Reflection and Traceability Report on CXR

Team 27, Neuralyzers
Ayman Akhras
Nathan Luong
Patrick Zhou
Kelly Deng
Reza Jodeiri

1 Changes in Response to Feedback

1.1 SRS and Hazard Analysis

Here is the feedback we received on the SRS and Hazard Analysis documents, and the changes we made in response to that feedback.

Table 1: Feedback and Changes for SRS Documentation and Hazard Analysis

Feedback Source	**Feedback Item**	**Response**	**Issue**
TA Feedback	Formatting and Style	Mention figures in paragraphs and fix title	#125
TA Feedback	What not How(Abstract)	Improve constraints details as some constraints lack detail ("what")	#126
TA Feedback	Complete, Correct and Unambiguous	Template explanation (already addressed in previous versions)	#124
TA Feedback	Traceable Requirements	Fix referencing for section 5.2.	#123
TA Feedback	Document Content	Fix functional requirements	#122
Peer Review	Project Goals	Goal Statements Inconsistency with the rest of the doc	#55
Team Feedback	Document Content	Fix FR and NFR to align with the current scope of project	#201

Feedback Source	Feedback Item	Response	Issue
TA Feedback	Document Content	Fixed Citation in reference section	#202
Peer Review	What not How (Abstract)	Compatibility of DICOM image is not a priority of this project and adding support for legacy systems would require additional development, validation, and maintenance efforts.	#60
Peer Review	What not How (Abstract)	Privacy measures are already implicitly covered in SR1 and SR2	#57
Peer Review	Document organization	detailed theoretical models like ELBO optimization would shift the focus from specifying what the system must do to how certain algorithms work internally, which is beyond the typical scope of an SRS	#46

1.2 Design and Design Documentation

Here is the feedback we received on the Design and Design Documentation, and the changes we made in response to that feedback.

Feedback Source	Feedback Item	Response	Issue
TA Feedback	Document Content	Formalization	#191
TA Feedback	Document Content	Input represenation	#192
TA Feedback	Document Content	Specific Definition of JSON	#193
TA Feedback	Document Content	HTTP Design	#194

1.3 VnV Plan and Report

Here is the feedback we received on the VNV Plan and VNV Report, and the changes we made in response to that feedback.

Table 3: Feedback and Changes for VNV Plan

Feedback Source	**Feedback Item**	**Response**	**Issue**
TA Feedback	Nondynamic testing used as necessary	Improve Testing, add linting test (already part of the doc)	#196
TA Feedback	General Information	update objectives based on the scope of vnv plan, instead the scope of the entire project	#195
Team Feedback	System Tests for Functional / Nonfunctional Requirements are specific	Update based on team feedback, remove some fr and nfr test items as they are no longer in the scope of the project	#203
Peer review	Unit Test Description	Updated unit test description and adding traces	#81
Peer review	Section 4.1.6 - Structured Report Generation Tests	We decided not to address this suggestion because structured report generation testing is focused on validating correct outputs under normal conditions. Failure handling will be tested separately during full system reliability testing to better align with project priorities and scope.	#83
Peer review	Section 4 (System Tests)	Removing user log in the test step as it is already stated in the initial state.	#84

Feedback Source	Feedback Item	Response	Issue
Peer review	Section 4.3 - Security Tests	We chose not to add session timeout and concurrent login tests because they are outside the core security scope for this release. These aspects will be considered in future system hardening phase	#85
Peer review	Section 5.2.6 - Periodic Health Checks	Add specific criteria for measuring Periodic Health Checks	#86

2 Challenge Level and Extras

2.1 Challenge Level

The challenge level of this project is classified as advanced, reflecting the complexity of integrating AI for diagnostic purposes in medical imaging. The project required implementing sophisticated machine learning models for chest X-ray analysis, ensuring robust data handling with privacy compliance, and developing an intuitive user interface for healthcare professionals.
Key technical challenges included:

- Developing and training AI models capable of detecting diseases in chest X-rays with high accuracy
- Implementing comparative analysis capabilities to track disease progression between scans
- Demonstration of lung segmentaion using DETR model and visulzing them on the lungs
- Ensuring secure storage and handling of sensitive medical data through cloud services
- Building a system architecture that supports regular model updates without service disruption

Our team leveraged our knowledge of tensors, linear algebra, machine learning, and image processing to address these challenges effectively.

2.2 Extras

Our project included the following extras:

- **Research Report:** We developed a research paper proposing a unified Transformer-based framework for comprehensive chest X-ray analysis with an emphasis on anatomical localization. Our approach utilized a Detection Transformer (DETR) to identify anatomical regions and support downstream tasks including disease classification and progression monitoring.

- **Norman Principle Report:** We created a detailed evaluation of our system's user interface based on Don Norman's seven usability principles. This report assessed how intuitive, user-friendly, and efficient our interface is from multiple perspectives, providing valuable insights for further UI refinements.

3 Design Iteration (LO11 (PrototypeIterat)

Our project underwent significant design iterations throughout its development lifecycle, with each phase building upon lessons learned from previous stages and incorporating feeback from our instructors and stakeholders.

3.1 POC Stage: Disease Classification and Initial Feedback

Our proof-of-concept (POC) focused primarily on establishing baseline disease classification capabilities using the TorchXRayVision library with its pre-trained DenseNet model. This approach provided several immediate advantages:

- The model was pre-trained on multiple chest X-ray datasets including NIH ChestX-ray14 (112,120 images), MIMIC-CXR (377,110 images), CheXpert (223,648 images), and PadChest (160,868 images)

- It could detect 18 different thoracic pathologies with reasonable accuracy

- The DenseNet architecture allowed for efficient feature extraction and reuse through dense connections

During this initial implementation phase, we developed a basic interface displaying classification results and confidence scores for individual X-ray images. While functionally capable of processing images and providing probability scores for various conditions, the interface lacked clinical workflow integration and user-specific design considerations.
Upon reviewing our POC implementation, Dr. Moradi provided pivotal feedback that significantly shaped our subsequent development approach:

- Our interface lacked a coherent user flow that aligned with clinical workflows

- The system needed clearer focus on a specific stakeholder group instead of trying to accommodate multiple user type

- He recommended we choose between designing primarily for physicians or patients, as these groups have fundamentally different needs and usage patterns

Dr. Moradi emphasized that our visualization of results was too technical and not optimized for clinical decision-making. He noted that while our focus on model accuracy was important, it needed to be balanced with usability and seamless integration into existing clinical workflows. This feedback highlighted a critical insight: technical capability alone would not ensure adoption in healthcare settings without thoughtful design for the end user's context and needs.

3.2 Rev 0: Redesign of Interface and Integrating Cloud Services

Dr. Moradi's feedback prompted a complete redesign of our application's flow and architecture for Revision 0. We made the strategic decision to focus exclusively on physicians as our primary users, recognizing that they are the final approvers of AI-assisted diagnoses and possess the medical expertise necessary to interpret results correctly. This decision also acknowledged the potential risks of patients directly accessing complex diagnostic information without professional guidance.
The redesigned physician-centric interface featured:

- A patient directory organized by disease severity to help prioritize cases
- Comprehensive patient-specific views showing clinical history, previous records, and prescription plans in a unified dashboard
- Backend migration to cloud services for improved scalability and accessibility
- Secure data storage and retrieval mechanisms compliant with healthcare data protection requirements

We structured the interface to follow a logical clinical workflow, starting with patient selection and proceeding through examination, diagnosis, and treatment planning. By incorporating design principles from established medical software, we aimed to create an environment that felt familiar and intuitive to healthcare professionals. When Dr. Spencer Smith reviewed this iteration, he identified several opportunities for further improvement:

- He observed that our interface still required excessive manual data entry from physicians
- He suggested automating the capture and input of patient history information where possible
- He recommended streamlining the prescription entry process to improve physician adoption

Dr. Smith emphasized the importance of prioritizing physicians' time efficiency through features like auto-complete and standardized templates. His feedback underscored a key insight: reducing cognitive load and administrative burden on physicians would allow them to focus more on patient care and less on system interaction, significantly increasing the likelihood of adoption in clinical settings.

3.3 Rev 1: DETR and Linear Transformer for Enhanced Disease Classification

For our final revision, we addressed Dr. Smith's feedback while simultaneously advancing our technical approach through more sophisticated AI models. We implemented a Detection Transformer (DETR) model trained on the Imagenome dataset to identify 12 distinct anatomical regions in chest X-rays. This approach enabled precise localization of findings rather than just whole-image classification, providing significantly more clinically relevant information.
The technical advancements in this phase included:

- Training a DETR model to identify specific anatomical regions within chest X-rays
- Implementing a Linear Transformer model that leveraged features extracted from the DETR model for disease classification
- Adding natural language processing capabilities to convert structured findings into human-readable reports

These technical improvements were complemented by user interface enhancements directly addressing the feedback we received. We added automated data entry features, including templated notes and prescription suggestions based on detected conditions. The system could now highlight specific regions of concern within X-rays while simultaneously reducing the documentation burden on physicians.

3.4 Final Implementation

Our final system successfully balanced technical sophistication with practical clinical utility. The evolution from a basic classification tool to an integrated clinical decision support system reflected our iterative approach and responsiveness to stakeholder feedback.
The completed system provides physicians with:

- Precise anatomical localization of findings through advanced transformer models
- Clear visual highlighting of affected regions
- Automated report generation to reduce documentation time
- An intuitive workflow aligned with clinical practice

- Secure, scalable cloud infrastructure

This iterative development process demonstrated the importance of balancing technical innovation with user-centered design. By incorporating feedback at each stage, we were able to create a solution that not only leveraged cutting-edge AI technology but also integrated seamlessly into clinical workflows. The progression from our initial focus on model accuracy to our final emphasis on physician workflow integration illustrates how our understanding of the problem space matured throughout the project lifecycle.

4 Design Decisions (LO12)

Our project's design evolved significantly from initial conception to final implementation, guided by a series of deliberate decisions shaped by limitations, assumptions, and constraints we encountered. This section explains our key design decisions and the factors that influenced them.

4.1 Limitations and Their Influence

Several technical and practical limitations significantly shaped our design approach:

- **Limited Clinical Expertise:** As software engineering students, our team lacked formal medical training. This limitation influenced our decision to:
    - Rely heavily on expert feedback from Dr. Moradi.
    - Implement clear confidence metrics for all AI predictions
    - Design the system as a diagnostic aid rather than a replacement for clinical judgment
- **Computational Resource Constraints:** Although we had access to McMaster's GPU servers, we faced limitations in training time and resource allocation, which led us to:
    - Utilize transfer learning with pre-trained models (TorchXRayVision in POC, DETR in Rev 1)
    - Implement efficient data pipelines to optimize training
    - Design am AWS cloud-based architecture that could scale according to available resources
    - Prioritize model architectures that balanced accuracy with computational efficiency
- **Dataset Limitations:** The available chest X-ray datasets had inherent limitations in terms of diversity, labeling consistency, and coverage of rare conditions. These limitations led us to:

    - Combine multiple datasets (NIH, MIMIC-CXR, CheXpert, PadChest) for broader coverage
    - Implement data augmentation to improve model generalization
    - Focus on the most common and well-represented conditions for initial deployment
    - Design a modular architecture that could incorporate new datasets as they become available

4.2 Assumptions and Their Impact

Our design was guided by several key assumptions that shaped both our approach and implementation:

- **Physicians as Primary Users:** We assumed that physicians would be the primary users of our system rather than patients. This assumption, reinforced by Dr. Moradi's feedback, fundamentally shaped our interface design by:
    - Incorporating medical terminology and visualizations familiar to healthcare professionals
    - Organizing the workflow around clinical decision-making processes
    - Prioritizing accuracy and detail over simplified explanations
    - Including features specifically for clinical documentation and follow-up
- **Integration with Existing Workflows:** We assumed that successful adoption would require seamless integration with existing clinical workflows rather than disrupting them. This led us to:
    - Design the UI to mimic familiar medical software interfaces
    - Implement a patient directory similar to electronic health record systems
    - Create automated documentation features to reduce administrative burden
    - Provide export functionality compatible with common medical data formats
- **Value of Region-Specific Analysis:** We assumed that localized disease detection would provide more clinical value than whole-image classification. This assumption guided our progression from DenseNet to DETR models and led to:
    - Implementation of anatomical region detection capabilities
    - Development of region-specific highlighting in the user interface
    - Creation of more detailed and specific diagnostic reports
    - Design of a comparison feature for tracking changes in specific regions over time

4.3 Constraints and Their Effects

Several external and internal constraints bounded our design possibilities:

- **Time Constraints:** The academic timeline of the capstone project imposed strict deadlines that influenced our decisions to:
    - Prioritize core functionalities (disease detection, region localization, reporting)
    - Defer some features (multilingual support, prescription integration in ML model, advanced security features) to future iterations
    - Choose established technologies and frameworks over experimental ones
    - Implement an iterative development approach with clearly defined milestones
- **Security and Privacy Requirements:** Healthcare data requires strict security measures, constraining our design in several ways:
    - Implementation of AES-256 encryption (included in AWS incognito) for all patient data
    - Design of secure authentication and authorization mechanisms
    - Creation of detailed audit logs for all system activities
    - Selection of compliant cloud infrastructure for deployment
- **Usability Requirements:** The need for high usability in clinical settings constrained our design choices by requiring:
    - Responsive interface with minimal loading times
    - Intuitive navigation requiring minimal training
    - Clear visualization of results without information overload
    - Consistent layout and interaction patterns throughout the application

4.4 Traceability of Changes from Rev 0 to Rev 1

The following table provides a comprehensive list of all changes made between Revision 0 and Revision 1 of our system, organized by component:

Table 4: Detailed Changes Between Rev 0 and Rev 1

Component	**Rev 0 Implementation**	**Rev 1 Changes**
AI Model Architecture	DenseNet-based classification using TorchXRayVision	Implemented DETR model for anatomical region detection and Linear Transformer for disease classification
Dataset Utilization	Used pre-trained models on standard datasets	Added training on Imagenome dataset to identify 12 specific anatomical regions
Disease Localization	Whole-image classification with no region-specific information	Added precise localization of findings with region-specific highlighting
User Interface - Patient Directory	Basic list of patients with minimal filtering	Enhanced directory with prioritization based on disease severity and status
User Interface - Patient View	Manual data entry for patient history and symptoms	Added automated data entry features with templates and suggestions
Reporting Capabilities	Basic probability scores for detected conditions	Implemented natural language processing to generate human-readable reports
Image Visualization	Basic image display with global classification results	Added region-specific highlighting and interactive visualization of affected areas
Backend Infrastructure	Local deployment with limited scalability	Migrated to cloud-based architecture with improved scalability and accessibility
Data Storage	Basic database with limited security features	Implemented comprehensive security measures and backup systems
Workflow Integration	Generic interface not optimized for clinical use	Redesigned workflow to match clinical decision-making processes
Performance Optimization	Base implementation with standard processing times	Optimized image processing pipeline for faster analysis and response

4.5 Rationale for Key Design Changes

The evolution from Rev 0 to Rev 1 was guided by specific feedback and insights gained during development:

- **Shift to DETR Architecture:** The transition from DenseNet to DETR was motivated by the need for more precise localization of findings, which physicians indicated would significantly enhance clinical utility. This architectural change enabled region-specific analysis rather than just whole-image classification.
- **Automated Data Entry:** Dr. Smith's feedback about excessive manual entry directly led to the implementation of automated features like templates and suggestions, aimed at reducing physician workload and improving adoption rates.
- **Cloud Migration:** The decision to move to a cloud-based architecture was driven by scalability needs identified during performance testing of Rev 0, which showed limitations in handling multiple concurrent users.
- **Enhanced Visualization:** The improved visualization capabilities in Rev 1 directly addressed Dr. Moradi's concern that our initial approach was too technical and not optimized for clinical decision-making.

These design decisions reflect our iterative approach to development, incorporating stakeholder feedback and addressing limitations to create a system that balances technical capabilities with practical clinical utility. By carefully considering the constraints, assumptions, and limitations we faced, we were able to develop a solution that effectively meets the needs of healthcare professionals while operating within our available resources.

5 Economic Considerations (LO23)

5.1 Market Demand and Opportunity

The market for AI-driven medical imaging analysis is growing rapidly due to increasing healthcare digitization and the global shortage of radiologists. The demand for fast, accurate, and cost-effective diagnostic tools makes this product highly viable. Potential users include hospitals, clinics, telemedicine providers, and government healthcare initiatives. Additionally, developing countries with limited access to radiologists represent a significant market where AI-powered solutions could improve diagnostic capabilities.

5.2 Marketing Strategy

Marketing would involve direct outreach to hospitals and healthcare providers, showcasing the efficiency and accuracy of the model through clinical trials and case studies. Partnerships with electronic health record (EHR) providers could

facilitate integration into existing workflows. Furthermore, regulatory approvals such as FDA or Health Canada certification would enhance credibility. However, our team plans more on leveraging via government healthcare programs and partnerships.

5.3 Production Cost Estimate

The cost of production includes:

- **Cloud Services (AWS ECS):** The hosting and management of the backend system will be powered by AWS Elastic Container Service. This service will handle containerized microservices, ensuring scalability. Costs for ECS typically range from x to x dollar per month, depending on traffic, load, and the number of services running.
- **GPU/Server Cost:** To train the AI model efficiently, a dedicated GPU or server instance is required for training and inference tasks. For optimal performance, services like AWS EC2 GPU instances would be suitable, which cost around 3 to 5 dollars per hour, depending on the selected instance type. In our case, using the Mcmaster GPU server, there was no cost for training and inference, but we estimate a cost of 500 to 1,000 dollars per month if we did not have access to the mcmaster GPU server. Note our product is the application not the computer or server it runs on.
- **Cost of API Services:** The integration of ChatGPT for natural language processing tasks, such as generating reports or patient interaction, incurs costs based on the number of API calls. This will vary depending on usage but can be estimated at a few cents per API call, estimating the monthly cost to be the amount of patients divded by 100.

5.4 Pricing Model

Since the application is enterprise-focussed, there are two models to compare when pricing the users:

- **Charging for number of licenses:** Each hospital pay a monthly fee for the number of licenses they have purchase. Each license represent a radiologist, and no matter the amount of data a radiologist produce, they will get charged the same as others.
  - **Advantages:** This model generate a predictable cashflow and monthly recurring revenue. In addition, it's very straight-forward, making it easy for hospitals to manage and scale licensing cost.
  - **Disadvantages:** Experienced radiologists might have significantly more patients than others users (i.e: doctors in training, resident). This creates huge friction for hospitals to buy licenses for specialists

with low expected patient count. Additionally, usage prediction becomes tricky for Lung Vision AI since any license can generate an infinite amount of data in theory.

- **Charging for number of patients:** Each hospitals pay a monthly fee for the number of patients they service.
  - **Advantages:** This model encourage a pay-as-you-go mentality, making usage and cost prediction for Lung Vision AI a lot simpler. Additionally, this model is suitable for private clinics where patient growth is not too drastic.
  - **Disadvantages:** This pricing model is unfavorable for big hospitals, since forecasting expected patients for a given month is difficult, making costs management unpredictable.

To balance long-term user satisfaction, affordability, and profitability, our has decided to go with the **charging for licenses model**, with an upper limit of how many patient are allowed to be created per license. If doctors would like to service more patients using their current license, a different pricing schema will be applied per user.

5.5 Break-even Analysis

The cost the project per month can be broken down as follow:

Category	Cost	Cost/Month	Description
Business Operation	**Domain Name**	$1.23	Cost associated with purchasing and maintaining the domain name for the application.
Hosting	**Database**	$13.00	NoSQL database (AWS DynamoDB) and blob storage expenses (AWS S3).
	Load Balancing	$17.60	Costs of load balancer (AWS ALB) to ensure high availability and scalability of the application.
	Computing	$5.00	Cloud container running cost (AWS Elastic Container Service).
	Other	$33.50	Any additional variable costs, such as OpenAI usage or additional AWS services (AWS ECR, IAM).
	Total Cost	$70.33	

5.6 Conclusion

Currently, our database contains roughly 20 radiologists, and 100 active patients. By simple calculations

- Cost per patient = 70.33/ 100 = 0.7033 dollars
- Cost per license = 70.33/ 20 = 14.066 dollars

Assuming that there are no other costs of this project (labour, transportation, etc.) the break price per license will be 14.066 dollars, with a max limit of 5 patient per license. For every new patient doctors would like to serve, an additional 0.7033 dollars per user will be charged.

6 Reflection on Project Management (LO24)

6.1 How Does Your Project Management Compare to Your Development Plan

- We closely followed our scheduled meeting plan, holding weekly stand-ups and additional ad hoc meetings during key development phases (Rev0, Rev1, Final Demonstration). These meetings helped us stay aligned and adapt quickly when challenges arose.

- Our primary communication tools (Discord and GitHub) were used effectively for both asynchronous updates and collaborative debugging. We maintained clear version control and documentation throughout development.

- We leveraged GitHub's project management features extensively, creating a structured kanban board with columns for Backlog, To Do, In Progress, and Done. Each task was represented as an issue with detailed descriptions, acceptance criteria, and assigned team members. This allowed us to visualize our workflow and track progress in real-time.

- Roles and responsibilities as outlined in our development plan were largely adhered to. Each member focused on their assigned modules (e.g., frontend UI, model training, backend integration), and we successfully coordinated via our Git branching workflow and task tracking system (GitHub issues).

- We used the technologies originally proposed in our plan, including Python, PyTorch, React JS. The only deviation was adopting a Detection Transformer model (DETR) earlier than planned due to its proven effectiveness in localization tasks during our initial model experimentation.

6.2 What Went Well?

- Discord and GitHub Discussions helped streamline both quick check-ins and in-depth technical discussions. Clear communication reduced misunderstandings and sped up decision-making.

- Each member had a clearly defined role (e.g., frontend, model dev, backend), which minimized overlap and improved accountability. Tasks were assigned and tracked using GitHub Issues and Project Boards.

- Our GitHub workflow was particularly effective—we created issues for each feature, bug, or enhancement with specific labels (e.g., "frontend", "model", "urgent"), milestones tied to sprint goals, and assigned team members. Pull requests were linked to issues, ensuring traceability between code changes and requirements.

- We organized work into two-week sprints with clear deliverables set at the beginning of each sprint. GitHub milestones helped us monitor sprint progress, with burndown charts showing the completion rate of issues. This allowed us to adjust resources or scope when necessary to meet deadlines.

- Weekly meetings kept everyone aligned, and sprint planning ensured we had clear short-term goals. Checkpoints during development helped catch issues early.

- Our planned technologies (PyTorch, Python, React JS) integrated well with each other. Version control via GitHub was used effectively for collaboration and rollback when needed.

- Independent module development (e.g., AI model, UI, API) allowed parallel progress without bottlenecks. CI/CD setup helped with automated testing and smooth deployment.

- GitHub's code review system was instrumental in maintaining code quality, with required reviews before merging any significant changes. This peer review process helped catch bugs early and ensured knowledge sharing across the team.

6.3 What Went Wrong?

- Some tasks, particularly around localized disease progression tracking, took significantly longer than expected. This affected sprint timelines and required scope adjustments mid-way.

- Although GitHub issues were created for most tasks, we occasionally had "shadow work" that wasn't properly tracked in the system. This made it difficult to accurately assess workload and progress at times.

- Our original plan didn't account for debugging, model tuning iterations, or integration testing delays. This compressed our final testing and documentation phases.

- We sometimes created issues that were too large or ambiguous, making them difficult to complete within a single sprint. This led to issues spanning multiple sprints and complicated progress tracking.

- Early-stage code and design decisions weren't always documented consistently, leading to rework during integration. We improved this mid-project by enforcing clearer commit messages and README updates.

- Adopting DETR introduced additional complexity (e.g., custom training routines, evaluation metrics), which required more experimentation and adaptation than anticipated.

- While GitHub provided excellent tracking tools, we didn't fully utilize all features like estimate points or time tracking, which would have helped with more accurate sprint planning and resource allocation.

6.4 What Would you Do Differently Next Time?

- Allocate extra time in the schedule for unexpected delays, model tuning, and integration/debugging.

- Build a rough end-to-end prototype early to identify integration pain points sooner, even before full model training is complete.

- Maintain consistent documentation practices from the start, including API specs, model details, and configuration notes, to reduce confusion later.

- Create smaller, more granular GitHub issues with clearer acceptance criteria and time estimates. This would improve our ability to track progress and identify bottlenecks more quickly.

- Implement a more formal GitHub workflow with standardized branch naming conventions, issue templates, and automated CI/CD pipelines from the beginning. This would reduce overhead and ensure consistent practices across the team.

- Involve test users (or at least team cross-reviews) earlier in the UI development cycle to catch usability issues before final sprint.

- Use GitHub's milestone features more effectively to track progress against larger goals and provide better visibility into overall project status.

- Identify high-risk components (e.g., novel model architectures) up front and create extra plans or fallback options.

- Schedule regular backlog refinement sessions to ensure issues remain relevant and appropriately sized as the project evolves.

7 Reflection on Capstone

7.1 Which Courses Were Relevant

- **4ML3 Machine Learning and AI:** The principles and techniques from this course provided a foundation for understanding how to train and deploy machine learning models, especially the convolutional neural networks (CNNs) used for image classification in medical X-rays. Key topics such as model validation, overfitting, and the use of pre-trained models were directly applicable to the AI model development for disease prediction.

- **4HC3 Human Computer Interfaces:** The knowledge gained from this course on Norman's principles of design was crucial in designing an intuitive and effective user interface for the capstone project. Understanding the user experience was essential for ensuring that the system is accessible and user-friendly, especially for medical professionals who will interact with the system.

- **4A03 Ethics:** This course allowed our group to have some former background in the ethical implications of using AI in healthcare, especially regarding privacy, consent, and ensuring that the model works equitably across different patient demographics. This understanding guided the way our team handled data and made design decisions for this project.

- **4X03 Scientific Computation:** This course helped the group understand numerical methods and how to implement efficient algorithms for scientific computing. It was particularly useful when optimizing performance for processing large-scale image datasets, such as chest X-rays, and for ensuring that the system was computationally feasible and efficient.

7.2 Knowledge/Skills Outside of Courses

- **Deep Learning Frameworks (PyTorch):** The team collectively enhanced its understanding of PyTorch, a deep learning framework, to implement the AI model for chest X-ray analysis. This involved collaboratively building convolutional neural networks (CNNs), fine-tuning pre-trained models, and developing robust image data pipelines for preprocessing and augmenting the X-ray images.

- **Regulatory Compliance in Healthcare (HIPAA):** Since the project involves medical data and aims to have real-world applications in healthcare, the team researched and gained knowledge about healthcare regulations like Health Insurance Portability and Accountability Act for AI-based medical tools. This included understanding privacy concerns and compliance measures for handling patient data safely and legally.

- **API Integration and Deployment:** To integrate the machine learning model into a production environment, the team learned how to set

up RESTful APIs and integrate them with the frontend. This included working with cloud services such as AWS to deploy the system and ensure scalability for real-world use. This was crucial for making the AI model accessible to healthcare professionals and ensuring that it could handle multiple requests simultaneously.

- **Data Annotation and Augmentation for Medical Imaging:** The team acquired knowledge on handling medical image datasets, including proper annotation of X-ray images for training purposes and applying image augmentation techniques to improve model robustness. This was particularly important as public datasets for chest X-rays can be limited and require preprocessing for real-world applications.

- **Performance Monitoring and Model Optimization:** To ensure that the AI system performed well in production, the team explored model optimization techniques, such as hyperparameter tuning, and utilized performance monitoring tools to track the model's real-world effectiveness in detecting lung diseases. This included setting up logging and feedback mechanisms to ensure continuous learning and improvement.