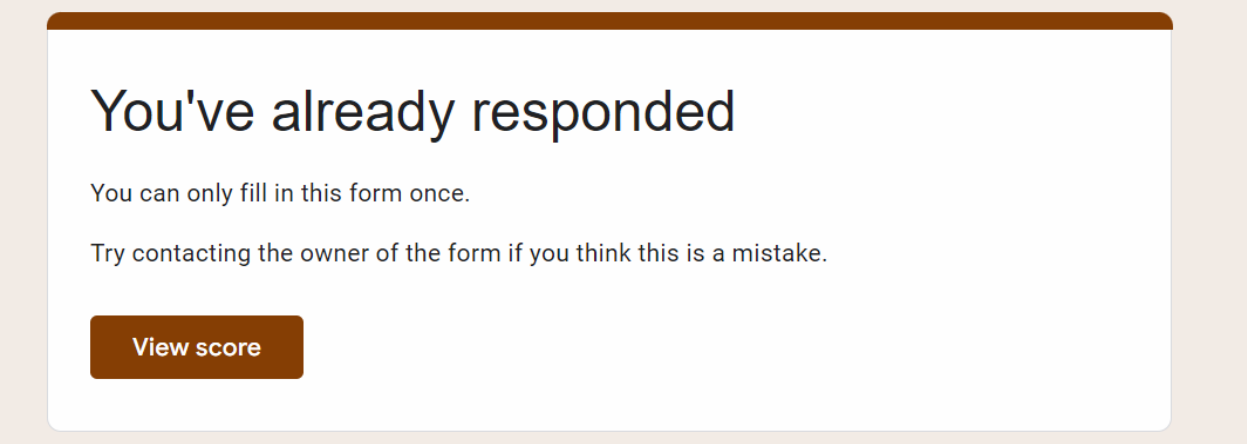
Sivamayuori Premnath – 19020635

Section 1 - Online Quizzes on Concepts

Quiz 01:



Quiz 02:



Section 2 - Industry Use Case - QA Management

Quality Assurance Plan

ActiTime:

Validating the HR Management application

30th October 2023

1 Introduction

1.1 Purpose

The purpose of this Quality Assurance Plan is to verify that the ActiTime, an HR Management application fulfills the specified requirements as the following:

- HR can log in with a valid username and valid password.
- HR can view the Employee profiles.
- HR can review the leaves and attendance reports of the empoyees.
- HR can approve or reject the timesheets

1.2 Project Overview

ActiTime, an HR Management application allows HR to log in with a valid username and valid password. The HR once logged into the system successfully can be able to investigate Employee profiles and review leaves and attendance reports and Approval/rejection of the timesheets.

In order for the application to function properly according to the requirements, it needs to be tested for its core modules, their flow between each other and issues related to authentication.

2 Scope

2.1 In-Scope

- Login functionality: Ability to login with a valid username and password.
- Employee profiles functionality: Ability to Iook into the profiles of various employees in the organization.
- Leaves and Attendance reports functionality: Ability to keep track of the attendance of the employees and leaves taken by them.
- Timesheets functionality: Ability to view and approve/reject the timesheets of the employees.
- Non-functional testing: Performance and security testing.

2.2 Out-of-Scope

With the given scenario, there are certain features that cannot be tested in the current context.

- Other modules such as payroll, performance and recruitment functions specific to HR.
- Integration with other systems within the organization (Example: ERP system of the company)

It is assumed that, in the current context, only functional and non-functional testing is performed with the above-mentioned functionalities in the in-scope. However when most scenarios are tested that are specified, with the stakeholders' agreement, other testing strategies can be implemented before release.

3 Testing Strategy

3.1 Product/ Application/ Solution Risks

Risks	Criticality	Mitigation Strategy
Invalid username or password	High	• Implement strong password policies and requirements. • Use two factor authentication. • Implement lockout policies for failed login attempts.
Unable to view employee profiles	Medium	• Implement access control lists to restrict access to employee profiles (Usage of Bel La Padulla model) • Test the application with different types of employee profiles and user roles. • Log and monitor all employee profile access.
Unable to review leaves and attendance reports	Medium	• Implement access control lists to restrict access to leaves and attendance reports. • Log and monitor all leaves and attendance report access. • Test the application with different types of leave and attendance data. • Test the application with different date ranges (past, present and future date conditions).
Unable to approve/reject timesheets	Medium	• Test the application with different types of timesheets and approval workflows. • Implement access control lists to restrict access to timesheets. • Log and monitor all timesheet approval/rejection activity.
Data corruption	High	• Implement regular data backups. • Use a version control system to track changes to the application code and database.

		• Implement data validation checks to prevent invalid data from being entered into the system.
Unauthorized access	High	• Implement strong authentication and authorization mechanisms. • Use firewalls and other security measures to protect the application from unauthorized access. • Monitor the application for suspicious activity.
Performance Issues	Medium	• Perform load and performance testing to identify and address any performance bottlenecks. • Use a scalable architecture to support the expected load on the application. • Monitor the application's performance and make adjustments as needed.
Integration Issues	Medium	• Perform integration testing to ensure that the application integrates correctly with other systems. • Develop and document clear integration requirements. • Use a standard integration framework.
Usability Issues	High	• Perform usability testing with real users to identify and address any usability issues. • Develop a user-friendly interface that is easy to navigate. • Provide clear and concise instructions for using the application.

3.2 Level of Testing

Test Type	Description
Functional Testing	To verify that the HR Management application meets the requirements of the users and stakeholders. It includes testing all of the features of the application, such as login, employee profiles, leaves and attendance reports, and timesheets.

Non-Functional Testing	To verify that the application meets non-functional requirements such as performance, security, and usability.
Regression Testing	To ensure that changes to the application do not break existing functionality. It includes re-running the test cases and making sure that no new defects have risen.
Integration Testing	To ensure that the HR Management application integrates correctly with other systems, such as the company's ERP system.

4 Test Approach

4.1 Test Design Approach

The HR application Test Approach will focus on the following testing approach:

- Data-driven approach: As the internal logic is the same, the application is tested using a variety of different data values as different authorized users will have different usernames, passwords and access levels.
- Risk based: The test cases will be prioritized based on the risks identified with their criticality during the risk assessment phase.
- Coverage: The test cases will be designed to cover all the specified requirements of the application.

The following test design techniques in relation to the application will be used:

- Equivalence partitioning: The input domain for the login functionality can be divided into the following equivalence classes: valid usernames and passwords, invalid usernames, and invalid passwords. Test cases would be created to represent each equivalence class.
- Boundary value analysis: The boundary values for the leave request functionality are the first and last days of the year. Test cases would be created to verify that a user can submit a leave request for the first and last days of the year.
- State-based testing: The state-based test cases for the timesheet functionality would verify that the application correctly transitions between the following states: draft, submitted, approved, and rejected.
- Decision table testing: The decision table test cases for the timesheet approval workflow would verify that the application correctly approves/rejects timesheets based on the following factors: the type of timesheet, the employee's department, and the manager's approval.
- Error guessing: The error guessing test cases include the following: trying to log in with an empty username or password, trying to submit a leave request for a date in the past, and trying to approve a timesheet for an employee who is not in the manager's department.

4.2 Execution Strategy

4.2.1 Entry Criteria

Entry Criteria	Conditions	Comments
User requirements have been agreed by the stakeholders and documented		
Test environment is available and configured properly		
All test cases have been written and reviewed		
All test data has been prepared		
The database is populated with the necessary test data		
Code has been merged successfully		
Development has conducted unit testing		

4.2.2 Exit Criteria

Exit Criteria	Conditions	Comments
100% Test Scripts have been executed		
90% pass rate of test scripts		
All expected and actual results are captured and documented with the test script		
Deviations/ Defects that have been identified are fixed and tested		
No High critical and severe defects are open		
Number of estimated remaining defects is below 5%		
All test metrics collected based on reports from daily and weekly status reports		

4.3 Defect Management

The defect management process for the HR Management application will follow the following steps:

I. Defect identification: Defects can be identified by testers, developers, or other stakeholders during testing, development, or production use.

II. Defect logging: Defects should be logged in a defect tracking system such as Jira. The defect log should include the following information:
    - Defect description
    - Steps to reproduce the defect
    - Expected behavior
    - Actual behavior
    - Severity
    - Priority
    - Assigned to

III. Defect triage: The development team will triage defects to determine their severity and priority. Defects will be prioritized based on the following factors:
    - Impact on users
    - Impact on business
    - Risk to production

IV. Defect resolution: The development team will fix the defects based on their priority.

V. Defect verification: The testers will verify that the defects have been fixed correctly.

VI. Defect closure: Once the defect has been fixed and verified, the defect will be closed.

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

Assuming that the organization developing the application and testing is relatively small and consists of 40 employees in the QA team.

#	Role	Resource Count
1	QA Manager	01
2	QA Leads	02
3	Senior QA Engineers	12
4	QA Engineers	25

5.2 Roles and Responsibilities

QA Manager - The QA Manager is responsible for overseeing the entire QA process, including planning, execution, and reporting. Their specific responsibilities may include:

- Developing and implementing the QA strategy for the project
- Managing the QA budget and resources
- Hiring, training, and managing the QA team
- Creating and maintaining test plans, test cases, and test data
- Executing and reporting on test results
- Working with the development team to resolve defects within the time frame

QA Leads - QA Leads are responsible for managing specific QA teams and ensuring that they meet their goals. Their specific responsibilities may include:

- Assigning and managing test cases to QA Engineers
- Providing technical support and guidance to QA Engineers
- Reviewing test results and reporting defects to the QA Manager
- Working with the QA Manager to identify and mitigate risks
- Helping to develop and implement the QA strategy

Senior QA Engineers - Senior QA Engineers are responsible for designing and executing complex test cases, as well as mentoring and training QA Engineers. Their specific responsibilities may include:

- Designing and developing test plans and test cases for complex functionality
- Automating test cases
- Mentoring and training QA Engineers
- Working with the QA Manager and QA Leads to identify and mitigate risks
- Helping to develop and implement the QA strategy

QA Engineers - QA Engineers are responsible for executing test cases and reporting defects. Their specific responsibilities may include:

- Executing test cases according to the test plan
- Reporting defects to the QA Lead
- Working with the development team to resolve defects
- Helping to maintain and update test plans and test cases

In addition to the above, the QA team will also be involved in other activities such as:

- Performance testing
- Security testing
- Usability testing

6 Test Schedule

Assuming that the project needs to be tested within 1 month with the budgetary constraints and specified requirements of the stakeholders.

Week 1

Day 1: Set up the test environment and install the application.

Day 2: Review the test plan and test cases.

Day 3: Start executing test cases for the login functionality.

Day 4: Continue executing test cases for the login functionality.

Day 5: Fix any defects found and re-execute the affected test cases.

Week 2

Day 1: Start executing test cases for the employee profiles functionality.

Day 2: Continue executing test cases for the employee profiles functionality.

Day 3: Start executing test cases for the leaves and attendance reports functionality.

Day 4: Continue executing test cases for the leaves and attendance reports functionality.

Day 5: Fix any defects found and re-execute the affected test cases.

Week 3

Day 1: Start executing test cases for the timesheets functionality.

Day 2: Continue executing test cases for the timesheets functionality.

Day 3: Start executing integration tests.

Day 4: Continue executing integration tests.

Day 5: Fix any defects found and re-execute the affected test cases.

Week 4

Day 1: Execute performance tests.

Day 2: Execute security tests.

Day 3: Execute usability tests.

Day 4: Fix any defects found and re-execute the affected test cases.

Day 5: Generate a final test report and present it to the stakeholders.

7 Test Reporting

7.1 Test Reporting Approach

#	Report Name	Owner	Audience	Frequency
1	TEST SUMMARY REPORT	QA Manager	Stakeholders (project manager, product manager, development team)	Weekly
2	TEST CASE EXECUTION REPORT	QA Lead	QA Team, development team	Daily
3	DEFECT TRACKING REPORT	QA Manager	Stakeholders, development team	Daily
4	PERFORMANCE TEST REPORT	QA Engineer	Stakeholders, development team	On Demand
5	SECURITY TEST REPORT	QA Engineer	Stakeholders, development team	On Demand
6	USABILITY TEST REPORT	QA Engineer	Stakeholders, development team	On Demand

7.2 Quality Metrics

The following quality metrics can be used in the scenario described:

- Test case execution rate: The percentage of test cases that were executed.
- Defect detection rate: The percentage of defects that were found during testing.
- Defect resolution rate: The percentage of defects that were fixed before the application was released to production.
- Test coverage: The percentage of the application's functionality that was covered by the test cases.
- Mean time to detect (MTTD): The average time it took to find a defect.
- Mean time to repair (MTTR): The average time it took to fix a defect.

8 Test Environment Requirements

The following are the test environment requirements for the HR Management application:

Hardware:

A. Server:
    - CPU: 4 cores

- Memory: 16 GB RAM
- Storage: 200 GB

B. Client workstations:
    - CPU: 2 cores
    - Memory: 8 GB RAM
    - Storage: 100 GB

Operating system:

A. Server: Windows Server 2022 or Linux
B. Client workstations: Windows 10 or macOS

Database: Microsoft SQL Server or PostgreSQL

Web browser: Google Chrome, Mozilla Firefox, Microsoft Edge

Other software:

- Java Development Kit (JDK)
- Maven
- Selenium
- TestNG

9 Dependencies and Assumptions

Dependencies:

- The HR Management application must be installed and configured on the test server.
- The database must be configured and populated with test data.
- The test environment must have a dedicated network and firewall.
- The test team must have access to the necessary hardware, software, and resources.

Assumptions:

- The HR Management application is developed using Java.
- The database is Microsoft SQL Server or PostgreSQL.
- The test environment is isolated from the production environment.
- The test data is representative of production data.
- The test team has the necessary skills and experience to test the HR Management application.
- The HR Management application must be used by a variety of users with different roles and permissions.
- The HR Management application must be able to handle a variety of different data types, such as employee profiles, leave requests, and timesheets.