System Verification and Validation Plan for Chest Scan

Team 16, Ace
Harrison Chiu
Hamza Issa
Ahmad Hamadi
Jared Paul
Gurnoor Bal

November 4, 2024

Revision History

Date	Author	Notes
4 November 2024	Harrison	Section 4
4 November 2024	Hamza	Section 5
4 November 2024	Gurnoor	Section 1-3
4 November 2024	Jared	Section 3, 5
4 November 2024	Ahmad	Section 5

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

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

Symbol, Abbreviation or Acronym	Definiton or Description
ML	Machine Learning: A branch of artificial intelligence that involves the use of algorithms to allow computers to learn from and make predictions based on data. This is a core technology used in the project for analyzing chest X-rays.
DL	Deep Learning: A subset of machine learning involving neural networks with many layers, used to analyze various types of data, including images.
DICOM	Digital Imaging and Communications in Medicine: A standard for transmitting, storing, and sharing medical imaging information. It is used to manage medical images in the proposed solution.
CNN	Convolutional Neural Network: A type of deep learning model specifically designed for processing structured grid data like images, used in the project for chest X-ray analysis.
EHR	Electronic Health Record: A digital version of a patient's paper chart, used for storing patient information and history that may be integrated with the proposed solution.
API	Application Programming Interface: A set of rules and protocols for building and interacting with software applications, enabling the integration of the proposed solution with other systems.
MC	Mandated Constraints: Various constraints placed on the project's proposed solution that must be adhered to throughout the development process.
FR	Functional Requirement: A requirement that specifies what functionality the project's proposed solution must provide to meet user needs.
NFR	Nonfunctional Requirement: A requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors (e.g., performance, usability).
BUC	Business Use Case: A scenario that describes how the proposed solution can be used within a business context to achieve specific goals.
PUC	Product Use Case: A scenario that details how an individual user will interact with the proposed solution to achieve specific tasks.
MVP	Minimum Viable Product: A version of the proposed solution that includes only the essential features required to meet the core needs of the users and stakeholders.

MG	Module Guide
MIS	Module Interface Specification
PoC	Proof of Concept
SRS	Software Requirements Specification
FRTC	Functional Requirements Test Case
NFRTC	Nonfunctional Requirements Test Case
VnV	Verification and Validation

This document outlines the verification and validation (VnV) plan for the project's proposed solution. It begins with a summary of the software's general functions, the objectives of the VnV plan (including in-scope and out-of-scope elements), and references relevant documentation.

It lists the VnV team members and describes the process for verifying the SRS, design, VnV plan, and implementation, including the automated testing tools used. The VnV roadmap starts with verifying the SRS, followed by the design, VnV plan, and implementation. After these steps, the software will undergo validation, with milestones achieved through automated tools and guidance from the project supervisor. It also shows the tests done (system and unit tests) to each major module to verify its functionality and output.

2 General Information

2.1 Summary

The software being developed is the "Chest Scan" AI diagnostic tool, designed to support radiologists by automating chest X-ray analysis and generating diagnostic reports. This tool will leverage convolutional neural networks (CNNs) to detect and classify lung and heart conditions from X-ray images, thus enhancing diagnostic efficiency and accuracy. The system aims to reduce radiologist workload, improve diagnostic precision, and ultimately elevate patient care. Key functions include image processing, disease identification, and report generation accessible via a user-friendly web interface.

2.2 Objectives

The V&V Plan for the "Chest Scan" diagnostic tool is designed with three core objectives that ensure the tool achieves its intended clinical accuracy, secures sensitive patient information, and provides a practical, usable interface for healthcare professionals. Each objective addresses a critical aspect of the tool's functionality and clinical readiness, from the AI model's diagnostic performance to the protection of data and ease of use. These objectives guide the verification and validation steps necessary to support confidence in the system's effectiveness within healthcare environments.

1. AI Model Accuracy and Diagnostic Reliability
    - Verify that the model can accurately analyze chest X-ray images, identifying targeted conditions such as pneumonia and cardiomegaly with sufficient reliability.
    - Minimize false negatives and positives to ensure diagnostic reliability, helping radiologists make better-informed decisions.
    - Validate that the model produces clinically useful outputs, including structured findings and location-based annotations on X-ray images, to streamline interpretation by radiologists.
2. Data Security and Access Control

- Confirm that patient data is stored and accessed securely, meeting healthcare industry data protection requirements.
- Ensure that the system supports secure authentication, granting access only to authorized users while preventing unauthorized access.
- Validate compliance with privacy standards (such as HIPAA), focusing on data encryption, secure storage practices, and access restrictions for sensitive information.

3. Usability and System Accessibility
    - Ensure that the software interface is user-friendly, enabling radiologists to log in, access diagnostic results, and interact with generated reports intuitively.
    - Verify that the web-based interface is efficient, visually accessible, and provides an organized layout, which supports radiologists in reviewing diagnostic information quickly and accurately.
    - Check that the system integrates smoothly with medical imaging infrastructure (e.g., PACS) and supports data transfer between adjacent systems in clinical environments.

2.2.1 Out of Scope for the V&V Plan

Given the project's timeframe and resource limitations, the following aspects will not be included in this V&V Plan. While they may be desirable in a fully deployed product, these areas are either lower priority compared to the core objectives or outside the feasible scope for this development phase. The team has identified these exclusions to maintain focus on critical system functionality, while acknowledging areas that may require future exploration or enhancement.

- Industry-Standard UI/UX Optimization: Advanced user interface refinement is deprioritized relative to AI accuracy and data security objectives.
- External Library Verification: This plan will not validate the reliability of third-party libraries used, assuming their integrity based on their widespread clinical and research use.
- Full Scalability Testing: This MVP will support single-user diagnostic testing, and rigorous multi-user load testing will be out of scope.

2.3 Challenge Level and Extras

This project is classified at an advanced challenge level due to its dual focus on high-stakes medical diagnostics and complex AI model development, both of which require rigorous standards for accuracy, security, and usability. The challenge lies not only in developing a convolutional neural network (CNN) capable of accurately analyzing chest X-rays but also in ensuring that the software integrates seamlessly within clinical workflows, safeguarding

patient privacy, and delivering reliable results under various conditions. Achieving these goals requires the team to balance research and development in AI with a strong emphasis on usability, data security, and healthcare compliance.

The complexity also arises from the project's need to synthesize academic research with practical implementation, making it a hybrid between a research-oriented and a product-driven development effort. The system must be robust enough to serve as a proof-of-concept for healthcare settings, meeting accuracy benchmarks while maintaining a secure and accessible interface for radiologists.

2.3.1 Extras

To enhance the tool's functionality and make it more user-friendly for clinical adoption, the following extras are planned if resources allow:

- Usability Testing with Radiologists: To confirm that the user interface is intuitive, simple usability tests with practicing radiologists will be conducted. This will provide valuable feedback on real-world application and ease of use, ensuring that the tool aligns with the expectations and needs of healthcare professionals.
- Research Paper and Documentation: Comprehensive documentation will accompany the MVP, including a research paper summarizing the project's development, methodologies, and findings. This paper will be valuable for other researchers, detailing insights into the model's design, training process, and performance metrics. Additionally, this research will be shared with the academic and medical communities, supporting further exploration of AI applications in radiology.

These extras, though secondary to the primary objectives, will enhance the tool's viability and user experience while contributing to the broader knowledge base on AI-driven diagnostics in healthcare.

2.4 Relevant Documentation

A set of comprehensive documents will guide and support the verification and validation of the "Chest Scan" diagnostic tool, detailing its requirements, modular structure, and interface specifications. These documents serve as essential references for understanding the system's architecture and ensuring that all aspects of the tool are accurately validated and verified.

- **Software Requirements Specification (SRS)** The SRS outlines both functional and nonfunctional requirements for the tool, setting the foundation for development and testing. It details core features, such as image analysis, disease classification, report generation, and security measures, along with performance criteria and usability goals. The SRS will also include traceability matrices to ensure that all requirements are covered within the testing framework, aligning with the system's core objectives.
- **Problem Statement Document** This document defines the problem the diagnostic tool seeks to address, including the high volume of chest X-rays and the need for efficient diagnostic support for radiologists. It also describes the project's primary

goals, stakeholders, and scope, which will help ensure that the V&V process aligns with the tool's intended purpose and expected clinical impact.

- **Module Guide (MG)** The MG will provide a structured breakdown of the tool's software modules, detailing the hierarchy and interdependencies within the system. It will follow principles of modular decomposition to define each module's purpose, services, and role within the larger system. By organizing modules based on anticipated changes, this document supports future scalability and ease of maintenance. The MG will also include traceability to ensure that each module meets its corresponding requirements from the SRS, enhancing the effectiveness of the validation process.
- **Module Interface Specification (MIS)** The MIS will define the precise interfaces for each module outlined in the MG, including exported constants, access programs, and the semantics of each module's functionality. It will also specify assumptions, environment variables, and potential state transitions where relevant. This document is critical for the V&V process, as it ensures that each module can interact correctly with others, conforming to the system's overall design and requirements.

Together, these documents will facilitate a comprehensive understanding of the tool's architecture and requirements, aiding in the development of thorough and precise V&V processes to achieve a reliable and secure diagnostic solution.

3 Plan

In this section, we will lay out the roadmap for ensuring each part of the project meets the intended standards through verification and validation. Responsibilities are assigned to team members for each task within the V&V process. This plan will address how we will confirm that the project's requirements, design documents, V&V plan, and final implementation align with the specified criteria. Tools and automated testing solutions that support our verification efforts will also be outlined. Additionally, a detailed process for validating the software's functionality and performance will be included. Our verification tasks will be presented in a structured sequence, moving from requirements analysis to design inspection, implementation testing, and review of the V&V process itself. Requirements verification will continue throughout development, with design checks occurring as the code is produced. Validation of the implementation will be an ongoing effort, eventually leading to a proof-of-concept phase and final project demonstrations.

3.1 Verification and Validation Team

VnV Team Member Name	Summary of Role(s)
Amirhossein Sabour	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
Gurnoor Bal	Review other team members' work to maintain high standards, provide suggestions for improvements, maintain feedback checklists for each work item.
Jared Paul	Review other team members' work to maintain high standards, provide suggestions for improvements, maintain feedback checklists for each work item.
Ahmad Hamadi	Review other team members' work to maintain high standards, provide suggestions for improvements, maintain feedback checklists for each work item.
Hamza Issa	Review other team members' work to maintain high standards, provide suggestions for improvements, maintain feedback checklists for each work item.
Harrison Chiu	Review other team members' work to maintain high standards, provide suggestions for improvements, maintain feedback checklists for each work item.

3.2 SRS Verification Plan

The SRS Verification Plan for the "Chest Scan" diagnostic tool ensures that all specified requirements are complete, accurate, and verifiable. This section outlines the approach to confirm that the SRS document meets quality standards, providing a solid foundation for subsequent design, implementation, and testing phases.

3.2.1 Objectives

The objectives of this SRS Verification Plan are to:

1. **Ensure Requirement Completeness:** Confirm that all essential functionalities, such as X-ray image processing, condition classification, diagnostic report generation, and user interface features, are fully specified in the SRS.
2. **Validate Requirement Clarity:** Verify that each requirement is unambiguous, consistent, and comprehensible for both the development and testing teams, reducing the risk of misinterpretation.
3. **Check Requirement Feasibility:** Assess each requirement to confirm that it is achievable within the scope and technical constraints of the project.
4. **Verify Traceability:** Ensure that each requirement has a unique identifier and can

be traced through design and testing phases, facilitating streamlined verification and validation.

3.2.2 Verification Checklist

To comprehensively verify the SRS, the following key elements will be checked:

1. **Functional Requirements**
    - Clear definitions for each core function, such as:
    - Processing and analysis of chest X-ray images.
    - Identification and classification of specific conditions (e.g., pneumonia, cardiomegaly).
    - Structured diagnostic report generation with condition summaries and annotations.
    - Specifications for data input/output formats and requirements for public dataset integration (e.g., CheXpert, MIMIC).
2. **Nonfunctional Requirements**
    - Performance: Defined thresholds for processing speed, response times, and accuracy of diagnoses.
    - Security: Compliance requirements for patient data protection, including secure storage, access controls, and adherence to privacy standards (e.g., HIPAA).
    - Usability: User interface design criteria, accessibility considerations, and ease-of-use requirements for radiologists and medical technologists.
    - Reliability: Expected system uptime and fault-tolerance levels to ensure system stability in clinical environments.
3. **Constraints**
    - Verification that all system constraints, such as hardware compatibility, software dependencies (e.g., TensorFlow, PyTorch), and budget limitations, are clearly specified.
    - Identification of external systems the tool must integrate with, like PACS or other hospital information systems.
4. **Traceability and Identification**
    - Each requirement in the SRS has a unique identifier for easy tracking.
    - Requirements are linked to corresponding modules (as per the Module Guide) and future test cases, supporting consistency across the V&V process.
5. **Glossary and Terminology**

- Verification that all relevant medical and technical terminology, acronyms, and abbreviations are defined for clarity, preventing misinterpretation.

6. **Assumptions and Dependencies**
    - Clarity on assumptions (e.g., availability of high-quality datasets, necessary hardware resources).
    - Dependencies on external datasets, third-party libraries, or existing healthcare infrastructure are explicitly noted.

3.2.3 Methods

The following methods will be used to verify the SRS:

- **Requirement Review Meetings:** Conduct collaborative review sessions with stakeholders, including radiologists, AI researchers, and development team members, to verify that all specified requirements align with user needs and project objectives.
- **Traceability Matrix Construction:** Create a traceability matrix linking each requirement to corresponding test cases, ensuring that all requirements can be tested effectively in later stages.
- **Cross-Referencing with MG and MIS:** Check the consistency of requirements with the Module Guide (MG) and Module Interface Specification (MIS) to ensure that each SRS item is appropriately decomposed into modules and interfaces.

3.2.4 Verification Criteria

The SRS will be considered verified if:

- All functional and nonfunctional requirements are clearly stated, necessary for the system's objectives, and achievable.
- Each requirement is supported by a corresponding entry in the traceability matrix.
- Stakeholder review feedback confirms that the requirements are complete and aligned with real-world diagnostic and security needs.

By following this SRS Verification Plan, we aim to establish a robust foundation for the "Chest Scan" tool, ensuring that every feature aligns with clinical and operational standards while providing clear, testable specifications for development.

3.3 Design Verification Plan

This Design Verification Plan focuses on verifying that the system's modular structure and interface specifications, as outlined in the Module Guide (MG) and Module Interface Specification (MIS), align with the requirements in the SRS. Each module's functionality, integration, and adaptability to potential changes will be reviewed to ensure the system is robust, maintainable, and ready for clinical use.

3.3.1 Objectives

The objectives of this Design Verification Plan are to:

1. Verify Modular Consistency with Requirements: Ensure that each module in the MG fulfills its corresponding SRS requirements.
2. Validate Information Hiding and Encapsulation: Confirm that modules are designed to encapsulate specific "secrets," promoting independence and ease of maintenance.
3. Check Interface Compatibility: Verify that module interfaces in the MIS support accurate data exchange and align with design constraints.
4. Assess Anticipated Changes and Traceability: Ensure that anticipated changes are contained within specific modules, preventing system-wide impact and maintaining traceability.

3.3.2 Verification Checklist

The following checklist covers key design aspects to be verified based on the MG and MIS documents:

1. Module Decomposition
    - Modules follow the information-hiding principle, addressing functional and non-functional requirements.
    - Modules are organized by hardware-hiding, behavior-hiding, and software decision layers to support system structure.
    - Requirements in the SRS are clearly mapped to modules, as shown in the traceability matrix.
2. Module Interfaces (MIS)
    - Interfaces include well-defined input and output formats to ensure seamless integration.
    - Exported constants, access routines, and state variables are specified for each module's functionality.
    - Interface compatibility is confirmed to support consistent interactions without conflicts.
3. Anticipated and Unlikely Changes
    - Design isolates anticipated changes within specific modules, reducing potential impact across the system.
    - Unlikely changes are documented to avoid unnecessary complexity and improve system simplicity.
4. Traceability and Use Hierarchy

- The traceability matrix links design elements to SRS requirements, supporting thorough validation.
- Use hierarchy between modules is verified to ensure logical dependency, with higher-level modules relying on lower-level modules without cyclic dependencies.

5. User Interface Design (if applicable)
   - For user interaction modules, verify that interface elements are intuitive and accessible.
   - Ensure compatibility of UI elements with any defined communication protocols, as outlined in the MG.

3.3.3 Methods

The following methods will be used to verify the design:

- Design Walkthroughs: Conduct walkthroughs with stakeholders and designers to confirm that the modular structure aligns with project goals.
- Consistency Checks with SRS: Cross-reference modules in the MG with the SRS requirements to confirm alignment.
- Interface Testing: Initial tests on module interfaces based on the MIS to confirm data and function accessibility across modules.
- Traceability Matrix Review: Verify that all modules and interfaces link back to requirements, ensuring comprehensive validation.

3.3.4 Verification Criteria

The design is considered verified if:

- Each module aligns with the requirements in the SRS, validated through walkthroughs and interface testing.
- The modular structure isolates changes effectively, supporting maintainability.
- The traceability matrix confirms all requirements are mapped to design elements.

This plan ensures the system's design is ready for development, with flexibility and reliability suited to clinical applications.

3.4 Verification and Validation Plan Verification Plan

This Verification and Validation (V&V) Plan Verification Plan outlines the process to ensure that the V&V document itself is thorough, accurate, and effectively designed to support all project goals. By verifying this V&V document, we aim to confirm that all sections—objectives, scope, methods, and criteria—are clearly defined and cover the requirements, design, and implementation needs outlined in the SRS, MG, and MIS documents.

3.4.1 Objectives

The objectives of this V&V Plan Verification Plan are to:

1. Ensure Completeness of Verification and Validation Processes: Verify that this V&V Plan includes all necessary sections to thoroughly validate the tool's requirements, design, and implementation.
2. Confirm Clarity and Traceability: Ensure that each section is clearly written, unambiguous, and easy to follow for all project stakeholders, with traceability to project documents.
3. Validate Feasibility of Methods: Check that the methods and criteria specified in the V&V Plan are feasible given project resources and timelines.
4. Identify Gaps and Overlaps: Ensure that all elements of the system are addressed without unnecessary repetition, minimizing gaps or redundant validations.

3.4.2 Verification Checklist

The following checklist identifies key elements in the V&V Plan to verify for completeness and accuracy:

1. Objectives and Scope
    - Verify that the V&V Plan includes well-defined objectives and scope covering all essential elements (SRS, MG, MIS).
    - Confirm that the plan covers functional and nonfunctional requirements, with a clear boundary on what is out of scope.
    - Verification and Validation Methods
    - Ensure that each section (e.g., SRS Verification Plan, Design Verification Plan) includes clearly defined methods and criteria for verifying requirements, design, and implementation.
    - Confirm that the V&V methods align with the constraints and objectives laid out in the project's SRS.
2. Traceability and Consistency
    - Check that the V&V Plan is traceable to all primary project documents (SRS, MG, MIS) with appropriate references, ensuring alignment and consistency across documents.
    - Ensure that each V&V Plan section is traceable back to specific requirements or design aspects from the SRS and MG.
3. Feasibility and Practicality of Methods
    - Confirm that each verification and validation method in the V&V Plan is achievable within the project's resources, timeline, and technical constraints.

- Check that tools, data, and stakeholder inputs needed for each V&V process are specified and accessible.

4. Criteria for Success
    - Ensure that the V&V Plan defines clear criteria for determining the success of each verification and validation step.
    - Verify that these criteria are realistic and aligned with project requirements, design goals, and intended system functionality.

3.4.3 Methods

To verify the V&V Plan:

- Document Review Meetings: Conduct meetings with project stakeholders and team members to review each section of the V&V Plan for clarity, feasibility, and traceability.
- Cross-Referencing with Project Documents: Check each verification and validation method against the SRS, MG, and MIS to confirm alignment and completeness.
- Feasibility Assessment: Evaluate the methods and tools proposed in the V&V Plan to ensure they are practical and achievable with available resources.

3.4.4 Verification Criteria

The V&V Plan will be considered verified if:

- All sections are complete, clearly written, and aligned with project documents (SRS, MG, MIS).
- The verification and validation methods are feasible and cover all essential elements.
- Stakeholders confirm that the V&V Plan is practical and provides a robust approach for testing and validating the tool.

This Verification Plan ensures that the V&V Plan itself is reliable, thorough, and ready to support a comprehensive validation process for the "Chest Scan" diagnostic tool.

3.5 Implementation Verification Plan

This Implementation Verification Plan outlines the steps to confirm that the "Chest Scan" diagnostic tool's final implementation accurately reflects its design, fulfills all specified requirements, and operates as intended. This plan ensures that each implemented feature undergoes rigorous testing to verify functionality, security, and usability as defined in the SRS, MG, and MIS documents.

3.5.1 Objectives

The objectives of the Implementation Verification Plan are to:

- Verify Functional Accuracy: Ensure that each implemented feature, such as disease detection and report generation, performs correctly according to the requirements.
- Confirm Compliance with Security and Usability Standards: Verify that patient data handling, access controls, and user interface meet security and usability guidelines.
- Check Integration of Modules: Validate that modules work cohesively and interface accurately with one another, as defined in the MG and MIS.

3.5.2 Verification Methods

The following methods will be used:

- Unit Testing: Test each module independently for correct functionality, ensuring it meets individual requirements from the SRS.
- Integration Testing: Test interactions between modules to confirm they work together as specified in the MIS, particularly for data handling and image processing.
- System Testing: Evaluate the full system under realistic conditions to verify end-to-end functionality, security, and usability.

3.5.3 Verification Criteria

The implementation will be considered verified if:

- All unit and integration tests pass according to defined requirements.
- Security and usability standards are met in system testing.
- Stakeholders confirm that the system functions accurately and meets project goals.

This plan ensures the implementation is verified to support accurate, secure, and user-friendly operations for clinical use.

3.6 Automated Testing and Verification Tools

Automated testing will ensure consistent verification of the "Chest Scan" diagnostic tool's functionality, accuracy, and security throughout its development. Based on the development plan, a selection of tools will be used to optimize testing efficiency, maintain code standards, and support continuous integration.

3.6.1 Objectives

The goals of automated testing are to:

- Enhance Testing Efficiency: Use automated tools to streamline testing processes, minimizing manual oversight.
- Support Continuous Integration and Deployment (CI/CD): Integrate tests within a CI pipeline to verify code on each update, reducing the risk of regressions.

3.6.2 Tools and Methods

Based on the development plan, the following tools will be employed:

- Unit Testing:
  - Pytest: Automates unit tests to ensure each Python module performs correctly, useful for validating core functionalities of the neural network.
  - Unittest: Alternative for straightforward unit tests, with easy integration into CI workflows.
- UI and Integration Testing:
  - Selenium: For automated testing of the web interface, checking usability, functionality, and responsiveness.
  - React Testing Library: For component-level testing within the React frontend, ensuring reliable user interactions.
- Continuous Integration:
  - GitHub Actions: Handles CI/CD pipeline setup, running tests on each pull request to verify code standards, formatting, and passing tests before merges.
  - Flake8, Black, and ESLint: Linting and formatting tools integrated with CI to enforce style standards and catch code errors early.

3.6.3 Success Criteria

Automated testing will be deemed effective if:

- CI pipeline tests pass consistently, ensuring code quality on each update.
- Functional, integration, and regression tests run smoothly with each deployment, supporting a reliable development workflow.

This automated approach ensures a high standard of quality, reducing manual testing needs and enabling quick identification of issues throughout the development lifecycle.

3.7 Software Validation Plan

This Software Validation Plan outlines the approach to confirm that the tool meets defined requirements and aligns with stakeholder expectations. External datasets, such as those from MIMIC-CXR and CheXpert, will be utilized to validate model accuracy and functionality. Additionally, user testing sessions with radiologists will help validate usability and practical applicability within clinical workflows.

3.7.1 Objectives

The objectives of the Software Validation Plan are to:

- Validate Requirement Alignment: Ensure that all requirements defined in the SRS are fulfilled through task-based inspections and stakeholder review sessions.
- Gather Stakeholder Feedback: Use review sessions, including the Rev 0 demonstration, as an opportunity to gather feedback from the supervisor and other stakeholders to refine and improve the tool.
- Confirm Real-World Usability: Validate that the tool's performance and interface meet clinical usability standards through task-based testing with radiologists.

3.7.2 Validation Methods

The following methods will be employed to confirm software effectiveness:

- Task-Based Inspections: Test key functionalities using real-world tasks that simulate common diagnostic processes to ensure accurate performance.
- Review and Demo Sessions: Conduct Rev 0 and Rev 1 demonstrations with the supervisor to validate the implementation against the requirements and incorporate feedback for improvement.
- User Testing: Gather feedback from radiologists during user testing sessions, focusing on ease of use, accuracy, and workflow compatibility.

3.7.3 Validation Criteria

The software will be considered validated if:

- External datasets confirm the model's diagnostic accuracy against real-world X-ray data.
- Review and demo session feedback indicates alignment with requirements and stakeholder expectations.
- User testing results show that radiologists find the tool effective and user-friendly in a clinical setting.

This validation plan leverages both external data and stakeholder input to ensure the tool is reliable, accurate, and clinically relevant.

4 System Tests

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, processing and disease signature generation, disease identification and classification, report generation, display functionality, user interface access, data storage and security, and authentication and authorization mechanisms. These subsets cover the key functionalities described in the functional requirements necessary for the success of this project.

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 Tests for Functional Requirements

4.1.1 Handling Input Data

FRTC1
Title: Chest X-ray Image 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 a valid format (e.g., DICOM, JPEG, PNG).
Output: The system accepts and reads the chest x-ray image successfully, with no error messages or system anomalies.
Test Case Derivation: The expected output is justified based on FR1 (The system shall accept and read chest x-ray images as input).
How the test will be performed:

1. Manually upload a sample chest X-ray image in a valid format through the user interface.
2. Observe the system's response to the input, checking for any error messages or unexpected behavior.
3. Verify that the system successfully accepts and reads the chest x-ray image.

FRTC2
Title: Invalid chest x-ray Image Format Rejection
Control: Manual
Initial State: The system is in a stable state with all components initialized and ready to receive input.
Input: A sample image in an invalid format (e.g., .txt, .docx).
Output: The system rejects the invalid image input, accompanied by an appropriate error message.
Test Case Derivation:

1. Manually attempt to upload an image in an invalid format through the user interface.
2. Observe the system's response to the input, checking for any error messages or unexpected behavior.
3. Verify that the system rejects the invalid image and displays an appropriate error message.

4.1.2 Processing and Disease Signature Generation

FRTC3
Title: Disease Signature Generation Using Diffusion Model
Control: Automatic
Initial State: The system is ready with the diffusion model loaded and operational.
Input: A sample chest x-ray image and specified disease locations (if applicable).
Output: The system processes the chest x-ray image using the diffusion model and generates an output image with added disease signatures at specified locations.
Test Case Derivation:

1. Submit a sample chest x-ray image with specified disease locations to the system.
2. Allow the system to process the image using the diffusion model.
3. Verify that the output image has disease signatures generated at the specified locations.
4. Compare the output with expected results to ensure correctness.

FRTC4
Title: Diffusion Model Error Handling with Invalid Input
Control: Automatic
Initial State: The system is ready with the diffusion model loaded.
Input: An invalid or corrupted chest x-ray image.
Output: The system handles the error gracefully, providing an appropriate error message without crashing.
Test Case Derivation:

1. Submit an invalid or corrupted chest x-ray image to the system.
2. Observe the system's response.
3. Verify that the system detects the invalid input and provides an appropriate error message without crashing.

4.1.3 Disease Identification and Classification

FRTC5
Title: Correct Disease Classification on chest x-ray Image
Control: Automatic (with manual verification)
Initial State: The system is in a stable state with the disease identification model loaded and ready.
Input: A chest x-ray image containing known disease patterns for Pneumonia, Atelectasis, Cardiomegaly, and Pleural Effusion.
Output: The system identifies and classifies the diseases present in the image with an accuracy greater than 90% for each disease.
Test Case Derivation:

1. Input a set of chest x-ray images with known diagnoses into the system.
2. Allow the system to process the images and identify diseases.
3. Compare the system's output with the known diagnoses.
4. Calculate the accuracy for each disease classification.
5. Verify that the accuracy meets or exceeds 90% for each disease.

FRTC6
Title: Disease Absence Correct Identification
Control: Automatic (with manual verification)
Initial State: The system is in a stable state with the disease identification model loaded.
Input: A chest x-ray image with no disease present (healthy patient).
Output: The system correctly identifies the absence of diseases, with an accuracy greater than 90%.
Test Case Derivation:

1. Input a set of normal (healthy) chest x-ray images into the system.
2. Allow the system to process the images and identify diseases.
3. Verify that the system reports the absence of diseases accurately.
4. Calculate the accuracy of disease absence identification.
5. Confirm that the accuracy is greater than 90%.

4.1.4 Report Generation

FRTC7
Title: Structured Diagnostic Report Generation
Control: Automatic (with manual verification)
Initial State: The system has successfully identified diseases in the input chest x-ray image.
Input: chest x-ray images with known disease findings.
Output: The system generates a diagnostic report detailing the diseases, their severity, and the locations in the image where the abnormalities were detected.
Test Case Derivation:

1. Input a chest x-ray image with known disease findings into the system.
2. Allow the system to process the image and generate the diagnostic report.
3. Review the generated report to ensure it includes:
4. Detected diseases
5. Severity levels
6. Locations of abnormalities in the image

7. Compare the report to the known findings to verify correctness.

FRTC8
Title: Report Generation Error Handling
Control: Automatic
Initial State: The system is in a stable state but with incomplete disease identification results (simulate a processing error).
Input: A scenario where disease identification results are incomplete or missing.
Output: The system handles the incomplete data gracefully, indicating in the report that certain findings could not be determined.
Test Case Derivation:

1. Simulate a scenario where the disease identification module fails to provide results.
2. Attempt to generate a diagnostic report.
3. Observe how the system handles the missing data.
4. Verify that the report indicates which findings are missing and does not crash or produce misleading information.

4.1.5 Display of Heatmaps and Reports

FRTC9
Title: Heatmap Display on chest x-ray Images
Control: Manual
Initial State: The system has successfully identified diseases and generated heatmaps indicating disease locations.
Input: A processed chest x-ray image with heatmaps generated.
Output: The system overlays heatmaps on the chest x-ray images, accurately highlighting regions where disease signatures are present.
Test Case Derivation:

1. Access the user interface and retrieve the processed chest x-ray image with heatmaps.
2. Observe the displayed image with heatmaps overlaid.
3. Verify that the heatmaps accurately correspond to the locations of detected diseases.
4. Compare the heatmaps to the original findings to ensure correctness.

FRTC10
Title: Diagnostic Report and Heatmap Access via Web Interface
Control: Manual
Initial State: The system has processed chest x-ray images, generated reports and heatmaps, and stored them.
Input: User access through the web-based user interface.

Output: The system successfully displays diagnostic reports and heatmaps on the web interface for authorized users.
Test Case Derivation:

1. Log in to the web-based user interface as an authorized user.
2. Navigate to the section containing diagnostic reports and heatmaps.
3. Open a diagnostic report and the associated heatmap.
4. Verify that the report and heatmap are displayed correctly and are accessible.

4.1.6 User Interface Access

FRTC11
Title: Authorized User Access to Diagnostic Results
Control: Manual
Initial State: The system is operational, with diagnostic reports and heatmaps available, and user accounts set up.
Input: Authorized user login credentials.
Output: The user gains access to the diagnostic reports and heatmaps via the web-based interface.
Test Case Derivation:

1. Attempt to log in to the web-based user interface using valid credentials of an authorized user.
2. Upon successful login, navigate to the diagnostic results.
3. Verify that the user can access and view the diagnostic reports and heatmaps.

FRTC12
Title: Unauthorized User Access Denied
Control: Manual
Initial State: The system is operational with diagnostic results available.
Input: Invalid login credentials or an unauthorized user.
Output: The system denies access to diagnostic reports and heatmaps, displaying an appropriate error message.
Test Case Derivation:

1. Attempt to log in using invalid credentials or as a user without the necessary permissions.
2. Observe the system's response.
3. Verify that access is denied and an appropriate error message is displayed.

4.1.7 Data Storage and Security

FRTC13
Title: Secure Storage of Patient Data and Diagnostic Results
Control: Manual and Automatic
Initial State: The system is operational and has processed chest x-ray images and generated diagnostic reports.
Input: chest x-ray images, patient data, and diagnostic reports.
Output: The system securely stores and retrieves chest x-ray images, diagnostic reports, and patient information in a secure database.
Test Case Derivation:

1. Input patient data and chest x-ray images into the system.
2. Allow the system to process the images and generate diagnostic reports.
3. Verify that all data is stored in the backend database securely.
4. Attempt to retrieve the stored data and ensure that data integrity is maintained.
5. Verify that unauthorized access to the database is prevented.

FRTC14
Title: Data Encryption in Storage and Transit
Control: Automatic
Initial State: The system is operational with data storage and network communication components active.
Input: Patient data, chest x-ray images, and diagnostic reports during storage and retrieval operations.
Output: The data is encrypted during storage and transit, ensuring patient confidentiality.
Test Case Derivation:

1. Monitor data storage processes to verify that encryption is applied to data at rest.
2. Monitor network communication during data transmission to ensure data is encrypted in transit (e.g., using HTTPS).
3. Use security tools to attempt to intercept or access data without authorization.
4. Verify that data remains secure and encrypted, preventing unauthorized access.

4.1.8 Authentication and Authorization

FRTC15
Title: User Authentication with Valid Credentials
Control: Manual
Initial State: The system is operational with user accounts created.
Input: Valid login credentials for an authorized user.

Output: The system authenticates the user and grants access according to the user's role.
Test Case Derivation:

1. Attempt to log in using valid credentials.
2. Verify that the system authenticates the user successfully.
3. Verify that the user has access to functions appropriate for their role.

FRTC16
Title: User Authentication with Invalid Credentials
Control: Manual
Initial State: The system is operational.
Input: Invalid login credentials.
Output: The system denies access and displays an appropriate error message.
Test Case Derivation:

1. Attempt to log in using invalid credentials (wrong username or password).
2. Observe the system's response.
3. Verify that access is denied and an appropriate error message is displayed.

FRTC17
Title: Role-Based Access Control Enforcement
Control: Manual
Initial State: The system is operational with users assigned different roles (e.g., radiologist, administrator, technician).
Input: Users attempting to access functions outside their authorized roles.
Output: The system restricts actions based on user roles, preventing unauthorized access.
Test Case Derivation:

1. Log in as a user with a specific role (e.g., technician).
2. Attempt to perform actions that are restricted to other roles (e.g., accessing patient reports).
3. Verify that the system denies access to unauthorized functions.
4. Repeat the test with other roles to ensure proper enforcement of access controls.

4.2 Tests for Nonfunctional Requirements

In this subsection, the system test cases are designed to validate key non-functional requirements that are critical to the user experience and overall system performance. The tests focus on the appearance, usability, performance, and security aspects of the software. Each test case references the relevant non-functional requirements from the SRS to ensure comprehensive coverage and traceability.

4.2.1 Appearance Requirements

NFRTC1
Title: Verification of Calming Color Scheme
Control: Manual
Initial State: The web application is deployed and accessible via a web browser.
Input: Access the web application using a standard web browser.
Output: The web app displays a color scheme that combines white and soft tones of blue.
Test Case Derivation: The expected output is justified based on NF-AR1 (The web app will incorporate a cool, and calming color scheme that combines white and soft tones of blue).
How the test will be performed:

1. Open the web application in a web browser.
2. Observe the color scheme of the interface, noting the primary colors used.
3. Verify that the colors are white and soft tones of blue.
4. Conduct a user survey where participants rate whether the color scheme contributes positively to their experience.

NFRTC2
Title: Consistency of Simple and Uncluttered Style
Control: Manual
Initial State: The web application is fully functional with all interface elements loaded.
Input: Navigate through different pages and features of the web application.
Output: The web app maintains a consistent, simple, and uncluttered style throughout.
Test Case Derivation: The expected output is justified based on NF-SR1 and NF-SR2 (The web app will have a consistent simple style; The software interface shall maintain a clean and organized aesthetic).
How the test will be performed:

1. Access the main page of the web application.
2. Navigate through all available features and pages.
3. Observe the layout, design elements, and interface consistency.
4. Ensure that there are no unnecessary or distracting design elements.
5. Conduct usability tests where participants navigate the interface and report on ease of navigation and aesthetic appeal.

4.2.2 Usability and Humanity Requirements

NFRTC3
Title: Ease of Performing Core Functionality Without Guidance
Control: Manual

Initial State: The web application is accessible, and new users are ready to interact with it.
Input: First-time users attempt to generate a chest X-ray image without any prior training or guidance.
Output: Users are able to perform basic tasks, like generating an image, without needing extra guidance, and no irrelevant features are present.
Test Case Derivation: The expected output is justified based on NF-EUR0 (The web app should focus only on core functionality, with no extra features that might confuse users).
How the test will be performed:

1. Recruit a group of participants with no prior experience using the web app.
2. Ask participants to generate a chest X-ray image using the app without providing any instructions.
3. Observe their interactions, noting any difficulties or confusion.
4. Verify that users can generate an image without assistance within the first minute of use.
5. Ensure that no irrelevant features distract or confuse the users during the process.

NFRTC4
Title: Intuitiveness and Memorability of the Software
Control: Manual
Initial State: Users have completed 1-2 training sessions on using the software.
Input: One month after training, users attempt to navigate the software and perform basic functions without assistance.
Output: Users report they can navigate the software and perform basic functions without assistance.
Test Case Derivation: The expected output is justified based on NF-EU2 (The software shall be intuitive enough that users can remember how to operate it after 1-2 training sessions).
How the test will be performed:

1. Provide 1-2 training sessions to a group of users on how to use the software.
2. Wait one month without any further interaction with the software.
3. Ask users to perform basic tasks on the software without any assistance.
4. Collect feedback through surveys or interviews regarding their ability to navigate and use the software.
5. Analyze responses to ensure that at least 80% of users could operate the software effectively without assistance.

NFRTC5
Title: Learning Curve Assessment

Control: Manual
Initial State: New users with no prior experience are ready to use the web application.
Input: Users attempt to generate a chest X-ray image upon first use of the app.
Output: Users are able to generate an image within the first minute of using the app, even without prior experience.
Test Case Derivation: The expected output is justified based on NF-LR0 (The web app should be easy to learn, allowing users to quickly understand how to use it without needing tutorials or training).
How the test will be performed:

1. Recruit participants with no prior experience using the web app.
2. Provide them with access to the web application without any instructions or tutorials.
3. Ask them to generate a chest X-ray image.
4. Measure the time taken from first accessing the app to successfully generating an image.
5. Verify that users can accomplish this within one minute.
6. Collect any feedback on their learning experience.

NFRTC6
Title: Realism of Generated Chest X-Ray Images
Control: Manual (with expert evaluation)
Initial State: The system is operational and capable of generating chest X-ray images.
Input: Generate a set of chest X-ray images using the diffusion model.
Output: Radiologists or other medical professionals are unable to easily distinguish between the generated X-rays and real-world X-rays after reviewing them.
Test Case Derivation: The expected output is justified based on NF-UPR0 (The chest X-ray images generated by the diffusion model should closely resemble real-world X-rays used in hospitals).
How the test will be performed:

1. Generate a sample set of chest X-ray images using the diffusion model.
2. Compile a mixed set of images including both generated images and real-world X-rays.
3. Present this set to a group of medical professionals (e.g., radiologists).
4. Ask them to identify which images are real and which are generated.
5. Collect and analyze the results.
6. Verify that medical professionals cannot easily distinguish between the two sets, indicating high realism in the generated images.

4.2.3 Performance Requirements

NFRTC7
Title: Image Generation Speed Test
Control: Automatic
Initial State: The system is operational and ready to process image generation requests.
Input: A request to generate a chest X-ray image.
Output: The system generates the chest X-ray image within 1 minute of the user request.
Test Case Derivation: The expected output is justified based on NF-SLR0 (The system should generate a chest X-ray image within a reasonable amount of time after a request is made).
How the test will be performed:

1. Initiate an image generation request through the web app.
2. Start a timer immediately upon submission of the request.
3. Record the time taken for the system to generate and display the chest X-ray image.
4. Ensure that the image is generated and available to the user within 1 minute.
5. Repeat the test multiple times to account for variations in processing time.

NFRTC8
Title: System Stability and Uptime Monitoring
Control: Automatic (over an extended period)
Initial State: The system is deployed and accessible to users.
Input: Continuous monitoring of system availability over a specified period (e.g., one month).
Output: The system remains available at least 99% of the time, with minimal downtime (no more than 1 hour of downtime per week).
Test Case Derivation: The expected output is justified based on NF-RFTR0 (The system should be stable and available for use most of the time).
How the test will be performed:

1. Set up monitoring tools to track system uptime and downtime continuously over the testing period.
2. Record any instances of system downtime, noting the duration and cause.
3. Calculate the total uptime percentage over the testing period.
4. Verify that the system maintains at least 99% uptime.
5. Ensure that any downtime does not exceed 1 hour per week.

NFRTC9
Title: Single Image Generation Capacity Enforcement
Control: Manual and Automatic

Initial State: The system is operational and processing an image generation request.
Input: Attempt to initiate multiple simultaneous image generation requests.
Output: The system allows only one image to be generated at a time, rejecting additional simultaneous requests with an appropriate message.
Test Case Derivation: The expected output is justified based on NF-CR0 (The system should only generate one chest X-ray image at a time).
How the test will be performed:

1. Initiate an image generation request and confirm that the process has started.
2. While the first request is being processed, attempt to initiate additional image generation requests from the same or different user accounts.
3. Observe the system's response to the additional requests.
4. Verify that the system prevents additional simultaneous image generation requests and provides an appropriate notification to the user.
5. Ensure that the first image generation process continues uninterrupted.

4.2.4 Security Requirements

NFRTC10
Title: Public Accessibility Without Authentication
Control: Manual
Initial State: The web application is deployed and accessible over the internet.
Input: Access the web app from multiple devices and locations without providing any login credentials.
Output: The web app is accessible to anyone without needing to sign in or provide any credentials.
Test Case Derivation: The expected output is justified based on NF-AR0 (The web app will be publicly accessible to anyone, without the need for login or credentials).
How the test will be performed:

1. Access the web application from different devices (e.g., desktop, laptop, tablet, smartphone) and different web browsers.
2. Attempt to use the web app's functionalities without logging in or providing any personal information.
3. Verify that all core functionalities are accessible without any authentication prompts. Ensure consistent access across various devices and locations.

NFRTC11
Title: Verification of Non-Collection of Personal Data
Control: Manual and Automatic
Initial State: The web application is operational and ready for user interaction.
Input: Interact with the web app and monitor data collection processes.

Output: The web app does not request or store any personal information from users during their interactions with the system.
Test Case Derivation: The expected output is justified based on NF-PR0 (The system will not collect any personal data from users interacting with the web app).
How the test will be performed:

1. Use the web application extensively, attempting to access all available features.
2. Observe whether the app requests any personal information (e.g., name, email, contact details).
3. Use network monitoring tools to inspect data being sent to the server during interactions.
4. Verify that no personal data is being transmitted or stored.
5. Review the application's privacy policy and any data handling practices to ensure compliance.

4.2.5 Supportability Requirements

NFRTC12
Title: Availability and Sufficiency of User Documentation
Control: Manual
Initial State: The web application is operational with user documentation provided.
Input: Access the user documentation and attempt to experiment with the diffusion model based on the guidelines.
Output: Users can access the documentation and experiment with the model without needing further assistance.
Test Case Derivation: The expected output is justified based on NF-SR0 (The system will be self-sufficient, with all necessary documentation provided so that users can understand how to experiment with the diffusion model).
How the test will be performed:

1. Locate and access the user documentation or guidelines provided within the web app.
2. Read through the documentation to understand how to use the diffusion model.
3. Attempt to perform tasks such as generating images and adjusting any available parameters based on the instructions.
4. Evaluate whether the documentation provides clear and sufficient guidance for using the model.
5. Collect feedback from users on the clarity and usefulness of the documentation.

4.2.6 Cultural Requirements

NFRTC13
Title: Functionality of Browser-Based Language Translation
Control: Manual
Initial State: The web application is operational and written in English.
Input: Access the web app using a web browser's translation feature to translate it into another language.
Output: The web app is fully functional in English, and users are able to translate it into other languages using their browser if necessary.
Test Case Derivation: The expected output is justified based on NF-CR0 (The web app will be usable by people from any background or culture with no restrictions such as in region).
How the test will be performed:

1. Access the web application using a browser that supports automatic translation (e.g., Google Chrome).
2. Use the browser's translation feature to translate the web app into a different language.
3. Navigate through the app and perform core functionalities while in the translated version.
4. Verify that the translation is applied consistently across the interface.
5. Ensure that the app remains fully functional after translation, with no broken elements or errors.

4.3 Untested Non-Functional Requirements

Some non-functional requirements are not tested due to limitations in scope, feasibility, or time constraints. These requirements are recognized but are not included in the testing plan.

- **NF-SER0 (Scalability or Extensibility Requirements):** Testing future scalability and updates is beyond our current project scope.
- **NF-LR0 (Longevity Requirements):** We cannot test the system's operation over a span of three years within our testing timeline.
- **NF-MR0 (Maintenance Requirements):** Assessing future maintenance activities isn't feasible during this phase.
- **NF-EPE0 (Expected Physical Environment):** This is an operational expectation and doesn't require specific testing.
- **NF-RIAS0 (Interfacing with Adjacent Systems):** Since the system is self-contained with no external interfaces, no tests are necessary.
- **NF-IR0 (Integrity Requirements):** The system outputs images without additional data protection measures; thus, there's nothing specific to test here.

- **NF-AR0 (Audit Requirements):** No formal security audits are planned, so no related tests are designed.

4.4 Traceability Between Test Cases and Requirements

Functional Requirement	**Test Cases**
FR1: The system shall accept and read CXR (Chest X-ray) images as input.	FRTC1, FRTC2
FR2: The system shall process CXR images using a diffusion model to generate disease signatures at specified locations.	FRTC3, FRTC4
FR3: The system shall identify and classify multiple diseases in the CXR image, including (but not limited to) Pneumonia, Atelectasis, Cardiomegaly, and Pleural Effusion.	FRTC5, FRTC6
FR4: The system shall generate a structured diagnostic report from the findings on the CXR image.	FRTC7, FRTC8
FR5: The system shall display heatmaps on the CXR images to indicate the locations of the detected disease signatures.	FRTC9
FR6: The system shall allow users to access the diagnostic reports and heatmaps via a web-based user interface.	FRTC10, FRTC11
FR7: The system shall store patient data, CXR images, and diagnostic reports securely in a backend database.	FRTC13, FRTC14
FR8: The system shall provide authentication and authorization mechanisms to control access to the system.	FRTC11, FRTC12, FRTC15, FRTC16, FRTC17

Non-Functional Requirement	**Test Cases**
NF-AR1: The web app will incorporate a cool, and calming color scheme that combines white and soft tones of blue.	NFRTC1
NF-SR1: The web app will have a consistent simple style that makes it easy to use and experiment with the model, without adding unnecessary complexity.	NFRTC2
NF-SR2: The software interface shall maintain a clean and organized aesthetic.	NFRTC2
NF-EUR0: The web app should focus only on core functionality, with no extra features that might confuse users.	NFRTC3
NF-EU2: The software shall be intuitive enough that users can remember how to operate it after 1-2 training sessions.	NFRTC4
NF-LR0: The web app should be easy to learn, allowing users to quickly understand how to use it without needing tutorials or training.	NFRTC5
NF-UPR0: The chest X-ray images generated by the diffusion model should closely resemble real-world X-rays used in hospitals.	NFRTC6
NF-SLR0: The system should generate a chest X-ray image within a reasonable amount of time after a request is made.	NFRTC7
NF-RFTR0: The system should be stable and available for use most of the time.	NFRTC8
NF-CR0: The system should only generate one chest X-ray image at a time.	NFRTC9, NFRTC13
NF-AR0: The web app will be publicly accessible to anyone, without the need for login or credentials.	NFRTC10
NF-PR0: The system will not collect any personal data from users interacting with the web app.	NFRTC11
NF-SR0: The system will be self-sufficient, with all necessary documentation provided so that users can understand how to experiment with the diffusion model.	NFRTC12

5 Unit Test Description

This section outlines the approach and rationale for the unit tests designed to verify each module within the chest X-ray diagnostic application. The overall philosophy for test case selection centers on ensuring each module functions correctly in isolation, covering both standard functionality and potential edge cases. This approach provides a robust foundation to identify and address any issues at the unit level before integrating modules into the broader system.

To streamline testing and avoid redundancy, each module's tests are designed to validate both typical and boundary conditions. This strategy includes creating a baseline test case for normal expected behavior, accompanied by additional edge cases to capture unexpected or extreme inputs. This allows for comprehensive testing without unnecessary detail for

every possible scenario.

For efficiency, test cases prioritize high-risk functions and core functionalities. Tests are organized to:

- Validate standard functionality with a single test for typical cases.
- Test critical edge cases that reflect likely and realistic error conditions (e.g., handling of empty data, invalid formats, and unexpected large data inputs).
- Ensure modules perform within expected limits for nonfunctional requirements, such as performance and security, through specialized tests.

Where possible, detailed input/output descriptions are omitted in favor of referring to well-documented unit testing code, which maintains readable and meaningful test names. This approach not only saves space but also facilitates future maintenance and expansion, as new edge cases can be added to the codebase directly.

5.1 Unit Testing Scope

The scope of unit testing for this project includes all the core modules that are necessary to the chest X-ray diagnostic application. Each core module will be tested independently to verify its functionality (unit tests), handling expected and edge cases.

The primary modules in scope include:

- **ReadXRayImage Module** – Responsible for reading and validating DICOM images
- **DiagnosticAnalysis Module** – Performs AI-based analysis on images to generate diagnostic predictions.
- **ReportBuilder Module** – Compiles the AI-generated results into structured diagnostic reports.
- **DatabaseOperations Module** – Manages patient data storage and retrieval

PredictiveModel Module – Executes the core disease classification logic, supporting accurate and efficient diagnosis. Modules that are not included in the unit testing scope are:

- **Third-Party Libraries and External APIs** – Any external libraries or APIs used in the system are assumed to have been verified by their respective developers. These components are not unit-tested here but are included in system and integration testing.

5.2 Tests for Functional Requirements

This section lists unit tests grouped by module, focusing on both standard and edge case scenarios.

5.2.1 ReadXRayImage Module

Tests for this module verify that DICOM images are read, validated, and handled correctly.

1. **Test Name:** Retrieve and Validate DICOM Files
   Type: Automatic
   Initial State: System ready to receive DICOM file
   Function Tested: `fetch_and_validate_dicom`
   Input: Valid patient data entries from Database
   Expected Output: Returns list of valid DICOM files
   Test Case Derivation: Confirms that valid directories are read and files retrieved correctly.
   How the test will be performed: provide a directory path containing DICOM files; verify that each file is returned as expected.
   Edge Case: Empty patient directory or inaccessible directory
   Expected Output for Edge Case: Returns an empty list or a specific error indicating no files found.

2. **Test Name:** Check File Format Compatibility
   Type: Automatic
   Initial State: System ready to receive DICOM file
   Function Tested: `is_dicom_format`
   Input: Sample DICOM, JPEG, and corrupted files
   Expected Output: True for DICOM, False for JPEG, Error for corrupted file
   Test Case Derivation: Verifies correct file identification and error handling for incompatible file types.
   How test will be performed: Provide sample files to the function; confirm correct identification or error logging.

3. **Test Name:** Handle Corrupted DICOM File Input
   Type: Automatic
   Initial State: System ready to receive DICOM file
   Function Tested: `read_dicom_data`
   Input: Corrupted DICOM file
   Expected Output: Returns an error or logs issue without crashing
   Test Case Derivation: Ensures model handles unreadable data without crashing.
   How test will be performed: Provide a corrupted DICOM file; confirm the model logs an error and skips further processing.
   Edge Case: Partially transferred or incomplete DICOM file
   Expected Output for Edge Case: Gracefully handles error, logs details, and skips further processing

5.2.2 DiagnosticAnalysis Module

This module generates diagnostic predictions for chest X-rays. It includes tests for typical image inputs, as well as abnormal images.

1. **Test Name:** Prediction on Standard X-ray Image

Type: Automatic
Initial State: Model loaded and ready for analysis
Function Tested: `analyze_image`
Input: Raw data for a valid single X-ray image
Expected Output: Prediction results for any identified conditions.
Test Case Derivation: Confirms the model's diagnostic capability with a basic and simple input.
How test will be performed: Submit a valid X-ray image; verify prediction results match the expected conditions.
Edge Case: X-ray image that does not portray any health issues
Expected Output for Edge Case: Processes image as expected and is able to identify there are no health issues in a reasonable processing time.

2. **Test Name:** Process Empty Image Data
Type: Automatic
Initial State: Model loaded and ready for analysis
Function Tested: `analyze_image`
Input: Empty or null image data
Expected Output: Error message or safe default response
Test Case Derivation: Confirms that invalid or blank images are appropriately flagged without prediction.
How test will be performed: Submit empty data; confirm error message or safe response output.
Edge Case: image with unidentifiable image
Expected Output for Edge Case: Returns "unreadable image" error

5.2.3 ReportBuilder Module

This module compiles the prediction results into readable reports. The tests validate accurate report generation and the handling of incomplete input data.

1. **Test Name:** Generate Summary Report from Prediction Results
Type: Automatic
Function Tested: `compile_summary`
Initial State: Ready to compile report from prediction data
Input: Prediction results for a single patient (could contain multiple x-ray images)
Expected Output: Well-structured report summary
Test Case Derivation: Confirms that valid prediction data generates a complete report.
How test will be performed: Provide prediction data; verify that the report summary is correctly structured.

2. **Test Name:** Handle low quality/quantity Prediction Results

Type: Automatic
Initial State: Ready to compile report from prediction data
Function Tested: `compile_summary`
Input: Partially complete prediction data
Expected Output: Complete report with placeholders or indicators for missing data
Test Case Derivation: Ensures that reports can be generated even when data is partially missing.
How test will be performed: Provide partial data; verify the report includes placeholders or notes for missing information.
Edge Case: No predictions available at all
Expected Output for Edge Case: Outputs a "no findings" report or similar message indicating a lack of results.

5.2.4 DataBaseOperations Module

This module focuses on storing and retrieving patient data, while ensuring security and data integrity.

1. **Test Name:** Store Data Attempt
   Type: Automatic
   Initial State: Ready to authenticate user
   Function Tested: `store_patient_data`
   Input: Valid patient data, User Credentials
   Expected Output: Data stored successfully
   Test Case Derivation: Confirms only authorized users can enter data.
   How test will be performed: Submit patient data as an authorized user; verify successful data entry.
   Edge Case: User Credentials are invalid
   Expected Output for Edge Case: Logs warning for invalid user credentials and it timeouts after 5 attempts.

2. **Test Name:** Access Data Attempt
   Type: Automatic
   Initial State: Ready to authenticate user
   Function Tested: `retrieve_patient_data`
   Input: User Credentials, valid patient data
   Expected Output: retrieves patient data
   Test Case Derivation: Confirms only authorized users can read data.
   How test will be performed: Read patient data as an authorized user; verify that data was successfully read.
   Edge Case: User Credentials are invalid
   Expected Output for Edge Case: Logs warning for invalid user credentials and it timeouts after 5 attempts.

5.2.5 PredictiveModel Module

This module is a core component responsible for processing chest X-ray images using advanced AI techniques, including diffusion models with bounding box and inpainting capabilities.

1. **Test Name:** Diagnose with Inpainting for Object Removal
   Type: Automatic
   Initial State: Model ready for prediction
   Function Tested: `process_and_predict`
   Input: X-ray image containing a visible artifact (e.g., shadow)
   Expected Output: Clear diagnostic results after the artifact is removed and replaced with realistic content.
   Test Case Derivation: Confirms model's ability to enhance image by removing artifacts.
   How test will be performed: provide X-ray with artifact; verify that model removes the artifact and produces an accurate diagnosis.
   Edge Case: Artifact located close to an area of suspected abnormality
   Expected Output for Edge Case: The model correctly removes only the artifact while preserving the surrounding area for correct analysis.

2. **Test Name:** Diagnose Condition with Focused Enhancement
   Type: Automatic
   Initial State: Model ready for focused analysis
   Function Tested: `enhance_and_predict`
   Input: X-ray with a faint or hard-to-see abnormal region marked by a bounding box
   Expected Output: Clearer image in the specified area and a correct diagnosis
   Test Case Derivation: Confirms that unfocused areas receive focus- enhancement for better accuracy.
   How test will be performed: Provide X-ray with hard to see areas; confirm that the enhanced area supports accurate diagnosis.

3. **Test Name:** Restore Blurred or Damaged Areas in X-ray
   Type: Automatic
   Initial State: Model ready for restoration
   Function Tested: `restore_and_predict`
   Input: X-ray image with blurry or degraded areas
   Expected Output: Restored image with improved clarity and accurate diagnostic results
   Test Case Derivation: Verifies model's ability to improve clarity in damaged areas.
   How test will be performed: Submit X-ray with blurred regions; confirm that model restores these areas.

5.3 Tests for Nonfunctional Requirements

Tests in this section ensure that the application meets nonfunctional requirements such as performance and security.

5.3.1 Performance Testing - PredictiveModel Module

1. **Test Name:** Bulk Image Processing Speed
   Type: Automatic
   Initial State: Model loaded and ready for batch processing
   Input: High volume batch of X-ray images
   Expected Output: Processes batch within acceptable time frame.
   How test will be performed: A large batch of X-ray images is submitted to the model for processing. The system's response time is measured and logged.
   Edge Case: Maximum batch size allowed by the system
   Expected Output for Edge Case: Handles maximum load with minimal delay, logs if processing times exceed threshold.

5.3.2 Security Testing - DataBaseOperations Module

1. **Test Name:** Unauthorized Access Block
   Type: Automatic
   Initial State: Ready to Authenticate User
   Input: Access attempt from unauthorized user
   Expected Output: Denies access and logs attempt
   How test will be performed: Send unauthorized access attempts to patient data, verifying the system's response. The test checks that unauthorized access is blocked.

5.4 Traceability Between Test Cases and Modules

Table 2: Legend for Module Labels

Module Label	Description
M1	ReadXRayImage
M2	DiagnosticAnalysis
M3	ReportBuilder
M4	DatabaseOperations
M5	PredictiveModel

Table 3: Traceability Matrix for Functional Requirement Test Cases (FRTC)

FRTC	M1	M2	M3	M4	M5
FRTC1	X			X	
FRTC2		X			
FRTC3		X			X
FRTC4			X		
FRTC5				X	
FRTC6		X			X

Table 4: Traceability Matrix for Non-Functional Requirement Test Cases (NFRTC)

NFRTC	M1	M2	M3	M4	M5
NFRTC1	X				
NFRTC2		X			X
NFRTC3			X		
NFRTC4				X	
NFRTC5	X			X	
NFRTC6		X			X
NFRTC7				X	
NFRTC8		X			X
NFRTC9					X
NFRTC10			X		
NFRTC11				X	

6 Appendix

In this section, additional information that complements the V&V Plan is included.

6.1 Symbolic Parameters

The definition of test cases will call for certain symbolic constants. Their values are defined in this section for easy maintenance. See the following table for reference.

Symbolic Name	Definition or Description
DATASET-NAME	The chest X-ray imaging datasets used for development and testing. Details are as follows:
MIMIC-CXR	A standard labeled chest X-ray dataset commonly used for medical imaging tasks.
Chest ImaGenome	An extension of MIMIC-CXR, with detailed annotations including 29 bounding boxes on anatomical regions. This dataset links textual report segments to specific bounding boxes, supporting tasks in segmentation and disease localization.

6.2 Usability Survey Questions?

The following survey should be filled out by users after using the system for 5 to 15 minutes. This feedback helps evaluate the user experience and identifies areas for improvement.

6.2.1 User Experience Survey

Time using system: Please provide a rating between 0 and 10 for each of the following categories, with 0 being very difficult or undesirable, and 10 being very easy or desirable.

1. Ease of Use: 0 1 2 3 4 5 6 7 8 9 10
   (0 = very difficult to use, 10 = very easy to use)
2. Navigation: 0 1 2 3 4 5 6 7 8 9 10
   (0 = hard to find what you're looking for, 10 = easy to find what you're looking for)
3. Readability: 0 1 2 3 4 5 6 7 8 9 10
   (0 = hard to understand information, 10 = very easy to understand)
4. Look and Feel: 0 1 2 3 4 5 6 7 8 9 10
   (0 = visually unappealing, 10 = visually appealing)

6.2.2 Additional Feedback:

Notes: (Space for users to leave any additional comments or suggestions about the system.)

Appendix — Reflection

6.3 Question 1

What went well while writing this deliverable?

Harrison Chiu: Planning out the modules for functional testing went pretty smoothly. Reviewing SOLID principles helped a lot and I was able to structure each module with a clear purpose, which made creating the tests feel straightforward. I even tried some initial tests on my own machine to catch edge cases early on, and it gave me confidence that I was covering all the important scenarios. That fuzz testing experiment helped me feel like I was on the right track.

Hamza Issa: Working on identifying tests for functional requirements felt really intuitive. After we reached a consensus regarding the functional requirements and our ability to implement it, it was quite simple to devise tests that were complete and meaningful, as part of creating functional requirements is that the expectation action or output is typically discrete.

Gurnoor Bal: Working on the SRS verification was actually pretty satisfying because I got to make sure everything was crystal clear. I found a few areas in the SRS that needed improvement, and being able to fix those early felt good. Plus, setting up the traceability matrix was helpful since it kept things organized and made it easy to link each requirement with its specific test case. It felt like things were clicking into place.

Jared Paul: I liked working on the design verification part because it let me focus on making sure the system was organized and easy to work with. I emphasized modularity and made sure each module was self-contained, which will make the system easier to maintain down the road. I also created a checklist to make sure each module followed solid design practices, so I felt like I was covering the important aspects without missing anything.

Ahmad Hamadi: Writing the validation plan was rewarding because I got to think about how people would actually use the tool, especially radiologists. I set up task-based testing and feedback sessions, so we'll get a real sense of how practical the tool is. Using actual datasets to check the model's accuracy felt like a solid choice too. It felt like I was creating a plan that would make sure the tool was both accurate and user-friendly.

6.4 Question 2

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

Hamza Issa: I found that in order to create a V&V that was really complete meant that I needed to not just need a basic abstract understanding of the theory around Diffusion models, rather I needed to actually understand the theory, how it is typically implemented today in research and industry, so much so that I needed to find out the type of data structure to expect as output which would be a rather complex matrix. This entire process of learning was really challenging as I was a novice in the subject. But I managed to solve this by purchasing a course on Udemy regarding Theory and Implementation of Diffusion models

which helped me really better understand and write a V&V plan that actually made sense in the context of diffusion models.

Harrison Chiu: A bit of a challenge was figuring out just how detailed each test case needed to be. I kept wondering if I was getting too specific or not specific enough. To fix this, I made a quick checklist of essential edge cases based on the project requirements and ran it by the team. Getting their input reassured me that I was covering the right bases without going overboard.

Gurnoor Bal: The main problem was with some of the requirements that were a bit too general or vague. It was hard to nail down exactly what needed verifying. I decided to check in with our project supervisor to get a better handle on what was expected, and I left a few notes in the document to flag areas we might need to clarify later. That way, the process stayed clear without me having to guess on details.

Jared Paul: One thing that wasn't easy was verifying the interfaces for each module. Some of the connections needed more detail to line up with the rest of the design. To sort this out, I went back and added specific inputs and outputs for each module, which made sure everything matched up well with the Module Guide. It took a bit more time, but in the end, it made the design much clearer.

Ahmad Hamadi: The tricky part was making sure the tests weren't just covering easy scenarios. I wanted to be sure they were meaningful. I ran into a few issues with environment consistency, which caused random test failures. To fix this, I standardized the setup and added a few retries for tests that occasionally flaked. Then I went back to the requirements to make sure the tests were focused on the important functions. This way, the tests ended up being both reliable and relevant.

6.5 Question 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.

Several different and diverse skills are required in order to create the verification and validation plan for this Chest X-ray Diffusion Model Research project.

Software Specification Design
Our team will need a strong understanding of software specification design to grasp the project context and outline the required functions and modules. By defining each function's purpose, inputs, and edge cases, we can plan the implementation of functions accurately according to the specifications. This design phase will serve as the foundation for later development stages.

Unit Test Planning
Developing robust unit tests will be crucial. Our team must create comprehensive tests that

ensure each function achieves its intended purpose reliably. This will require us to account for all relevant edge cases to avoid misleading results and ensure the accuracy and completeness of our unit tests as a core skill for our V&V plan.

Software Design Principles
To maintain a flexible and maintainable codebase, we'll need to prioritize software design principles, particularly the SOLID principles. For example, we plan to structure modules to follow the Single Responsibility Principle, which will help ensure clarity and cohesion in the design, allowing future engineers to work with the code more effectively.

Diffusion Model Concepts and Implementation
Understanding both the theoretical and practical aspects of diffusion models will be essential for designing an accurate V&V plan. Our team will need knowledge of the inputs, expected outputs, and common data structures used in similar applications to develop thorough and effective tests.

6.6 Question 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?

1. **Software Specification Design**
   Revise existing papers on software specification design, for example material on MIS specifications, and work done by David Parnas Reflect on previous Mcmaster Software Engineering course on software design (i.e. 3A04)
2. **Unit Test Planning**
   Revise testing blogs produced by large tech companies such as Meta, Google and Amazon. These blogs devise the strategies these companies adopt in their creation of effective unit tests. Reflect on previous Mcmaster Software Engineering course, 3S03 Software Testing
3. **Software Design Principles**
   Revise SOLID principles via informational blogs such as DigitalOcean, Medium, and FreecodeCamp Implement personal projects that are small but correctly implement each of the design principles.
4. **Diffusion Model Concepts/Implementation**
   Utilize Introduction to Diffusion Models course on Udemy. This goes over theory and implementation in practice today. Read existing papers on experiments with diffusion models by other researchers that have similar applications

Now, each team member selected a particular method to pursue, and stated their justifications below:

Jared Paul: I'm going to revise more formal and professional software specification design practices by looking at existing papers, such as that from David Parnas. This is because I

recall being introduced to him in 2nd year, and found the content really straightforward and informative the, so I thought it would be able to get me up to speed quite quickly today.

Hamza Issa: I chose to revise existing engineering blogs from big tech companies like Amazon, as although there are papers and lots of theory out there for devising unit tests. I think that in designing and creating the best unit tests we can look to companies that are highly dependent on them, to find tests that are not just complete but not overly verbose.

Ahmad Hamadi: I plan on revising my SOLID principles using popular blogs like I've seen from FreeCode camp. I think this would be the simplest way to learn all the principles in a manner that is easy to digest and also easy to implement myself for when I want to experiment and practice.

Harrison Chiu: I am going to try and complete the Udemy course on Introduction to Diffusion models. I have some experience looking at research papers for Diffusion models, but they have been really difficult to understand due to the advanced mathematics. I think using a Udemy course would make it simpler to learn and easily transferable to our project

Gurnoor Bal: I am going to investigate existing research papers in the Diffusion Models space that have some similarity to our application. I think this would not only help me learn the required concepts, but also show me how to actually produce research and an application catered for this specific project. If i were to just watch the Udemy course, I'll understand the concepts, but I would still need to review papers to understand how to go about it for this particular project.