System Verification and Validation Plan for SyncMaster

Team 15, SyncMaster
Kyle D'Souza
Mitchell Hynes
Richard Fan
Akshit Gulia
Rafeed Iqbal

March 10, 2025

Revision History

Date	Version	Notes
11/4/24	1.0	Initial Draft of VnV Plan for Rev0
03/04/25	1.1	Modifications to Plan for Rev1

Contents

List of Tables

Symbols, Abbreviations, and Acronyms

Refer to *Section 4 Naming Conventions and Terminology* in the SRS document(3) for all relevant symbols, abbreviations, and acronyms.

1 Overview

This document outlines the Verification and Validation plan which will be used to ensure the SyncMaster application meets the requirements specification and the City of Hamilton's acceptance. The verification plan is specified in detail outlining what methods will be used to verify the functional and non-functional requirements. The system tests to support this process are specified in detail. The validation plan to ensure stakeholder acceptance is further identified.

2 General Information

2.1 Summary

The software being tested is 'SyncMaster', an application being developed for the City of Hamilton, Water Division. The general functions of the application are:

- Document Management: The application will have a document hosting functionality, enabling effetive document management and a single source of truth. It will allow contractors and employees to view all relevant station documentation. It allows contractor users to acknowledge they have read required documents.
- Site Visits: The application will have a portal accessible to contractors attending site to complete work. The portal will allow users to acknowledge they have completed the necessary health and safety training and are aware of any hazards. The portal will also collect information from the user regarding the purpose of their site visit and allow the contractor to view the station documents.
- Displaying and Exporting Data: Data is collected through the portal including arrival time of contractors, the time they sign out, and any files uploaded during their visit. This data is available to be viewed and exported by users with the correct level of access.
- Permission Based Access: Users registered on the system will have different access to features depending the permissions given to their account type.

- Geolocation Verification: The application will be able to verify if a user is at the appropriate site using GPS.
- Notifications: The application will be able to indicate to admin users when documents or and contractor annual Health and Safety Training is expired.

2.2 Objectives

The objectives of this document are to be able to build confidence in the correctness of the application being built, and to demonstrate adequate usability of the application by performing the tests outlined in this document.

It is out of scope for the verification and validation plans outlined in this document to test external libraries. Instead it is assumed that the implementation team for the external library has already validated the library.

2.3 Challenge Level and Extras

The challenge level for this project is a general level, and the extras are conducting user testing and developing a user manual. For more information refer to the Problem Statement and Goals (2, *Challenge Level and Extras*).

2.4 Relevant Documentation

1. Problem Statement and Goals (2): Provides context on the goals and scope of the application.
2. Software Requirements Specification (3): Specification of all requirements being tested and validated in this document.
3. Hazard Analysis (4): Additional specifications for security and safety requirements.
4. Design Documents (1; 5): Reference for the mechanisms and design behind functionality and systems being tested.

3 Plan

This section outlines our plans for verifying and validating the software under development, along with its accompanying documentation. It provides an overview of the validation team and details the approach to verifying the software's documentation, design, and implementation, as well as validating the final product.

3.1 Verification and Validation Team

The verification and validation team will consist of members from the development team as well as members from the project stakeholders. A summary of the members and their roles are given in the table below.

Names and Role	Responsibilites
Mitchell Hynes, Kyle D'Souza, Richard Fan, Akshit Gulia, Rafeed Iqbal - Development Team	• Responsible for verification of implementation details and adherance of the system to the SRS. The development team will own the most of the verification activities pertaining to the software. This includes activities such as developing verification plans, code reviews, unit testing, integration testing, and SRS verification.
Matthew Yakymyshyn - Stakeholder from the City of Hamilton	• Responsible for the verification of the system. They will be responsible for ensuring, and determining to what degree the system fulfils its intended purpose. Some examples of the activities that they would be responsible for would be SRS validation and acceptance testing.
Tarnveer Takhtar, Matthew Bradbury, Harman Bassi, Kyen So - Peer Reviewers	• Responsible for critiquing and providing suggestions for artifacts produced. This would include reviewing design documents to ensure that they are complete and unambiguous. They will provide opinions and insights as third parties who are not directly involved in the project.
Spencer Smith, Yiding Li - Teaching Team for 4G06	• They will also be responsible for the final evaluation of the system and will determine if the system meets the initially specified functional and non-functional requirements. They will also play a role in providing feedback on produced artifacts throughout the project.

Table 1: Testing Team

3.2 SRS Verification Plan

To verify the SRS document, we will use the following methods:

1. Formal reviews with our TA, Yiding. A checklist will be used to track the status of the review and be used as an instrument to verify our SRS. The checklist to be used is at the end of this section of the document.

2. Ad-hoc peer reviews tracked through GitHub issues. Throughout the course we receive peer feedback on the quality of our SRS and VnV Plan. The peer feedback provided is valuable input which we will action through issues and pull requests as identified in the development plan.

3. Milestone feedback provided through Avenue will also be used to improve the checklist which we use as our instrument to measure SRS verification.

The checklist below will be used during formal reviews for SRS verification with our TA.

- Have the requirements been listed in the appropriate sections of the SRS document?
- Are all definitions and acronyms defined in the glossary of terms?
- Are the system inputs properly specified?
- Are the system outputs properly specified?
- Do the functional requirements avoid conflicts with other requirements?
- Do the functional requirements avoid specifying the system design?
- Are the requirements clear enough that they could be implemented by an independent engineering team correctly?
- Is each requirement testable?
- Is independent testing of each requirement possible?
- Are the functional requirements traceable to a use case of the system?
- Are there any open issues from reviewers on a requirement?

3.3 Design Verification Plan

To verify the design, we will use the following methods:

1. Ensure that each module specified in section 8 of MG.pdf satisfies its corresponding requirement outlined in the table. This will be completed as a check in our team meeting on Mar 6, 2025.

2. Ensure all outstanding GitHub issues connected to the Design are addressed and closed.

3.4 Verification and Validation Plan Verification Plan

To ensure the quality and completeness of the verification and validation plan, the testing team will ensure the following:

1. There will be peer reviews of the document coming from our assigned Peer Review team, to provide feedback on the plan.

2. The TA assigned to the project will review and provide feedback on the plan when grading.

3. The development team will review the document to ensure the quality of the plan.

4. It should be checked by the testing team that each requirement has a test case associated with it. This can be done using traceability matrices and cross-referencing requirements with what exists in the SRS.

The following is a checklist that can be used to verify the verification and validation plan:

- ☐ Ensure all issues on the project GitHub related to the document from the Peer Review team is closed.

- ☐ Ensure that all feedback provided by the TA in the rubric for the document on Avenue is addressed.

- ☐ Ensure the Development team has reviewed the document and is in agreement that the document is up to quality and completeness standards.

☐ Check the traceability matrices to ensure that all requirements have a test case associated with them.

3.5 Implementation Verification Plan

- To ensure proper implementation of application functionality, unit tests will be conducted on all functions and modules as specified in this document.
- Continuous Integration (CI) tools will be utilized to automatically test all code before deployment, verifying code integrity and identifying issues early in the development process.
- All pull requests will undergo a detailed review process to ensure code quality, adherence to standards, and functional accuracy before merging into the main codebase.
- Following each revision, a team code walkthrough will be conducted to review changes collectively, fostering knowledge sharing and alignment on implementation details.

3.6 Automated Testing and Verification Tools

- Python unit testing: PyTest
- JavaScript/TypeScript unit testing: Jest

Detailed information about automated testing and verification tools that we plan to use are explained in the Development Plan (2, *Expected Technology, Coding Standard*). In terms of quality metrics, we plan on using GitHub actions to generate a report every time a pull request is made against the main branch. The report will be generated from the unit testing framework selected, more specifically jest-html-reporter and pytest-cov. Initially, these tests will include unit tests with more types of automated testing to be determined as the project progresses.

3.7 Software Validation Plan

1. Rev0 prototype demonstration to Technical Services staff. The rev0 prototype will be demonstrated to the City Staff on this team at an in-person meeting. Each requirement from the SRS will be demonstrated in the demo so the full scope of the application is displayed. The demonstration will also show how it satisfies the use cases identified in the SRS. Feedback received from the City at this meeting will be used to improve the prototype for rev1.

2. Stakeholder progress check-ins. Short periodic meetings will be arranged with the Technical Services team as required to demonstrate the user interface design prototypes and receive feedback from the facilities managers. Email updates will also be sent for items which require stakeholder clarification. The user interface is the most important part of the application to validate with the end-users. Consistent communication on the direction of the interface design to ensure it is usable will catch problems early and greatly improve the quality and acceptance of the application.

4 System Tests

This section outlines tests to determine if requirements from the SRS are satisfied.

4.1 Tests for Functional Requirements

Below are tests for the functional requirements.

4.1.1 File Handling

1. TC-FR1

   Control: Manual

   Initial State: The system is running and user is logged in

   Input: A file (can have any extension)

Output: File should be uploaded successfully and visible in the folder/path it was uploded under with the following details:

(a) File Name

(b) Date Modified

(c) Type

(d) Size

(e) Owner

Test Case Derivation: The system is expected to support the upload of files regardless of their file types and also show the details mentioned above once they are successfully uploaded.

How test will be performed: Files with common file types (.pdf, .docx, .xlsx, .txt, .png, .jpeg, .jpg, .zip and .rar will be used for testing) and uncommon file types (.rb, .rda, .bin and .vdi will be used for testing) will be manually uploaded. Afterwards, their visibility on the portal will be verified.

2. TC-FR2

Control: Manual

Initial State: The system is running, and the user is logged in

Input: A file present in the portal (i.e., an uploaded file)

Output: The user should able to download the file in the portal if it is one of the following types:

(a) .doc

(b) .docx

(c) .xls

(d) .xlsx

(e) .ppt

(f) .pptx

(g) .pdf

(h) .csv

(i) .txt

Test Case Derivation: The above mentioned types are commonly used by the stakeholders in day-to-day operations and therefore, they should be able to download them. Less common file types (ones not mentioned above) are not used by our stakeholders. Moreover, file types like .xlsm (macro-enabled excel workbook) can be malcious and thus, can compromise the safety of the system (malcious macro present in a macro-enabled excel workbook executes when file is opened/viewed). Therefore, they are not viewable on the portal.

How test will be performed: One file of each of the above file types will be selected and downloaded.

4.1.2 Access Control

1. TC-FR3

Control: Manual

Initial State: The system is running and different user roles of Admin, Contractor, and Employee user exist.

Input: User Role (Admin, Contractor, Employee user)

Output: Admin users should be able to change permissions and access all documents. Contractor users should only have read access to the contractor portal and the ability to acknowledge documents. Test Case Derivation: The system should enforce different access levels as specified in the functional requirements in the SRS document.

How test will be performed: Manually log in with different users roles (as mentioned above) and verify the access permissions and capabilities.

4.1.3 User Information Visibility

1. TC-FR4

Control: Manual

Initial State: The system is running and the number of users in the system is greater than or equal to 2 and there is atleast one admin user and atleast one contractor user.

Input: A valid user id

Output: Admin users should be able to view all the details of the other user with the corresponding user id provided.

Test Case Derivation: Admin users need to access contractor information to manage site visit details. In some cases, admin users will also need to access details of other admin users.

How test will be performed: Manually log in as an admin user and verify the visibility of the user details based on the user id provided.

4.1.4 Geolocation Verification

1. TC-FR5

   Control: Manual

   Initial State: The system is running and the location service is enabled on the contractor's device

   Input: Contractor user's location

   Output: Authentication and acknowledging of the document should be allowed if within the allowed geolocation.

   Test Case Derivation: To ensure presence of contractors on the site, the should restrict the actions mentioned above if they are not within premises of the plant.

   How test will be performed: Manually attempt to authenticate and sign/upload documents from different locations and verify the results.

4.1.5 Notifications

1. TC-FR6

   Control: Manual

   Initial State: The system is running, and a document with an expiry date is present in the system

Input: Expired document

Output: The system should indicate to admin users that a document has expired.

Test Case Derivation: Notifications for expired documents ensure documents are compliant with any regulations and up-to-date.

How test will be performed: Manually expire a document and verify that the system indicates this to admin users on the admin portal.

2. TC-FR7

Control: Manual

Initial State: The system is running, and there are contractor users who have not completed their annual health and safety training.

Input: Contractor user without completed training.

Output: The system should indicate to admin users when a contractors annual health and safety training is expired.

Test Case Derivation: Admin users need to be aware of contractor's training status to ensure compliance.

How test will be performed: Manually create a contractor user with an expired training date and observe that the system indicates this to admin users on the admin portal.

4.1.6 Document Acknowledgement

1. TC-FR8

Control: Manual

Initial State: The system is running, and documents are available for acknowledgement.

Input: Acknowledgement of documents during site visit.

Output: The should store the date, time, and name of the user who acknowledged the documents during the site visit.

Test Case Derivation: Accurate record-keeping is required for accountability and traceability. How test will be performed: Manually acknowledge documents and verify that the system stores the correct information.

4.1.7 Training Verification

1. TC-FR9

   Control: Manual

   Initial State: The system is running, and a contractor user has not completed their annual health and safety training

   Input: A contractor user that has not completed the training accesses portal for a site visit without an employee accompanying them. They acknowledge this during sign in.

   Output: The system notifies contractor they are not authorized to access the site and should contact their facilities manager.

   Test Case Derivation: Contractors are required to complete training before they can access site, or else must be accompanied by an employee.

   How test will be performed: Manually try to authenticate as a contractor user without completing required training or an emplyee during authentication process and verify that the system prevents it.

4.2 Tests for Nonfunctional Requirements

Below are tests for the nonfunctional requirements.

4.2.1 Usability and Humanity Requirements

Ease of Use

1. TC-EU-1

   Type: Dynamic, Manual

Initial State: System is functional with test accounts made for users. Users are between MIN_AGE and MAX_AGE age limits. No prior instructions or training are provided.

Input/Condition: User who has no prior experience using the system attempts to use it.

Output/Result: Users are able to discover at least 70% of the main functionalities of the system within 10 minutes without assitance.

How test will be performed: The test will be performed by letting users from our stakeholder who have no prior knowledge of the system use the system. After using the system they will fill out a short quiz that asks them about the specific features/uses of the system. Scoring above 70% is considered a pass. If 70% or more of testers pass, we can say that this test passes. See **Section 6.2** Usability Survey Questions.

2. TC-EU-2

   Type: Dynamic, Manual

   Initial State: System is functional with test accounts made for users. Users are between MIN_AGE and MAX_AGE age limits.

   Input: Users logs in, and performs an action classified as irreversible (e.g. deletion).

   Output: The user sees a notification informing them that their action is irreversible or an undo option.

   How test will be performed: The test will be performed by a user from our stakeholder. The user will be instructed to perform an irrversible action on the system. After performing the action and viewing the prompt, the user will be asked if they understood the implications of their actions. If the user understands the prompt and actions correctly, the test is considered successful.

Learning

1. TC-LR-1

   Type: Dynamic, Manual

Initial State: System is functional with test accounts made for users. Users are between MIN_AGE and MAX_AGE age limits.

Input: User logs in, and uses a feature that they have not used before (e.g. viewing documents)

Output: User has explored the new feature and is confident in using the feature.

How test will be performed: The test will be performed by a user from our stakeholder. The user will be instructed to use a feature of the system that they have not seem before. After 10 minutes, the user will be asked to rate their understanding of the feature and how confident they are using it. The questions will be a simple 1-10 scale. If the user selects a 7 or above, the test is considered passed. See **Section 6.2** Usability Survey Questions.

2. TC-LR-2

   Type: Dynamic, Manual

   Initial State: System onboarding documents are avaliable. Users are between MIN_AGE and MAX_AGE age limits.

   Input: User views the system onboarding documents.

   Output: User has finished viewing the system onboarding documents and reports their confidence on how they feel about using the system.

   How test will be performed: The test will be performed by a user from our stakeholder. They will be asked to view the onboarding documents. After 5 minutes the user will be asked a few questions to gauge their confidence when using the system. See **Section 6.2** Usability Survey Questions.

Understandability and Politeness

1. TC-UP-1

   Type: Dynamic, Manual

   Initial State: System is functional with test accounts made for users. Users are between MIN_AGE and MAX_AGE age limits.

Input: User logs in, and views all available screens of the user interface.

Output: The user has seen the images and text on all available screens of the user interface and notes the content seen.

How test will be performed: The test will be performed by a user from our stakeholder. They will be guided through all screens for every type of role (General/Contractor, Manager, Admin). They will then be asked if any of the images or text seen contains anything that they might have found offensive or politically charged. If the user reports they haven't seen anything that may have been offensive or poitically charged, the test is successful.

2. TC-UP-2

   Type: Dynamic, Manual

   Initial State: System is functional with test accounts made for users. Users are between MIN_AGE and MAX_AGE age limits.

   Input: User navigates through the system as either a contractor or admin user.

   Output: User has performed the actions and the system has informed the user of the error. User actions the feedback given and understands the potential resolution paths.

   How test will be performed: The test will be performed by a user from our stakeholder. The user will be instructed to freely explore the menus in the portal and see if an error is encountered. If an error is encountered, The user will then be asked if they understood the error and if they were able to understand possible solutions. If no errors are encountered, the test is successful.

Accessibility

1. TC-AS-1

   Type: Dynamic, Manual

   Initial State: System is functional with test accounts made for users. Users are between MIN_AGE and MAX_AGE age limits.

Input: User logs in, and views all available screens of the user interface.

Output: The user seen all available screens of the user interface and notes the content seen.

How test will be performed: The test will be performed by a user from our stakeholder. They will be guided through all screens for every type of role (contractor, ddmin, employee). They will then be asked if the design of the user interface aligns with accessibility features that they have seen in other applications used by the city. If the user reports that the accessibility features seen are similar to other applications that they already use, the test is successful.

4.2.2 Look and Feel Requirements

Appearance

1. TC-LF-1

   Type: Manual

   Initial State: The system is in an operational state, with all components running

   Input/Condition: User logs in as any possible user and views all available screens of the user interface.

   Output/Result: The user has seen all available screens of the user interface and feels that most of the colours used by the application are aesthetically pleasing and pleasant.

   How test will be performed: The test will be performed by a user acting as each of the user roles (contractor, admin, employee). A member of the development team will guide them to all available screens for their role and ask them to evaluate how aesthetically pleasing the colour palette is. If they report the colour palette is pleasing then the test is successful, otherwise it is failed.

2. TC-LF-2

   Type: Dynamic, Manual.

Initial State: The system is in an operational state, with all components running

Input/Condition: Load and navigate through the application on various supported operating systems, with varying supported browsers and screen sizes.

Output/Result: User is able to interact with the system on all screen sizes and understand the interface regardless of screen size

How test will be performed: One user for each type of role (contractor, admin, employee) will ensure that functionalities work as expected across screen sizes, browsers, and operating systems. The tests should be performed using Windows 10, IOS, and Android, which are the supported operating systems for this application. The browsers used should be Microsoft Edge and Google Chrome. Testing should include a mobile device and a desktop to ensure accommodation for different screen sizes and device types.

The core functionalities for a contractor user will be signing into the system, viewing a document, acknowledging a document, and logging out. The core functionalities of an admin will be uploading a document, viewing document expiry status, viewing site visit logs, creating users, deleting users, and assigning roles to users. The core functionalities of an employee user will be viewing documents.

4.2.3 Performance Requirements

Speed and Latency Requirements

1. TC-PR-1

   Type: Manual

   Initial State: The system is running, and documents are available in the database.

   Input/Condition: Request to retrieve a document.

   Output/Result: The document should be retrieved in less than 2 seconds

   How test will be performed: Manually request document retrieval and measure time taken. Verify that it does not exceed 2 seconds

2. TC-PR-2

   Type: Manual

   Initial State: The system is running

   Input/Condition: Navigation to a different page

   Output/Result: The system should be able to navigate to a different page in under 1.5 seconds.

   How test will be performed: Manually navigate to a different page and measure time taken. Verify that it does not exceed 1.5 seconds

Safety-Critical Requirements

1. TC-PR-3

   Type: Manual

   Initial State: The system is running and ready to upload and retrieve documents.

   Input/Condition: User uploads and downloads a document

   Output/Result: The downloaded document should look identical to the document which was uploaded

   How test will be performed: User manually uploads and then downloads a document. If the downloaded document looks identical to the uploaded document, then the test is successful.

2. TC-PR-4

   Type: Manual

   Initial State: The system has different user roles configured (contractor, admin, employee)

   Input/Condition: Attempt to access documents with different user roles.

   Output/Result: Document access should be restricted based on user roles. Admin users should have the ability to read and write documents

while contractor and employee users should only have the abilty to read documents.

How test will be performed: Manually log in with each user role and attempt to access documents. Verify that access is correctly restricted.

Precision or Accuracy Requirements

1. ~~TC-PR-5~~
   ~~Type: Manual Initial State: The system is running with a populated database with number of users and documents greater than equal to 1. Input/Condition: Perform search queries related to documents or user data Output/Result: At least 95% of the search results should be relevant to search query How test will be performed: Manually perform search queries and evaluate the relevance of the results~~
   Search queries will no longer be a feature of the application.

2. ~~TC-PR-6~~
   ~~Type: Manual Initial State: The system is running and ready for document upload and retrieval Input/Condition: Upload and retrieve documents Output/Result: No corruption or data loss in 99.99% of the cases How test will be performed: Manually upload and retrieve docuements. Then, verify their integrity.~~
   Removing due to assumption cloud services are fully functional and secure.

Robustness or Fault-Tolerance Requirements

1. TC-PR-7

   Type: Manual

   Initial State: The system is running

   Input/Condition: Provide unexpected inputs and events

   Output/Result: The system should handle them without crashing

   How test will be performed: Manually input unexpected data (such as files named with special characters) and observe the system's response. The test passes if the system does not crash.

2. ~~TC-PR-8~~
   ~~Type: Manual Initial State: The system is running Input/Condition: Introduce anomalies or potential failures Output/Result: The system should detect and alert on these issues How test will be performed: Manually introduce anomalies and verify the system's detection and alert mechanism.~~
   Removing as corresponding requirement no longer exists.

Capacity Requirements

1. ~~TC-PR-9~~
   ~~Type: Manual Initial State: The system is running Input/Condition: Upload and download documents Output/Result: The system should support a peak data transfer rate of 1 Gbps. How test will be performed: Manually upload and download documents and measure the data transfer rate.~~
   Removing as this is dependent on the internet speed of the user which we can not control.

2. TC-PR-10

   Type: Manual

   Initial State: The system is running

   Input/Condition: Attempt to upload individual files up to 1 GB.

   Output/Result: The files should be successfully uploaded

   How test will be performed: Manually upload files of varying sizes up to 1 GB and verify successfull uploads.

Scalability or Extensibility Requirements

1. TC-PR-11

   Type: Manual

   Initial State: The system is running

   Input/Condition: Create nested document categories with multiple sub-categories.

Output/Result: The system should allow nesting of documents and categories.

How test will be performed: Manually create nested document categories and verify their functionality.

2. ~~TC-PR-12~~
~~Type: Automatic Initial State: The system is running under normal conditions Input/Condition: Simulate high traffic to the system. Output/Result: The system should distribute traffic evenly and avoid single points of failure. How test will be performed: Simulate high-traffic using a script to generate users and perform the task of uploading a document. Verify the system's load balancing capabilities.~~
Removing due to elimination of corresponding requirement.

Scalability or Extensibility Requirements

1. ~~TC-PR-13 Type: Automatic Initial State: The system is running Input/Condition: Add new feature into the system Output/Result: The system is functional with the new functionalites and the behaviour of the old functionalities is not changed (the ones which were not intended to be impacted by the introduction of the new feature). The ease of introduction should atleast be 8. How test will be performed: Add code for the new feature and measure the ease of introduction of the feature on a scale of 1-10. Verify that the ease of introduction is greater or equal to 8.~~
Removing due to elimination of corresponding requirement.

4.2.4 Maintainability and Support Requirements

Maintenance

1. ~~TC-MS-1~~
~~Type: Manual Initial State: Cloud platform used for deployments is functional, GitHub actions is functional. Input/Condition: A workflow run in GitHub actions is triggered to deploy the application into a specified environment, and building/automated unit testing of the application is completed Output/Result: The application is deployed to the specified environment in under 30 minutes How test will be performed: The test will be performed by triggering a GitHub actions workflow run to deploy the application into the development environment, and waiting~~

~~for it to complete. Once completed, the unit testing and building time will be subtracted from the total runtime of the workflow run, to determine the deployment time, this should be under 30 minutes to be considered a success.~~
Removing due to elimination of corresponding requirement.

2. ~~TC-MS-2~~
   ~~Type: Manual Initial State: GitHub actions is functional. Input/Condition: A workflow run in GitHub actions is triggered to deploy the application into a specified environment, and automated unit testing of the application is completed. Output/Result: The application is built in under 10 minutes. How test will be performed: The test will be performed by triggering a GitHub actions workflow run to deploy the application into the development environment, and waiting for it to complete. Once completed, the unit testing and deployment time will be subtracted from the total runtime of the workflow run, to determine the build time, this should be under 10 minutes to be considered a success.~~
   Removing due to elimination of corresponding requirement.

3. ~~TC-MS-3~~
   ~~Type: Manual Initial State: GitHub actions is functional. Input/Condition: A workflow run in GitHub actions is triggered to perform all automated testing (end-to-end or unit testing) of the application. Output/Result: The automated tests for the application run in under 10 minutes. How test will be performed: The test will be performed by triggering a GitHub actions workflow run to perform all automated tests, and checking that once this run is completed, the total runtime is under 10 minutes.~~ Removing due to elimination of corresponding requirement.

4. TC-MS-4

   Type: Manual

   Initial State: GitHub actions is functional.

   Input/Condition: A push is made to a branch with a pull request open off of it.

   Output/Result: The branch is found to have line coverage $\geq 95\%$ and the branch coverage is found to be $\geq 90\%$

How test will be performed: The test will be performed by triggering a GitHub actions workfow run to run unit tests on the application upon every push to a branch with a PR open off of it. The workflow will generate a coverage report, and when the coverage report is generated, it should be checked that the line coverage and branch coverage meet the expectations.

5. TC-MS-5

   Type: Manual

   Initial State: GitHub is functional

   Input/Condition: The GitHub repository for the project is checked to ensure that all functional requirements listed in the SRS have a unit test corresponding to them.

   Output/Result: All functional requirements found in the SRS have a unit test corresponding to them

   How test will be performed: The test will be performed by checking the Traceability Matrices in this document, to make sure that there exists a unit test for all functional requirements.

6. TC-MS-6

   Type: Manual

   Initial State: GitHub is functional

   Input/Condition: The GitHub repository for the project is checked to ensure that all appropriate documentation existed for users to be able to maintain the system.

   Output/Result: Instructions are provided in the GitHub repository for the project on how users can continue to maintain the system. This includes contribution guidelines, descriptions of all GitHub actions workflows, and documentation on the design of the system.

   How test will be performed: The test will be performed by checking that the documentation listed in the output/result exists in the repository, and by verifying with the Matthew Yakymyshyn that this documentation is able to be understood by the city of Hamilton.

7. TC-MS-7

   Type: Manual

   Initial State: GitHub is functional

   Input/Condition: The GitHub repository for the project is checked to see the user manual.

   Output/Result: There exists a user manual in the github repository which describes, at a minimum, how to leverage the functionalities described in the functional requirements of the SRS from the user interface.

   How test will be performed: The test will be performed by checking that the user manual exists, and that there is user-facing documentation in the manual on how to achieve all functionalities described in the functional requirements.

8. TC-MS-8

   Type: Manual

   Initial State: GitHub is functional

   Input/Condition: The GitHub repository for the project is checked to see the API documentation.

   Output/Result: There exists API documentation in the github repository which follows the OpenAPI Specification (OAS) standard.

   How test will be performed: The test will be performed by checking that an OpenAPI Specification for the API's provided by the system exists on the project repository.

9. TC-MS-9

   Type: Manual

   Initial State: GitHub is functional

   Input/Condition: The GitHub repository for the project is checked to see the the internal documentation.

Output/Result: There exists documentation on all internal functions and classes defined in the project repository.

How test will be performed: The test will be performed by checking that documentation exists for all functions and classes defined in the project repository.

10. TC-MS-10

    Type: Manual

    Initial State: GitHub is functional, Cloud deployment platform is functional

    Input/Condition: The GitHub repository for the project is checked to see the the deployment documentation

    Output/Result: There exists documentation on how to deploy the project, which is up-to-date and works when attempted.

    How test will be performed: The test will be performed by checking that the deployment documentation exists and that a member of our development team can follow the provided steps to successfully deploy the application.

4.2.5 Compliance Requirements

Compliance

1. ~~TC-L-1~~
   ~~Type: Manual Initial State: Github is functional, cloud deployment platform is functional Input/Condition: A contractor uses the authentication feature of the application to verify their presence on site for the facilities managers. Output/Result: The contractor successfully authenticates in the application and successfully complies with each step of the entry/exit procedure document for the station provided through the application. How the test will be performed: The contractor user will authenticate at the site for a work order to fix an exhaust fan. Through the authentication process, the entry/exit procedure will be used as a checklist to ensure it is followed and the legislation specified in CR-L1 is not violated.~~ Removing as corresponding requirement has been eliminated.

2. ~~TC-L-2~~
   ~~Type: Manual Initial State: Github is functional, cloud deployment platform is functional Input/Condition: The admin account registers a contractor account with the contractors personal email address. Output/Result: The contractors account is successfully registered, and the email address is only possible to view either from the facility managers account, the admin account, and the contractors account. How the test will be performed: The facility manager will use the interface to register an account in the application. The application will then be opened in different views based on profile. These profiles will be the registered contractor, the admin user, the facilities manager, and a second contractor profile. The test is successful when the email address of the contractor is only visible in the profile of that contractor, the facilities manager, and the admin user.~~
   Removing as corresponding requirement has been eliminated.

3. ~~TC-L-3~~
   ~~Type: Manual Initial State: Github is functional, cloud deployment platform is functional Input/Condition: A facilities manager signs into the application and downloads a compliance document. Output/Result: This document is sent to the administrative team of the BCOS system who are able to record the document in their system. How the test will be performed: The facilities manager signs into the system and downloads a compliance document. The test is successful if this document is confirmed by the City it is in a form that is able to be uploaded to the BCOS system.~~
   Removing as corresponding requirement has been eliminated.

4.2.6 Cultural Requirements

Cultural

1. TC-CR-1

   Type: Manual

   Initial State: Github is functional, cloud deployment platform is functional

Input/Condition: The Github repository code is checked and the code for the display of the user interface is opened.

Output/Result: The user interface and user experience is aligned with the 5 City of Hamilton cultural pillars. These are Collective Ownership, Steadfast Integrity, Courageous Change, Sensational Service, and Engaged and Empowered Employees.

How the test will be performed: The capstone team will compare the qualities of the user interface application code with the 5 pillars and assess whether each pillar is met by the application. The test passes if the team determines that the pillars are satisfied.

4.2.7 Operational and Environmental Requirements

Ability to Connect and Use

1. TC-OE-1

   Type: Manual

   Initial State: Device connected to internet.

   Input/Condition: Attempt to connect to the application.

   Output/Result: The device is successfully able to connect to the application.

   How test will be performed: The test is successful if a user is able to connect to the application through the internet.

2. TC-OE-2

   Type: Manual

   Initial State: Devices connected to the internet and have the supported browsers available.

   Input: Desktop Browsers being tested are Chrome and Edge, and the mobile devices being tested are Android and iPhone.

   Output: The devices successfully connect to and are able to use the application.

How test will be performed: Exploratory testing using the supported devices and browsers. The devices should be able to connect and use all functionality in the application, and provide a similar experience.

Future Integration

1. ~~TC-OE-3~~
   ~~Type: Manual Initial State: Branch with functionality to fill in data fields when provided a valid work order number Input/Condition: Valid work order number Output/Result: Relevant data fields should be filled in by the application How test will be performed: Dummy API used which returns work order information given a work order number. Will test if the functionality in the branch is able to fill in the fields with data from the API, indicating it is viable to add this integration in the future.~~
   Removing due to elimination of corresponding requirement.

Release and Change Log

1. TC-OE-4

   Type: Manual

   Initial State: New revision

   Input/Condition: Revision released, change log produced

   Output/Result: Revision is released by planned date, change log details all changes.

   How test will be performed: Whenever a revision is released, will inspect the change log to check if all changes have been noted.

4.2.8 Safety and Security Requirements

Account Permissions

1. TC-SS-1

   Type: Manual

Initial State: User IDs of each possible access level created.

Input/Condition: Attempt to use every feature available using each ID

Output/Result: Users only have access to features allowed by their permissions.

How test will be performed: Accounts will be created on the application for each available level of access. Using each account, the use of each feature of the application will be attempted. Wether an action was possible or not will be noted and compared to the permissions of the account.

2. TC-SS-2

   Type: Manual

   Initial State: User IDs of each possible access level created.

   Input/Condition: Attempt to upload, delete, modify files in different directories.

   Output/Result: Users only have access to files and directories allowed by their permissions.

   How test will be performed: Accounts will be created on the application for each available level of access. Using each account, the creation, deletion and modification of of files with varying permission requirements will be attempted. The testing will be done to cover directories of varying permission requirements. Wether an action was possible or not will be noted and compared to the permissions of the account.

Data Validation

1. ~~TC-SS-3~~
   ~~Type: Manual Initial State: Initial set of files uploaded to application Input/Condition: View files on the application Output/Result: Files on the application are the appropriate version available on SharePoint and/or MySDS How test will be performed: Files will be checked against their current versions. Users with appropriate permissions should have the ability to update the files if outdated.~~ Removing due to elimination of corresponding requirement.

2. TC-SS-4

   Type: Manual

   Initial State: Data fields empty.

   Input/Condition: Submit incomplete, impossible, or malicious data through data fields

   Output/Result: Fail-safes trigger, noting the bad data and informing the administrator.

   How test will be performed: Attempt to submit data fields filled in with a set of entries which vary between incomplete, impossible, and malicious and check how the application handles the inputs.

Encryption

1. TC-SS-5

   Type: Manual

   Initial State: The application is accessed via a web browser with tools for inspecting network requests.

   Input/Condition: Access and use the application using https:// in a browser

   Output/Result: Network requests displayed as HTTPS in the browser's developer tools, and the security certificates are valid and properly configured.

   How test will be performed: Connect to and use the application using https:// and ensure the connection is secured with SSL/TLS. Use the browser's developer tools to inspect the requests made by the site. Check that all requests are made over HTTPS and the lock icon/secure status is visible in the browser's address bar. Check if SSL certificate is valid and correctly configured.

Adherance to Policies, Regulations, and Laws

1. ~~TC-SS-6~~
   ~~Type: Manual Initial State: The application is complete, checklist(s)~~

~~outlining all Federal, Provincial, Municipal and City of Hamilton, Water Division rules, regulations and policies created. Input/Condition: Review the application's functionality, data handling, and security measures against checklist(s) Output/Result: The application should align with all defined requirements, policies, and regulations, showing full compliance with legal and departmental standards without any deviations. How test will be performed: Develop a comprehensive checklist that includes all applicable government regulations, standards, and internal policies. Cross-reference the application's design and operational documentation with the compliance checklist. Perform a manual inspection of the application's source code, configuration settings, and security protocols to ensure they meet the required standards. Test relevant features to confirm that they adhere to the compliance checklist.~~ Removing due to elimination of corresponding requirement.

Adherance to Policies, Regulations, and Laws

1. TC-SS-7

   Type: Manual

   Initial State: The application is being actively maintained.

   Input/Condition: A security vulnerability or weakness is discovered.

   Output/Result: Pull requests created by Dependabot are merged within a week.

   How test will be performed: Inspection of whether pull requests created by Dependabot are merged within a week.

Hazard Handling

1. TC-SS-8

   Type: Manual

   Initial State: Unacknowledged documents during site visit and a contractor user account exists in application

Input/Condition: Contractor user declines acknowledgement, attempts to acknowledge but does not complete action, or ignores acknowledgement.

Output/Result: Visit logs identify the failure to complete acknowledgement.

How test will be performed: Attempt to complete a visit without acknowledgements and check how the application handles the action.

4.3 Traceability Between Test Cases and Requirements

Req. ID	**Test ID's**
FR1	TC-FR1, TC-FR2
FR3	TC-FR3
FR4	TC-FR4
FR5	TC-FR5
FR6	TC-FR6
FR7	TC-FR8
FR8	TC-FR7
FR9	TC-FR9
LF-AP1	TC-LF-1
LF-ST1	TC-LF-2
UH-EU1	TC-EU1
UH-EU2	TC-EU2
UH-LR1	TC-LR1
UH-LR2	TC-LR2
UH-UP1	TC-UP1

UH-UP2	TC-UP2
UH-AS1	TC-AS1
PR-SL1	TC-PR-1
PR-SL3	TC-PR-2
PR-SC1	TC-PR-3
PR-SC2	TC-PR-4
PR-PA1	TC-PR-5
PR-RFT1	TC-PR-7
PR-CR2	TC-PR-10
PR-SE1	TC-PR-11
OE-PE1	TC-OE-1
OE-WE1	TC-OE-2
OE-WE2	TC-OE-2
OE-REL1	TC-OE-4
OE-REL2	TC-OE-4
OE-REL3	TC-OE-4
OE-REL4	TC-OE-4
MS-MTN4	TC-MS-4
MS-MTN5	TC-MS-5
MS-MTN6	TC-MS-6
MS-SUP1	TC-MS-7
MS-SUP2	TC-MS-8
MS-SUP3	TC-MS-9

MS-SUP4	TC-MS-10
MS-ADP1	TC-LF-2
MS-ADP2	TC-LF-2
MS-ADP3	TC-LF-2
SR-AR1	TC-SS-1
SR-AR2	TC-SS-1
SR-AR3	TC-SS-1
SR-AR4	TC-SS-2
SR-IR1	TC-SS-2
SR-IR3	TC-SS-4
SR-PR1	TC-SS-5
SR-AU1	TC-SS-6
SR-IMR1	TC-SS-7
SR-S1	TC-SS-8
CR-CR1	TC-CR-1

Table 2: Requirements to Test Case Traceability Matrix

5 Unit Test Description

The approach for unit testing the application is to test normal behaviour, including each reachable branch, of each access routine, test all error behaviours, and to test all edge cases applicable.

5.1 Unit Testing Scope

All modules except the Routing Module, Function Compute Module, and Analytics and Reporting Module will be tested. The exclusions are due to the modules being implemented by AWS, and therefore not needing to be implemented by us.

5.2 Tests for Functional Requirements

5.2.1 Database Interaction Module

1. UT-DB1

   Type: Functional, Automatic

   Initial State: No items in the database

   Input: Put method called with an item to put in the database

   Output: The item gets placed into the database

   Test Case Derivation: After putting an item into a database it should exist there

   How test will be performed: Unit test with mocked AWS services using moto

2. UT-DB2

   Type: Functional, Automatic

   Initial State: Database exists

   Input: Put method called with a failing precondition

   Output: Database not updated and item is not added

   Test Case Derivation: If provided precondition to check is not true, then item creation should fail

   How test will be performed: Unit test with mocked AWS services using moto

3. UT-DB3

Type: Functional, Automatic

Initial State: Database exists

Input: Put method called by caller with lacking AWS permissions

Output: Database not updated and item is not added, permission error raised

Test Case Derivation: If caller has insufficient permission, then item creation should fail

How test will be performed: Unit test with mocked AWS services using moto

4. UT-DB4

   Type: Functional, Automatic

   Initial State: Database exists

   Input: Put method called, but and unknown error occurs in AWS

   Output: Database not updated and item is not added, error raised indicating external error

   Test Case Derivation: When we get a failure from AWS the item was presumably not added, and we also want to make sure the correct type of error was raised

   How test will be performed: Unit test with mocked AWS services using moto

5. UT-DB5

   Type: Functional, Automatic

   Initial State: Database has an item in it

   Input: Get method called

   Output: The details of the item in the database

   Test Case Derivation: If an item is in the database, we should be able to retrieve it

   How test will be performed: Unit test with mocked AWS services using moto

6. UT-DB6

   Type: Functional, Automatic

   Initial State: Empty database exists

   Input: Get method called

   Output: Resource not found error raised

   Test Case Derivation: When the item we want to retrieve does not exist a sensible error should be raised to indicate that

   How test will be performed: Unit test with mocked AWS services using moto

7. UT-DB7

   Type: Functional, Automatic

   Initial State: Empty database exists

   Input: Get method called, but unknown error occurs from AWS

   Output: Error raised indicating external error

   Test Case Derivation: When we get a failure from AWS we want to make sure the correct type of error was raised

   How test will be performed: Unit test with mocked AWS services using moto

8. UT-DB8

   Type: Functional, Automatic

   Initial State: Database has an item in it

   Input: Delete method called on the existing item

   Output: Item no longer exists in the database

   Test Case Derivation: When delete is called the specified item should be deleted

   How test will be performed: Unit test with mocked AWS services using moto

9. UT-DB9

   Type: Functional, Automatic

   Initial State: Database has an item in it

   Input: Delete method called on the existing item, but with a failing precondition

   Output: Item still exists, error is raised indicating conditon check failed

   Test Case Derivation: When delete is called with a failing precondition, item should not be deleted

   How test will be performed: Unit test with mocked AWS services using moto

10. UT-DB10

    Type: Functional, Automatic

    Initial State: Database has an item in it

    Input: Delete method called on the existing item, but caller does not have write permission

    Output: Item still exists, error is raised indicating permission error

    Test Case Derivation: When delete is called by a caller with insufficient permission, item should not be deleted

    How test will be performed: Unit test with mocked AWS services using moto

11. UT-DB11

    Type: Functional, Automatic

    Initial State: Database has an item in it

    Input: Delete method called on the existing item, but unknown AWS error occurs

    Output: Item still exists, error is raised indicating external error

    Test Case Derivation: When delete is called and an AWS error occurs, item should not be deleted, and an appropriate error should be raised

How test will be performed: Unit test with mocked AWS services using moto

12. UT-DB12

    Type: Functional, Automatic

    Initial State: Database has an item in it

    Input: Update method is called to update the existing item, adding new attributes, and modifying existing ones

    Output: The requested attributes of the item to update are updated

    Test Case Derivation: If the update method is called you would epect the requested changes to show up in the database

    How test will be performed: Unit test with mocked AWS services using moto

13. UT-DB13

    Type: Functional, Automatic

    Initial State: Database has an item in it

    Input: Update method is called to update the existing item, removing existing attributes

    Output: The requested attributes to remove from the item are removed

    Test Case Derivation: If the update method is called you would expect the requested changes to show up in the database

    How test will be performed: Unit test with mocked AWS services using moto

14. UT-DB14

    Type: Functional, Automatic

    Initial State: Database has an item in it

    Input: Update method is called with a failing precondition

    Output: The requested changes are not made, and an error is raised reflecting the condition failure

Test Case Derivation: If the precondition provided to the function is not met, then you would expect it to fail, and not update

How test will be performed: Unit test with mocked AWS services using moto

15. UT-DB15

    Type: Functional, Automatic

    Initial State: Database has an item in it

    Input: Update method is called by a caller without write permissions

    Output: The requested changes are not made, and an error is raised reflecting the permission error

    Test Case Derivation: Without sufficient permissions a caller should not be able to perform an update

    How test will be performed: Unit test with mocked AWS services using moto

16. UT-DB16

    Type: Functional, Automatic

    Initial State: Database has an item in it

    Input: Update method is called, but an unknown error occurs in AWS

    Output: The requested changes are not made, and an error is raised reflecting the external error

    Test Case Derivation: When AWS fails, no changes should be made, and an error should be raised to callers can handle it appropriately

    How test will be performed: Unit test with mocked AWS services using moto

17. UT-DB17

    Type: Functional, Automatic

    Initial State: Database has multiple items in it, some are documents, and some are site visits

Input: Query method is called, requesting only documents

Output: A list of all documents are recieved, and no site visits are in the list

Test Case Derivation: When calling query, it is expected that output is filtered based on parameters that were passed

How test will be performed: Unit test with mocked AWS services using moto

18. UT-DB18

    Type: Functional, Automatic

    Initial State: Database has multiple items in it, some are documents, and some are site visits

    Input: Query method is called, requesting only documents

    Output: A list of all documents are recieved, and no site visits are in the list

    Test Case Derivation: When calling query, it is expected that output is filtered based on parameters that were passed

    How test will be performed: Unit test with mocked AWS services using moto

19. UT-DB19

    Type: Functional, Automatic

    Initial State: Database has multiple items in it, some are documents, and some are site visits

    Input: Query method is called, requesting only documents, but with the reverse flag on

    Output: A list of all documents are recieved, and no site visits are in the list, and the list is the reverse of the output for UT-DB19

    Test Case Derivation: When calling query, it is expected that output is filtered based on parameters that were passed, and if the reverse parameter is used, the list should be on the opposite order.

How test will be performed: Unit test with mocked AWS services using moto

20. UT-DB20

    Type: Functional, Automatic

    Initial State: Database has multiple items in it, all documents

    Input: Query method is called, requesting documents, with a start key provided

    Output: A list of all document that appear after the given start_key in the database

    Test Case Derivation: If a start key is provided, only database entries that appear after the start key in the database should be returned by the queries

    How test will be performed: Unit test with mocked AWS services using moto

21. UT-DB21

    Type: Functional, Automatic

    Initial State: Database has multiple items in it, all documents

    Input: Query method is called, requesting documents, with a filter for documents that require acknowledgement

    Output: A list of all document that require acknowledgement

    Test Case Derivation: Queries in DynamoDB have two types of filter conditions, ones on the keys, and ones on the non-key attributes. Conditions on key attributes are more efficient and quick then ones on non-key attributes. This test is just making sure that conditions on non-key attributes also work (even if they are still inefficient).

    How test will be performed: Unit test with mocked AWS services using moto

22. UT-DB22

Type: Functional, Automatic

Initial State: Database has multiple items in it, all documents

Input: Query method is called, using a non-equivalence key condition on a partition key

Output: A error is raised reflecting the fact that this is an invalid condition

Test Case Derivation: In DynamoDB, partition keys can only be queried using an exact equivalence condition, not with a greater then, less than, starts with, etc. So this is testing that we can catch that and raise an appropriate error.

How test will be performed: Unit test with mocked AWS services using moto

23. UT-DB23

    Type: Functional, Automatic

    Initial State: Database has multiple items in it, all documents

    Input: Query method is called, but an unknown error occurs from AWS

    Output: A error is raised reflecting the external error

    Test Case Derivation: Ensures that an appropriate error is raised on unexpected issues in AWS,

    How test will be performed: Unit test with mocked AWS services using moto

5.3 Tests for Nonfunctional Requirements

N/A, no non-functional unit tests for our modules

References

[1] SyncMaster Team. Module guide for syncmaster. https://github.com/Spitgranger/SyncMaster/blob/main/docs/Design/SoftArchitecture/MG.pdf, 2024.

[2] SyncMaster Team. Problem statement and goals. https://github.com/Spitgranger/SyncMaster/blob/main/docs/ProblemStatementAndGoals/ProblemStatement.pdf, 2024.

[3] SyncMaster Team. System requirements specification. https://github.com/Spitgranger/SyncMaster/blob/main/docs/SRS-Volere/SRS.pdf, 2024.

[4] SyncMaster Teams. Hazard analysis. https://github.com/Spitgranger/SyncMaster/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf, 2024.

[5] SyncMaster Teams. Module interface specification for syncmaster. https://github.com/Spitgranger/SyncMaster/blob/main/docs/Design/SoftDetailedDes/MIS.pdf, 2024.

6 Appendix

6.1 Symbolic Parameters

The following symbolic parameters are used:

$$\text{MIN\_AGE} = 18$$
$$\text{MAX\_AGE} = 70$$

6.2 Usability Survey Questions

1. Were you able to understand how to use the system after reading the onboarding documentation?
2. On a scale of 1 to 10, how confident are you using the system?
3. On a scale of 1 to 10, how easy was it to discover features of the system?
4. How confident were you in interpreting the system feedback? Was it able provide confirmation or possible solutions to your input?
5. Did you find the content of the system to be politically neutral and unoffensive?

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

   When writing this deliverable, our team was able to quickly meet and delegate sections of the document between members. We had a very productive TA meeting for the VnV plan which answered many questions we had. One example was we were able to clarify that section 5 of the document would be completed later when the design of the system had been done.

2. What pain points did you experience during this deliverable, and how did you resolve them?

   One pain point we experienced was deciding what the best verification and validation methods would be. We wanted to ensure that we chose the right approaches for this application which would result in the highest quality of work. We resolved this challenge by discussing and investigating the merits of different options, and developing a test suite which covered every requirement we had at least once.

3. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project?

Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage, Valgrind etc. You should look to identify at least one item for each team member.

Akshit Gulia - Needs to get familiar with the PyTest testing framework for python.

Mitchell Hynes - Needs to get familiar with the PyTest testing framework for python and Figma for prototyping designs.

Richard Fan - Needs to get familiar with Jest testing framework for javascript and PyTest testing framework in python.

Kyle D'Souza - Needs to get familiar with Jest test framework for javascript and Figma for prototyping designs.

Rafeed Iqbal - Needs to get familiar with PyTest framework for python and GitHub actions for running tests in GitHub.

4. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

   For all of the aforementioned skills some potential ways of acquiring knowledge or mastering the skill would be to get hands on experience with the skill by applying it in a project, reading documentation, and watching online tutorials.

   Akshit Gulia - Plans on watching tutorials on youtube related to PyTest, and also plans on building a small personal learning project on the side in python which uses PyTest for testing.

Mitchell Hynes - Plans to follow the introductory tutorials provided by the creators of PyTest and Figma and will also read documentation on them.

Richard Fan - Plans to read documentation on Jest and PyTest, and also use them in a personal project.

Kyle D'Souza - Plans to watch online tutorials on Jest and Figma. Also plans to read documentation on them.

Rafeed Iqbal - Plans to read documentation on GitHub actions and Pytest. Also plans to watch online tutorials on them.