Software Quality Engineering (Fall 2024)

IEEE Compliant

TEST PLAN FOR INTRA EMAIL AUTH

Department of Computer Science

FAST – National University of Computer & Emerging Sciences

Islamabad Campus

Group Members

Hammad Ali (22i-0628)

Zaim Abbasi (22i-2462)

Hussain Murtaza (22i-1513)

Course Instructor: Nigar Azhar Butt

Table of Content

1) Test Plan Identifier
2) References
3) Introduction
4) Test Items
5) Software Risk Issues
6) Features to be Tested
7) Features not to be Tested
8) Approach
9) Item Pass/Fail Criteria
10) Suspension Criteria and Resumption Requirements
11) Test Deliverables
12) Remaining Test Tasks
13) Environmental Needs
14) Staffing and Training Needs
15) Responsibilities
16) Schedule
17) Planning Risks and Contingencies
18) Approvals
19) Glossary

1. Test Plan Identifier

Test Plan ID: TP-NADRA-INTRA-EMAIL-SERVICE-001
Software Level: Application-Level Test Plan
Plan Type: Integration Test Plan
Version: 1.0

Reviewed and documented by

Syed Hussain Murtaza	i221513@nu.edu.pk
Hammad Ali	i220628@nu.edu.pk
Zaim Khan Abbasi	i222462@nu.edu.pk

This test plan will be updated dynamically as the project evolves, and future revisions will reflect any changes in the scope, test strategy, or project requirements.

2. REFERENCES

None identified

3. Introduction

The **NADRA INTRA Email Service** is a secure, web-based platform developed using Java Spring Boot to facilitate internal email communication within NADRA. This test plan aims to outline the strategy, scope, and specific test cases designed to ensure the application's functionality, reliability, and security. Given the critical role of this service in managing internal communications, the testing process focuses on key features like user authentication, QR code-based two-factor authentication, captcha verification, CSV data management, and robust session handling.

The purpose of this test plan is to validate that the application meets its requirements, adheres to best practices for secure communication, and provides a seamless user experience. This document details the testing approach for each functionality, ensuring comprehensive coverage and alignment with the project's objectives.

4. Test Items (Functions)

The following items represent the functions and components that will be tested within the scope of this Test Plan. These test items are derived from the Requirements Specification Document and the High-Level Design Document. The testing activities will ensure that all components function as expected and meet the required quality standards.

1. Email Management Module

- Creation, sending, receiving, and deletion of emails.
- Email folder organization (e.g., Inbox, Sent, Trash).
- Drafts and scheduled email functionalities.

2. User Management

- User registration and login functionalities.
- Role-based access control (e.g., Admin, Regular User).
- Password reset and profile management.

3. Security Features

- End-to-end email encryption.
- Secure authentication mechanisms.
- Protection against spam and unauthorized access.

4. Captcha Service

- Generation of captchas for login and form submissions.
- Validation of captcha inputs to prevent automated access.

5. OTP Service

- Generation of one-time passwords (OTPs) for two-factor authentication.
- Validation of OTPs during the login process.

6. Encryption Utility Service

- Secure encryption and decryption of sensitive data.
- Validation of encryption standards and data integrity.

7. System Administration Features

- Logging and monitoring of email activities.
- Configuration of system parameters and email templates.

These areas are prioritized to ensure the system's functionality, security, and compliance with the specified requirements.

5. Software Risk Issues

This section identifies potential risks in the NADRA INTRA Email Service and highlights critical areas that require special attention during the testing process. These risks are associated with the software's complexity, security vulnerabilities, and operational dependencies. Addressing these risks is essential to ensuring the system's quality, reliability, and compliance with requirements.

1. Critical Areas

- **OTP Fetching Vulnerabilities:** Risks of interception or unauthorized access to one-time passwords (OTPs) used for two-factor authentication.
- **Admin Login Security:** Unauthorized access to the admin panel due to weak credentials or lack of robust access controls.
- **Email Verification Flaws:** Issues in validating email addresses that could lead to unauthorized account creation or spam accounts.
- **Captcha Creation and Validation:** Risks of bypassing captchas or generating weak captchas that do not effectively prevent automated access.
- **Encryption Utility Misuse:** Weak or improper encryption of sensitive data, risking exposure of user credentials or email content.

2. Inherent Risks

- **Data Safety:** Ensuring data integrity during email transactions and preventing data loss or unauthorized data access.
- **Authentication Mechanism Security:** Protecting against brute-force attacks or replay attacks targeting the login and OTP mechanisms.
- **Session Management Flaws:** Risks related to session hijacking or insufficient session timeout mechanisms.
- **Operational Impact:** Failures in key functionalities like email sending or receiving could directly disrupt internal communication workflows.

3. Requirements Issues

- **Vague or Unclear Requirements:** Ambiguities in specifications for features like OTP expiration time or captcha difficulty could lead to improper implementation.
- **Untestable Requirements:** Features that lack measurable acceptance criteria, such as "secure user experience," pose challenges during testing.

4. Defect History

- **Frequent Defect Areas:**
    - Modules related to email attachment handling and drafts that previously exhibited defects.
    - Early-stage issues in OTP validation and captcha generation mechanisms.

Risk Identification Process

To address these risks effectively, the following steps will be taken:

- Conduct brainstorming sessions with stakeholders to uncover potential concerns, focusing on questions such as:
    - "What are the most likely points of failure in the system?"
    - "What security concerns could arise from implementing these features?"
- Analyze historical defect data to identify patterns in error-prone modules, such as email handling and authentication services.
- Implement enhanced testing and validation for critical modules, focusing on security and performance under various scenarios.

By identifying and mitigating these risks during testing, we aim to ensure the NADRA INTRA Email Service meets its objectives while maintaining a high standard of security and functionality

6. Features to be Tested

This section lists the features of the NADRA INTRA Email Service to be tested, described from the user's perspective. Each feature is assigned a risk level (High, Medium, Low) based on its importance and potential impact.

1. User Authentication

- **Description:** Verifying that users can securely log in and log out using their credentials and two-factor authentication (OTP).
- **Risk Level:** High (Critical for system access and overall security).

2. Captcha Service

- **Description:** Testing the captcha generation and verification process to ensure automated logins are blocked effectively.

- **Risk Level:** High (Critical for preventing unauthorized and automated access).

3. OTP Service

- **Description:** Validating OTP generation, delivery, and expiry for secure two-factor authentication during login.
- **Risk Level:** High (Essential for user account protection).

4. Encryption Utility Service

- **Description:** Verifying the encryption of sensitive data, including email content, credentials, and attachments, for end-to-end security.
- **Risk Level:** High (Vital for maintaining data confidentiality and compliance).

5. Draft Saving and Recovery

- **Description:** Ensuring emails are saved as drafts automatically and can be recovered in case of interruptions (e.g., tab closure or session timeout).
- **Risk Level:** Medium (Important for user convenience and productivity).

6. Spam Detection and Filtering

- **Description:** Testing the system's ability to automatically detect and filter spam and malicious emails.
- **Risk Level:** High (Critical for security and user satisfaction).

By focusing on these features, the test plan aims to ensure the NADRA INTRA Email Service meets the expected quality, functionality, and security standards.

7. Features Not to Be Tested

This section identifies the features that will not be included in the current testing cycle. Each feature's exclusion is justified from the perspective of the user and configuration management/version control. The exclusions are based on factors like risk level, stability, or their planned inclusion in future releases.

1. **Email Signature Customization**
    1. **Reason:** Not included in this release; planned for a future update.
    2. **Explanation:** This feature enables users to add and customize email signatures but is deferred to a subsequent software release.
2. **Third-Party Integration with Messaging Apps**

    1. **Reason:** Low risk, already tested in prior versions.

2. **Explanation:** Integrations with messaging apps (e.g., WhatsApp, Slack) have demonstrated stability in earlier testing cycles, so no additional testing is required for this release.

3. **Advanced Reporting Features**

    1. **Reason:** Low risk, proven stability in previous versions.
    2. **Explanation:** Reporting functionalities tested in earlier releases have shown consistent performance and will not be re-evaluated in this cycle.
4. **Dark Mode**

    1. **Reason:** Planned for a future update.
    2. **Explanation:** The dark mode feature is not part of the current release and will undergo testing in an upcoming version.
5. **Mobile App Compatibility**

    1. **Reason:** Low risk, tested previously.
    2. **Explanation:** The mobile app's compatibility with the software has been verified in earlier versions and is considered stable, thus excluded from the current testing cycle.
6. **Enhanced Accessibility Features**

    1. **Reason:** Scheduled for a future release.
    2. **Explanation:** New accessibility features like screen reader compatibility and voice command support are planned for later releases and are not part of this version's testing scope.

8. Approach (Strategy)

The test strategy outlines the approach for the testing efforts in this plan, ensuring alignment with high-level project objectives while addressing testing requirements, tools, and processes. This strategy aims to achieve efficiency, adequate coverage, risk mitigation, and effective configuration management.

1. Test Tools and Training

Test Tools:

- JIRA: For tracking issues and monitoring progress.
- JUnit: For unit testing.

Training:

- Basic training sessions for testers on using JUnit.
- Advanced training for test leads on configuring test environments and integrating tools into workflows.

2. Test Case Design

Test Cases:

- Detailed test cases will be created and documented, covering both **White-Box** and **Black-Box Testing:**
    - **White-Box Testing:**
        - *Statement Coverage*
        - *Decision Coverage*
    - **Black-Box Testing:**
        - Techniques such as *Equivalence Partitioning* and *Boundary Value Analysis* will be applied.

Test Case Format:

All test cases will follow the format used in previous assignments, including:

- Test Case ID
- Objective
- Pre-conditions
- Steps to Execute
- Expected Results
- Actual Results
- Status (Pass/Fail)

3. Defect Reports

Defects will be documented using a defect-tracking tool such as Jira including:

- Defect ID
- Summary
- Steps to Reproduce
- Expected Results vs. Actual Results
- Severity and Priority
- Status (e.g., New, Open, Fixed)

4. Automation Scripts

Automation will focus on three key test cases using tools like JUnit. Scripts will be:

- Well-documented with inline comments explaining logic.
- Selected based on critical functionalities to ensure reliable automation coverage.

5. Metrics Collection

Test Metrics:

- Test Case Execution: Track the number of executed, passed, and failed test cases.
- Test Coverage: Measure the percentage of requirements tested.
- Defect Metrics: Monitor defect density, open vs. closed defects, and trends.

6. Static Analysis Review

Static analysis will be conducted on selected project artifacts (e.g., source code or requirement documents). Techniques include:

- Manual peer review.

Outcome:

Document issues found, their impact, and recommendations for improvement.

9. Item Pass/Fail Criteria

The pass/fail criteria establish clear standards for each level of testing, ensuring that testing objectives are met and critical defects are resolved before software release. These criteria focus on test case completion, defect severity, and overall testing quality.

1. Unit Test Level Criteria

At the unit test level, the focus will be on validating individual components of the system. The criteria for passing unit testing are as follows:

- **Completion of All Test Cases:** Every test case for the unit must be executed; no test cases should be pending.
- **Test Case Pass Rate:** A minimum percentage of test cases must pass (e.g., 95% pass rate).

- **Defects:** Minor defects, such as UI glitches or non-critical issues, may be accepted within a predefined limit. These defects must be documented and prioritized for future releases.
- **Code Coverage:** A code coverage tool will be employed to ensure all lines and paths in the unit are tested. The target code coverage is typically set at 90% or higher.
- **Defect Severity:**
    - **Critical Defects:** Any critical defect causing system failure or halting functionality must be fixed before the unit passes.
    - **Minor Defects:** A limited number of minor defects may be tolerated, but they must be documented and addressed in future releases.

2. Acceptance Criteria for Pass/Fail

The system will pass the test plan if the following conditions are met:

- **Critical Functionality:** All critical functionality must be verified and pass without failures.
- **Severe Defects:** The number of severe defects should not exceed the predefined threshold (e.g., no more than two high-severity defects).
- **Test Case Success Rate:** The percentage of successfully executed test cases should exceed the minimum acceptable threshold (e.g., 95% pass rate).
- **Code Coverage:** Code coverage should be deemed adequate (typically 90% or higher).

If the system does not meet these criteria, it will fail the test phase. All defects must be addressed before moving to the next round of testing or release.

10. Suspension Criteria and Resumption Requirements

Suspension Criteria:

Testing will be paused if critical conditions or defects render further testing ineffective. The criteria for suspension are:

1. **High Severity of Defects:** If a critical defect (e.g., system crash, data loss) impacts core functionality or usability, testing will be suspended until resolved.
2. **Defect Threshold Exceeded:** If five or more critical defects are found, or if 20% of test cases fail due to defects, testing will pause until the defects are addressed.
3. **Test Environment Issues:** Testing will be suspended if environment issues (e.g., incorrect hardware or tool malfunctions) prevent proper execution.
4. **Unclear Requirements or Test Cases:** If requirements or test cases are unclear, testing will pause until clarification is provided.
5. **Ghost Errors or Invalid Defects:** Fatal errors may cause cascading defects. Testing will be paused until the root cause is fixed to avoid further invalid errors.

Resumption Requirements:

Testing will resume once the issues causing suspension are resolved. Resumption criteria include:

1. **Resolution of Critical Defects:** All critical defects must be fixed and verified before testing resumes.
2. **Defects Fix and Verification:** If testing was paused due to defects, they must be prioritized and fixed. Retesting of affected areas will be required.
3. **Test Environment Stability:** Once environment issues are fixed and confirmed stable, testing will resume.
4. **Clear and Updated Test Cases:** Test cases must be clarified and updated. Testing will resume once the scope is clear.
5. **No Ghost Errors:** Testing will only resume after investigating and resolving any ghost errors or invalid defects.

11. Test Deliverables

The following deliverables will be produced and provided as part of this test plan. These deliverables ensure that all critical aspects of the testing process are documented, traceable, and verifiable, contributing to a thorough validation of the software.

1. **Test Cases and Execution**
    1. **Description:** A complete set of test cases will be developed, specifying the test scenarios to be executed during various testing phases such as unit testing, integration testing, and system testing. Each test case will include:
        1. **Input Conditions:** The specific input data or parameters required for executing the test case.
        2. **Expected Results:** The expected outcome of the test case based on the input conditions.
        3. **Execution Instructions:** Detailed steps outlining how the test case will be executed.
    2. **Execution Results:** The results of executing the test cases will be documented, indicating whether each test case has passed or failed. This will also include any deviations from expected results, and detailed logs where necessary to capture the execution process.

2. **Static and Dynamic Generators (Static Analysis Report - 10%)**

1. **Description:** Static and dynamic code analysis tools will be utilized to assess the quality and performance of the code. This will help identify areas of potential vulnerability, inefficiencies, or code anomalies.
    1. **Static Analysis Tools:** These tools will analyze the codebase without executing the program, focusing on areas such as security vulnerabilities, code style issues, and potential bugs.
2. **Deliverables:**

    1. **Static Analysis Report:** A detailed report generated from static analysis that highlights potential vulnerabilities, inefficiencies, and areas for improvement in the code.
3. **Defect Reporting**
    1. **Description:** Defect reports will document all defects and issues discovered during the testing process. The report will include detailed information about the defects, such as:
4. **Automation Scripts**

    1. **Description:** Automation scripts will be developed for automating the execution of repetitive or complex test cases. These scripts are crucial for improving testing efficiency, consistency, and repeatability, especially for regression testing and performance testing.

12. Reamining Test Tasks

Certain functionalities of the NADRA INTRA Email Service are not covered in this test plan and will need to be addressed in future phases. The following tasks and considerations should be noted:

1. **Uncovered Application Areas:**
    1. Some features are excluded from this phase either due to their future development or scope limitations. These should be explicitly listed for clarity, and new features will require separate testing plans.
2. **Multi-Phase Releases:**

    1. This test plan focuses on the current phase, including authentication and CSV management. Future phases, with additional features, will require their own test plans.
3. **Third-Party Development:**

    1. If external vendors are involved, the responsibilities for testing should be clearly defined to avoid overlaps. Coordination with third-party teams is essential for defect management.
4. **Future Functionality Testing:**

    1. Features not tested in this phase due to their development timeline must be identified, allowing testers to focus on currently relevant tasks.
5. **Defect Tracking and Resolution:**

    1. Defects should be tracked and assigned to the correct phase. It's important to ensure defects unrelated to the current phase are not misclassified.

13. Environmental Needs

To ensure the successful execution of this test plan, the following environmental requirements must be met:

1. **Test Data Requirements:**

    1. **Predefined Datasets:** Test data will consist of predefined datasets that cover typical user scenarios and edge cases.
    2. **Data Ranges:** Attention will be given to testing both typical and extreme input values to ensure system robustness.
    3. **Special Data Configurations:** Specific data formats (e.g., database entries, file formats) will be required to simulate real-world user interactions.
2. **Testing Multi-Part Features:**

    1. Some features consist of multiple components that need to be tested individually and as part of an integrated system.
    2. Each sub-component will be tested both in isolation and as part of the larger system to ensure overall functionality.
3. **Supporting Software Versions:**

    1. Testing will require certain versions of supporting software (e.g., operating systems, databases, development tools) to ensure compatibility and consistency with the production environment.
    2. The test environment must mirror the production setup as closely as possible to avoid discrepancies.
4. **System Use Restrictions:**

1. During testing, some system areas may be restricted to ensure accurate test results, particularly for tests that impact system performance or require exclusive access to resources.
2. Coordination with other teams is essential to avoid conflicts and system usage interference during critical test phases.

13. Staffing and Training Needs

1. **Application/System Training**:
    - Testers must be trained on the system's features, functions, and architecture to understand how it works and identify potential issues.
2. **Test Tools Training**:
    - Testers need training on any test tools used for automation, defect tracking, and performance testing, ensuring effective tool usage.
3. **Staffing Responsibilities**:
    - **Test Leads**: Oversee testing, manage test plans, and ensure requirements are met.
    - **Test Engineers**: Execute test cases, report defects, and assist with tool setup.
    - **Specialized Testers**: Handle complex areas like security or performance testing.
    - **Configuration Management Team**: Manage version control and configurations for testing.
4. **Coordination**:
    - Training should align with the test items and environment requirements to ensure staff is prepared for specific tools and test conditions.

14. Responsibilities

1. **Risk Management**:
    - **Test Manager**: Responsible for identifying and setting risks, ensuring that all potential risks are assessed and managed.
2. **Feature Selection**:
    - **Test Manager & Test Leads**: Decide which features will be tested and which will not, based on the test scope, risk assessment, and project requirements.
3. **Test Strategy**:
    - **Test Manager**: Sets the overall strategy for the test plan, ensuring alignment with the project's goals, resources, and schedule.
4. **Test Readiness**:
    - **Test Leads**: Ensure that all elements (tools, data, environment) are in place and ready for testing.
5. **Scheduling Conflicts**:
    - **Project Manager & Test Manager**: Resolve any scheduling conflicts, particularly when testing needs to occur on a live system or shared environment.
6. **Training**:

    - **Test Leads/HR/Training Manager**: Organize and provide required training, whether for the application, testing tools, or processes.
7. **Go/No-Go Decisions**:
    - **Test Manager & Key Stakeholders**: Make the final decisions on whether certain test items, not explicitly covered in the test plan, should proceed based on severity and project impact.

15. Schedule

The testing schedule should be realistic, factoring in potential development delays and adjusting accordingly.

- **Testing in the Overall Project:** Testing is a key part of the project. If development is delayed, inform users early, emphasizing that more testing time leads to a more reliable product.
- **Handling Delays:** If development delays occur, adjust the testing schedule. Set realistic expectations by discussing possible defects upfront, so users are prepared and avoid blaming the testing phase.
- **Milestone Identification:** Identify key milestones and align test dates with development milestones to avoid blaming the testing team for delays.
- **Test Start Timing:** Test dates should be based on the actual delivery of the development build, not fixed dates, ensuring testing begins as soon as the build is delivered, even with delays.

17. Planning Risks and Contingencies

This section outlines potential risks in the testing process and contingency plans to address them.

Potential Risks:

- **Lack of Personnel:** Insufficient testers or unavailability of team members.
- **Resource Unavailability:** Missing hardware, software, data, or tools for testing.
- **Delayed Deliveries:** Delays in receiving necessary components or tools.
- **Training Delays:** Delay in providing training for the application or testing tools.
- **Changes in Requirements/Designs:** Alterations to requirements or designs after testing begins.

Contingency Actions:

- **Requirements Changes:** Adjust the schedule if requirements change, possibly reducing tests or shifting deadlines.
- **Test Schedule Adjustment:** Shift test and development schedules if delays occur, but fixed dates may limit flexibility.
- **Acceptable Defects:** Increase the number of acceptable defects if needed, but this may affect product quality.
- **Resource Augmentation:** Add more testers or work overtime, though this could impact team morale.
- **Scope Reduction:** Reduce the scope of testing or optimize resources, but avoid this to maintain quality.
- **Extreme Measures:** In rare cases, testing may stop, but this should be avoided.

The key takeaway is that testing must never be eliminated, as it is essential for ensuring product quality.

18. Approvals

Project Sponsor:

- **NADRA**
  - Responsible for overall project approval and ensuring alignment with organizational goals.

Development Management:

- **Manager**
  - Responsible for approving technical development processes and ensuring resources are allocated effectively.

Test Team:

- **Syed Hussain Murtaza**
- **Zaim Khan Abbasi**
- **Hammad Ali**
  - Testers responsible for executing test cases, tracking defects, and reporting the progress of the testing phase.