AI for Chest X-Ray Read: System Verification and Validation Plan for Software Engineering

Team 17, Team RAdiAIdance
Allison Cook
Ibrahim Issa
Mohaansh Pranjal
Nathaniel Hu
Tushar Aggarwal

November 3, 2023

Revision History

Date	Version	Notes
10/30/2023	0.0	Initial Draft of VnV Plan
10/31/2023	0.1	Various Sections filled out with bullet points of content
11/01/2023	0.2	Symbols, Abbreviations, and Acronyms Section table and description completed; Introductory blurb and general roadmap for VnV Plan completed
11/01/2023	0.3	General Information Section completed
11/02/2023	0.4	Plan and System Test Description Sections mostly completed
11/03/2023	0.5	Plan and System Test Description Sections completed, reviewed entire document

Contents

List of Tables

List of Figures

1 Symbols, Abbreviations, and Acronyms

The following table, Table 1, includes the definitions and descriptions of all relevant symbols, abbreviations and acronyms used throughout this VnV Plan document.

Symbol, Abbreviation or Acronym	**Definition or Description**
AI/ML	Artificial Intelligence/Machine Learning
AUC	Area Under (ROC) Curve
BUC	Business Use Case
DICOM	Digital Imaging and Communications in Medicine, the standard for medical images to ensure data and quality necessary for clinical use.
FR	Functional Requirement
ICU	Intensive Care Unit
IT	Information Technology
MC	Mandated Constraints
MG	Module Guide
MIS	Module Interface Specification
NFR	Non-functional Requirement
PoC	Proof of Concept
ROC curve	Receiver Operating Characteristic curve, a graph to show the performance of a model, plots true positive rate and false positive rate
SRS	Software Requirements Specification
T	Test
VnV	Verification and Validation

Table 1: Symbols, Abbreviations and Acronyms

This document intends to discuss the verification and validation plan for the project's proposed solution in more detail. The General Information section summarizes the software being tested and gives a brief overview of its general functions. Also stated are the objectives of this VnV Plan, including both those in and out of the scope of this project and which ones will be prioritized. The relevant documentation that will be referenced in this plan will also be noted and described in detail.

In the Plan section, the VnV Team members are noted, with the plans for verifying the SRS, design, VnV plan and implementation. The automated testing and verification tools that will be used are also noted. The plan for validating the software will also be detailed. The general roadmap of this Verification and Validation plan starts with the first milestone of verifying the SRS. Then, the next milestones are to verify the design, the VnV plan itself and the implementation, likely in that order. These milestones will be reached using a combination of automated testing and verification tools noted in this document, as well as from meeting with this project's supervisor. After these milestones are completed, the software will be validated.

In the System Test Description section, the test cases for verifying and validating the functional and nonfunctional requirements will be described in more detail. A traceability table will also be provided to show the traceability between the various aforementioned test cases and the requirements for this project, as described in the SRS. The Unit Test Description section will be filled in later once the design and software implementation details are specified. The Appendix will be filled in as needed, and any tables or figures in this document will be listed in the table of contents as well.

2 General Information

2.1 Summary

The software being tested, [Insert Name], is a web-based application for performing AI-driven readings of chest x-ray images. The application will perform user authentication, retrieve patient information, read chest X-ray images using an AI model and generate radiology reports (elements) of its findings. This application will interface with a medical institution or diagnostic centre's IT systems to retrieve, analyze and save patient data,

information, and chest X-ray read results.

2.2 Objectives

The main objective of this VnV Plan that is intended to be accomplished is to build confidence in the software correctness of the AI model being trained to read chest x-ray images and generate radiology report components of its disease/condition findings and ensure the results are accurate. This objective can be broken down into the following points:

- Check the correctness of reading and identifying areas/labelling of diseases.
- Minimize false negatives/positives of possible diseases/conditions that are identified or missed.
- Check the correctness of the interpretation and display of the chest X-ray analysis.

Another major objective of this VnV Plan that is intended to be accomplished is to build confidence in the security and authentication capabilities of this software. This objective can also be broken down into points:

- Check the security of patient data, ensuring its secure storage and access
- Check the software correctly. authenticates authorized users and grants them access to patient data, information, and chest x-ray read results.
- Check the software correctly blocks unauthorized users and prevents them from accessing patient data, information, and chest x-ray read results.

A third final major objective of this VnV Plan that is intended to be accomplished is to build confidence in the software to effectively deliver on the first two aforementioned major objectives. This can be broken down into the following points:

- Check the software provides a usable interface for users to log in, be authenticated and then access patient data, information, and chest x-ray read results. (i.e. frontend functions)

- Check the software is able to interface with supporting IT systems (i.e. of a medical institution or diagnostic centre) and retrieve, display and save patient data, information, and chest x-ray read results across systems. (i.e. backend functions)

In short, the three main objectives of this VnV Plan that are intended to be accomplished are to build confidence in the software's:

- AI model to accurately read chest x-ray images and produce correct radiology results
- Security and authentication capabilities to ensure patient data privacy is protected, and
- Ability to deliver on the above two objectives by interfacing with users and supporting IT systems

Some objectives that are out of scope for this project are described as follows:

- Meeting exact industry standards for every element of the system: this is due to lack of time and lower priority of the UI compared to the AI model's correctness
- Verifying the correctness of data sets used in training and testing the AI model/ system: we are not re-verifying them as they were verified by their respective authors
- Verifying the usability of the system: usability testing is currently of a lower priority, and this can be re-prioritized should we complete our higher priority verification and validation ahead of schedule
- Verifying the external libraries used for the AI model and the system frontend and backend: e.g. graphical user interface libraries, database interfacing/communication libraries; we will assume that these external libraries have been verified by their respective implementation teams

2.3 Relevant Documentation

The relevant documentation that will be referenced in this document (i.e. the other project documents) are listed below. Explanations are also given of why they are relevant and how they relate to our VnV efforts as described in this document:

- SRS: This document contains the descriptions, rationales and fit criteria on all the requirements (functional and non-functional) that will be verified and tested as outlined in this document. In essence, it serves as the basis for the system testing plans.
- MG: This document outlines the various modules that the software is composed of, giving a detailed overview of the software architecture. The MG details the module decomposition and uses hierarchy & traceability of requirements to their implementing module(s). The MG provides the basis for guiding the system's functional and nonfunctional testing plans in grouping the tests to each module and tracing them to the requirements they are intended to meet.
- MIS: This document describes the more specific design details of the various module interfaces that the unit testing shall use and cover. The MIS influences the unit testing scope and affects how functional and nonfunctional unit tests will be written to test specific functions and properties of the software.

3 Plan

This section will outline the verification and validation roles of each team member and the team supervisor in the context of the VnV plan. The verification plans for the SRS, the design (i.e. MIS, MG and System Design documents), the VnV plan itself, and the implementation are also outlined in detail here. The automated testing and verification tools the VnV plan will use are also listed. The software validation plan will also be discussed.

The tentative plans for verifying the SRS, the design (i.e. MIS, MG and System Design documents), the VnV plan itself and the implementation will be executed by the VnV team members in the order they are presented, as the relevant work for each is done (e.g. demo, design documents, software and unit tests, etc.). The verification of the SRS document is an ongoing process, while the design documents will be verified as the source code is written. The VnV plan and implementation will be verified next in an ongoing process leading up to the proof-of-concept and final demonstrations. The software validation will also be taking place concurrently, alongside other VnV plans leading up to the proof-of-concept and final demonstrations.

3.1 Verification and Validation Team

In this section, Table 2 details the members of the Validation and Verification team responsible for performing the tasks outlined and described in this VnV Plan document. For each member, their role(s) in the project's verification are summarized with key details noting their responsibilities.

VnV Team Member Name	**Summary of Role(s)**
Dr. Mehdi Moradi (Supervisor)	Advisor, primary reviewer of documentation, a contributor to validation of documentation and code, provide suggestions and corrections of the software and its functionality
Other Design Teams	Peer reviewers, raise issues and provide feedback/ suggestions for documentation improvements
Allison Cook	Review other team members' work to maintain high standards, provide suggestions for improvements, maintain feedback checklists for each work item
Ibrahim Issa	Review other team members' work to maintain high standards, provide suggestions for improvements and maintain feedback checklists for each work item
Mohaansh Pranjal	Review other team members' work to maintain high standards, provide suggestions for improvements and maintain feedback checklists for each work item
Nathaniel Hu	Review other team members' work to maintain high standards, provide suggestions for improvements and maintain feedback checklists for each work item
Tushar Aggarwal	Review other team members' work to maintain high standards, provide suggestions for improvements and maintain feedback checklists for each work item

Table 2: Verification and Validation Team

3.2 SRS Verification Plan

The following points outline the approaches we intend to use (and have already begun using) for SRS verification. This includes both formal and

ad-hoc feedback from our various reviewers:

- Formal reviews, creating new checklists using existing checklists and feedback from grading to update documentation as a group, reviewing our requirements through use cases and other personas.
- Formal and ad-hoc review meeting(s) with the supervisor, where they go over the document and point out mistakes or suggestions for improvements.
- Ad-hoc peer reviews by other teams in the course shall provide suggestions and corrections.
- For the SRS, the following needs to be checked:
  - The fit criterion for all requirements is adequate, feasible and verifiable.
  - The functional requirements are traceable to all the use cases for the application.

The following is an initial SRS verification plan checklist that will be updated with items being added/removed as reviews to verify the SRS are completed over time:

- ☐ Does each functional requirement contain a detailed and accurate description, rationale and fit criteria?
- ☐ Is each requirement (functional and non-functional) relevant and necessary?
- ☐ Do the functional requirements capture the intended software functionality?
- ☐ Are all functional requirements traceable to at least one use case?
- ☐ Are all fit criteria for requirements unambiguous and verifiable?
- ☐ Is the project plan timeline feasible given the time constraints?
- ☐ Have all issues opened by the reviewers been closed?

3.3 Design Verification Plan

The following points outline the plan for verifying the design of the software, as is captured in the various design documents (i.e. the MG, MIS and System Design documents):

- Shall conduct a review meeting with the supervisor after the design documents have been completed.
- Peer review from classmates shall provide critical suggestions.
- Shall verify the conformity of the code with SOLID design principles.
- Conduct ad-hoc review(s) with other teams.
- Conduct a formal review with teammates sometime after the initial creation, which allows us to reflect on the document prior to review, using checklists and comparison to the SRS (after verification)

The following is an initial design document verification plan checklist that will be updated with items being added/removed as reviews to verify the design documents are completed over time:

- ☐ Are all requirements (functional and non-functional) traceable to at least one implementing module in the MG?
- ☐ Have all issues opened by the reviewers been closed?
- ☐ Do all modules and other components conform to the SOLID design principles?
- ☐ Do all the modules have an unambiguous task that they are created for, and input and output is well defined?

3.4 Verification and Validation Plan Verification Plan

The following points outline the plan for verifying the VnV plan itself:

- Review(s) with the supervisor will be done to verify testing and verification plans.
- Peer reviews from other teams shall provide critical feedback on the VnV plan on areas of improvement.
- Ad-hoc reviews with teammates shall also provide critical feedback on the VnV plan for areas of improvement.

The following is an initial VnV plan verification plan checklist that will be updated with items being added/removed as reviews to verify the VnV plan are completed over time:

- ☐ Does the VnV Plan verify all requirements (functional and non-functional) are met?
- ☐ Have all issues opened by the reviewers been closed?
- ☐ Are all the stakeholders/supervisor included in the review processes described?
- ☐ Are all the aspects of the software product being tested by the testing tools?
- ☐ Do the system and unit tests cover all requirements?

3.5 Implementation Verification Plan

The following points outline the plan for verifying the implementation, using both static and dynamic techniques:

- Walkthroughs of key components of the code with the supervisor (i.e. AI model, user authentication, display of findings, UI design).
- Walkthroughs and inspections with other teammates who worked on different sections (static).
- Running static analyzers (i.e. linters (e.g., Flake8), security scans, etc.) to help discover bugs and potential problems in the code, and to make it more readable.
- Running the system tests (functional and non-functional) described in this VnV plan document to verify the implementation meets requirements (functional and non-functional).
- Running the unit tests (to be implemented and described in this VnV plan document) to verify the implementation matches designs specified in MG, MIS, and System Design documents (will be done automatically using GitHub Actions for each branch/pull request).
- Major code commits or new features shall be reviewed by at least one other member of the team before merging to prevent bugs.

3.6 Automated Testing and Verification Tools

The following are the automated testing and verification tools to be used during the validation and verification process for the software being tested in this VnV plan:

- Linters: Prettier, ESLint, Flake8, Black
- Unit testing: Jest, Pytest
- Code coverage: Istanbul, Coverage.py
- Continuous Integration: GitHub Actions

It should be noted that all of these tools listed are also mentioned in the Development Plan document.

Our plans for summarizing the code coverage metrics of our unit tests mainly revolve around our two code coverage tools, Istanbul and Coverage.py. Instanbul will check the statement, branch, function and line coverage of our unit tests and will present the percentage coverage results for each aforementioned coverage metric. Coverage.py will check the line (statement) and branch coverage and present the percentage coverage results for those coverage metrics. It should also be noted that these details are subject to change as the implementation is completed and unit tests are written and run to validate the code.

3.7 Software Validation Plan

Our plan for validating the software revolves around using reserved subsets of the training data used to train the AI model driving the analysis of chest X-rays. The specific datasets to be used are the MIMIC-III Clinical Database and the Chest ImaGenome Dataset. A larger subset from each of these datasets will be used for training the AI model. A smaller subset from each dataset will be used during the PoC, Revision 0 and Revision 1 demos. These (more) formal demos will be done to validate the requirements themselves, the AI model and the software's ability to deliver on its objectives outlined in this VnV plan and meet all requirements, functional and nonfunctional. They will allow us to get critical feedback from the professor and TA on how the software could be improved upon.

Informal demos and review sessions using the smaller subsets will also be

conducted to validate our software for our supervisor. These demos and review sessions will be conducted around the more formal demos mentioned above. This will allow the supervisor to provide critical feedback so we can improve the AI model and any other part of the software. During these review sessions, we will also check with the supervisor (our main stakeholder representative) to verify our SRS document captures the right requirements using task-based inspections. This will ensure we are defining the right things in the requirements to correctly guide the software implementation. We will also be conducting regular user testing throughout the software implementation process to validate our software little by little.

4 System Test Description

This section outlines and describes the system test cases that will be used for validating the implementation, with regard to the functional and nonfunctional requirements previously outlined in the SRS. For each test case, the control (manual or automatic), initial state, input, output, derivation, and a short procedure of how the test will be performed are given.

4.1 Tests for Functional Requirements

In this subsection, the system test cases are categorized based on different areas of functionality to ensure a thorough verification process. The following subsets of test cases are designed to validate the handling of input data, display functionality, disease classification, data access, and user authentication and authorization. These subsets cover the key functionalities the functional requirements have described are necessary for the success of this software. For each test case, references to the relevant functional requirements from the SRS that are covered by it are included in the test case derivation part.

4.1.1 Handling Input Data

- test-id1

  Title: DICOM Input Acceptance

  Control: Manual

Initial State: The system is in a stable state with all components initialized and ready to receive input.

Input: A sample chest X-ray image in DICOM format.

Output: The DICOM input is accepted and converted into a JPEG successfully, with no error messages or system anomalies.

Test Case Derivation: The expected output is justified based on FR.1 and FR.7 in section 2.4.1 in the SRS document.

How the test will be performed:

1. Manually upload a sample chest X-ray image in DICOM format through the user interface.
2. Observe the system's response to the input, checking for any error messages or unexpected behaviour.
3. Verify that the DICOM input is successfully accepted and converted into a JPEG image.

- test-id2

Title: Non-DICOM Input Rejection

Control: Manual

Initial State: The system is in a stable state with all components initialized and ready to receive input.

Input: A sample chest X-ray image in a format other than DICOM.

Output: The expected result is a rejection of the non-DICOM input, accompanied by an appropriate error message.

Test Case Derivation: The expected output is justified based on FR.1 in section 2.4.1 in the SRS document.

How the test will be performed:

1. Manually attempt to upload a sample chest X-ray image in a format other than DICOM through the user interface.
2. Observe the system's response to the input, checking for any error messages or unexpected behaviour.

3. Verify that the non-DICOM input is rejected and flagged for further action.

- test-id3

Title: DICOM conversion to JPEG

Control: Manual

Initial State: The system is in a stable state with all components initialized and ready to receive input.

Input: A sample chest X-ray image DICOM format.

Output: The expected result is an output of the same x-ray in JPEG format.

Test Case Derivation: The expected output is justified based on FR.8 in section 2.4.1 in the SRS document.

How the test will be performed:

1. This is a test for a part of the process involved in processing the DICOM file in the ML model.
2. A chest x-ray sample DICOM file will be submitted to the system
3. The system should automatically convert it to JPEG format
4. Manually confirm that the conversion was successful and both file formats show the same chest x-ray.

4.1.2 Display

- test-id4

Title: Image Display Correctness

Control: Manual

Initial State: The system is in a stable state with all components initialized, and relevant data has been processed.

Input: Processed chest X-ray images with known medical conditions

Output: The expected result is the correct display of chest X-ray

images on the user interface, and it reflects the processed medical conditions accurately.

Test Case Derivation: The expected output is justified based on FR.2 and FR.3 in section 2.4.1 in the SRS document.

How the test will be performed:

1. Manually access the user interface.
2. Select the processed chest X-ray images with known medical conditions for display.
3. Observe the displayed images, checking for visual correctness in terms of identified medical conditions.
4. Verify that the displayed images correspond to the expected outcomes based on the processed data.

- test-id5

Title: Report and Terms Correctness

Control: Manual

Initial State: The system is in a stable state with all components initialized, and relevant input data has been processed.

Input: Processed chest X-ray images with known medical conditions

Output: The expected result is the correct display of medical reports and associated terms on the user interface and accurately reflects the processed information.

Test Case Derivation: The expected output is justified based on FR.4 and FR.5 in section 2.4.1 in the SRS document.

How the test will be performed:

1. Manually access the user interface.
2. Select the processed chest X-ray images with known medical conditions to generate an associated list of findings.
3. Observe the displayed reports/terms, checking for correctness in terms of identified medical conditions and terminology.
4. Verify that the displayed list of findings corresponds to the

expected outcomes based on the processed data.

4.1.3 Classification of Diseases

- test-id6

Title: Disease Classification Correctness Performance

Control: Automatic (offline experiment)

Initial State: The system used in the experiment is only the AI model so no state is needed

Input: DICOM chest X-ray images, approximately 20% of the dataset DATASET_NAME will be reserved and used to run this test/experiment, with known medical conditions representing various diseases.

Output: The expected result is an AUC for the ROC curves for the classification of diseases based on the expected area under the ROC curves as given in Table 6 in the SRS.

Test Case Derivation: The expected output is justified based on FR.6 in section 2.4.1 in the SRS document.

How the test/experiment will be performed:

1. Submit all the test data of the DICOM chest X-ray images with known medical conditions.
2. Collect all findings and determine the area under the ROC curve for the respective findings, based on how much the responses match the expected result of the test data.
3. Verify the system's classification results, and the AUC for the ROC curve, and determine whether this parameter passes the required threshold for the different diseases.

4.1.4 Access of Database

- test-id7

Title: Database Search Functionality

Control: Manual

Initial State: The system is in a stable state with all components initialized, and relevant patient records have been processed and stored in the database.

Input: User-initiated search queries for patient records based on specified criteria such as patient ID (unique), name, date of X-ray, or medical condition(s).

Output: The expected result is the accurate retrieval of patient records matching the specified search criteria, displayed on the user interface.

Test Case Derivation: The expected output is justified based on FR.6, FR.9, and FR.11 in section 2.4.1 in the SRS document.

How the test will be performed:

1. Manually initiate search queries through the user interface, specifying different criteria such as patient ID, date, or medical conditions.
2. Observe the system's response, checking for the accuracy and completeness of the retrieved patient records.
3. Verify that the displayed patient records correspond to the specified search criteria.

4.1.5 Authorization

- test-id8

Title: Correct User Login

Control: Manual

Initial State: The system is in a stable state with all components initialized, and user accounts set up.

Input: User-initiated login attempts with valid credentials.

Output: The expected result is the successful login of authorized

users, granting them access to the system.

Test Case Derivation: The expected output is justified based on FR.10 in section 2.4.1 in the SRS document.

How the test will be performed:

1. Manually attempt to log in with valid user credentials through the user interface.
2. Observe the system's response, checking for successful user authentication and system access granted to the user.

- test-id9

Title: Incorrect User Login

Control: Manual

Initial State: The system is in a stable state with all components initialized, and user accounts have been set up.

Input: User-initiated login attempts with invalid credentials.

Output: The expected result is the rejection of login attempts with invalid credentials, accompanied by the appropriate error message.

Test Case Derivation: The expected output is justified based on FR.10 in section 2.4.1 in the SRS document.

How the test will be performed:

1. Manually attempt to log in with invalid user credentials through the user interface.
2. Observe the system's response, checking for successful rejection of the login attempt and display of the error message.

- test-id10

Title: Creation of New Authorized User

Control: Manual

Initial State: The system is in a stable state with all components

initialized.

Input: User-initiated creation of a new authorized user account through the user interface.

Output: The expected result is the successful creation of a new authorized user account recorded and saved in the system.

Test Case Derivation: The expected value is justified based on FR.10, in section 2.4.1 in the SRS document.

How the test will be performed:

1. Manually initiate the creation of a new user account through the user interface.
2. Observe the system's response, checking for successful account creation.
3. Verify that the system records and saves the created user.

4.2 Tests for Nonfunctional Requirements

In this subsection, the system test cases are categorized into two major areas to cover the two key properties mandated by the nonfunctional requirements. The subsets of test cases are designed to validate the security and performance of the software. The non-functional requirements for accuracy reference the appropriate functional tests from the above section. Tests related to usability are currently of a lower priority are will be generically defined. It should also be noted that static tests, reviews, inspections, and walkthroughs will not follow the format for the tests below.

4.2.1 Security

- test-id11

Title: Data Encryption

Control: Manual

Initial State: The system is in a stable state with all components initialized, and patient records have been processed and stored in the database.

Input: N/A (Unauthorized access to database granted not through the system)

Output: The expected result is encrypted data (not plain text data)

Test Case Derivation: The expected output is justified based on NF.IR0 and NF.PR1 in section 3.6.2 in the SRS document.

How the test will be performed:

1. Manually attempt to log in with valid user credentials through the user interface.
2. Observe the system's response, checking for successful user authentication and access to the system.

See Authorization in the functional requirements for NF.AR0, NF.AR1, NF.PR0, NF.PR1 as they all relate to limiting access to records for authorized individuals.

4.2.2 Performance

- test-id12

  Title: System Response Time

  Control: Automated

  Initial State: The system is in a normal operational state with an average user load.

  Input: Initiate automated requests for all the various system functionalities, including image processing and database searches.

  Output: The expected output is that the system provides responses within the defined acceptable time thresholds for each functionality as specified in section 3.3.1 of the SRS.

  Test Case Derivation: The expected output is justified based on NF.SLR0 in section 3.3.1 in the SRS document.

  How the test will be performed:

  1. Automatically simulate user interactions through testing scripts covering various functionalities.
  2. Monitor and record the system's response times for each of the simulated interactions.
  3. Verify that the recorded response times align with the defined acceptable thresholds.

- test-id13

  Title: Accuracy of the ML model

  Reference: Refer to test-id6 in section 4.1.3 above, included in the tests for the Functional Requirements which covers NF.PAR0 in section 3.3.3 in the SRS document.

- test-id14

Title: Patient Data Collection

Control: Manual

Initial State: The system is in a normal operational state with all components initialized and ready to receive input.

Input: A New patient is added to the system.

Output: The expected output is that the system will retain the patient name, past diagnoses, assigned physician, past visit dates and other relevant data, while not retaining any unnecessary data.

Test Case Derivation: The expected output is justified based on NF.SCR0 in section 3.3.2 in the SRS document.

How the test will be performed:

1. Manually upload new patient data for a patient file/X-ray?.
2. Record the data held by the system.
3. Verify that the recorded data kept by the system is minimal.

- test-id15

Title: Capacity

Control: Manual

Initial State: The system is in a normal operational state with all components initialized and ready to receive input.

Input: Three sample chest X-ray images in DICOM format divided between three computers using the system.

Output: The expected result is the correct display of chest X-ray images on the user interface, reflecting all the processed medical conditions accurately for each individual upload.

Test Case Derivation: The expected output is justified based on NF.CR0 in section 3.3.5 in the SRS document.

How the test will be performed:

1. Each user will manually upload a sample chest X-ray image

in DICOM format through the user interface.
2. Monitor the system's response to the input, checking for any error messages or unexpected behaviour.
3. Verify that the displayed results correspond to the expected outcomes for each independent upload.

4.2.3 Usability

Usability testing, **test-id16** as mentioned in our objectives, is of a lower priority but will be generically defined in this section should we complete our higher priority testing. This will completed by surveying a group of individuals who have never used the system before and would be regular users of the system such as doctors and nurses. This group will be given access to the system and asked to complete a survey, which at this point in time, due to low priority, has not been designed, after they have used the system to complete given tasks, such as uploading X-rays or viewing data, for a determined amount of time.

This test will satisfy the following nonfunctional requirements and the expected output is based on their defined fit criteria. NF.A0 in section 3.1, NF.EUR0, NF.LR0 and NF.UPR0 in section 3.2, and NF.C0 in section 3.7 in the SRS document.

4.2.4 Operation and Support

The nonfunctional requirements NF.PR0, the system is accessible through a web application, and NF.SR0, the system is self-supporting, is both required in the set-up of all tests that use the complete system and is therefore tested in almost all listed test cases. Similarly with NF.RIAS0, the system will require access to the host for image upload, is necessary in all cases of image input into the system and therefore tested in all these instances.

4.3 Not Tested Nonfunctional Requirements

The following listed nonfunctional requirements will not be tested as they fall under the out-of-scope objectives or are not feasible to test within the timeline of this project:

- NF.SR0

- NF.PIR0
- NF.RFTR0
- NF.SER0
- NF.EPE0
- NF.MR0
- NF.AR0
- Compliance Requirements (NF.LR0, NF.SCR0)

4.4 Traceability Between Test Cases and Requirements

Requirement	Test
FR.1	T.1, T.2
FR.2	T.4
FR.3	T.4
FR.4	T.5
FR.5	T.5
FR.6	T.7
FR.7	T.1
FR.8	T.3
FR.9	T.8, T.9, T.10
FR.10	T.7
FR.11	T.7

Continuation of Table	
Requirement	**Test**
NF.A0	T.16
NF.EUR0	T.16
NF.LR0	T.16
NF.UPR0	T.16
NF.SLR0	T.12
NF.SCR0	T.14
NF.PAR0	T.13/T.6
NF.CR0	T.15
NF.RIAS0	T.1, T.2, T.3, T.4, T.5
NF.PR0	All (minus T.6)
NF.SR0	All (minus T.6)
NF.AR0	T.8, T.9, T.10
NF.AR1	T.8, T.9, T.10
NF.IR0	T.11
NF.PR0	T.8, T.9, T.10
NF.PR1	T.11, T.8, T.9, T.10
NF.C0	T.16
End of Table	

Table 3: Requirements Traceability

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 lay out 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, your 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 the 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 the 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 the test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How the 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

6 Appendix

In this section, all of the additional information to complement this VnV Plan are placed here.

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. See the following Table 4 for reference:

Symbolic Constant	Definition or Description
DATASET_NAME	The name of the chest x-ray imaging dataset to be used during the development and testing of the software's AI model; i.e. Chest ImaGenome Dataset or MIMIC-III Clinical Database

Table 4: Symbolic Constants

6.2 Usability Survey Questions

As mentioned above a usability survey will only be constructed if the project is progressing quicker than expected and we can move on to lower-priority testing.

Appendix — Reflection

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

1. 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 etc. You should look to identify at least one item for each team member.

For the verification and validation of the project, the skills that are required and play an important role are the following:

1. SOLID Design Principles: The code needs to be verified such that the system is designed keeping in mind these design principles to make an efficient and easy-to-maintain system.
2. Testing Knowledge: Knowledge related to the usage of testing approaches, like unit testing and regression testing that play a big role in debugging software, is crucial.
3. Knowledge of Testing tools: Knowledge of using testing tools like pytest and linter is also very helpful.
4. Knowledge of AI/ML Development: Knowledge related to the development and training of AI/ML models, as it will be the driving force behind this software's main functionality.

Each team member will aim the acquire the following knowledge and skills needed to successfully complete the verification and validation of the project. The specific knowledge and skills each team member will aim to acquire are listed below:

1. Allison Cook: Shall develop skills related to verifying if code is based on SOLID principles
2. Ibrahim Issa: Learn about different types of software testing such as unit testing

3. Mohaansh Pranjal: Shall learn about implementing and verifying code according to SOLID design principles to make code efficient, readable and easy to maintain.
4. Nathaniel Hu: Gain skills in using testing tools like pytest, gain further knowledge in AI/ML development and training (needed for software validation)
5. Tushar Aggarwal: Gain skills in AI/ML testing

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

For each of the knowledge areas and skills identified above, at least two approaches to acquiring the knowledge or mastering the skill are shown below:

1. SOLID Design Principles:
   (a) online coding courses (e.g. freeCodeCamp, LinkedIn Learning)
   (b) personal projects
2. Testing Knowledge:
   (a) External resources (websites) to learn about different testing approaches
   (b) Implementation of test cases for simple standalone programs or structured software systems
3. Knowledge of Testing Tools:
   (a) Running tools like pytest for test cases to observe parameters such as runtime
   (b) learning about and using linters like flake8 to write code in a standardized format
4. Knowledge of AI/ML:

Of the identified approaches, the ones each team member will pursue are shown below, with the justifications for why those choices were made:

1. Allison Cook: Learning about design principles by writing neat code and making personal projects as implementation is an efficient way to learn.
2. Ibrahim Issa: Using websites and online resources to learn about diferrent types of testing approaches as different situations require different types of testing.
3. Mohaansh Pranjal: Shall use online resources (e.g. freeCodeCamp, LinkedIn Learning) to learn about the design principles as they are also useful in overall implementation and an important part of writing readable code.
4. Nathaniel Hu: Running pytest on test cases for programs which allows analysis of runtime and efficiency.
5. Tushar Aggarwal: Use research papers in AI/ML to select parameters to test accuracy of the model