Test Strategy

Recipes

Revision History

Date	Version	Author	Description

1. Overview

This is the Test Strategy for the mobile application Recipes. This document shall be completed and used by the project test team to guide how testing will be managed for this project. This is a living document that may be refined as the project progresses.

Testing will cover the Functional and Non-functional testing of the mobile application Recipes. Functionality for this release is detailed in the Requirements specifications documents.

2. Test Approach

The objective of system testing is to verify the correctness of the newly designed items, and their interaction with the existing functions. Testing will focus on functionality of the mobile application The Recipes.

Testing will be accomplished through an organized testing process that will have repeatable tests. This process will be accomplished by use of the test cases created to match the requirements being developed for the mobile application.

Planning the execution of test cases for new functionality and regression tests will be done in coordination with the plan for developing. Testing and development will be executed in parallel, based on phased implementations, wherever possible.

Test cases will be structured to give a full range of coverage to the converted functions in both a Positive and Negative fashion, simulating what a potentially unfamiliar user might do during use. Positive test cases will reflect that the application functions as expected and described in the Requirements Specifications. Negative test cases are tests that exercise the limits and boundaries outside the expected designs. The results of this testing will give us some idea as to the stability for the application and its components.

Additional testing beyond the created tests may be done where feasible to exercise the application to verify error handling and system recovery due to incorrect data or entry into fields.

3. Test Levels

During application testing, Unit testing, Integration testing, System testing and User Acceptance Testing will be performed.

3.1 Unit testing

Unit testing is testing performed to determine that individual program modules perform per the design specifications.

Owners:Dev Team

3.2 Integration testing

Integration testing will consist of the following:

- Verifying links between internal application components.
- Focusing on complete end-to-end processing of programs, threads, and transactions.
- Boundary value analysis (testing modules by supplying input values within, at, and beyond the specified boundaries).
- Cause-effect testing (supplying input values to cause all possible output values to occur).
- Ensuring traceability to requirements, use cases, user interface (UI) design, and test objectives.
- Testing each business function end-to-end through the application, including positive and negative tests.

Owners Dev Team/ QA Team

3.3. System testing

System testing is the process of testing an integrated system to verify that it meets specified requirements. This testing will determine if the results generated by information systems and their components are accurate and that the system performs according to specifications.
Owners: QATeam

3.4 User Acceptance testing

UAT will be conducted to gain acceptance of all functionality from the user community. UAT will verify that the system meets user requirements as specified.

Owners: QA Team

4. Test Types

4.3 Functional testing

For functional testing, the following testing techniques will be used:

- Equivalence Partitioning
- Boundary value analysis
- Error guessing
- State Transition diagrams

- Cause / effect

Also exploratory and ad-hoc testing techniques will be used.

4.2. Non-functional testing

Non functional testing will include:

- Installation testing
- Compatibility testing
- Interruptions testing
- Performance/Load testing
- Security testing
- UI/UX testing

4.3. Change related testing

4.3.1 Smoke testing

Smoke testing will be performed by the QA Team for every new build of the application to determine whether the deployed software build is stable or not.

4.3.2. Regression testing

The QA Team will do a pass through all the test cases that were developed for this project. This will encompass the re-testing of each item in each test case as well as the re-verification of each repaired defect that is decided on as an items to be regressed based on the severity of the defect and the knowledge of the the mobile application Recipes development staff as to which areas of the application are the most volatile due to new features being implemented.

The Recipes QA Team will perform as many test cases as is feasible. Testing of the application will include limits and boundaries. The functional testing of the application will be covered this way. Positive test cases will reflect that the application functions as expected. Negative test cases are tests that exercise the limits and boundaries outside the expected designs.

4.3.3. Sanity Testing

Sanity testing will be performed by QA Team after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes.

5. Environment requirements

The application will be tested on the following operating systems: Android 10, 11, MIUI, One UI

Approved devices for testing:

- Xiaomi Redmi Note 8 Pro, MIUI Global 12.5.1 Stable, Android 11, 2340x1080
- Samsung Galaxy A20, version One 2.0 UI, Android 10, 720x1560

Testing will also be carried out using emulators on devices:

- Xiaomi Redmi 9
- Samsung Galaxy A50
- Huawei P30 lite
- Google Pixel 3A

6. Testing tools

Process	Tool
Test case creation	Qase TMS
Test case tracking	Qase TMS
Test case execution	Manual
Test case management	Qase TMS
Bug tracking system	Jira
APK Builder	Android Studio
Log monitoring	ADB
API testing	Postman
Debugging proxy server tool	Charles Proxy

Server-side Performance Testing	Jmeter
Client/Device-side Performance Testing	Apptim or BlazeMeter
Debug Tool	DevTools
Security Testing Tool	LGTM

7. Test deliverables

No.	Deliverable
1	Test Strategy
2	Test Plan
3	Requirement Traceability Matrix
4	Test Cases
5	Test Summary Report

8. Testing metrics

Base Metrics:

- Total number of test cases
- Number of test cases passed
- Number of test cases failed
- Number of test cases blocked
- Number of defects found
- Number of defects accepted
- Number of defects rejected
- Number of defects deferred
- Number of critical defects

- Number of planned test hours
- Number of actual test hours

Calculated Metrics:

- Passed Test Cases Percentage
- Failed Test Cases Percentage
- Blocked Test Cases Percentage
- Fixed Defects Percentage
- Accepted Defects Percentage
- Defects Rejected Percentage
- Defects Deferred Percentage
- Critical Defects Percentage

Test effort Metrics:

- Number of test run per time period
- Number of bugs per test

Test coverage:

- Test Execution Coverage Percentage

9. Risk Analysis

Potential risks during testing:

- Lack of personnel resources when testing is to begin
- Lack of required hardware, software, data or tools.
- Changes to the original requirements or designs.
- The number of tests performed will be reduced.
- The number of defects allowed will be increased.
- The scope of the plan is subject to change.

10. Reporting tool

JIRA will be the reporting tool for this project.