Test Plan Document

#### Project Name: Soar

#### Version: V 17.1.1

#### Author(s): Vikas Sangwan

#### Date: 21/11/2024

## 1. Introduction

**Purpose**: To ensure functionality, security, and robustness of the following user stories:

1. Registered User Login From a New Device
2. Individual Investor - Upgrade To Premium
3. Approvals Notifications | SMS

**Scope**: Outline the scope of testing, including features to be tested and not tested.

**Features to be Tested:**

1. **Registered User Login From a New Device:**
   * Successful login flow from a new device.
   * Input validation for login credentials and OTP.
   * Notification for successful login via SMS.
   * Logout functionality for all other devices.
   * Redirection to the home page after successful login.
2. **Individual Investor - Upgrade to Premium**:
   * Option selection for premium account eligibility criteria.
   * Document upload functionality.
   * Notifications for document upload success.
   * Compliance team's ability to review and approve requests.
   * Notification to the relationship manager regarding request status.
3. **Approval Notifications | SMS:**
   * SMS notifications for loan approval.
   * SMS notifications for loan rejection with specified reasons.
   * Communication process for rejection reasons through the relationship manager.
4. **Departments Approval:**
   * Loan request review and approval functionality for department staff.
   * Visibility of department approvals with staff names and timestamps.
5. **Corporate Investor - Commercial Registration Verification:**
   * Integration with third-party service [X] for commercial registration.
   * Verification of valid and invalid registration numbers.
   * Handling of invalid commercial registration data and process restart.
   * Validation of company start date (less than or more than two years).

#### Features Not to be Tested:

1. **Unspecified functionalities:**
   * Features not mentioned in the acceptance criteria of the user stories.
2. **Third-party service integration failures:**
   * Failures on the part of third-party service [X] beyond the application's ability to handle.
3. **System-wide performance and load testing:**
   * Performance aspects not directly related to the user stories.
4. **Advanced edge cases:**
   * Scenarios not outlined in the user stories, such as extreme edge-case validations or non-functional requirements like usability or accessibility.

**Objective:** The primary goal of the testing efforts is to ensure the seamless functionality, reliability, and security of the system in alignment with the provided user stories. The testing aims to:

1. Validate that all critical user functionalities, including login, account upgrades, notifications, approvals, and third-party integrations, work as intended under specified conditions.
2. Identify and address potential functional, security, and logical issues that could impact user experience or system integrity.
3. Confirm compliance with business requirements by verifying acceptance criteria for each user story.
4. Ensure robust error handling, notification mechanisms, and accurate data processing across all modules.
5. Provide confidence in the system’s readiness for deployment and usability by end users.

**Test Items:**

The testing efforts will focus on the following items, modules, and features:

1. **Registered User Login From a New Device**:
   * Login page functionality.
   * Credential input fields (phone number and password).
   * OTP validation and input.
   * Notification system for successful login (SMS).
   * Logout mechanism for all other logged-in devices.
   * Redirection to the home page after login.
2. **Individual Investor - Upgrade to Premium**:
   * "Upgrade to Premium" functionality.
   * Eligibility criteria selection:
     + Asset worth verification.
     + Employment history in the financial sector.
     + Financial certification validation.
   * Document upload process.
   * Compliance team’s review and approval interface.
   * Notifications for document upload success and status updates.
   * Communication flow to the relationship manager.
3. **Approval Notifications | SMS**:
   * Loan approval notification system via SMS.
   * Loan rejection notification system via SMS, including reasons for rejection.
   * Communication mechanism for rejection reasons through the relationship manager.
4. **Departments Approval**:
   * Loan request review functionality for department staff.
   * Approval workflow within the department.
   * Display of department approvals, including staff details and timestamps.
5. **Corporate Investor - Commercial Registration Verification**:
   * Integration with third-party service [X].
   * Validation of commercial registration number (valid and invalid scenarios).
   * Restart of verification process for invalid inputs.
   * Validation of company start date (less than or more than two years).
   * Notification for ineligible companies based on start date.
6. **Common Features Across Modules**:
   * Error handling and input validation across all forms and fields.
   * Notification systems (SMS and system alerts).
   * User interface flow and consistency.
   * Security checks for sensitive operations like login and document uploads.

## 3. Test Approach

**Testing Levels:**

1. **Unit Testing**:
   * Verify individual components such as input validation, document upload, and notification generation.
   * Performed by developers to ensure code-level functionality.
2. **Integration Testing**:
   * Test interactions between modules like login/logout workflows, premium upgrade processes, and third-party API integration ([X] for commercial registration).
   * Focus on data flow between modules and subsystems.
3. **System Testing**:
   * Validate end-to-end scenarios, ensuring the complete system meets functional requirements.
   * Includes testing user stories like loan request approvals and investor account upgrades.
4. **Regression Testing**:
   * Ensure new changes or fixes do not negatively impact existing functionalities.
5. **User Acceptance Testing (UAT)**:
   * Conduct tests with end users or stakeholders to confirm the system aligns with business requirements.

**Testing Types:**

1. **Functional Testing**:
   * Ensure core functionalities such as login, account upgrade, and notifications work as expected.
2. **Non-Functional Testing**:
   * Validate usability, performance, and reliability.
3. **Performance Testing**:
   * Conduct load and stress testing on endpoints like /client\_register and /client\_login.
4. **Security Testing**:
   * Check for vulnerabilities in APIs (/client\_registration, /client\_login) and sensitive user operations like login and document uploads.
5. **Usability Testing**:
   * Evaluate user interface elements, ensuring seamless navigation and intuitive design.

**Testing Techniques:**

1. **Black Box Testing**:
   * Focus on inputs and expected outputs without considering internal code structure.
   * Suitable for testing user stories and workflows.
2. **White Box Testing**:
   * Examine internal code logic and paths, primarily during unit and integration testing.
3. **Exploratory Testing**:
   * Perform ad-hoc testing to discover potential issues in under-specified areas.

**Test Tools:**

1. **Automation Tools**:
   * **Selenium**: For web application end-to-end automation.
   * **Appium**: For mobile application automation.
   * **Cucumbre**: For BDD
2. **Performance Testing Tools**:
   * **JMeter**: For load and stress testing.
   * **Apptim**: For mobile app performance analysis.
3. **Test Management Tools**:
   * **Jira**: For test case management and tracking.
4. **Development Tools**:
   * **Postman**: For API testing and validation.

## 4. Test Environment

**Hardware Requirements:**

1. **Servers**:
   * A dedicated server for hosting the application and APIs during testing.
   * Hardware configuration for the server:
     + CPU: Quad-core (3.0 GHz or higher).
     + RAM: Minimum 16 GB.
     + Storage: Minimum 500 GB SSD.
   * Test server should mirror the production environment as closely as possible.
2. **Devices**:
   * Desktop/Laptop for web testing:
     + Windows (Intel i5/i7) or Mac (M1/M2 or equivalent) systems.
   * Mobile devices for app testing:
     + Android devices (Android 10 or higher).
     + iOS devices (iOS 14 or higher).
   * Virtualization tools for simulating additional devices (e.g., Android Studio Emulator, iOS Simulator).

**Software Requirements:**

1. **Operating Systems**:
   * Windows 10/11, macOS (latest stable version), and Linux (Ubuntu 20.04 or higher).
2. **Browsers**:
   * Google Chrome (latest stable version).
   * Mozilla Firefox (latest stable version).
   * Safari (for macOS/iOS users).
   * Microsoft Edge (latest stable version).
3. **Testing Tools**:
   * Selenium, Appium, JMeter, OWASP ZAP, and Postman as listed in the test tools section.
4. **Databases**:
   * The test database should replicate the production database structure.
   * Preferred DBMS: MySQL or PostgreSQL (based on the application stack).
5. **Third-party Services**:
   * Integration with the third-party service [X] for corporate investor commercial registration verification.

**Test Data:**

1. **Preparation**:
   * Test data will be created based on the user stories and acceptance criteria.
   * Examples include:
     + Valid and invalid login credentials.
     + OTP values (mocked for testing purposes).
     + Documents for account upgrade verification.
     + Loan request datasets covering different scenarios (approval, rejection, and multi-department reviews).
     + Commercial registration numbers (valid, invalid, and boundary conditions).
2. **Data Storage**:
   * Use of test databases or sandbox environments to isolate test data from production systems.
3. **Security Considerations**:
   * Ensure all test data complies with data protection regulations (e.g., GDPR, if applicable).
   * Avoid using real user data; instead, use anonymized or synthetic datasets.

## 5. Test Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Activities | Timeline | Dependencies |
| Requirement Analysis | - Review user stories, acceptance criteria, and business requirements. - Identify test scope. | Day 1–Day 2 | Finalization of requirements. |
| Test Planning | - Create test plan. - Identify risks and dependencies. - Prepare test strategy and environment. | Day 3–Day 4 | Completion of requirement analysis. |
| Test Environment Setup | - Configure test hardware and software environments. - Install necessary tools and frameworks. | Day 5–Day 6 | Availability of hardware and software resources. |
| Test Case Design | - Write detailed test cases for functional, integration, and performance testing. - Design test data. | Day 7–Day 9 | Completion of test planning and requirements. |
| Unit Testing (by Dev Team) | - Validate individual components at the code level. - Fix defects found during this phase. | Day 8–Day 10 | Code development completion. |
| Integration Testing | - Validate interactions between components. - Test API integrations (e.g., third-party [X]). | Day 11–Day 13 | Unit testing completion. |
| System Testing | - Conduct end-to-end tests for user stories. - Validate functional and non-functional requirements. | Day 14–Day 17 | Integration testing completion. |
| Performance Testing | - Perform load, stress, and BDD testing on critical endpoints and workflows. - Analyze results. | Day 18–Day 19 | System testing readiness. |
| Security Testing | - Validate APIs (/client\_registration and /client\_login) for vulnerabilities. - Test sensitive operations (e.g., login, uploads). | Day 20–Day 21 | System testing readiness. |
| User Acceptance Testing | - Conduct testing with stakeholders and end users. - Ensure requirements meet business needs. | Day 22–Day 23 | Completion of functional and non-functional tests. |
| Defect Retesting and Closure | - Retest and verify fixes for identified defects. - Close all test cases. | Day 24–Day 25 | Completion of all previous test phases. |
| Test Sign-Off | - Deliver test results and reports. - Obtain stakeholder approval for production readiness. | Day 26 | Successful completion of all testing phases. |

## 6. Test Deliverables

**Documents:**

1. **Test Plan**:
   * A comprehensive document outlining the scope, objectives, approach, and resources for testing activities.
2. **Test Cases**:
   * Detailed test cases covering functional, non-functional, performance, and security aspects based on user stories and acceptance criteria.
3. **Test Scripts**:
   * Automated scripts for:
     + Web testing (Selenium-based scripts for E2E scenarios).
     + Mobile testing (Appium scripts for the Wikipedia app).
     + API testing (Postman collections and security validation).
     + Performance testing (JMeter and BDD scripts).
4. **Bug Reports**:
   * Detailed reports for all identified issues, including severity, steps to reproduce, expected vs. actual results, and resolution status.
5. **Test Summary Report**:
   * A final report summarizing all testing activities, outcomes, defect trends, and recommendations for release readiness.

**Reports:**

1. **Daily Status Reports**:
   * Progress on executed test cases.
   * Details of defects found, resolved, and pending.
   * Any blockers or risks identified.
2. **Weekly Status Reports**:
   * Cumulative summary of testing activities.
   * Key milestones achieved.
   * Defect trends and progress toward resolution.
   * Planned activities for the next week.
3. **Defect Logs**:
   * Centralized tracking of all defects found during testing phases.
   * Includes priority, severity, status, and resolution timelines.
4. **Performance Test Results**:
   * Reports from JMeter and Apptim with metrics like response times, throughput, and system performance under load/stress.
5. **Security Test Results**:
   * Findings from vulnerability assessments (e.g., OWASP ZAP) and risk analysis for APIs and sensitive operations.

## 7. Roles and Responsibilities

|  |  |  |
| --- | --- | --- |
| Role | Name | Responsibility |
| Test Manager | Vikas Sangwan | Managing overall testing activities |
| Test Lead | - | - |
| Test Engineer | - | - |

## 8. Test Criteria

**Entry Criteria:**

1. **Code Freeze**:
   * All development tasks are completed, and the codebase is stable.
2. **Test Environment Availability**:
   * Test environments are fully set up and configured, mirroring production conditions.
3. **Test Data Preparation**:
   * Test data is created, verified, and ready for use.
4. **Requirement and Design Sign-Off**:
   * User stories and acceptance criteria are finalized and approved.
5. **Unit Testing Completion**:
   * Developers have completed and verified unit testing for individual modules.
6. **Test Cases Approval**:
   * All test cases have been reviewed and approved by stakeholders.

**Exit Criteria:**

1. **Test Case Execution**:
   * At least 95% of planned test cases are executed.
   * All critical and high-priority test cases are passed.
2. **Defect Closure**:
   * No critical or high-severity defects remain open.
   * All medium and low-severity defects are either resolved or have acceptable workarounds.
3. **Performance Metrics**:
   * Performance testing meets the defined benchmarks for response time, throughput, and error rate.
4. **Security Validation**:
   * All identified vulnerabilities have been addressed or documented with mitigation plans.
5. **UAT Completion**:
   * User Acceptance Testing is successfully completed with stakeholder approval.
6. **Documentation Completion**:
   * Test summary reports, defect logs, and all other deliverables are completed and shared with stakeholders.

## 9. Risk Management

**Risks:**

1. **Resource Unavailability**:
   * **Risk**: Key team members or required hardware/software resources may not be available during testing.
   * **Mitigation**:
     + Maintain a resource backup plan with cross-trained team members.
     + Ensure early procurement and setup of hardware and software resources.
2. **Tight Deadlines**:
   * **Risk**: Insufficient time for thorough testing due to project deadlines.
   * **Mitigation**:
     + Prioritize test cases based on risk and criticality.
     + Use automation for repetitive tasks to save time.
     + Communicate the impact of reduced timelines to stakeholders early.
3. **Incomplete Requirements**:
   * **Risk**: Ambiguities or changes in user stories and acceptance criteria may impact test planning.
   * **Mitigation**:
     + Collaborate with stakeholders for clarification during requirement analysis.
     + Use exploratory testing to uncover potential gaps or edge cases.
4. **Defect Leakage to Production**:
   * **Risk**: Critical defects might remain undetected during testing.
   * **Mitigation**:
     + Perform risk-based testing to focus on high-impact areas.
     + Include thorough regression and integration testing cycles.
5. **Third-party Dependencies**:
   * **Risk**: Integration with third-party services (e.g., [X] for commercial registration) may fail or cause delays.
   * **Mitigation**:
     + Use stubs or mock services during initial testing phases.
     + Establish clear communication channels with third-party providers for timely resolution.
6. **Test Environment Issues**:
   * **Risk**: Unstable or incomplete test environments may delay testing efforts.
   * **Mitigation**:
     + Set up environments early and perform environment validation before test execution.
     + Maintain backup environments for critical tests.
7. **Inadequate Test Data**:
   * **Risk**: Lack of realistic or comprehensive test data may lead to incomplete testing.
   * **Mitigation**:
     + Generate diverse test data during the test planning phase.
     + Use data generation tools to simulate various test scenarios.
8. **Stakeholder Unavailability**:
   * **Risk**: Delayed approvals or feedback from stakeholders can impact testing progress.
   * **Mitigation**:
     + Schedule regular check-ins and reviews to ensure stakeholder involvement.
     + Maintain clear documentation to facilitate quick decision-making.
9. **Automation Challenges**:
   * **Risk**: Flaky automation scripts or tool limitations may cause false positives/negatives.
   * **Mitigation**:
     + Regularly update and debug automation scripts.
     + Choose robust and reliable tools suitable for the application’s tech stack.
10. **Security Vulnerabilities**:
    * **Risk**: Security vulnerabilities in APIs or workflows may remain undetected.
    * **Mitigation**:
      + Conduct security assessments using tools like OWASP ZAP.
      + Follow secure coding practices and review results with the development team.

## 10. Defect Management

**Defect States:**

1. **New**:
   * The defect is newly reported and not yet reviewed.
   * Logged with complete details such as steps to reproduce, severity, priority, screenshots/logs, and environment details.
2. **Assigned**:
   * The defect is reviewed and assigned to the responsible developer or team for resolution.
3. **In Progress**:
   * The assigned developer is actively working on fixing the defect.
4. **Resolved**:
   * The defect has been fixed, and the fix is ready for verification by the testing team.
5. **Retested**:
   * The testing team has re-executed the test cases related to the defect to confirm the fix.
   * If the defect persists, it is reopened.
6. **Closed**:
   * The defect is verified as fixed and no longer exists in the system.
   * Closed only after confirmation from the testing team or stakeholders.
7. **Reopened**:
   * If the defect is not resolved after the fix, it is moved back to the "Assigned" or "In Progress" state.
8. **Deferred**:
   * The defect is acknowledged but postponed for a future release due to low priority, limited impact, or resource constraints.
9. **Duplicate**:
   * The defect is identified as a duplicate of another defect and is marked accordingly.
10. **Rejected**:
    * The defect is invalid, not reproducible, or does not match the requirements or specifications.

**Defect Tracking Tool**

* **Tools**:
  + **Jira**: Comprehensive defect tracking and project management tool, suitable for agile teams.

## 11. Approval

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Designation | Signature | Date |
|  |  |  |  |