System Verification and Validation Plan for SFWRENG 4G06A

Team #25, RapidCare
Pranav Kalsi
Gurleen Rahi
Inreet Kaur
Moamen Ahmed

November 4, 2024

Revision History

Date	Version	Notes
02-11-2024	Gurleen Rahi	Summary, Objectives, Challenge Level and Extras, Relevant Documentation
03-11-2024	Inreet Kaur	Tests for functional requirements and safety and security requirements, traceability matrix
03-11-2024	Pranav	VnV Verification Plan, Implementation Verification Plan, Automated Testing Plan + Tools, Software Validation Plan
03-11-2024	Moamen Ahmed	Test for NFRs and safety and security requirements
04-11-2024	Inreet Kaur	Verification and Validation team, SRS Verification Plan, Design Verification Plan
04-11-2024	Gurleen Rahi	Tests for functional requirements and safety and security requirements, traceability matrix
04-11-2024	Pranav Kalsi	Tests for functional requirements and safety and security requirements
04-11-2024	Gurleen Rahi	Reflection
04-11-2024	Pranav Kalsi	Usability Survey

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

symbol	description
SRS	Software Requirements Specification
MG	Module Guide
MIS	Module Interface Specification
FR	Functional Requirements
NFR	Non-Functional Requirements

2 General Information

2.1 Summary

This document provides a comprehensive description of the system tests specific to all requirements for a software application, RapidCare, that aims to streamline the healthcare documentation process aimed to be run as a web application. This document will be used as a verification tool to ensure that system fulfills all the requirements and needs of the user. This document will allow for an in-depth description of system tests for functional, non-functional, safety and security requirements. Additionally, it will outline the verification plan for design and implementation which will make sure that all requirements are fulfilled. Along with this, the document will also include the software validation plan that will validate the requirements by making sure that the client who intends to use this application is satisfied with its features. The system will allow the users to automate the documentation process by transcribing audio to text and generating reports based on the transcribed data. The system will also provide diagnostics and medicine suggestions based on the reports to further optimize the process.

2.2 Objectives

The objective of verification and validation plan is to build confidence in the correctness of the software and decide if the software is correctly following the requirements. Additionally, we want to achieve adequate usability of our software by making sure it satisfies with the user's needs, thus speeding up the documentation process. Another goal of this document is to test the accuracy of the input data to reflect that the software suggests correct diagnosis based on the written text that has been transcribed from the audio conversation. Assuming that the external libraries are already tested by their implementation team, the priority of this document will be to test the requirements of the software to ease the patient care. Due to time constraints and limited resources, we are not going to test the quality of usability. While the usability is important to test, the priority of this document is to test the fundamental requirements to ensure the software is accurate and reliable.

2.3 Challenge Level and Extras

In terms of the project challenge level, the project will come in the general category. The reasoning for this choice is supported below:

- **Domain Knowledge** – The documentation process has a lot of ins and outs which may differ between health organizations. Our supervisors and stakeholders will provide us with a base of the requirements, but further elicitation will be required to ensure that the requirements reflect a problem that truly exists. Additionally, since the whole patient journey is tracked we will have to survey other hospital staff as well to gain a further understanding.
- **Implementation Challenges** – There will be quite a few microservices required for this project where each microservice has high complexity. The performance of the microservices must be high as this use case requires quick response time. Additionally, since we are dealing with patient data security and privacy must be upheld. Lastly, the integration between all of the parts must be secure and undergo rigorous integration testing.

As part of the extras for this project, we will accomplish the following extras:

- **Usability testing** – This is designed for the users to assess how easily they can navigate and use the software. This will ensure that the requirements of the software perfectly align with the user's needs.
- **User documentation** – This will include instructions that will help the user to get started with the software. It will have an overview of the software along with some help resources for setup instructions and easy navigation for the user.

2.4 Relevant Documentation

The documents that are relevant to this project are:

- SRS (Kaur et al., 2024c)
- Hazard analysis (Kaur et al., 2024b)

- MG
- MIS

SRS and Hazard analysis documents list the FR, NFR, safety and security requirements which will assist us in developing tests for each of them. Moreover, MG and MIS documents give a structured approach of system's architecture and interface, ensuring that the system is thoroughly tested. This will help us to develop test cases for vital areas of the software.

3 Plan

This section will include details about the verification and validation team. In addition to this, this section will include SRS, design, validation and verification plan, and implementation verification plan. Lastly, it will outline details about automated testing and verification tools and software validation plan.

3.1 Verification and Validation Team

Here is a basic outline of the verification and validation team and their responsibilities. Please note that team members will switch roles between the tasks to ensure a well-rounded verification of the system. All team members will work collectively toward user documentation for this project.
Gurleen Rahi: Will focus on functional testing, specifically for transcription and report generation module. Will also focus on usability survey and design verification throughout the project.
Pranav Kalsi: Will focus on functional testing, specifically functional tests related to machine learning models. Also, will focus on integration testing and code verification throughout the project.
Inreet Kaur: Will focus on functional testing, specifically for various components in the data layer. Will also focus on safety requirement testing for this project.
Moamen Ahmed: Will focus on functional testing specifically for authentication and account management components. Will also focus on non-functional testing of the system.
Kristen Burrows (Supervisor): Will help the team verify and provide feedback to improve the SRS, design, and verification and validation plan verifi-

cation plan.

3.2 SRS Verification Plan

Each team member will perform a detailed review of the SRS to verify clarity, feasibility, and consistency across requirements. Identified issues will be added to the appropriate GitHub issue of the specific section. The team members can then make appropriate changes based on the feedback. Each team member will complete functional requirement tests for their assigned components as well as non-functional and safety requirement testing (as assigned). Once completed, at least two other team members will review and verify the tests using the sample input for the tests and verify the desired behavior. We will also ask for feedback from our peers, i.e. have another team review SRS and update it based on the feedback in the GitHub issues. In addition to this, we will then conduct a structured meeting review with our supervisor to gain feedback on the correctness and relevance of the requirements of the SRS document. During this meeting, we will have a walk-through of the key sections of the SRS, outline any challenging or critical requirements, and discuss any issues flagged during internal reviews. Once the system is complete, we will then have the supervisor and other stakeholders verify the system's functionality using specified inputs. They will also complete the usability survey (section 6.2) for the system.

3.3 Design Verification Plan

Our design verification plan ensures that the system design meets requirements for consistency, maintainability, scalability, and usability. This process will involve systematic reviews, integration testing, and structured feedback sessions with the supervisor and stakeholders.

Each team member will review the design of specific components to verify consistency with project requirements, maintainability, and scalability. Identified issues will be added to the appropriate GitHub issue of the specific section. The team members can then make appropriate changes based on the feedback. The team will also conduct integration testing to check the compatibility of different components of the system and compatibility with the hardware.

Once the system is completed, the team will conduct a review session with

the supervisor. In this meeting, we will provide a high-level overview of architecture and key components. We will present a sample user journey and gather feedback on the system's design for improvement.

Following is a checklist for the design verification plan:

- Verify the design meets all requirements identified in SRS and hazard analysis
- Verify each module is modular and maintainable
- Verify the design meets the coding standards
- Verify hardware and software compatibility
- Verify alignment with project objectives and stakeholder needs

3.4 Verification and Validation Plan Verification Plan

This document must be verified to make sure it is robust and reliable. The plan must have extensive coverage to ensure that all requirements are covered. The two main methods we will be using are the following:

- **Mutation Testing**: This is essentially making small changes to the code to see if the test cases can detect the changes. This ensures the test cases are robust.
- **VnV Reviews**: We will identify gaps in the Validation and Verification plan through reviews that we conduct with our peers and supervisors. This will help ensure completeness, accuracy, and feasibility of the plan.

Through both of these processes we will be able to ensure that the plan covers all input and is clear and consistent such that it can be carried out by any reader.

3.5 Implementation Verification Plan

The implementation verification plan is really a lot of static review this is include code walkthroughs and code inspections in conjunction with static analyzers. The plan will be as follows:

1. Static analysis will be present at the CI/CD pipeline level essentially whenever code is committed or a pull request is created. This will help catch code errors or style violations.
2. Next, the team will do a high level code walkthrough to catch any high level logic errors.
3. Finally, in any critical sections or features we will do a thorough code inspection where we will go through a detailed checklist to ensure the implementation is verified.

This strategy will be implemented in section 4.

3.6 Automated Testing and Verification Tools

An outline of the CI/CD strategy can be found in section 7 of the Development Plan (Kaur et al., 2024a). A list of relevant frameworks will be found in section 10 of the same document.

3.7 Software Validation Plan

Validation of the project is critical and our supervisor will be vital for this process. Our supervisor will make sure that our Software solves the user groups needs. Our supervisor is in an excellent position to validate the software as not only are they a domain expert they also fit the potential user group.

4 System Tests

This section will include system tests for functional, non-functional, and safety and security requirements. In addition to this, we will include a traceability table for test cases and requirements.

4.1 Tests for Functional Requirements

This section contains the tests for the Functional Requirements. The subsections for these tests were created based on the subsections of the Functional Requirements listed in the SRS. Traceability for these requirements and tests can be found in the section 4.4.

4.1.1 Add a document to the database

This subsection covers FR1, FR4, and FR8 from of the SRS document by testing that the system is able to add a document to the database only when a valid input is provided.

1. test-FR1,4,8-1

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input.

   Input: Correct and complete input data for all required fields.

   Output: A confirmation message and a new entry is added to the appropriate database.

   Test Case Derivation: The system will validate the input data, accept a complete and correct data input, and confirm a successful addition to the database.

   How test will be performed: The test controller will input a valid data object and check if the system is able to validate and accept the valid input. The controller will verify that a confirmation message appears and there is a valid new entry in the appropriate database.

2. test-FR1,4,8-2

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input.

   Input: Invalid input data.

   Output: An appropriate error message.

   Test Case Derivation: The system will validate the input data and prompt an error message outlining why the system is not able to accept the input data. No new document is added to the database.

   How test will be performed: The test controller will input an invalid data object and check if the system is able to validate and reject the invalid input. The controller will also verify that an error message appears and there is no new entry in the appropriate database.

4.1.2 Remove a document from the database

This subsection covers FR2, FR5, and FR9 from of the SRS document by testing that the system is able to remove a document to the database only when a valid input idetifier is provided.

1. test-FR2,5,9-1

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input. A document exists in the appropriate database.

   Input: A correct identifier for the document to be deleted.

   Output: A confirmation message and relevant entry no longer exist in the database.

   Test Case Derivation: The system should allow the deletion of an existing document when the correct identifier is provided.

   How test will be performed: The test controller will input a valid identifier and check if the system is able to validate and accept the valid input. The controller will verify that a confirmation message appears and the associated document no longer exists in the database.

2. test-FR2,5,9-2

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input. A document exists in the appropriate database.

   Input: An invalid identifier for the document to be deleted.

   Output: An appropriate error message.

   Test Case Derivation: The system should be able to validate the provided identifier and prevent the deletion of any existing document.

   How test will be performed: The test controller will input an invalid identifier and check if the system is able to validate and reject the invalid identifier. The controller will verify that an error message appears, and no document is deleted from the database.

4.1.3 Update document in the database

This subsection covers FR3, FR6, FR10, and FR11 from of the SRS document by testing that the system is able to update a document to the database only when a valid input idetifier is provided.

1. test-FR3,6,10,11-1

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input.

   Input: A correct identifier for the document to be updated.

   Output: A confirmation message and relevant entry shows the updated data in the database.

   Test Case Derivation: The system should allow to update the existing document when the correct identifier is provided.

   How test will be performed: The test controller will input a valid identifier and check if the system is able to validate and accept the valid input. The controller will verify that a confirmation message appears and the current document is updated in the database.

2. test-FR3,6,10,11-2

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input.

   Input: An invalid identifier for the document to be updated.

   Output: An appropriate error message.

   Test Case Derivation: The system should be able to validate the provided identifier and prevent the update of any document.

   How test will be performed: The test controller will input an invalid identifier and check if the system is able to validate and reject the invalid identifier. The controller will verify that an error message appears, and no document is updated in the database.

4.1.4 login for valid/invalid credentials

This subsection covers FR7 from of the SRS document by testing that the system is able to allow to access the database only when a valid input credentials is provided.

1. test-FR7-1

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input.

   Input: The correct credentials for login.

   Output: A confirmation message and user can access the homepage of the database.

   Test Case Derivation: The system should be able to validate the provided credentials and allow to login the database.

   How test will be performed: The test controller will input the valid credentials and check if the system is able to validate and accept the valid input. The controller will verify that a confirmation message appears and the user is able to login the database.

2. test-FR7-2

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input.

   Input: Invalid credentials for login.

   Output: An appropriate error message.

   Test Case Derivation: The system should be able to validate the provided credentials and prevent unauthorized access to the database.

   How test will be performed: The test controller will input any invalid credentials and check if the system is able to validate and reject the invalid credentials. The controller will verify that an error message appears and unauthorized access will be denied.

4.1.5 Voice-to-text-transcription check

This subsection covers FR7 from of the SRS document by testing that the system is able to transcribe audio data to the written text.

1. test-FR11-1

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input.

   Input: A valid sentence from the conversation between the healthcare professional and the patient.

   Output: A confirmation message indicating successful transcription and the screen shows the transcribed text.

   Test Case Derivation: The system should be able to validate the voice input by processing it and transcribe to written text in real-time.

   How test will be performed: The test controller will input a valid sentence from the conversation and check if the system is able to process the valid input. The controller will verify that a confirmation message appears and the voice input is transcribed to written text.

2. test-FR11-2

   Control: Manual

   Initial State: The system is set up, open to the relevant section (e.g., Healthcare Network, Healthcare Professional, or Patient Records), and ready to take the user input.

   Input: An unclear sentence from the conversation.

   Output: An appropriate error message.

   Test Case Derivation: The system should be able to validate the invalid input and prevent transcription of that sentence.

   How test will be performed: The test controller will input an unclear sentence and check if the system is able to validate and reject the input. The controller will verify that an error message appears and audio input will not be transcribed to written text.

4.1.6 Validate output of correct diagonsis and medication

This subsection covers FR12 and FR13 from of the SRS document by testing that the system is able to create predictions on the diagnosis and medicines with a high accuracy and confidence.

1. test-FR12,13-1

   Control: Automatic

   Initial State: The user has uploaded the blank chart, and the chart has been populated with the patients symptoms.

   Input: Correct and complete input data where all the symptoms fields of the chart are filled out.

   Output: A suggestion on what the diagnosis should be based on the symptoms, then based on the diagnosis provide possible medicine.

   Test Case Derivation: The system will create a validation set of data, where the model will be evaluated.

   How test will be performed: When the model is deployed it will be automatically evaluated such that it will have to have a classification accuracy greater or equal than 85%.

4.1.7 Validate input data for models

This subsection covers FR12, FR13, and IR5 from of the SRS document by testing that the system is able to validate the input data in the charts such that it may be inputted into the prediction module.

1. test-FR12,13,IR5-1

   Control: Automatic

   Initial State: The user has uploaded a blank chart.

   Input: Correct and complete input data where all the symptoms fields of the chart are filled out.

   Output: A suggestion on what the diagnosis should be based on the symptoms, then based on the diagnosis provide possible medicine.

   Test Case Derivation: The system will test if the chart data is relevant and correct.

How test will be performed: When the system is deployed the model input will be tested such that it only accepts valid inputs.

2. test-FR12,13,IR5-2

   Control: Automatic

   Initial State: The user has uploaded the blank chart a blank chart.

   Input: Incorrect and incomplete input data where all the symptoms fields of the chart are filled out.

   Output: An error message asking to fix the respective fields.

   Test Case Derivation: The system will test if the chart data is relevant and correct.

   How test will be performed: When the system is deployed the model input will be tested such that it only accepts valid inputs. It will output a error message if the output is wrong.

4.1.8 Verify completeness and correctness of data

This subsection covers FR14 from of the SRS document by testing that the data sent over is complete.

1. test-FR14-1

   Control: Manual

   Initial State: The user has a completed patient profile and wishes to send it over.

   Input: Correct and completed patient chart to be sent to another healthcare network.

   Output: A successful send where the data is validated at the endpoint, if incomplete data is received an error message should pop up.

   Test Case Derivation: The system will test if the patient profile properly got sent over.

   How test will be performed: Tester will send a sample patient profile from one dummy network to another, then the test cases will check if the integrity of the profile is in tact.

4.2 Tests for Nonfunctional Requirements

4.2.1 Aesthetic and Design (NFR1)

1. test-AD1

   Type: Non-functional, Dynamic, Manual

   Initial State: UI is designed and implemented.

   Input/Condition: Healthcare workers view the UI under normal operating conditions.

   Output/Result: Feedback collected on UI's aesthetic appeal and simplicity.

   How this test will be performed: A test group of users and the supervisor will be given the system, and a set of routine UI interactions to perform. They will be asked to complete these interactions using the interface. After they do this, they will be given a UI design survey. In the UI design survey, the supervisor will verify that the interface meets aesthetic appeal and simplicity requirements.

2. test-AD2

   Type: Non-functional, Dynamic, Manual

   Initial State: UI is operational and accessible to users.

   Input/Condition: Users interact with the UI during routine tasks and provides feedback.

   Output/Result: 80% of users report satisfaction with the UI design.

   How this test will be performed: A test group of users and the supervisor will be given the system, and a set of common tasks to perform.

They will be asked to complete these tasks using the interface. After they do this, they will be given a UI satisfaction survey. In the UI satisfaction survey, the supervisor will verify that at least 80% of users are satisfied with the design.

4.2.2 Usability Requirement(NFR2)

1. test-UR1

   Type: Non-functional, Dynamic, Manual

   Initial State: System is fully available and accessible to healthcare workers.

   Input/Condition: Healthcare workers perform key functions after a 30-minute training session.

   Output/Result: 90% of healthcare workers complete tasks without assistanceand within the expected time frame.

   How this test will be performed: A test group of healthcare workers and the supervisor will be given the system, and a set of key functions to perform after a 30-minute training. They will be asked to complete these functions independently. After they do this, they will be given a task completion survey. In the task completion survey, the supervisor will verify that 90% of workers completed tasks without assistance.

2. test-UR2

   Type: Non-functional, Dynamic, Manual

   Initial State: System is deployed and accessible.

   Input/Condition: Users navigate and explore the system features independently.

Output/Result: Majority of users can locate core functions without additional guidance.

How this test will be performed: A test group of users and the supervisor will be given the system, and a set of core functions to locate. They will be asked to find these functions without guidance. After they do this, they will be given a navigation survey. In the navigation survey, the supervisor will verify that users could locate core functions independently.

4.2.3 Performance Requirement(NFR3)

1. test-PR1

   Type: Non-functional, Dynamic, Automated

   Initial State: System with voice transcription feature is active.

   Input/Condition: Input of a 30-second audio recording under typical clinic noise levels.

   Output/Result: System completes transcription within 30 seconds with an accuracy of 85%.

   How this test will be performed: A test group of users and the supervisor will be given the system, and a set of 30-second audio recordings with typical clinic noise. They will be asked to process these recordings through the transcription system. After they do this, they will be given a performance survey. In the performance survey, the supervisor will verify the transcription speed and accuracy meet the 85% standard.

2. test-PR2

   Type: Non-functional, Dynamic, Manual

Initial State: Transcription interface open and ready.

Input/Condition: Real-time voice input provided by healthcare professional.

Output/Result: Real-time transcription displayed within a 1 second delay.

How this test will be performed: A test group of healthcare professionals and the supervisor will be given the system, and a set of voice inputs to transcribe. They will be asked to speak these inputs for real-time transcription. After they do this, they will be given a latency survey. In the latency survey, the supervisor will verify that transcription delay remains under 1 second.

4.2.4 Operational Requirement(NFR4)

1. test-OR1

   Type: Non-functional, Dynamic, Automated

   Initial State: System is live and connected to monitoring software.

   Input/Condition: 30-day operational period with intermittent load testing.

   Output/Result: Uptime is consistently maintained at 99.9% or above.

   How test will be performed: Use uptime monitoring tools to track server availability over the 30-day period.

2. test-OR2

   Type: Non-functional, Dynamic, Manual

Initial State: System in operational use.

Input/Condition: Users access the system over a 30-day period under normal and peak loads.

Output/Result: Log data and user feedback confirm uptime meets standards with no significant disruptions.

How this test will be performed: A test group of users and the supervisor will be given the system, and a set of usage patterns to follow over 30 days. They will be asked to use the system according to these patterns while monitoring performance. After they do this, they will be given an uptime survey. In the uptime survey, the supervisor will verify that no significant disruptions occurred during the test period.

4.2.5 Maintainability Requirement(NFR5)

1. test-MR1

   Type: Non-functional, Dynamic, Manual

   Initial State: System running on the latest version with recent update logs.

   Input/Condition: Regular software update is applied for bug fixes and improvements.

   Output/Result: System successfully applies updates without impacting stability.

   How this test will be performed: A test group of users and the supervisor will be given the system, and a set of software updates to apply. They will be asked to implement these updates following standard procedures. After they do this, they will be given a stability survey. In the stability survey, the supervisor will verify that system functionality remained stable after updates.

2. test-MR2

   Type: Non-functional, Dynamic, Manual

   Initial State: Previous version of the system with identified bugs or issues.

   Input/Condition: Apply updates addressing known issues.

   Output/Result: System functions as expected with resolved issues and no new errors introduced.

   How this test will be performed: While developers update the system, a test group of users and a supervisor will monitor it. The group will test functionality as updates are deployed, after which they'll complete a regression survey. The supervisor will verify that issues are resolved without introducing new errors.

4.2.6 Security Requirement(NFR6)

1. test-SR1

   Type: Non-functional, Dynamic, Automated

   Initial State: System with live patient data encryption protocols active.

   Input/Condition: Security audit test is performed on the system.

   Output/Result: No vulnerabilities are detected; all data remains encrypted in transit and at rest.

   How test will be performed: Conduct automated vulnerability scans and manual security audit to confirm full compliance with PIPEDA and encryption standards.

2. test-SR2

   Type: Non-functional, Dynamic, Manual

   Initial State: System fully operational with access logs enabled.

   Input/Condition: Simulate unauthorized access attempts.

   Output/Result: Unauthorized attempts are blocked; access logs capture details.

   How this test will be performed: A test group of users and the supervisor will be given the system, and a set of unauthorized access scenarios. They will be asked to attempt these scenarios while monitoring system responses. After they do this, they will be given an access control survey. In the access control survey, the supervisor will verify that all unauthorized attempts were properly blocked and logged.

4.2.7 Cultural Requirement (NFR7)

1. test-CR1

   Type: Non-functional, Dynamic, Manual

   Initial State: System operational in default language (English).

   Input/Condition: User selects an alternate language from the settings menu.

   Output/Result: System displays all content in the selected language without loss of functionality.

   How this test will be performed: A test group of users and the supervisor will be given the system, and a set of language switching scenarios. They will be asked to change the system language and verify content

display. After they do this, they will be given a language functionality survey. In the language functionality survey, the supervisor will verify that all content displays correctly in each language.

2. test-CR2

   Type: Non-functional, Dynamic, Manual

   Initial State: System set to the default language with alternative language packs installed.

   Input/Condition: User navigates through various sections in alternative language selected.

   Output/Result: No functional errors or misaligned text appear in the UI.

   How this test will be performed: A test group of users and the supervisor will be given the system, and a set of UI navigation tasks in different languages. They will be asked to complete these tasks in various language settings. After they do this, they will be given a language consistency survey. In the language consistency survey, the supervisor will verify that no text misalignment or functional errors occur.

4.2.8 Legal Requirement(NFR8)

1. test-LR1

   Type: Non-functional, Dynamic, Manual

   Initial State: System is fully functional and contains patient records.

   Input/Condition: Conduct a compliance audit against PIPEDA and other relevant data protection standards.

Output/Result: System passes all compliance checks with no exceptions.

How test will be performed: A certified compliance auditor verifies data handling, encryption protocols, and data access policies.

2. test-LR2

Type: Non-functional, Dynamic, Automated

Initial State: System storing and transmitting patient data over a network.

Input/Condition: Monitor data handling and transfer processes during operation.

Output/Result: All patient data is handled in compliance with regulations, without any unauthorized access or data breaches.

How test will be performed: Automated compliance tools track data handling practices over a defined period, and alerts are set up for any regulatory deviations.

4.2.9 Scalability (NFR9)

1. test-S1

Type: Non-functional, Dynamic, Automated

Initial State: System deployed on a test server environment capable of scaling horizontally.

Input/Condition: Simulate an increasing number of concurrent users accessing the system, starting from 100 users up to 10,000 users.

Output/Result: System maintains consistent performance and response times without a drop in performance.

How test will be performed: Use load testing tools like Apache to simulate concurrent user traffic and monitor system response times, server load, and throughput during the test.

2. test-S2

   Type: Non-functional, Dynamic, Automated

   Initial State: System operational with data processing services enabled.

   Input/Condition: Increase the data input rate gradually from standard load to peak operational load.

   Output/Result: System processes data without bottlenecks, maintaining efficient load distribution.

   How this test will be performed: Conduct automated stress tests with data processing simulators and monitor resource usage, CPU, memory, and load balancer performance to ensure scalability standards are met.

4.3 Tests for Safety and Security Requirements

4.3.1 Access Requirements Tests

1. test-AC1-1

   Type: Functional, Dynamic, Automated

   Initial State: System deployed with authentication module enabled and test user accounts configured.

Input/Condition: Attempt to access protected resources with invalid credentials 5 times consecutively from the same IP address.

Output/Result:

- System logs each failed attempt
- Account is temporarily locked after 5 failed attempts
- Security team receives notification
- User receives lockout notification

How this test will be performed: A test group of users and the supervisor will be given the system, and a set of sample invalid credentials for the system. They will be asked to use these credentials repeatedly to attempt system access. After they do this, they will be given a security test survey. In the security test survey, the supervisor will verify the logging, lockout, and notification requirements are met.

2. test-AC1-2

Type: Functional, Dynamic, Manual

Initial State: System operational with authentication logs enabled.

Input/Condition: Attempt to access system resources without authentication.

Output/Result:

- All unauthorized access attempts are blocked
- Each attempt is logged with timestamp, IP address, and attempted resource

How this test will be performed: A test group of users and the supervisor will be given the system, and a set of protected resources to access. They will be asked to attempt accessing these resources without authentication. After they do this, they will be given a security

test survey. In the security test survey, the supervisor will verify that access was properly blocked and logged.

3. test-AC2-1

   Type: Functional, Dynamic, Manual

   Initial State: System operational with standard user and admin accounts configured.

   Input/Condition: Attempt to create, update, and delete user accounts using non-admin credentials.

   Output/Result:

   - All unauthorized actions are blocked
   - Actions are logged with user details
   - Security team can review blocked attempts

   How this test will be performed: A test group of users and the supervisor will be given the system, and a set of standard user credentials. They will be asked to attempt various administrative operations using these credentials. After they do this, they will be given a security test survey. In the security test survey, the supervisor will verify that administrative actions were properly restricted and logged.

4.3.2 Integrity Requirements Tests

1. test-IR1-1

   Type: Non-functional, Dynamic, Automated

   Initial State: System operational with test user accounts and authentication database.

Input/Condition: Simulate multiple concurrent failed login attempts while monitoring credential storage.

Output/Result: User credentials remain unchanged and system maintains stability.

How this test will be performed: A test group of users and the supervisor will be given the system, and a set of test credentials for concurrent authentication testing. They will be asked to perform multiple simultaneous authentication attempts. After they do this, they will be given a system integrity survey. In the system integrity survey, the supervisor will verify that all credentials remained intact and system stability was maintained.

2. test-IR2-1

   Type: Functional, Dynamic, Manual

   Initial State: The system is set up, open to the relevant section and ready to take the user input.

   Input/Condition: Submit an invalid data input.

   Output/Result: An error message is displayed to the user. No data is added to any database.

   How test will be performed: The test controller will input an invalid data input and check if the system is able to validate and reject the invalid data. The controller will verify that an error message appears.

3. test-IR3-1

   Type: Functional, Static, Automated

   Initial State: The prediction model is ready for to predict medication and diagnosis.

Input/Condition: A test set of various patient medical charts will be inputted.

Output/Result: The prediction will create a diagnosis prediction based on the chart and a medication prediction based on the diagnosis. Each prediction will include a confidence score and the confidence score of the testing set must exceed 85%.

How test will be performed: Automated testing of model outputs and confidence scores this assists in understanding how sure the model is in its classification.

4. test-IR4-1

   Type: Functional, Dynamic, Manual

   Initial State: The system is set up, open to the relevant section and ready to take the user input. A document already exists in the relevant database.

   Input/Condition: A new input with the same data as an existing document.

   Output/Result: An error message. No new document is added to the database.

   How test will be performed: The test controller will have a new input with the same data as an existing document and check if the system is able to validate and reject the duplicate record. The controller will verify that an error message appears.

5. test-IR6-1

   Type: Functional, Dynamic, Manual

   Initial State: The system is set up, open to the relevant section and ready to take the user input. A document already exists in the relevant database.

Input/Condition: The written text transcribed from the audio data.

Output/Result: The data is correctly classified in the generated report with no diagnosis errors.

How test will be performed: The test controller will have a new data input and check if the system is able to validate and classify the data accurately. The controller will verify that there are no diagnosis errors in the generated report.

6. test-IR7-1

   Type: Functional, Dynamic, Manual

   Initial State: The system is set up, open to the relevant section and ready to take the user input.

   Input/Condition: The audio conversation between the patient and healthcare professional.

   Output/Result: The written text transcribed from the input data matches with the conversation with minimal to no interference.

   How test will be performed: The test controller will have a new data input and check if the system is able to validate and classify the data accurately. The controller will verify that the generated text matches the spoken conversation.

4.4 Traceability Between Test Cases and Requirements

Test ID	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10	FR11	FR12	FR13	FR14
test-FR1,4,8-1	×			×				×						
test-FR1,4,8-2	×			×				×						
test-FR2,5,9-1		×			×				×					
test-FR2,5,9-2		×			×				×					
test-FR3,6,10,11-1			×			×				×	×			
test-FR3,6,10,11-2			×			×				×	×			
test-FR7-1							×							
test-FR7-2							×							
test-FR11-1											×			
test-FR11-2											×			
test-FR12,13-1												×	×	
test-FR12,13,IR5-1												×	×	
test-FR12,13,IR5-2												×	×	
test-FR14-1														×

Table 1: **Functional Requirements Tests Traceability**

TestID	AD1	AD2	UR1	UR2	PR1	PR2	OR1	OR2	MR1	MR2	SR1	SR2	CR1	CR2	LR1	LR2	S1	S2
test-AD1	×																	
test-AD2		×																
test-UR1			×															
test-UR2				×														
test-PR1					×													
test-PR2						×												
test-OR1							×											
test-OR2								×										
test-MR1									×									
test-MR2										×								
test-SR1											×							
test-SR2												×						
test-CR1													×					
test-CR2														×				
test-LR1															×			
test-LR2																×		
test-S1																	×	
test-S2																		×

Table 2: **Non-Functional Requirements Tests Traceability**

Test ID	AC1	AC2	IR1	IR2	IR3	IR4	IR5	IR6	IR7
test-AC1-1	×								
test-AC-2	×								
test-AC-3		×							
test-IR-1			×						
test-IR-2				×					
test-IR-3					×				
test-IR-4						×			
test-FR12,13,IR5-1							×		
test-IR6-5								×	
test-IR7-6									×

Table 3: **Safety and Security Requirements Traceability**

5 Unit Test Description

5.1 Unit Testing Scope

To be completed in Rev 1.

5.2 Tests for Functional Requirements

References

Inreet Kaur, Pranav Kalsi, Moamen Ahmed, and Gurleen Rahi. Development plan: Rapidcare, 2024a. URL https://github.com/Inreet-Kaur/capstone/blob/main/docs/DevelopmentPlan/DevelopmentPlan.pdf.

Inreet Kaur, Pranav Kalsi, Moamen Ahmed, and Gurleen Rahi. Hazard analysis: Rapidcare, 2024b. URL https://github.com/Inreet-Kaur/capstone/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf.

Inreet Kaur, Pranav Kalsi, Moamen Ahmed, and Gurleen Rahi. Software requirements specification for software engineering: Rapidcare, 2024c. URL https://github.com/Inreet-Kaur/capstone/blob/main/docs/SRS/SRS.pdf.

6 Appendix

6.1 Symbolic Parameters

N/A

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.

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?
   This document has let us build more on the rough ideas of the requirements that we had brainstormed initially. While going through the outline of this document, we were able to provide the tests for functional requirements, non-functional requirements, and safety and security requirements. It also made us better understand the detailed procedure of system testing and plan for design validation and verification.

2. What pain points did you experience during this deliverable, and how did you resolve them?
   There are obstacles in any team project that must be overcome for it to proceed successfully. To ensure seamless operations, we had to develop a strategy for contributions. We had to collectively make a list of tests that are needed to be conducted for functional requirements, non-functional requirements, and safety and security requirements. We also needed to create a schedule to contribute to the template and review each other's work in the best way possible.

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.
As different abilities are added to the overall development plan, the project progresses more quickly thanks to the diversified knowledge of the team members. Technical expertise such as static and dynamic testing knowledge are very helpful to come up with a successful verification and validation plan for our project. Knowledge about how the external libraries can be used is also an asset for validating the system. Finally, being able to work on backend programming with an understanding of different server-side technologies, like Python, Java, and React.js will be helpful in creating a dynamic database that secures patient data.

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?
One can use a variety of resources to learn Java from Spring and Python from Flask to become effective in backend programming. With these resources, developers can gain practical experience by building web applications that offer them flexibility. Using LeetCode to practice coding skills is an additional strategy. This allows users to modify the difficulty of the problems and proceed with their solution. Moreover, setting goals and taking regular pauses between tasks might help with time management and allow one to work more effectively. It's critical that each member of the team grasp every talent for everyone to be moving at the same speed. This is because it's a fantastic chance to learn and implement it in practical situations.