Email Client Project

Test Plan

CSE6214 Software Engineering
Lab Group 13
Chad Whitney, Joseph Howard, William Giddens, Trey Oneal

ChangeLog

Version	Change Date	By	Description
0.9	10/19/2020	Joe Howard	Initial Release
1.0	10/20/2020	Full team	Team Review and Update
1.1	10/23/2020	Joe Howard	Final Additions and Edits

1 Introduction

The Test Plan outlines the scope, approach, resources, and schedule of all testing activities. It identifies the items and features to be tested; types of testing. The Email Client system needs to be tested with all the features. It lists the tests for Version 1.0.

The objective of the test suite is to provide adequate coverage metrics, requirements validation, and system quality data such that sufficient data is provided for those making the decision to release.

1.1 Scope

1.1.1 In-Scope

The scope of this test plan is to make sure that all the requirements are met and developed. We also need to make sure that all the existing email client services are kept during the development of the requirements and they still works as required. Both scheduled and unscheduled test plan changes will be done and changes will be documents in this document by changing the version numbers of the document.

This project has the objective to develop an online prototype tool that uses Nylas API email services. This online tool is packaged within our email client system. This test plan should be able to show all the email client system functionality and features defined in the requirements document.

The tests referenced herein are written to validate use cases, requirements (both functional and non-functional), system architecture, and object design. The structured tests for object design will be run first as the components of the system are developed. The structured tests to validate the system architecture will be run next as the system is integrated in bottom-up fashion during integration test.

1.1.2 Out-of-Scope

The Server-side Nylas infrastructure for receiving API requests is out of scope for testing.

1.2 Quality Objective

For each level of testing, a separate test plan is prepared with the following set of deliverables:

- Test Case Number for each test
- Features to be tested specifically documented.
- Items to be tested are listed and concise
- Pass / Fail criteria is explicitly determined and defined

- Input File Names / Output File Names if applicable are documented
- Expected Results criteria is explicitly determined and defined
- Actual Results criteria is explicitly determined and defined
- Bugs/issues are identified and fixed before go live

1.3 Roles and Responsibilities

Name	Net ID	GitHub username	Role
Chad Whitney		Cwgu	Lead Developer, Product Manager
Trey Oneal		Trekkertrey	Developer, Component Tester
William Giddens		williamgiddens	Developer, System Tester
Joe Howard		Jhowardmsstate	Developer, Test Manager

2 Test Methodology

2.1 Overview

Incremental testing is used in agile development methods and hence, every release of the project is tested thoroughly. This ensures that any bugs in the system are fixed before the next release.

The test activities will be performed by using requirements document, user stories and acceptance criteria, and design specifications.

This section specifies generic pass/fail criteria for the tests covered in this plan. They are supplemented by pass/fail criteria in the test design specification. Note that “fail” in the IEEE standard terminology means “successful test” in our terminology.

Component Pass/Fail criteria

Tests executed on components only pass when they satisfy the signatures, constraints, and interfaces dictated by the Object Design Specification for that component. This includes positive tests, negative and stress tests, and boundary tests.

If a test exhibits a product failure to meet the objectives of the object design specification, it will fail and a defect/issue will be reported in the defect tracking system for review by the triage team.

Integration Pass/Fail criteria

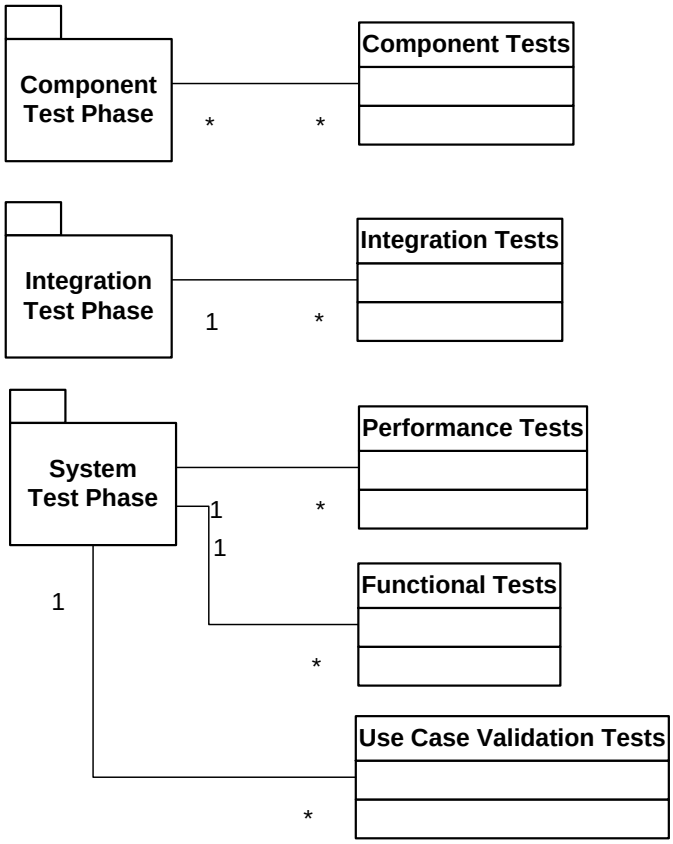
Tests executed on integrated components only pass when they satisfy the signatures, constraints, and interfaces dictated by both the object design specification and the system architecture specification. This includes positive tests, negative and stress tests, boundary conditions, and tests that explicitly manipulate the interface environment (such as the physical connection to the database server).

If a test exhibits a product failure to meet the objectives of both the object design specification and the system architecture specification, it will fail and a defect/issue will be reported in the defect tracking system for review by the triage team.

System Pass/Fail criteria

Tests executed against the system use the functional requirements, non-functional requirements, and use cases as the oracle to determine pass or fail.

If a test exhibits a product failure to meet the objectives of any of the functional requirements, non-functional requirements, or the use cases, it will fail and a defect/issue will be reported in the defect tracking system for review by the triage team.



Test Case alignment with test phases

2.2 Unit Testing

Unit testing will be conducted to verify the implementation of each necessary requirement changes. The requirements will be tested and verified. After any new requirement is added, unit testing will be done for these features.

2.3 Data Integrity Testing

Data integrity testing will be done to ensure schema data is not corrupted by the parser or internal data structures. This testing will be done independent of the User Interface in a white box fashion.

2.4 Functional Testing

Functional testing will be performed to verify all functional requirements are met successfully. This will be accomplished through black box testing.

2.5 User Interface (UI) Testing

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the UI provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

2.6 Performance Testing

Performance (Stress) testing is a type of performance test implemented and executed to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. This typically involves low resources or competition for resources. Low resource conditions reveal how the target-of-test fails that is not apparent under normal conditions. Other defects might result from competition for shared resources, like database locks or network bandwidth, although some of these tests are usually addressed under functional and load testing.

2.7 Regression Testing

Regression Testing will be conducted to ensure that changes to the application have not adversely affected previously tested functionality. In other words all the previously defined test cases will be applied with the new ones and pass or fail of each test case will be determined. Test cases that will be tested during the regression testing section of this document. Especially for an online application that will be used by many users simultaneously, this testing is important to complete.

2.8 Pass / Fail Criteria

The Pass/Fail criteria are identified in the test cases created in the section 6. Each test case has its own pass/fail criteria defined.

2.9 Test Levels

The Testing Levels primarily will include a Pass/Fail criteria are identified in the test cases created in the section 6. Each test case has its own pass/fail criteria defined.

2.10 Bug Triage

The goal of issue identification and triage is to

- To define the type of resolution for each bug
- To prioritize bugs and determine a schedule for all "To Be Fixed Bugs'.

2.11 Suspension Criteria and Resumption Requirements

Automated Unit Test Suite
As components are being developed, unit tests will be developed to test the interfaces of the components and low-level unit tests will be developed to test the methods of the underlying classes in the components.

As a prerequisite to the BAT, the automated unit test suite will be run by the build server on a per-build basis.

When the unit-test suite reports failures, testing will not occur on that build until the failures have been analyzed and resolved. Testing will resume on a build that passes the automated unit test suite.

Build Acceptance Test (BAT)
When a build is deemed ready to test by development, a build acceptance test will be run on the build. The BAT will consist of a broad but shallow set of tests to determine the overall stability of the build and decide if it is worth testing.

If the BAT fails on a particular build, testing will suspend until another build is created with any BAT failure issues fixed, verified by running the BAT again. Testing will resume on a build that passes the BAT.

Different build acceptance tests will be developed and used for the different test phases. Component BATs will be small and localized for each of the components. Integration BATs will vary based on the level of integration testing being performed. The System Test BAT will contain a set of tests that will utilize each of the components of the system.

Regression Testing

On a build by build basis, major bug fixes or code changes will be reviewed to determine the effects they may have on the system. If the changes are deemed to cause a sufficient amount of risk, regression test sets of the appropriately judged size will be created and executed.

A system-wide regression will also be run on the release candidate build to ensure incremental changes to the system have not altered the results of the tests that were run early in the test cycle.

System Design Changes
If at any point in time issues are submitted that require a design change to the system, all testing will be suspended. After the changes to the requirements, system architecture, and object design are made, a review and updates will be performed of the test specifications to ensure they properly align with the revised system changes. After updates are made, testing will resume. Tests in the vicinity of the change must all be rerun. A 20% regression of other tests must also be performed to ensure the changes did not adversely affect other parts of the system.

2.12Test Completeness

Here you define the criterias that will deem your testing complete.
For instance, a few criteria to check Test Completeness would be

- 100% test coverage
- All Manual & Automated Test cases executed
- All open bugs are fixed or will be fixed in next release

3 Resource & Environment Needs

3.1 Issues

Below are the issues that may affect the development and operation of the test plan, e.g. communications, security, training.

Ref	Issue	Action
1.	Testing Group needs to be trained.	Requirements will be explained to the testing group.
2.	Testing Group needs to be trained on security processes check.	Security is an important component of this prototype. Testing group needs to be training on security processes and the interaction with the API back-end.

3.2 Assumptions

All assumptions regarding the test plan are below.

Ref	Assumption	Impact
1.	All users have their security keys created and ready for tester to use. These security keys will be available from the server during testing.	To be able to test some of the test cases.
2.	All testers have network and internet connectivity.	To be able to test most of the test cases.
3	Testers have username and passwords setup.	Login pages will be required for access. To be able to test some test cases.
4	TravisCI will be used to launch test cases	To be able to run various tests and capture results.
5	All appropriate packages are installed and operational for testing.	See GitHub dependencies list

3.3 Testing Tools

Make a list of Tools like

- Requirements Tracking Tool
- Bug Tracking Tool
- Automation Tools

Required to test the project

3.4 Test Environment

It mentions the minimum **hardware** requirements that will be used to test the Application.

Following **software's** are required in addition to client-specific software.

- Any Internet connected system with web browser and ability to launch the software within Visual Studio Code.

3.5 Risks

Risk	Probability	Risk Type	Owner	Contingencies / Mitigation Approach
Unable to acquire some of the necessary hardware and software required for integration and system testing	5%	Equipment	Program Manager Test Manager Development Manager	Utilize existing acquired hardware. Split test execution into morning and evening shifts such that testing can occur for multiple teams in the same day using the limited hardware. This requires support of the development during both shifts.
Unable to acquire the necessary number of skilled personnel as the components become ready to test.	10%	Personnel Schedule	Test Manager	Resources for components will be split between the existing resources. Schedule must be adjusted accordingly.
Third party services utilized in the system become unavailable during testing	5%	Third party	Alliance Manager	Setup a communication channel to 3rd party to report and handle issues when they occur. Use the communication channel above to stay aware of planned outages and maintenance to help schedule test execution.
Components are not delivered on time	25%	Schedule	Development Manager	Integration testing with those components must be delayed until the component is delivered Overall integration test approach may be modified to do an appropriate amount of bottom-up as well as top-down or sandwich integration. Schedule must be adjusted accordingly.
Time Constraints	60%	Personnel	Test Manager	Testers will work in pairs on components. If a single member of the team decides to leave, a secondary testing with the knowledge of the component will still be able to train a new tester or finish the work. Schedule must be adjusted accordingly.

4 Test Cases

4.1 Test Deliverables

Test result template

Test Objective:	Exercise the target-of-test functions under the following stress conditions to observe and log target behavior that identifies and documents the conditions under which the system **fails** to continue functioning properly
User Story Mapping:	
Test Procedure:	Test steps outlined
Required Tools:	Transaction Load Scheduling and control tool installation-monitoring tools (registry, hard disk, CPU, memory, and so on) resource-constraining tools (TravisCI
Success Criteria:	Description: eg. The email client gracefully deals with too few resources, or connectivity errors in a reasonable amount of time.
Test Date & Results:	PASS/FAIL

4.1.1 Test Evaluation Summaries

These summaries will outline the tests conducted and their results.

4.1.2 Incident Logs and Change Requests

Incident log entries will be made for all bugs found during testing. The log will be used to track the status of the bugs.
Any modifications to the requirements must be done through change requests, which will ensure the proposed change is fully reviewed before being incorporated into dbViZ.

4.2 Testing Schedule

This section of the test plan covers responsibilities, staffing and training needs, risks and contingencies, and the test schedule.

Test Phase	Owner	Artifact(s)
Test Plan Creation	Test Manager	Test Strategy & Metrics
Test Specification Creation	Test Manager	Test Cases
Test Specification Team Review	Project Team	Requirements Traceability Matrix
Component Testing	Component Tester	Issue Log & Tracking

Integration Testing	Component & System Testers	Issue Log & Tracking
System Testing	System Tester	Issue Log & Tracking
Performance Testing	System Tester	Issue Log & Tracking
Use Case Validation	System Testers	Issue Log & Tracking
Alpha Testing	Product Manager / Lead Developer	Issue Log & Tracking
Beta Testing Pilot Program	Pilot Customers	Issue Log & Tracking

4.3 Responsibilities

The Test Manager is responsible for the overall test plan (this document) and test resources throughout the course of the project. He needs to be assigned to the project to review the requirements analysis, system architecture design, and object design of the system. From those specifications, he will generate the Test Plan. He will also generate section 9 “Test Cases” in this document. He will generate the list of test specifications and a brief description of each one in this document. He will generate and communicate the test strategy for the project to the test team and the rest of the project team as well as locate, acquire, and/or allocate the proper resources. He will provide periodic updates to the Program Director on the progress of test execution versus the plan as well as the metrics on the quality status of the product, focusing on key issues that need immediate attention from the Project Office.

The Test Leads are responsible for the creation of the detailed test specifications and will generate those and revise section 9 of this document as needed. The leads manage the day-to-day progress of each of their subcomponents and compile and report the metrics to the test manager. They are also responsible for ensuring the testers make adequate progress and follow the overall strategy defined by the Test Manager.

The Component Testers are responsible for the test execution on a daily basis for the component of the system to which they’ve been assigned. They also lead the effort during most of the integration test cycle and hand off the testing to the System Testers during the last states of integration testing.

4.4 Staffing and Training Needs

Test Manager
- responsible for finding and training the following test resources

Test Lead
- must be trained on the process being used for this project
- must be trained on the test specification format utilized
- must be trained on the defect/issue tracking system utilized

Component Testers
- must know Python, TravisCI, and GitHub
- must be familiar with Flask and Nylas
- must be skilled at unit testing, API testing, and integration testing
- must be trained on the process being used for this project, the test specification format utilized, and the defect/issue tracking system utilized

System Testers
- must know how to use Load testing & Stress testing automation tools
- must be experienced in system testing and use case validation testing
- must be trained on the process being used for this project.

- must be trained on the test specification format utilized
- must be trained on the defect/issue tracking system utilized

The System Testers are responsible for functional testing, performance testing, and use case validation testing during the System Test Phase of the project.

During Alpha Testing, the Product Managers / Business Analysts who wrote the requirements will test the system to determine if the system truly aligns with the original vision of the system.

During Beta Testing, a small group of specially chosen customers will pilot the product to ensure it suits their needs.

4.5 Test Cases

This section, the core of the test plan, lists the test cases that are used during testing. Each test case is described in detail in a separate Test Case Specification document. Each execution of these tests will be documented in a Test Incident Report document.

Test specifications derived from nonfunctional requirements

nfrs_3.3.1_usability_MouseKeyboardNavigation
nfrs_3.3.1_usability_EaseOfUnderstandabilityAndUse
nfrs_3.3.2_reliability_ErrorRecovery
nfrs_3.3.3_performance_SendException
nfrs_3.3.3_performance_10secondSendException
nfrs_3.3.3_performance_PageResponseTime15seconds
nfrs_3.3.6_interface_WebBrowserSupport

Test specifications derived from the use cases in the functional requirements

uc_3.4.2.2_1_Login
uc_3.4.2.2_1_LoginFlow
uc_3.4.2.2_2_CreateEmailFlow
uc_3.4.2.2_2_CreateEmailRespondFlow
uc_3.4.2.2_3_FileAttach
uc_3.4.2.2_3_FileAttachRemove
uc_3.4.2.2_4_ FileAttachSave
uc_3.4.2.2_4_ FileAttachForward
uc_3.4.2.2_5_FailLoginUsername
uc_3.4.2.2_6_ FailLoginPassword
uc_3.4.2.2_7_ FailLogingConnectivity
uc_3.4.2.2_8_ CreateEmailForwardFlow
uc_3.4.2.2_9_ CreateEmailRespondFlow
uc_3.4.2.2_10_DeleteEmail
uc_3.4.5_Logout

Test specifications derived from the system architecture specification

arch_3.3_api_NetworkErr_LoginForced
arch_3.3_api_NetworkErr_LogoutForced
arch_3.3_api_NetworkErr_ConnectivityFailure
arch_3.3_api_NetworkErr_BrowserError
arch_3.3_api_NetworkErr_ManageUsers

Test specifications derived from the subsystem decomposition and component diagram

odd_4.3_UserManagement_ValidateUser
odd_4.3_UserManagement_CreateUser

Test case specifications that cover end-to-end test scenarios derived from the use cases

e2e_LoginLogout
e2e_LoginSendEmailOneLogout
e2e_LoginViewEmailLogout
e2e_LoginSendEmailAttachFileLogout
e2e_LoginResendEmailLogout
e2e_LoginWalkAwayTimeOut

4.6 User Story Acceptance Testing

4.6.1 User Story 1 -Login Credential Verification

Test Objective:	To login to the email client with unique user credentials and be validated.
User Story Mapping:	e2e_LoginLogout uc_3.4.2.2_1_Login uc_3.4.2.2_1_LoginFlow odd_4.3_UserManagement_ValidateUser *odd_4.3_UserManagement_CreateUser* *uc_3.4.5_Logout* uc_3.4.2.2_5_FailLoginUsername uc_3.4.2.2_6_ FailLoginPassword uc_3.4.2.2_7_ FailLogingConnectivity nfrs_3.3.1_usability_MouseKeyboardNavigation nfrs_3.3.1_usability_EaseOfUnderstandabilityAndUse nfrs_3.3.2_reliability_ErrorRecovery nfrs_3.3.3_performance_SendException nfrs_3.3.3_performance_10secondSendException nfrs_3.3.3_performance_PageResponseTime15seconds nfrs_3.3.6_interface_WebBrowserSupport
Test Procedure:	1. Access email client system 2. Enter username 3. Enter password 4. Verify access by viewing emails 5. Logout of email client system

Required Tools:	Tester use and validation TravisCI automated account access with incorrect credentials and then correct credentials.
Success Criteria:	Access is only provided to valid users who provide correct username and password.
Test Date & Results:	10/19/2020 PASS. Required Regression Testing with every change

4.6.2 User Story 2 – Access and View Emails

Test Objective:	To login to the email client with unique user credentials; be validated and view emails.
User Story Mapping:	odd_4.3_UserManagement_ValidateUser *odd_4.3_UserManagement_CreateUser* e2e_LoginViewEmailLogout uc_3.4.5_Logout arch_3.3_api_NetworkErr_ConnectivityFailure arch_3.3_api_NetworkErr_BrowserError
Test Procedure:	1. Access email client system 2. Enter username 3. Enter password 4. Verify access 5. Logout of email client system
Required Tools:	Tester use and validation TravisCI automated account access with incorrect credentials and then correct credentials.
Success Criteria:	Access is only provided to valid users who provide correct username and password and displays emails relevant to their account.
Test Date & Results:	10/19/2020 PASS. Required Regression Testing with every change

4.6.3 User Story 3 – Access, View, & Compose Emails

Test Objective:	To login to the email client with unique user credentials; be validated and view and compose emails.

User Story Mapping:	odd_4.3_UserManagement_ValidateUser *odd_4.3_UserManagement_CreateUser* uc_3.4.2.2_2_CreateEmailFlow e2e_LoginSendEmailOneLogout arch_3.3_api_NetworkErr_ConnectivityFailure arch_3.3_api_NetworkErr_BrowserError uc_3.4.2.2_8_ CreateEmailForwardFlow uc_3.4.2.2_9_ CreateEmailRespondFlow uc_3.4.5_Logout
Test Procedure:	1. Access email client system 2. Enter username 3. Enter password 4. Verify access and view emails unique to user 5. Logout of email client system
Required Tools:	Tester use and validation TravisCI automated account access with incorrect credentials and then correct credentials.
Success Criteria:	Access is only provided to valid users who provide correct username and password and displays emails relevant to their account.
Test Date & Results:	10/19/2020 PASS. Required Regression Testing with every change

4.6.4 User Story 4 – User Forward Emails

Test Objective:	To login to the email client with unique user credentials; be validated and forward emails.
User Story Mapping:	odd_4.3_UserManagement_ValidateUser *odd_4.3_UserManagement_CreateUser* uc_3.4.2.2_2_CreateEmailFlow e2e_LoginSendEmailOneLogout arch_3.3_api_NetworkErr_ConnectivityFailure arch_3.3_api_NetworkErr_BrowserError uc_3.4.2.2_8_ CreateEmailForwardFlow uc_3.4.2.2_9_ CreateEmailRespondFlow uc_3.4.2.2_3_FileAttach uc_3.4.2.2_3_FileAttachRemove uc_3.4.2.2_4_ FileAttachSave uc_3.4.2.2_4_ FileAttachForward uc_3.4.5_Logout

Test Procedure:	1. Access email client system 2. Enter username 3. Enter password 4. Verify access and view emails unique to user 5. Forward email to valid email address (account) 6. Logout of email client system
Required Tools:	Tester use and validation TravisCI automated account access with incorrect credentials and then correct credentials.
Success Criteria:	Access is only provided to valid users who provide correct username and password and displays emails relevant to their account.
Test Date & Results:	10/19/2020 PASS. Required Regression Testing with every change

4.6.5 User Story 5 – User Send Emails with Attachments

Test Objective:	To login to the email client with unique user credentials; be validated and send or forward emails with file attachments.
User Story Mapping:	odd_4.3_UserManagement_ValidateUser *odd_4.3_UserManagement_CreateUser* uc_3.4.2.2_2_CreateEmailFlow e2e_LoginSendEmailOneLogout arch_3.3_api_NetworkErr_ConnectivityFailure arch_3.3_api_NetworkErr_BrowserError uc_3.4.2.2_8_ CreateEmailForwardFlow uc_3.4.2.2_9_ CreateEmailRespondFlow e2e_LoginSendEmailAttachFileLogout e2e_LoginResendEmailLogout uc_3.4.5_Logout
Test Procedure:	1. Access email client system 2. Enter username 3. Enter password 4. Verify access and view emails unique to user 5. Compose or Forward email to valid email address (account) with attachment that meets attachment file-type and size requirements. 6. Logout of email client system

Required Tools:	Tester use and validation TravisCI automated account access with incorrect credentials and then correct credentials.
Success Criteria:	Access is only provided to valid users who provide correct username and password and displays emails relevant to their account.
Test Date & Results:	10/19/2020 FAIL. 10/22/2020 PASS. ISSUE #34 Required Regression Testing with every change