A picture containing logo

Description automatically generated

**Test Report**

University of Maryland Global Campus

SWEN 670 – Team A

Fall Semester

Version 1.0

November 5, 2022

Document Control

Document Information

|  |  |
| --- | --- |
| © | Information |
| Document Id | MailSpeak Test Report |
| Document Owner | UMGC SWEN 670 TEAM A |
| Issue Date | November 5, 2022 |
| Last Saved Date | November 5, 2022 |
| File Name | USPS\_Informed\_Delivery\_App\_Enhancements\_TR\_Team\_A.docx |

Document History

|  |  |  |
| --- | --- | --- |
| Version | Issue Date | Changes |
| 1.0 | 10/5/2022 | Initial Version |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[1 Executive Summary 1](#_Toc118556420)

[2 Introduction 3](#_Toc118556421)

[2.1 Purpose 3](#_Toc118556422)

[2.2 Scope 3](#_Toc118556423)

[2.2.1 Project Documents 3](#_Toc118556424)

[3 Testing 4](#_Toc118556425)

[3.1 Test Strategy 4](#_Toc118556426)

[3.2 Tests Conducted 4](#_Toc118556427)

[4 Functional Tests Execution Findings 5](#_Toc118556428)

[4.1 Functional Test Results Summary 5](#_Toc118556429)

[4.2 Defect Summary 6](#_Toc118556430)

[4.2.1 Test Case 03 (§ 5.3) Defect 6](#_Toc118556431)

[4.2.2 Test Case 04 (§ 5.4) Defect 6](#_Toc118556432)

[4.2.3 Test Case 05 (§ 5.5) Defect 6](#_Toc118556433)

[4.2.4 Test Case 06 (§5.6) Defect 7](#_Toc118556434)

[4.2.5 Test Case 07 (§5.7) Defect 7](#_Toc118556435)

[4.2.6 Test Case 08 (§5.8) Defect 7](#_Toc118556436)

[5 Test Results and Summary 8](#_Toc118556437)

[5.1 Test Case 01: QR Code Reading from Mail Piece 8](#_Toc118556438)

[5.2 Test Case 02: Embedded Links Navigate User to the Browser 8](#_Toc118556439)

[5.3 Test Case 03: Google Assistant Search for a Mailpiece 9](#_Toc118556440)

[5.4 Test Case 04: Google Assistant Add a new Sender/Keyword for Notifications 9](#_Toc118556441)

[5.5 Test Case 05: Google Assistant Add an existing Sender/Keyword for Notifications 10](#_Toc118556442)

[5.6 Test Case 06: Google Assistant Opens the most recent Digest email 10](#_Toc118556443)

[5.7 Test Case 07: Google Assistant Opens the notifications page 11](#_Toc118556444)

[5.8 Test Case 08: Google Assistant Opens Application 11](#_Toc118556445)

[5.9 Test Case 09: Add Sender/Keyword Notification list visually/manually 12](#_Toc118556446)

[5.10 Test Case 10: Delete Sender/Keyword Notification list visually/manually 12](#_Toc118556447)

[5.11 Test Case 11: Receive Notification 13](#_Toc118556448)

[5.12 Test Case 12: View Notification visually/manually 13](#_Toc118556449)

[5.13 Test Case 13: Clear Notification 14](#_Toc118556450)

[5.14 Test Case 14: Clear All Notification 14](#_Toc118556451)

[5.15 Test Case 15: Display QR code and click Mail View able link 15](#_Toc118556452)

[5.16 Test Case 16: Mail View Display URL and a clickable link 15](#_Toc118556453)

[5.17 Test Case 17: Mail View Display text from mail 16](#_Toc118556454)

[5.18 Test Case 18: Mail View Display verify contact information 17](#_Toc118556455)

[5.19 Test Case 19: Analytics Dashboard confirms screen time 17](#_Toc118556456)

[5.20 Test Case 20: Analytics Dashboard displays link metrics 18](#_Toc118556457)

[5.21 Test Case 21: Analytics Dashboard displays searches performed 18](#_Toc118556458)

[5.22 Test Case 22: Mail View Call Phone Number 19](#_Toc118556459)

[5.23 Test Case 23: Mail View Text Phone Number 19](#_Toc118556460)

[5.24 Test Case 24: Mail View Email provided email address. 20](#_Toc118556461)

[6 Appendix 21](#_Toc118556462)

[6.1 Test Result Summary and STP Report Key 21](#_Toc118556463)

# Executive Summary

The purpose of this report is to document the strategies and results of the software testing that were conducted during the development of the MailSpeak mobile application as well as provide an overall assessment of latest application release.

During this execution of this project, the team focused on two testing strategies: automated and manual. The automated tests were controlled by the continuous integration (CI)/continuous delivery (CD) pipeline, were used to control the integration of new code during pull requests (PR) to ensure the main branch was always maintained in a buildable state. When these tests reported a failure, the PRs were automatically sent back to the submitting developer to review and address the failure.

In reference to manual testing, Team A had a passing percentage of 75%. There was a total of 24 tests planned and conducted, 18 of which passed. The main features that are included in the passing tests were mail piece processing (OCR), notifications, mail piece display, analytics integration. The remaining 25% of tests which failed, were related to the implementation of the Google Assistant integration.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST CASE NUMBER​ | TEST REPORT PARAGRAPH​ | FEATURE CATEGORY​ | TEST  NAME​ | RESULT​ |
| 03​ | 5.3​ | Google  Assistant​ | Google Assistant Search for a Mail piece​ | **Fail -inconclusive**​ |
| 04​ | 5.4​ | Google  Assistant​ | Google Assistant Add a new Sender/Keyword for Notifications​ | **Fail -inconclusive**​ |
| 05​ | 5.5​ | Google  Assistant​ | Google Assistant Add an existing Sender/Keyword for Notifications​ | **Fail -inconclusive**​ |
| 06​ | 5.6​ | Google  Assistant​ | Google Assistant Opens the most recent Digest email​ | **Fail -inconclusive**​ |
| 07​ | 5.7​ | Google  Assistant​ | Google Assistant Opens the Notifications Page​ | **Fail -inconclusive**​ |
| 08​ | 5.8​ | Google  Assistant​ | Google Assistant Opens Application​ | **Fail -inconclusive**​ |

Although, the team encountered a failure rate of 25% of their tests. These failures were due to the inability to test the integration of the feature due to the inability to move forward because of an issue obtaining Google Play store approval which was needed to move forward with the testing of that feature. Fortunately, these failures would not prohibit users from using or interacting with the MailSpeak application. They only impact the ability for MailSpeak to directly interact with the Google Assistant.

Our assessment of the latest build is that the application is stable. It is critical that the issues surrounding the Google Play and Apple Appstore are addressed in order to accommodate a formal release plan, but once corrected, the Google Assistant tests could be conducted in parallel with a limited a limited release before any formal launch.

# Introduction

## Purpose

This test report outlines the scope, strategy, and findings of the testing process for the MailSpeak application. The MailSpeak application is a mobile application that attempts to make the USPS Informed Delivery system more accessible for the visually impaired. This document will help the reader understand the developers’ overall testing strategy and code coverage established by implementing various types of tests. Additionally, this document will outline any findings discovered by the developers while performing said tests.

This test report is a critical document since it indicates if the application meets the agreed upon system requirements. Since requirements are always changing in an Agile development cycle, it is important to note that successive iterations of this application and document will build upon the foundation of this document. The listed test procedures show the reader how bugs are mitigated by implementing thorough and successively broader tiers of tests. Stakeholders should use this document to decide if the listed application features are working as intended and that the work meets their expectations.

## Scope

The scope of this Test Report shall include the test strategy, tests conducted, test findings, and an overall description of the test suite created by the developers. Out of scope components include, but are not limited to, usability testing, penetration testing, and performance testing. This Test Report shall be used by stakeholders to verify that the MailSpeak application meets their expectations and requirements.

### Project Documents

The following documents are included in the project's software documentation package:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Document** | **Version** | **Date** |
| 1 | Project Management Plan (PMP) | 1.2 | 9-17-2022 |
| 2 | Software Requirements Specification (SRS) | 1.2 | 10-29-2022 |
| 3 | Technical Design Document (TDD) | 1.2 | 10-29-2022 |
| 4 | Software Test Plan (STP) | 1.1 | 10-29-2022 |
| 5 | Programmers Guide (PG) | 1.1 | 11-05-2022 |
| 6 | Deployment and Operations (DevOps) | 1.1 | 11-05-2022 |
| 7 | User Guide (UG) | 1.0 | 11-05-2022 |
| 8 | Test Report (TR) | 1.0 | 11-05-2022 |

# Testing

## Test Strategy

Ongoing testing and verification of results will be part of each software engineer’s day to day work on implementing the features. The application will be formally tested when it has been determined that the core application functionality is ready to test.

The application will have unit tests for functions that can be proven to work when the output and input can be known. Additionally, all unit, functional, and end-to-end tests are required to be passing for new code to be successfully merged into the existing codebase. This ensures that the main development branch is always in a buildable and releasable state. Thus, the unit tests must pass when checking in code so it can be assumed that if code is in the main development branch all unit tests have passed.

Many of the features are difficult to test with automated unit tests, such as verifying test mail pieces have been properly scanned and are available for viewing in the application. Visual confirmation of proper operation will be utilized in this case. Because of the variability of mail pieces, multiple mail pieces, on multiple dates, will be utilized to verify the test cases. Additionally, multiple email accounts will be tested, as noticeable differences do occur between email account providers such as their organization of Multipurpose Internet Mail Extension (MIME) parts such as “text/html” and “image/jpeg”.

All tests must be completed on both iOS and Android, but where applicable, some requirements are only necessary for Android. Manual testing will be completed after features are considered “coding complete”, and any defects will be recorded in this document as well as documented on GitHub for review by the team. These defects will be categorized and fixed in order of priority to be decided by the Product Owner.

## Tests Conducted

The test suite for the Mailspeak application consists of 24 total test cases, described in the “Test Results and Summary” tables of Section 5. A list of tests and their associated description in the Software Test Plan (STP) document is available in the Appendix.

Descriptions of each test case are included in the “Description" column and configurations required are in the “Prerequisites” column. Test Steps include what the application is being asked to do to test the case. Expected results are what is expected to pass the test. If a defect is observed and the test fails, actual results will be reported, and a description will be provided to try to explain why it is a defect.

Status of the test is reported in each test as either “**PASS**” or “**FAIL**”, and for tests unable to be run, they will be reported as “**FAIL – INCONCLUSIVE**”.

Additional information includes the test case ID and name, along with any notes.

# Functional Tests Execution Findings

## Functional Test Results Summary

Of the 24 tests, 18 passed and 6 failed due to not being able to test. A summary is provided in the table below.

| **TEST CASE NUMBER** | **TEST NAME** | **RESULT** |
| --- | --- | --- |
| 01 | QR Code Reading from Mail Piece | **PASS** |
| 02 | Embedded Links Navigate User to the Browser | **PASS** |
| 03 | Google Assistant Search for a Mailpiece | **FAIL - INCONCLUSIVE** |
| 04 | Google Assistant: Add new Keyword – Notifications | **FAIL - INCONCLUSIVE** |
| 05 | Google Assistant: Add existing Keyword - Notifications | **FAIL - INCONCLUSIVE** |
| 06 | Google Assistant Opens the most recent Digest email | **FAIL - INCONCLUSIVE** |
| 07 | Google Assistant Opens the Notifications Page | **FAIL - INCONCLUSIVE** |
| 08 | Google Assistant Opens Application | **FAIL - INCONCLUSIVE** |
| 09 | Add Sender/Keyword Notification list visually/manually | **PASS** |
| 10 | Delete Sender/Keyword Notification list visually/manually | **PASS** |
| 11 | Receive Notification | **PASS** |
| 12 | View Notification visually/manually | **PASS** |
| 13 | Clear Notification | **PASS** |
| 14 | Clear All Notification | **PASS** |
| 15 | Display QR code and click Mail View able link | **PASS** |
| 16 | Mail View Display URL and a clickable link | **PASS** |
| 17 | Mail View Display text from mail | **PASS** |
| 18 | Mail View Display verify contact information | **PASS** |
| 19 | Analytics Dashboard confirms screen time | **PASS** |
| 20 | Analytics Dashboard displays link metrics | **PASS** |
| 21 | Analytics Dashboard displays searches performed | **PASS** |
| 22 | Mail View Call Phone Number | **PASS** |
| 23 | Mail View Text Phone Number | **PASS** |
| 24 | Mail View Email provided email address. | **PASS** |

## Defect Summary

As with any projects on a tight deadline and with demanding requirements, there are always some findings from the tests which fail. Each finding has a test case which failed, the requirement which failed, the expected output, actual output, defect and a screenshot if applicable. This project is no exception, and there are some failures that will be documented in the findings document below. One major finding is Google Assistant, unfortunately due to the way applications are required to be deployed and reviewed to the Google Play Store before it can work with Google Assistant, this functionality was unable to be tested directly with Google Assistant in the allotted time frame. These are reported as findings to provide insight to any teams that come after us, however these tests neither passed nor failed, they were unable to be run. These failed test cases all involve Google Assistant, however all functionality that Google Assistant integration achieves can also be completed by either using the application normally or by using the chatbot. As such the risk for deploying without this functionality operating properly is low.

### Test Case 03 (§ 5.3) Defect

**Test Case:** 03

**Failing Requirement:**The user will be able to search for Keywords via Google Assistant.

**Expected Output**: The application opens the search page with Test Mailpiece filled in, in the search box.

**Actual Output**: Google Assistant did not recognize the application.

**Defect:**The application did not search for a MailPiece via Google Assistant.

**Screenshot of failure:** N/A

### Test Case 04 (§ 5.4) Defect

**Test Case:** 04

**Failing Requirement:**The user will be able to add a new Sender/Keyword to Notifications via Google Assistant.

**Expected Output**: The application opens the notifications page and adds the Plumbing keyword to the Sender/Keyword list.

**Actual Output**: Google Assistant did not recognize the application.

**Defect:**The user was not able to and a new Sender/Keyword to the notifications list via Google Assistant.

### Test Case 05 (§ 5.5) Defect

**Test Case:** 05

**Failing Requirement:**A duplicate item will not be set if a sender/keyword is attempted to be added where it already exists via Google Assistant.

**Expected Output**: The application does not add plumbing a second time to the notifications keywords list.

**Actual Output**: Google Assistant did not recognize the application.

**Defect:**The application did attempt to even add the keyword.

**Screenshot of failure:** N/A

### Test Case 06 (§5.6) Defect

**Test Case:** 06

**Failing Requirement:**The user can open the most recent digest email from Google Assistant.

**Expected Output**: The application opens the mail digest page and shows the most recent digest message.

**Actual Output**: Google Assistant did not recognize the application.

**Defect:**The application did not open the digest page via Google Assistant.

**Screenshot of failure:** N/A

### Test Case 07 (§5.7) Defect

**Test Case:** 07

**Failing Requirement:**The user will be able to open the notifications page using Google Assistant.

**Expected Output**: The application opens the manage notifications page

**Actual Output**: Google Assistant did not recognize the application.

**Defect:**The application did not open the notifications page via Google Assistant.

**Screenshot of failure:** N/A

### Test Case 08 (§5.8) Defect

**Test Case:** 08

**Failing Requirement:**The user will be able to open the application via Google Assistant

**Expected Output**: The application opens

**Actual Output**: Google Assistant did not recognize the application.

**Defect:**The application was not opened via Google Assistant.

**Screenshot of failure:** N/A

**Screenshot of failure:** N/A

# Test Results and Summary

## Test Case 01: QR Code Reading from Mail Piece

|  |  |
| --- | --- |
| **Description** | In this test, the user will open a digest with a single mail piece that has a QR code with a URL embedded in it and the application will navigate to the correct page. |
| **Requirements** | This test case validates that the application can read a QR code with a URL in it and make it selectable. |
| **Prerequisites** | A mail digest with mail containing a QR code with a valid link is received or scanned in. |
| **Test Steps** | 1. The customer opens the email digest message with the previous mail piece in it. 2. The user clicks on the QR code in the body of the email.link from the QR code. |
| **Expected Output** | The application opens the browser and navigates the user to that page |
| **Assumptions** | None. |
| **Result** | **PASS** |

## Test Case 02: Embedded Links Navigate User to the Browser

|  |  |
| --- | --- |
| **Description** | In this test, the user takes an email that has links in the body, and will be able to navigate to those links. |
| **Requirements** | This test case validates that all links will be linkifed in the body of the message and the user will be able to open the browser for the link. |
| **Prerequisites** | Application running, user login to the app with their email address and an email received from a sender that is in the notification list. |
| **Test Steps** | 1. The customer opens a piece of email that contains a link. 2. The user clicks on the link |
| **Expected Output** | The application opens the browser and navigates the user to that link. |
| **Assumptions** | User has an email account |
| **Result** | **PASS** |

## Test Case 03: Google Assistant Search for a Mailpiece

|  |  |
| --- | --- |
| **Description** | In this test, the user will search for a mail piece via keyword using Google Assistant and will not provide a date range. |
| **Requirements** | This test case validates that the user can search for a mailpiece without a date range via Google Assistant |
| **Prerequisites** | The application is running on an Android device.  The application has Google Assistant installed. |
| **Test Steps** | 1. The user says “Ok Google Search for Test Mailpiece On Mailspeak”. |
| **Expected Output** | The application opens the search page with Test Mailpiece filled in, in the search box |
| **Assumptions** |  |
| **Result** | **FAIL - INCONCLUSIVE**  Test unable to be run, Google Assistant not recognizing Mailspeak as it has not been approved in the Play Store and Google’s Test Group does not appear to work. |

## Test Case 04: Google Assistant Add a new Sender/Keyword for Notifications

|  |  |
| --- | --- |
| **Description** | In this test, the user will add a Sender/Keyword to receive notifications for it via Google Assistant. |
| **Requirements** | This test case validates that the user can add a Sender/Keyword to the notifications list via Google Assistant |
| **Prerequisites** | The application is running on an Android device.  The application has Google Assistant installed. |
| **Test Steps** | 1. The user says “Ok Google Add Plumbing to MailSpeak” |
| **Expected Output** | The application opens the notifications page and adds the Plumbing keyword to the Sender/Keyword list |
| **Assumptions** |  |
| **Result** | **FAIL - INCONCLUSIVE**  Test unable to be run, Google Assistant not recognizing mailspeak as it has not been approved in the Play Store and Google’s Test Group does not appear to work. |

## Test Case 05: Google Assistant Add an existing Sender/Keyword for Notifications

|  |  |
| --- | --- |
| **Description** | In this test, the user will add a Sender/Keyword that already exists to the notifications list via Google Assistant. |
| **Requirements** | This test case validates that a duplicate item will not be set if a sender/keyword is attempted to be added where it already exists. |
| **Prerequisites** | The application is running on an Android device.  The application has Google Assistant installed.  The “plumbing” keyword has already been added. |
| **Test Steps** | 1. The user says “Ok Google Add Plumbing to MailSpeak” |
| **Expected Output** | The application does not add plumbing a second time to the notifications keywords list. |
| **Assumptions** |  |
| **Result** | **FAIL - INCONCLUSIVE**  Test unable to be run, Google Assistant not recognizing mailspeak as it has not been approved in the Play Store and Google’s Test Group does not appear to work. |

## Test Case 06: Google Assistant Opens the most recent Digest email

|  |  |
| --- | --- |
| **Description** | In this test, the user will open the most recent digest mail using a voice command on an Android device. |
| **Requirements** | This test case validates that the user will be able to open the most recent digest email using Google Assistant. |
| **Prerequisites** | The application is running on an Android device.  The application has Google Assistant installed. |
| **Test Steps** | 1. Say “Ok Google Get Digest on MailSpeak |
| **Expected Output** | The application opens the mail digest page and shows the most recent digest message. |
| **Assumptions** |  |
| **Result** | **FAIL - INCONCLUSIVE**  Test unable to be run, Google Assistant not recognizing mailspeak as it has not been approved in the Play Store and Google’s Test Group does not appear to work. |

## Test Case 07: Google Assistant Opens the notifications page

|  |  |
| --- | --- |
| **Description** | In this test, the user will open the notifications page from Google Assistant. |
| **Requirements** | This test case validates that the user will be able to open the notifications page using Google Assistant. |
| **Prerequisites** | The application is running on an Android device.  The application has Google Assistant installed. |
| **Test Steps** | 1. The user says "Ok, Google Open Notifications On MailSpeak. |
| **Expected Output** | The application opens the manage notifications page |
| **Assumptions** |  |
| **Result** | **FAIL - INCONCLUSIVE**  Test unable to be run, Google Assistant not recognizing mailspeak as it has not been approved in the Play Store and Google’s Test Group does not appear to work. |

## Test Case 08: Google Assistant Opens Application

|  |  |
| --- | --- |
| **Description** | In this test, the user will open the app using Google Assistant |
| **Requirements** | This test case validates that the user will be able to open the application via Google Assistant. |
| **Prerequisites** | The application is running on an Android device.  The application has Google Assistant installed. |
| **Test Steps** | 1. The user says "Ok, Google Open MailSpeak |
| **Expected Output** | Google Assistant opens the Application. |
| **Assumptions** |  |
| **Result** | **FAIL - INCONCLUSIVE**  Test unable to be run, Google Assistant not recognizing mailspeak as it has not been approved in the Play Store and Google’s Test Group does not appear to work. |

## Test Case 09: Add Sender/Keyword Notification list visually/manually

|  |  |
| --- | --- |
| **Description** | In this test, the user will visually/manually add a Specific sender/Keyword to receive a notification. |
| **Requirements** | This test case validates that the user can add a Sender/Keyword to the notifications list manually/visually. |
| **Prerequisites** | Application running on Android device and app is logged in with customer's email address |
| **Test Steps** | 1. The Customer selects the Notification button from the home page 2. The Customer selects the 'Manage' button under 'Notifications Center.' 3. The Customer then selects the 'Add' button under the 'Manage' section 4. The Customer types in the specific sender/keyword they want to be notified for 5. The Customer clicks on the save button next to the text box specified sender information |
| **Expected Output** | The application adds the typed and saved sender/keyword into the notification list. |
| **Assumptions** | User has an email account |
| **Result** | **PASS** |

## Test Case 10: Delete Sender/Keyword Notification list visually/manually

|  |  |
| --- | --- |
| **Description** | In this test, the user will visually/manually Delete a Specific Sender/Keyword from the existing notification lists. |
| **Requirements** | This test case validates that the user can delete a previously existing Sender/Keyword from the notifications list manually/visually. |
| **Prerequisites** | Application running on Android device and app is logged in with customer's email address. |
| **Test Steps** | 1. The Customer selects the Notification icon from the home page 2. The Customer selects the Manage button 3. The Customer then selects the clear button next to the item to be deleted. |
| **Expected Output** | The application deletes the sender/keyword next to the pushed 'clear' button. It is no longer on the notification list. |
| **Assumptions** | User has an email account |
| **Result** | **PASS** |

## Test Case 11: Receive Notification

|  |  |
| --- | --- |
| **Description** | In this test, a sample email will be sent to the user's phone with a sender id from within the notification list. |
| **Requirements** | This test case validates the application notifying the user as any of the senders/keywords in the notification list to arrive. |
| **Prerequisites** | Application running, user logged in to the app with their email address, and there is a minimum of one notification list set. |
| **Test Steps** | A test email is sent to the Customer from a sender's name within the notification list. |
| **Expected Output** | The app sends notifications as the user receives an email from a sender/keyword form within the notification list. |
| **Assumptions** | User has an email account |
| **Result** | **PASS** |

## Test Case 12: View Notification visually/manually

|  |  |
| --- | --- |
| **Description** | In this test, the user will be viewing an email for which the user received a notification. |
| **Requirements** | This test case validates the notification properly, opens the email notified as the user clicks on the notification. |
| **Prerequisites** | Application running, user login to the app with their email address and an email received from a sender that is in the notification list. |
| **Test Steps** | 1. A test email is sent to the Customer from a sender's name within the notification list. 2. A notification alert from the application 3. The Customer clicks on the popped-up notification |
| **Expected Output** | A click on the popped-up notification opens the email, and the customer can go through the email. |
| **Assumptions** | User has an email account |
| **Result** | **PASS** |

## Test Case 13: Clear Notification

|  |  |
| --- | --- |
| **Description** | In this test, the user will be deleting a specific notification from the existing notification lists. |
| **Requirements** | This test case validates that the user can delete a previously existing notifications from the notifications list manually/visually. |
| **Prerequisites** | Application running, user login to the app with their email address and an email received from a sender that is in the notification list. |
| **Test Steps** | 1. The Customer selects the Notification icon from the home page 2. The Customer selects the Notifications button 3. The Customer then selects the clear button next to the item to be deleted. |
| **Expected Output** | The application deletes the notification next to the pushed 'clear' button. It is no longer on the notification list. |
| **Assumptions** | User has an email account |
| **Result** | **PASS** |

## Test Case 14: Clear All Notification

|  |  |
| --- | --- |
| **Description** | In this test, the user will be deleting all the notifications from the existing notification lists. |
| **Requirements** | This test case validates that the user can delete all notifications from the notifications list manually/visually. |
| **Prerequisites** | Application running, user login to the app with their email address and an email received from a sender that is in the notification list. |
| **Test Steps** | 1. The Customer selects the Notification icon from the home page 2. The Customer selects the Notifications button 3. The Customer then selects the clear all button at the top of the notification list |
| **Expected Output** | The application deletes all notifications, and the notification list will be empty |
| **Assumptions** | User has an email account |
| **Result** | **PASS** |

## Test Case 15: Display QR code and click Mail View able link

|  |  |
| --- | --- |
| **Description** | In this test, the user will view their mail and see a generated URL link from a QR code that will redirect them to the appropriate page. |
| **Requirements** | This test case validates that the OCR is able to properly extract a URL from the image of the mail, as well as properly generate an actionable link that the user can click on. |
| **Prerequisites** | Application running, user login to the app with their email address, and the OCR has cached all the potential data for the user to work with. |
| **Test Steps** | 1. Test mail is sent with a QR code embedded in an image. 2. The user opens the application. (Back-end should trigger OCR and caching process) 3. The user opens the mail and sees their mail and our transcription with an actionable link. 4. The user clicks on the link, and we confirm this is the correct location. |
| **Expected Output** | When the user opens the mail view page, there is a generated link from a QR code in the text we insert. Upon clicking the URL, it sends the user to the appropriate webpage. |
| **Assumptions** | None. |
| **Result** | **PASS** |

## Test Case 16: Mail View Display URL and a clickable link

|  |  |
| --- | --- |
| **Description** | In this test, the user will be viewing their mail piece and see a URL link from the “Do more with your mail” section of the Informed Delivery email related to the mail piece. This will display a “Set a Reminder” link and if available, a “Learn More” link. The link should open the default phone web browser and open the appropriate webpage. |
| **Requirements** | This test case validates that the email processing properly extracted the mail piece identifying characteristics such that the associated URL for the “Do more with your mail” section from the users USPS informed delivery emails can be accessed from the application.  The links must properly take the user to the appropriate associated webpage. |
| **Prerequisites** | Application running, user login to the app with their email address, and the OCR has cached all the potential data for the user to work with. Additionally, the mail piece must be obtained from a USPS informed delivery email with ID. If no ID, then it is a mail piece that was manually entered and will not display the “Do more with your mail” links. |
| **Test Steps** | 1. A test mail account is used with various USPS informed delivery emails with mail pieces. These include the embedded scanned mail piece images and associated “Set a Reminder” and possibly “Learn More” links. 2. The user opens the application. (Back-end should trigger OCR and caching process which will store mail piece data). 3. The user searches for mail pieces and sees their mail and a transcription with a actionable links. 4. The user clicks on the link, and it is confirmed this is the correct webpage. |
| **Expected Output** | When the user opens the mail view page, there is a link from a URL in the “Do more with your mail” section. Upon clicking the URL, it sends the user to the appropriate webpage. |
| **Assumptions** | User has a USPS Informed Delivery account and has received USPS emails. The mail piece is recent enough that the “Set a Reminder” USPS page is still valid. |
| **Result** | **PASS** |

## Test Case 17: Mail View Display text from mail

|  |  |
| --- | --- |
| **Description** | In this test, the user will be viewing their mail piece and see content from the mail transcribed. |
| **Requirements** | This test case validates that the OCR can properly extract the text content of the mail piece and have it displayed in the application. |
| **Prerequisites** | Application running, user login to the app with their email address, and the OCR has cached all the potential data for the user to work with. |
| **Test Steps** | 1. At least two separate test mail accounts are used with USPS informed delivery emails with mail pieces in different trials. 2. The user opens the application. (Back-end should trigger OCR and caching process) 3. The user opens the mail and sees their mail and our transcription of the text content. |
| **Expected Output** | When the user opens the mail view page, there is a generated transcription of the text content. The text similarly matches what is in the mail piece. Also, an image of the scanned image will be displayed. |
| **Assumptions** | User has a USPS Informed Delivery account and has received USPS emails. |
| **Result** | **PASS** |

## Test Case 18: Mail View Display verify contact information

|  |  |
| --- | --- |
| **Description** | In this test, the user will be viewing their mail and see the content of their mail transcribed. Specifically contact information (email, phone number). |
| **Requirements** | This test case validates that the OCR is able to properly extract the text content of the mail. |
| **Prerequisites** | Application running, user login to the app with their email address, and the OCR has cached all the potential data for the user to work with. |
| **Test Steps** | 1. Test mail is sent with text content. 2. The user opens the application. (Back-end should trigger OCR and caching process) 3. The user opens the mail and sees their mail and our transcription of the text content. 4. The user checks for contact information. |
| **Expected Output** | When the user opens the mail view page, there is a generated transcription of the text content. The text displays contact information. |
| **Assumptions** | User has an account |
| **Result** | **PASS** |

## Test Case 19: Analytics Dashboard confirms screen time

|  |  |
| --- | --- |
| **Description** | In this test, the user will start a new session, view the Mail View screen for 3 minutes, and then view the analytics dashboard. |
| **Requirements** | This test case validates that session times are being sent to an analytics aggregator. |
| **Prerequisites** | The analytics server is running. |
| **Test Steps** | 1. The user logs into the application, making a note of the current date and time. 2. After three minutes, the user logs out of the application. 3. There is a delay in analytics reporting, so the user will need to wait at least an hour. 4. The user logs into the analytics dashboard and confirms that the three minutes of session time is displayed. |
| **Expected Output** | The user's session time is reported to the analytics server. |
| **Assumptions** |  |
| **Result** | **PASS**  The analytics server tracks engagement time per user. |

## Test Case 20: Analytics Dashboard displays link metrics

|  |  |
| --- | --- |
| **Description** | In this test, the user will open a link and confirm that data was transmitted to the analytics server. |
| **Requirements** | This test case validates that link data is being sent to an analytics server |
| **Prerequisites** | MailSpeak is running in Debug mode in Android Studio, and it is connected to the Firebase DebugView. |
| **Test Steps** | 1. The user logs into the application 2. The user opens the Mail View screen and opens an email containing a link. 3. The user clicks on the link, and the URL is opened in a web browser. 4. The user views the Firebase Debug view – the user can see an event Link\_Navigated that is recorded and the parameter itemId that records the URL. |
| **Expected Output** | The user's selected links are reported to the analytics server |
| **Assumptions** |  |
| **Result** | **PASS**  The event Link\_Navigated and parameter itemId were successfully recorded by the analytics server. |

## Test Case 21: Analytics Dashboard displays searches performed

|  |  |
| --- | --- |
| **Description** | In this test, the user will perform a search and confirm that data was transmitted to the analytics server |
| **Requirements** | This test case validates that search data is being sent to an analytics server |
| **Prerequisites** | MailSpeak is running in Debug mode in Android Studio, and it is connected to the Firebase DebugView. |
| **Test Steps** | 1. The user logs into the application. 2. The user opens the Search screen and performs a keyword search. 3. The user views the Firebase Debug view – the user can see an event Mail\_Search that is recorded and the parameter keyword along with the entered search term. |
| **Expected Output** | The user's keyword search is reported to the analytics server |
| **Assumptions** |  |
| **Result** | **PASS**  The event Mail\_Search and parameter keyword were successfully recorded by the analytics server. |

## Test Case 22: Mail View Call Phone Number

|  |  |
| --- | --- |
| **Description** | In this test, the user will view their mail and can open the phone for a given mail piece with a number listed. |
| **Requirements** | This test case validates that the application can call a phone number from the ma |
| **Prerequisites** | Application running, user login to the app with their email address, and the OCR has cached all the potential data for the user to work with. The given phone number is in the mail view |
| **Test Steps** | 1. Test mail is sent with text content and includes contact information. 2. The user opens the application. 3. The user opens the mail and sees their mail and our transcription of the text content. 4. The phone number is available, and the user selects it. 5. The application asks if the user would like to call or text, select call |
| **Expected Output** | The application opens the device-specific dialer for the given phone number |
| **Assumptions** | User has an account |
| **Result** | **PASS** |

## Test Case 23: Mail View Text Phone Number

|  |  |
| --- | --- |
| **Description** | In this test, the user will be viewing their mail and can open the text application for a given mail piece with a number listed. |
| **Requirements** | This test case validates that the application can text a phone number from the mail screen. |
| **Prerequisites** | Application running, user login to the app with their email address, and the OCR has cached all the potential data for the user to work with. The given phone number is in the mail view. |
| **Test Steps** | 1. Test mail is sent with text content and includes contact information. 2. The user opens the application. 3. The user opens the mail and sees their mail and our transcription of the text content. 4. The phone number is available, and the user selects it. 5. The application asks if the user would like to call or text, select text |
| **Expected Output** | The application opens the device-specific text application for the given phone number. |
| **Assumptions** | None. |
| **Result** | **PASS** |

## Test Case 24: Mail View Email provided email address.

|  |  |
| --- | --- |
| **Description** | In this test, the user will be viewing their mail and will be able to open their default email application. |
| **Requirements** | This test case validates the application can email and email address from the mail screen. |
| **Prerequisites** | Application running, user login to the app with their email address, and the OCR has cached all the potential data for the user to work with. The given email address is in the mail view |
| **Test Steps** | 1. Test mail is sent with text content and includes contact information. 2. The user opens the application. 3. The user opens the mail and sees their mail and our transcription of the text content. 4. The email address is available, and the user selects it. |
| **Expected Output** | The application opens the device-specific email application for the given email address |
| **Assumptions** | User has an account |
| **Result** | **PASS** |

# Appendix

## Test Result Summary and STP Report Key

