Verification and Validation Report: SyncMaster

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

March 10, 2025

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

Refer to *Section 4: Naming Conventions and Terminology* of the Software Requirements Specification document SRS.pdf.

Contents

List of Tables

List of Figures

This document ...

3 Functional Requirements Evaluation

4 Nonfunctional Requirements Evaluation

4.1 Usability

Usability testing was conducted on the application prototype to validate humanity, look and feel, operational, and cultural requirements. Two users were given the following scenarios for the usability test, to simulate a typical use of the system:

Phase 1: Testing the contractor portal

You are a contractor hired by the City of Hamilton to perform work at a station. You arrive and scan the QR code at the station to authenticate your presence and explain the work you will perform. We will ask you to perform a variety of tasks to assess their discoverability and usability. We welcome you explaining your thought process as you navigate through the screens.

- You have scanned the QR code. Please follow the initial steps for the health and safety acknowledgements. Did the experience of viewing the acknowledgements screen feel easy and intuitive? Did you encounter any errors? After the acknowledgements, are you able to find the station documents?
- Are you able to find and fill out the instructions for the purpose of your visit?

Phase 2: Testing the admin portal

You are a Facility Manager who has an account with admin permissions. You sign into the application and would like to perform some routine tasks during the day.

- You would like to navigate to the station documents and add a site specific document for station HC057. Please attempt to do so. Were you frustrated trying to find where this was located?

- Please locate the site wide documents which would be displayed for all stations.
- Please locate and view the site visit logs.

2 Users were then directed to fill out the Usability Survey identified in the VnV Plan.
User 1: Man between age of 60 - 65, no prior experience with the system. User if comfortable using a smartphone and laptop.
User 2: Women between age of 60 - 65, no prior experience with the system. User is comfortable using a smartphone and laptop. Through these actions, the following results were observed:

Test. ID	User	**Result**
TC-EU-1	User 1	
TC-EU-1	User 2	
TC-EU-2	User 1	
TC-EU-2	User 2	
TC-LR-1	User 1	
TC-LR-1	User 2	
TC-LR-1	User 1	
TC-LR-2	User 2	

Table 1: Usability Test Cases

4.2 Performance

4.3 Maintainability and Support

Test. ID	Input	Expected Output	Result
TC-MS-4	-	There is 95% line coverage and 90% branch coverage on our code	Pass, this is being checked in GitHub actions
TC-MS-5	-	All functional requirements have a corresponding unit test	Pass
TC-MS-6	-	Contribution guidelines and maintainer documentation of system approved by the City of Hamilton	Fail, not yet approved
TC-MS-7	-	User manual exists and has been approved by the City of Hamilton	Fail, not yet created
TC-MS-8	-	OAS3 compliant documentation has been provided for all API's	Fail, not yet created
TC-MS-9	-	Internal abstractions (classes and functions) in the system have documentation associated with them	Pass, linter checks this
TC-MS-10	-	Documentation for deployment of the system exists	Fail, not yet created

Table 2: Maintainability and Support Test Cases

4.4 etc.

5 Comparison to Existing Implementation

This section will not be appropriate for every project.

6 Unit Testing

Note: Unit tests all use moto to mock AWS services.

6.1 Database Interaction Module Unit Tests

Test. ID	Description	Result

UT-DB1	Use the DBTable.put method to create an item, and check that it exists in the database afterwards	Pass
UT-DB2	Use the DBTable.put method with a failing precondition, and check that a ConditionCheckFailed is raised, and the item is not created	Pass
UT-DB3	Use the DBTable.put method with a permission error from AWS, and check that a PermissionException is raised, and the item is not created	Pass
UT-DB4	Use the DBTable.put method with an arbitrary error from AWS, and check that a ExternalServiceException is raised, and the item is not created	Pass
UT-DB5	Use the DBTable.get method to get a pre-existing item	Pass
UT-DB6	Use the DBTable.get method to get an item which does not exist, and check that a ResourceNotFound is raised	Pass
UT-DB7	Use the DBTable.get method with an arbitrary error from AWS, and check that a ExternalServiceException is raised	Pass
UT-DB8	Use the DBTable.delete method on a pre-existing item, and check that the item is no longer in the database	Pass
UT-DB9	Use the DBTable.delete method with a failing precondition, and check that a ConditionCheckFailed is raised, and the item is not deleted	Pass
UT-DB10	Use the DBTable.delete method with a permission error from AWS, and check that a PermissionException is raised, and the item is not deleted	Pass
UT-DB11	Use the DBTable.delete method with an arbitrary error from AWS, and check that a ExternalServiceException is raised, and the item is not deleted	Pass
UT-DB12	Use the DBTable.update method to update an item adding new attributes, and modifying existing ones, and check that these changes are reflected in the database	Pass
UT-DB13	Use the DBTable.update method to update an item removing some attributes and check that these changes are reflected in the database	Pass

UT-DB14	Use the DBTable.update method with a failing precondition, and check that a ConditionCheckFailed is raised, and the item is not updated	Pass
UT-DB15	Use the DBTable.update method with a permission error from AWS, and check that a PermissionException is raised, and the item is not updated	Pass
UT-DB16	Use the DBTable.update method with an arbitrary error from AWS, and check that a ExternalServiceException is raised, and the item is not updated	Pass
UT-DB17	Use the DBTable.query method, and ensure the returned items all match the query criterion	Pass
UT-DB18	Use the DBTable.query method, under a different GSI, and ensure the returned items all match the query criterion	Pass
UT-DB19	Use the DBTable.query method, with a reversed query direction, and ensure the returned items all match the query criterion	Pass
UT-DB20	Use the DBTable.query method, with using the start_key of an existing db item, to determine where to start the query from, and ensure the returned items all match the query criterion	Pass
UT-DB21	Use the DBTable.query method, using a filter expression alongside a key condition, and ensure the returned items all match the query criterion	Pass
UT-DB22	Use the DBTable.query method, with an invalid key condition, and ensure that a ConditionValidationError is raised	Pass
UT-DB23	Use the DBTable.query method with an arbitrary error from AWS, and check that a ExternalServiceException is raised	Pass

Table 3: Unit Test Cases for Database Interaction Module

6.2 File Storage Interaction Module Unit Tests

Test. ID	Description	Result
UT-FS1	Use the S3Bucket.create_upload_url method to create an upload url, upload some content to the url, and check that the content exists in the S3 bucket	Pass
UT-FS2	Use the S3Bucket.create_upload_url method with a S3Bucket object that does not have write permissions, and check that a PermissionException gets raised	Pass
UT-FS3	Use the S3Bucket.create_get_url method for a pre-existing file in S3, and check that the file content can be accessed through the url, and check that a PermissionException gets raised	Pass
UT-FS4	Use the S3Bucket.delete method on a pre-existing file in S3, and check that the file no longer exists	Pass
UT-FS5	Use the S3Bucket.delete method on a pre-existing file in S3, with an ETag mismatch, and check that a ResourceNotFound is raised	Pass
UT-FS6	Use the S3Bucket.delete method with a permission error, and check that a PermissionException is raised	Pass
UT-FS7	Use the S3Bucket.delete method with an arbitrary error from AWS, and check that a ExternalServiceException is raised	Pass

Table 4: Unit Test Cases for File Storage Interaction Module

6.3 Location Verification Module Unit Tests

Test. ID	Description	Result
UT-LV1	Use the verify_location method to verify a coordinate within a defined radius, and ensure the return value is True	Pass
UT-LV2	Use the verify_location method to verify a coordinate outside a defined radius, and ensure the return value is False	Pass

UT-LV3	Use the verify_location method to verify a coordinate just on the boundary of a defined radius, and ensure the return value is True	Pass

Table 5: Unit Test Cases for Location Verification Module

6.4 User Authentication Module Unit Tests

Test. ID	**Description**	**Result**
UT-UA1	Attempt to authenticate with a pre-existing user, and ensure that a token is successfully generated	Pass
UT-UA2	Attempt to authenticate with an invalid location, and ensure that an UnauthorizedException is raised	Pass
UT-UA3	Attempt to authenticate with an initial one-time password, and ensure that an ForceChangePasswordException is raised	Pass
UT-UA4	Attempt to authenticate with the wrong password, and ensure that an UnauthorizedException is raised	Pass
UT-UA5	Attempt to authenticate with a user that does not exist, and ensure that a ResourceNotFound is raised	Pass
UT-UA6	Attempt to authenticate with an arbitrary error from AWS, and ensure that a ExternalServiceException is raised	Pass
UT-UA7	Attempt to signout a user, with a valid access token, and ensure that the token is successfully invalidated	Pass
UT-UA8	Attempt to signout a user, with an invalid access token, and ensure that a BadRequestException is raised	Pass
UT-UA9	Attempt to signout a user, with an arbitrary error from AWS, and ensure that a ExternalServiceException is raised	Pass

Table 6: Unit Test Cases for User Authentication Module

6.5 User Management Module Unit Tests

Test. ID	Description	Result
UT-UM1	Create an admin, employee, and contractor account, using admin credentials	Pass
UT-UM2	Create a user, using admin credentials, where there is already another user using the same email, and check that a ConflictException is raised	Pass
UT-UM3	Create a user, using employee and contractor credentials, and ensure an UnauthorizedException is raised	Pass
UT-UM4	Create a user, with an arbitrary error occuring from AWS, and check that an ExternalServiceException is raised	Pass
UT-UM5	Get the details of an existing user, using admin and employee credentials	Pass
UT-UM6	Get the details of a user that does not exist, using admin and employee credentials, and ensure a ResourceNotFound is raised	Pass
UT-UM7	Use contractor credentials to get the details of their own user	Pass
UT-UM8	Use contractor credentials to get the details of another user, and ensure an UnauthorizedException is raised	Pass
UT-UM9	Get the details of a user, with an arbitrary error occuring from AWS, and check that an ExternalServiceException is raised	Pass
UT-UM10	Update a user, using admin credentials	Pass
UT-UM11	Update a user, using admin credentials, where the user to update does not exist, and check that a ResourceNotFound is raised	Pass
UT-UM12	Update a user, using employee and contractor credentials, and ensure an UnauthorizedException is raised	Pass
UT-UM13	Update a user, with an arbitrary error occuring from AWS, and check that an ExternalServiceException is raised	Pass
UT-UM14	Delete a user, using admin credentials	Pass
UT-UM15	Delete a user, using admin credentials, where the user to delete does not exist, and check that a ResourceNotFound is raised	Pass

UT-UM16	Delete a user, using employee and contractor credentials, and ensure an UnauthorizedException is raised	Pass
UT-UM17	Delete a user, with an arbitrary error occuring from AWS, and check that an ExternalServiceException is raised	Pass

Table 7: Unit Test Cases for User Management Module

6.6 Logging Module Unit Tests

Test. ID	**Description**	**Result**
UT-LG1	Create a site visit log, and ensure it gets added to the database	Pass
UT-LG2	Create a site visit log, with the same site id, user id, and entry time as an existing log, and ensure a ResourceConflict is raised	Pass
UT-LG3	Add an exit time to an existing log with no logged exit time, and ensure the change is reflected in the database	Pass
UT-LG4	Add an exit time to an existing log with no logged exit time, but an entry time prior to the exit time we are attempting to add, and ensure a TimeConsistencyException is raised	Pass
UT-LG5	Add an exit time to an existing log with a logged exit time, and ensure an ExitTimeConflict is raised	Pass
UT-LG6	Add an exit time, where there are no existing logs for the user at the given site, and ensure an ResourceNotFound is raised	Pass
UT-LG7	Get a list of all site visit logs, using admin and employee credentials	Pass
UT-LG8	Get a list of all site visit logs, using contractor credentials, and ensure that an UnauthorizedException is raised	Pass
UT-LG9	Get a list of all site visit logs using filters to only include logs modified between two given dates, and ensure that logs are correctly filtered	Pass

UT-LG10	Get a list of all site visit logs using a database start_key to start the listing from a certain log, and ensure that logs start from the correct location	Pass

Table 8: Unit Test Cases for Logging Module

7 Changes Due to Testing

[This section should highlight how feedback from the users and from the supervisor (when one exists) shaped the final product. In particular the feedback from the Rev 0 demo to the supervisor (or to potential users) should be highlighted. —SS]

8 Automated Testing

N/A, the only automated testing we have is the unit tests.

9 Trace to Requirements

Req. ID	**Test ID's**
FR1	TC-FR1, TC-FR2
FR3	TC-FR3, UT-UM3
FR4	TC-FR4, UT-LG7
FR5	TC-FR5, UT-UA2
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 9: Requirements to Test Case Traceability Matrix

10 Trace to Modules

Note: * indicates that any test prefixed with the test case ID, covers the given module

Module	**Test ID's**
Database Interaction	UT-DB*
Logging	UT-LG*
File Storage	UT-FS*
Location Verification	UT-LV*
User Management	UT-UM*
User Authentication	UT-AU*

Table 10: Module to Test Case Traceability Matrix

11 Code Coverage Metrics

The unit testing achieves 95% line coverage and 90% branch coverage. This is checked by our GitHub Actions on every pull request.

Appendix — Reflection

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

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?

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

3. Which parts of this document stemmed from speaking to your client(s) or a proxy (e.g. your peers)? Which ones were not, and why?

4. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)