**Testing Plan**

**<Internet Banking System>**

**QA TRACK ITI**

**SMART VILLAGE ITI**

**GIZA, SMART VILLAGE 12511**

**07/05/2023**

Contents

[**Introduction:** 2](#_Toc134318098)

[**Objective of Testing:** 2](#_Toc134318099)

[**Scope of Testing:** 2](#_Toc134318100)

[**-In Scope:** 2](#_Toc134318101)

[**-Out Of Scope:** 2](#_Toc134318102)

[**Testing Levels:** 2](#_Toc134318103)

[**Testing Types:** 2](#_Toc134318104)

[**Testing Environment:** 2](#_Toc134318105)

[**Entry Criteria:** 2](#_Toc134318106)

[**-For Write Test Cases:** 2](#_Toc134318107)

[**-For Executing Test Cases:** 2](#_Toc134318108)

[**Exit Criteria:** 3](#_Toc134318109)

[**-From Write Test Cases:** 3](#_Toc134318110)

[**-From Executing Test Cases:** 3](#_Toc134318111)

[**Test Case Lifecycle:** 3](#_Toc134318112)

[**Review Strategy:** 3](#_Toc134318113)

# **Introduction:**

The purpose of this testing plan is to ensure that the internet banking system meets all of the required specifications and functions as intended. The testing plan will identify the scope of testing, test levels, objective, entry & exit criteria, test case lifecycle, test types, and the testing environment. The goal of this plan is to ensure that the system is reliable, functional, and user-friendly.

# **Objective of Testing:**

* Ensure the system under test conforms to functional software requirements specification (SRS).
* Bugs/issues are identified and fixed before go live.

# **Scope of Testing:**

## **-In Scope:**

Defines the features, functional requirements of the software that will be tested

## **-Out Of Scope:**

Nonfunctional requirements and anything else outer from features in software requirements specification (SRS).

Unit Testing.

Integration Testing.

# **Testing Levels:**

**System testing:** verify that the system is meeting all of the specified functional requirements

And we will not tested any interaction or integration with other systems.

# **Testing Types:**

**Functional Testing:** to ensure that the system is functioning as intended and that it meets all of the specified requirements.

# **Testing Tools**

**The following tools will be used for testing:**

* Excel sheets for test cases and bug reports writing
* Git and GitHub version control system
* TFS for project management

# **Testing Environment:**

* Windows 10 and above.
* Chrome version 106.0.5249.103 and above.
* Running Application using Chrome, Xampp (v3.3.0)

# **Entry Criteria:**

## **-For Write Test Cases:**

SRS is reviewed and approved.

Design Document is reviewed and approved.

## **-For Executing Test Cases:**

Code is reviewed and approved.

Test cases are reviewed and approved.

# **Exit Criteria:**

## **-From Write Test Cases:**

* Writing test cases, finishing reviewing, approving and documentation of test cases document.
* Each requirement will be linked with at least one test case.

## **-From Executing Test Cases:**

* Execute all test cases written in test case report.
* Finishing bug report for any bug found within executing.

# **Test Case Lifecycle:**

## **-Test Case Generation:**

In this phase, testing team documents the test cases.

## **-Test Case Documentation:**

1- Testers will write test cases report in excel file on drive and this file will uploaded daily to our GitHub repository this sheet include these parameters (Test Case ID, Name, Description, steps, Expected Result, Actual result, Designed By, Status)

2- Excel file for test cases separated into sheets every sheet named by one feature and include test cases relate this feature.

3- Tester will write ID manually depend on feature’s name and number of test cases.

## **-Test Case Review Strategy:**

Test cases that are prepared can be moved to other team member to review the test cases,

And may be review by coach or team’s assessor and write Review comments

## **-Rework:**

Test cases with Review comments are reworked and follow test case lifecycle steps again.

## **-Test Case Finalization:**

Upon successful review, test cases are ready to execute. This is an important step. Once test cases are finalized, we can version the test case artifacts and baseline them.

# **Bug Lifecycle:**

**New:** When a bug is reported for the first time.

**Assigned:** When the bug is assigned to a developer or tester.

**In Progress:** When the developer or tester starts working on the bug.

**Fixed:** When the bug has been fixed by the developer.

**Retest:** When the fixed bug is ready for retesting.

**Verified:** When the bug fix has been verified and confirmed.

**Closed:** When the bug is considered resolved and closed.

## **-Criteria for Opening a Bug:**

**Clear description:** A bug report should have a concise and detailed description of the issue, including steps to reproduce and expected results.

**Reproducibility:** The bug should be reproducible consistently to help identify and resolve the issue.

**Supporting materials:** Include any relevant screenshots, logs, or additional information that can help the development team understand and reproduce the bug.

## **-Bug Report Contents:**

**ID:** unique for bug.

**Title:** A brief and descriptive title for the bug report.

**Description:** Detailed information about the bug, including steps to reproduce, observed behavior, and expected behavior.

**Preconditions:** if needed

**Environment details:** Mention the platform, operating system, device, and software versions on which the bug occurred.

**Attachments:** Include any relevant screenshots, logs, or files that can aid in understanding and resolving the bug.

**Bug assignment:** Assign the bug to the respective developer or team responsible for its resolution.

**Status updates:** Track the bug's life cycle, including changes in status, comments, and progress updates.

## **-Bug Fixing and Verification:**

Once a bug has been reported and assigned to a developer, the bug fixing and verification process begins. This section outlines the steps involved in resolving the reported bugs and ensuring their successful closure.

## **Bug Fixing Process:**

**1-Developer Analysis:** The assigned developer analyzes the bug report, reproduces the issue, and identifies the root cause of the bug.

**2-Bug Fixing:** The developer makes the necessary code changes or fixes to address the reported bug.

**3-Code Review:** The fixed code undergoes a code review process to ensure quality and adherence to coding standards.

**4-Compilation and Build:** The fixed code is compiled, and a new build or version of the software is created.

**5-Confirmation Testing:**

The developer performs confirmation testing on the fixed bug to verify that the issue has been resolved.

## **Bug Verification Process:**

**1-Regression Testing:**

The fixed bug undergoes regression testing, which involves retesting the affected functionality to ensure that the fix did not introduce any new issues.

**2-Confirmation Testing:**

By Testers: Testers also perform confirmation testing on the fixed bug to validate that the issue has indeed been resolved.

**3-Verification Criteria:**

The bug is considered successfully fixed and can be closed if the following criteria are met:

* The bug no longer exists or behaves as expected.
* The fix does not introduce new bugs or adversely affect other parts of the system.
* The bug fix has been confirmed and verified by both developers and testers.
* The fix has been tested in different environments (if applicable) to ensure compatibility.

During the confirmation and regression testing stages, it is crucial to document and track the test cases used, any issues encountered, and their resolutions. This helps ensure thorough testing and provides a comprehensive overview of the bug fixing and verification process.