System Verification and Validation Plan for Software Engineering

Team 21, Alkalytics
Sumanya Gulati
Kate Min
Jennifer Ye
Jason Tran

November 4, 2024

Revision History

Date	Version	Notes
4 Nov 2024	0.0	Add initial draft for PoC.

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test

[symbols, abbreviations, or acronyms — you can simply reference the SRS (Author, 2019) tables, if appropriate —SS]

[Remove this section if it isn't needed —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

2 General Information

2.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

2.2 Objectives

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

[You should also list the objectives that are out of scope. You don't have the resources to do everything, so what will you be leaving out. For instance, if you are not going to verify the quality of usability, state this. It is also worthwhile to justify why the objectives are left out. —SS]

[The objectives are important because they highlight that you are aware of limitations in your resources for verification and validation. You can't do everything, so what are you going to prioritize? As an example, if your system depends on an external library, you can explicitly state that you will assume that external library has already been verified by its implementation team. —SS]

2.3 Challenge Level and Extras

[State the challenge level (advanced, general, basic) for your project. Your challenge level should exactly match what is included in your problem statement. This should be the challenge level agreed on between you and the course instructor. You can use a pull request to update your challenge level (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

[Summarize the extras (if any) that were tackled by this project. Extras can include usability testing, code walkthroughs, user documentation, formal proof, GenderMag personas, Design Thinking, etc. Extras should have already been approved by the course instructor as included in your problem statement. You can use a pull request to update your extras (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

2.4 Relevant Documentation

[Reference relevant documentation. This will definitely include your SRS and your other project documents (design documents, like MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. You can create BibTeX entries for your documents and within those entries include a hyperlink to the documents. —SS]

Author (2019)

[Don't just list the other documents. You should explain why they are relevant and how they relate to your VnV efforts. —SS]

3 Plan

This section outlines the Verification and Validation (VnV) responsibilities of each team member and the team supervisor. It presents comprehensive verification plans for the Software Requirements Specification (SRS), design documents (including MIS, MG, and System Design), the VnV plan itself, and the system implementation.

Additionally, the automated testing and verification tools intended for use in the VnV process have been specified. Verification tasks will be carried out by the VnV team in a sequential manner as each phase progresses, with continuous verification of the SRS and design documents as source code is developed. Software validation will be conducted in parallel with other VnV activities, supporting the Proof of Concept (PoC) and final demonstration phases.

3.1 Verification and Validation Team

In this section, Table 1 presents the members of the VnV team designated to carry out the tasks specified in this document. For each team member, their role in the project's verification process is outlined, with key details highlighting their respective responsibilities.

Table 1: Validation and Verification Team Members and Responsibilities

Team Member	Role and Responsibilities
Bassel Abdelkader	Advisor and one of the primary reviewers of documentation; contributor to the validation of user-based testing, providing suggestions and feedback to improve software functionality and user experience.
Dr. Charles de Lannoy	Advisor and one of the primary reviewers of documentation; contributor to the validation of user-based testing, providing suggestions and feedback to improve software functionality and user experience.
Jason Tran	Review the work of other team members to uphold high standards, provide suggestions for improvement, and maintain feedback checklists for each work item, with an emphasis on backend code.
Jennifer Ye	Review the work of other team members to uphold high standards, provide suggestions for improvement, and maintain feedback checklists for each work item, with an emphasis on frontend code.
Kate Min	Review the work of other team members to uphold high standards, provide suggestions for improvement, and maintain feedback checklists for each work item, with an emphasis on unit testing.
Sumanya Gulati	Review the work of other team members to uphold high standards, provide suggestions for improvement, and maintain feedback checklists for each work item, with an emphasis on the documentation and VnV Plan.
Other Design Teams	Peer reviewers identify issues and offer feedback and suggestions for improving documentation.

3.2 SRS Verification Plan

The Software Requirements Specification (SRS) is essential for defining the project scope and guiding implementation. To ensure it aligns with project standards and objectives, a rigorous verification plan will be implemented, incorporating iterative feedback from peers and our assigned Teaching Assistant, Chris Schankula.

The verification process will begin with an initial review to evaluate the SRS for clarity, completeness, and alignment with project goals. Feedback will be systematically gathered and categorized to identify areas needing refinement. Exploratory assessment techniques will allow reviewers to interact with application prototypes and design mockups, ensuring requirements are realistically translated into application features.

3.2.1 Functional Testing

Functional testing based on the SRS will validate application behavior, confirming that each function performs according to specified requirements. The feedback will be integrated into the SRS, leading to revisions that enhance clarity and accuracy. A comprehensive final review will ensure all feedback has been effectively addressed, solidifying the SRS as a true reflection of the intended system.

3.2.2 Formal and Ad-hoc Reviews

The verification approach includes formal and ad-hoc feedback methods. Formal reviews will create updated checklists based on existing ones and grading feedback, while review meetings with the supervisor will identify mistakes and suggest improvements. Ad-hoc peer reviews from other design teams will also provide valuable insights. Key verification checks will ensure the adequacy, feasibility, and verifiability of requirements, along with traceability to use cases.

3.2.3 Checklist

An initial verification plan checklist will be maintained and updated over time, focusing on detailed descriptions, relevance, traceability, verifiability, and feasibility of requirements, as well as closing all reviewer-identified issues. By leveraging diverse insights, the SRS will be refined to facilitate a smooth transition into the development phase, laying a strong foundation for the project's success.

The following checklist has been created as an initial draft that will be iteratively updated as SRS reviews are completed over time:

- ☐ Does each functional requirement have a clear and precise description?
- ☐ Are the rationales for each requirement clearly articulated?
- ☐ Is there a defined fit criterion for each requirement that specifies success criteria?
- ☐ Are all functional requirements complete and unambiguous?
- ☐ Are all requirements consistent with one another, avoiding conflicts?
- ☐ Are all dependencies on external systems or components identified and addressed?

3.3 Design Verification Plan

This section delineates the strategies and procedures the team will employ to verify the correctness and reliability of the design of the Alkalytics application. This plan will serve as a guideline during the testing phase to ensure that the design aligns with the intended requirements and effectively mitigates potential hazards identified by the VnV team.

3.3.1 Document Review

After the completion of the initial draft of the design documentation (including the Management Guide (MG), Management Information System (MIS), and System Design documents), each member of the testing team will conduct a comprehensive review before submission. The objectives of this review process are to:

- Ensure that the system design aligns with all functional and non-functional requirements.

- Assess the accuracy of the documentation in describing the intended functionality and behavior of the system.
- Record and report any design elements that deviate from specified requirements for further discussion.

3.3.2 Review Meetings

A structured review meeting will be conducted with the supervisors once the design documents are completed. Additionally, peer reviews from classmates will provide critical suggestions for improvement.

3.3.3 Code Conformity Verification

The team will verify that the code adheres to SOLID design principles, ensuring modular and maintainable code structures.

3.3.4 Formal Team Review

A formal review with team members will occur after initial document creation, allowing for reflection on the design prior to final review. Checklists will be utilized to compare the design documents against the Software Requirements Specification (SRS) post-verification.

3.3.5 Checklist

This checklist will be updated as reviews progress to ensure comprehensive verification of the design documents:

- [ ] Are all requirements (functional and non-functional) traceable to at least one implementing module in the MG?
- [ ] Have all issues raised by reviewers been addressed and resolved?
- [ ] Do all modules and components conform to the SOLID design principles?
- [ ] Are all modules assigned unambiguous tasks with well-defined inputs and outputs?
- [ ] Is the design reflective of the database structure and functionality required for the application?

3.4 Verification and Validation Plan Verification Plan

The Verification and Validation (VnV) plan, i.e., this document, must also have a verification plan to ensure its correctness, completeness, and feasibility. The following methods will be applied to verify the document:

- **Internal document review**: All team members review each section to ensure quality and provide feedback/suggestions for improvement.
- **Peer review by classmates**: Another team reviews the contents of the V&V plan and provide feedback/suggestions for improvement.
- **Feedback integration**: The team refines the VnV plan appropriately after reviewing feedback received from internal reviews, peer reviews, and the grading Teaching Assistant (TA).
- **Mutation testing**: The team performs mutation testing by injecting mutations (i.e., faults) into the tests to evaluate whether the tests can detect the mutant and verify if its behaviour is the expected outcome.

The following checklist serves as a guide for verifying the VnV plan:

- ☐ Each verification plan is complete, feasible, and unambiguous.
- ☐ Roles and responsibilities for verification are explicitly and clearly defined.
- ☐ All requirements are covered by the test cases.
- ☐ Each test specifies clear inputs, expected outputs, and criteria for pass/fail.
- ☐ Test procedures are well-described.
- ☐ A traceability matrix is provided, mapping each test case to their relevant requirement(s).

3.5 Implementation Verification Plan

Both static and dynamic techniques will be employed to verify the implementation for the Alkalytics project, as outlined below:

- **Static Analyzers**: Linting tools will be used for static code analysis to identify bugs or stylistic errors, maintain code readability, and enforce coding standards. Specific tools to be used are listed in Section 3.6.

- **Code Inspection/Walkthroughs**: New features or significant code changes will be reviewed by at least two other team members before merging into the main branch. The primary developer may walk the reviewers through the code. Code reviews will be conducted through pull requests on GitHub; only reviewed and approved code may be merged into the main branch. This process ensures code quality, readability, functionality, and allow for mutual understanding of all system components among the team members.

- **System Testing**: System testing will verify that the application meets both functional and non-functional requirements as specified. Details regarding system testing procedures and planned test cases are outlined in Section **??** of this document.

- **Unit Testing**: Unit testing will verify the behaviour of the application's individual components to ensure they perform as expected. The unit testing plans in Section **??** will be developed upon completion of the design specification.

3.6 Automated Testing and Verification Tools

Sections 10 and 11 of the Development Plan discuss the tools to be used and coding standards the team will adhere to. Below is an expansion on the referenced sections:

- **Linters**: Flake8 (for Python) and ESLint (for JavaScript) are the linting tools the team will use, as outlined in Section 10 of the Development Plan. These tools will help maintain code quality by automatically detecting syntax errors, code inconsistencies, and potential bugs.

- **Unit Testing Frameworks**: The team will use Jest for front-end unit testing and Pytest for back-end unit testing.

- **Continuous Integration (CI)**: CI will be implemented using GitHub Actions, automating the building and testing processes throughout development. The team currently has a workflow that runs on all LaTeX documentation, and additional workflows for automated testing will be added as development progresses.

- **Code Coverage Metrics**: Code coverage will be assessed using the appropriate plugins/flags for the respective unit testing frameworks mentioned above (e.g., `pytest-cov` for Pytest, `--coverage` in Jest).

3.7 Software Validation Plan

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS]

[For those capstone teams with an external supervisor, the Rev 0 demo should be used as an opportunity to validate the requirements. You should plan on demonstrating your project to your supervisor shortly after the scheduled Rev 0 demo. The feedback from your supervisor will be very useful for improving your project. —SS]

[For teams without an external supervisor, user testing can serve the same purpose as a Rev 0 demo for the supervisor. —SS]

[This section might reference back to the SRS verification section. —SS]

4 System Tests

This section covers all tests for different areas of the system. Table 2 defines the relationships between functional and non-functional requirements defined in the SRS to test cases outlined in this document in the form of a traceability matrix.

4.1 Tests for Functional Requirements

The subsections below covers each major functional of the application, from uploading data to the various outputs of post query analysis. Many of the functions come from the same user flow. By testing the subsections below guarintees that all possible user flows involving the main application functioanlities are working as expected.

4.1.1 Data Input and Storage

The tests below provide a way to evaluate the correctness of data input and storagee for the following functional requirements:

- FR-1
- FR-2
- FR-3
- FR-4

1. FR-ST1

   Control: Manual

   Initial State: Database is running and ready to intake data

   Input: Dataset to be stored, in .CSV format

   Output: Data that is inputted is sent into the system, then labelled and stored successfully

   Test Case Derivation: When the data is sent to the system, the data should be stored somewhere and labelled properly before it can be queried.

   How test will be performed: The test can be performed by sending a test sample of varying sizes to the storage in the system.

4.1.2 Data Querying and Results

Th tests below provide a way to evaluate the data querying and visualization process of the system for the following functional requirements:

- FR-5
- FR-6
- FR-7
- FR-8
- FR-9

1. FR-ST2

   Control: Manual

   Initial State: The system is not running any jobs, and the user interface is cleared.

   Input: A selection of different combinations of parameters and datasets to be queried on

   Output: A human-readable and customizable visualization of correct results corresponding to the selected paramters from the input

   Test Case Derivation: The expected output of the system is based on the query parameters selected. A user expects the data analysis to match with what they asked for, and the user is allowed to customize the visuallized data.

   How test will be performed: The database will be queried using multiple combinations of parameters, and the results will be compared against the expected output. The outputted visualization will then be tested for customizability.

4.1.3 Data analysis

The tests below provide a way to evaluate the data analysis in the application for the following functional requirements:

- FR-10
- FR-11

1. FR-ST3

   Control: Manual

   Initial State: The application's cleared user interface which has not yet been used to query data, with no graph showing yet.

   Input: A combination of parameters to query on for a selected dataset

   Output: A small written human-readable paragraph explaining the input data.

   Test Case Derivation: To be able to understand the returned data in ways other than through a graph, a written response gives the user a variety of choices.

How test will be performed: The website interface will allow the user to pick a written analysis response. The application will look for patterns and trends in the data and will output the findings.

4.1.4 Data Hygiene

The tests below provide a way to evaluate how the application maintains the data hygiene of the datasets related to the following functional requirements:

- FR-12
- FR-13

1. FR-ST4

   Control: Manual

   Initial State: The application's cleared user interface which has not yet been used to query data, with no graph showing yet.

   Input: Dataset to be stored, in .CSV format

   Output: A log file documenting errors found in the input data and/or removals of missing data.

   Test Case Derivation: This is to ensure the efficency of the querying and ensuring the database is only as big as it needs to be. This will also ensure that errors are dealt with by the application and are recorded to document any inconsistancies to increase traceability.

   How test will be performed: After loading in a CSV file with some error in the data. A log file will be generated documenting the error after the application attempts to fix it.

4.1.5 User Access

This tests below provide a way to evaluate how the application allows for user login related to the following functional requirements:

- FR-14

1. FR-ST5

   Control: Manual

Initial State: User interface shows a login page, with no login credentials currently used

Input: Sample user credentials

Output: The page redirects to the page designated after login

Test Case Derivation: The system must be able to authenticate users properly, and when authenticated, they should be given access to the application

How test will be performed: Sample credentials with different combinations of characters will be used to log in to ensure the fields handle credentials correctly.

4.1.6 Data Export

The tests below provide a way to evaluate the export of query reports after a session for the following functional requirement:

- FR-15

1. FR-ST6 Control: Manual

   Initial State: User interface after multiple usages of data queries

   Input: User clicking the button for saving or downloading

   Output: Query report will be downloaded to the user's device

   Test Case Derivation: The user needs to be able to get a system generated report of the queries from their session, and be able to save or download that report as needed.

   How Test Will Be Performed: After making multiple queries on the data, the save/download button will be pressed to test functionality.

4.2 Tests for Nonfunctional Requirements

Non-functional requirements will diverge from typical functional requirements testing. Listed below are different testing methods that will be used throughout the section.
User Demo Assessment: We will have users engage in a demo of the application, and observe how they interact with the application with respect to the requirements and criteria defined in SRS.

4.2.1 Look and Feel Requirements

Look and feel testing is heavily subjective, and will require users to test. These requirements will be heavily tested with the user demo assessment.

1. NFR-LF1 (User Interface)

   NFR: LFR-1, LFR-3, LFR-4, LFR-5, LFR-6, LFR-7

   Type: User Demo, Manual

   Initial State: Fully functional application ready for user interaction, starting at the login page

   Input/Condition: User engagement with application

   Output/Result: Recorded observations of how the user was able to interact with the system

   How test will be performed:

   - User will be given temporary, working credentials
   - User will use the app, starting from the login page, and navigate through its functionality
   - An observer will record how long it takes for them to figure out functionality and how to navigate the application
   - Users will also be given a survey about the interface after testing

2. NFR-LF2 (Device Compatibility)

   NFR: LFR-2

   Type: Manual

   Initial State: Application running on multiple different devices

   Input/Condition: Manual tester's engagement with application

   Output/Result: A list of inconsistencies through multiple devices

   How test will be performed:

   - Each member of the team will use the application across multiple platforms, and try to identify any inconsistencies between them

4.2.2 Usability and Humanity Requirements

Usability is also a subjective area, and thus will also require the use of a user demo assessment.

1. NFR-UH1 (User Experience)

   NFR: UHR-1, UHR-4, UHR-5

   Type: User Demo, Manual

   Initial State: Fully functional application ready for user interaction, starting at the login page

   Input/Condition: User engagement with application

   Output/Result: Recorded observations of how the user was able to interact with the system without clicking the help button or asking for outside help*

   How test will be performed:

   - User will be given temporary, working credentials
   - User will use the app, starting from the login page, and navigate through its functionality
   - An observer will record how long it takes for them to figure out functionality and how to navigate the application

2. NFR-UH2 (Language and Localization)

   NFR: UHR-2

   Type: Manual

   Initial State: Fully developed application, starting at the login page

   Input/Condition: Manual tester using application

   Output/Result: A list of all identified language discrepancies

   How test will be performed:

   - A manual tester will navigate through each web page to ensure that the only language being used is English (US)

3. NFR-UH3 (System Notation)

   NFR: UHR-3

   Type: Manual

   Initial State: Application ready to take in data

   Input/Condition: Sample input data

   Output/Result: Error logs from unrecognized characters, or successful upload

   How test will be performed:

   - A tester will upload a file that has scientific and mathematical symbols
   - They will note whether or not the file was uploaded successfully, and if the data was transferred correctly, with the symbols

4. NFR-UH4 (Accessibility)

   NFR: UHR-6

   Type: Manual

   Initial State: Application ready for use

   Input/Condition: Tester engagement using third party tools

   Output/Result: List of accessibility issues in accordance to WCAG, and checklists for if page is screen-readable

   How test will be performed:

   - A tester launch third party tools (NVDA Reader, SiteImprove)
   - Third party tools will examine web page and provide results

4.2.3 Performance Requirements

These requirements can be tested in ways similar to functional requirements. Many of these requirements have thresholds defined, making testing a pass/fail basis.

Test Data Generation: Some tests will require a large, dummy set of data to be used as input in order to test how well the system can handle heavy payloads. Thus, large test sets of data will be automatically generated accordingly.

1. NFR-P1 (Upload Speed)

   NFR: PR-1

   Type: Manual

   Initial State: Application navigated to upload page, ready to upload file

   Input/Condition: Sample .CSV files for input

   Output/Result: Upload duration

   How test will be performed:

   - Different .CSV files of varying sizes will be uploaded to the system
   - The upload functionality will include instructions for the system to time how long the upload took
   - The total duration will be logged on the browser's console.

2. NFR-P2: (Query Response Time)

   NFR: PR-2, PR-4

   Type: Manual

   Initial State: Application navigated to querying page, ready to make query

   Input/Condition: Test queries

   Output/Result: Query response durations

   How test will be performed:

   - A set of parameter/dataset combinations will be used as queries
   - The duration of each query will be logged and compared to defined criteria

3. NFR-P3 (Website Response Time)

   NFR: PR-3

   Type: Manual

   Initial State: Application ready to use

   Input/Condition: Tester engagement

   Output/Result: Average response time of buttons on website

   How test will be performed:

   - A tester will have a timer
   - They will time the response time of different buttons, and record it

4. NFR-P4 (Data Visualization Speed)

   NFR: PR-5

   Type: Manual

   Initial State: Application navigated to query page, ready to make query to generate graphs

   Input/Condition: Query parameters

   Output/Result: Time taken to generate graphs/visualizations

   How test will be performed:

   - A set of parameter/dataset combinations will be used as queries
   - A tester will manually time how long it takes for a graph to be generated, or the system will log it in the console

5. NFR-P5 (System Accuracy)

   NFR: PR-6, PR-7, PR-8

   Type: Manual

   Initial State: Application navigated to query page, ready to make query

   Input/Condition: Query parameters

Output/Result: A check for precision of numbers in different components of system

How test will be performed:

- A set of parameter/dataset combinations will be used as queries
- A tester will manually time how long it takes for a graph to be generated, or the system will log it in the console

6. NFR-P6 (Robustness - Backend Disruption)

   NFR: PR-9

   Type: Manual

   Initial State: Application running as normal

   Input/Condition: Tester temporarily taking down back end

   Output/Result: Error message displayed

   How test will be performed:

   - The tester will run the application as normal
   - The tester will then disable back end services and ensure error messages are generated

7. NFR-P7 (Robustness - Internet Disruption)

   NFR: PR-10

   Type: Manual

   Initial State: Application running as normal

   Input/Condition: Tester temporarily disconnects internet connection

   Output/Result: Previously generated plots and previous queries still working

   How test will be performed:

   - The tester will run the application as normal
   - The tester will then disconnect their device from their internet connection

- They will then try to load previous queries and previously generated plots

8. NFR-P8 (User Capacity)

   NFR: PR-11

   Type: Manual

   Initial State: Three devices ready to run application

   Input/Condition: Multiple different queries run by the different devices

   Output/Result: System response time while under load

   How test will be performed:

   - Multiple devices will make queries simultaneously
   - Page responsiveness will be measured while system runs multiple queries

9. NFR-P9 (Storage Capacity)

   NFR: PR-12

   Type: Data Generation, Automated

   Initial State: A database with a known amount of experiment data

   Input/Condition: Large suite of dummy test data

   Output/Result: Observations on system health after large payload

   How test will be performed:

   - Multiple, dummy sets of experiment data will be automatically generated
   - Using a script, this data will be uploaded into the database
   - System health will be monitored after upload is complete

4.2.4 Operational and Environmental Requirements

These tests are to ensure that the application works in the expected environment, for the expected users. They will be manual tests done by a person to check for an environment's compatibility with the system.

1. NFR-OE1 (Operating Environment)

   NFR: OER-2, OER-3, OER-4

   Type: Manual

   Initial State: Application running on a windows device as a web application on a Chromium based browser.

   Input/Condition: Tester engagement

   Output/Result: A list of all discovered issues with the application that arise due to environment compatibility

   How test will be performed:

   - A tester will run the web application on their Windows device, on Google Chrome
   - They will use the application as normal and try to find any issues that are caused due to operating environment

2. NFR-OE2 (User Onboarding)

   NFR: OER-5

   Type: Manual, User Demo

   Initial State: Application running and ready for use on home screen

   Input/Condition: User engagement

   Output/Result: Survey results depicting subjective complexity of onboarding process

   How test will be performed:

   - A user will go through the onboarding process
   - A survey will be given after asking the user how complex they found the onboarding to be

4.2.5 Maintainability and Support Requirements

These tests aim to ensure that future users of the application will not have any issues using the application.

1. NFR-MS1 (Interface Intuitiveness)

   NFR: MSR-4

   Type: Manual, User Demo

   Initial State: Application running, ready for use

   Input/Condition: User engagement

   Output/Result: Observations on user's ability to complete tasks without support

   How test will be performed:

   - A pre-defined list of tasks will be created
   - A user will be given this set of tasks to perform on their own
   - The user will be observed to see if they are able to perform these tasks without outside help from those conducting the test

2. NFR-MS2 (Cross-Browser Compatibility)

   NFR: MSR-6

   Type: Manual

   Initial State: Multiple Chromium-based web browsers open

   Input/Condition: Tester Engagement

   Output/Results: All abnormal behaviour of web pages observed on each different web browser

   How test will be performed:

   - A tester will use the application as normal on multiple different browsers
   - They will note down any issues that they find, especially if the issues are unique to the usage of a certain browser

4.2.6 Security Requirements

These security assessments are designed to verify that the application operates securely within the intended environment and meets the needs of its users. These evaluations will be conducted manually by a member of the VnV team described in 3.1 to ensure that the system adheres to established security standards and practices, identifying any potential vulnerabilities or misconfigurations within the environment.

1. NFR-SR1 (Authentication)

   NFR: SR-1

   Type: Manual

   Initial State: Application login page is displayed.

   Input/Condition: Tester attempts to access the application with various credentials.

   Output/Result: Access is granted or denied based on the validity of the credentials provided.

   How test will be performed:

   - The tester will attempt to log into the application using both valid and invalid credentials.
   - Valid credentials will be verified against the application's user database.
   - Invalid credentials will include common mistakes (e.g., incorrect passwords, unregistered usernames) to ensure that the system properly restricts access.
   - The tester will also verify the application's response to unauthorized access attempts, noting any error messages or behavior.
   - The testing will include attempts to access the application without logging in to ensure the login mechanism is enforced.

2. NFR-SR2 (Permissions)

   NFR: SR-2, SR-3

Type: Manual

Initial State: Application logged in with multiple user roles (e.g., admin, editor, viewer).

Input/Condition: Tester interacts with the application using on different user roles.

Output/Result: Access to query or modify data and perform sensitive operations is restricted according to user roles.

How test will be performed:

- The tester will attempt to query or modify the same data in the application using each role one by one.
- For each role:
  - Level 1: Should be able to query, modify and export all data.
  - Level 2: Should be able to query and modify only specific data (as per role permissions), should not be able to export any data.
  - Level 3: Should be able to query data but not modify it, should not be able to export any data.
- Verify that appropriate error messages are displayed for unauthorized access attempts, and check if these events are logged for security auditing.

3. NFR-SR3 (Timeout)

   NFR: SR-4

   Type: Manual

   Initial State: User is logged into the application.

   Input/Condition: User remains inactive for a specified period.

   Output/Result: User is automatically logged out after a predefined period of inactivity.

   How test will be performed:

   - The tester will remain idle (not performing any actions such as clicks, key presses, or navigations) for a predetermined period that matches the session timeout setting (e.g., 10 minutes).

- During the inactivity period, the tester will keep track of the time elapsed and ensure no session activity occurs.
- After the specified inactivity period, the tester will verify that the session times out and the user is automatically logged out.
- The user must confirm that the user is redirected to the login page and a message is displayed, indicating that the session has timed out due to inactivity.
- Attempt to navigate to any application page after the timeout occurs without logging back in.
- Ensure that access is denied, confirming that the session has ended.

4. NFR-SR4 (Data Input Integrity)

   NFR: SR-5, SR-9

   Type: Manual

   Initial State: Application is open, ready for data entry and CSV upload.

   Input/Condition: Tester submits various data entries, including valid and invalid values, and uploads CSV files with both valid and invalid data formats.

   Output/Result: All invalid inputs and CSV uploads are rejected, and only valid data entries are processed.

   How test will be performed:

   - The tester will input data into different fields that accept data inputed directly by users, using a variety of both valid and invalid formats (e.g., date formats, numeric fields, text fields).
   - For valid data, ensure the fields accept data in the correct format (e.g., "DD-MM-YYYY" for dates, numerical values for numeric fields).
   - For invalid data, verify that the application rejects these entries with appropriate error messages.
   - Once that is done, the tester uploads a CSV file containing correctly formatted, complete data. Confirm that the application accepts the file and processes the entries.

- To test the invalid data, the tester:
  - *Incorrect Format*: Upload a CSV file with incorrect or mismatched headers. Verify that the application rejects the file and provides a descriptive error.
  - *Corrupted/Incomplete Data*: Upload a CSV file with missing values, invalid characters, or mismatched data types (e.g., text in numeric columns). Confirm that the application detects and rejects the file due to validation issues.
  - *Boundary Testing in CSV*: Test with CSV files containing data at boundary limits (e.g., maximum character limits in text fields, maximum/minimum numerical values).

5. NFR-SR5 (Data Validation)

   NFR: SR-6, SR-7, SR-8

   Type: Manual

   Initial State: Application database contains a set of unique, validated records. Application interface is open for data entry, processing, and transfer actions.

   Input/Condition: Tester processes and transfers various data entries, including attempts to introduce duplicate records and test transfer accuracy.

   Output/Result: Duplicate records are detected and prevented, and data accuracy is maintained during all transfer operations.

   How test will be performed:

   - The tester will trigger any automated processes (e.g., batch processing, import) that might introduce duplicates.
   - The application should flag and reject duplicate entries, ensuring they are not added to the system.
   - After transfer, perform a consistency check by comparing a subset of transferred records with the original database.
   - As part of the consistency check, ensure that no data is modified unnecessarily.

6. NFR-SR6 (Data Storage Capacity)

   NFR: SR-12

   Type: Manual/Automated

   Initial State: Database is operational, with storage capacity at or below normal usage. Alert system is configured, and the administrator contact information is set up to receive notifications.

   Input/Condition: Tester simulates increasing database storage usage to exceed the 80% threshold.

   Output/Result: System successfully detects when storage usage exceeds 80% and sends a timely alert to administrators.

   How test will be performed:

   - Gradually add data to the database to simulate storage increase. This can be done by adding records, uploading large files, or running data generation scripts until storage usage surpasses the 80% threshold.
   - Verify that the system immediately detects the threshold breach and triggers an alert to administrators.
   - Review the alert content to ensure it provides clear information, including:
     - Current storage usage percentage.
     - Implications of reaching the threshold (e.g., potential performance impact).
     - Recommended actions for the administrator (e.g., freeing up space or provisioning additional storage).
   - After testing, reduce storage usage (e.g., by deleting test data) to observe if the system updates the storage capacity status accordingly.
   - Ensure that no system interruptions or crashes occur during and after the alert, verifying that the system remains functional even when approaching capacity limits.

7. NFR-SR7 (System Audits)

NFR: SR-13, SR-14, SR-15

Type: Manual/Automated

Initial State: The application is operational, with logging features configured and permissions for accessing logs assigned to administrators only.

Input/Condition: Tester performs various access and modification actions within the application, then attempts to access the audit logs with both authorized and unauthorized user accounts.

Output/Result: All actions are logged with timestamps and user identities, and audit logs are accessible only to authorized users, with proper encryption.

How test will be performed:

- Tester performs a range of actions within the application, including logging in and out of the application, accessing different data records and modifying specific data fields.
- Verify that an entry is created in the audit log capturing the type of action (e.g., login, access, modification), timestamp, and user identity.
- Using an authorized account with administrative privileges, retrieve the audit logs to verify the following:
  - That each performed action is accurately recorded.
  - That no events are missing, confirming 100% coverage of all access and modification events.
  - Check for any inconsistencies or inaccuracies in timestamps or user identifiers.
- The tester attempts to access the audit logs using a non-administrative (unauthorized) account.
- The system should deny access to the audit logs, displaying an error or access-restricted message.
- The tester attempts to modify an audit log entry (if possible) to ensure that the logs are tamper-resistant. The system should prevent any unauthorized changes, preserving the integrity of the logs.

8. NFR-SR8 (Intrusion Prevention)

   NFR: SR-16

   Type: Manual

   Initial State: Application login screen is open. Administrator contact information is configured to receive security alerts.

   Input/Condition: Tester attempts multiple failed logins to trigger the suspicious activity detection mechanism.

   Output/Result: Application detects and blocks access after three failed attempts, sends an alert to administrators, and locks out the user temporarily.

   How test will be performed:

   - Using an invalid username-password combination, the tester attempts to log in repeatedly.
   - After each failed attempt, verify that the system accurately counts the login failures.
   - On the fourth failed attempt, the application should block further login attempts for that user account or IP address temporarily.
   - Confirm that an alert is sent to administrators upon detecting the suspicious activity.
   - The alert should provide details about the incident, such as:
     - IP address and location.
     - Timestamp of the failed attempts.
     - Username or account targeted.
   - Check that each failed login attempt and the subsequent lockout are recorded in the audit logs, providing a clear trail for security auditing.

9. NFR-SR9 (Resource Optimization)

   NFR: SR-17

   Type: Automated/Manual

Initial State: Application is running under typical workload conditions. Monitoring tools and alerts for CPU and memory usage are enabled.

Input/Condition: Tester simulates a high workload to approach system resource limits, observing the system's monitoring and optimization response.

Output/Result: The system actively manages CPU and memory usage, preventing overload and maintaining performance stability.

How test will be performed:

- Gradually increase the workload on the system (e.g., by running intensive processes, generating simultaneous user sessions, or processing large data sets) until CPU and memory usage approach high utilization levels.
- Observe the system's resource usage in real time to confirm that CPU and memory metrics are accurately monitored.
- Confirm that the system remains stable, responsive, and does not crash or slow down significantly during and after the high-load test.
- As resource usage approaches critical levels (e.g., 90%), verify that the system triggers alerts to administrators, notifying them of potential overload risks.
- Check that the system dynamically adjusts resource allocation to manage load (e.g., by limiting non-critical processes or optimizing memory usage).
- Post this, CPU and memory usage should stabilize below critical thresholds (e.g., preventing usage from exceeding 90%) due to the system's proactive adjustments.
- Check that all significant resource usage events, optimization actions, and alerts are logged for auditing and system performance analysis.

4.3 Traceability Between Test Cases and Requirements

Table 2: Traceability Matrix for Test Cases and Requirements

Req. ID	**System Test ID**
FR-1 FR-2 FR-3 FR-4	FR-SLN1
FR-5 FR-6 FR-7 FR-8 FR-9	FR-SLN2
FR-10 FR-11	FR-SL3
FR-12 FR-13	FR-SL4
FR-14	FR-SL5
FR-15	FR-SL6
UHR-1 UHR-4 UHR-5	NFR-UH1
UHR-2	NFR-UH2
UHR-3	NFR-UH3
UHR-6	NFR-UH4
PR-1	NFR-P1
PR-2 PR-4	NFR-P2
PR-3	NFR-P3
PR-5	NFR-P4
PR-6 PR-7	NFR-P5

PR-8	
PR-9	NFR-P6
PR-10	NFR-P7
PR-11	NFR-P8
PR-12	NFR-P9
OER-2 OER-3 OER-4	NFR-OE1
OER-5	NFR-OE2
MSR-4	NFR-MSR1
MSR-6	NFR-MSR2
SR-1	NFR-SR1
SR-2 SR-3	NFR-SR2
SR-4	NFR-SR3
SR-5 SR-9	NFR-SR4
SR-6 SR-7 SR-8	NFR-SR5
SR-12	NFR-SR6
SR-13 SR-14 SR-15	NFR-SR7
SR-16	NFR-SR8
SR-17	NFR-SR9

5 Unit Test Description

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

Author Author. System requirements specification. https://github.com/..., 2019.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]

Appendix — Reflection

[This section is not required for CAS 741 —SS]

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

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. 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.

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?