Medical Imaging and AI: Breast Cancer Detection

Team Name: **B**reast **C**ancer **D**etection

SEDE Studios Autumn 2025

(Scientific Foresight (STOA), 2020)

Product Owner: Steve Ling

Team Members:

Gajan Sutherachelvan [24553760]

Tanish Satre [24632541]

Eunice Anne Javier [13539705]

Aung Pyay [14390759]

Batool Al Salman [24452310]

Deepti Mallampalli [13614829]

Table of Contents:

Table of Figures:

Table of Tables:

1. Introduction

1.1 Defining the Problem

This report presents the development of an AI-powered medical imaging product designed to improve early detection of breast cancer. Breast cancer is one of the most diagnosed cancers in Australia, and early diagnosis plays a critical role in improving survival outcomes. However, in many regional or resource-limited settings, patients face delays due to a lack of timely access to radiologists or diagnostic specialists. Globally, many individuals face financial and geographic barriers, restricting their access to accurate and timely diagnostic services. Manual analysis of breast cancer scans is shown to be labour-intensive, subject to delays, and reliant on specialist interpretations, particularly in already overburdened healthcare settings.

The proposed solution addresses this gap by developing an accessible platform, allowing healthcare professionals to upload and analyse medical scans using AI. By automating key parts of the diagnostic process, the product aims to support medical professionals with a more consistent, accurate, and faster image assessment, potentially helping to accelerate patient care timelines. AI offers a flexible way to standardise and accelerate this process. This product integrates AI-based image segmentation into a mobile and web application, enabling efficient analysis across both clinical and remote environments. The goal is to enhance diagnostic accessibility and support early intervention of breast cancer.

1.2 Aims and Objectives

The project aims to develop an AI model that assists medical professionals by automatically identifying and segmenting the location of potential breast cancer tissue from medical images.

To achieve this, the objectives are:

- Develop a U-net method-based CNN model to segment cancerous regions in ultrasound or x-ray breast images
- Integrate the model into a smartphone and web interface for ease of use by stakeholders
- Ensure accurate segmentation to support breast cancer diagnostics

1.3 Stakeholders

Key stakeholders include medical professionals such as radiologists, oncologists, and diagnostic technicians, who benefit from reduced workload and improved diagnostic accuracy. Patients who are in rural and remote areas gain earlier and consistent access to breast cancer screening, helping to close the gaps in healthcare. Healthcare providers, such as hospitals and clinics, are also shown to be key stakeholders as the product aims to enhance service efficiency while helping to reduce operational expenses. Additionally, AI-focused research and development centres play an important role in advancing the integration of machine learning in healthcare. The product's development was guided by stakeholder requirements, prioritising accessibility, accuracy, and speed. Feedback from the product owner helped ensure that the technical development of the product stays aligned with real-world applications and practical implementation.

---

2. Background and Existing Solutions

2.1 Context and Motivation

Medical imaging currently plays a central role in diagnosing a wide range of conditions, including breast cancer, which affects approximately one in seven Australian women. Despite significant advances in imaging technology, delays in diagnosis remain a concern, particularly in rural and remote areas where access to specialists is limited. This has led to growing interest in using artificial intelligence (AI) to support clinical workflows and reduce reliance on manual interpretation alone.

AI-based diagnostic tools offer the potential to enhance consistency, improve accuracy, and reduce workload for healthcare professionals. Their ability to process high volumes of medical data quickly makes them well-suited for settings where timely analysis is critical. However, most current tools are either highly resource-intensive, limited to research contexts, or difficult to deploy at scale in low-resource environments.

2.2 Review of Existing Solutions

Several AI-powered tools, such as Google Health's mammography AI, Zebra Medical Vision, and IBM Watson Health, have shown strong performance in detecting abnormalities in medical images. These systems use deep learning models to support clinical decision-making and have demonstrated accuracy comparable to human specialists.

However, current solutions are often limited by their reliance on cloud infrastructure, high computational requirements, and complex interfaces designed for expert users. Most are not optimised for mobile use and lack adaptability for offline or low-resource environments. Licensing costs and technical barriers further restrict access in public and rural healthcare settings.

These limitations highlight a clear market gap: the need for accurate, affordable, and user-friendly diagnostic tools that are accessible across a wider range of devices and settings, including regional and low-connectivity environments.

3. Product Overview

3.1 Concept

The product is a cross-platform medical imaging application that uses artificial intelligence to assist in the early detection of breast cancer. It is designed to operate on both web browsers and smartphones, enabling broad accessibility for healthcare professionals and patients alike. The core functionality is centred around an image processing pipeline that begins with the user uploading a breast scan, such as an ultrasound or x-ray image, and concludes with the system generating a visual output that highlights potentially cancerous regions.

Upon uploading an image, the system runs a preprocessing phase that ensures the file is in a usable format and standardises the input for the AI model. This includes resizing, greyscale conversion (if required), and normalisation of pixel values to prepare the image for analysis. The backend is powered by a U-Net convolutional neural network, a well-established architecture for image segmentation tasks in medical contexts. The model performs pixel-level classification to isolate tumour-like structures, outputting a segmentation mask that overlays the original image.

The overlay serves as a visual diagnostic aid. Regions with high probability of abnormality are marked, allowing users to assess the result intuitively without requiring technical interpretation. In addition to the visual output, the system also generates a confidence score, offering transparency around the model's prediction. For further use, results can optionally be forwarded to a healthcare provider via an integrated communication option.

3.2 Key Features and Technical Design

The frontend of the application is built using React and Tailwind CSS, designed to be responsive and user-friendly across various screen sizes. The backend, developed in Python using Flask, handles image processing and model inference. The AI model itself was trained on publicly available medical imaging datasets, specifically curated for breast cancer research. TensorFlow and TensorFlow Lite are used to support both standard and mobile-optimised versions of the model, ensuring fast inference even on devices with limited processing power.

Key features include:

- **Drag-and-drop or mobile upload support** for ultrasound and x-ray images in formats such as PNG, JPG, and DICOM.
- **Real-time AI-based segmentation** using a trained U-Net model.
- **Overlay output** for visual clarity, with adjustable sensitivity options to balance false positives and false negatives.
- **Cross-platform deployment** through web and mobile integration.
- **Optional forwarding feature**, enabling users to share results with healthcare providers.

The interface is deliberately kept minimal, with large, clearly labelled buttons, tooltips, and progress indicators to guide the user through the process in three steps or fewer: upload, analyse, and review.

3.3 Intended Use and Practical Application

The product is intended for use in a range of contexts — from hospitals and diagnostic centres to general practice clinics and even patient homes. A general practitioner in a rural setting can quickly upload a scan for immediate analysis, providing faster feedback to patients without waiting for a specialist report. Similarly, a patient who has

undergone a breast scan can use the tool at home to conduct a preliminary check while awaiting further consultation.

In low-resource environments, the product provides a practical alternative to expensive diagnostic systems, reducing dependency on centralised infrastructure. Community healthcare workers can also use the mobile version during outreach programs to conduct on-the-spot screening and refer patients who require further care.

The application is not intended to replace clinical diagnosis but to function as a decision support tool that enhances speed, consistency, and accessibility in the diagnostic process.

---

4. Requirement Specification

4.1 Functional Requirements

The system must allow users to upload breast medical images, process them through an AI model, and receive a clear, annotated output. Key requirements include:

- Support for PNG, JPG, and DICOM image formats
- AI-based segmentation and classification of tumour regions
- Visual overlay of detected areas with accompanying confidence scores
- Adjustable sensitivity for model predictions
- Ability to download or forward results for follow-up

4.2 Usability Requirements

To ensure broad accessibility, the system must:

- Present a simple, step-by-step interface requiring no technical knowledge
- Operate on both desktop and mobile devices
- Complete upload and analysis in three steps or fewer
- Provide visual feedback during processing (e.g. loading indicators)
- Be navigable on low-bandwidth or lower-spec devices

4.3 Technical Requirements

The application architecture comprises a React frontend and a Flask + TensorFlow backend. Key technical needs include:

- TensorFlow and TensorFlow Lite compatibility for scalable deployment
- API integration for frontend–backend communication
- Image preprocessing pipeline to normalise input before inference
- Fast processing time (~5 seconds per image) on modern consumer hardware
- Compatibility with current major web browsers

4.4 Dependencies

The system's functionality relies on the following external components and frameworks:

- **Software Frameworks**: React (frontend), Flask (backend), TensorFlow for model inference
- **Hardware Resources**: GPU-enabled environments (e.g. Google Colab) used for training the U-Net model
- **Libraries and Tools**: Pandas, NumPy, Scikit-learn for data handling; Tailwind CSS for interface styling

- **Data**: Publicly available breast imaging datasets for model training
- **Browser Compatibility**: Functionality requires support for JavaScript ES6 and modern web APIs

Regulatory dependencies include adherence to data privacy frameworks such as **GDPR**, ensuring that medical image data is processed and stored in compliance with legal standards.

4.5 Assumptions

The system was developed with the assumption that users would have access to internet-enabled devices, although offline use was later supported through local deployment. This shaped the decision to keep the model lightweight and browser-based. It was also assumed that users operate in regions without strict AI regulatory barriers, allowing the team to focus on technical performance rather than certification. Basic digital literacy and access to standard image formats were presumed, influencing the design of a minimal, guided interface. Testing was limited to the devices available to the team, meaning full cross-platform compatibility could not be guaranteed.

4.6 Constraints

Development was constrained by limited access to hardware and browser types, restricting full compatibility testing. The React framework used for the frontend is not supported on some older browsers, potentially affecting interface stability. Hardware limitations on low-end devices impacted performance, influencing the decision to streamline UI elements. Regulatory compliance was scoped only to general data protection (e.g. GDPR), as the product is not yet intended for certified clinical use. Additionally, UI personalisation features were not implemented due to time and development priorities.

4.7 Deliverables

The following components constitute the deliverables for the project:

- Web and mobile-compatible user interface with upload and analysis functionality
- Trained U-Net segmentation model for tumour detection
- Integrated system architecture linking frontend, backend, and AI model
- Functioning API enabling communication between interface and inference engine

5. Development Process

5.1 Team Composition and Roles

Our team consisted of six members with varied academic backgrounds and technical strengths. A brief breakdown of the individuals within the team is provided in the table below.

Table 1 Team Composition Table

Name	Major	Studio Level	Experience
Gajan	Data	Professional A	• Python • Machine Learning • Databases • Project Management
Tanish	Data Science	Applications A	• Python • PyTorch • Tensorflow
Eunice	Electrical	Professional A	• Python • Presentation skills • Team coordination skills • Project management
Aung	Electrical and Electronic	Fundamentals B	• Python • Signal processing • CNN • Presentation and communication skills • Project management • Report writing
Batool	Data	Professional B	• Report writing • JavaScript • HTML • CSS • VScode • Python • React
Deepti	Data	Fundamentals A	• Python • CNNs

5.2 Development Methodology

The team adopted an **iterative development approach**, combining aspects of agile methodology with weekly milestone-based planning. This allowed for flexibility in responding to challenges while ensuring steady progress across the design, implementation, and testing phases.

Development was divided into weekly sprints, each focusing on specific deliverables such as model prototyping, frontend builds, or integration testing. Regular team meetings and informal check-ins were used to assess progress, allocate new tasks, and resolve blockers. Progress tracking was managed via Trello, with tasks assigned to individuals or sub-teams based on current priorities.

Early stages focused on research, dataset preparation, and low-fidelity interface prototypes. Once the core model and frontend were stable, integration and validation became the primary focus. Feedback from internal testing and roadshow presentations informed improvements in usability, accuracy, and performance.

This approach allowed the team to remain adaptive while meeting critical deadlines and balancing individual contributions across technical areas.

5.3 Development Timeline and Milestones

Project development was structured around a 12-week timeline, aligned with key studio checkpoints and internal team goals. The work was divided into five phases: planning, design, implementation, integration, and validation. Progress was tracked using several tools, including a Gantt chart, Microsoft Teams and milestone logs.

The Gantt chart below outlines development periods and their timing across the semester:

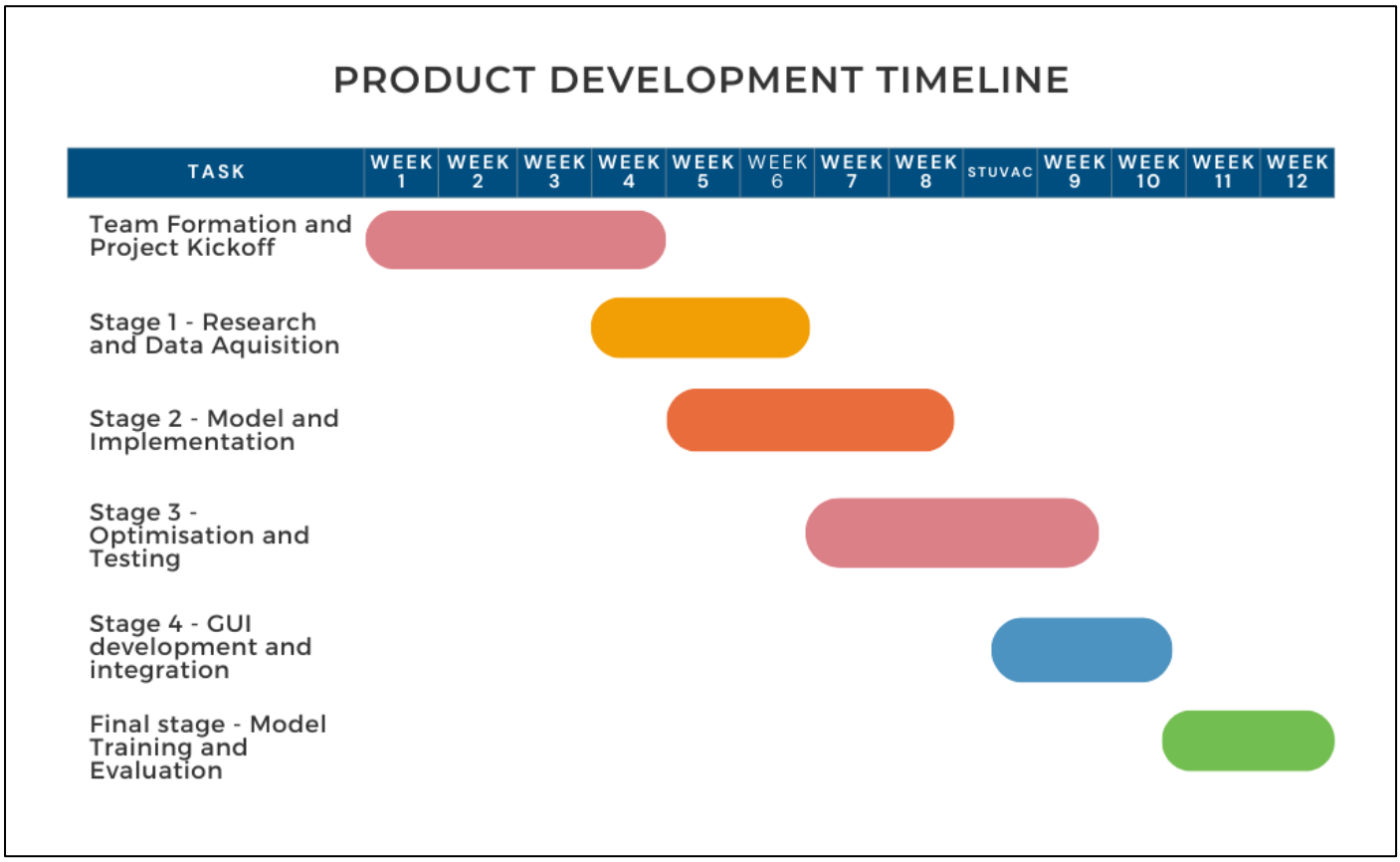

Figure 1. Project Timeline

Key development phases included:

- **Weeks 1–4**: Team formation, Project scoping and kickoff
- **Weeks 4–6**: Research and Data Acquisition, Prototype UI implementation
- **Weeks 7–9**: Model integration, UI testing, result visualisation, backend refinement
- **Weeks 10–12**: End-to-end validation, stakeholder feedback, roadshow preparation, and final documentation

To manage workflow, the team used Microsoft Teams to track weekly progress, allocate tasks, and identify priority areas for development. Team members also maintained milestone tables to log personal contributions against expected weekly outcomes. These tables supported reflections, time tracking, and evaluation of individual progress.

[See Section 12.2, Milestone Table]

5.4 Task Allocation and Workflow

Task allocation was based on team members' technical strengths and interests, while maintaining flexibility to adapt as the project evolved. The team was structured into two main areas: model development and user interface design. Integration, testing, and debugging were shared responsibilities in the final stages. The table below outlines the primary roles and contributions of each team member:

Table 2 Team Workflow Table

Team Member	Primary Area	Responsibilities
Tanish	Model Development	Co-led project; designed and trained CNN model using PyTorch; contributed to testing
Gajan	Model Development	Co-led project; handled data pre-processing, feature tuning, model testing & bugs
Batool	Frontend/UI Design	Designed and built user interface using React; ensured responsiveness and usability
Eunice	Frontend/UI Design	Assisted in interface layout, styling, and UX enhancements
Aung	Backend Integration	Helped develop Flask API, link model inference with frontend
Deepti	Backend Integration	Supported API integration, model response handling, and end-to-end testing

Regular group meetings and continuous messaging kept team members aligned throughout development. While primary roles were defined, collaboration was encouraged across tasks—particularly during model–UI integration and validation phases.

5.5 Tools and Technology Used

A range of tools and frameworks were used across different stages of the project, from model development to frontend deployment. These tools were selected based on their suitability for rapid prototyping, cross-platform compatibility, and ease of integration.

Table 3 Table of Project Software Tools

Category	Tool/Technology	Purpose
Frontend Development	React.js, Tailwind CSS	Built the user interface with responsive design and minimal layout styling
Backend Development	Flask	Served as the API layer between the frontend and AI model

Model Training	PyTorch, Google Colab	Used for CNN (U-Net) model development and training with GPU acceleration
Machine Learning	TensorFlow Lite	Enabled efficient model inference on web and mobile environments
Data Processing	NumPy, Pandas	Handled data formatting, normalisation, and feature testing
Version Control	Git, Github	Managed source code, tracked changes, and facilitated collaborative work
Testing and debugging	Browser Dev Tools, Postman	Used for frontend testing and API endpoint validation
Communication	Microsoft Teams	Facilitated team discussions, coordination, and file sharing

The chosen stack ensured a balance between performance, usability, and deployment flexibility. Tools such as Google Colab allowed for GPU-supported model training, while Flask and React enabled modular development and clear separation between logic layers.

5.6 Challenges Encountered During Development

During the project, the team faced several technical and coordination-related challenges. The table below summarises the key issues, their impact on the development process, and how they were resolved:

Table 4 Key Issue Summary

Challenge	Description	Resolution
Model–API Integration	Difficulty connecting the PyTorch-trained model to the Flask backend due to format inconsistencies	Standardised input/output processing and conducted multi-platform testing
Performance Variability	Slow model inference on lower-end devices compared to Colab/GPU environments	Switched to TensorFlow Lite for local/browser inference to reduce processing load
UI Responsiveness and Overlay Issues	Problems displaying AI-generated overlays clearly across screen sizes and devices	Refined layout and rendering logic; tested responsiveness across browsers
Non-Ultrasound Image Misclassification	The model initially produced outputs for irrelevant or invalid images (e.g., unrelated photos)	Added basic input validation and improved dataset curation during training
Confidence Threshold Configuration	Sensitivity slider failed to adjust model output thresholds reliably	Reconfigured post-processing logic to apply thresholds correctly across confidence outputs

6. Implementation

6.1 System Architecture Overview

The overall system architecture is composed of three main layers: the frontend interface, the backend API, and the AI model inference engine. The frontend, developed in React with Tailwind CSS for styling, enables users to interact with the system by uploading medical images. These images are passed to the backend via HTTP requests handled by a Flask-based API. The backend then communicates with a TensorFlow Lite model that performs the segmentation task and returns the results to the frontend. The modular separation ensures scalability and supports future integration with clinical systems or alternative models.

Data Flow Overview:

1. **User Action** – Upload scan (PNG, JPG, or DICOM) →
2. **Frontend** – Sends file and sensitivity parameter to Flask API →
3. **Backend** – Receives input, preprocesses the image →
4. **Model** – Performs inference with U-Net architecture →
5. **Output** – Backend returns overlay + confidence score →
6. **Frontend** – Displays output visually with optional export/forwarding

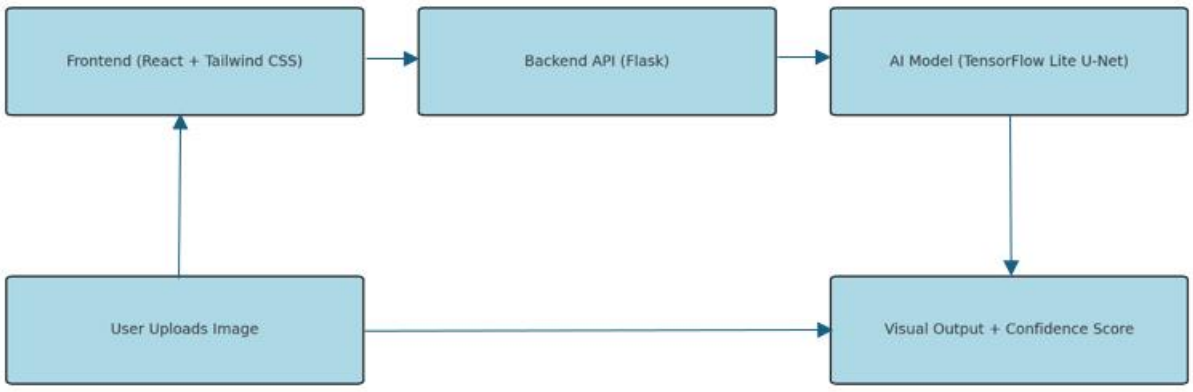

Figure 2. System Architecture Visualisation

6.2 Frontend Implementation

The frontend was implemented using **React** and styled with **Tailwind CSS**, chosen for its flexibility and responsiveness across devices. Key components of the interface include:

- **Upload Form**: Enables drag-and-drop or manual selection of image files
- **Progress Indicators**: Shows loading states during model inference
- **Visual Output**: Displays the original scan with the segmentation overlay
- **Sensitivity Slider**: Allows users to adjust confidence thresholds dynamically
- **Result Actions**: Options to download or forward analysis output

The design follows a minimalist three-step user flow: *Upload → Analyse → Review*. All elements are clearly labelled to reduce cognitive load and support accessibility.

6.3 Backend & AI Model Integration

The backend, built using **Flask**, acts as the bridge between the user interface and the TensorFlow Lite model. Upon receiving an image:

1. The backend **validates the file type** and processes it via a **preprocessing pipeline** (resizing, greyscale conversion, normalisation).
2. The image is then passed to the **U-Net model**, which has been converted to a TFLite format for lightweight inference.
3. The **segmentation mask** is generated and overlaid on the original image.
4. A **confidence score** is calculated, and both the overlay and score are returned as a JSON response.

This architecture supports local and browser-based deployment, with low latency (~5 seconds per scan on consumer hardware).

6.4 Data Flow and User Interaction

The user interaction follows a clearly defined flow:

- **Step 1**: The user uploads a breast scan image via the frontend
- **Step 2**: The image is sent via a POST request to the Flask server
- **Step 3**: The image is pre-processed and passed into the TensorFlow Lite model
- **Step 4**: The model returns a binary mask with areas of concern
- **Step 5**: The mask is overlaid on the original image and shown to the user along with a confidence score
- **Step 6 (optional)**: The results can be downloaded or forwarded to a healthcare professional

All errors (e.g. invalid formats or empty uploads) are handled with feedback messages for a smoother user experience.

6.5 Key Implementation Decisions

- *Why U-Net? Why TensorFlow Lite?*
- *How was mobile optimisation and local deployment supported?*
- *Simplifications, optimisations and trade-offs made*

Throughout development, several key decisions shaped the direction of our product, particularly in terms of architecture, model choice, and performance optimisation. One of the first major choices was to use a U-Net convolutional neural network. This model is widely used in medical image segmentation because of its ability to capture both high-level features and fine-grained details. For us, this meant achieving more accurate tumour localisation, even with a relatively small dataset. The U-Net architecture's encoder–decoder structure helped preserve important spatial information, crucial for identifying irregular shapes and sizes in breast tissue scans.

Initially, we trained our model using PyTorch due to its flexibility and active development tools. However, to support faster inference and broader compatibility across devices, we later converted the model to TensorFlow Lite. This allowed us to run predictions directly in the browser or on lower-spec hardware, which was essential for meeting our accessibility goals, especially for users in rural or remote settings with limited computing power.

Simplicity was another guiding principle, especially when designing the user interface. Inspired by frameworks like Gradio, we opted for a minimal three-step workflow — upload, analyse, and review — to keep things intuitive even for users without technical backgrounds. We deliberately avoided clutter and included only the necessary controls, such as a sensitivity slider to adjust confidence thresholds.

From the beginning, we also knew that the product needed to work offline or in low-connectivity environments. While we assumed access to internet-enabled devices, we designed the system to support local deployment where needed. This informed our decision to keep the model lightweight and avoid reliance on external cloud services.

Ultimately, these implementation decisions helped us balance performance with usability, resulting in a product that's not only technically sound, but also practical and accessible in real-world healthcare contexts.

7. Testing and Validation

7.1 Testing Strategy

The approach to testing our product focused on three key areas: functionality, usability, and performance. While the methodology wasn't formalised through automated test suites or structured QA pipelines, we continuously tested and iterated on each component as it was developed to ensure the system worked reliably across its core features.

For functional testing, each feature was evaluated as it was built. We checked the upload mechanism, the AI inference pipeline, and the output display to ensure they performed as expected. Any major bugs or unexpected behaviours were either flagged internally or resolved on the spot. This hands-on and iterative approach helped us stabilise the product progressively across development stages.

Usability testing was driven by one simple goal: making the interface intuitive for first-time users, including non-technical healthcare professionals. To achieve this, we drew inspiration from Gradio, a lightweight UI framework popular among machine learning developers for its simplicity. Our design mirrored this user-first mindset. After our initial roadshow, we received valuable feedback suggesting improvements to the layout and interface flow. These insights prompted changes in visual hierarchy, clarity of buttons, and responsiveness across devices. By refining the interface based on this feedback, we significantly improved the user experience.

Performance testing focused on ensuring the application could run smoothly in typical usage scenarios. The backend was evaluated for its ability to process uploaded images and return segmentation results with minimal latency. We observed that our Flask API, when paired with the TensorFlow Lite model, maintained reliable performance even on mid-range hardware. During local testing, image inference consistently returned results within 5 seconds — a practical speed for real-time use in clinical environments.

Although informal, this hands-on testing approach — reinforced by user feedback and continuous iteration — was effective in helping us deliver a functional, user-friendly prototype that aligns with the product's goals.

7.2 Validation Outcomes

Validation involved confirming that our product worked as intended, both in terms of technical accuracy and system reliability. We conducted several layers of validation: model integrity checks, output performance tests, backend responsiveness measurements, and architectural correctness. This ensured that each component of the pipeline, from scan upload to segmentation mask output, was functioning cohesively.

To begin, the runtime logs during model launch gave us a snapshot of the backend model's readiness and configuration. As shown in **Figure 3**, the console output confirms that the input image size was correctly set to

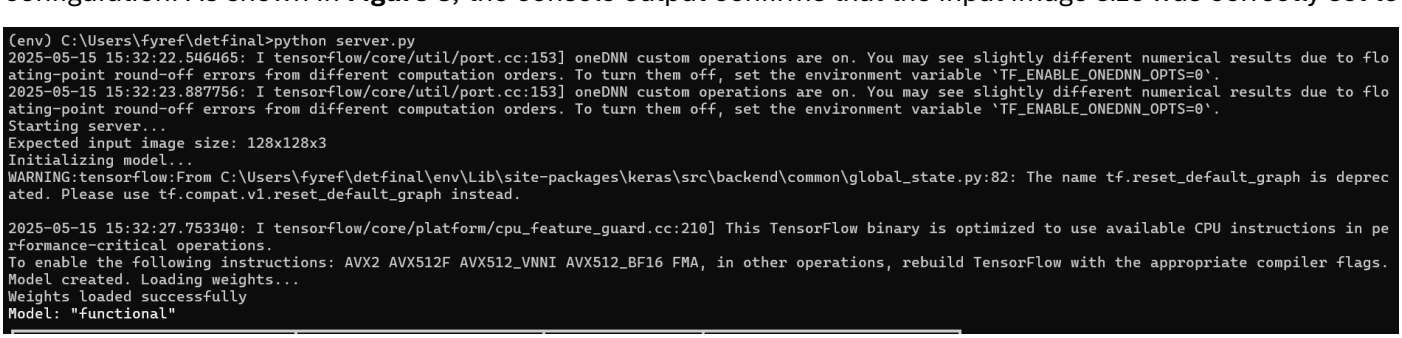

Figure 3 Backend console output showing model initialisation, input expectations, and successful weight loading.

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 128, 128, 3)	0	-
conv2d (Conv2D)	(None, 128, 128, 64)	1,792	input_layer[0][0]
conv2d_1 (Conv2D)	(None, 128, 128, 64)	36,928	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 64, 64, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 64, 64, 128)	73,856	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 64, 64, 128)	147,584	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 128)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 32, 32, 256)	295,168	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None, 32, 32, 256)	590,080	conv2d_4[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 256)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 16, 16, 512)	1,180,160	max_pooling2d_2[0][0]
conv2d_7 (Conv2D)	(None, 16, 16, 512)	2,359,808	conv2d_6[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 8, 8, 512)	0	conv2d_7[0][0]
conv2d_8 (Conv2D)	(None, 8, 8, 1024)	4,719,616	max_pooling2d_3[0][0]
conv2d_9 (Conv2D)	(None, 8, 8, 1024)	9,438,208	conv2d_8[0][0]
conv2d_transpose (Conv2DTranspose)	(None, 16, 16, 512)	2,097,664	conv2d_9[0][0]
concatenate (Concatenate)	(None, 16, 16, 1024)	0	conv2d_7[0][0], conv2d_transpose[0][0]
conv2d_10 (Conv2D)	(None, 16, 16, 512)	4,719,104	concatenate[0][0]
conv2d_11 (Conv2D)	(None, 16, 16, 512)	2,359,808	conv2d_10[0][0]
conv2d_transpose_1	(None, 32, 32, 256)	524,544	conv2d_11[0][0]

Figure 4 U-Net architecture summary showing sequential layer structure, output tensor shapes, and parameter counts.

128x128, weights were loaded without issue, and the U-Net model compiled in "functional" mode. The system initialised the model in under two seconds on a CPU, indicating efficient loading and minimal boot overhead. These outputs confirmed that our deployment environment was stable and that the TensorFlow Lite implementation was correctly aligned with the preprocessing pipeline.

Beyond runtime behaviour, we examined the architecture of the U-Net model in granular detail to validate its structural integrity. **Figure 4** shows the complete layer summary generated during model compilation. This includes the full encoding and decoding pipeline — from the initial Conv2D layers through each max-pooling and transpose-convolution stage, down to the final output shape. Each output tensor size matches the expected feature map resolution at each stage. The concatenation layer connections (skip connections) confirm that spatial context is retained across the down-sampling and up-sampling paths. This layer breakdown also gave insight into parameter distribution across the network, with deeper layers having progressively higher parameter counts — a typical and expected characteristic of effective semantic segmentation models.

Performance testing was then used to verify whether the backend system could respond reliably during real-time usage. The Flask server was tested using multiple scan uploads in succession to simulate usage under load. **Figure 4** displays the console log of these tests, showing a series of HTTP POST requests to the /predict endpoint. Prediction times ranged from 129ms to 491ms per image, with all responses returning a 200-status code. This confirmed that the API remained stable under repeated calls, and that the model's inference time was well within

acceptable limits, even when running on a local CPU. In a production setting with GPU acceleration or cloud deployment, even faster responses could be expected.

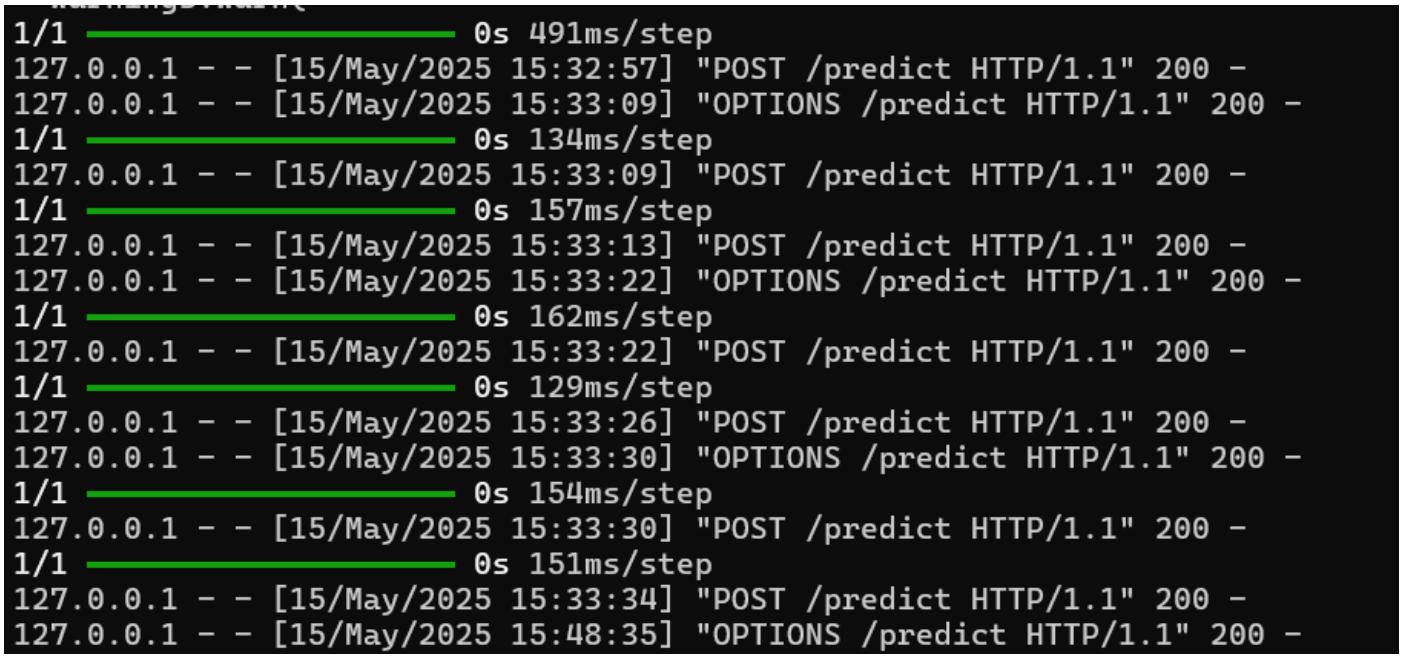

Figure 5 Flask API logs confirming real-time inference and consistent response times under repeated requests.

These validation outcomes, spanning architectural correctness, backend resilience, and model output speed, collectively ensured that the product was technically sound. Our testing confirmed that the model could be confidently deployed for further usability and clinical testing. Future iterations could expand this validation layer by integrating automated test suites, using more extensive datasets, and deploying the model within a containerised or cloud-based environment to assess cross-platform consistency.

7.3 Comparison

To better understand the value of our product, we compared its capabilities with existing commercial solutions in the medical imaging space. While our project was built on a smaller scale and with limited resources, this comparison helps highlight where our system excels, especially in terms of accessibility, affordability, and real-time performance.

The table below outlines key differences between our model and several well-known commercial tools. While professional solutions are often built on proprietary infrastructure and trained on vast datasets, our model demonstrates that lightweight, open-source systems can still achieve high segmentation accuracy and practical utility in clinical settings. This is particularly valuable for remote or resource-constrained environments, where cloud-based platforms may be impractical or unaffordable.

Table 5 Commercial Comparison Table

Key Considerations	Our Model	Existing Commercial Solutions
Segmentation Accuracy	Above-average accuracy based on small academic datasets	High accuracy using large, high-resolution datasets
Infrastructure	Can run locally without a GPU or cloud resources	Often requires expensive hardware or cloud computing
Code & Cost	Open-source, free to deploy, no subscription or usage fees	Closed-source, commercial licensing typically incurs ongoing costs

Ease of Use	Simple UI, designed for accessibility without technical expertise	Complex dashboards may require training or integration support
Optimisation	Tailored for low-latency inference using TensorFlow Lite	Tuned for performance at scale, but often less adaptable to local devices

While our system may not offer the same level of scalability or regulatory certification as enterprise-level platforms, its strengths lie in fast, accurate predictions and low-cost deployment. These attributes make it particularly suited to small clinics, educational use, or pilot trials in regions where high-tech infrastructure is not available.

8. Outcomes and Delivery

8.1 Final Product

The outcome of the project is a functional, cross-platform prototype application that performs breast cancer tumour segmentation using an AI model. Users can upload medical images (PNG, JPG, or DICOM), which are then processed through a backend system powered by a TensorFlow Lite implementation of U-Net. The result is returned as a visual overlay with a highlighted segmentation mask and an accompanying confidence score. This system is fully integrated: the React frontend communicates with the Flask backend via API, and inference is performed locally using lightweight model files. The application performs reliably on standard consumer hardware, maintaining low inference times of around 5 seconds per image. While primarily designed for desktop use, the frontend is responsive and mobile-compatible, though mobile optimisation is still in early stages and not fully deployed.

8.2 Requirement Fulfilment

The final product meets most of the functional and technical requirements outlined in Section 4. All core features, including image upload, AI segmentation, overlay generation, and confidence scoring, were implemented and tested successfully. The frontend offers drag-and-drop file support, a clean and intuitive interface, and real-time feedback during prediction.

From a usability perspective, the system operates in under three steps, adheres to accessibility design principles, and runs effectively on both desktop and mobile browsers. Technical goals such as TensorFlow Lite integration, API communication, and responsiveness across screen sizes were also met.

However, some stretch features remain incomplete. While the sensitivity slider exists in the interface, the backend logic for dynamically adjusting the confidence threshold was not fully stable at submission. Similarly, while the UI includes placeholders for “download” and “forward to specialist” functions, these were not fully integrated into the current build due to time constraints. Offline use support, although conceptually planned, has only been partially achieved through lightweight model deployment rather than full PWA packaging.

8.3 Demo Link

A working prototype of the system is available at the following link:

https://drive.google.com/file/d/1wadpOrz_M0e2-4NpMHhqfN7JprI9OdId/view?usp=drive_link

*Please note that the demo requires downloading project dependencies (React frontend, Flask backend, and Python environment) and may require configuration depending on the system setup.

9. Impact and Future Potential

9.1 Stakeholder Benefit

The development of this AI-powered breast cancer detection tool has the potential to offer significant value across the healthcare ecosystem, particularly for frontline medical professionals, rural patients, and diagnostic service providers.

In its current form, the product provides diagnostic support by identifying potential tumour regions within X-ray or ultrasound scans. This can assist general practitioners or nurses in regional and remote areas where specialist radiologists may not be readily available. By allowing fast and accessible first-stage screening, the tool helps reduce delays in diagnosis and speeds up referral decisions for further testing.

For radiologists and clinicians, the segmentation overlay offers a second opinion, one that is consistent and not subject to fatigue or variation. The confidence scoring mechanism further supports clinical decision-making by giving users transparency over how reliable the AI's prediction is for any given scan.

For patients, particularly in rural and underserved areas, this tool helps address disparities in access to care. Faster results can reduce the time between initial consultation and treatment, improving outcomes and reducing the anxiety that often accompanies diagnostic delays. In this way, the system contributes not only to clinical workflow efficiency but also to patient well-being.

9.2 Broader Potential

Beyond its immediate application, the product holds long-term potential in both the medical and technical spheres. As a research prototype, it provides a robust foundation for future development in AI-powered diagnostic support tools. With further training on larger, annotated clinical datasets, the model could be refined for use in real-world environments, including hospitals, telehealth platforms, or mobile health units.

The product’s lightweight architecture and compatibility with consumer hardware make it an ideal candidate for deployment in low-resource settings, both in Australia and internationally. Its modular structure also opens the door to supporting additional diagnostic tasks, such as lung nodule detection, liver tumours, or skin lesion analysis, through retraining the AI model on different imaging modalities.

Future versions of the system could incorporate secure databases for storing and forwarding medical records, as well as dropdown selection menus for specifying scan types or suspected conditions. Regulatory certification (e.g. through the TGA or FDA) would be a longer-term goal if the product were to be developed for commercial use.

Moreover, the project has value as an academic or training tool. It offers medical students and early-career radiographers a way to visualise segmentation outcomes, experiment with diagnostic reasoning, and better understand how AI integrates into healthcare workflows.

In short, the product’s greatest strength lies in its flexibility, both in how it can be used today and how it could evolve in the future. With continued development, ethical oversight, and stakeholder collaboration, this tool has the potential to meaningfully reshape the diagnostic journey for both patients and clinicians.

10. Reflections on Development

10.1 Level of Completion

At submission, the product is approximately 80% complete. We successfully delivered a working prototype that fulfils the core goals of the project: to build an accessible, lightweight diagnostic tool for breast cancer segmentation using AI. The final system allows users to upload medical images, process those images through a TensorFlow Lite implementation of a U-Net model, and return a segmentation overlay with a confidence score, all via a user-friendly web interface.

Many of the features outlined in our requirement specification were implemented. However, a few were only partially completed. The mobile version was conceptualised but not fully deployed. Similarly, some features visible in the frontend (such as the “forward to specialist” or download button) remain as design placeholders rather than active functions. The sensitivity slider is functional in the UI but not yet dynamically connected to the model's output.

Time constraints were the primary reason for these incomplete features. With a limited window for testing and deployment, efforts were prioritised toward building a stable, functional pipeline rather than stretching into unfinished territory. With additional time or resourcing, these remaining features could be implemented in future iterations.

10.2 Impact of Product

This project has a significant practical and social impact. By design, it targets underserved populations, particularly rural or remote communities, where access to specialist radiology services is limited. The ability to run locally without reliance on the internet or cloud infrastructure makes the tool ideal for use in mobile clinics, schools, or low-resource healthcare settings.

From a commercial perspective, the product has strong potential as a scalable health-tech solution. Its modular design means that with regulatory clearance and further validation, it could be developed into a licensed tool for GP clinics or integrated into existing telehealth platforms. Moreover, it could serve as a research or training tool for medical education, allowing students or early-career clinicians to interact with AI-assisted imaging outputs.

As proof of concept, the prototype clearly demonstrates feasibility. It segments images with high accuracy and does so quickly, with average prediction times under five seconds on a CPU. This balance of performance and simplicity validates our technical approach and suggests strong future viability.

Stakeholder feedback during testing and the roadshow further reinforced the product's relevance. Viewers consistently noted the clarity of the interface and the potential benefit of such a system in rural care. Some even suggested broader diagnostic applications (e.g. for skin or lung cancer), which aligns with our goal of generalising the system architecture for multiple scan types.

10.3 Potential for Further Development

There are two immediate directions for future development. First, within the Studio course environment, the product could be developed into a more robust tool with extended scan support, integrated feedback loops, and secured patient data handling. A follow-on Studio product could also focus on UI accessibility, refining cross-device responsiveness and offline usage through PWA (Progressive Web App) capabilities.

Second, in a commercial context, the system has the potential to evolve into a licensable diagnostic aid. This would require additional steps, including regulatory certification (e.g. via the TGA or FDA), a privacy-compliant

patient record system, and larger-scale clinical testing on real-world datasets. Integration with existing electronic health record (EHR) systems or telehealth interfaces could further boost adoption and utility in real practice.

We also envision future versions supporting dropdown options for scan types (e.g. mammogram, ultrasound, CT), auto-generated PDF summaries, and the option for clinicians to give manual input to augment AI predictions. These features would enhance both usability and trust in the system.

10.4 Roadshow Feedback

Presenting the product to a wider audience offered valuable insights into how it would be received in real-world settings. Many users responded positively to the interface, especially the clarity of the segmentation overlay and the minimal steps required to get results. The tool's speed and simplicity were consistently highlighted as key strengths.

Audience suggestions also prompted us to consider broader applications, such as supporting other types of scans (e.g. mammograms, ultrasounds) or detecting additional medical conditions. Some raised ideas for clinician input fields or integration into rural health outreach programs, which aligned with our original vision of expanding access to diagnostic tools in low-resource settings.

We also observed how people interacted with the system, which helped us identify UI improvements around labelling and result interpretation. These changes were implemented before final submission, reinforcing our commitment to accessibility and continuous refinement.

Overall, the feedback confirmed that the product was not only technically sound but also practical and relevant, especially for those working outside major healthcare centres.

11. Conclusion

11.1 Summary

The project successfully produced a working prototype of an AI-powered breast cancer detection tool. By utilising a U-net convolutional neural network, the efficiency and reliability of detecting segments in breast cancer ultrasounds and x-ray images that could pose a threat as breast cancer was improved. The product can be used via both smartphone devices and web platforms, which allows users to upload their scans and get instant results of highlighted regions of concern.

The biggest differentiator of our product from competitors is our emphasis on accessibility and ease of use, as evident in **Table 5**. Unlike other forms of breast cancer detecting tools that require a much more sophisticated infrastructure and technical expertise, our product was designed to cater for individuals in rural and remote communities, who may have limited education, healthcare awareness, or access to resources. The application helps address a major barrier by bridging the gap between the initial signs of potential breast cancer before medical professionals deliver of confirmed late-stage diagnoses. The user interface is easy and intuitive, and the backend model has been trained to provide fast and reliable results, helping to improve the standard of care for users.

Through this project, we addressed a need in the current healthcare landscape that was currently missing in the healthcare sector. The healthcare sector needs diagnostic support tools that operate more quickly and reliably while becoming easier to access. We delivered a product that meets the functional and technical requirements.

11.2 Final Thoughts

This project provided an opportunity to apply technical knowledge to a real-world healthcare challenge with meaningful impact. By working on a solution aimed at improving early breast cancer detection, the team gained firsthand experience in developing a product that balances machine learning performance, user accessibility, and design simplicity.

One of the most important lessons was understanding that technical excellence must be matched with usability. While accuracy and model performance were crucial, these alone were not enough — speed, stability, and a clear user interface proved equally vital. Developing for a diverse, potentially non-technical user base meant we had to consistently prioritise accessibility and responsiveness over complexity.

The project also highlighted the importance of adaptability and collaboration. The number of hours spent by each team member, as provided in **Table 6,** highlights the team's efforts to not only collaborate but contribute to reach our target product end goal**.** Whether resolving integration issues or refining the interface under time constraints, team members worked across domains to support one another and move the project forward. These challenges fostered a deeper understanding of end-to-end product development, beyond writing code or training models.

Overall, this experience strengthened both our technical and professional skillsets and reinforced the value of building solutions that are not only functional but also truly usable in real-world contexts.

12. Individual Contributions

12.1 Hours Table

Table 6 Individual Hours Contribution

Team Members	Gajan	Tanish	Eunice	Aung	Deepti	Batool	Total
Dataset Research	5	5	5	5	5	6	31
Team leadership/coordination	12	10	10	10	8	13	58
AI Model Training & Conversion	7	10	4	2	6	5	33
Backend Integration (Flask API, Model Deployment)	5	8	N/A	2	2	8	19
Frontend Development (React UI, Visual Output)	8	12	7	2	2	12	38
Testing & Validation (Accuracy, Inference, Feedback)	6	8	5	1	3	5	28
Report/Presentation preparation & review	7	3	7	8	7	7	39
Total	50	56	38	30	33	62	

12.2 Milestone Table

Gajan Suntherachelvan (24553760)

Product specific INDIVIDUAL Milestones	% done				
	W4	W6	W8	W10	W12
W4 Milestone • Background research • Programming on React • Finalise Gantt Chart	100%				
W6 Milestone • Data preprocessing • Developing a machine learning model • Programming on React	20%	70%	100%		
W8 Milestone • Preprocess Dataset • Developing a machine learning model • Build Frontend Prototype • Develop Backend prototype	0%	33%	100%		
W10 Milestone • Complete Frontend and Backend • Implement development changes for UI and AI Model • Make UI/UX concepts for mobile phone • Roadshow preparation	0%	0%	50%	100%	
W12 Milestone • Finalise ML model and integrate systems. • Conduct end-to-end testing. • Submit final report and documentation.	0%	0%	10%	35%	100%

Tanish Satre (24632541)

Product specific INDIVIDUAL Milestones	% done				
	W4	W6	W8	W10	W12
W4 Milestone • Flask Architecture research • U-net model Research	100%				
W6 Milestone • Finish creating the first prototype User interface • Assist in creating TensorFlow model (initial versions) • Establish flask routes	0%	20%	100%		
W8 Milestone • All UI components are functional • Flask API Working	0%	50%	100%		
W10 Milestone • All other backend core features are working and tested • Finish UI completely	0%	0%	20%	100%	
W12 Milestone • Final product ready • No errors or bugs • Model is working at max optimisation	0%	0%	0%	100%	

Eunice Anne Javier (13539705)

Product specific INDIVIDUAL Milestones	% done				
	W4	W6	W8	W10	W12
W4 Milestone • Researched u-net method • Starting to implement and play around with the u-net method in python	100%				
W6 Milestone • Tried to implement the -net methodology in visual studio code • Successfully presented our pitch and working prototype • Gained feedback from Steve regarding our prototype and what he wants us to fix	40%	65%	75%	100%	
W8 Milestone • Research on user interface design and played around with the software to understand how it works → figma, react & tailwind CSS • Started implementing a user face design • Researched regulatory constraints of medical devices	0%	45%	85%	90%	
W10 Milestone • Continued to work on front end design and its implementation • Road show preparation	0%	20%	45%	100%	
W12 Milestone • Finalised product • Submit final report and video presentation	0%	0%	0%	0%	100%

Aung Pyay (14390759)

Product specific INDIVIDUAL Milestones	% done				
	W4	W6	W8	W10	W12
W4 Milestone • Researched u-net method • Completed Python tutorials and practical exercises with the U-net model	75%	25%			
W6 Milestone • Designed Roadshow presentation slides and polished up teams' slides • Started drafting up short presentation for "teach the team" about communication skills. • Presented at roadshow	0%	90%	10%		
W8 Milestone • Delivered my communication skills presentation for "teach the team" • Continued to work on backend and supported team with testing	0%	10%	50%	40%	
W10 Milestone • Meeting with project owner to gain feedback for final presentation • Contributed to roadshow presentation and preparation for the roadshow • Interreacted with front-end team to learn more about the integration process and provided encouragement • Meeting with project owner and	0%	0%	0%	75%	25%
W12 Milestone • Final report writing and voice recording for video presentation • Document control	0%	0%	0%	0%	100%

Batool Al Salman (24452310)

Product specific INDIVIDUAL Milestones	% done				
	W4	W6	W8	W10	W12
W4 Milestone - Research - Meeting with product owner	100%				
W6 Milestone • U-net method • Figma designs	50%	25%	25%		
W8 Milestone - Front end development - Meeting with product owner	0%	50%	40%	10%	
W10 Milestone - Front end development - Roadshow prep - Making web app responsive	0%	50%	40%	10%	
W12 Milestone - Report writing	0%	0%	0%	70%	30%

Deepti Mallampalli (13614829)

Product specific INDIVIDUAL Milestones	% done				
	W4	W6	W8	W10	W12
W4 Milestone • Research on Medical AI topic, similar projects and existing use cases • Meeting with product owner	100%				
W6 Milestone • Research CNN models, CNN architecture and applications of CNN models for image processing • Research U-Net and practice with Python	0%	65%	30%		
W8 Milestone • Research React • Understand Front End UI designing • Python implementation	0%	0%	15%	30%	
W10 Milestone • Model Development and testing • Roadshow preparation	0%	0%	30%	40%	
W12 Milestone • Product finalisation and review • Final presentation and report	0%	0%	0%	45%	65%

12.3 Communication Log with Product Owner

Table 7 Communication Log with Project Owner

Meetings with Product Owner (Dr Steve Ling)			
Meeting	**Date/Time**	**No. Attendees**	**Meeting Focus**
Meeting 1	06/03/25, 4-5 pm	6	Product direction and objectives
Meeting 2	03/04/25, 4-5 pm	4	AI model review + software development
Other Key Product Owner Interactions			
Event	**Date/Time**	**Response**	
Emailed Steve to set up an online meeting with the team to discuss the project	1:33pm 1st March 2025	Confirmed to discuss the project in person on the 6th of March during the studio class	
Emailed Steve to attend the week 6 roadshow	12:02pm 19th March 2025	Steve confirmed his attendance to see our working prototype	
Showed Steve our working prototype at the week 6 roadshow	27th March 2025	Received feedback from Steve regarding what he wanted us to implement for the final product	
Emailed Steve to attend the week 12 final roadshow	5:39pm 6th May 2025	Steve confirmed his attendance to see our finalised product at the roadshow	

Presentation / Video

Final presentation.pptx

Final presentation 1

References

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). *Dermatologist-level classification of skin cancer with deep neural networks*. Nature, 542(7639), 115–118. https://doi.org/10.1038/nature21056

Google Health. (2020). *AI for breast cancer screening*. Retrieved from https://health.google/health-research/tools-and-research/breast-cancer-screening/

Long, J., Shelhamer, E., & Darrell, T. (2015). *Fully convolutional networks for semantic segmentation*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3431–3440). https://doi.org/10.1109/CVPR.2015.7298965

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional networks for biomedical image segmentation*. In Medical Image Computing and Computer-Assisted Intervention (MICCAI) (pp. 234–241). Springer. https://doi.org/10.1007/978-3-319-24574-4_28

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., & Liang, J. (2018). *UNet++: A nested U-Net architecture for medical image segmentation*. In Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (pp. 3–11). Springer. https://doi.org/10.1007/978-3-030-00889-5_1

Zhou, L. Q., Wu, X. L., Huang, S. Y., Wu, G. G., Ye, H. R., Wei, Q., … & Li, A. H. (2019). *Lung ultrasound in COVID-19 and beyond: A systematic review and meta-analysis*. Journal of Ultrasound in Medicine, 38(7), 1729–1741. https://doi.org/10.1002/jum.14943