**Sprint 2 Test Plan**

**Version No: 2.0**

**Date: 27-11-2024**

**Project Name: Netomi**

**Copyright Notice**

This document contains proprietary information of Appinventiv Tech. No part of this document may be reproduced, stored, copied, or transmitted in any form or by means of electronic, mechanical, photocopying or otherwise, without the express consent of Appinventiv Tech. This document is intended for internal circulation only and not meant for external distribution.

**Revision History**

| **Version No** | **Date** | **Prepared by / Modified by** | **Significant Changes** | | **Reviewed and Approved By** | **Approved On** |
| --- | --- | --- | --- | --- | --- | --- |
| 2.0 | 27/11/2024 | Parul Sharma | | Initial Draft |  |  |

1. **Table of Contents**

[**1**](#_heading=h.49x2ik5) **Introduction 4**

[1.1](#_heading=h.3znysh7) Purpose 4

[1.2](#_heading=h.2et92p0) Objectives 4

[1.3](#_heading=h.tyjcwt) Scope

[1.4](#_heading=h.3dy6vkm) References 4

[**2**](#_heading=h.1t3h5sf) **Test Strategy 5**

[2.1](#_heading=h.4d34og8) Test Case Development 5

[2.2](#_heading=h.2s8eyo1) Testing process 6

[**3**](#_heading=h.17dp8vu) **Types of Testing Involved 7**

[**4**](#_heading=h.3rdcrjn) **Test Items 8**

[**5**](#_heading=h.26in1rg) **Test Item Linkage to Requirement / Design Specifications 9**

[5.1](#_heading=h.lnxbz9) Testable Items 9

[5.2](#_heading=h.1ksv4uv) Non Testable Items 9

[**6**](#_heading=h.z337ya) **Entry & Exit Criteria 11**

[6.1](#_heading=h.3j2qqm3) Build Suspension & Resumption Criteria: 11

[**7**](#_heading=h.1y810tw) **Test Execution Strategy 13**

[7.1](#_heading=h.4i7ojhp) Understanding Requirements: 13

[**8**](#_heading=h.2xcytpi) **Tools Utilized 15**

[**9**](#_heading=h.1ci93xb) **Test Environment 16**

[**10**](#_heading=h.3whwml4) **Item Pass Fail Criteria 17**

[**11**](#_heading=h.2bn6wsx) **Defect Analysis and Resolution 18**

[**12**](#_heading=h.qsh70q) **Test Deliverables 19**

[**13**](#_heading=h.3as4poj) **Risks, Assumptions & Constraints 20**

[**14**](#_heading=h.1pxezwc) **Training Requirements 21**

**15**  **Test Responsibilities 22**

# 1.Introduction

## Purpose

**Netomi** is a chat widget AI-driven, customer experience platform designed to enhance customer support and engagement by leveraging artificial intelligence. Its primary purpose is to help businesses provide fast, efficient, and personalized customer service while reducing operational costs.

## Objectives

The objective of the Test plan is to define the various Testing strategies and Testing tools used for complete Testing life cycle of this project.

## Scope

The document mainly targets the testing and validating data in report output as per Requirements Specifications provided by Client.

**1.3.1 Functions to be tested.**

1.Mobile Application

* User can create the account
* User can able to login
* User can tap on a menu option to launch the application
* User can view received text message
* User can send text message
* User can logout the Application

**1.3.2 Functions not to be tested.**

1. Typing Indication  
2. Session Management

## References

1. Guidelines provided by Client.
2. User Stories
3. Figma
4. Test Case Document
5. Basecamp Communication
6. Client Calls
7. Current Live System

# 2.Test Strategy

This section describes in detail the testing strategy and processes to be followed through the entire testing phase of Test.

Phase wise activities are as follows :

| **S.no** | **Phase** | **Description** |
| --- | --- | --- |
| 1. | Requirement analysis | 1. Through User stories and Designs 2. Through Client Call 3. Through Existing Live System |
| 2. | Test case development | 1. Test case writing on basis of user stories/designs 2. On basis of user experience 3. On basis of existing live system 4. On basis of internal discussion |
| 3. | Environment setup / Test Execution | 1. Test executions on QA/Staging server  2. Sprint wise Bug Reporting on JIRA  3. Test execution on Browsers and Mobile Device |
| 4 | Test Closure | 1. Sharing Test case execution report along with QA metrics 2. Retrospectives after each sprint 3. Grooming session for next sprint cycle |

## Test Case Development

To develop the Test Cases of respective features and functionalities present within Test, bottom-up approach would be applied.

* The entire application is broken down into a set of features and functionalities within Test functional requirements. These represent the planned units, respectively.
* Once the test cases for a module have been designed, they will be reviewed by the QA Lead/peer – depending on the bandwidth - and corresponding feedback will be shared across with respective resources.
* The testers will then incorporate the suggested feedback (if any) within the test cases and share back the same for a second review.
* In case of requirement of further modification, testers will update the test cases based on the suggestions received and the final draft of the test cases will thus be ready.
* This draft will be sent over to Test for review. If there are any modifications/additions suggested by Test.
* While in the Test execution phase, any Test Scenario or Test Case which is felt to be included into the complete test cases set, and is not part of the baseline version; QA team would author the test case and execute it to find any defect against the same.

## Testing Process

Testing will validate the implemented application against the specifications agreed upon for each of the functionalities.

* For any functionality, testing will be carried out at three levels, namely:
  + Unit Testing: By the Development team
  + Module Level Testing, Integration testing, Regression testing etc.: By the Testing Team
  + User Acceptance testing: By Testing Team.
* Regression testing of outstanding errors will be performed on an ongoing basis.
* The below mentioned would be considered to be the prerequisites for entering into a Test Cycle:
  + Code Freeze must be done by the development team for a release, after all the required functionality in the, to be, released version is complete and unit-tested.
  + Every release should be accepted with a release document that contains the modules covered in the release.
  + Application should pass all the sanity tests before actual execution.

Working on the optimum testing process, where test preparation commences as soon as the Requirements gathering are produced, will benefit from Quality initiatives throughout the project lifecycle.

# 3.Types of Testing Involved

This section is a critical input to the test developers. Identify the various types of testing specific or applicable for the project and elaborate all other below sections in the Test Plan specific to each testing type.

| **Test Group** | **Testing Coverage** |
| --- | --- |
| Unit Testing | **Unit Testing** is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output. In procedural programming a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. |
| Sanity Testing | All modules are available for Functionality Testing – testers can follow main positive and negative paths through the application to successfully execute all channels, menu items, and screens to continue the further testing |
| UI Testing | The content displayed on each screen is as specified in the UI guideline and BRD.  Screen elements are ordered, aligned, and formatted according to the wireframes, approved.  Navigation of the buttons is as per requirement & the wireframes, approved.  Display formatting (font, color, relative positioning) is consistent between all screens in the application and are mapping exactly the wireframes.  Aspects of each screen follow the user interface design approved as wireframes. |
| Functional Testing | Ensure that the complete functionality specified within requirements has been implemented. |
| Browser / OS Compatibility | Screens are rendered consistently across browsers on various operating system platforms specified within the requirement documents. |
| Negative Testing | Error handling will be covered in this test. Apart from negative test cases designed with negative test data. |
| Regression testing | This should be carried out for every release to ensure that:   * Defect verification is complete as per the defects logged on Jira for the release. * The new release is functional as earlier, testing all the possible impact areas which might get affected due to defect fixing.   This test will be performed for each new version of the system to detect unexpected impact resulting from the modifications. |
| Integration Testing | Integration testing would be done manually after each milestone to ensure that modules delivered in the previous milestone have been integrated successfully in the current milestone. Also, Integration testing would be done from 2nd milestone onwards, upon delivery of further milestones, as well as a complete system integration nearing the Milestone 3. |

# 4.Test Items

| **Test Item** | **Version** | **Description** |
| --- | --- | --- |
| **Mobile** | | |
| User can create the account | <2.0> | verification of scenarios related to create a new account |
| User can able to login | <2.0> | verification of scenarios related to login. |
| User can tap on a menu option to launch the application | <2.0> | verification of scenarios related to the app is getting launched once users tap on the menu option. |
| User can view received text message | <2.0> | verification of scenarios related to the user can receive the message. |
| User can send text message | <2.0> | verification of scenarios related to sending text messages. |
| User can logout the Application | <2.0> | verification of scenarios related to logout the application. |

# 5.Test Item Linkage to Requirement / Design Specifications

## 

## 5.1 Testable Items

| **Test Item** | **Item Requirements/Design Specification** |
| --- | --- |
| **Mobile** | |
| User can create the account | JIRA |
| User can able to login | JIRA |
| User can tap on a menu option to launch the application | JIRA |
| User can view received text message | JIRA |
| User can logout the Application | JIRA |

## 

## 5.2 Non Testable Items

| **Test Item** | **Item Requirements / Design Specification** | **Reason** |
| --- | --- | --- |
| Typing Indication |  | It will come in next sprint |
| Session Management |  | It will come in next sprint |

## 

## 5.3 Testing Coverage for test items

| **Test Item** | **Manual** | **Load** |
| --- | --- | --- |
| Mobile App | Yes | No |

# 6.Entry & Exit Criteria

| **Testing Type** | **Entry Criteria** | **Exit Criteria** |
| --- | --- | --- |
| Unit Testing | Testing may start, if the code has been reviewed successfully and all previous known vulnerabilities fixed if any | A program is considered tested if a cycle of all test cases for the program is executed and all the bugs have been identified and fixed and also all the vulnerabilities identified have been fixed if any. |
| Integration Testing | All the units, which share the interface, have been unit tested, and unit test plan exit criteria are met and all previous known vulnerabilities fixed if any | Integration testing is considered complete if all the interfaces have been tested and there are no major defects pending and not more than one minor defect pending for any interface and also all the vulnerabilities identified have been fixed if any |
| System Testing | All the interfaces between the units have been successfully tested and no major defects are pending and all previous known vulnerabilities fixed if any | The system is considered tested in following scenarios   * There are no major defects pending. * Not more than 5 minor defects are pending. * There are no vulnerabilities pending if applicable |

## 6.1 Build Suspension & Resumption Criteria:

Specify the criteria used to suspend all or portion of the testing activity on the test item. This situation arises in conditions like crashes and loss of major functionality during testing.

Specify the testing activities, which must be repeated, when testing is resumed.

For example:

**6.1.1 Build Suspension Criteria:**

The guideline criteria detailed below would be used as a reference to determine the level of quality expected from the Test application when passed to the QA team. If the below mentioned criteria are not met, respective modules delivered to the QA team would be deemed un-testable. Testing would be suspended in such a scenario and the build would be returned to the Development Team for improvement. Testing would be resumed only once the quality reaches the acceptable level mentioned in section Entry Criteria.

The QA team may suspend partial or full-testing activities on a given build if any of the following occurs:

* Some Features are missing from the new build.
* QA cannot access the new build or a component as mentioned in release notes.
* There is a fault with a feature that prevents its testing and there is no suitable work around.
* 2 Blocker/ Major Severity defects found within a few minutes of testing the provided build.
* New build does not contain the specified change(s) and versioning naming convention is not defined as per the guideline provided above.
* Latest Source code and its associated components, reflective of the build shared, along with the Release notes are missing.
* Development has not updated/fixed the problem(s)/Issue(s) that previously suspended the testing.
* A new version of the software is available to test.

The QA Lead/Resource would make the decision on whether the testing of a build should be suspended or carried out further.

**6.1.2 Build Resumption Criteria:**

A Sanity test would be run after the new build is applied and problems that caused the last build to be rejected would be verified as being resolved.

The new build should include:

* Details on why the initial build failed and areas the fix has an impact on
* Build Release Notes

# 7.Test Execution Strategy

## 7.1 Understanding Requirements:

* Requirement specifications will be sent by client.
* Understanding of requirements will be done by QA along with Respective lead and developer and queries are raised if any.
* Raised queries will be sent by lead to the client.
* Response to queries will be sent by client.
* Queries can be discussed over the call with the client and MOM will be dropped for the same.

**7.2 Preparing Test Cases:**

QA will be Preparing Test Cases in excel sheet based on the requirement specifications. This will cover all scenarios for requirements.

**7.3 Traceability to Requirements**

This section addresses the traceability between Test cases and requirements.

QA will be preparing a test matrix which maps test cases to respective requirements . This will ensure the coverage for requirements.

**7.4 Reviewing test cases and matrix:**

* Peer review will be conducted for test cases and test matrix by senior QA member in QA team
* In certain cases for e.g. complex requirements, lead’s help will be taken for conducting review
* Any comments or suggestions on test cases and test coverage will be provided by reviewer respective Author of Test Case and Test Matrix
* Suggestions or improvements will be re-worked by author and will be sent for approval
* Re-worked improvements will be reviewed and approved by reviewer

**7.5 Creating Test Data**

Test data will be created by respective QA on client’s developments/test site based on scenarios and Test cases.

**7.6 Executing Test Cases:**

* Test cases will be executed by respective QA on the client's development/test site based on designed scenarios, test cases and Test data.
* Test results (Actual Result, Pass/Fail) will be updated in Xray Management in JIRA.

**7.7 Defect Logging and Reporting:**

QA will be logging the defect/bugs in JIRA found during execution of test cases and will assign the Bug id generated by JIRA to respective test cases document. After this, QA will inform the respective developer about the defect/bugs.

**7.8 Retesting and Regression Testing:**

Retesting for fixed bugs will be done by respective QA once it is resolved by the respective developer and bug/defect status will be updated accordingly. In certain cases, regression testing will be done if required.

**7.9 Deployment/Delivery:**

* Once all bugs/defects reported after complete testing are fixed and no major bugs are found, the code will be deployed to the client's test site by the developer.
* Once a round of testing will be done by QA on the client's test site if required, a report will be delivered along with sample output by email to the respective lead and Report group.
* QA will be submitting the filled hard copy of the delivery slip to the respective QA Lead and PM
* Once the lead gets the hard copy of the delivery slip filled by QA and developer, he will send the report delivery email to the client along with QA approved builds.

# 8.Tools Utilized

The Description of tools that shall be utilized in the project needs to be specified under this section.

Tools used in conducting Tests should be reviewed at defined intervals to determine if there is a need to improve and/or upgrade them .The effect of using such tools on the quality of the software product should be carefully considered.

| **Tool Name** | **Vendor/In-House** | **Version No.** | **Purpose** |
| --- | --- | --- | --- |
| JIRA | Defect Tracking | 9.12.10 | For Logging defects against failed Test Case and writing the test cases with the xray Management Integration tool. |

# 9.Test Environment

Specify the necessary and desired properties of the test environment configuration.

| **Items** | **Configuration** |
| --- | --- |
| Hardware | Android & IOS devices |
| Software | Netomi App |
| Test Data | Data will be created by QA |
| Client Specific | Client Specific Requirements, Test data / Test Environment provided by client |

# 10.Item Pass Fail Criteria

Defects will be classified as follows according to severity of the impact on the system:

| **Severity** | **Impact** |
| --- | --- |
| Major | System inoperable, An implementation that does not meet the requirements (or any other input document), Defects that may cause the system to hang, crash, produce incorrect/ unexpected results or behavior, or corrupt user data with no known work around. |
| Minor | Major function disabled/ incorrect, Non-conformance to standards. Defects that cause incorrect results or behavior with known work around. Large and/or critical portion of the system is affected that would not cause operational failure. |
| Trivial | Defects that affect limited areas of functionality that either can be worked around or ignored. |

**10.1 Pass / Fail Criteria**

| **Item** | **To pass the total number of defects on running the complete suite of test cases bug count should be as below** | | |
| --- | --- | --- | --- |
|  | Major | Minor | Trivial |
| BUG | 0 | 1-5 | 1-2 |

# 11.Defect Analysis and Resolution

All the defects found while testing the Test application within various testing cycles, would be logged on a Defect tracking tool called “Jira”.

The Defect Life cycle would follow the standard set of processes where the QA team would be sharing the found defects with Development team as per the procedure mentioned below:

* A testable build would be delivered to QA Team and after successful Smoke/Sanity testing pass, QA Team would test the entire build to find any deviation from the requirements traced into Test Cases.
* Defects found during this exercise would be reported on Jira with respective severity, summary, steps to reproduce, screen shots & videos and assigned to Development lead or corresponding developer.
* Status of all the defects reported within this testing cycle would be “To Do”.
* Status of defects under development would be “In Progress”.
* Status of defects fixed by the Developer would be ‘Ready for QA’.
* Status of bugs defects by QA would be “Ready for UAT or Done”

# 12.Test Deliverables

1. Test Plan
2. Test Cases
3. Test Summary Reports
4. Defect status Reports
5. QA Metrics
6. Sign Off
7. QA Verified Builds
8. Demo

# 13.Risks, Assumptions & Constraints

**13.1 Risks Foreseen**

List down below the Risks foreseen while planning Testing strategy and associated activities:

| **Sr. No.** | **Risk Description** | **Reference for Tracking in Risk Tracking Sheet** |
| --- | --- | --- |
| 1 | Hardware problems |  |
| 2 | Problem in accessing the test cases |  |
| 3 | Changes in requirement, use case document |  |
| 4 | Sanity testing failures |  |
| 5 | Insufficient time per test cycle/Delay in QA release |  |
| 6 | Test documents, test scripts and test results data loss |  |
| 7 | New OS version available in Devices |  |
| 8 | Server Down |  |
| 9 | Poor Internet Connectivity |  |
| 10 | Last Minute changes in code |  |

**13.2 Assumptions**

| **Sr. No.** | **Assumptions** |
| --- | --- |
| 1 | N/A for Now |

**13.3 Constraints**

| **Sr. No.** | **Constraints** |
| --- | --- |
| 1 | N/A for Now |

# 14.Training Requirements

Based on the requirements of the project, training will be provided to all members who newly join the team. The training would focus on:

* Understanding of the requirements of the entire application.
* How to test on Mobile Devices
* Understanding of the Test Plan & Test cases
* To get a good understanding of the scope of the project, each QA team member will have to read the following documents:
  + System Requirement Document
  + Functional Requirement Document
  + Test Plan
  + Test Cases
  + User Stories
  + Follow Figma
  + Read Out Basecamp Communication

The training exercise would include:

**Induction**: To be initiated by the QA Lead/Manager and with required assistance from Project Manager and BA.

**Product familiarity**: To be done for Team members to get familiar with the product.

**Constant Review with new team members**: All new team members will be considered as certified when the reviewer, generally QA Lead or PM, feels comfortable with the knowledge possessed by all team members and is confident to start the testing activity with them.

# 15.Test Responsibilities

**Project Manager**:

* Name: [Ankur Kushwaha]
* Responsibilities: Overseeing the testing process, ensuring timelines are met, coordinating with team members, reviewing test cases, and reporting progress to management.

**Test Engineer**:

* Name: [Parul Sharma]
  + Responsibilities: Writing and executing test cases for features , logging defects, and verifying fixes.