Test Plan for Taka App

Version History:

Ver. #	Date of Release	Changes	Author	Reviewed / Approved by
1.0.0	2023/09/01	Initial version	Enisha Ashrefa, Nigah Hossain Anika	Enisha Ashrefa
1.0.1	2023/09/08	Scope of work and QA Scope of work file is update	Enisha Ashrefa, Nigah Hossain Anika	Enisha Ashrefa
1.0.2	2023/09/15	Scope of work and QA Scope of work file is update	Enisha Ashrefa, Nigah Hossain Anika	Enisha Ashrefa
1.0.3	2023/09/22	Scope of work and QA Scope of work file is update	Enisha Ashrefa, Nigah Hossain Anika	Enisha Ashrefa

Abbreviations

- GUI - Graphical User Interface
- OS - Operating System
- API - Application Programming Interface

Table of contents

1. Introduction

The project is a Mobile Financial Service which is named as “Taka”

There are several existing MFS in the market. Client wants to make this system similar to all those systems but customizing the needs of users. The programming language for this project is C#. Our company will develop this whole new system.

This document deals with the strategies, procedures, tools and resources which will be used to conduct tests on the “Taka” to ensure the product’s quality.

2. Test Objectives

Since we are going to evaluate the product’s quality and fulfillment, here are the main motive of the test plan :

- Verifying the leading features, such as registration, Money transfers, bill payments etc.
- Ensuring the security of user data and transactions.
- Checking the performance of the app response time.
- Checking the compatibility of the app with different mobile devices and operating systems.
- Identifying and fixing issues within the app.

3. Test Strategy

3.1 Strategy :

While testing the MFS, we have to consider some test strategies to develop test scenarios. The test strategies are such as,

- **UI Testing Strategy:**

  We have to check the UI components Whether they are placed as per the client’s Figma design or not.

- **Functional Testing Strategy:**

  Each module and their features need to be tested to ensure the functionalities of the product are working as expected.

- **Non-Functional Testing Strategy:**

  We have to check trying to log in with incorrect credentials to test authentication and authorization, test the system with the increase of the users and by testing for vulnerabilities such as SQL injection check the security and performance.

3.2 Test type :

We will incorporate different testing types to accomplish the test of our system and the types are following:

- **Unit testing:**

  We will conduct unit testing for the code base which will be basically done by the developers.

- **GUI Testing:**

  In GUI testing, we will test the components of the UI and their appearance on the screen. Such as buttons, images, links, input fields, dropdown, etc. Testing user inputs by providing valid and invalid values in text fields, dropdowns, and checkboxes. Checking if error messages are displayed for invalid inputs. Evaluating the navigation of each page and grammatical errors

- **API Testing:**

  In API testing, we have to test all the API related functionalities of modules to ensure the easy api services. API testing is to ensure that the backend functionalities of the app work as intended and that the communication between the app and the server is secure and reliable.

- **Pixel Perfection:**

  Here we will test how the components integrate with different devices. How they look in different screen sizes. How they behave on individual operating systems, etc. Additionally, during testing, many details must be reviewed, for instance: Elements alignment. Image's location and quality. Icon's positioning.

- **Integration Testing:**

  Testing will be done integrating modules together to test the data flow.

- **System Testing:**

  The system will be tested as a whole to ensure the consistency with each module and features.

- **Retesting:**

  After the fixation of bugs, retesting will be done to check if it has really been fixed.

- **Regression Testing:**

  If any module or feature modification has happened, then the regression testing will be conducted.

4. Modules/Features to be tested

Phase	Modules / Features
1	1. Partial UI of the Landing Page, Registration module(Log In / Sign Up) 2. Login mechanism 3. Sign mechanism
2	1. Complete UI of the Landing Page, Registration module(Log In/Sign Up), Cash In, Cash Out, Pay bill, Mobile Recharge, Send Money 2. Cash In, Cash out mechanism 3. Pay bill feature
3	Mobile Recharge, Send money features
4	1. API Integration to features such as Pay bill(Payment gateway), Mobile recharge (Telecom Services), Send money, Cash In etc. 2. Balance check 3. Notification message

5. Features not to be tested

- Add Money
- Savings
- Loan
- Remittance
- Any hardware failure will not be tested, because this is the client 's responsibility
- Since regression testing takes much time, after bug fixing there won't be any regression test
- Network variability Test won't be done

6. Test Estimation

Factors Affecting Test Estimation:

- Product Scope and Size
- Problem Domain Complexity
- Documentation Requirements
- Time Constraints
- Defect Count and Rework

- Regression Testing

7. Release Procedure

To Release the product, we will follow below procedures:

- **Step 1: Requirement Analysis**

  Analyzing project requirements to establish a clear understanding of scope and objectives.

- **Step 2: Development and Internal Release for QA**

  Beginning development based on the analyzed requirements. Create an internal release for Quality Assurance (QA) testing.

- **Step 3: QA Testing and Bug Reporting**

  The QA team rigorously will test the internal release. Identify and report any issues or bugs encountered during testing.

- **Step 4: Bug Fixing by Developers**

  Development team will address and fix reported bugs.

- **Step 5: Final Release for the Current Phase**

  We will prepare and release the final version of the app for the current development phase.

- **Step 6: QA Verification of Bug Fixes**

  QA will validate that the reported bugs from the internal release have been successfully fixed in the final release.

- **Step 7: Comprehensive Phase Testing and Bug Recording**

  The QA team will perform comprehensive testing of the entire phase, recording any remaining bugs or issues.

- **Step 8: Release Evaluation**

  QA team will assess the status of the project: if there are no blocking issues and low-severity bugs remain, proceed with the release. Otherwise, the release will be postponed until issues are resolved.

- **Step 9: Future Phase Bug Resolution**

Any outstanding bugs that were not addressed for the current phase will be prioritized for resolution in the next development phase.

8. Test Suspension Criteria

Testing might get stop if any of these below events get encountered:

- Any showstopper bug found while testing.
- Bug has not been fixed or addressed in the respective release.
- The release has been made without the provision of release notes to provide essential information to users and stakeholders.

9. Test Acceptance Criteria:

Application functional, nonfunctional features and UI should not miss any requirement which client gave and also will be following the below criteria:

- Application does not have any significant bugs.
- Applications have 90% Test case coverage.
- Application does not have any security issue

10. QA Task List and Testing Process

QA Team will be conducting the following tasks:

- Requirement analysis and knowledge transfer
- Identify Test areas
- Test Case writing on identified test areas
- Prepare Test environment
- Test Cases Execution
- Bug reporting/retest
- Deliver Test report
- Perform Test closure activity
- Daily morning meeting
- Spec grooming meeting

11. Test Environment

To set-up the test environment for **Taka** app, we are going to follow these hardware and software requirements:

Hardware Requirement:

- Smartphone
- iPad(Emulator)
- iPhone(Emulator)

Software Requirement:

Operating System: Android(Latest version), iOS(Latest Version)

Network Requirement:

Internet connectivity to mobile.

Tools to be used:

- **Test Case management**: Microsoft Excel/Google Drive, JIRA
- **Document management**: Google Drive
- **Project management**: Redmine, JIRA
- **Design Tool**: Figma
- **UI/Pixel Perfections Test Tool**: Browsers' Developers Tools.

12. Schedule

Within feature release per plan, schedule may get varied / updated:

Phase	Features / Modules	Test Iems	# of Test Cases	Internal Release	Final Release
1	1.Partial UI of the Landing Page, Registration module(Log In / Sign Up) 2. Login mechanism 3. Sign mechanism	3	120	2023-09-8	
2	1. Complete UI of the Landing Page, Registration module(Log In / Sign Up), Cah In, Cash Out, Pay bill, Mobile Recharge, Send Money	3	90	2023-9-25	

	2. Cash In, Cash out mechanism 3. Pay bill feature				
3	Mobile Recharge, Send money features	2	50	2023-10-27	
4	1. API Integration to features such as Pay bill(Payment gateway), Mobile recharge(Telecom Services), Send money, Cash In etc. 2. Balance check 3. Notification message	7	140	2023-10-2	
5	System testing(Blackbox) + Issue reporting + Retesting	1	230	2023-10-13	

13. QA Summary Report

Phase	**URL**
1st Release (V-1.0.0)	http://jira.com/jira/issues/first-issue
2nd Release (V-1.0.1)	http://jira.com/jira/issues/second-issue
3rd Release (V-1.0.2)	http://jira.com/jira/issues/third-issue
4th Release (V-1.0.3)	http://jira.com/jira/issues/fourth-issue

14. Resource Allocation

Recourse Name	**Roles and Responsibilities**
Enisha Ashrefa	Project Manager
XXX	Software Engineer

XXY	
XXZ	
Nigah Hossain	QA
XYZ	Customer

15. Risk

Identify potential risks that could impact the testing and overall project.

Data Security Breach

- The app deals with sensitive financial data. A security breach could lead to data compromise and financial losses.

Performance Degradation under Load

- High user load might lead to app slowdowns or crashes during peak usage times.

Lack of Usability

- Poor user experience due to complex user interface or confusing navigation.

Schedule:

- Failure to clear SRS as scheduled might hinder us from meeting the testing deadline.
- In case Releases for Testing aren't provided per the schedule, it could jeopardize meeting the deadline.
- Modifications to requirements/scope may potentially impact the test schedule.

Testing:

- The introduction of new features and function enhancements could inadvertently lead to side effects and disrupt the existing functionality due to inadequate testing time.
- To mitigate this, we need to establish a well-defined testing scope and allocate sufficient testing time.

Application Risk:

- The application's behavior might deviate from the norm, and core functions could malfunction on unsupported devices/interfaces.

- There's a possibility of unexpected behavior in the latest browsers if the application hasn't been adapted and thoroughly tested.

16. Contingency plans

Define specific actions to be taken if risks materialize.

Data Security Breach Contingency:

- Isolate affected systems and servers.
- Notify relevant stakeholders and users about the breach.
- Engage cybersecurity experts to investigate the breach and implement remediation measures.

Performance Degradation Contingency:

- Scale up server resources to handle the increased load.
- Notify users of potential performance issues and expected resolution time.
- Communicate with the development team to identify and address bottlenecks.

Lack of Usability Contingency:

- Prioritize and address critical usability issues immediately.
- Conduct emergency usability testing to identify pain points and possible quick fixes.
- Communicate with UX designers to implement changes that enhance usability.

17. Test Exit Criteria

Testing process of the **Taka** application will be ended if following criteria are met:

- Define the conditions that must be met for testing to be considered complete.
- Include criteria related to test case coverage, defect closure, and stability.

- All specified functions are functioning properly.
- Major bugs are identified, resolved and retested.
- All test cases are executed and passed.

18. Bug Status Explanation

The status of the bug is maintained as follows in our report on test execution:

New: The status of any defect or bug is logged in as To-do when it is discovered and verified. When a developer who has been assigned to the ticket is currently at work.

Done: This status denotes that the problem has been resolved, and further testing is permitted.

Verified: The bug is retested by the tester after the developer fixes it. If no software flaw is found, the flaw has been addressed, and the status has been changed to "verified."

Reopened: The tester updates the state to "reopened" if the bug still exists after the developer has fixed it. The bug goes through its life cycle once more.

Closed: When a bug is declared to be fixed, the tester marks it as "Closed."

Duplicate: The status is changed to "duplicate" if the defect occurs twice or relates to the same notion as the bug.

Rejected: When a developer determines that a flaw is not actually present, the defect is changed to "rejected."

Deferred: A bug is given the classification "Deferred" if it is not a top priority at the moment and is anticipated to be fixed in the upcoming version.

Not an error: The status of a problem is "Not a bug" if it has no impact on the application's operation.

19. Test Deliverables

Followings are the deliverables from QA for **Taka** project:

Test Deliverables before Testing

- Test Plan
- Test Cases
- Test Scripts

Test Deliverables after Testing

- Test execution report

20. Test Plan Approvals

Name	Role	Signature	Date
Enisha Ashrefa	Project Manager		2023/09/01
Enisha Ashrefa	Project Manager		2023/09/08

Enisha Ashrefa	Project Manager		2023/09/15
Enisha Ashrefa	Project Manager		2023/09/22

21. References

The following documents are used as sources of information for this test plan:

- System Requirement Specification_TAKA.doc
- Requirement Traceability Matrix_TAKA.xls