

**Software Test Plan (STP)**

Mail Speak Application (MSA)

University of Maryland Global Campus (UMGC)

Software Engineering (SWEN) 670

Fall Cohort 2022

Team B

November 5, 2022

**Document Control**

**Document Information**

|  | **Information** |
| --- | --- |
| Document Identification | USPS- MSA-STP-20220917-Fall2022 |
| Document Owner | UMGC SWEN 670 |
| Issue Date | September 17, 2022 |
| Last Saved Date | October 29, 2022 |
| File Name | STP - Team B - MSA.docx |

**Document History**

| Version | Issue Date | Changes |
| --- | --- | --- |
| 0.1 | 9/14/2022 | Initial Draft |
| 0.2 | 9/17/2022 | Review Updates |
| 1.0 | 9/17/2022 | Final Deliverable |
| 1.1 | 10/9/2022 | Logo and name change updates |
| 2.0 | 10/29/2022 | Post-development update |
| 2.1 | 10/30/2022 | References style and document suite updates |
| 2.2 | 11/01/2022 | Functional test cases update |
| 3.0 | 11/5/2022 | Final deliverable |

Approval Signatures

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ☐ APPROVE  ☐ DISAPPROVE | |  | | --- | |  | | Tatiana Kozhevnikova, Product Owner  Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ | |  | |
| ☐ APPROVE  ☐ DISAPPROVE | |  | | --- | |  | | Michael Conatser, Project Manager  Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ | |

**Table of Contents**

[1 Introduction 7](#_Toc118487685)

[1.1 Purpose 7](#_Toc118487686)

[1.2 System Overview 7](#_Toc118487687)

[1.3 Project Scope 7](#_Toc118487688)

[1.4 Identify Stakeholders 8](#_Toc118487689)

[1.5 Project Documentation 8](#_Toc118487690)

[1.5.1 Project Suite of Documents 8](#_Toc118487691)

[1.5.2 Document References 9](#_Toc118487692)

[2 Evaluation Goals and Test Motivators 9](#_Toc118487693)

[2.1 Evaluation Goals 9](#_Toc118487694)

[2.2 Test Motivators 9](#_Toc118487695)

[2.3 Target Test Items 9](#_Toc118487696)

[3 Test Approach and Process 10](#_Toc118487697)

[3.1 Test Approach 10](#_Toc118487698)

[3.2 Test Process 10](#_Toc118487699)

[3.2.1 Test Planning 10](#_Toc118487700)

[3.2.2 Test Definition 10](#_Toc118487701)

[3.2.3 Test Execution 11](#_Toc118487702)

[4 Test Strategy 11](#_Toc118487703)

[4.1 Testing Types and Techniques 11](#_Toc118487704)

[4.1.1 Unit Testing 11](#_Toc118487705)

[4.1.2 Integration Testing 11](#_Toc118487706)

[4.1.3 System Testing 12](#_Toc118487707)

[4.1.4 User Acceptance Testing 12](#_Toc118487708)

[4.1.5 Regression Testing 12](#_Toc118487709)

[4.2 Measuring the Extent of Testing 12](#_Toc118487710)

[4.2.1 Entrance Criteria 12](#_Toc118487711)

[4.2.2 Exit Criteria 13](#_Toc118487712)

[4.2.3 Suspension and Resumption Criteria 13](#_Toc118487713)

[5 Testing Resources and Standards 13](#_Toc118487714)

[5.1 Roles and Responsibilities 13](#_Toc118487715)

[5.1.1 Project Sponsors 13](#_Toc118487716)

[5.1.2 Project Manager 14](#_Toc118487717)

[5.1.3 Project Test Team 14](#_Toc118487718)

[5.1.4 Project Stakeholders 14](#_Toc118487719)

[5.2 Test Environment 14](#_Toc118487720)

[6 Test Layout 15](#_Toc118487721)

[6.1 Software Configuration Management 15](#_Toc118487722)

[6.1.1 Tester workstations configuration 15](#_Toc118487723)

[6.1.2 Source Code Configuration 15](#_Toc118487724)

[6.2 Hardware Configuration Management 15](#_Toc118487725)

[7 Project Testing Milestones 15](#_Toc118487726)

[7.1 Issue Tracking/Defect Management 17](#_Toc118487727)

[8 Test Execution 17](#_Toc118487728)

[8.1 Test Procedures/Process 17](#_Toc118487729)

[8.2 Bug Reporting and Tracking 17](#_Toc118487730)

[8.3 Severity Values 18](#_Toc118487731)

[8.4 Test Reporting/Requirements Traceability 18](#_Toc118487732)

[9 Software Test Plan 19](#_Toc118487733)

[9.1 Unit Test Plan 19](#_Toc118487734)

[9.1.1 Items Tested 19](#_Toc118487735)

[9.1.2 Items Not Tested 19](#_Toc118487736)

[9.1.3 Test Data 20](#_Toc118487737)

[9.1.4 Test Environment 20](#_Toc118487738)

[9.1.5 Test Deliverables 20](#_Toc118487739)

[9.2 Integration Test Plan 20](#_Toc118487740)

[9.2.1 Items Tested 21](#_Toc118487741)

[9.2.2 Items Not Tested 21](#_Toc118487742)

[9.2.3 Test Data 21](#_Toc118487743)

[9.2.4 Test Environment 21](#_Toc118487744)

[9.2.5 Test Deliverables 21](#_Toc118487745)

[9.3 System Test Plan 21](#_Toc118487746)

[9.3.1 Items Tested 22](#_Toc118487747)

[9.3.2 Items Not Tested 22](#_Toc118487748)

[9.3.3 Test Data 22](#_Toc118487749)

[9.3.4 Test Environment 22](#_Toc118487750)

[9.3.5 Test Deliverables 22](#_Toc118487751)

[9.4 User Acceptance Test Plan 22](#_Toc118487752)

[9.4.1 Items Tested 23](#_Toc118487753)

[9.4.2 Items Not Tested 23](#_Toc118487754)

[9.4.3 Test Data 23](#_Toc118487755)

[9.4.4 Test Environment 23](#_Toc118487756)

[9.4.5 Test Deliverables 23](#_Toc118487757)

[10 Risks, Dependencies, Assumptions and Constraints 23](#_Toc118487758)

[10.1 Risks 23](#_Toc118487759)

[10.2 Dependencies 24](#_Toc118487760)

[10.3 Assumptions 24](#_Toc118487761)

[10.4 Constraints 24](#_Toc118487762)

[11 Management Process and Procedures 25](#_Toc118487763)

[11.1 Problem Reporting, Escalation, and Issue Resolution 25](#_Toc118487764)

[11.2 Approval and Signoff 26](#_Toc118487765)

[Attachment A – Acronyms and Abbreviations 27](#_Toc118487766)

[Attachment B - Test Requirements Traceability Matrix 28](#_Toc118487767)

[Attachment C - Test Cases 28](#_Toc118487768)

[C – 1 Unit Test Cases 28](#_Toc118487769)

[C – 2 Functional Test Cases 30](#_Toc118487770)

# Introduction

## Purpose

This Software Test Plan (STP) outlines the overall approach that the Development Team will utilize in the testing of Mail Speak Application (MSA) during its development, test, and user acceptance test phases. This plan encompasses test processes, responsibilities, and tools required to verify the system’s adherence to all defined functional and non-functional requirements. Ultimately, this document will serve as a formal definition of the system test procedures and practices employed to ensure that the USPS MSA meets the rigorous quality requirements of the client and the contractor.

This Software Test Plan also supports the following specific objectives:

• Identifies items that should be targeted by the tests.

• Identifies the motivation for and ideas behind the test areas to be covered.

• Outlines the testing approach that will be used.

• Identifies the required resources.

• Lists the deliverable elements of the test project.

## System Overview

The MSA provides users access to emails stored in the inboxes of their email accounts, with specific attention given to emails received through the United States Postal Service (USPS) Informed Delivery service. The application places emphasis on accessibility for the visually impaired through the utilization of text-to-speech and speech-to-text functionality.

The enhancements to the informed delivery application that are to be implemented will expand the product’s current functionality. These enhancements will provide users with more powerful email search functionality, enhanced usability (with visually impaired users as the primary focus), and user behaviour reporting.

## Project Scope

The application under development is a mobile application that has been created to be used in conjunction with USPS Informed Delivery service. Its purpose is to provide visually impaired and sighted users access to the service with accessibility features. The features will enhance the pre-existing text-to-speech and voice input functionality already implemented in the application and will provide enhanced searching capabilities to allow for better navigation and organization of emails. A Chatbot will also be implemented to improve the user experience (UX) through a more natural interaction versus reading documentation on functionality.

Table 1.1 - Project Requirements

| Requirement ID | Description |
| --- | --- |
| 1 | Search View: search past emails using specific date range and/or keyword |
| 2 | Chatbot: Provide chatbot feature to assist user in navigating the application |
| 3 | Gesture: Provide gesture driven command feature |
| 4 | Voice driven: Provide enhanced voice driven command features |
| 5 | Reading Mode: Provide enhanced text-to-speech capabilities |

## Identify Stakeholders

Table 1.2 - Project Stakeholders

| Stakeholder Name | Organization | Project Role |
| --- | --- | --- |
| Mir Assadullah | UMGC | Professor, Program Manager |
| Roy Gordon |  | Project Mentor |
| Robert Wilson |  | DevSecOps Mentor |
| Robert Dixon | USPS Informed Delivery | Project Sponsor |
| Alexander Chan | UMGC | Software Engineer (SE) II |
| Andrew Asavarungsrikul | UMGC | Software Engineer (SE) I |
| Erin Sauter | UMGC | Software Engineer (SE) I |
| Jonay Simmons | UMGC | Software Engineer (SE) I |
| Lawrence Van Tassel | UMGC | Software Engineer (SE) III |
| Michael Conatser | UMGC | Project Manager (PM), Scrum Master |
| Minyahil Kebebegn | UMGC | Software Engineer (SE) II |
| Sarah Johnson | UMGC | Software Engineer (SE) I |
| Shane Knowles | UMGC | DevSecOps Engineer (Principal) |
| Tatiana Kozhevnikova | UMGC | Product Owner |
| TraMel Perry | UMGC | Principal Software Engineer (SE) |

## Project Documentation

### Project Suite of Documents

This Software Test Plan is part of a set of essential documents created to adequately manage, control and deliver the Mail Speak Application. Artifacts that are provided within the document package contain vital information for the application’s ongoing support and operation throughout its life cycle. Each document is created within the specific Milestone of the project. Therefore, the version and date of some documents could be marked as “to be determined” (TBD) in Table 1.3.

Table 1.3 - Project Documentation Package

| Document | Version | Date |
| --- | --- | --- |
| Project Management Plan (PMP) | 4.0 | 11/05/2022 |
| Software Requirements Specification (SRS) | 4.0 | 11/05/2022 |
| Technical Design Document (TDD) | 3.0 | 11/05/2022 |
| Software Test Plan | 3.0 | 11/05/2022 |
| Programmer Guide | 2.0 | 11/05/2022 |
| Development and Operations Guide | 2.0 | 11/05/2022 |
| User Guide | 1.0 | 11/05/2022 |
| Test Report | 1.0 | 11/05/2022 |

### Document References

During the process of writing current STP the following documents were referenced:

* Team B. (2022). *United Global Master Coders Team B PMP*. <https://umgcdev361.sharepoint.com/:w:/r/sites/SWEN670Fall2022/_layouts/15/Doc.aspx?sourcedoc=%7B671384DF-E89D-46D7-A258-3B60416FB909%7D&file=Informed%20Delivery%20Enhancement%20Team%20B%20PMP.docx&action=default&mobileredirect=true>
* Team B. (2022). *United Global Master Coders Team B SRS*. <https://umgcdev361.sharepoint.com/:w:/r/sites/SWEN670Fall2022/_layouts/15/Doc.aspx?sourcedoc=%7B9B27AD03-C5C1-45A0-9EDC-6B22427BC8BA%7D&file=Informed%20Delivery%20Enhancement%20Team%20B%20SRS.docx&action=default&mobileredirect=true>
* Team B. (2022). *United Global Master Coders Team B TDD*. <https://umgcdev361.sharepoint.com/:w:/r/sites/SWEN670Fall2022/_layouts/15/Doc.aspx?sourcedoc=%7BF0A49AE7-2FC0-497B-9DFB-0322352B8025%7D&file=Informed%20Delivery%20Enhancement%20Team%20B%20TDD.docx&action=default&mobileredirect=true>

# Evaluation Goals and Test Motivators

## Evaluation Goals

The evaluation goals for the testing of the MSA are to evaluate the developed system to the data, functional, system, performance, system security (at the application level), and usability requirements provided by the system owner and stakeholder. The testing effort will seek to verify requirements, verify compliance with applicable standards, validate performance based on acceptable parameters requested by the system owner, and identify issues/defects. The testing efforts will also strive to find problems that could be perceived as quality risks and will provide advice about the impact of these risks on the project.

## Test Motivators

Testing is driven by the following motivators:

1. Verify that data, functional, system, performance, 508, system security (at the application level), and usability requirements have been met.
2. Identify and document risks and limitations.
3. Validate application stability.
4. Identify defects and potential issues.

## Target Test Items

The following is a proposed list of the target test items that will be tested by the MSA project team during Milestone 3:

1. Search View
2. Accessibility
   1. Gesture-based commands
   2. Voice-Driven commands
   3. Reading mode
3. Chatbot

# Test Approach and Process

This section outlines the testing approach which will be employed by MSA project team.

## Test Approach

All test phases are designed with an emphasis on validating the established use cases as defined by the project team. All defined use case processes and test scenarios are tested at some point in the testing process. Testing should strive to mirror the production environment as much as possible. Each phase should look more and more like production, so that the last phase matches production as close as possible.

Testing, in general, is an iterative process. Each phase builds upon the prior. There are iterations of the testing cycle every time a change to the application or interface is made. Successful completion of the test phases is predicated on all test scripts being executed and test results meeting pre-established expectations of quality. In other words, test scenarios must pass within an acceptable level of pre-defined error.

At the end of each test phase, sign-off of successful test documentation by the PM, Product Owner (PO), and Testing team leads are required for phase completion. It is this group that has final signoff at the end of each testing phase and is responsible for defining an acceptable level of errors.

## Test Process

The test process will be broken down in (3) stages. The stages assigned for the test methodology will ensure the integrity and correctness of the testing and that the test cases are traceable to the requirement and specification assigned to the testing effort.

### Test Planning

The MTP (Master Test Plan) document focuses on the testing scope (issues/enhancements, functionality to-be-tested, testing types, data validation) testing schedule, resources, and test environment.

The test plan will define the overall approach, activities, and responsibility for testing the application.

Deliverable: ***Master Test Plan*** (this document).

### Test Definition

The test definition stage includes the test procedures that cover the functionality and business processes defined in the Test Planning stage.

The test procedures are written in an “Action; Expected Results” format and when necessary, the test procedure will be associated with the data sets which support the test procedure.

Deliverable: Master Test Plan Appendices: *Test Case Document and Test Datasets*.

### Test Execution

During this stage, Test Cases are executed. Defects will be reported for each discrepancy between Expected and Actual results, as stated in the respective Test Case.

At the end of this stage, the Test Summary Report will be generated that includes the test results to allow the project management team to evaluate the quality level and decide about software release or if an additional testing cycle is required. The Test Summary Report will be used as the final test artifact during project.

Deliverables: *Test Summary Report*

# Test Strategy

The test strategy defines the scope and general direction of the test effort. It defines the testing techniques and types that will be used for this development effort and the proposed sequence of test and area of responsibility. The types and techniques that will be used are defined in Section 4.1.

The Quality Assurance (QA) Team consists of development, test engineering, and engineering team personnel to conduct their areas of responsibility within the testing cycle. The areas of responsibility may overlap, and the individual test teams will work together to complete testing cycle.

The following is the proposed sequence of testing to be conducted for the MSA project:

1. Development Testing: Unit Testing
2. Development Testing: Integration Testing
3. System Testing
4. User Acceptance Testing (UAT)
5. Regression Testing

*\*Testing cycles will be iterative. Sequence is suggested, but several of the tests to be conducted may be combined such as user and integration testing, or system testing and user acceptance testing.*

*\*\*Testing cycles will not only be iterative for each module, but for the system as additional modules are added to the system in production.*

## Testing Types and Techniques

Testing techniques and types described below will be used during MSA testing:

### Unit Testing

Unit testing is executed by the development staff to ensure that the separate parts built from the requirements work properly independently of the entire system. The unit as defined as the individual requirement validates the function being developed. This testing provides the foundation to putting the separate functionality together to initiate integration testing. Unit testing is required to help support the quality of the coding effort and overall quality of the application. These tests will be automated within the GitHub repository and managed by development staff.

### Integration Testing

Integration testing will be executed in conjunction with unit testing. Where unit testing tests the individual components, integration testing will ensure that the individual components are able to work together in harmony. This testing is completed by the development team in partnership with engineering and testing teams. The overall purpose is to better solidify the quality and completeness of the final deliverable by validating the entry criteria for entrance into the next phase of testing.

### System Testing

System testing is executed to ensure proper application functionality, including navigation, data entry, processing, and retrieval. The testing will execute each use case, use case flow, or function, using valid and invalid data, to verify the following:

* The expected results occur when valid data is used.
* The appropriate error/warning messages are displayed when invalid data is used.
* Each business rule is properly applied.
* Applicable 508 considerations have been applied.

### User Acceptance Testing

User interface testing will verify the navigation through the application properly reflects business functions and requirements, including screen to screen, field to field, and use of access methods (tab keys, mouse movements, and accelerator keys). Mobile application objects and characteristics, such as menus, size, position, state, and focus will also be exercised and evaluated against User Interface/ User Experience (UI/UX) concepts. Testing will create/modify tests for each screen to verify

### Regression Testing

Regression testing is the selective retesting of a software system that has been modified to ensure that any bugs have been fixed and that no other previously working functions have failed because of the reparations and that newly added features have not created problems with previous versions of the software. Also, referred to as verification testing, regression testing is initiated after a programmer has attempted to fix a recognized problem or has added source code to a program that may have inadvertently introduced errors. It is a quality control measure to ensure that the newly modified code still complies with its specified requirements and that unmodified code has not been affected by the maintenance activity. Regression testing will be completed within milestone 4 of this effort.

## Measuring the Extent of Testing

### Entrance Criteria

Entrance criteria are the required conditions and standards for work product quality that must be present or met prior to the start of a test phase.

Entrance criteria shall include the following:

* Affirmation by the development team that the system configuration is ready for testing.
* The development team has agreed that the application is ready to start testing. This will include a list of functionalities that are ready for testing and an updated list of testable functionalities will be provided with each build.
* Concurrence by project management that system testing shall begin.
* Review of completed test script(s) for the prior test phase (if applicable).
* No open critical/major defects remaining from the prior test phase (if applicable).
* Correct versioning of components moved into the appropriate test environment.
* Testing environment is configured and ready.

### Exit Criteria

Exit criteria are the required conditions and standards for work product quality that block the promotion of incomplete or defective work products to the next test phase of the component.

For each development phase, exit criteria shall include the following:

* Successful execution of the test scripts(s) for the current test phase.
* No open critical, major, or average severity defects unless the issue is determined to be low impact and low risk.
* A report on all outstanding Average or Minor defects has been presented by the test team.
* Component stability in the appropriate test environment.
* The test team concurs that the system is ready for the next phase.

### Suspension and Resumption Criteria

Suspension will be considered when any of the following are true:

* If any defects are found which seriously impact the test progress the test lead may choose to suspend testing.
* Hardware / software are not available at the time indicated in the project schedule.
* The build contains many serious defects which seriously prevent or limit testing progress.
* Assigned test resources are not available when needed by the test team.
* Unavailability of external dependent systems during execution.

Resumption will be considered when any of the following are true:

* When a fix is successfully implemented, and the Testing Team is notified to continue testing.
* Hardware / software become available, and schedule has been adjusted for time delay.
* Defects are corrected and the system re-enters the test cycle from beginning.
* Test resources become available.
* When the external dependent systems become available again.

If testing is suspended, resumption will only occur when the problem(s) that caused the suspension have been resolved. When a critical defect is the cause of the suspension, the “FIX” must be verified by the testing team before testing is resume.

# Testing Resources and Standards

## Roles and Responsibilities

### Project Sponsors

Robert Dixon is serving as the project sponsor for the USPS Informed Delivery Enhancement project.

The project sponsor is responsible for the following test-related responsibilities:

* Software meets business needs: USPS Informed Delivery application adds further value to the end-customers (including vision impaired and non-vision impaired individuals)
* Provide a high-level direction and project oversight
* Approve software release to customers

### Project Manager

Michael Conatser is serving as the project manager for the USPS Informed Delivery Enhancement project.

The project manager is responsible for the following test-related responsibilities:

* Establishing and maintaining communication with the project sponsor on the projects progress on a scheduled basis and as circumstances dictate.
* Ensures that all testing meets the expectations of the project sponsors.
* Ensures that processes, deliverables, and artifacts adhere to client’s standards.
* Reviews and approves all testing tools, procedures, and quality goals.

### Project Test Team

The majority of the remaining project team members will serve as the project’s test team. Individuals in the project test team include Alexander Chan, Andrew Asavarungsrikul, Lawrence Van Tassel, Tatiana Kozhevnikova, Minyahil Kebebegn, and Shane Knowles.

These individuals are responsible for the following test-related responsibilities:

* Authors test plan and other test-related documentation, such as Test Results Matrices and User Acceptance Testing (UAT) scenarios.
* Identifies testing resources.
* Conducts testing.
* Communicates testing problems, risks, scope changes, and quality concerns to the contract Project Manager.

### Project Stakeholders

The numerous project stakeholders have varying interests in the implementation of the system. Stakeholders may or may not have any direct responsibility for project tasks, but their participation and support is essential to project success.

## Test Environment

Testing will be performed using Android and iOS devices as intended by the production environment. Artifacts will be generated as part of the Continuous Integration/Continuous Deployment (CI/CD) pipelines and uploaded to respective application store test environments for Android and iOS (Google Play Console and TestFlight). Builds of the application will be accessible through both methods and allow the test team to perform verification and validation testing for all features and bug fixes throughout the software development lifecycle (SDLC).

The Development testing environment will be that of the development teams’ working environments consisting of Android Studio and Flutter’s test environment. This will allow developers to make changes and test them locally to confirm the feature works as intended before attempting to push the change to the working branch for test. Development testing will cover integration and unit testing. No other third-party testing framework or tools will be used during this project.

The primary Test environment will be using the iOS/Android artifacts developed from the pipeline and those uploaded to the respective pre-production application stores. Builds will be uploaded here once they have passed all testing and flagged ready for pre-production. This test environment will be responsible for system, user acceptance, and regression testing. Once all functionality has been verified and validated in this environment the build will be flagged ready for production. At this point the build can be promoted to the respective application store’s production environment where it will be scanned and verified ready for production.

# Test Layout

## Software Configuration Management

### Tester workstations configuration

Testers will use personally owned equipment for testing. Equipment will only be required to have internet access.

### Source Code Configuration

All source code required for MSA development, including the existing MSA code, and infrastructure code, will be maintained in a source and versioning control system. The development team will use GitHub, a file-level version control system. Developers will be responsible for committing all code changes to the GitHub repository as changes are made. Quality Assurance will be responsible for using current code from GitHub when deploying to any environment.

## Hardware Configuration Management

Configuration of the basic systems (hardware, firmware, operating systems, networks, etc.) are the responsibility of the development team.

# Project Testing Milestones

| Milestone | Planned  Start Date | Actual  Start Date | Planned  End Date | Actual  End Date |
| --- | --- | --- | --- | --- |
| Testing Planning | 9/3/22 | 9/3/22 | 10/27/22 |  |
| Draft Test Plan | 9/3/22 | 9/17/22 | 9/17/22 | 9/17/22 |
| **Chatbot** | | | | |
| Review and Understand Chatbot Requirements | 9/3/22 | 9/3/22 | 9/12/22 | 9/12/22 |
| Publish Test Plan – Chatbot | 9/12/22 | 9/12/22 | 9/17/22 | 9/17/22 |
| Define Test Procedures – Chatbot: Chat suggestions | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Chatbot Help documentation | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Chatbot notifications | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Chatbot Search | 9/13/22 | 9/13/22 | 9/30/22 |  |
| **Gesture** | | | | |
| Review and Understand Gestures Requirements | 9/3/22 | 9/3/22 | 9/12/22 | 9/12/22 |
| Publish Test Plan – Gestures | 9/12/22 | 9/12/22 | 9/17/22 |  |
| Define Test Procedures – Gestures actions | 9/13/22 | 9/13/22 | 9/30/22 |  |
| **Voice Driven** | | | | |
| Review and Understand Voice Driven Requirements | 9/3/22 | 9/3/22 | 9/12/22 | 9/12/22 |
| Publish Test Plan – Voice Driven | 9/12/22 | 9/12/22 | 9/17/22 |  |
| Define Test Procedures – Activation | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Camera | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Chatbot | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Email View | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Main Menu | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Notification | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Search | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Settings | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Reading Mode | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Error | 9/13/22 | 9/13/22 | 9/30/22 |  |
| **Reading Mode** | | | | |
| Review and Understand Reading Mode Requirements | 9/3/22 | 9/3/22 | 9/12/22 | 9/12/22 |
| Publish Test Plan – Reading Mode | 9/12/22 | 9/12/22 | 9/17/22 |  |
| Define Test Procedures – Settings | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Main Menu | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Search | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Email View | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Notification | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Chatbot | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Enable/Disable | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Microphone | 9/13/22 | 9/13/22 | 9/30/22 |  |
| Define Test Procedures – Navigation | 9/13/22 | 9/13/22 | 9/30/22 |  |
| **Release** | | | | |
| Receive Software Build (Prototype) | 9/16/22 |  | 9/16/22 |  |
| Receive Software Build Release 1 | 10/19/22 |  | 10/19/22 |  |
| Receive Software Build Release 2 | 10/26/22 |  | 10/26/22 |  |
| Bug Validation (Prototype) | 9/17/22 |  | 9/17/22 |  |
| Bug Validation Release 1 | 10/29/22 |  | 10/27/22 |  |
| Bug Validation Release 2 | 11/5/22 |  | 11/3/22 |  |
| Conduct Test Procedure Clean-up | 10/31/22 |  | 11/4/22 |  |
| Generate Test Summary Report | 10/31/22 |  | 11/4/22 |  |

## Issue Tracking/Defect Management

Throughout the test phase, GitHub will be used to track all Bugs, Change Requests, and detailed test execution results. In addition, GitHub will be used to provide important QA metrics and ensure that all identified issues are resolved before any external release.

# Test Execution

## Test Procedures/Process

Test cases and test procedures for Testing will be created by the Development test team. These test cases will be entered into GitHub and will be traceable to the system requirements.

Every test case will have a test overview, preconditioning requirements, detailed test steps, verification points, and expected results. These test cases will be executed, and results will be reported in the test report.

## Bug Reporting and Tracking

The bug tracking and reporting process used by the test team will be used for all phases of this project. GitHub will be used for tracking and reporting all discrepancies identified during test execution. During test execution once a tester observes a potential bug (any aspect of the system that does not conform to requirements which includes hardware, software, stored data, use cases, etc.) the bug is entered into the GitHub repository as an issue with label bug. Also, any authorized member of the project team may enter or request a bug be entered into the system. The testers will validate all bugs entered into GitHub before consideration by the development team. All items that are determined to be valid will be assigned for immediate resolution or will be deferred for resolution in a subsequent release. When an item has been resolved by the assignee and validated by the testers, the bug will be closed. A high-level description of the process follows:

Originator enters issue into GitHub and labels issue with bug/enhancement tag.

Items that should be entered are:

Bug – Any aspect of the system that does not conform to requirements is a bug. This includes hardware, software, stored data, use cases, etc. Any individual who detects a bug should enter it into GitHub. In cases where it is unclear whether the observation is a bug or not, it should be entered into the system.

Enhancement Requests – Any idea for improvement to the software, or any aspect of the system that should be changed, even though it conforms to requirements, should be entered into GitHub as an issue with the enhancement label.

Assigned developer will review and work issue assigned changing status to in review when issue is completed.

Testers will review and retest as needed and either close issue or reopen for further development action.

## Severity Values

**Severity Values defined below as follows: 1-Critical, 2-Major, 3-Average, 4-Minor.**

1. A Minor Bug is defined as one of the following:

Operates as designed

Does not violate a significant requirement

Does not violate a significant principle of good programming practice

Does not violate a significant design or look & feel standard

Does not impact data integrity for valid data

1. An Average Bug is defined as one of the following:

Relatively minor operational deficiency

Does not violate a significant requirement

Does not violate a significant principle of good programming practice

Does not violate a significant design or look & feel standard

Operation does not impact data integrity for valid data

1. A Major Bug is defined as one of the following:

A significant operational deficiency that interferes with normal program execution

Violates a significant requirement

Violates a significant principle of good programming practice

Violates significant design or look & feel standards

Operation impacts data integrity for valid data

1. A Critical Bug is defined as one of the following:

Major-Bug related to an operational deficiency that prevents further operation of the application.

## Test Reporting/Requirements Traceability

At the conclusion of testing, a formal test report will be generated which will summarize all testing activities performed during all phases. This report will present the pass/fail status of tests executed, a summary of all bugs reported and resolved during that phase. Each report will include a listing of known bugs still outstanding. In addition to the formal report, a briefing will be held to describe the conduct and results of the testing activities.

A Test Requirements Traceability Matrix (TRTM) will be created which will address all testable requirements associated with the project. This matrix will identify the verification method (test, inspection, demonstration, etc.) for each requirement. The pass/fail status for each requirement will be maintained.

# Software Test Plan

## Unit Test Plan

This section discusses the tests that examine individual software units. A unit is the smallest piece of testable software. Unit testing verifies that each system component performs according to user requirements, and demonstrates internal integrity and correctness of the component.

The goals of unit testing are to:

1. Determine the extent to which unit behavior matches design specifications
2. Increase to an acceptable level the confidence of the developer that the unit will behave correctly under all circumstances of interest
3. Determine the extent to which the unit code structure conforms to the intended design structure

It is the responsibility of the developer to perform unit testing and report the results to the project manager. Unit testing will be conducted in the system development environment.

The unit tests produced for this project will cover only the functionality implemented in this project. Unit tests covering existing functionality in the application are excluded from this section. Unit tests strictly covering functionality implemented in any concurrent projects are excluded from this section.

Unit tests will be performed in GitHub as part of the Continuous Integration/Continuous Deployment (CI/CD) pipeline. Before a code branch may be integrated into the application’s main branch, intended for release to production, all unit tests must pass successfully. This minimizes the introduction of code defects to the application.

All functionality implemented prior to the initiation of this project or implemented by any project being executed concurrently is not included in this test plan; however, all unit tests covering functionality outside of this project’s scope must pass successfully for the application to be considered ready for integration testing.

## Items Tested

1. Email date and keyword search functionality
2. Chatbot notification setting
3. Chatbot help
4. Search email by date and sender via chatbot
5. Search stored mail pieces by sender, mail body text, start date, and end date
6. Storage of mail pieces
7. Deletion of mail pieces

## Items Not Tested

1. Software functionality included with the operating environment
2. Functionality implemented by any project being executed concurrently

## Test Data

All test data will be hard coded into the application’s test project. Management of this test data will be the sole responsibility of the application’s software engineers.

## Test Environment

Testing will be performed in the system development environment and as part of the CI/CD pipeline as described in section 5.2.

## Test Deliverables

The deliverables for all testing phases are the completed test cases and test scripts along with any change requests due to defects; this will be included in the Test Report document. Any additional information validating the results of test scripts will be attached to the testing documentation, such as table printouts for data, replication testing or screenshots of web pages. Once all testing documents have been completed, the application can be deemed ready for integration test.

The developer may raise a problem as an issue in the risk and issue log if the problem cannot be fixed with the coding.

Detailed documentation of all unit tests included in this project is to be included in the Test Report. All unit test execution documentation will be recorded in GitHub. While a unit test execution log is not included in the deliverables for this project, it may be produced upon demand by the customer.

## Integration Test Plan

This section discusses the tests that examine the integrated groupings of software units, called software modules. Integration testing verifies that the inter-relationships between components behave as expected and that individual components retain unit testing level functionality when integrated as part of the whole system. Integration testing verifies interface requirements, a subset of data and functional requirements that require more than one component to be fully satisfied, and performance requirements allocated to a subset of components.

The goals of integration testing are to:

1. Determine the extent to which an integrated component is correct and consistent with detailed design and requirements specifications
2. Ensure that units work together correctly
3. Increase to an acceptable level the confidence of the developer that integrated components will behave correctly under all circumstance of interest
4. Identify problems that arise in inter-connecting units or lower-level components

It is the responsibility of the developer to perform integration testing and report the results to the project manager. Integration testing will be conducted in the system development environment.

The integration tests performed for this project will cover only the interactions between code modules implemented in this project. Integration tests covering interactions between existing code modules which already exist the application are excluded from this section. Integration tests strictly covering code modules implemented in any concurrent projects are excluded from this section.

All functionality implemented prior to the initiation of this project or implemented by any project being executed concurrently is not included in this test plan; however, all integration tests covering functionality outside of this project’s scope must pass successfully for the application to be considered ready for system testing.

### Items Tested

1. UI control Input to email search
2. Chatbot input to command execution
3. Voice input to command execution

### Items Not Tested

1. Interactions between device hardware and code modules
2. Interactions between code modules implemented by any project being executed concurrently

### Test Data

All test data will be hard coded into the application’s test project. Management of this test data will be the sole responsibility of the application’s software engineers.

### Test Environment

Testing will be performed in the system development environment.

### Test Deliverables

The deliverables for all testing phases are the completed test cases and test scripts along with any change requests due to defects; this will be included in the Test Report document. Any additional information validating the results of test scripts will be attached to the testing documentation, such as table printouts for data, replication testing or screenshots of web pages. Once all integration testing has been completed, the application can be deemed ready for system testing.

The developer may raise a problem as an issue in the risk and issue log if the problem cannot be fixed within the application’s code.

Detailed documentation of all integration tests included in this project is to be included in the Test Report.

## System Test Plan

This section discusses the tests that examine the levels of formal System Testing that take place during the system development process.

The goals of system testing are to:

* 1. Determine the extent to which an adheres to required specifications
  2. Ensure that systems work together correctly
  3. Increase to an acceptable level the confidence of the developer that the system will behave correctly under all circumstance of interest
  4. Identify problems that arise

System testing shall be performed by the Quality Assurance team. System testing shall be performed to ensure that the systems of the application operate successfully within the boundaries of the expected usage conditions.

Technical support during the System Testing event will be provided by the development team. This support will include the establishment of the test environment and resolution of any technical issues during the System Testing event.

The QA team may raise a problem as an issue in the risk and issue log if the problem cannot be fixed within the application’s code.

### Items Tested

* 1. Manual interface operation and navigation performance
  2. Voice command operation and navigation performance
  3. Inbox search performance

### Items Not Tested

* 1. Speech-to-text recognition performance
  2. Gesture recognition performance
  3. Performance of existing features
  4. System security

### Test Data

Test data shall be provided by the customer and maintained by the development team. This will include an email account, which is accessible by the application, and USPS Informed Delivery Daily Digest emails as well as other non-USPS Informed Delivery emails.

### Test Environment

Testing will be performed on all platforms on which the application is intended to be operable. Testing shall be performed on Android and iOS devices or devices that emulate the Android and iOS environments. The operating systems for these devices shall be the latest stable versions for Android (Android 13) and iOS (iOS 16). The device hardware (or emulated device hardware) shall be representative of modern device hardware.

### Test Deliverables

The deliverables for all testing phases are the completed test cases and test scripts along with any change requests due to defects; this will be included in the Test Report document. Any additional information validating the results of test scripts will be attached to the testing documentation, such as table printouts for data, replication testing or screenshots of web pages. Once all system testing has been performed and all defects have been addressed, the system shall be considered ready for user acceptance testing.

Detailed documentation of all system tests included in this project is to be included in the Test Report.

## User Acceptance Test Plan

User Acceptance Testing will be performed by the development Team and the business users. If it is determined that the business users will not perform UAT, the product owner shall be responsible for the final acceptance of the product. UAT shall be conducted to gain acceptance of all functionality from the user community. UAT shall verify that the system meets user requirements as specified.

Technical support during the User Acceptance Testing will be provided by the development team. This support will include the establishment of the test environment, product demonstration, and resolution of any technical issues during the User Acceptance Testing event.

Upon the successful completion of User Acceptance Testing and the confirmation from the customer that the application satisfies all project requirements, the product will be considered ready for release.

### Items Tested

* 1. Email date and keyword search functionality
  2. Chatbot functionality
  3. Voice driven commands
  4. Gesture navigation
  5. Screen reader functionality
  6. UI/UX enhancements

### Items Not Tested

* 1. Any functionality implemented by any project being executed concurrently

### Test Data

Test data shall be provided and maintained by the development team. This will include an email account, which is accessible by the application, and USPS Informed Delivery Daily Digest emails as well as other non-USPS Informed Delivery emails.

### Test Environment

Testing will be performed on all platforms on which the application is intended to be operable. Testing shall be performed on Android and iOS devices or devices that emulate the Android and iOS environments. The operating systems for these devices shall be the latest stable versions for Android (Android 13) and iOS (iOS 16). The device hardware (or emulated device hardware) shall be representative of modern device hardware.

### Test Deliverables

The deliverables for all testing phases are the completed test cases and test scripts along with any change requests due to defects; this will be included in the Test Report document. Any additional information validating the results of test scripts will be attached to the testing documentation, such as table printouts for data, replication testing or screenshots of web pages.

The acceptance test team will produce the test report document.

# Risks, Dependencies, Assumptions and Constraints

## Risks

This section describes the possible risks, mitigation strategy and contingency plan to realize the risks.

Table 10.1 – Testing Risks

|  |  |  |
| --- | --- | --- |
| Risk | Mitigation Strategy | Contingency  (Risk is realized) |
| Lack of Test account. | Project manager and developers ensure to setup test account before testing begin. | Developers/Testers use personal account. |
| Lack of experienced iOS tester. | Tester along with requirements will ensure complete clarification of requirements before testing can begin. | Reassess functional requirements provided. |
| Prerequisite requirements are not met. | Developers will fix existing bugs that must be met before Testing can start.  Program Manager will ensure the requirements are fixed by developers. | Consider percentage of testing failures due to outstanding issues. |
| Test data proves to be inadequate. | Program Manager will ensure a full set of suitable and protected test data is available.  Developers and Testers will indicate what is required and will verify the suitability of test data. | Redefine test data based on requirements.  Review Test Plan and modify as needed. |

## Dependencies

This section describes the dependencies, potential impact of dependencies and responsible stakeholders.

Table 10.2 - Testing Dependencies

|  |  |  |
| --- | --- | --- |
| Dependency | Potential Impact of Dependency | Owners |
| Test data availability for all scenarios. | Functional testing may be incomplete if data is not available for all scenarios. | Project management, Product owner. |
| Test environment is reliably available. | Testing might not be completely executed. | Project management, Product owner. |
| Full information about the system is available. | In the absence of information, testing plans may change later than is desired. | Project management. |

## Assumptions

This section describes the Assumptions, impact of assumptions, and responsible stakeholders.

Table 10.3 - Testing Assumptions

|  |  |  |
| --- | --- | --- |
| Assumption | Impact of Assumption | Owners |
| Test account will be configured according to specification. | Testing may be affected. | Project management, Product owner. |
| Test data will be made available in testing account for all scenarios. | Testing may be affected if enough data are not made available for all possible scenarios of the requirements at time of testing. | Project management, Product owner. |

## Constraints

This section describes the constraints, impact of the constraints on the testing effort and responsible stakeholders.

Table 10.4 - Testing Constraints

|  |  |  |
| --- | --- | --- |
| Constraint On | Impact Constraint has on Test Effort | Owners |
| iOS Environment availability | iOS testing may be affected | Developer and Tester |
| iOS Developer and Tester | May affect the testing effort | Project manager, Developer |

# Management Process and Procedures

The management process describes activities related to identifying, documenting, assessing, prioritizing, assigning, resolving and controlling issues.

## Problem Reporting, Escalation, and Issue Resolution

The process followed for issue management is depicted in Figure 11.1

Figure 11.1 - Problem Management Process

GitHub will be used for issue logging, classification and prioritization, controlling, status reports and closure details. Issues can be identified or raised by any Project Stakeholder throughout the project lifecycle, using different communication channels specified by Communication Management Plan. Team members reporting the issue will create tickets with meaningful names and problem descriptions, how the problem occurred or what is causing the problem. Issues will be classified and prioritized by the Project Manager and Project Owner according to issue priority levels presented in Table 11.1. GitHub labels different from the “bug” label shall be used to mark the following issue types:

* Related to change management, such as business, customer and environmental changes.
* Related to resources, e.g., equipment, material or people problems.
* Related to a third party, i.e., vendors, suppliers and other outside parties.

Table 11.1 - Issue Priority Levels

| Priory Level | Description |
| --- | --- |
| Critical | Critical to the whole system, might cause system failure and requires immediate resolution. |
| High | Impacts main activities in the process of system development. |
| Medium | Causes minimal deviation from product requirements and development process. |
| Low | Has a minor effect on the project realization. |

The above-specified issue types that fall under critical and high priority will require attention and further approval from the Project Sponsor or Program Manager, and after that, the process will continue to the next stage. Issues with medium and low priority will be assigned by PM/PO to the most qualified team member(s) to perform analysis and investigation of the problem. During the investigation process, any questions shall be clarified with PM/PO or a person who reported a specific issue and logged it in GitHub. After the problem resolution was proposed, further verification will be performed to confirm that the issue was solved. Only after all previous stages are done, the issue will be closed by PM or PO as additional confirmation.

## Approval and Signoff

Approval granted by the Project Manager and the Product Owner will serve to provide the test team with permission to proceed with the test plan presented in this document. Upon receipt of approval, the test team will conduct testing within the scope established in this document. Modifications to the test plan may be needed to ensure that discoveries made during test preparation with the development team are documented. Any modifications will be assessed to determine if they are in scope or if timelines and associated estimates should be modified to address test inclusion and test items considered out of scope but vital to deployment approval.

Attachment A – Acronyms and Abbreviations

| Acronym/Abbreviation | Definition |
| --- | --- |
| CI/CD | Continuous Integration/Continuous Deployment |
| DEV | Development |
| iOS | iPhone Operating System |
| MSA | Mail Speak Application |
| MTP | Master Test Plan |
| PM | Project Manager |
| PMP | Project Management Plan |
| PO | Product Owner |
| QA | Quality Assurance |
| SDLC | Software Development Lifecycle |
| SE | Software Engineer |
| SRS | Software Requirements Document |
| STP | Software Test Plan |
| SWEN | Software Engineering |
| TBD | To Be Determined |
| TDD | Technical Design Document |
| TRTM | Test Requirements Traceability Matrix |
| UAT | User Acceptance Testing |
| UI | User Interface |
| UMGC | University of Maryland Global Campus |
| USPS | United States Postal Service |
| UX | User Experience |

Attachment B - Test Requirements Traceability Matrix

| Requirement ID | Requirement Description | Test Case ID |
| --- | --- | --- |
| 3.1.1.1 | Search past emails using specific date range and keyword | U-008, U-009, U-010, U-011, U-012, U-013, U-014, U-015, U-016, U-017, U-018, U-019, U-020, U-021, U-022, U-023, U-024, U-025, U-026, U-027, U-028, U-029, U-030, U-031 |
| 3.1.2.2 | Chatbot request help documentation | U-007 |
| 3.1.2.4 | Chatbot adding notifications | U-005 |
| 3.1.2.5 | Chatbot removing notifications | U-006 |
| 3.1.2.6 | Chatbot search mail by name | U-002 |
| 3.1.2.7 | Chatbot search mail by date | U-003, U-004 |
| 3.1.2.8 | Chatbot search mail by date range | U-003, U-004 |

Attachment C - Test Cases

C – 1 Unit Test Cases

Unit test scripts are available in GitHub repository.

| Test ID | Test Case Name | Feature Area | Description |
| --- | --- | --- | --- |
| U-001 | Verify Chatbot to homepage | Chatbot | Ensures the chatbot navigates to the home page. |
| U-002 | Verify Search with Parameters multiple keywords | Chatbot | Ensures the chatbot is able to perform search using multiple keywords. |
| U-003 | Verify Search with Parameters Start and End Dates | Chatbot | Ensures the chatbot is able to search with start and end date. |
| U-004 | Verify Search with Parameters Start Exceeds End Dates | Chatbot | Ensures the chatbot is able to handle invalid date input (start exceeds end). |
| U-005 | Verify Create Notification | Chatbot | Ensures the chatbot is able to add a notification. |
| U-006 | Verify Delete Notification | Chatbot | Ensures the chatbot is able to delete notifications. |
| U-007 | Verify performChatFunction Help | Chatbot | Ensures the chatbot returns help documentation upon request |
| U-008 | Search mail by keyword | Search | Ensures mail is searchable by keyword. |
| U-009 | Search mail by keyword null | Search | Ensures search is able to handle null value for keyword. |
| U-010 | Search mail by date range | Search | Ensures functionality of search by date range. |
| U-011 | Search mail by single date | Search | Ensures functionality of search by single date. |
| U-012 | Search mail by date null start | Search | Ensures search is able to handle null value for start date. |
| U-013 | Search mail by date null end | Search | Ensures search is able to handle null value for end date. |
| U-014 | Search mail sender | Search | Ensures mail is searchable by sender. |
| U-015 | Search mail body | Search | Ensures mail is searchable by mail body text. |
| U-016 | Search mail sender and body | Search | Ensures mail is searchable by sender and mail body text. |
| U-017 | Search mail by keyword no match | Search | Ensures no mail pieces are returned when no matching mail pieces exist. |
| U-018 | Search mail by date range start today | Search | Ensures functionality of search by date range, starting today and ending the next day. |
| U-019 | Search mail by date range none | Search | Ensures no mail pieces are returned when no mail pieces within the provided date range exist. |
| U-020 | Search mail sender and body no results | Search | Ensures no mail pieces are returned when no matching mail pieces exist |
| U-021 | Can update a mail piece | Mail Storage | Ensures a locally stored mail piece can be updated. |
| U-022 | It can fetch the latest timestamp | Mail Storage | Ensures the timestamp of the most recent piece of mail in local storage can be retrieved. |
| U-023 | It does not saveMailPiece duplicate mail pieces | Mail Storage | Ensures the system will not save multiple entries of the same mail piece. |
| U-024 | It can retrieve a mail piece by its id | Mail Storage | Ensures mail pieces can be retrieved by id from local storage. |
| U-025 | It can saveMailPiece a new mail piece | Mail Storage | Ensures mail pieces can be fetched, saved, and notifications are updated. |
| U-026 | Can delete a mail piece | Mail Storage | Ensures mail pieces can be deleted, notifications are cleared, and subscriptions are cleared. |
| U-027 | Can delete all mail pieces | Mail Storage | Ensures all mail pieces can be deleted with a single method call. |
| U-028 | It defaults the last timestamp to 90 days ago | Mail Storage | Ensures the latest timestamp defaults to 90 days ago if there are no mail pieces. |
| U-029 | It retrieves matching mail pieces | Mail Storage | Ensures mail pieces that match the given keyword are returned. |
| U-030 | It retrieves all mail pieces when provided a null query | Mail Storage | Ensures all mail pieces are returned if no search parameters are provided. |
| U-031 | It returns an empty list when no mail pieces match | Mail Storage | Ensures an empty list is returned if no mail pieces match the given search parameters. |

C – 2 Functional Test Cases

|  |  |  |
| --- | --- | --- |
| **Test Case 1(a)**: Search Past Emails (date range)  **Description**  This test case will test the user’s ability to search the inbox for emails that are within a given date range.  **Pre-conditions for this test case**   * User is logged into application interface * User has active internet connection * User account has email data available | **SRS Use Case Mapping** | **UC-1(a)** |
|  |
|  |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Select the Search Mail option. |  | Mail Search screen will be displayed. |  |  |  |
| 2(V) | Verify the Search Mail screen displays:  Calendar  Start Date  End Date  Calculation of # of days  Keyword text box |  | All items are accounted for on Search screen. |  |  |  |
| 3 | Select date on calendar to designate the Start Date. |  | Start Date is shown. |  |  |  |
| 4 | Select End Date and select a date on calendar. |  | End Date is shown in End Date field and date range is highlighted on calendar. Duration is calculated and shown on screen. |  |  |  |
| 5 | Select Search button at bottom of screen. |  | Search Result will be shown on next page. |  |  |  |
| 6(V) | Verify the search results displayed are within the date range selected in steps 7 & 8. |  | Results provided are within the appropriate date range. |  |  |  |
| 7 | Select Back arrow at top left of the screen to return to the search page. |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 1(b)**: Search Past Emails (keyword)  **Description**  This test case will test the user’s ability to search the inbox for emails by using a keyword.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection * User account has email data available | **SRS Use Case Mapping** | **UC-1** |
|  |
|  |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Select the Search Mail option. |  | Mail Search screen will be displayed. |  |  |  |
| 2(V) | Verify the Search Mail screen displays:  Calendar  Start Date  End Date  Calculation of # of days  Keyword text box |  | All items are accounted for on Search screen. |  |  |  |
| 3 | Select keyword field on Search screen. |  | Field activates allowing for typing and activates keyboard. |  |  |  |
| 4 | Enter keyword to search on. |  | Keyword is entered in search field and displays keyword results as drop down from keyword field. |  |  |  |
| 5 | Select appropriate mail searched on by keyword. |  | Result will be shown on next page. |  |  |  |
| 6(V) | Verify the keyword is in email shown. |  | Results provided contain keyword. |  |  |  |
| 7 | Select Back arrow at top left of the screen to return to the search page. |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 2**: Chatbot – Show chat suggestions  **Description**  This test case will test the ability for the Chatbot to provide the user suggested commands.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection | **SRS Use Case Mapping** | **UC-2** |
|  |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 2(V) | Verify Chat Support Screen contains:  Back button  Exit button  Time  “How may I assist you?” response  Message field  Function bar displaying:  Search icon  Camera icon  Chatbot icon |  | All items are accounted for on Chat Support screen. |  |  |  |
| 3 | Enter a request other than digest, help, home, logout, notifications, scan, search, settings, or upload and select Send icon. |  | Chatbot returns message “Unable to parse command: XXX. Enter ‘help’ to see a list of available options” |  |  |  |
| 4 | Enter ‘home’ in the message field and select Send icon. |  | Home screen is displayed. |  |  |  |
| 5 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed. |  |  |  |
| 6 | Enter ‘settings’ in the message field and select Send icon. |  | Settings screen will be displayed. |  |  |  |
| 7 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed. |  |  |  |
| 8 | Enter ‘search’ in the message field and select Send icon. |  | Search screen will be displayed. |  |  |  |
| 9 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed. |  |  |  |
| 10 | Enter ‘logout’ in the message field and select Send icon. |  | Login screen will be displayed. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 3**: Chatbot – Request help  **Description**  This test case will test the ability for the Chatbot to provide acceptable help commands.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection | **SRS Use Case Mapping** | **UC-3** |
|  |
|  |
|  |
|  |
|  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case | | | | | | |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 2 | Enter ‘help’ in the message field and select Send icon. |  | Chatbot returns message “Available commands on this page: digest, help, home, logout, notifications, scan, search, settings, or upload.  Note: If you need help with a specific command, enter ‘<command> help’ to view any extra command options.” |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 4**: Chatbot – Notifications  **Description**  This test case will test the ability for the Chatbot to list, add, and removing notifications.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection | **SRS Use Case Mapping** | **UC-4** |
| **UC-5** |
|  |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 2 | Enter ‘notifications’ in the message field and select Send icon. |  | Notifications screen will be displayed. |  |  |  |
| 3 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 4 | Enter ‘notifications help’ in the message field and select Send icon. |  | Chat Support screen will display: “‘notifications’: Navigates to Notifications  ‘add <keyword>’: Adds a notification for the suggested keyword.  ‘delete <keyword>’: Deletes notification for the suggested keyword” |  |  |  |
| 5 | Enter ‘notifications add XXX’ in the message field and select Send icon. |  | Chat Support screen will display: “Notification for XXX has been added” |  |  |  |
| 6(V) | Enter ‘notification’ in the message field and select Send icon. |  | Verify XXX notification has been added to the notifications list. |  |  |  |
| 7 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 8 | Enter ‘notifications delete XXX’ in the message field and select Send icon. |  | Chat Support screen will display: “Notification for XXX has been deleted” |  |  |  |
| 9 | Enter ‘notification’ in the message field and select Send icon. |  | Verify XXX notification has been deleted from the notifications list. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 5**: Chatbot – Search  **Description**  This test case will test the ability for the Chatbot to initiate search using the criteria name, specific date, and specific date range.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection | **SRS Use Case Mapping** | **UC-6** |
| **UC-7** |
| **UC-8** |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 2 | Enter ‘search’ in the message field and select Send icon. |  | Search screen will be displayed. |  |  |  |
| 3 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 4 | Enter ‘search help’ in the message field and select Send icon. |  | Chat Support screen will display: “‘search’: Navigates to the search page  ‘search <date> <date> <keyword>’: Loads advanced search page using entered dates/keywords (optional).  Note: First date is always treated as the start date. Date format is mm/dd/yyyy” |  |  |  |
| 5 | Enter a ‘search <date> <date> <keyword>’ in the message field and select Send icon. |  | Chat Support screen will display: Search page will date and keyword populated. |  |  |  |
| 6(V) | Verify date range and keywords entered in Chatbot populate the search parameters. |  | Search screen is populated with entered search criteria. |  |  |  |
| 7 | Select Search button at bottom of screen. |  | Search Result will be shown on next page. |  |  |  |
| 8(V) | Verify the search results displayed are within the date range and have keyword identified in step 5. |  | Results provided are within the appropriate date range and contain keyword. |  |  |  |
| 9 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 10 | Enter a ‘search <date> <date>’ in the message field and select Send icon. |  | Chat Support screen will display: Search page will date populated. |  |  |  |
| 11(V) | Verify date range entered in Chatbot populate the search parameters. |  | Search screen is populated with entered search criteria. |  |  |  |
| 12 | Select Search button at bottom of screen. |  | Search Result will be shown on next page. |  |  |  |
| 13(V) | Verify the search results displayed are within the date range identified in step 10. |  | Results provided are within the appropriate date range. |  |  |  |
| 14 | Select the Chatbot icon located on function bar at the bottom of screen. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 15 | Enter a ‘search <keyword>’ in the message field and select Send icon. |  | Chat Support screen will display: Search page will keyword populated. |  |  |  |
| 16(V) | Verify keyword entered in Chatbot populate the search parameters. |  | Search screen is populated with entered search criteria. |  |  |  |
| 17 | Select Search button at bottom of screen. |  | Search Result will be shown on next page. |  |  |  |
| 18(V) | Verify the search results displayed contain keyword identified in step 14. |  | Results provided contain keyword searched. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 6**: Gestures  **Description**  This test case will test the ability for Gestures to navigate through application using following actions: left swipe, pull down, right swipe, and swipe up.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection | **SRS Use Case Mapping** | **UC-9** |
| **UC-10** |
| **UC-11** |
| **UC-12** |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Select the Search Mail option. |  | Mail Search screen will be displayed. |  |  |  |
| 2(V) | Verify the Search Mail screen displays:  Calendar  Start Date  End Date  Calculation of # of days  Keyword text box |  | All items are accounted for on Search screen. |  |  |  |
| 3 | Swipe Far Right on device screen. |  | Screen should navigate to the previous screen viewed before swipe. |  |  |  |
| 4(V) | Verify screen displayed is the previous screen that was viewed. |  | Screen is previous screen identified. |  |  |  |
| 5 | Select the Notification icon located on function bar at the bottom of screen. |  | Notification screen will be displayed. |  |  |  |
| 6 | Swipe Left on device screen. |  | Screen should navigate to the notification manage screen. |  |  |  |
| 7(V) | Verify manage screen is displayed. |  | Manage screen will display the ability to Add notification by entering keyword or Delete current notifications. |  |  |  |
| 8 | Swipe Right on device screen. |  | Screen will navigate to the Main Notification list screen. |  |  |  |
| 9(V) | Verify screen displayed is the Main Notification list screen that was viewed. |  | Main Notification list screen is displayed. |  |  |  |
| 10 | Execute a search to display search results. |  | Search results will display. |  |  |  |
| 11 | Pull down from top of device within the search results. |  | Application will query and update results to include any new mail pieces acquired. |  |  |  |
| 12(V) | Verify new piece is added to the current results displayed if applicable. |  | Verify mail piece. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 7**: Voice Driven – Login, Menu and Settings  **Description**  This test case will test the user’s ability to login and access menu functionality using voice driven functionality.  **Pre-conditions for this test case**   * User is logged out of application interface * User has active internet connection * User has voice driven accessibility activated through OS settings | **SRS Use Case Mapping** | **UC-13** |
| **UC-14** |
| **UC-15** |
| **UC-16** |
| **UC-25** |
| **UC-26** |
| **UC-27** |
| **UC-28** |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Hold down on screen to activate Voice Driven functionality. (iPhone anywhere on screen, Android choose Voice Activation Icon) |  | Voice Driven functionality is activated. |  |  |  |
| 2(V) | Verify Voice Driven functionality is available on screen by speaking ‘Show labels’ on login screen within the application. |  | Labels are displayed. |  |  |  |
| 3 | Speak number associated to the E-mail Address field on login screen. |  | Cursor moves focus to E-mail field. |  |  |  |
| 4 | Speak email address as written. (with spelling out unique name, stating at for @, and saying outlook.com) |  | Email is entered into E-mail field. |  |  |  |
| 5 | Speak ‘Show labels’ and speak number associated to Password |  | Cursor moves focus to Password field. |  |  |  |
| 6 | Speak password (spell out). |  | Password is entered into Password field. |  |  |  |
| 7 | Speak ‘Show labels’ and speak number associated to Terms and Conditions. |  | Terms and Conditions dialog is displayed. |  |  |  |
| 8(V). | Verify Terms and Conditions dialog box is displayed, and entire information can be read by scrolling. |  | Terms and Conditions can be read. |  |  |  |
| 9 | Speak ‘Tap Close’ in Terms and Conditions dialog box. |  | Terms and Conditions dialog box closes. |  |  |  |
| 10 | Speak ‘Show labels’ and speak number associated to Privacy Policy. |  | Privacy Policy dialog is displayed. |  |  |  |
| 11(V) | Verify Privacy Policy dialog box is displayed, and entire information can be read by scrolling. |  | Privacy Policy can be read. |  |  |  |
| 12 | Speak ‘Tap Close’ in Privacy Policy dialog box. |  | Privacy Policy dialog box closes. |  |  |  |
| 13 | Speak ‘Show labels’ and speak number associated with the check box for Terms and Conditions and Privacy Policy. |  | Checkbox will show check in box. |  |  |  |
| 14 | Speak ‘Tap Retrieve Mail’. |  | Application logs in and Home Screen is displayed. |  |  |  |
| 15 | Speak ‘Show labels’ and speak number associated to the Menu (Hamburger icon) on the top right. |  | Menu will expand. |  |  |  |
| 16(V) | Verify Menu is displayed showing Settings and Logout. |  | Menu displays with settings and logout on the drop down. |  |  |  |
| 17 | Speak ‘Settings’. |  | A number should be displayed on each item that Settings can be opened. |  |  |  |
| 18 | Speak the number of the Settings option located in the drop-down menu. |  | Settings Screen will be displayed. |  |  |  |
| 19 | Speak ‘Terms and Conditions’. |  | Terms and Conditions dialog box will be displayed. |  |  |  |
| 20 | Speak ‘Tap Close’ |  | Terms and Conditions dialog box closes. |  |  |  |
| 21 | Speak ‘Privacy Policy’. |  | Privacy Policy dialog box will be displayed. |  |  |  |
| 22 | Speak ‘Tap Close’. |  | Privacy Policy dialog box closes. |  |  |  |
| 23 | Speak ‘Logout’. |  | User will be logged out of application. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 8**: Voice Driven – Chatbot  **Description**  This test case will test the user’s ability to open the chatbot using voice driven functionality.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection * User has voice driven accessibility activated through OS settings * User has activated voice driven accessibility with application | **SRS Use Case Mapping** | **UC-23** |
| **UC-24** |
|  |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Speak the word Chatbot from within the application. |  | Chatbots Screen will be displayed. |  |  |  |
| 2(V) | Verify Chatbot Screen is displayed. |  | Chat Support screen will be displayed showing “How may I assist you?”. |  |  |  |
| 3 | Speak ‘type Notifications’. |  | Notifications will display in message field |  |  |  |
| 4(V) | Verify the word Notifications is displayed in message field. |  | Notification is ready to be sent. |  |  |  |
| 5 | Speak ‘Press Send’. |  | Message is submitted to the Chatbot. |  |  |  |
| 6(V) | Notification Screen appears. |  | Notification screen is navigated to. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 9**: Voice Driven – Search and Navigate Mail  **Description**  This test case will test the user’s ability to search and manipulate viewing of email functions using voice driven functionality.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection * User has voice driven accessibility activated through OS settings * User has activated voice driven accessibility with application | **SRS Use Case Mapping** | **UC-17** |
| **UC-18** |
| **UC-19** |
| **UC-34** |
| **UC-35** |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Select Search Mail. |  | Mail Search screen will be displayed. |  |  |  |
| 2(V) | Verify the Search Mail screen displays:  Calendar  Start Date  End Date  Calculation of # of days  Keyword text box |  | All items are accounted for on Search screen. |  |  |  |
| 3 | Speak number of Start Date to search. |  | Numbers are displayed for all numbers in the calendar that match requested number. |  |  |  |
| 4 | Speak selected Date number. |  | Start Date field populates with selected date. |  |  |  |
| 5 | Speak number of End Date to search. |  | Numbers are displayed for all numbers in the calendar that match requested number. |  |  |  |
| 4 | Speak selected Date number. |  | End Date field populates with selected date. |  |  |  |
| 5 | Speak the word ‘Search’. |  | Numbers are displayed for all functions available with word Search. |  |  |  |
| 6 | Speak number of Search button (6). |  | Search Result will be shown on next page. |  |  |  |
| 7 | Speak Sender name of email to open. |  | Sender will be identified with numbers for all mail items that match requested Sender name. |  |  |  |
| 8(V) | Verify all email with selected Sender name with icons show a numerical value. |  | Sender emails have identification number to allow user to use one of the numbers identified. |  |  |  |
| 9 | Speak the word 1. |  | Email will open. |  |  |  |
| 10(V) | Verify selected email is opened. |  | Email is opened for viewing. |  |  |  |
| 11 | Speak the word ‘Back’. |  | Application will navigate back to the Search Results screen. |  |  |  |
| 12(V) | Verify the Back navigates user Back to previous screen. |  | Previous Screen is displayed. |  |  |  |
| 13 | Speak the work ‘Home’ |  | Application will navigate to the Home Screen. |  |  |  |
| 14(V) | Verify the Home screen is displayed. |  | Home Screen is available. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 10**: Voice Driven – Notifications  **Description**  This test case will test the user’s ability to manipulate the notifications functionality using voice driven functionality.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection * User has voice driven accessibility activated through OS settings * User has activated voice driven accessibility with application | **SRS Use Case Mapping** | **UC-20** |
| **UC-21** |
| **UC-22** |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Speak ‘Tap Notifications’. |  | Numbers are displayed for all functions available with word Notifications. |  |  |  |
| 2 | Speak number of Notifications main button (1). |  | Notification screen is displayed. |  |  |  |
| 3(V) | Verify Notification Screen is displayed with list of active notifications. |  | Notification and list appear. |  |  |  |
| 4 | Speak ‘Tap Manage’. |  | Notification screen changes to the Manage Tab. |  |  |  |
| 5 | Speak ‘Tap Keyword’. |  | Cursor will appear in Keyword dialog box. |  |  |  |
| 6 | Speak ‘Type XXX’. |  | Text will be entered in Keyword field. |  |  |  |
| 7 | Speak ‘Tap Add’. |  | Notification is added to the list of notifications. |  |  |  |
| 8(V) | Verify notification was added to the available notification list. |  | Notification is displayed. |  |  |  |
| 9 | Speak ‘Show labels’ and speak number associated to Notification to delete. |  | Notification selected is deleted from notification list. |  |  |  |
| 10(V) | Verify notification is removed from notification list. |  | Notification has been removed. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 11**: Voice Driven – Digest  **Description**  This test case will test the user’s ability to dismiss a notification using voice driven functionality.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection * User has voice driven accessibility activated through OS settings * User has activated voice driven accessibility with application | **SRS Use Case Mapping** | **UC-28** |
| **UC-29** |
| **UC-30** |
|  |
|  |
|  |
|  |

| Test Case | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | From Home Screen, speak ‘Daily Digest’. |  | Application will begin process of retrieving information for the daily receipt of mail. |  |  |  |
| 2(V) | Verify new mail is displayed after process has completed. |  | New mail is shown on the Mail Screen to be navigated through. |  |  |  |
| 3 | Speak ‘Show labels’ and speak number associated to the Next button. |  | Next piece of retrieved mail is displayed. |  |  |  |
| 4(V) | Verify next piece of mail is displayed and count is updated correctly (i.e., 2/3) |  | Next piece of mail is shown with the correct count on screen. |  |  |  |
| 5 | Speak ‘Show labels’ and speak number associated to the Previous button. |  | Previous piece of retrieved mail is displayed. |  |  |  |
| 6(V) | Verify previous piece of mail is displayed and count is updated correctly (i.e., 2/3) |  | Previous piece of mail is shown with the correct count on screen. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 12**: Voice Driven – Upload/Scan  **Description**  This test case will test the user’s ability to upload and scan a piece of mail using voice driven functionality.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection * User has voice driven accessibility activated through OS settings * User has activated voice driven accessibility with application | **SRS Use Case Mapping** | **UC-32** |
| **UC-33** |
|  |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | From Home Screen, speak ‘Scan Mail’. |  | Application will open Camera. |  |  |  |
| 2(V) | Verify Camera icons show a 1 and 2. |  | Camera icons have identification number to allow user to use one of the two icons available. |  |  |  |
| 3 | Speak the word 1. |  | Camera will open from the Scan Mail button located on the Home screen. |  |  |  |
| 4(V) | Verify the Camera opens up ready to take picture. |  | Camera is available. |  |  |  |
| 5 | Speak the word Back. |  | Application will navigate back to the Home screen. |  |  |  |
| 6(V) | Verify Home Screen is displayed. |  | Application Home Screen is available. |  |  |  |
| 7 | Speak the word Camera from within the application. |  | Camera icons will be identified. |  |  |  |
| 8(V) | Verify Camera icons show a 1 and 2. |  | Camera icons have identification number to allow user to use one of the two icons available. |  |  |  |
| 9 | Speak the word 2. |  | Camera should be opened from the Camera icon located on function bar at the bottom of screen. |  |  |  |
| 10(V) | Verify the Camera opens up ready to take picture. |  | Camera is available. |  |  |  |
| 11 | Speak ‘Take Picture’. |  | Picture is taken and saved to the Gallery. |  |  |  |
| 12 | Speak ‘Tap Ok’. |  | Application navigates back to Home screen. |  |  |  |
| 13 | Speak ‘Upload Mail’. |  | Directory where Scan Mail is saved is displayed. |  |  |  |
| 14 | Select mail to upload. |  | Mail is selected. |  |  |  |
| 15(V) | Verify mail is uploaded by executing Search. |  | Verify uploaded mail item is available. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Test Case 13**: Reading Mode  **Description**  This test case will test the system’s ability to read UI elements to the user.  **Pre-conditions for this test case**   * User is logged in to application interface * User has active internet connection * User has screen reading accessibility activated through OS settings * User has sound on | **SRS Use Case Mapping** | **UC-36** |
| **UC-37** |
|  |
|  |
|  |
|  |
|  |

| Test Case | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Step Description | Data/Value | Expected Result | Actual Result  (if different from expected) | Pass/Fail | Defect # |
| 1 | Drag one finger around the screen to the top and select the header |  | System reads UI elements one by one until dragging is stopped |  |  |  |
| 2(V) | Verify screen header is read by the system |  | System reads screen header and stops |  |  |  |
| 3 | Triple tap two fingers (Android) / Two finger swipe up (iOS) |  | System starts reading all elements on the screen |  |  |  |
| 4(V) | Verify that all elements were announced |  | System reads all UI elements and stops the announcement |  |  |  |