AICardia: Smart Detection of Heart Conditions

NIMRA IQBAL
01-134212-148
REHAN AHMED
01-134221-068

Supervisor: Ma'am Maryam Bibi

Bachelor of Science in Computer Science

Department of Computer Science
Bahria University, Islamabad

December 2025

Certificate

We accept the work contained in the report titled "AICardia: Smart Detection of Heart Conditions", written by MS. Nimra Iqbal AND MR. Rehan Ahmed as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by:

Supervisor: Ma'am Maryam Bibi (Lecturer)

_____

Internal Examiner: Dr. Saba (Senior Assistant Professor)

_____

External Examiner: Shahid Mehmood

_____

Project Coordinator: Dr. Asim Ali (Senior Assistant Professor)

_____

Head of the Department: Dr. Khurram Ehsan (Senior Associate Professor)

_____

December, 2025

Abstract

Cardiovascular diseases (CVDs) remain the leading cause of death worldwide, emphasizing the need for timely and accurate diagnosis. Cardiac MRI is the gold standard for assessing heart conditions, but the manual interpretation of these images is a complex, time consuming process requiring specialized expertise. Cardiologists and radiologists must manually analyze MRI scans to evaluate heart function, segment key regions, and compute critical cardiac parameters such as End-Systolic Volume (ESV), End-Diastolic Volume (EDV), and Ejection Fraction (EF). This manual approach introduces the risk of delayed diagnoses, human error, and variability in results. In critical cases, such delays could mean the difference between life and death, as immediate intervention is often required for patients at risk of heart failure or cardiac arrest.

To address these challenges, this project aims to develop an AI-powered mobile application that automates cardiac MRI analysis, providing diagnostic support to healthcare professionals. The system will segment the left ventricle and myocardium, compute essential cardiac metrics, and classify heart conditions to assist in clinical decision making.

The project will choose and utilize datasets from well established cardiac MRI repositories. The MICCAI 2009 Sunnybrook dataset consists of MRI DICOM images from 45 patient batches, accompanied by coordinate files and patient data for precise segmentation and measurement. Alternatively, the ACDC dataset contains 1,800–2,000 MRI data files, enabling the classification of heart conditions such as normal, dilated cardiomyopathy, hypertrophic cardiomyopathy, and myocardial infarction.

This project is aimed at strengthening the accuracy, efficiency, and accessibility of cardiac MRI analysis by incorporating AI-based automation in the diagnosis process so that critical diagnoses could be performed in time and therefore, improve patient outcomes and decrease the workload of healthcare professionals.

Acknowledgments

In the name of Allah, the most beneficent and the most merciful.

All glory be to Allah Almighty, who enabled us to take note of this fantastic opportunity to expand our knowledge and provided us with the devotion and wisdom to work and plan with great passion.

We owe all our academic work and struggles to our dear parents and respected teachers; without them, we are nothing. We would like to convey our heartfelt gratitude to our project supervisor Ma'am Maryam Bibi, for her helpful assistance, comments, and recommendations during the project. Lastly, we would like to commend our friends who make study challenges and hard times full of excitement.

REHAN AHMED AND NIMRA IQBAL
Islamabad, Pakistan

December 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

DSA	Data Structure and Algorithms
OOP	Object Oriented Programming
PF	Programming Fundamentals
SE	Software Engineering
SQL	Structured Query Language
UNESCO	United Nations Educational, Scientific and Cultural Organization
UNICODE	Unique, Universal, and Uniform Character enCoding
XML	Extensible Markup Language

Chapter 1

Introduction

Cardiovascular diseases (CVDs) are a leading cause of death worldwide, requiring early and accurate diagnosis to improve patient outcomes. Among the most critical indicators of cardiac health are the left ventricle (LV), myocardium function and ejection fraction (EF), which help identify conditions such as heart failure, cardiomyopathy, and myocardial infarction. The cardiac MRI is universally regarded as the gold standard of heart functionality analysis because of its ability to provide a high-resolution image and the possibility to examine the cardiac system comprehensively [2]. The conventional MRI analysis however, is a time consuming procedure, which requires expert interpretation that may compel cardiologists and radiologists to manually segment MRI images, derive cardiac parameters, and categorize heart conditions. The risks presented by this manual approach include the possibility of errors and variations in diagnosis and delays in clinical decision making; these are potentially life threatening in an emergency situation.

Artificial Intelligence (AI) and deep learning have become the potent tools in the field of medical imaging, as they can provide automated and highly accurate responses to the detection and diagnosis of diseases in recent years [4]. Using the computer vision methods, this project will strive to create an AI-driven mobile app to analyze heart MRI scans automatically. Deep learning algorithms will be used to segment the left ventricle and to classify heart conditions using the system. Upon incorporation of these AI models into a friendly mobile app, the system will allow effective, standardized, and automated cardiac health monitoring.

The publicly available cardiac MRI datasets that will be used in this project will be the MICCAI 2009 Sunnybrook dataset or the Automated Cardiac Diagnosis Challenge (ACDC) dataset [10]. These databases are known to have thousands of MRI scans with

respective heart conditions and hence they are perfect in training deep learning models with regards to segmentation and classification. The suggested system will take out critical cardiac parameters, including End-Systolic Volume (ESV), End-Diastolic Volume (EDV), and EF computation, which will allow a medical specialist to gain more efficient and dependable diagnostic data.

This project seeks to overhaul cardiac MRI analysis by automating the segmentation and volume calculation/classification process to decrease the number of cardiologists required to carry out their respective duties, and offer a better diagnostic result. The mobile application created will offer accessible, time saving, AI-driven diagnostic application, which will guarantee faster cardiac screening and eventually enhance patient care.

1.1 Project Overview

One of the major causes of death in the globe is cardiovascular diseases (CVDs). Early intervention and treatment of heart conditions cannot be done without diagnosis because it is important to diagnose accurately and efficiently. Cardiac MRI is a non invasive imaging procedure that gives detailed information regarding the cardiac anatomy and functionality particularly the left ventricle (LV) and myocardium which is vital in determining the performance of the heart. Nonetheless, cardiac MRI images are time-consuming to manually analyze and likely to be inaccurate, and it involves the skills of experts.

The current proposal, titled "AICardia: Smart Detection of Heart Conditions", is an AI-driven system as illustrated in figure 1.1 that will be used to automatically segment and analyze cardiac MRI scans. The project will focus on creating a mobile application that uses deep learning models to detect the LV, calculate cardiac parameters that include End-Systolic Volume (ESV), End-Diastolic Volume (EDV), and Ejection Fraction (EF) and diagnose common heart diseases.

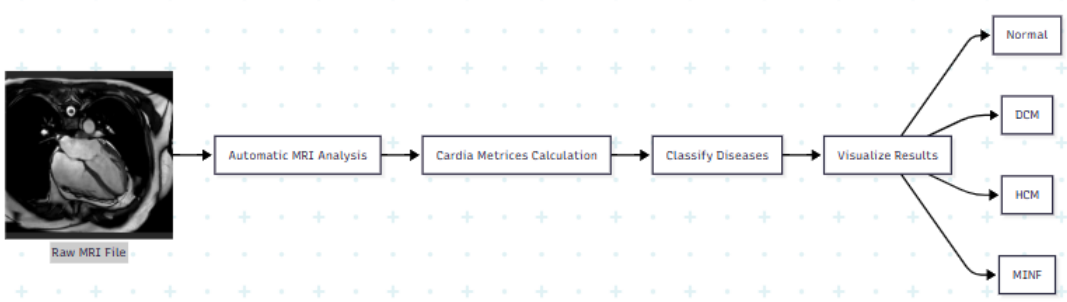

Figure 1.1: Working of Model

1.2 Problem Description

The cardiac MRI analysis is a labor intensive and complicated process which involves segmentation of heart areas, volume calculations and the derivation of measures such as EF. Manual analysis is likely to be erroneous, inconsistent and slow, and this may cause misdiagnosis or postponement of treatment. Moreover, cardiac MRI analysis is not adopted in resource constrained environment due to unavailability of easy and automated tools to analyze the acquired images.

This project addresses the following problems:
1. Long time-taking manual analysis: Cardiologists have to spend a lot of time manually segmenting MRI images and computing measures.
2. Human flaw and inconsistency: Human analysis is a subjective process that is likely to be biased.
3. Inaccessible tools: The available tools are either not part of the clinical processes, or they are too complicated and cannot be used.
The suggested system will shorten the time spent on diagnosing, enhance the accuracy and make cardiac MRI analysis more available to more medical professionals by automating these tasks.

1.3 Objectives

To create a mobile application with AI capabilities that will replace the manual analysis of cardiac MRI and allow the correct segmentation and calculation of the volume and identification of the disease to help cardiologists diagnose heart diseases more effectively.
• In order to retrieve and calculate significant cardiac health markers (ESV, EDV, EF).
• To classify heart conditions using a deep learning classification model.
• To deploy a mobile app with cross platform using Flutter.
• To implement the backend logic using FastAPI.

1.4 Project Scope

AICardia project is concerned with the creation of an AI-based mobile application through which the cardiac MRI scans can be processed automatically. The main idea is to help healthcare workers by means of prompt, precise, and dependable diagnostic information with the help of AI-driven segmentation and grouping by heart conditions. The system will segment the left ventricle and Myocardium of MRI images and extract important cardiac parameters like End-Systolic Volume (ESV), End-Diastolic Volume(EDV) and Ejection Fraction(EF). As well, the AI model will categorize heart diseases, such as dilated

cardiomyopathy, hypertrophic cardiomyopathy and myocardial infarction.

In order to do so, the project uses already existing datasets of cardiac MRI, such as the MICCAI 2009 Sunnybrook or ACDC dataset, which includes labeled MRI images and patient records. The datasets will be employed to train deep learning models to achieve proper segmentation and classification. The system will save time used in manually analyzing MRI and minimize human error, and enhance diagnostic accuracy by automating these tasks.

The mobile app will have a convenient interface where the medical workers will find it easy to upload MRI image file, access their diagnostic results, and get analysis. This will greatly simplify the task of the cardiologists and the radiologists and enable them to diagnose the heart conditions and enhance the treatment of the patients.

Nevertheless, the given project is not confined to processing rare cardiac conditions since the datasets provided do not discuss the conditions in detail. Also, the existing system will not be linked to the hospital Electronic Health Record (EHR) systems, but future efforts may involve such a connection. Instead of offering a full clinical endorsement or integration into the current medical processes, the project is aimed at automating the cardiac MRI analysis process and also offer a valuable diagnostic aid.

1.5 Summary

The present form of AICardia project wills to offer a practical segmentation and classification of heart aberrations through MRI scans. Nevertheless, our model is not fully utilized, as this project does not possess enough time. More time can be used to refine the performance of the model by adding a wider range of datasets and optimizing the deep learning structure. Moreover, the incorporation of the multi-modal data of imaging, e.g., CT scans, or the echocardiogram, may contribute to a more detailed diagnostic tool. Further developments can be the integration of additional cardiac disorders into the classification and streamlining the system to be used in clinical practice. The subsequent step of the system development could also be concerned with the integration of the system with hospital EHRs and the concept of cloud-based remote diagnostics.

Chapter 2

Literature Review

The chapter reviews all past and current AI-based cardiac diagnostic technology progress which focuses on AI systems that process cardiac MRI information. Academic publications contain research studies which demonstrate how AI and deep learning systems can identify cardiac disease. The following chapter aims to review existing research about this field by presenting essential methods and results together with the main obstacles which occur when using AI models for cardiac image analysis.

2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) serve as deep learning models which experts use to perform image processing operations through image classification and segmentation and object detection tasks [2]. CNNs have gained significant attention due to their ability to automatically learn features from images and their strong performance in tasks such as medical image analysis, particularly in detecting and classifying heart conditions from MRI scans.

CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers of the network perform feature extraction from input images through filter applications which identify different patterns including edges and textures and sophisticated image elements. The pooling layers decrease data dimensions while keeping essential features which results in better network performance and reduced computational requirements.

CNNs were first introduced by Yann LeCun in the late 1980s with the development of the LeNet-5 architecture. The first version of this model functioned for handwritten

digit identification yet it established the base which future image recognition systems use. However, CNNs only gained significant popularity after the development of deeper networks and the availability of large datasets and powerful GPUs for training. The 2012 ImageNet competition victory of AlexNet by Alex Krizhevsky established CNNs as the leading computer vision method through its impressive performance achievement.

The different types of CNNs exist to perform particular image processing operations. Basic CNNs consist of convolutional, pooling, and fully connected layers and are commonly used for classification and segmentation tasks, such as the cardiac MRI segmentation in the AICardia project. The architecture of Fully Convolutional Networks (FCNs) uses convolutional layers instead of fully connected layers to generate pixel-specific predictions which makes them suitable for segmentation work such as left ventricle boundary detection in MRI images [11]. The vanishing gradient problem in deep networks becomes solvable through Residual Networks (ResNets) which use skip connections to build networks that achieve better results in image classification tasks. U-Net is a CNN architecture designed for medical image segmentation, featuring an encoder-decoder structure with skip connections to preserve spatial information, making it highly effective for segmenting cardiac structures. The Inception Networks achieve their purpose through multiple kernel sizes which operate within one layer to process different scales of features while detecting objects at different sizes thus they work well for complex medical images including cardiac MRI scans.

The success of CNNs exists despite two main challenges which include needing big datasets with labels for training and the possibility of model overfitting when insufficient regularization techniques are used. The network faces challenges when dealing with different image quality levels which include both noisy and poorly resolved MRI scan data. The system faces two main obstacles which data augmentation methods together with proper model parameter adjustments help to overcome.

2.2 Fully Convolutional Dense Networks for Cardiac MRI Analysis

The correct identification of ventricles in cine cardiac MRI through cine cardiac MRI is essential for cardiac function assessment and automated diagnosis support. One model that's become especially popular for this kind of medical image segmentation is the Fully Convolutional Dense Network (FCDenseNet), often referred to as the Tiramisu model. It builds on the strengths of dense connections and a UNet style encoder decoder structure [12] to produce detailed, high resolution segmentations of anatomical regions.

The FCDenseNet system operates through a process which starts by decreasing the input slice resolution to detect general patterns across the entire image. The process operates in reverse sequence to improve feature definition through its upsampling operations. Throughout this upsampling path, it uses skip connections to bring back information from earlier layers in the encoder. The dense blocks function as basic components which form the structure of this system. The block enables information reuse through its mechanism which combines feature maps from all previous layers to enhance gradient flow. The design functions optimally for separating thin cardiac structures including the myocardium because it allows both exact boundary detection and full anatomical visualization.

2.2.1 Preliminaries

The architecture of fully convolutional networks (FCNs) replaces traditional fully connected layers with convolutional layers. The model produces pixel-wise predictions through its convolution-only operations which [11] introduced. FCDenseNet extends FCNs by incorporating the DenseNet concept, where each convolutional layer receives the concatenated outputs of all preceding layers within the block. This design:

- enables efficient feature propagation,
- reduces redundancy in learned features, and
- stabilizes training even with limited medical datasets.

The particular features of dense FCNs make them appropriate for cardiac MRI work because they contain these characteristics. The anatomical boundaries show different patterns between different image sections and different participants.

2.2.2 Segmentation Architecture Stages in Cardiac MRI

The analysis of cardiac MRI data employs deep convolutional models which detect and segment anatomical structures from cine sequence images. The UNet based architecture FCDenseNet and other models have gained popularity because they achieve high performance in ACDC and HVSMR challenge benchmarks. The models maintain detailed spatial information through their dense connections which also enable them to detect broader contextual features that become essential for heart structure analysis because of its thin walls.

2.2.3 Applications in Functional Quantification

The first stage requires the researcher to define the left ventricular (LV) cavity throughout all phases of the cardiac cycle. The segmentation process of the cavity throughout the

entire sequence allows the creation of a volume time curve. The highest and lowest points on this curve correspond to the clinically important phases: end diastole (ED) and end systole (ES). The two points enable researchers to calculate essential functional parameters which include end diastolic volume (EDV) and end systolic volume (ESV) and ejection fraction (EF).

The automated cardiac quantification process uses this method because it eliminates the requirement for human operators to label ED and ES frames which results in faster and more reliable operations.

2.2.4 Applications in Structural Assessment

A second segmentation step is typically introduced to measure myocardial thickness and mass, which are important structural markers in diseases like hypertrophic cardiomyopathy (HCM) and myocardial infarction (MINF) [13]. Research studies focus on myocardial segmentation during end diastole because this time offers the most suitable conditions to assess ventricular wall thickness.

Dense connected FCNs have been shown to improve the segmentation of the myocardium's thin, ring shaped structure. The system achieves better boundary detection results when it uses boundary aware loss functions together with edge focused training methods which scientists have previously studied.

2.2.5 Significance of Dense FCN Models in Cardiac MRI

Dense fully convolutional architectures have shown strong performance in a number of cardiac MRI benchmarks, including ACDC. Their strengths come from:

- effectively capturing multi scale contextual information,
- handling slice to slice variability with robustness,
- achieving high Dice scores for both the ventricular cavity and the myocardium, and
- performing well even on low resolution or noisy clinical data.

These characteristics make FCDenseNet and related models well suited for automated left ventricle and myocardium segmentation pipelines, providing a solid basis for downstream clinical metric estimation and diagnostic classification.

2.3 Related Research

Scientists have conducted numerous studies about AI-based cardiac imaging systems which focus on developing automated systems for cardiac MRI scan evaluation. The left

ventricle segmentation and heart disease detection tasks achieve better results through the application of AI techniques which use Convolutional Neural Networks (CNNs). The models achieved high success in identifying vital heart structures while they successfully recognized various heart-related health issues. The analysis speed and accuracy have improved through deep learning methods which perform automated tasks that used to require human analysis. The application of CNNs for left ventricle segmentation in MRI scans has shown success in obtaining exact boundaries which doctors need to evaluate heart function and determine Ejection Fraction (EF) and End-Diastolic Volume (EDV) values.

A recent paper A Novel Light U-Net Model for Left Ventricle Segmentation Using MRI [1], 2023 introduces a lightweight U-Net model specifically designed for the segmentation of the left ventricle (LV) from cardiac MRI scans. The U-Net model underwent modifications to create an optimized version which uses its encoder-decoder structure for cardiac structure segmentation while minimizing computational needs. The model was trained on the MICCAI 2009 dataset, a widely used dataset in the field of cardiac MRI, which consists of both MRI images and corresponding ground truth segmentations. The model produced outstanding results through its 97.7% Dice coefficient and 92% accuracy. Additionally as shown in the figure 2.1, histogram-based enhancement techniques were employed to further improve segmentation quality, making it one of the most efficient models for LV segmentation from MRI scans. The research shows deep learning models have developed better abilities to perform cardiac structure segmentation with high precision which enables cardiac function quantification.

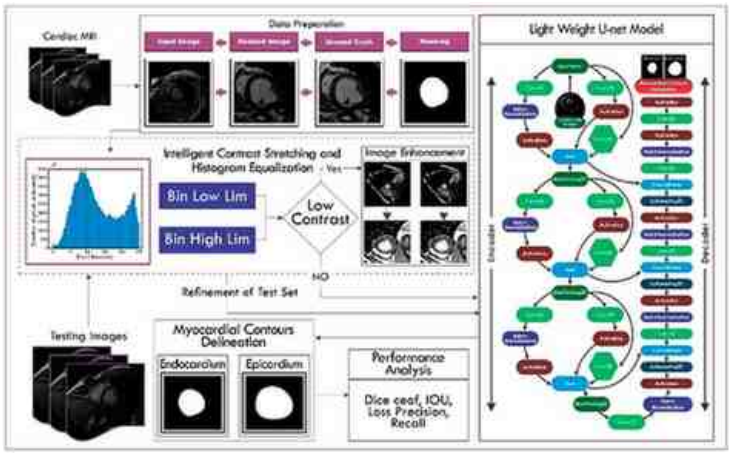

Figure 2.1: Phases of the Light U-Net model [1]

The research A CNN-Based Model for Heart Disease Detection [2] from 2024 describes a CNN model which uses MRI scans to identify heart-related medical conditions. The model analyzes MRI data along with additional features such as caffeine intake, which

is introduced as a novel risk factor for heart disease. The model was trained using the Enugu State University and CAD Kaggle dataset, which includes over 90,000 samples of heart disease data. The model achieved 94% accuracy which indicates it has potential to serve as an effective method for detecting heart disease at its beginning stages. This study highlights in the figure 2.2, the potential for CNNs to classify heart disease by learning complex patterns in MRI data and integrating additional risk factors, which could provide a more comprehensive prediction model for healthcare professionals.

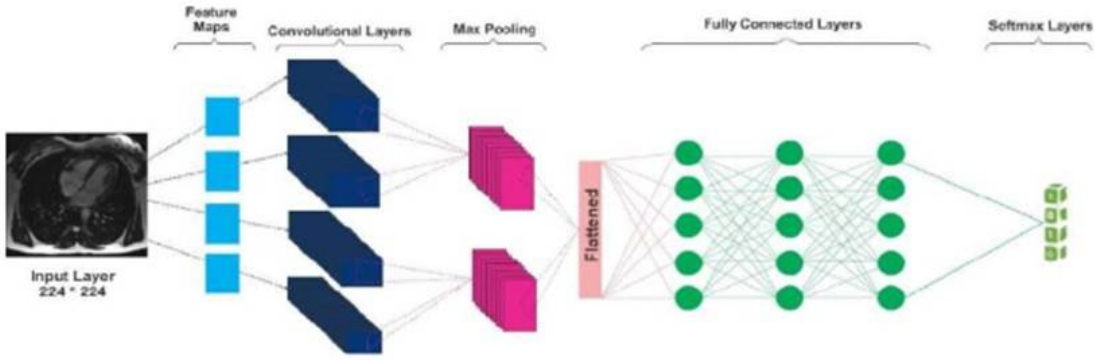

Figure 2.2: Architecture of the CNN adopted for the Classification Model. [2]

The article Cardiologist-Level Arrhythmia Detection with CNNs [3], 2017, is dedicated to the application of the deep learning algorithm and CNNs to ECG waveform arrhythmias. The research model was trained using a dataset of 64,121 ECG record of 29,163 patients. This was to develop a model that would identify arrhythmias just as well as a cardiologist. In the detection of arrhythmia, the CNN-based model was better than cardiologists as indicated in 2.3 indicating that deep learning can be able to perform better than human beings in specific diagnostic tasks. This work is noteworthy since the detection of arrhythmia is generally sensitive in time and may be rather difficult to detect by clinicians in practice. The fact that CNNs can process high amounts of ECG data in a short time and with high accuracy makes them extremely useful tools in a clinical practice, particularly in high-risk cardiac patients.

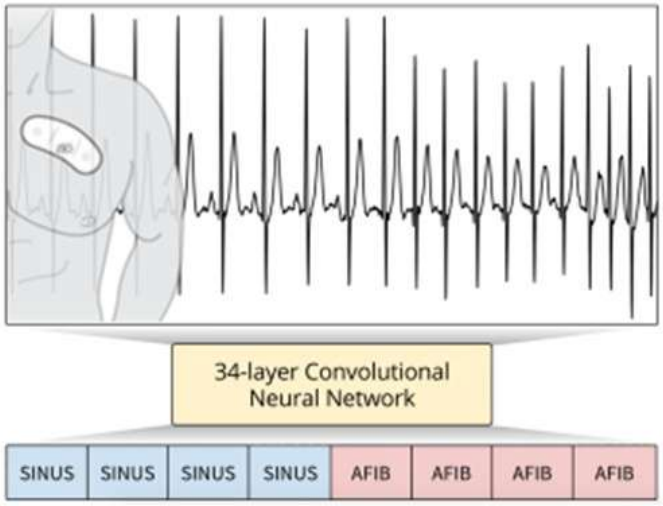

Figure 2.3: Trained convolutional neural network correctly detecting the sinus rhythm (SINUS) and Atrial Fibrillation (AFIB) [3]

The ongoing study Cedars Sinai ECG AI [14], at Cedars Sinai focuses on using models of AI prediction of sudden cardiac arrest (SCA) based on ECG data. The study makes use of hospital patient records (PRESTO) to train the AI models, and the aim is to detect faint patterns of ECG data, which are associated with the risk of cardiac arrest. It can save lives because the capacity to forecast the occurrence of cardiac arrest prior to its occurrence can save lives due to the provision of early intervention. The study is a continuous one, with the issues of explainability of the model and data privacy being considered to make the model useful in clinical models. The work is a great example of AI applied to the real-time clinical process to enhance the patient care and outcome.

Another article Machine Learning for Real Time Cardiac Arrest Prediction [4], 2018, describes a machine learning model trained to predict cardiac arrest in the hospital based on the results of the heart rate and electronic health record (EHR) data. The model was trained on EHR data of the NHS Foundation Trust that comprised patient vitals and medical history. Through this information, this model could detect at risk patients, as well as anticipate the occurrence of a cardiac arrest. The paper emphasizes the significance of predictive modeling in the medical field, where machine learning can assist clinicians to determine at risk patients early enough, thereby increasing their likelihood of survival due to the early intervention measures. The patient selection process employed in the study in question adheres to the organized pipeline in figure 2.4 reflecting on the screening, exclusion, and dataset division approach to the cardiac arrest prediction research presented in the article [4].

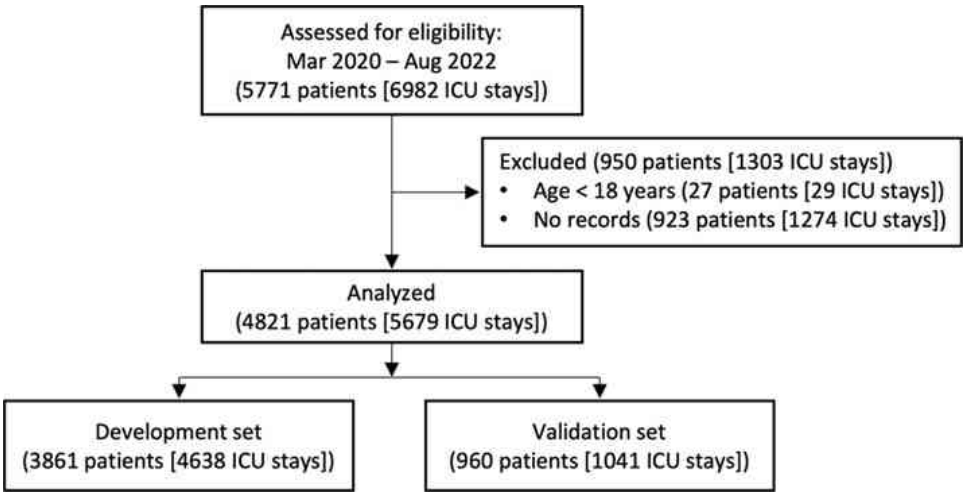

Figure 2.4: Flowchart of the study cohort. [4]

More study of Artificial Intelligence Enhanced Electrocardiography for Accurate Detection of Heart Disease [5], 2024 explores the use of AI to enhance the interpretation of ECG data for heart disease detection. The research as in figure 2.5 focuses on the analysis of IV/EV ECG leads and how AI can improve accuracy by learning from large datasets of patient ECGs. The paper discusses the challenges AI faces in processing ECG data, such as dealing with noisy signals and missing information. Despite these challenges, the study emphasizes the potential of AI to enhance diagnostic accuracy in heart disease detection. It also highlights the need for interoperability between AI systems and existing healthcare infrastructure to allow seamless integration into clinical workflows.

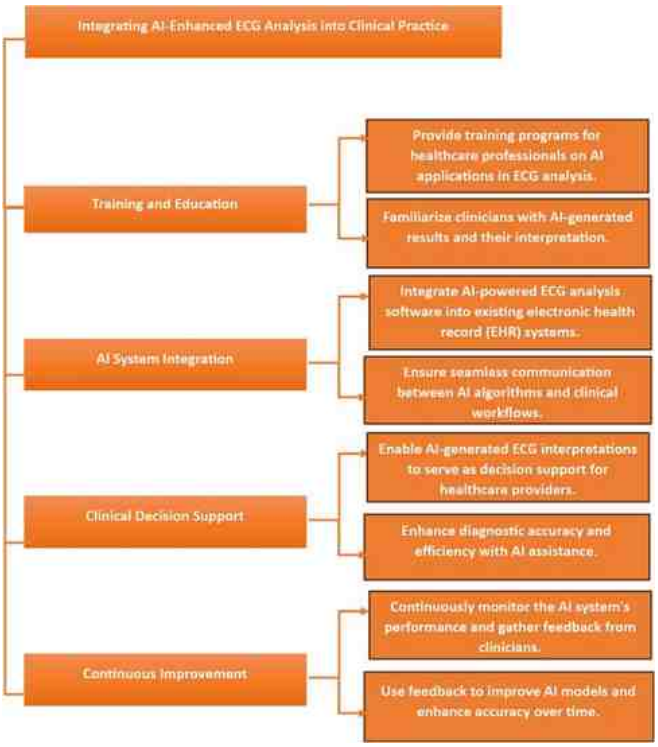

Figure 2.5: Integrating AI-enhanced ECG analysis into clinical practice. [5]

The review article, Prospects for Cardiovascular Medicine Using Artificial Intelligence [6], 2021, is a narrative review that includes a summary, Figure 2.6 showing an overview of AI applications in various cardiovascular diagnostic instrument, ECG, echocardiography, and CT scans. The paper elaborates on transformative opportunities for augmenting diagnostic accuracy and decision support in cardiovascular medicine through AI. Yet, the study also recognizes formidable challenges such as requirement for high quality data and the means to bring AI tools into real world clinical practice. It highlights that although AI is promising in cardiovascular diagnosis, several barriers in terms of data quality, model robustness, and clinical adoption still need to be overcome for a wider use of such technologies.

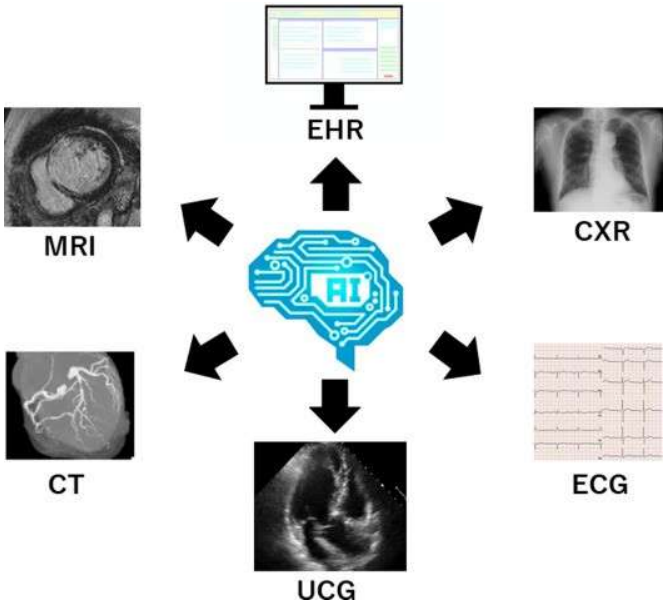

Figure 2.6: Artifical intelligence (AI) development status in the cardiovascular field. AI for electronic health record (EHR), chest X-ray (CXR), electrocardiogram (ECG), ultrasound cardiography (UCG), computed tomography (CT), and magnetic resonance imaging (MRI) has been developed. [6]

A further article, Artificial Intelligence and Cardiology: Current Status and Perspective [7], 2021, overviews the use of AI in cardiology, also depicted in figure 2.7, with a focus on ECG and electrophysiology among other things. The investigation highlights the burgeoning field of AI and its promise to transform cardiology by unlocking unanticipated knowledge, even including a patient's age and sex using the ECG. The paper also describes the need for trustworthiness and interpretability of AI models in the context of concerns around "black box" models in deep learning. It also outlines the fact that fostering better interpretability of models will be essential in order to gain clinical acceptance and trust, and ultimately in order to foster application of AI in cardiology.

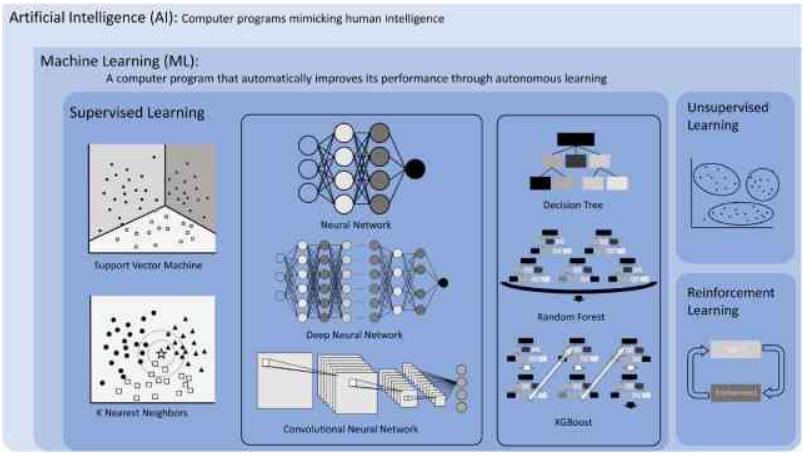

Figure 2.7: Types of artificial intelligence. [7]

Artificial Intelligence Enabled Tools in Cardiovascular Medicine [8], 2023, in this qualitative study, perceptions of cardiologists and health IT administrators regarding AI in cardiac care are investigated, as depicted in figure 2.8. Through interviews and focus groups, the study investigates the opportunities and barriers associated with AI for clinical adoption. Despite the promise of AI, the report highlights barriers that impede clinical use, including distrust and lack of interoperability. The study suggests the need for the formulation of unified protocols that can aid in the integration of AI tools in the existing healthcare systems and pave the way to enhancing the AI efficacy in real world clinical settings.

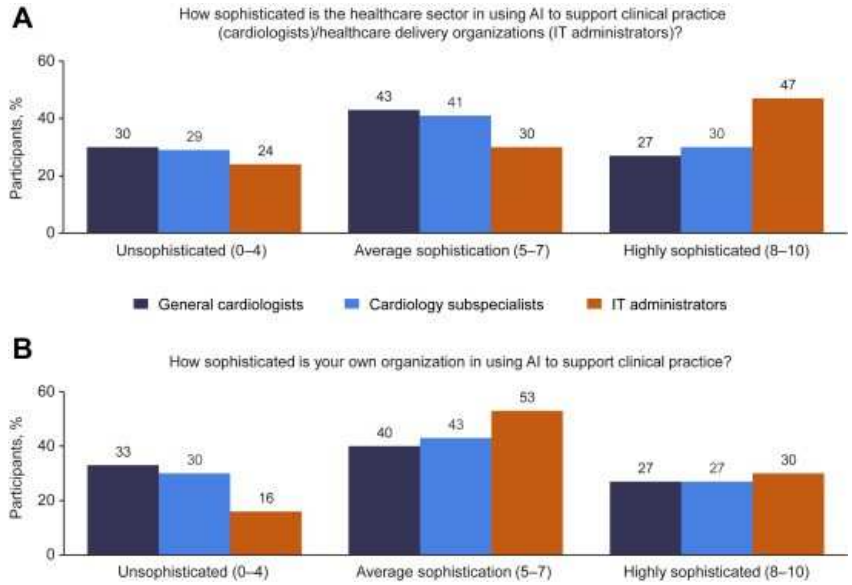

Figure 2.8: Use of artificial intelligence (AI) in health care. Participants' views of the sophistication of the health care industry in general (A), and of their respective organizations (B), in using AI. IT = information technology. [8]

The research An Artificial Intelligence Model for Heart Disease Detection Using Machine Learning [9], 2022, the authors presents an heart disease prediction system based on AI using machine learning algorithms in Python. The research makes use of external databases with patient characteristics like age, cholesterol, and blood pressure. Applying a random forest classifier, the model got 83% accuracy, which proves that ML can be successfully used for heart disease detection. Our work as depicted in figure 2.9 also underlines the suitability of Python in the healthcare domain, as it shows how machine learning can be employed for processing large volumes of medical data in aiding early diagnosis of heart disease.

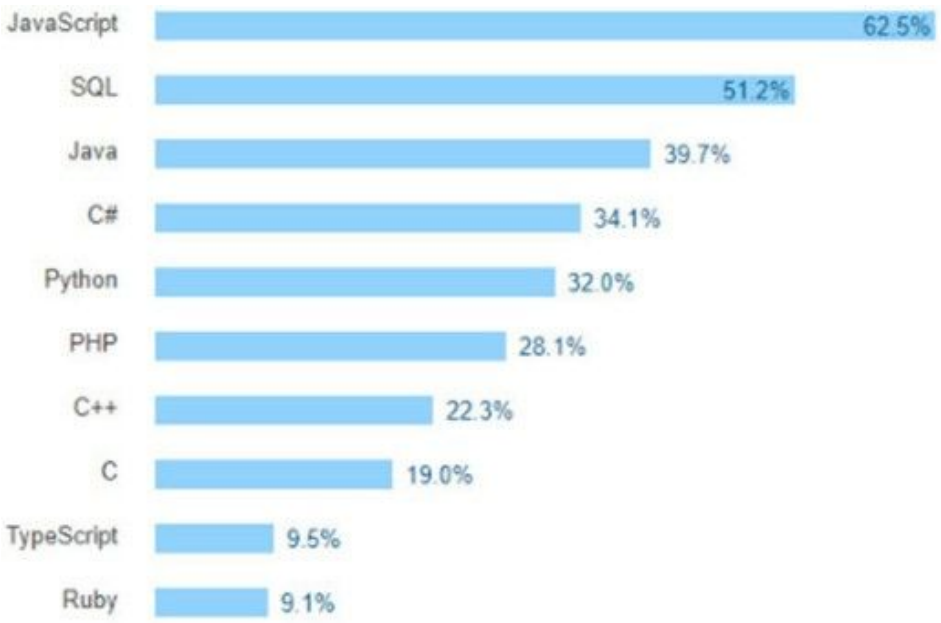

Figure 2.9: Percentage of people preferring Python programming language for healthcare applications. [9]

2.4 Comparative Analysis

The comparative analysis of all the reasearch related to AICardia is give in table 2.1

Table 2.1: Related Literature Review

Paper	**Work**	**Dataset**	**Observation**
A Novel Light U-Net Model for Left Ventricle Segmentation Using MRI [1], 2023	Left ventricle segmentation for cardiac MRI using a lightweight U-Net model.	MICCAI 2009 dataset	Achieved 97.7% Dice coefficient, 92% accuracy. Uses histogram-based enhancement for improved segmentation.
A CNN-Based Model for Heart Disease Detection, 2024 [2]	CNN-based classification of heart disease, analyzing MRI data with caffeine intake analysis.	Enugu State University & CAD Kaggle (90,500 samples)	Achieved 94% accuracy. Introduces caffeine intake as a novel risk factor.

Paper	Work	Dataset	Observation
Cardiologist-Level Arrhythmia Detection with CNNs, 2017 [3]	Deep learning model trained on ECG waveforms for arrhythmia detection.	Private dataset (64,121 ECG records from 29,163 patients)	Outperformed cardiologists in detecting arrhythmias.
Cedars-Sinai ECG AI, Ongoing [14]	AI model for sudden cardiac arrest prediction using ECG patterns.	Hospital patient records (PRESTO)	AI identifies hidden ECG patterns linked to cardiac arrest.
Machine Learning for Real-Time Cardiac Arrest Prediction, 2018 [4]	Predicts in-hospital cardiac arrest using heart rate and EHR data.	EHR dataset (NHS Foundation Trust)	Machine learning detects risk from patient vitals and history.
Artificial Intelligence–Enhanced Electrocardiography for Accurate Detection of Heart Disease (2024) [5]	Enhancement of ECG interpretation using AI for heart disease detection. It generally relates to the analysis of IV/EV for ECG leads.	The excerpt mentions "12, 6, single lead" data but doesn't provide specifics about the patient data used for the models	Highlights AI's potential to improve cardiovascular illness diagnosis and patient care, but notes further development is needed. Emphasizes the need for interoperability, data exchange, and addressing ethical concerns for AI in ECG.

Paper	Work	Dataset	Observation
Prospects for Cardiovascular Medicine Using Artificial Intelligence (2021) [6]	Development of various types of cardiovascular AI for evaluating examinations like ECG and imaging.	It provides an overview and introduction.	AI has been developed for evaluating examinations such as X-rays, ECG, echocardiography, and computed tomography, showing potential in cardiovascular medicine.
Artificial Intelligence and Cardiology: Current Status and Perspective (2021) [7]	Reviews the application of AI in cardiology, focusing on ECG and electrophysiology.	Mentions digital medical data including chronological data like pulse waves and ECGs.	Notes the rapid explosion of AI research in medicine, including cardiology. Highlights AI's potential for new insights from ECGs, such as age and sex prediction. Emphasizes the importance of trust in AI (accurate, robust, explainable) and clarifying the purpose of AI model development.

Paper	Work	Dataset	Observation
Artificial Intelligence–Enabled Tools in Cardiovascular Medicine (2023) [8]	Qualitative study on cardiologists' and health IT administrators' perceptions and use of AI in cardiac care.	Interviews and focus groups with cardiologists and health IT administrators	Explored awareness, perceptions, value, potential uses, and challenges of AI adoption. Found that while AI's potential is recognized, clinical implementation is still limited. Identifies lack of trust and interoperability as significant barriers.

Paper	Work	Dataset	Observation
An Artificial Intelligence Model for Heart Disease Detection Using Machine Learning (2022) [9]	Development of an AI-based heart disease detection system using machine learning algorithms in Python.	External databases with features like Chol, Free tops, sex, and age Dataset with attributes including age, gender, blood pressure, cholesterol	Developed a Python application using libraries like Matplotlib, NumPy, Pandas. Used a random forest classifier algorithm with approximately 83% accuracy over training data. Found K-neighbor classifier showed 87% score on their dataset. Highlights Python's suitability for healthcare due to ML/AI capabilities and HIPAA compliance.

Chapter 3

Requirement Specifications

In this chapter, the detailed requirements and specifications for the AICardia system are outlined. This includes the overall system description, its core functionalities, assumptions, and the technologies required. The chapter also describes the system's use cases and provides a clear representation of its functionality from the user's perspective.

3.1 Existing System

3.1.1 Existing Systems for Cardiac MRI Analysis

There are various AI-based solutions that facilitate the analysis of the cardiac MRI such as automated segmentation of cardiac components, classification of heart diseases and predictive models of the heart diseases. These solutions utilize deep learning methodologies, specifically Convolutional Neural Networks (CNNs) to analyze MRI scans and aid doctors.Below are a few of the related systems:

- **CardioNet**
  - **Overview**: CardioNet is an AI-based framework for automated cardiac MRI analysis, to our best knowledge, for the first time, tailored to cardiac region segmentation (left ventricle (LV), myocardium and epicardium). Employing a blend of CNNs and processing (image) techniques, the system segments the main heart structures and derives significant cardiac parameters, such as End-Diastolic Volume (EDV) and Ejection Fraction (EF).
  - **Limitations**: Despite that CardioNet has shown a promising result for LV and myocardium segmentation, it is still difficult for it to segment small or occluded cardiac structures, since the diversity of dataset and the real-time analysis are limited. Furthermore, the deployment in hospital in real world

is still challenging due to the compatibility issue with clinical system and the interpretability of the AI models which may impede clinical trust and adoption.

- **HeartFlow**
  - **Overview**: HeartFlow is a commerical product based on AI that evaluates coronary CT angiograms for coronary artery disease (CAD). It generates a 3D model of a patient's coronary arteries and applies computational fluid dynamics (CFD) to model blood flow to assess the impact of the blockage, whether mild, moderate or severe. Like this system, the researchers' approach uses machine learning and a data-centric model to estimate heart disease risk.
  - **Limitations**: HeartFlow provides accurate results for coronary arteries but applies to CT angiograms only, excluding magnetic resonance imaging (MRI) scans, which are commonly used to get more detailed views of cardiac structures. Its use is also restricted by cost and availability for hospitals in under-resourced areas. In addition, the system needs precise input data (ie, coronary CT scans), and any deviation in scan quality or patient positioning may affect the model's performance.
- **MRI HeartSeg**
  - **Overview**: MRI-HeartSeg is an opensource AI framework for fully automated left ventricle segmentation in cardiac MRI. It is based on U-Net architecture and produced promising results on public cardiac MRI datasets (MICCAI 2009, ACDC). This product is for research use. It may assist in automating segmentation tasks, enabling radiologists to save time.
  - **Limitations**: Although MRI-HeartSeg yields good segmentation results, it cannot be applied to the whole range of scan qualities and the full patient variability. The model is not fully integrated into clinical practice workflows, which limits the use in a real-world setting. Moreover, it does not perform diagnosis of heart diseases directly from the segmented MRI scans, which is a task AICardia is designed to realize.
- **QMedical Imaging**
  - **Overview**: QMedical Imaging provides an AI tool which enables fully automated processing of cardiac MRI to detect diseases including cardiomyopathies and heart failure. The model takes advantage of a deep learning model, which classifies various heart disease types by using the segmentation output and also computes important cardiac measures such as EF and EDV.
  - **Limitations**: Although QMedical Imaging has proven effective in classifying diseases, it is very sensitive to the input data quality. The QCs need to be

performed on high-quality, noise free MRI images to obtain reliable results. Besides, it has difficulties in multi-modal data fusion which means the model-design can not be naturally extended to fuse data from other diagnostics, e.g. ECG or EHR that may help to better distinguish more complicated diseases. Moreover, the system still needs manual confirmation by cardiologists, which precludes usage in real-time clinical practice.

3.1.2 Limitations of Existing Systems

While there have been significant advancements in AI-based cardiac MRI analysis, several limitations persist:

- **Data Quality and Variability**: Most of the current methods like CardioNet, MRI-HeartSeg, etc are prone to the quality of MRI scans, patient orientation, and resolution of the scan. Bad quality scans or protocols may result in mis-segmentation or mis-classification, and this may impact on the performance of such systems in clinical environment.
- **Real-Time Integration**: Many of the existing solutions such as HeartFlow and Q Medical Imaging are not fully embedded into the clinical workflow, therefore they are less applicable for on-the-fly analysis in hospital environments. Such systems usually need extra time for post-processing and verification, which may result in a delay of diagnosis and treatment.
- **Interpretability and Trust**: One of the major challenges for AI-based systems in healthcare is the low interpretability of models. Systems such as CardioNet and QMedical Imaging are black-box models, which means that it is not clear to healthcare providers how the AI arrived at its decisions. This opaqueness may erode clinicians' trust and prevent the diffusion of AI-based diagnostic tools in the clinical routine.
- **Limited Disease Classification**: Most of the existing methods are for segmentation only and have no disease classification. For instance, MRI-HeartSeg achieves excellent result in segmentation, but it does not identify diseases such as hypertrophic cardiomyopathy or myocardial infarction from MRI images. AICardia seeks to close this gap by integrating segmentation and disease classification in one workflow.
- **Data and System Scalability**: Scalability continues to be an issue for many systems as they tend to be trained against particular datasets. Consequently, their generalization to unseen patients and different image protocols and the resulting robustness and reliability in different clinical environments may be limited.

3.2 Proposed System

The AICardia system is based on deep learning techniques to facilitate the processing of cardiac MRI images. The method uses CNNs for segmentation of the left ventricle (LV) and myocardium and classification of heart disease, allowing quick and reliable diagnostic results for medical experts.

The system will be implemented as an Android mobile application that allows medical experts to upload MRI scans while on the go using their smartphones, obtain segmentation results in a real-time manner, and evaluate the condition of the heart. This mobile first design makes the tool accessible to cardiologists and radiologists at any location and time which can prompt more efficient and timely diagnoses.

The smartphone application will leverage deep learning models, which are going to be trained on publicly accessible datasets such as the MICCAI 2009 Sunnybrook or the ACDC dataset. Then, these models will be trained to segment the left ventricle and other cardiac structures, and to classify a range of heart diseases, including hypertrophic cardiomyopathy and myocardial infarction.

3.2.1 Accessibility

Its mobility and usability are a benefit of this mobile app as the users can upload MRI scans and analyze them directly on their smartphones. This allows for flexibility among healthcare providers, particularly for those in resource-constrained environments or in remote locations, where access to conventional desktop-based systems is minimal.

In addition, real time analysis enables the health care provider to obtain immediate results, enhancing decision making and patient management. As it is a mobile system, it can be seamlessly integrated into clinical routines, making diagnostic support for cardiac disease available at the point of care.

3.2.2 Deployment Areas

The AICardia mobile app can be implemented across several areas:

- **Cardiac MRI Segmentation and Disease Classification:** The app can be applied for cardiac MRI image segmentation and disease classification, which can improve the diagnosis efficiency of cardiac diseases.
- **Healthcare Providers**: The app could aid cardiologists, radiologists and clinicians in minimizing the time required for manual segmentation and increasing diagnostic

confidence. Time is of the essence in clinical practice, and this instrument can be of great assistance in a busy clinical setting.

- **Research and Education** Researchers and medical professionals can also employ the technology to gain deeper insights into cardiac conditions and enhance diagnostic endeavors. It also allows easy storage of segmented MRI for analysis and research.
- **Telemedicine:** The app promotes telemedicine through remote diagnostics, making this particularly useful in resource-limited regions. People in rural areas can also have their MRI scans processed and evaluated by specialists from a distance.

3.3 Requirement Specifications

3.3.1 System Preconditions and Dependencies

- **Dataset**: Our system will be based on high-quality labeled datasets for training. Initial training will be performed on the MICCAI 2009 Sunnybrook or ACDC datasets. It is expected that these datasets can be sufficiently large to train accurate models.
- **Hardware Resources**: To run the system, Minimum of 16 GB RAM for efficient processing.
- **Software Environment**: The system will be implemented in python using TensorFlow and Pytorch, two popular deep learning frameworks. The application' s front end will be developed in Flutter for access on mobile.
- **External Dependencies**: The project assumes the availability of the public datasets and might depend on third party APIS for additional functionalities (e.g., data preprocessing).

3.3.2 Functional Requirements

The functional requirements of the AICardia system are as follows:

3.3.2.1 User Registration & Login [FR 01]

- The system shall enable users to register, and log in and out securely.
- The analysis functions are available only to authenticated users.

3.3.2.2 MRI Upload [FR 02]

- Users need to be able to submit their cardiac MRI files from their own computer.
- The system should support DICOM or NIfTI formats of MRI.

3.3.2.3 Automatic MRI Analysis [FR 03]

- Upon upload, the system should automatically handle processing of the MRI scan without additional user interaction.
- The system must calculate heart measurements from the file that is uploaded.

3.3.2.4 Cardiac Metrics Calculation [FR 04]

- The system must compute and return key cardiac functional measurements, including:
  - End-Diastolic Volume (EDV),
  - End-Systolic Volume (ESV),
  - Ejection Fraction (EF).

3.3.2.5 Disease Classification [FR 05]

- The system shall assign the MRI scan to an appropriate diagnostic category from the set DCM, HCM, MINF, NOR.
- The class prediction should be displayed by the system.

3.3.2.6 Results Visualization [FR 06]

- A visual summary of the results shall be made available by the system containing at least the following information:
  - Cardiac metrics,
  - LV volume curve per frame,
  - Disease diagnosis.
- Results should be presented in a way that is easy to understand and is clear in the app.

3.3.3 Non-Functional Requirements

3.3.3.1 Reliability [NFR 01]

- The system should be robust, the segmentation and classification results should always be accurate and consistent with the data.
- The application shall execute without crashing or suffering major malfunctions while in use.

3.3.3.2 Performance [NFR 02]

- The proposed algorithm must process such large amounts of MRI scan data in a timely manner.
- The processing of MRI scans should be executed in a way that results are available almost in real-time or at least within a reasonable time delay.

3.4 Use Cases and Descriptions

The use case UML diagram gives a graphical view of different layers of the application and how they relates to one another. The user and more important use cases that represent the entire framework are captured in the Following is Figure. The functional requirements describe what the system should do, and the users view of actions that the system must take ar clearly outlined. In short, it shows the customer aspirations and how should the system behave to fulfill them.

Use case diagrams specify the behavior of a system by showing the actors that interact with the system, the use cases in which they participate, and the goals for each actor. Such type of UML diagrams enables the system developers to properly design the system as well as allows them to foresee any potential problem at an initial stage by looking at the system as from view of the customer.

On the whole, a use case diagram focuses on what goals the system offers and what requirements it has, giving a high-level or external view of system operation. In short, it summarizes the essential details of the system and the users who interact with it. This section of the report presents all the ways users will engage with the application.

3.4.1 Use Case Diagram

The Figure 3.1 below illustrates the whole process of the system from user's point of view according to functional requirements.

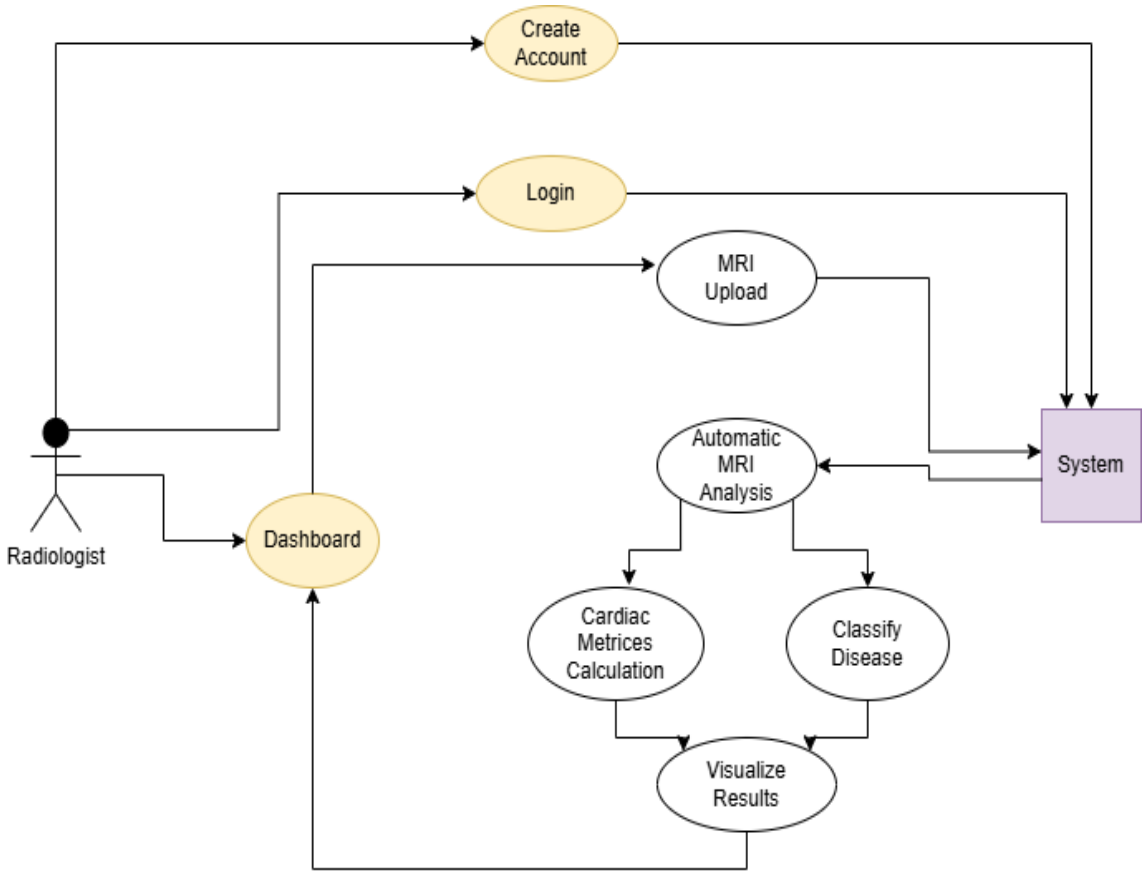

Figure 3.1: Use Case Diagram

3.4.2 Tabular Description for Use Case Diagram

The Table 3.1 below defines the whole description of the use case outlined above according to the functional requirements.

Table 3.1: Tabular Description for Use Case Diagram

Use Case ID	UC-01
Use Case Name	MRI Upload and Automated Cardiac Analysis.
Actor(s)	User (Cardiologist, Radiologist, Healthcare Provider)
Data	Cardiac MRI file (DICOM/NIfTI), extracted cardiac metrics (EDV, ESV, EF), LV volume curve, disease classification.
Trigger	The user wants to analyze a cardiac MRI scan using the AICardia mobile app to obtain automated analysis, cardiac functional metrics, and disease prediction.
Pre-condition	The user is authenticated and logged into the AICardia mobile app.
Assumption	The user has access to a valid MRI file on their device and an active internet connection for processing.
Description	1. The user navigates to the "Upload MRI" option from the app dashboard. 2. The system prompts the user to select a cardiac MRI file in DICOM or NIfTI format. 3. The user selects and uploads the file. 4. The system validates the file format and initiates automatic preprocessing. 5. The FastAPI backend reconstructs the MRI volume and performs LV segmentation on all frames. 6. The system computes cardiac functional metrics, including EDV, ESV, and EF, based on the LV volume curve. 7. The myocardium is segmented at end-diastole to calculate myocardial mass and wall-thickness statistics. 8. Extracted features are passed through the classifier to determine the diagnostic category (DCM, HCM, MINF, or NOR). 9. The backend returns all results, which the app displays as metrics, curves, and disease prediction.
Success	The system successfully processes the MRI scan, generates segmentation outputs, computes cardiac metrics, predicts the disease class, and displays the results clearly to the user.
Post-conditions	The results are displayed on the mobile app dashboard. The user may upload another MRI scan or return to the dashboard. Temporary processing data is cleared automatically after completion.
Comments	The system handles invalid or corrupted MRI files by displaying appropriate error messages. Future versions may include downloadable segmentation masks or report export functionality.

Chapter 4

Design

The main focus of this chapter is the design of the AICardia system. A smart detection tool for heart conditions based on cardiac MRI analysis. System design involves describing the overall architecture, components, interfaces, and constraints necessary to meet the project requirements.

4.1 System Architecture

The architecture of AICardia enables smooth interaction between users and the AI-powered cardiac MRI analysis engine via a mobile application. The system accepts MRI scans, and provides classified heart conditions, and displays results through an intuitive mobile interface.

4.1.1 Context Diagram

In Figure 4.1 below, the context diagram clearly shows the boundaries of our system and the interaction of different parts of the system with each other as the information is exchanged between them.

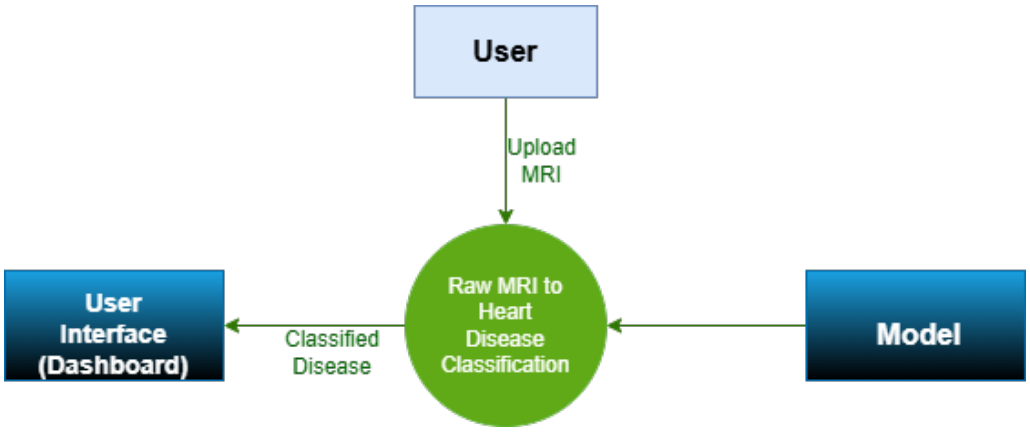

Figure 4.1: Context Diagram

4.2 Design Constraints

4.2.1 Performance vs. Device Resources

- Deep learning model execution on mobile devices is resource intensive; therefore, heavy computations are offloaded to backend servers.
- Mobile device limitations such as memory, CPU constraints, and battery life restrict local processing capabilities.

4.2.2 Technology Stack Constraints

- Flutter is used for mobile development, but limiting certain native image processing options.
- Python with TensorFlow or PyTorch is used for backend model execution.
- FastAPI manages communication between the mobile app and backend, and requires strict input validation to handle diverse MRI formats.

4.2.3 Compatibility Constraints

- Differences in MRI scanning protocols across institutions may impact input consistency.
- Ensuring compatibility across various mobile devices requires extensive testing for screen sizes, performance, and file systems.

4.3 Design Methodology

In order to AICardia objectives, AICardia makes use of a hybrid method. Design Driven Modeling We followed a Model Driven Architecture (MDA) approach in the design that consists of two-level modeling, PIM and PSM. MDA enables a clear distinction between the business logic and its execution technology, which improves the flexibility, scalability and maintainability of the system. During the execution phase, the project will adopt an iterative/predictive life cycle model; also known as incremental development.

4.4 High Level Design

High level design describes the system architecture and main components AICardia has and how these are structured and how the main components of the system relate to each other. It eliminates the low level design details, focusing on the higher level structure including the interrelationship among the various function part of the system.

The AICardia design at a high level is modular consisting of various layers which cooperate to enable the processing and analyzing of MRI images for detection of heart diseases. These are the Mobile User Interface (UI), Image Processing (IP) Module, and AI Classification Engine that are combined to empower the healthcare practitioner with a well-designed, user friendly, and effective diagnostic application.

The modular structure of the system also means that each part of the system can be designed, developed, tested and maintained independently of the others whilst the system as a whole still appears to be cohesive. In addition, this organization facilitates scalability, so design changes can be added in the future without affecting the operation of the basic system.

4.4.1 Component Diagram

The diagram 4.2 shows how the system takes a raw MRI from the user, sends it to the disease classification module, and returns the predicted condition through the user interface. The classifier relies on a trained model, which is created using a labeled MRI dataset during the model training process.

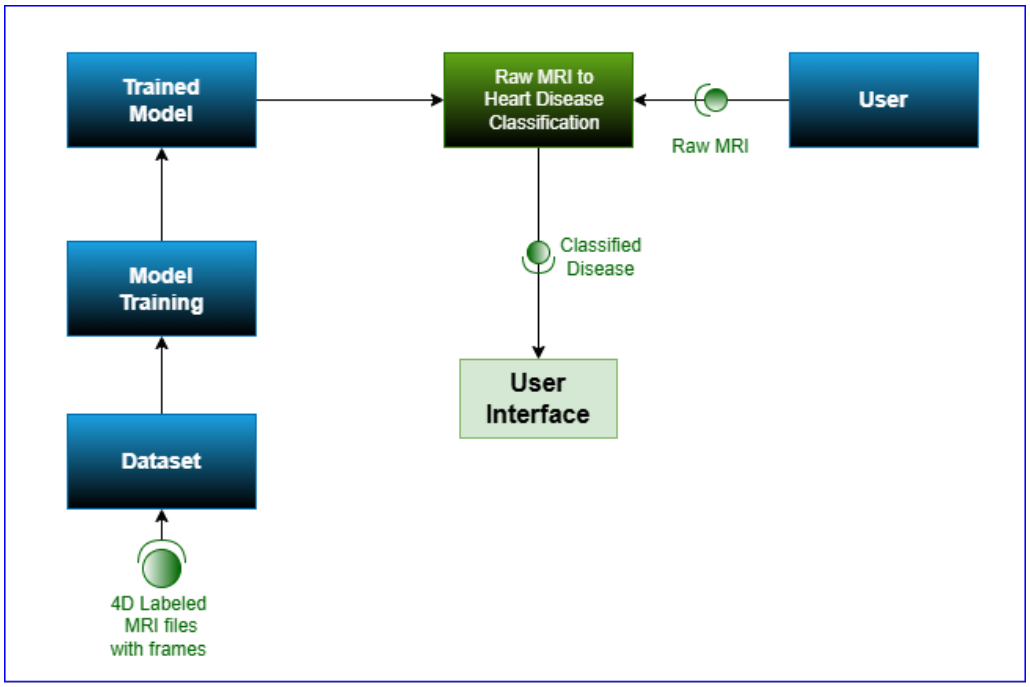

Figure 4.2: Component Diagram

4.4.2 Sequence Diagram

The main sequence diagram in figure 4.3 shows the end-to-end workflow where an uploaded MRI passes through preprocessing, segmentation, metric extraction, and classification modules before final results are returned to the user dashboard.

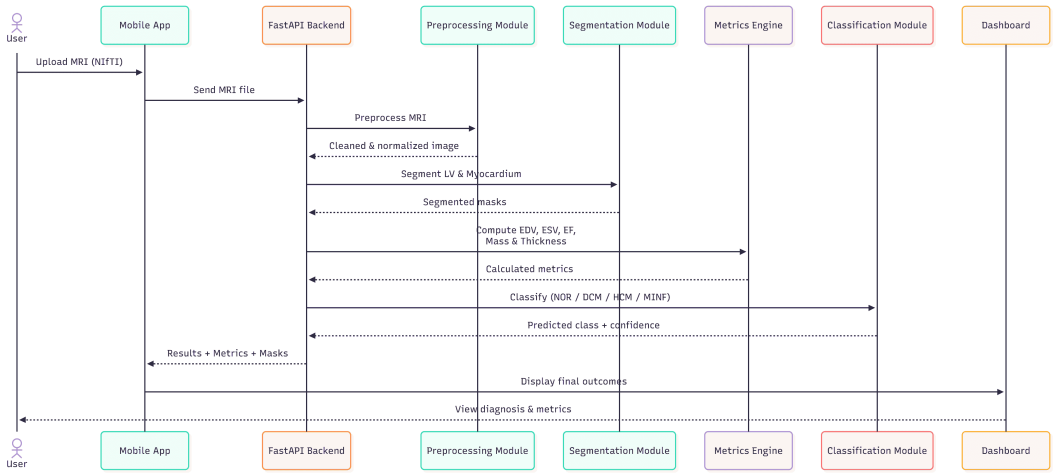

Figure 4.3: Main Sequence Diagram

4.4.2.1 System Sequence Diagram

The system sequence figure 4.4 shows how the system fully automates preprocessing, segmentation, metric extraction, and diagnosis after the user uploads an MRI scan.

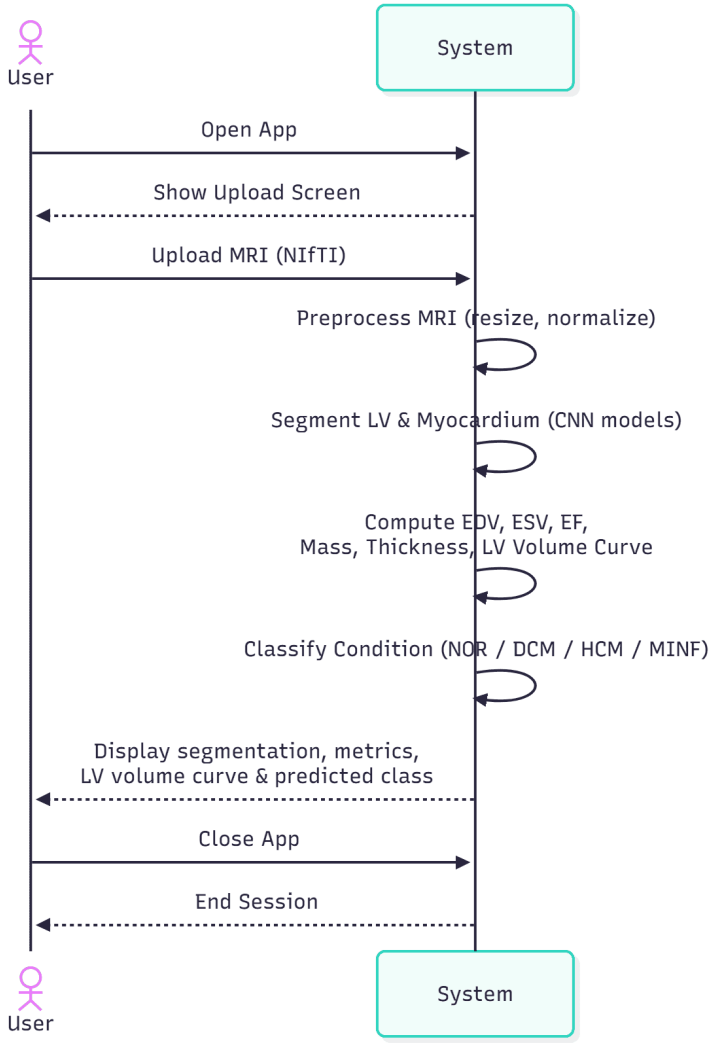

Figure 4.4: System Sequence Diagram

4.4.2.2 Preprocessing Sequence Diagram

The sequence diagram of preprocessing module is given below in figure 4.5 shows how the preprocessing module loads raw ACDC MRI data, detects the cardiac ROI, splits patients into train/validation/test, and saves standardized HDF5 files for later segmentation and classification.

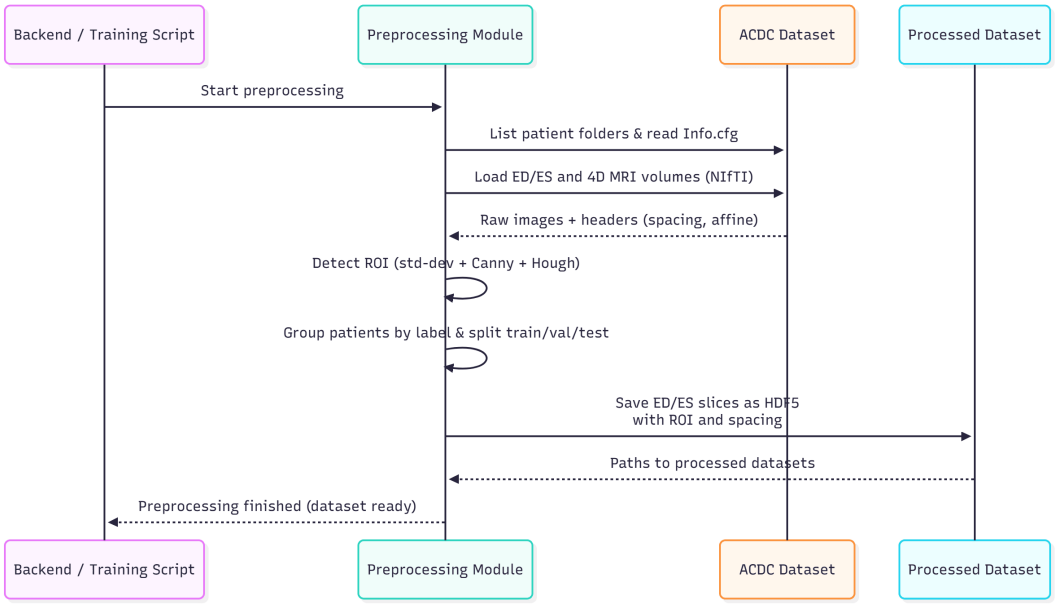

Figure 4.5: Preprocessing Sequence Diagram

4.4.2.3 Segmentation Sequence Diagram

The sequence diagram of the segmentation module is shown in figure 4.6 which shows how the system internally segments MRI to generate LV and myocardium masks for downstream metrics and classification.

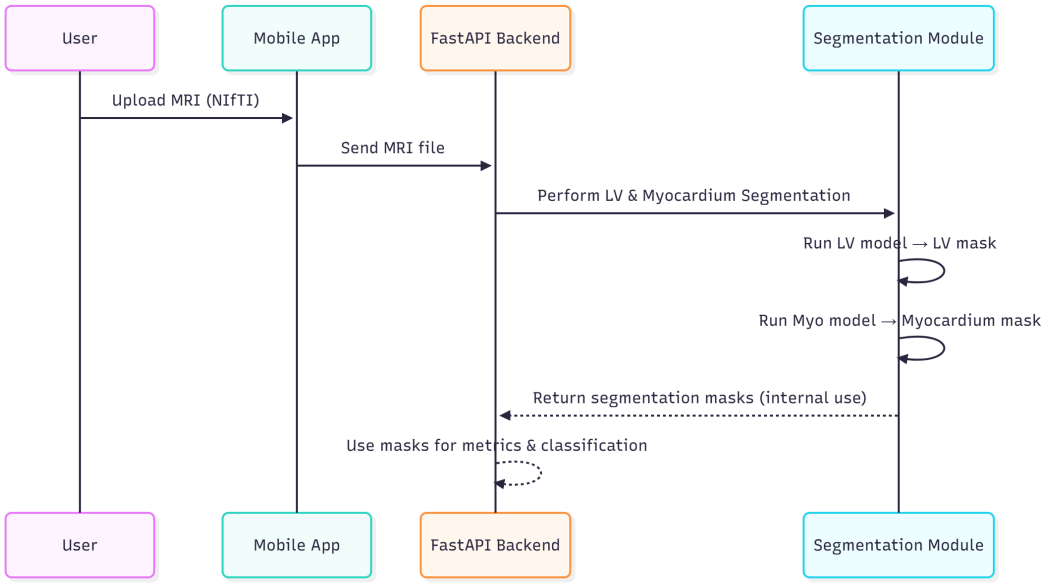

Figure 4.6: Segmentation Sequence Diagram

4.4.2.4 Classification Sequence Diagram

The sequence diagram of classification module is given below in figure 4.7 illustrates how extracted cardiac features are sent to the classification module, which predicts the patient's heart condition and returns the diagnosis to the dashboard for display.

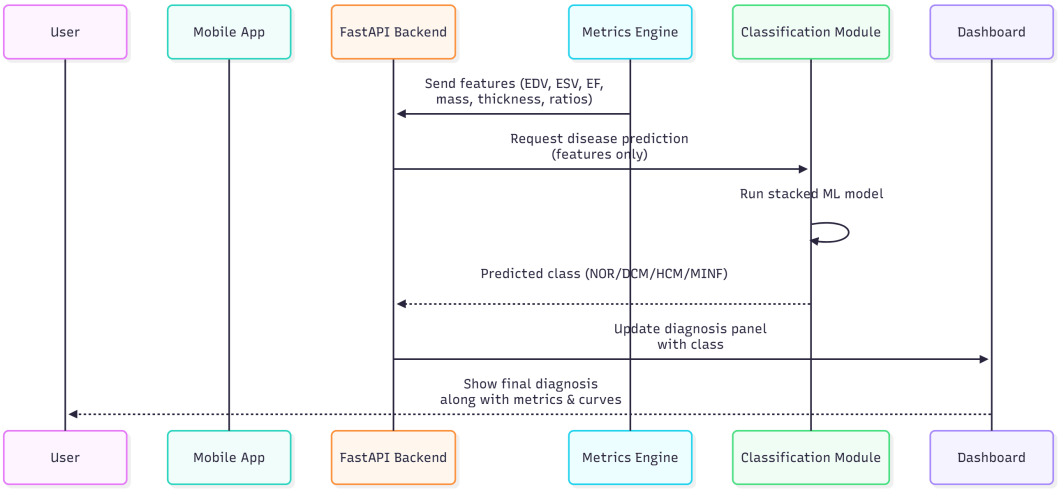

Figure 4.7: Classification Sequence Diagram

4.4.3 Data Flow Diagram

A data flow diagram (DFD) visually illustrates how information moves through a system. It focuses on the pathways data follows and where it comes from, where it is sent, and how it is stored along the way. The diagram is designed to be descriptive, showing the inputs, outputs and data stores involved in each process, and grouping the processes into a data flow subject area. There is a general set of symbols and notations representing those elements and their relations. In general, a DFD allows one to get a clear understanding of the flow of data at a very high level in the system. The high level design is shown in the following data flow diagram in figure 4.8, depicting the links among the modules and the data flow in the system.

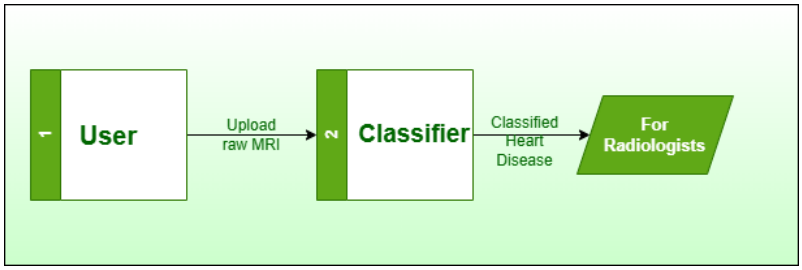

Figure 4.8: Data Flow Diagram of AICardia

4.5 Low Level Design

Low Level Design describes the inner workings of the system with details on each of the component. It decomposes the high level modules into classes, functions, data structures and exact flow of work. LLD describes the interaction between components, data processing and error processing and provides a clear technical design to allow developers to start coding and exactly how to develop the system.

4.5.1 Class Diagram

Figure 4.9 presents the internal structure and the classes, methods and interactions for each module, resulting a basis for the implementation and the construction of system.

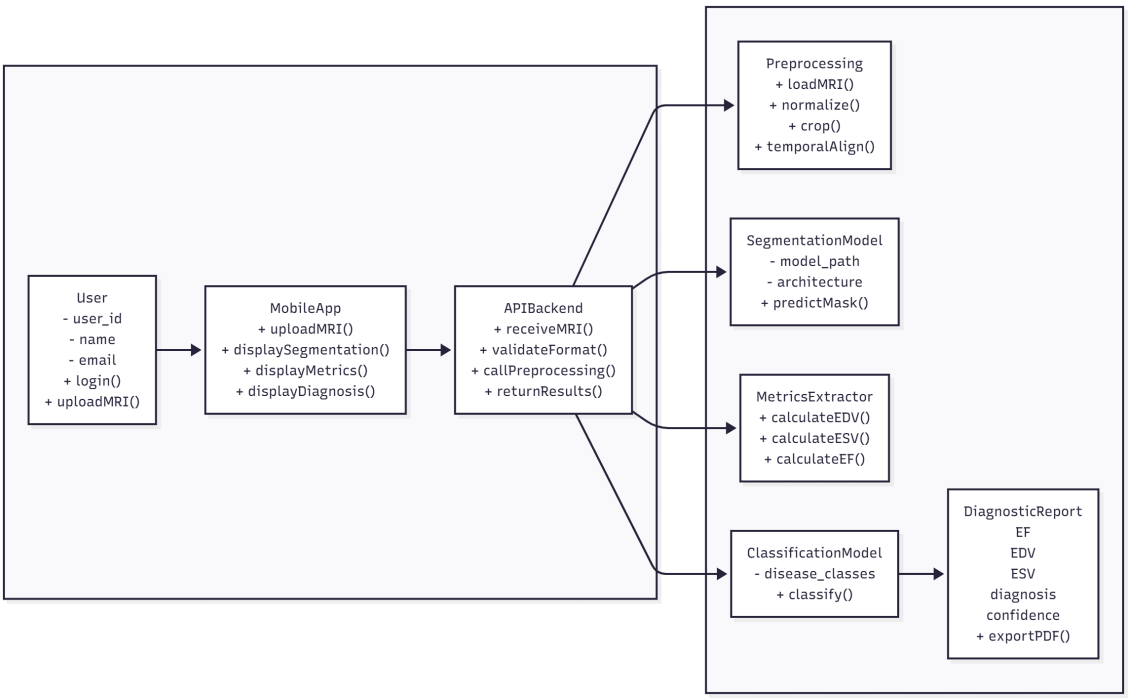

Figure 4.9: Class Diagram of AICardia

4.5.2 Activity Diagram

In Figure 4.10, the activity diagram depicts the workflow of a user towards ending up at the end result. It describes the different activities of the system. This kind of the UML diagram can be used for describing the behaviour of the system in time, so it represents the dynamic flow of the system. In principle, an activity diagram is a flowchart that depicts the flow from one activity to another.

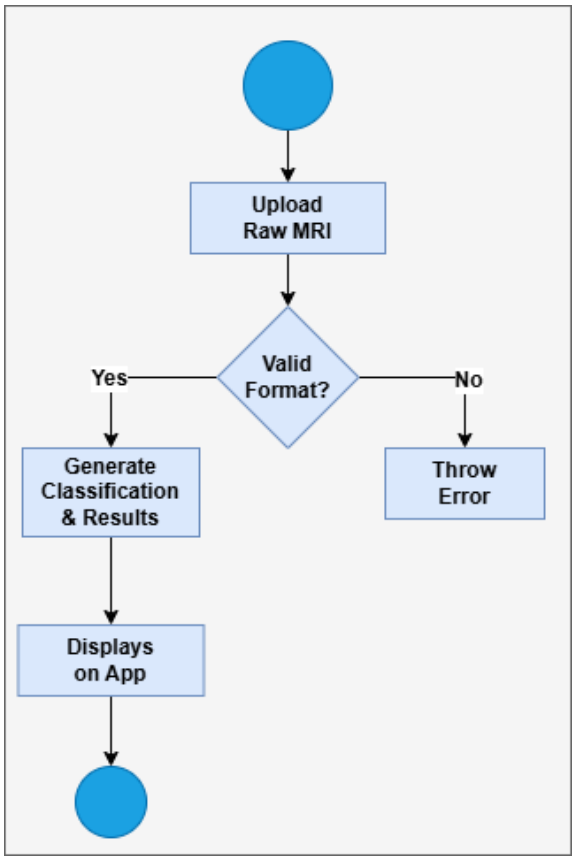

Figure 4.10: Activity Diagram of AICardia

4.5.3 Flowchart for Pseudocode

The flowchart of the methodology with a level of detail as illustrated in the figure 4.11 of how AICardia processes an MRI report is described in the following. The user has an ability to use an interface (or client) that uploads a scan, the system validates the file, and if all is well, it goes through preprocessing, segmenting, and computing heart metrics. These outputs are then utilized to estimate the heart state and the final diagnosis is sent to the user. If there is an error such as invalid or corrupted MRI, the error is instantly displayed by the system. When the status results are shown, the system returns to idle mode to wait for the next scan.

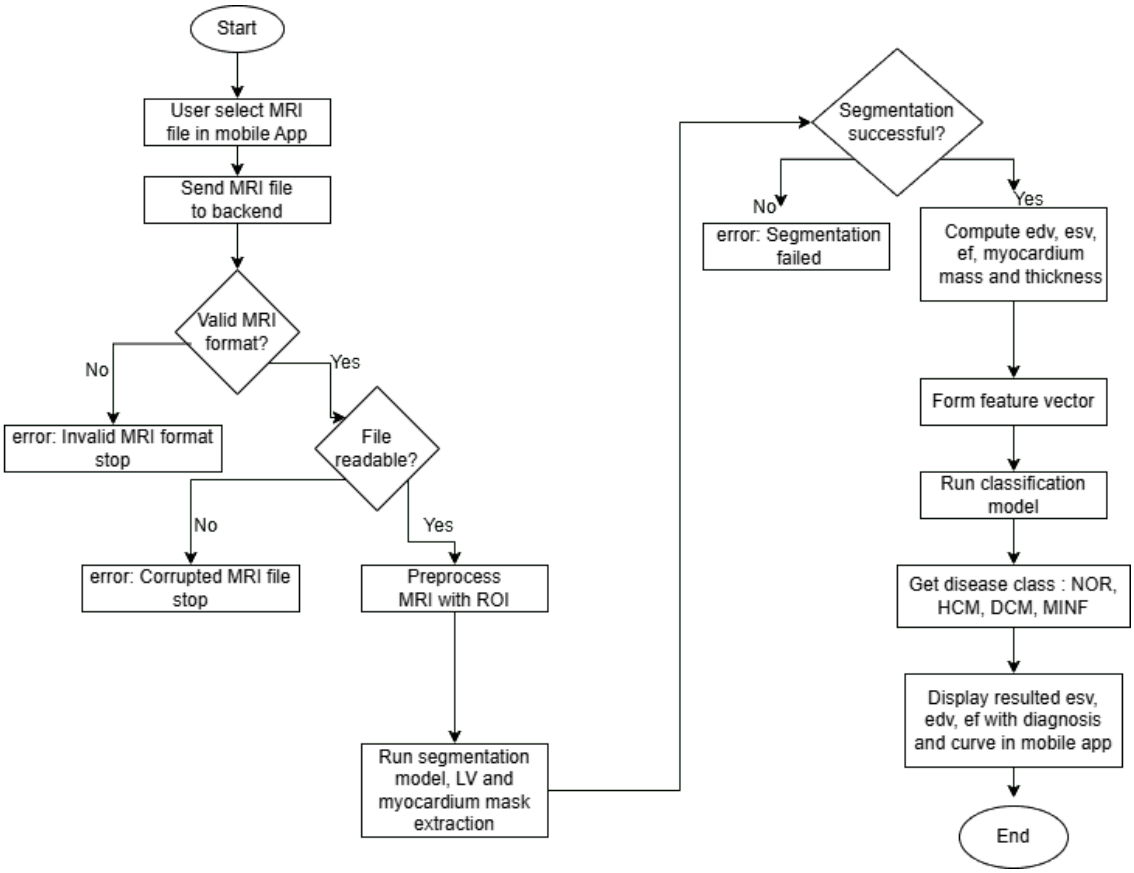

Figure 4.11: Flowchart of AICardia pseudocode

4.6 GUI Design

The accessibility of the whole system through the GUI makes it a crucial element of the AICardia app since it will directly affect the final output of our product. A simple, easy to use interface is essential for improving user experience and ensuring the app's effectiveness. They may look different from end product, based on feasibility to achieve proper usability and look

4.6.1 GUI Prototype

The high fidelity prototype of the app was created , showcasing the design and layout. It includes wireframes for the core features, such as uploading MRI images, viewing classification results, and displaying key cardiac metrics like EF, EDV, and ESV.

4.6.1.1 Logo

The sample illustration of logo given below in figure 4.12 shows how the logo will look like for AICardia.

Figure 4.12: Logo of AICardia

4.6.1.2 Landing

The sample illustration of landing page given below in figure 4.13 illustrates about the idea of front page for our system.

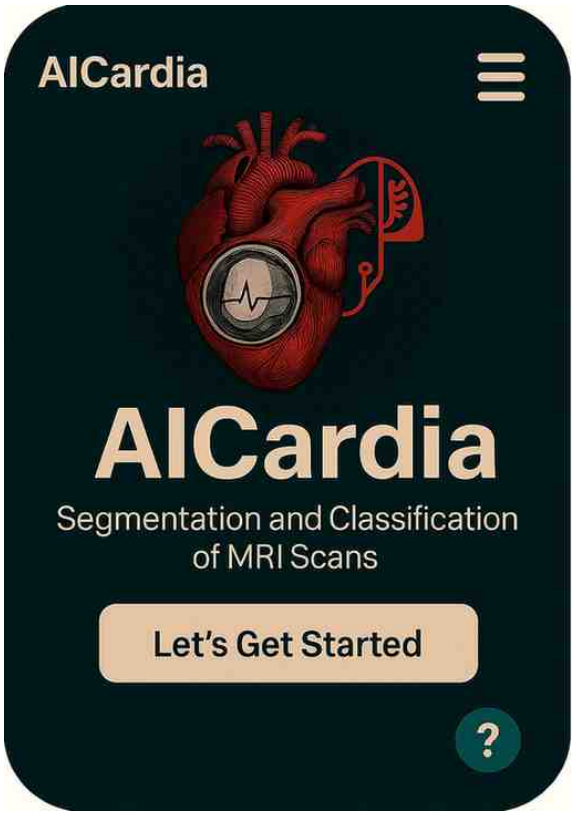

Figure 4.13: Landing Page of AICardia

4.6.1.3 Login/Sign Up

The sample illustration of Login/Signup given below in figure 4.14 gives an idea of login and signup pages for our actual product design.

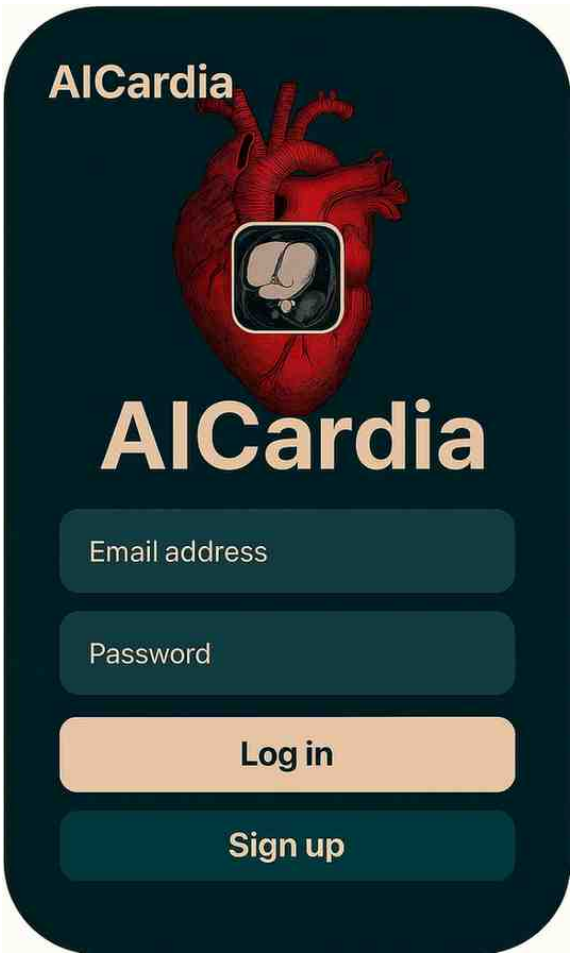

Figure 4.14: Login/Sign Up Page of AICardia

4.6.1.4 Navigation Page

The sample illustration of Navigation Page given below in figure 4.15 shows the navigation panel to be applied in the actual system.

Figure 4.15: Navigation Page of AICardia

4.6.1.5 Browse MRI Image

The sample illustration of Uploading MRI Image Page given below in figure 4.16 shows the idea of the main page where user will upload MRI file.

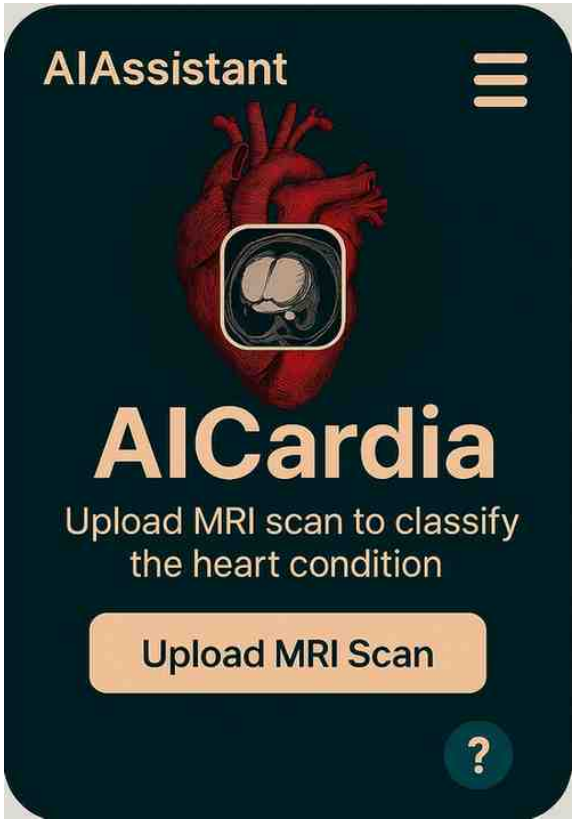

Figure 4.16: Uploading MRI Image

4.6.1.6 Results

The sample illustration of Classified Results page given below in figure 4.17 shows the diagnosed results of the patient.

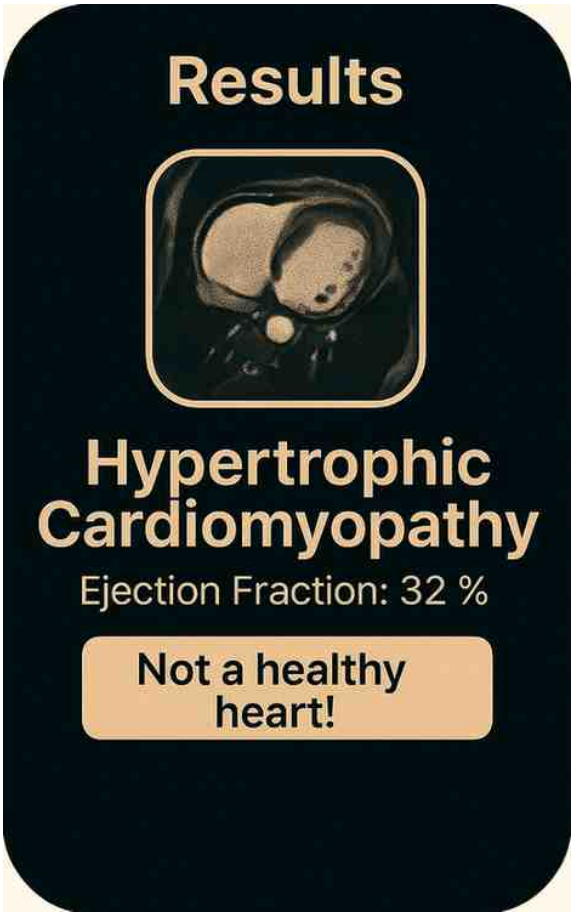

Figure 4.17: Results of Classification

4.6.1.7 Feedback Page

The sample illustration of Feedback Page given below in figure 4.18 shows the prototype that how we will design our actual product feedback page.

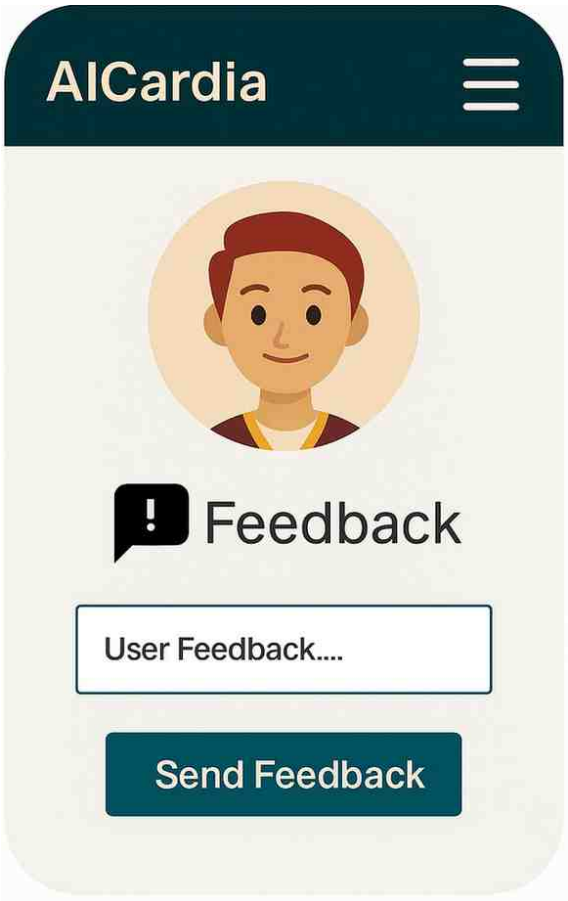

Figure 4.18: Feedback of Classification

The design of the application should be user centered. Application users, who are the largest stakeholders in any software system, need to be highly satisfied for the system to be successful. The interface should be simple and intuitive. Any complexity in the interface can lead to user frustration. The design should be clear and concise. Additionally, if the user provides incorrect input or clicks a button by mistake, a back button should be available to allow the user to return to the previous action. Error messages should be displayed when necessary.

4.6.2 Intuitive Navigation

According to GUI principles, an application must have a simple and easy-to-use design. The user should always know the next step to take. This happens with a sane design in which all the next-step option buttons are visible and within click distance.

4.6.3 Clear and Informative Feedback

The application must notify the user after every operation. On success an alert or a pop-up should notify the success. If the operation is cancelled or failed, an error message

describing the problem should be displayed by the system. The GUI design principle is not only to echo, but also give meaningful echo to effectively guide users.

4.6.4 Consistency in Design

All elements in the interface must be consistent with each other. This includes consistency in colors, button sizes, text box sizes, and error pop-up messages. Any unnecessary elements should be avoided as they can clutter the interface and detract from the user experience. A clean and organized design helps users focus on the essential tasks.

Chapter 5

System Implementation

Implementation turns a concept into a working system. This chapter presents the end-to-end build of our mobile assistant for cardiologists and radiologists. The app ingests 4D cardiac MRI, and works on 2D which automatically segments the left ventricle and myocardium, derives quantitative metrics (EDV, ESV, EF, myocardial mass, and wall thickness), and classifies each study as DCM, HCM, MINF, or NOR. We have outline the tools that make this possible which are TensorFlow/Keras for model training in PyCharm on the ACDC dataset, a FastAPI service for inference and data handling, and a Flutter client for mobile delivery together with the data flow and evaluation steps. As the chapter 4 covered the interfaces and here we focus on the backend logic that powers the system.

5.1 System Architecture

At runtime the system works as a single pipeline. As shown in the Figure 5.1 below, The Flutter app uploads a cine MRI .nii/.nii.gz or .dcm to the FastAPI service. The server reconstructs the 4D volume and then runs an FCDenseNet LV segmenter across all frames to build the volume time curve, locate ED/ES, and compute EDV, ESV, and EF. Using the ED frame, a second FCDenseNet segments the myocardium and computes mass and wall-thickness statistics (area-weighted mean, p05/p95, range). These LV and MYO features feed a stacked scikit-learn ensemble that returns one of DCM, HCM, MINF, NOR together with a confidence score. The API replies with a compact JSON (metrics, ED/ES indices, LV volumes per frame, and optional overlays) and then the app renders gauges, the LV volume chart, and the diagnosis. Models are trained in TensorFlow/Keras, served by FastAPI, and consumed by the Flutter client.

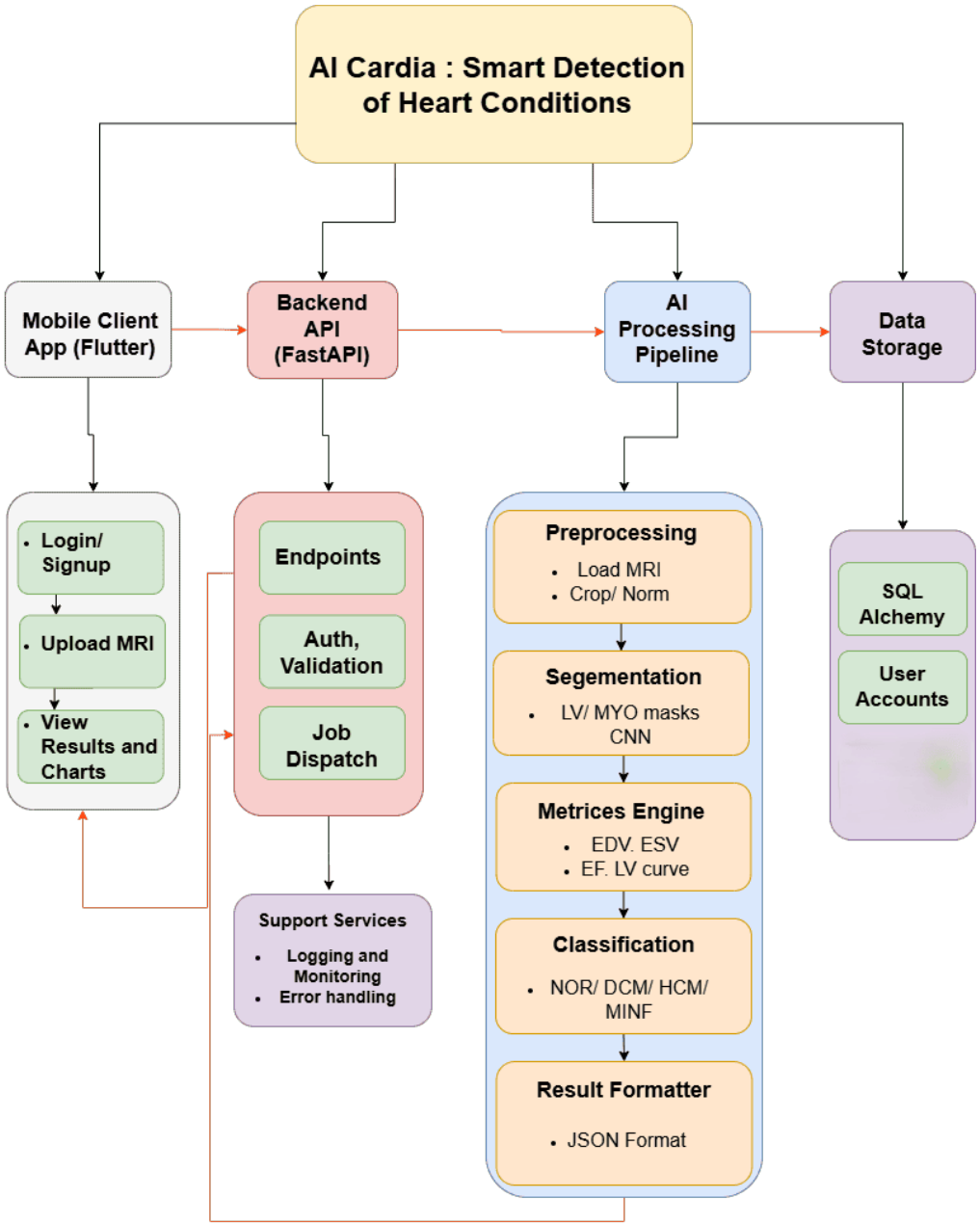

Figure 5.1: System Internal Components

Working across all four conditions proved challenging. A key difference from much of the literature we reviewed is that many studies rely on ground truth of ED/ES frames or manual cues provided with the dataset, and sometimes even use curated myocardium masks to simplify downstream steps and to get perfect results which are quite not realistic. In contrast, our project derives ED/ES directly from the raw MRI by segmenting every frame, building the volume time curve, and taking the true extrema. This choice makes the task harder and our results more modest, but it keeps the system honest and closer to real clinical input where ground-truth timings are not available.

At the early stages of building the project, we assumed to derive LV functional indices alone (EDV, ESV, EF, stroke volume and simple ratios) would be sufficient for four way diagnosis. An initial baseline trained only on LV metrics performed acceptably on function dominated types (e.g., DCM vs. MINF) but getting consistently confused morphology driven cases (notably HCM vs. NOR and some infarct presentations). This observation led us to add explicit myocardium segmentation at ED and to derive structural descriptors (mass and wall thickness: area-weighted mean, p05/p95, range). Incorporating these MYO features materially, improved separability in the preserved EF branch and stabilized the stacked meta learner.

5.1.1 Core Building Blocks

5.1.1.1 Mobile Application Interface

- MRI upload interface that allows users to upload MRI scans (DICOM or NIfTI files) directly from their mobile devices. The interface validates and transmits files to the backend.
- Results dashboard showing EF, EDV, ESV, predicted disease class, and LV volume per frame curve.

5.1.1.2 Image Preprocessing Module

- As shown in the Figure 5.2, the preprocessing module loads 4D cine MRI volumes from the ACDC dataset and reads metadata such as header spacing.
- Automatically detects the cardiac region of interest.
- Crops or resizes slices to a fixed 128×128 resolution.
- Saves the processed slices as structured HDF5/NumPy datasets for training, validation, and testing goes back to FastAPI backend.

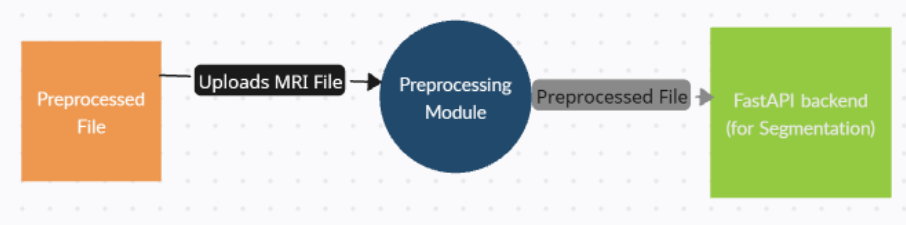

Figure 5.2: Preprocessing on Data

5.1.1.3 Segmentation Module

- As shown in the Figure 5.3, the segmentation module trains a model for left-ventricle (LV) segmentation.
- Trains a separate model for myocardium segmentation.
- Computes LV functional metrics (EDV, ESV, EF) using ground-truth masks and predicted segmentations.
- Runs an automated ED-frame selection and myocardium analysis pipeline that:
  - uses the LV model to identify the frame with maximum LV cavity,
  - segments the myocardium at that selected frame,
  - post-processes the resulting segmentation mask,
  - derives myocardium volume, myocardial mass, and thickness statistics per slice and per patient.

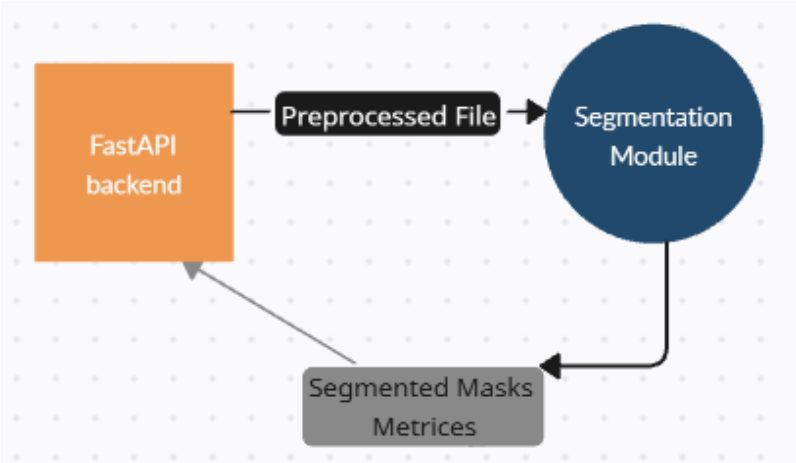

Figure 5.3: Segmentation Module

5.1.1.4 Classification Engine

- As shown in the Figure 5.4, the classification module Performs multi-class classification to categorize cardiac conditions such as NOR, DCM, HCM, and MINF.
- Estimates prediction confidence for each disease class to enhance reliability.
- Generates structured diagnostic reports combining metrics, visualizations, and predictions.

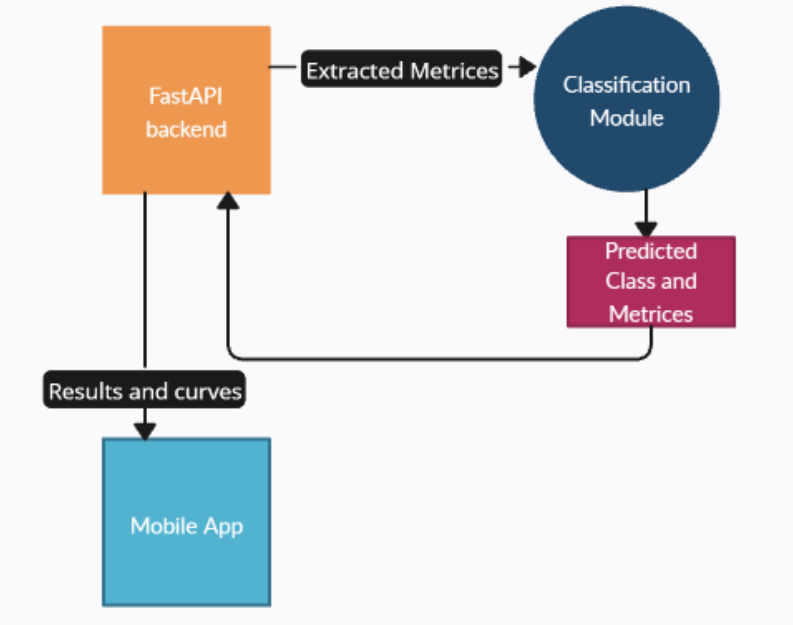

Figure 5.4: Classification Module

5.1.1.5 Data Storage and Database System

- Uses SQLAlchemy as the ORM layer to manage database operations such as managing user data, authentication and processing logs.
- Maintains user data, authentication tokens, and session logs of users.
- Ensures stable storage through indexing.

5.1.1.6 Frameworks, Languages, and Libraries

- Mobile application developed using Flutter [15].
- Backend written in Python due to its strong ecosystem for scientific computing and AI development.
- Deep learning implemented using TensorFlow or PyTorch for segmentation and classification [16].
- Data preprocessing uses libraries such as NumPy, SciPy, and scikit.
- MRI file handling uses NiBabel [17] for NIfTI images and pydicom for DICOM handling.
- FastAPI is used for backend communication, providing asynchronous endpoints, automatic validation, and high performance.
- SQLAlchemy is used for storing user accounts for authentication.

5.2 Responsibilities of the Parts

The whole pipeline is deterministic. The uploaded MRI is streamed to a temporary file, loaded into a 4D array (H, W, Z, T), normalized to [0,1], and center crop-padded to 128×128 per slice to match training. LV segmentation runs for every frame so we can compute per frame LV volumes and detect ED/ES. Those indices drive both reporting and the next stage [17]. As the MYO network runs once, at ED, to keep latency low and to align thickness/mass with clinical convention. The thickness is computed per slice and summarized with area weighting so larger cross sections contribute proportionally. The classifier then maps the full feature set to DCM, HCM, MINF, or NOR with confidence value and displays in the app.

5.3 Interactions and Data Exchange

The mobile client talks to a single endpoint. A POST /predict carries the study as multi-part/form data under the file field. The server returns JSON with EDV, ESV, EF, the LV volume series, ED/ES frame indices, myocardium volume and mass, thickness mean with p05, p95, and range, and the final class with confidence. The errors are surfaced with HTTP 400 and a short message in detail. And the CORS is permissive in development and can be restricted to the production origin.

5.4 Toolchain and Libraries

We used Python and Dart. Models were trained with TensorFlow/Keras[18]. The classifier stack uses scikit-learn. Imaging and numerics rely on numpy, scipy, scikit-image, nibabel and pydicom. The service uses FastAPI [19], with SQLAlchemy for light user management. The Flutter app uses http, file_picker, percent_indicator, fl_chart, and flutter_animate.

5.5 Working Environment

The training and experiments ran in PyCharm with TensorFlow. We have fixed seeds where practical. The main artifacts are LV, myocardium, and four serialized classifiers as, gbm_pipe.joblib, low_final.joblib, high_final.joblib, meta_pipe.joblib loaded at server start. During development we used uvicorn main:app host 0.0.0.0 port 8000 reload, connecting via local network, emulator loopbacks, or a tunnel when off site.

5.6 Implementation Details

5.6.1 Dataset

All models were trained on the Automated Cardiac Diagnosis Challenge (ACDC) dataset[10]. It contains 150 patients, each with a 4D cine MRI scan covering the full cardiac cycle as shown in the Figure 5.5 below. Every patient scan includes around 30 frames and 6–12 slices, providing a complete view of the heart across time. The full ACDC dataset contains around 1,800–2,000 MRI files in total. This includes: 150 patient folders, each patient has 12–15 files (ED/ES images, ground-truth masks, 4D volumes, config files, etc.)

Plus additional metadata files. The dataset also provides manually labeled masks for the left ventricle and myocardium at end-diastole (ED) and end-systole (ES), which are used for training the segmentation models. All spacing information from the MRI headers is preserved so that volumes and thicknesses are computed in true physical units. It provides short axis cine MRIs with labels for DCM, HCM, MINF and NOR. We used header spacings throughout so volumes and thicknesses are reported in physical units.

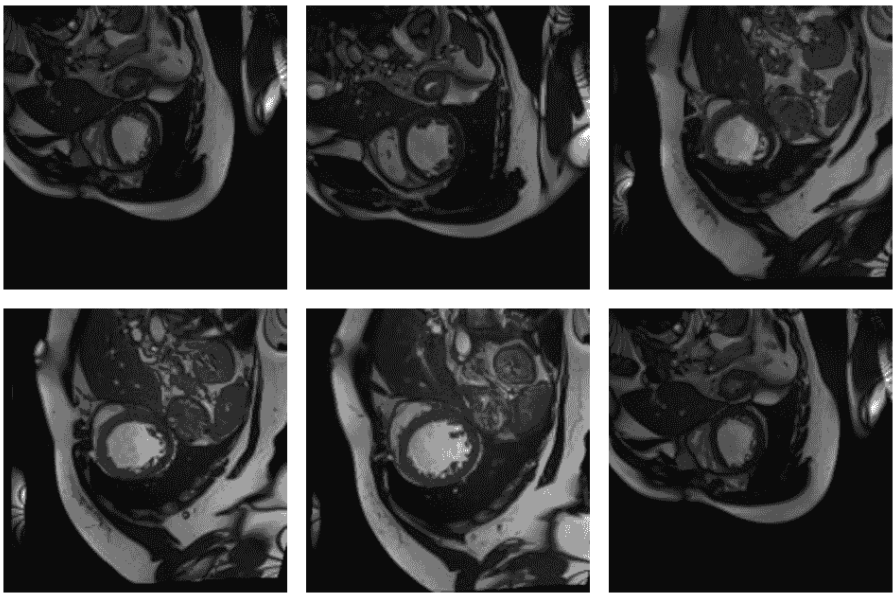

Figure 5.5: MRI Frames of ACDC Dataset

5.6.2 Preprocessing Pipeline

The preprocessing pipeline prepares the ACDC cine MRI data for training and evaluation. As shown in the Figure 5.6, it starts by loading the raw dataset and grouping patients according to their diagnosis (NOR, DCM, HCM, MINF, RV). Patient-level train, validation, and test splits are then created to ensure balanced and consistent evaluation.

A region of interest (ROI) is estimated for each patient by analysing the full 4D cine sequence using statistical measures and Hough circle detection. The end-diastole (ED)

and end-systole (ES) frames, along with their ground truth labels, are loaded, and the ROI centre and radius are added to the patient data.

Once processed, the information is saved as split-wise pickled files. The final step generates per-slice HDF5 files containing the image slice, corresponding label, ROI details, and pixel spacing. These structured HDF5 datasets serve as the input for training the LV and myocardium segmentation models.

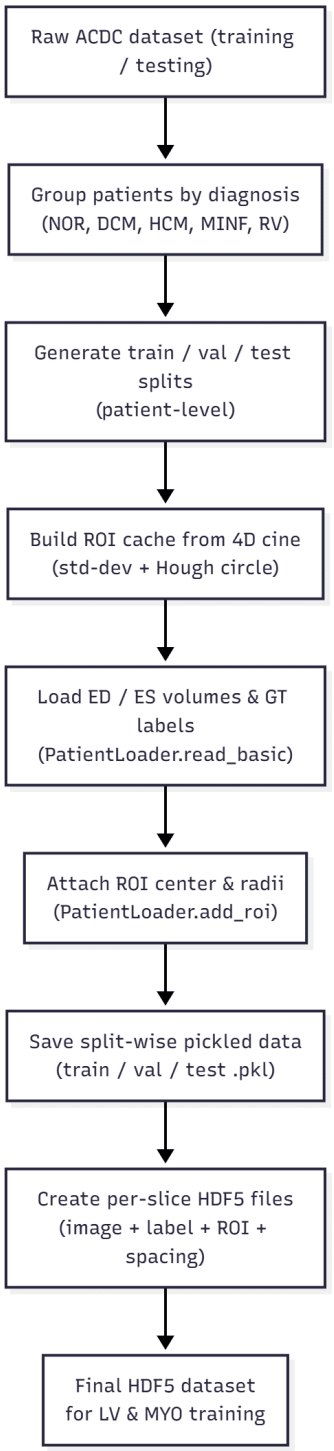

Figure 5.6: Preprocessing Pipeline

5.6.3 LV Model and Functional Indices

The LV network is an FCDenseNet[11] trained with a combined binary cross entropy and soft Dice objective to handle class imbalance.

Figure 5.7 illustrates the FCDenseNet architecture used for LV segmentation, comprising successive dense encoder blocks, a bottleneck, and symmetric decoder blocks with skip connections that preserve spatial detail and enable accurate frame wise LV mask prediction.

At inference, each frame is segmented independently, summing the voxels and multiplying by voxel volume gives the volume time curve. The maxima and minima define ED and ES, from which EDV, ESV, and EF follow.

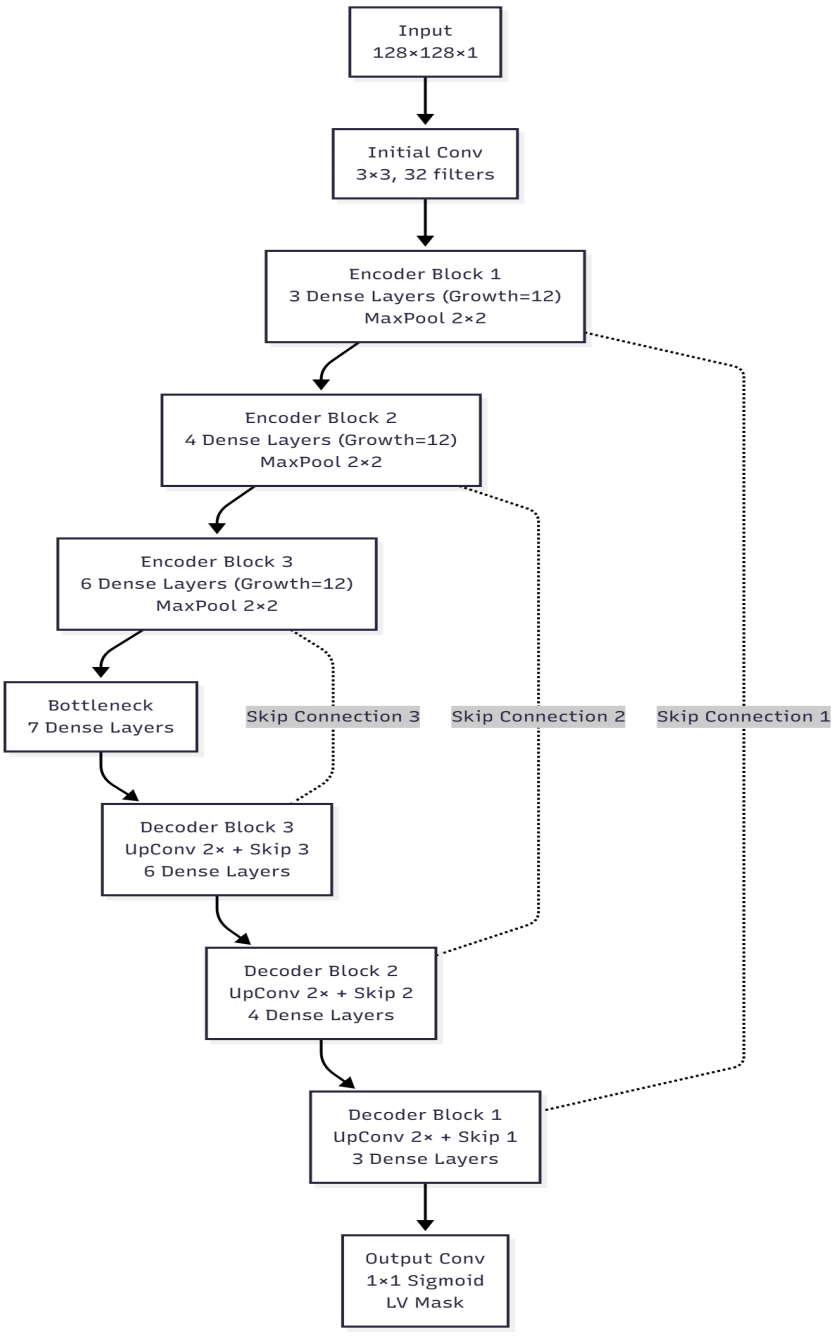

Figure 5.7: FCDenseNet LV Segmentation Pipeline

As shown in the figure 5.8 below, The perframe LV volume curve drops to a clear minimum at end-systole (frames 10–12) and then rises toward end-diastole, enabling automatic ED/ES detection and EF computation.

On our held out test split the LV segmenter (FCDenseNet) reached a Dice of 0.914 (91.4%), Soft-IoU 0.850, accuracy 0.9934, and a combined BCE+Dice loss of -0.4711. These results used the same 128×128 crop pad preprocessing as training.

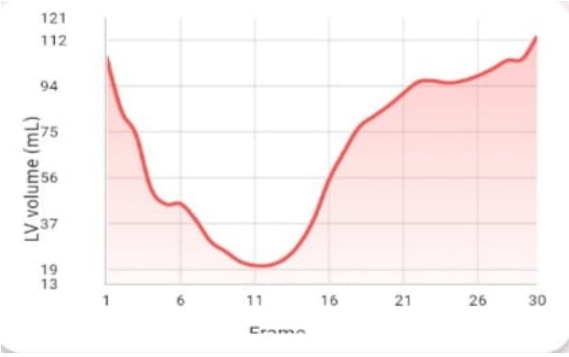

Figure 5.8: LV Volumes per Frame

5.6.4 Myocardium Model and Morphometrics

The myocardium uses the same family of networks with a light edge aware loss to sharpen the thin wall. The myocardium segmentation diagram as shown below in figure 5.9 shows a symmetric FCDenseNet encoder decoder, three dense encoder blocks downsample the ED frame, a bottleneck refines deep features, and three decoder blocks with upsampling and skip connections reconstruct a thin myocardial ring mask, which is then used to compute myocardium volume, mass, and wall thickness statistics.

We keep the main component, fill small gaps, and apply minimal smoothing for a stable ring. Volume is then computed from spacing, mass assumes 1.05 g/mL density. Thickness comes from the ring's centerline in physical units and is summarized as an area weighted mean plus the 5th and 95th percentiles and their range.

On the same test split the myocardium model (FCDenseNet)[11] achieved a Dice of 0.840 (84.0%), Soft-IoU 0.7723, accuracy 0.9893, and loss -0.3648. Thickness statistics are derived from our own masks (area-weighted mean with p05/p95 and range).

In early iterations the myocardium model struggled to track the thin wall across slices, which made thickness and mass unstable. We temporarily narrowed the diagnosis to LV function driven conditions while improving the MYO stage. After multiple training and inference passes, adding an edge emphasized loss, stronger connected component filtering and hole filling, and a refined resize/crop, we restored the full four class classifier. The final setup favors robustness over peak scores but runs end-to-end automatically.

We also chose to compute thickness from our own computed segmented masks rather than any provided contours. That costs some headline accuracy but keeps all structural indices (area-weighted mean, p05, p95, range) consistent with the actual predictions and measured in true physical units.

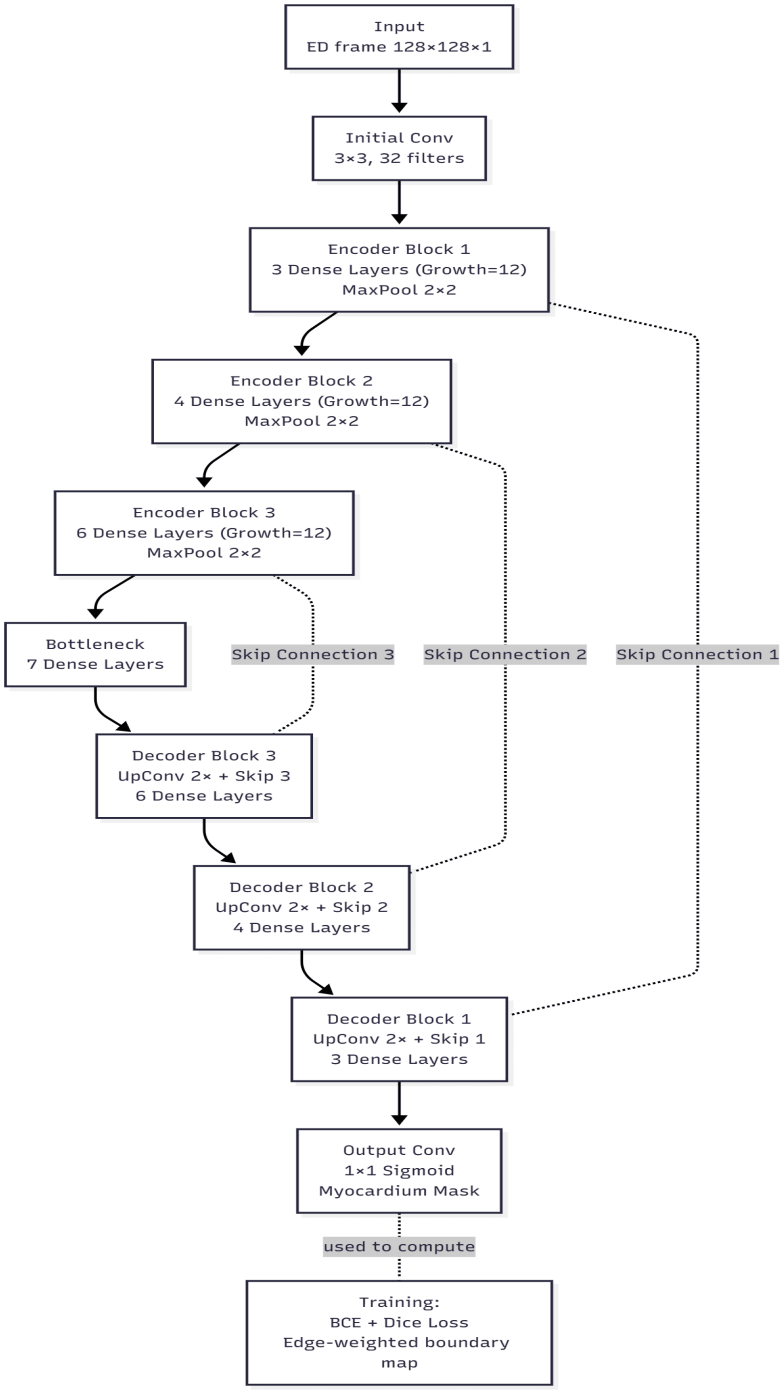

Figure 5.9: FCDenseNet Myocardium Segmentation Pipeline

5.6.5 Diagnostic Classifier

The diagnostic stage integrates function and structure using a stacked ensemble implemented in scikit-learn. As shown in the below figure 5.10, The classifier combines LV and myocardium features with base and expert model probabilities in a stacked ensemble to produce the final heart disease prediction with confidence. The base learner is a HistGradientBoostingClassifier that outputs four class probabilities (DCM, HCM, MINF, NOR). Two LogisticRegression models act as branch experts, one for low EF studies (DCM vs MINF) and one for high EF studies (HCM vs NOR). A multinomial LogisticRegression serves as the meta model and combines the base and branch probabilities with a compact subset of raw features to produce the final label and confidence.

On a held out test split, the stacked classifier achieved 85% accuracy. No test samples were used during training or hyper parameter selection.

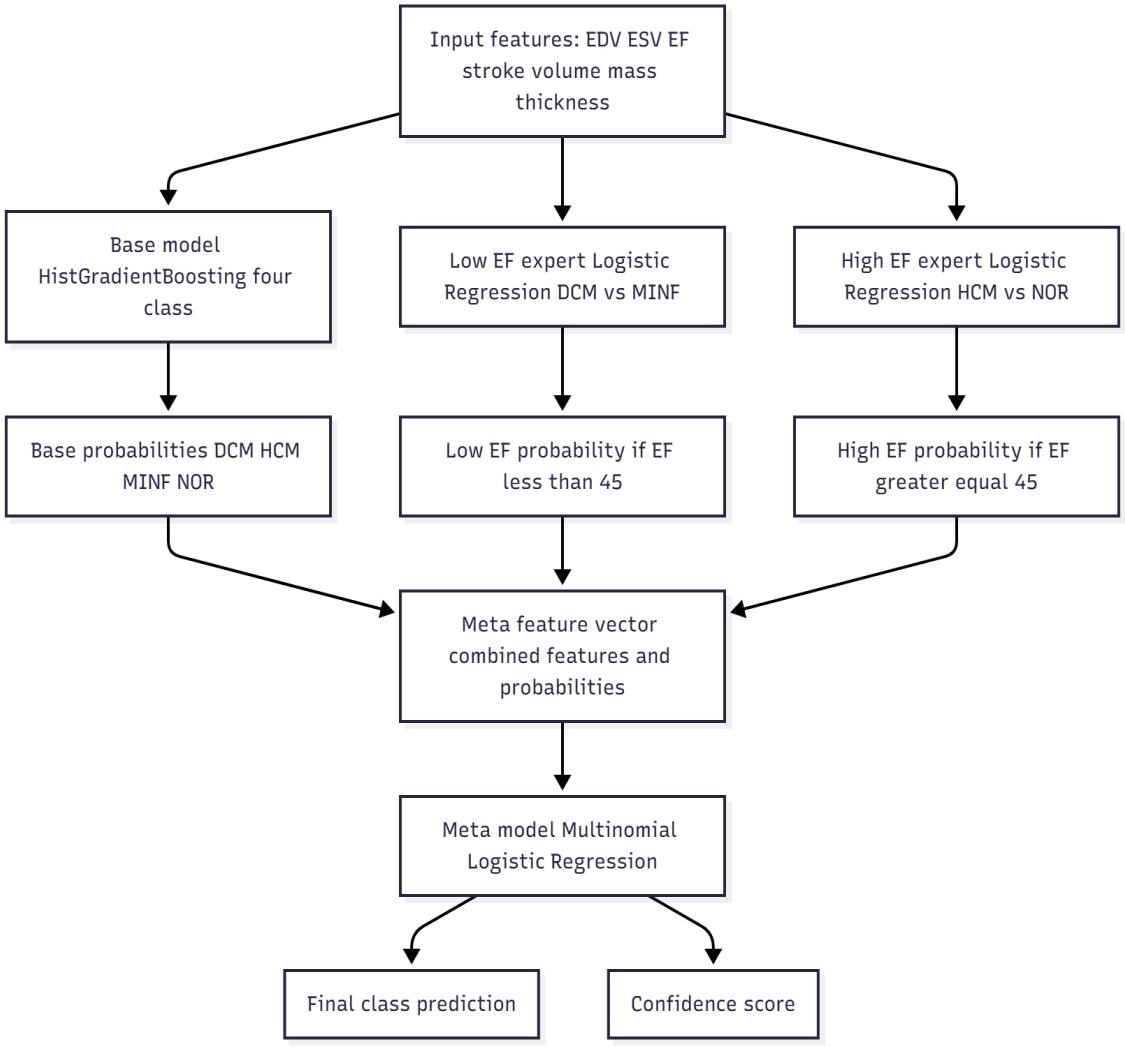

Figure 5.10: Classifier Pipeline

5.7 Client Application

The app keeps the flow simple. The client application is built in Flutter [15] to offer a simple and clear user experience, keeping the workflow easy to follow with only a few essential interactions. Prior to using any of the analysis utilities, the user is presented with the authentication page, where new users may sign up and existing users may log in securely. That step is to make sure that only authorized healthcare providers have the ability to upload studies and access the diagnostic information. This ensures only legitimate health care providers can upload cardiac MRI studies and they are the only ones who can access the diagnostic information.

When the login is successful, the user is lead to the dashboard page where navigation point for all cardiac MRI analysis is provided. The user picks an MRI study from the dashboard, and goes to the upload page. The upload panel shows the filename, size, and timestamp of the selected file, so you can make sure you are sending the correct scan. When the file gets uploaded, a brief progress indicator shows as the FastAPI backend is handling the MRI.

While processing the request, the application is interacting with the backend and polling for completion of segmentation and classification. When the results are ready, the app immediately changes to results mode and displays all computed cardiac measures in a straightforward and organized manner.

Important functional parameters, such as End-Diastolic Volume (EDV), End-Systolic Volume (ESV), and Ejection Fraction (EF) are visualized using circular gauge visual, so clinicians can easily interpret them at a quick glance. Also included is a frame by frame LV volume curve to demonstrate the full heart's behaviour. A diagnostic label that indicates the predicted disease class (NOR, DCM, HCM, or MINF) is also shown, which summarizes the decisions of the model. In general, the client-side application combines the authentication, file upload, server-side communication, and result visualization in a seamless user-experience. It enables users to go from MRI upload to a complete diagnostic overview in just a few steps, making the process both practical and efficient in clinical settings. The front-end and functionality of our app is given below.

5.7.1 Registration Screen

Users create an account with first/last name, email, and a password as shown in figure 5.11. The inline checks and the privacy link keep the details valid.

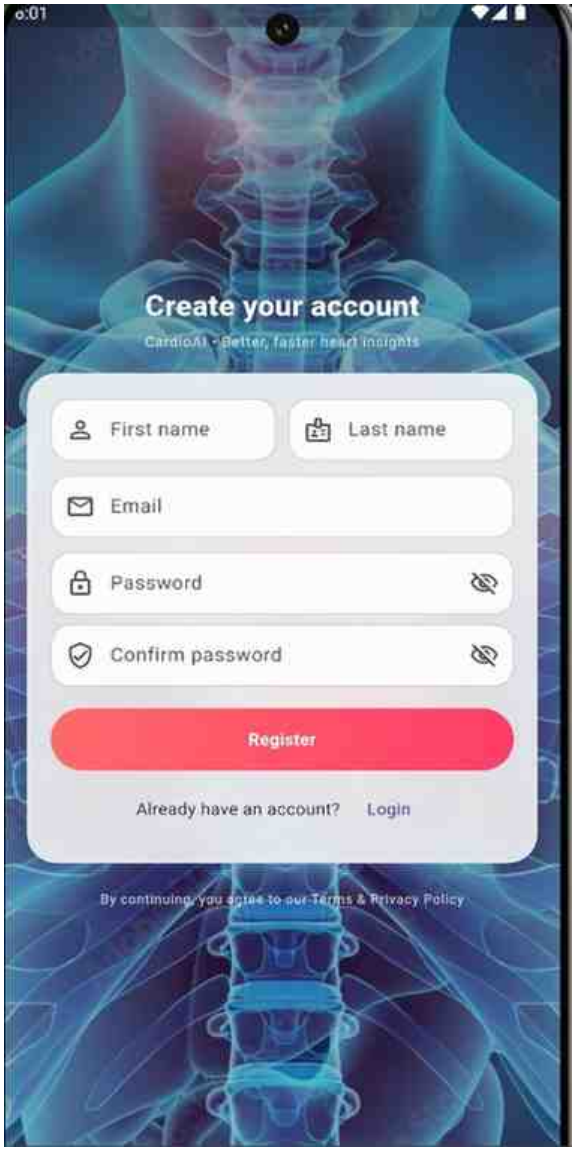

Figure 5.11: Register Account

5.7.2 Login Screen

Returning users sign in with email and password, with quick links for “Forgot password?” and “Create account” as shown in figure 5.12.

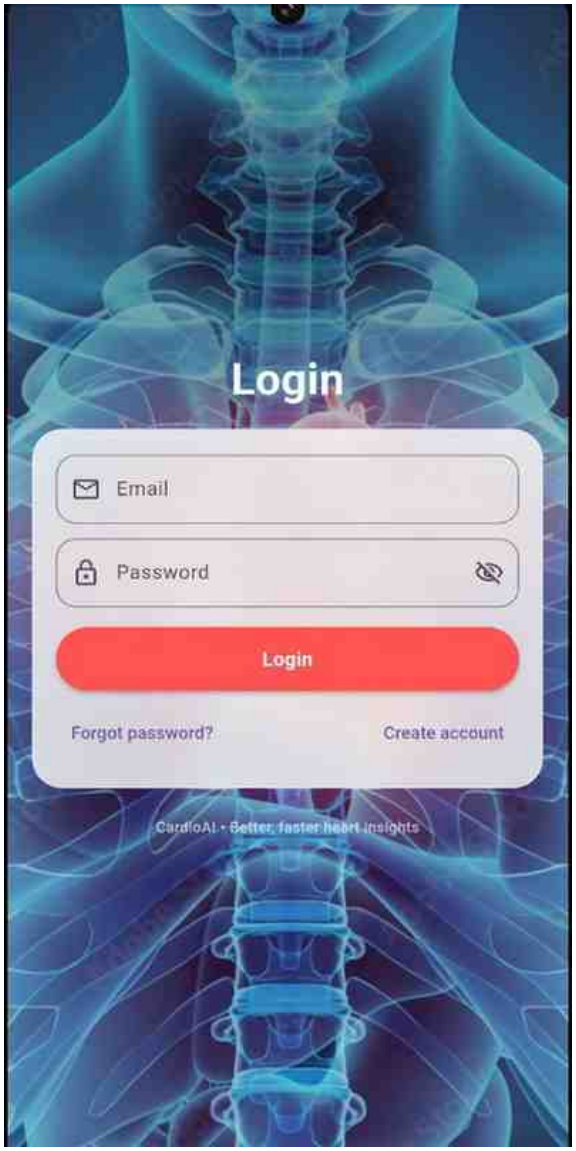

Figure 5.12: Login Account

5.7.3 Home / Dashboard Screen

The landing card as shown in figure 5.13 greets the user and provides the primary action like Upload Cardiac MRI with supported formats and quick tips.

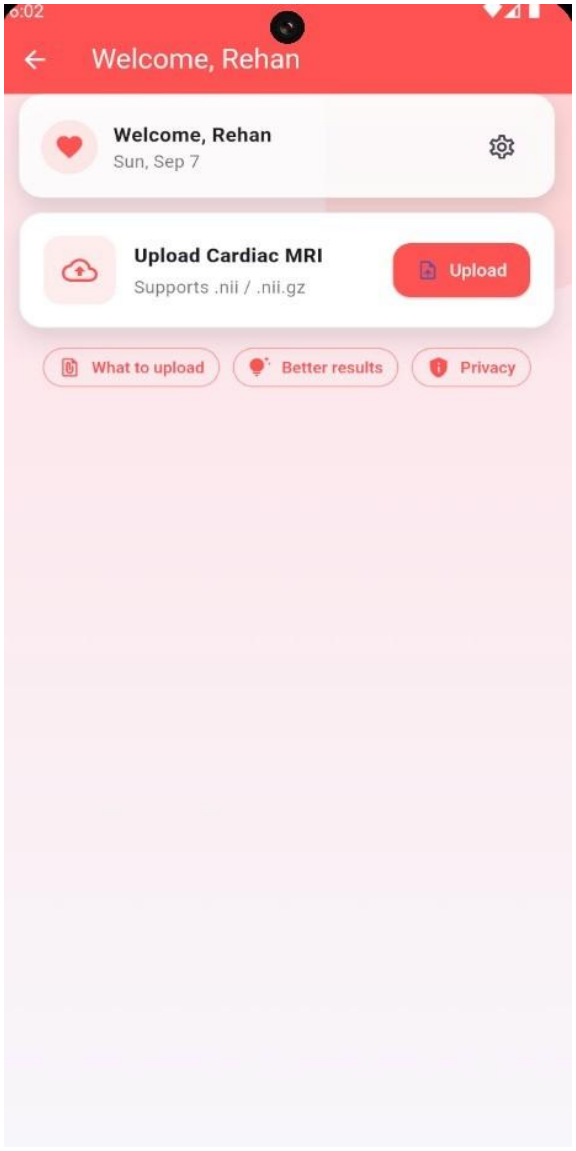

Figure 5.13: Home Screen with Upload Menu

5.7.4 Upload in Progress

After selecting a file as shown in figure 5.14, the app shows the filename and timestamp while the study is sent to the server for analysis.

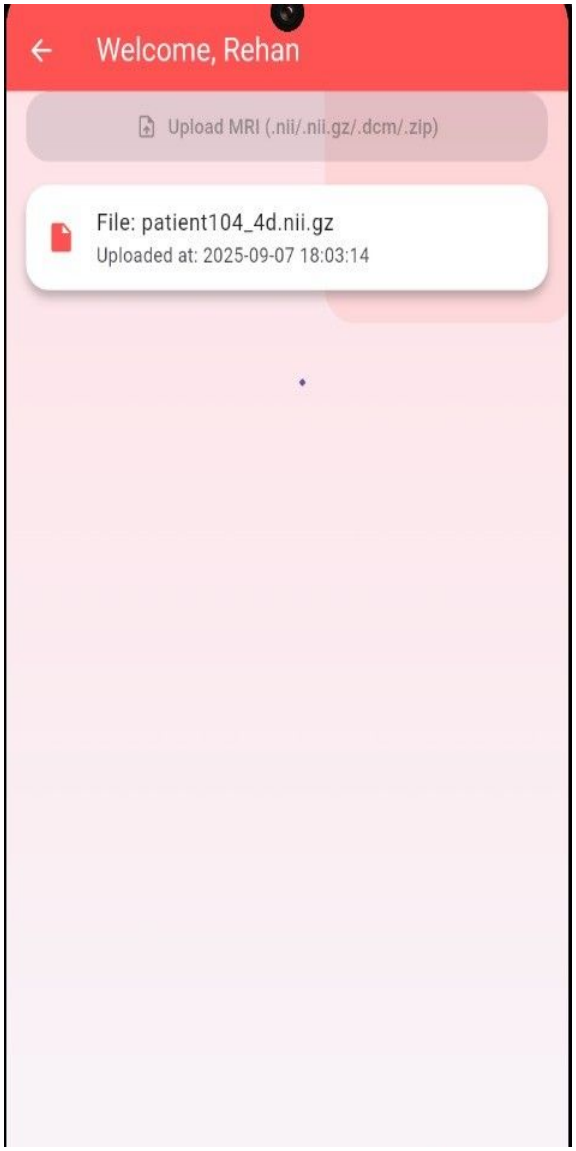

Figure 5.14: Uploaded Raw MRI

5.7.5 Diagnosis & Key Metrics

The predicted condition appears with a priority tag, alongside EDV, ESV, EF gauges and myocardium metrics (P95 wall thickness, mass) as shown in figure 5.15.

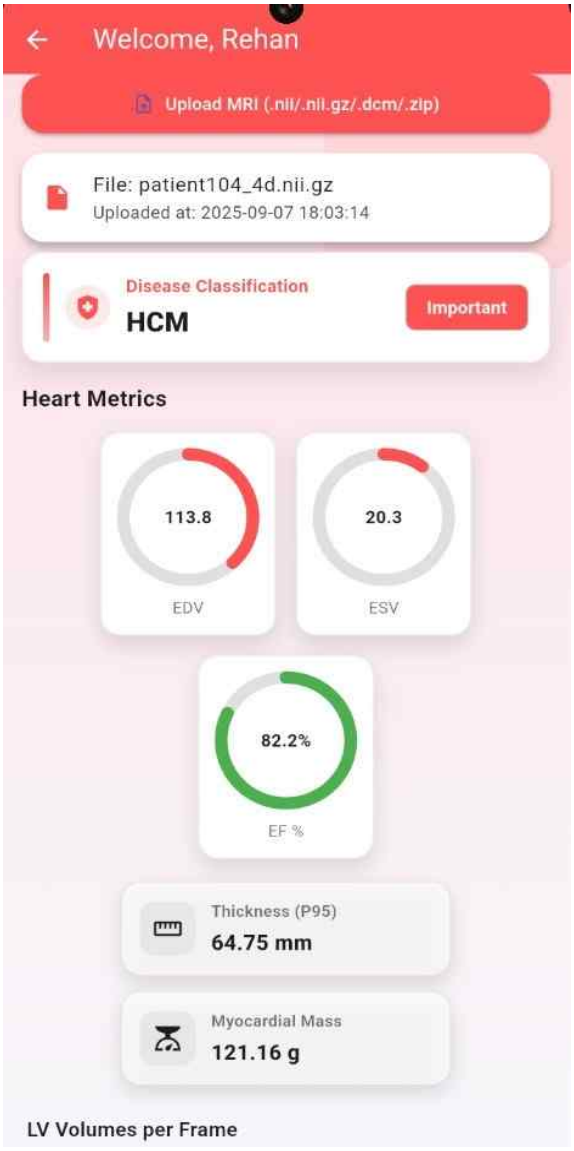

Figure 5.15: Generated Classified Condition

5.7.6 LV Volume Curve & History

A line chart as shown in figure 5.16 plots LV volume per frame (ED/ES visible at the extrema), followed by a "Recent Analyses" list with EF badges.

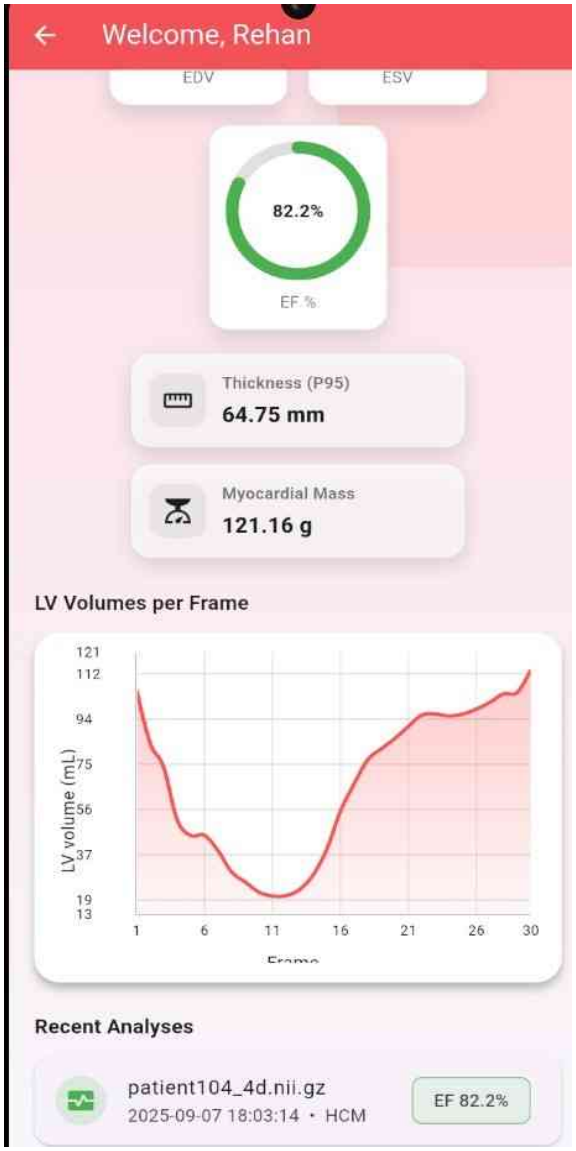

Figure 5.16: Generated Evaluated Results

5.8 Verification Activities

Segmentation quality was tracked with Dice and Soft-IoU on validation splits and by visually checking ED/ES overlays. The classification stack was evaluated with a held-out split of the meta model, reporting a confusion matrix and per class precision, recall, and F1. CSV reports and sanity checks (e.g., EF consistency with the LV curve; plausible thickness ranges) supported review.

We also enforced strict data hygiene. Training, validation, and testing were kept separate, no test samples were used for training or hyper parameter tuning, and we avoided cross split augmentation. Some published work mixes subsets or tunes on what later serves as “test”, which can inflate results.

5.9 Constraints and Prospects

Performance varies with acquisition protocols and scanners. And fine tuning on local data should improve generalization. Future work may add temporal or 3D models for smoother contours, calibrated uncertainty to guide use, and direct PACS/RIS integration (e.g., DICOM C-STORE) for automated ingestion and reporting.

Finally, our decision to avoid ground truth timing/mask shortcuts and to keep test data out of training sets a higher bar. The metrics may look less competitive than works that take those shortcuts, but the approach better reflects real world use and provides a trustworthy base for the next iteration.

Chapter 6

System Testing and Evaluation

This chapter documents how the mobile app and its backend were exercised, what was measured, and how we verified that the system behaves correctly from file pick to diagnosis. We combine model-level evaluation (segmentation/classification metrics) with functional, negative, and performance tests of the API and the Flutter client. In addition, we run end-to-end sanity checks e.g., ED/ES picked from the LV volume time curve, EF consistency and thickness/mass reported in correct physical units. Robustness is probed with corrupted or oversized files, missing DICOM metadata, and slow network conditions to confirm graceful error messages and timeouts. Finally, we enforce clean train, validation, test splits to rule out leakage and repeat critical experiments with fixed seeds to confirm reproducibility of the reported numbers.

Figure 6.1: The Value of Testing

6.1 Testing Strategy and Methodology

Our testing plan is designed to ensure that AICardia behaves like a practical clinical assistant not just a demo. We validate the full path. A user picks a file, uploads it, the server runs inference, and then the results appear on the device. The ACDC dataset is split into strict train,val,test sets (no leakage). Tests run across a small matrix of Android devices/OS versions and typical network conditions (good WiFi, slow 3G, offline). For automation, we use lightweight unit and API checks for end to end flows and we drive the app and verify the JSON from predict file and the onscreen metrics. A test passes when a valid study returns metrics and a single diagnosis within our latency budget and the UI remains responsive. It fails if inputs are not validated, errors are not explained, or resources are not cleaned up. Defects are logged with steps, device/OS, logs, and the offending file to guarantee reproducibility.

The following test types are covered in the next sections:

- **Functional Testing:** Does the pipeline do the right thing? (I/O, metrics, API, end to end flow.)
- **Non Functional Testing:** How well does it run? (Latency, reliability, compatibility.)
- **Usability Testing:** Can a clinician use it easily? (Clear flows, messages, readability, accessibility.)
- **Exception Handling:** Does it fail safely? (Bad files, network loss, missing models, timeouts.)

6.2 Functional Testing

This part includes a brief comparison with a similar 'Cardiac CINE MRI Segmentation' project in the table below 6.1 and 6.2 (to contextualize our results and choices) and then verifies that each building block of AICardia behaves as intended, first in isolation and then as a whole. We start with small, focused checks (unit tests), move to API/app wiring (integration), and finally exercise full user journeys (system). In parallel, we run black-box and white-box tests to confirm contracts and cover tricky branches.

Table 6.1: Methodology comparison: AICardia vs. Cardiac CINE MRI Segmentation

Aspect	**AICardia (ours)**	**Cardiac CINE MRI Segmentation**
Primary task	LV & myocardium segmentation + diagnosis UI (mobile app)	LV, RV, myocardium segmentation research demo
Input handling	Raw cine MRI (NIfTI/DICOM) full cycle, on device upload, server preprocess	Mostly ED/ES frames shown in figures, raw full cycle handling not demonstrated
Models	FCDenseNet segmentor; classifier head for DCM/HCM/MI/Normal	3D Autoencoder and 3D U-Net. no separate disease classifier app
Data practice	Segmentation trained/evaluated with held out split, no test leakage	Author notes ACDC usage, risk in literature of ED only curation or inadvertent train/test mixing, not an app
Deliverable	Android app (upload, progress, metrics, curves, history)	Research code + figure comparisons, no end-user app
Why results may differ	Harder setting: segmenting myocardium directly from raw cines (52 frames typical) and real uploads	Often easier ED/ES slices only, curated figures, may omit raw to mask pipeline complexity

Table 6.2: Results comparison (validation/test).

Component	**Metric**	**AICardia (ours)**	**Cardiac CINE MRI**
LV segmentation	Dice / Soft-IoU	**0.914** / **0.850**	0.855 / —
	Accuracy / Loss	0.9934 / −0.4711	— / —
MYO segmentation	Dice / Soft-IoU	**0.840** / **0.7723**	0.883 / —
	Accuracy / Loss	0.9893 / −0.3648	— / —
Classifier	Overall accuracy	**85%**	92%

Notes. Other project Dice (means): UNET LV = (0.84, 0.86, 0.87, 0.85) ⇒ 0.855; UNET MYO = (0.89, 0.90, 0.86, 0.88) ⇒ 0.883. AE numbers are lower (LV ≈ 0.783, MYO ≈ 0.788) and omitted here for brevity. They did not report IoU/Soft-IoU or pixel accuracy. Our pipeline evaluates full cine volumes from raw images; the other work showcases ED/ES slices, which is typically easier and can inflate Dice. Also, some reports in the area accidentally mix test into train we did not; our splits are clean. Finally, AICardia ships an end-to-end mobile app, while the other is a research codebase + figures.

6.2.1 Unit Testing

Backend (Python/FastAPI). We checked image I/O (NIfTI/DICOM and header spacing), LV and MYO metric calculators, classifier utilities (feature engineering and pipeline preprocessing), JSON response schema, and safe file handling for temporary paths.

Visual sanity check for segmentation To verify we have include a visual sanity check of the LV and myocardium segmentation at ED as shown in figures 6.2 and 6.3. The panels show the input slice, its ground truth mask, and AICardia's prediction. Agreement along the endocardial/epicardial borders and a clean, closed ring indicate a correct outcome.

LV segmentation visual sanity checks

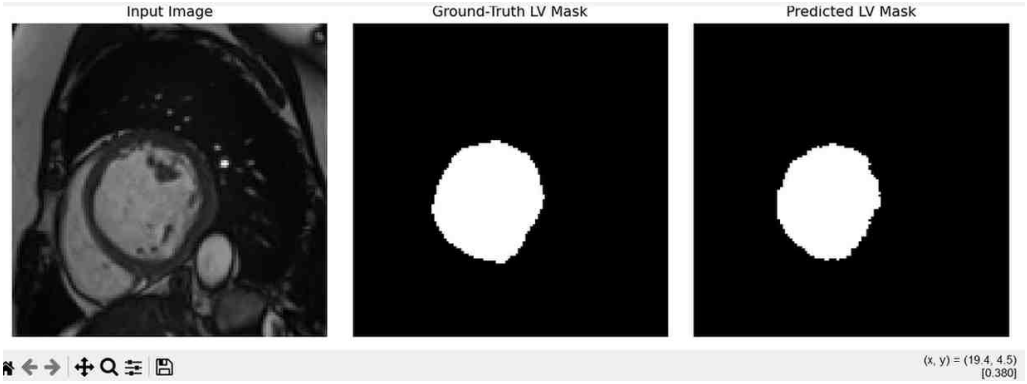

Figure 6.2: (a) LV segmentation visual sanity checks.

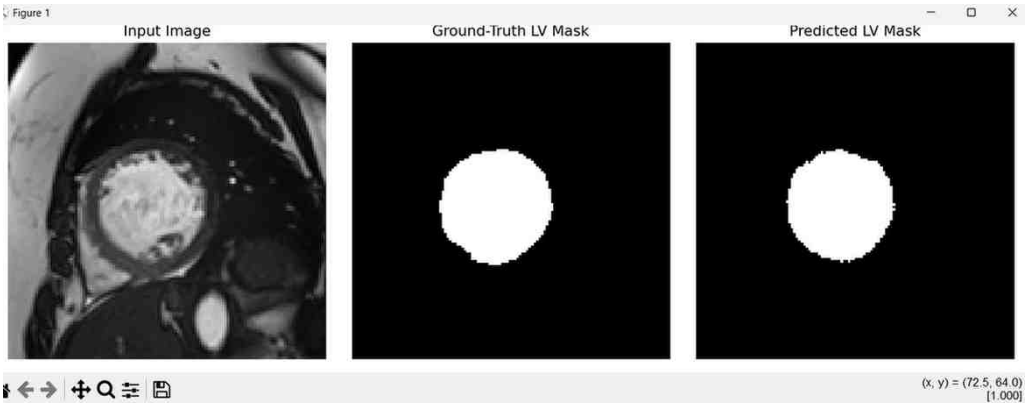

Figure 6.2: (b) LV segmentation visual sanity checks.

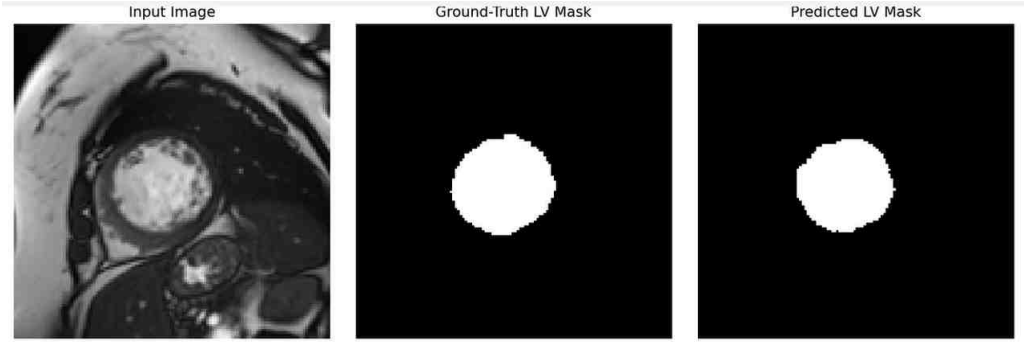

Figure 6.2: (c) LV segmentation visual sanity checks.

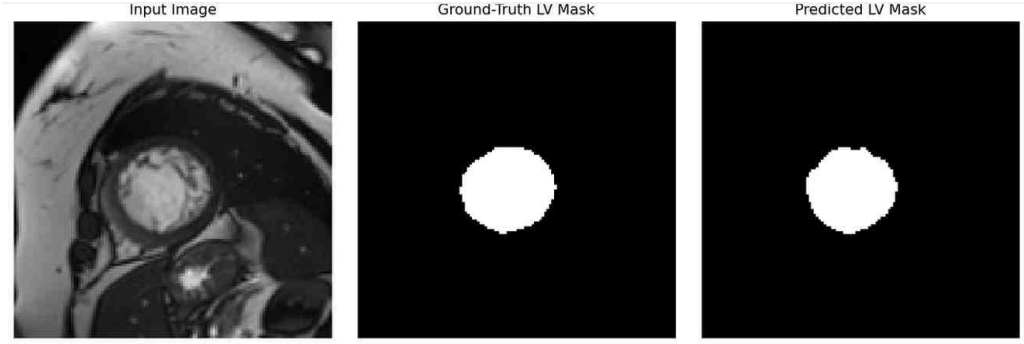

Figure 6.2: (d) LV segmentation visual sanity checks.

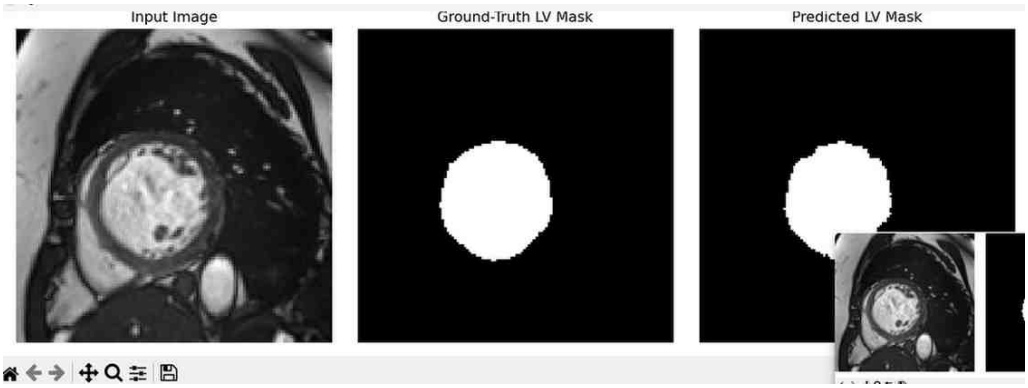

Figure 6.2: (e) LV segmentation visual sanity checks.

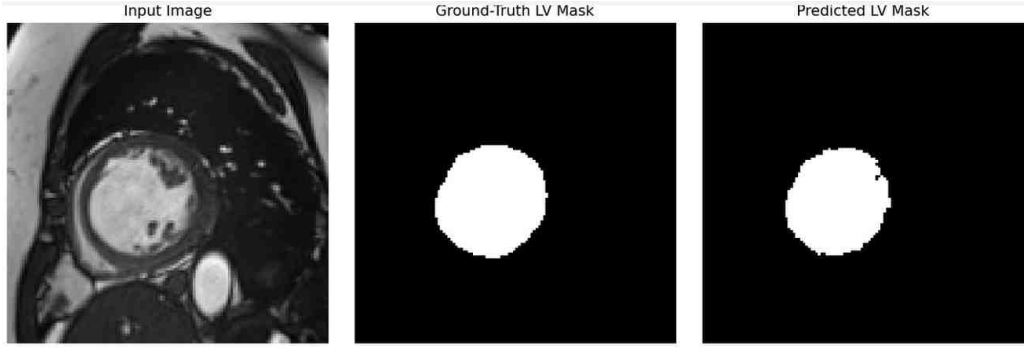

Figure 6.2: (f) LV segmentation visual sanity checks.

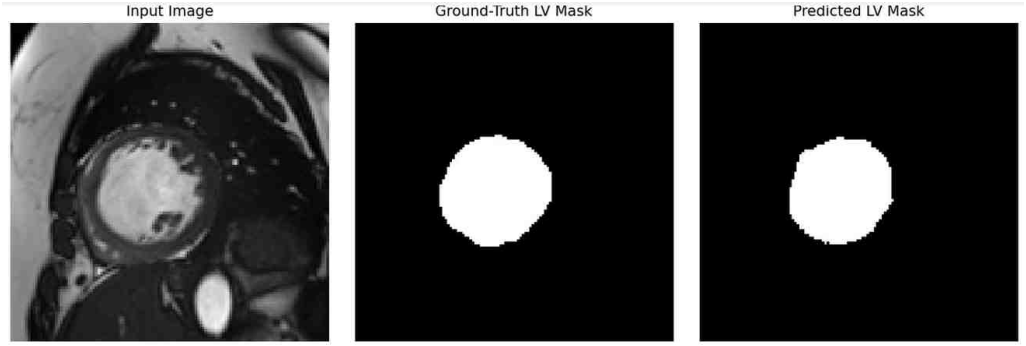

Figure 6.2: (g) LV segmentation visual sanity checks.

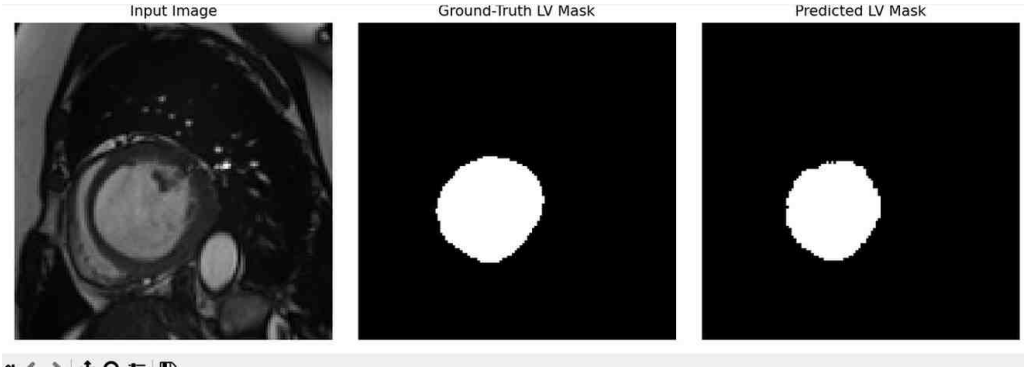

Figure 6.2: (h) LV segmentation visual sanity checks.

Myocardium segmentation visual sanity checks

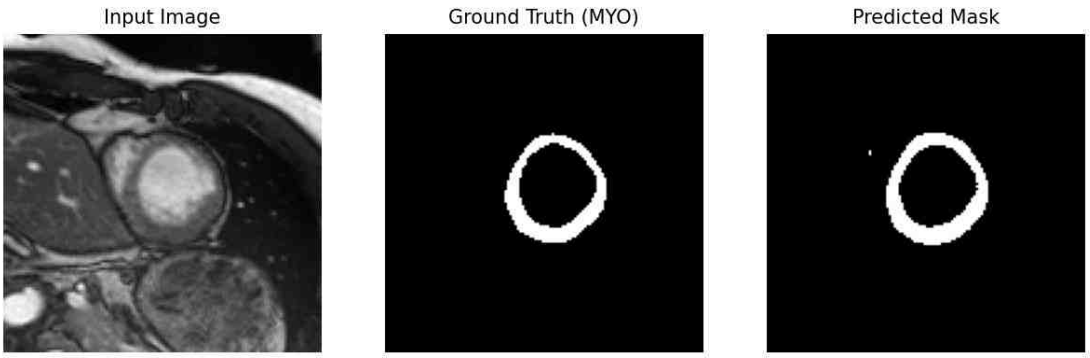

Figure 6.3: (a) Myocardium segmentation visual sanity checks.

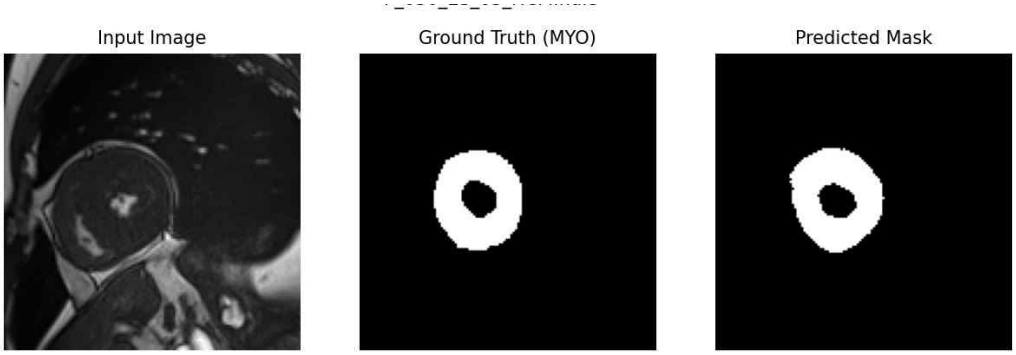

Figure 6.3: (b) Myocardium segmentation visual sanity checks.

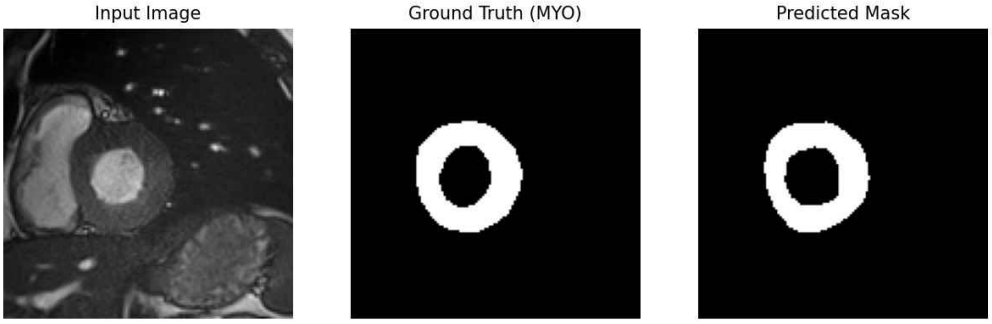

Figure 6.3: (c) Myocardium segmentation visual sanity checks.

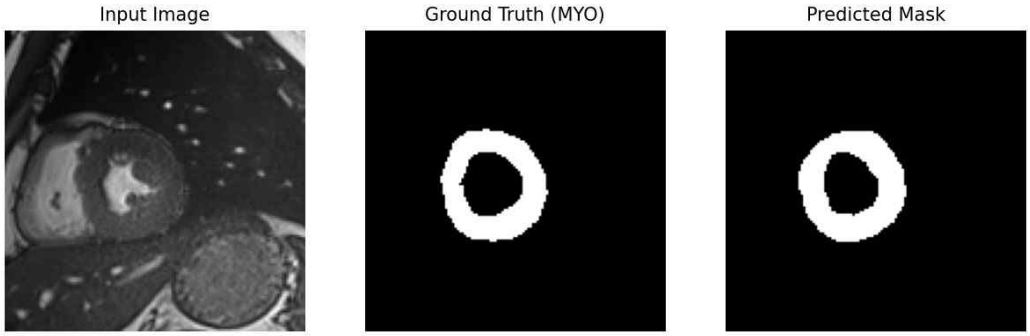

Figure 6.3: (d) Myocardium segmentation visual sanity checks.

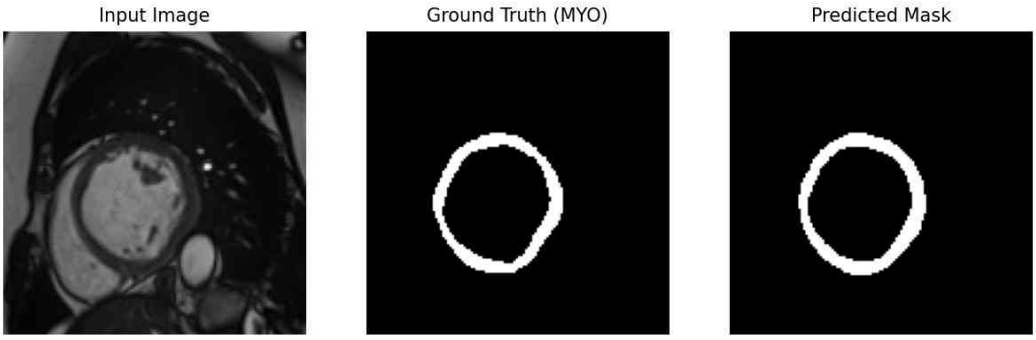

Figure 6.3: (e) Myocardium segmentation visual sanity checks.

Test cases (Model backend)

Table 6.3: NIfTI/DICOM parsing and spacing carry through

Test Case ID	U01
Testable Functionality	Image I/O: read .nii/.nii.gz and .dcm; preserve PixelSpacing/SliceThickness
Initial State	API running locally; sample ACDC files available
Input	One 4D NIfTI cine and one DICOM series of the same study
Expected Result	Loaded array shapes match headers; spacing in mm matches header values; no exceptions
Result	Loaded array shapes match headers; spacing in mm matches header values; no exceptions
Status	Pass

Table 6.4: LV metrics: EDV/ESV/EF and volume time curve

Test Case ID	U02
Testable Functionality	Compute LV volume per frame, detect ED (max) and ES (min), report EDV/ESV/EF
Initial State	LV model loaded in memory
Input	Cine with a known ED/ES index
Expected Result	ED/ES within ± 1 frame of reference; EF $= (EDV - ESV)/EDV \times 100$; curve is non-negative and smooth
Result	ED/ES within ± 1 frame of reference; EF $= (EDV - ESV)/EDV \times 100$; curve is non-negative and smooth
Status	Pass

Table 6.5: MYO metrics: thickness and mass

Test Case ID	U03
Testable Functionality	Thickness via skeletonization and $2 \times EDT$; mass with density $1.05\,\mathrm{g/mL}$
Initial State	MYO mask available for ED frame
Input	Synthetic ring phantoms with known wall thickness and area
Expected Result	Mean thickness within ± 0.5 mm of ground truth; mass proportional to voxel count and spacing
Result	Mean thickness within ± 0.5 mm of ground truth; mass proportional to voxel count and spacing
Status	Pass

Table 6.6: Classifier utilities and preprocessing

Test Case ID	U04
Testable Functionality	Feature engineering (ratios/ranges), class order mapping, pipeline imputation/scaling
Initial State	Scikit-learn pipelines serialized with joblib
Input	Rows with missing values and shuffled class order
Expected Result	No crash; imputation applied; class probabilities in fixed order [DCM, MINF, HCM, NOR]
Result	No crash; imputation applied; class probabilities in fixed order [DCM, MINF, HCM, NOR]
Status	Pass

Mobile (Flutter). We validated the file picker, JSON decoding and null guards, and UI widgets (charts vs. data length and progress states).

Test cases (App)

Table 6.7: JSON schema and error model

Test Case ID	U05
Testable Functionality	Response keys/types; error messages and HTTP codes
Initial State	API live; JSON Schema defined
Input	Valid cine; then invalid payloads (wrong MIME, empty file)
Expected Result	Valid: schema passes; Invalid: HTTP 400/415 with detail explaining the fault
Result	Valid: schema passes; Invalid: HTTP 400/415 with detail explaining the fault
Status	Pass

Table 6.8: File picker validation (Flutter)

Test Case ID	U06
Testable Functionality	Accept only .nii/.nii.gz/.dcm; size guard; show friendly errors
Initial State	App home screen
Input	Pick valid cine; then pick a .jpg and a $>$ 1 GB file
Expected Result	Valid proceeds; invalid shows toast/dialog and blocks submission
Result	Valid proceeds; invalid shows toast/dialog and blocks submission
Status	Pass

Table 6.9: Charts and progress state machine

Test Case ID	U07
Testable Functionality	JSON decode with NaN guards; LV curve length equals frame count; progress shows correct states
Initial State	App connected to API stub
Input	Inject JSON with 25 frames, known EF; inject NaN in thickness to test guards
Expected Result	Chart draws 25 points; EF matches; NaNs render as "–" without crash; progress transitions idle→uploading→processing→done
Result	Chart draws 25 points; EF matches; NaNs render as "–" without crash; progress transitions idle→uploading→processing→done
Status	Pass

6.2.2 Integration Testing

We verified the app can talk to predict, that CORS/auth and retry logic behave, and that the JSON response renders correctly on device.

Table 6.10: End-to-end upload and JSON verification

Test Case ID	I01
Testable Functionality	Multipart upload; metrics + diagnosis returned; device UI updates
Initial State	API reachable on Wi-Fi; models loaded
Input	file=@patient002_4d.nii.gz
Expected Result	HTTP 200; JSON contains EDV/ESV/EF/mass/thickness/label/confidence; app shows gauges, curve, chip
Result	HTTP 200; JSON contains EDV/ESV/EF/mass/thickness/label/confidence; app shows gauges, curve, chip
Status	Pass

Table 6.11: CORS/auth and retry/backoff

Test Case ID	I02
Testable Functionality	Preflight and token (if enabled); retry on 502/timeout
Initial State	App configured with API origin; flaky network emulator
Input	Trigger one transient failure then a success
Expected Result	First call fails with 5xx; app retries after backoff and succeeds; no duplicate uploads
Result	First call fails with 5xx; app retries after backoff and succeeds; no duplicate uploads
Status	Pass

6.2.3 System Testing

We exercised realistic user journeys and failure modes.

Table 6.12: Happy path: select → upload → results

Test Case ID	S01
Testable Functionality	End-user flow and responsiveness
Initial State	Fresh app session
Input	Valid cine; default settings
Expected Result	No UI freeze; results within target latency; numbers match API payload
Result	No UI freeze; results within target latency; numbers match API payload
Status	Pass

Table 6.13: Robustness to corrupt/unsupported files

Test Case ID	S02
Testable Functionality	Server side validation and user feedback
Initial State	API live
Input	Truncated .nii.gz; 3D-only volume; wrong Content Type
Expected Result	HTTP 400/415 with clear detail; app shows friendly message and stays usable
Result	HTTP 400/415 with clear detail; app shows friendly message and stays usable
Status	Pass

Table 6.14: Failure handling: network/timeout/disk/model-missing

Test Case ID	S03
Testable Functionality	Graceful degradation and recovery
Initial State	Faults injected (drop Wi-Fi; slow API; remove one model file)
Input	Standard upload
Expected Result	App shows progress + retry or cancel; server emits precise errors; missing model reports “temporarily unavailable” without crash
Result	App shows progress + retry or cancel; server emits precise errors; missing model reports “temporarily unavailable” without crash
Status	Pass

6.2.4 Black-Box Testing

We treated the API as an opaque service and verified contracts and ranges.

Table 6.15: Contract/range checks on metrics

Test Case ID	B01
Testable Functionality	Output schema and value ranges
Initial State	JSON Schema compiled
Input	20 random cines across classes
Expected Result	EF in $[0, 100]$; volumes/mass ≥ 0; arrays sized correctly; schema valid
Result	EF in $[0, 100]$; volumes/mass ≥ 0; arrays sized correctly; schema valid
Status	Pass

Table 6.16: Negative cases return correct HTTP codes

Test Case ID	B02
Testable Functionality	Error codes/messages for bad input
Initial State	API live
Input	Empty body; unknown field names; bad MIME
Expected Result	400/415 with human-readable detail, no 500s
Result	400/415 with human-readable detail, no 500s
Status	Pass

6.2.5 White-Box Testing

We covered edge branches and serialization paths to catch subtle errors.

Table 6.17: ED/ES edge conditions and percentile stats

Test Case ID	W01
Testable Functionality	Plateaus at ED/ES; noisy frames; percentile (p05/p95) stability
Initial State	Synthetic curves and noisy cines
Input	Curves with flat maxima/minima; added jitter
Expected Result	Deterministic tie-breaking; p05/p95 monotone with added noise; no index out of range
Result	Deterministic tie-breaking; p05/p95 monotone with added noise; no index out of range
Status	Pass

Table 6.18: joblib pipelines (save/load round trip)

Test Case ID	W02
Testable Functionality	Serialization/deserialization of GBM and meta/logistic models
Initial State	Trained artifacts on disk
Input	Save and reload, then score the same rows
Expected Result	Bitwise equal predictions/probabilities; class order preserved; no version errors
Result	Bitwise equal predictions/probabilities; class order preserved; no version errors
Status	Pass

6.2.6 Patient level Testing Results

The table below 6.19 lists each test patient's true vs. predicted class, a 0/100% accuracy flag, key cardiac indices (EF, EDV, ESV), and the model's confidence to transparently

audit test outcomes.

Table 6.19: AICardia patient-level test results.

Patient	True	Predicted	Correct	Acc. (%)	EF (%)	EDV (mL)	ESV (mL)	Conf.
patient101	DCM	MINF	FALSE	0	33.86	143.33	94.80	0.870
patient106	DCM	DCM	TRUE	100	18.29	206.76	168.95	0.921
patient113	DCM	DCM	TRUE	100	17.79	260.78	214.40	0.986
patient117	DCM	DCM	TRUE	100	14.72	174.08	148.45	0.954
patient122	DCM	MINF	FALSE	0	37.39	205.13	128.44	0.671
patient131	DCM	DCM	TRUE	100	13.29	216.78	187.97	0.986
patient132	DCM	DCM	TRUE	100	14.80	217.93	185.67	0.977
patient133	DCM	DCM	TRUE	100	22.72	301.90	233.33	0.993
patient136	DCM	DCM	TRUE	100	30.95	250.03	172.64	0.862
patient149	DCM	DCM	TRUE	100	23.76	325.38	248.08	0.982
patient104	HCM	HCM	TRUE	100	64.53	126.15	44.75	0.854
patient105	HCM	HCM	TRUE	100	82.38	133.68	23.55	0.776
patient108	HCM	HCM	TRUE	100	76.67	89.04	20.78	0.558
patient111	HCM	HCM	TRUE	100	64.81	84.18	29.62	0.542
patient114	HCM	HCM	TRUE	100	62.50	190.32	71.37	0.893
patient116	HCM	HCM	TRUE	100	63.78	129.69	46.97	0.517
patient134	HCM	NOR	FALSE	0	60.27	104.01	41.32	0.632
patient138	HCM	HCM	TRUE	100	72.24	96.77	26.86	0.942
patient142	HCM	HCM	TRUE	100	67.09	111.33	36.64	0.551
patient146	HCM	HCM	TRUE	100	71.87	153.20	43.10	0.691
patient103	MINF	MINF	TRUE	100	33.58	152.08	101.00	0.714
patient112	MINF	MINF	TRUE	100	49.70	152.85	76.88	0.729
patient115	MINF	MINF	TRUE	100	43.71	185.50	104.41	0.927
patient118	MINF	MINF	TRUE	100	30.00	135.53	94.88	0.807
patient120	MINF	MINF	TRUE	100	34.58	109.30	71.51	0.836
patient135	MINF	MINF	TRUE	100	31.79	171.24	116.81	0.542
patient137	MINF	NOR	FALSE	0	73.62	84.82	22.37	0.565
patient143	MINF	MINF	TRUE	100	33.06	224.25	150.12	0.513
patient145	MINF	MINF	TRUE	100	33.23	167.94	112.13	0.675
patient148	MINF	MINF	TRUE	100	32.42	116.07	78.44	0.872
patient102	NOR	NOR	TRUE	100	71.96	84.03	23.56	0.582
patient107	NOR	NOR	TRUE	100	56.23	128.34	56.17	0.617
patient110	NOR	NOR	TRUE	100	45.04	57.13	31.40	0.717
patient123	NOR	HCM	FALSE	0	68.89	93.00	28.93	0.695
patient125	NOR	NOR	TRUE	100	51.35	102.50	49.86	0.697
patient128	NOR	NOR	TRUE	100	54.02	138.39	63.63	0.487

Continued on next page

Patient	True	Predicted	Correct	Acc. (%)	EF (%)	EDV (mL)	ESV (mL)	Conf.
patient130	NOR	HCM	FALSE	0	61.89	176.77	67.37	0.553
patient139	NOR	NOR	TRUE	100	49.56	54.90	27.69	0.799
patient144	NOR	NOR	TRUE	100	48.83	92.63	47.40	0.496
patient150	NOR	NOR	TRUE	100	67.27	98.28	32.17	0.523

6.3 Non Functional Testing

6.3.1 Performance

We timed the full tap result path. It is naturally slower because a NIfTI cine MRI must load 50–60 frames before inference begins, cold starts add a small extra delay. We also watched CPU/RAM on phone and server. A run passes when 95% of cases stay within our latency budget and no timeouts appear under parallel uploads.

6.3.2 Reliability

Submitting the same cine multiple times gives the same JSON (allowing tiny rounding). Brief network drops trigger retry with back off, and duplicate taps don't double process. We expect consistent results across reruns.

6.3.3 Security & Robustness

Only allow listed types .nii/.nii.gz/.dcm/ and size limited files are accepted. Malformed or headerless inputs fail safely with clear messages and temp files are sandboxed and cleaned. CORS is restricted to approved origins in production.

6.3.4 Compatibility

We exercised entry, mid range, and flagship Android devices across versions, on Wi-Fi and mobile data. With slow/variable networks emulated, the progress UI remained stable and timeouts behaved as expected.

6.4 Usability Testing

Usability checks ensure AICardia feels simple and clear the first time a clinician picks it up. We looked for obvious actions like where to tap, plain language, sensible defaults, readable numbers or units while an upload is in progress or when something goes wrong. The following tables summarize the key usability test cases.

Table 6.20: Signup

Test Case ID	U01
Testable Functionality	Registration form clarity and guidance
Initial State	App opened on *Registration* screen
Input	Enter first/last name, email, password; toggle password visibility; press *Register*
Expected Result	Required fields marked, invalid email caught inline, clear error text, "Login" link visible
Result	Required fields marked, invalid email caught inline, clear error text, "Login" link visible.
Status	Pass

Table 6.21: Login flow

Test Case ID	U02
Testable Functionality	Login with helpful errors and recovery
Initial State	*Login* screen visible
Input	Try wrong password, then correct password; use *Forgot password?* link
Expected Result	Clear message for wrong credentials; email preserved; reset flow reachable; correct credentials navigate to Dashboard
Result	Clear message for wrong credentials; email preserved; reset flow reachable; correct credentials navigate to Dashboard
Status	Pass

Table 6.22: Upload MRI and guidance

Test Case ID	U03
Testable Functionality	Discoverability of Upload Cardiac MRI and file guidance
Initial State	*Home / Dashboard* visible
Input	Tap Upload; attempt unsupported file; then select valid .nii/.nii.gz
Expected Result	Supported formats shown; unsupported types blocked with plain message; valid file accepted and progress starts
Result	Supported formats shown; unsupported types blocked with plain message; valid file accepted and progress starts
Status	Pass

Table 6.23: Progress and state continuity

Test Case ID	U04
Testable Functionality	Upload progress feedback and resilience
Initial State	File selected; upload in progress
Input	Switch apps / rotate device / navigate back then return
Expected Result	Progress indicator persists; duplicate upload disabled; state is restored without losing the task
Result	Progress indicator persists; duplicate upload disabled; state is restored without losing the task
Status	Pass

Table 6.24: Results readability

Test Case ID	U05
Testable Functionality	Comprehension of diagnosis and key metrics
Initial State	*Results* screen after analysis
Input	Review diagnosis chip, EDV/ESV/EF gauges, thickness (P95) and mass; inspect LV volume curve
Expected Result	Units shown (mL, g, mm, %); labels readable; curve axes/legend clear; large tap targets
Result	Units shown (mL, g, mm, %); labels readable; curve axes/legend clear; large tap targets
Status	Pass

Table 6.25: Helpful error messages

Test Case ID	U06
Testable Functionality	Plain language errors and recovery actions
Initial State	App ready for upload
Input	Submit corrupt/unsupported file; simulate network drop
Expected Result	Friendly message (what happened + what to do); *Try again* action; no crashes
Status	Pass

6.5 Exception Handling

Our Project, AICardia treats the failures as recoverable events. Invalid files, backend issues (e.g., missing models, low disk), or network drops are detected early and mapped to clear HTTP errors with human readable messages. Temporary files are cleaned, the app stays responsive with Retry and Cancel, and repeated submissions are idempotent (no duplicate jobs, safe CPU fallback if GPU is unavailable).

Table 6.26: Unsupported file type

Test Case ID	E01
Testable Functionality	Reject non-NIfTI/DICOM uploads
Initial State	API live; app on *Upload*
Input	Submit .jpg/.pdf/.txt file
Expected Result	HTTP 415/400 with clear detail (“unsupported type”); app shows friendly message + *Try again*; no crash
Result	HTTP 415/400 with clear detail (“unsupported type”); app shows friendly message + *Try again*; no crash
Status	Pass

Table 6.27: Corrupted or empty file

Test Case ID	E02
Testable Functionality	Safe handling of unreadable inputs
Initial State	API live
Input	Upload zero-byte or truncated .nii.gz
Expected Result	HTTP 400 with “file unreadable/corrupt”; temp files cleaned; app stays responsive with *Retry*
Result	HTTP 400 with “file unreadable/corrupt”; temp files cleaned; app stays responsive with *Retry*
Status	Pass

Table 6.28: Wrong dimensions (not 4D cine)

Test Case ID	E03
Testable Functionality	Validation of expected 4D shape
Initial State	API live
Input	Upload 3D volume or 5D array
Expected Result	HTTP 422/400 with “unsupported dimensionality”; no processing started; app shows guidance
Result	HTTP 422/400 with “unsupported dimensionality”; no processing started; app shows guidance
Status	Pass

Table 6.29: Missing spacing headers

Test Case ID	E04
Testable Functionality	Graceful failure when PixelSpacing/SliceThickness absent
Initial State	API live
Input	NIfTI/DICOM without spacing metadata
Expected Result	HTTP 422/400 with “missing spacing”; no crash; temp files removed
Result	HTTP 422/400 with “missing spacing”; no crash; temp files removed
Status	Pass

Table 6.30: Missing model artifacts

Test Case ID	E05
Testable Functionality	Server error when models not loaded
Initial State	API started without .keras/.joblib files
Input	Normal upload
Expected Result	HTTP 503/500 with "model unavailable"; request logged; app shows *Please try later*
Result	HTTP 503/500 with "model unavailable"; request logged; app shows *Please try later*
Status	Pass

Table 6.31: Processing timeout

Test Case ID	E06
Testable Functionality	Timeouts handled cleanly
Initial State	API live with short request timeout
Input	Upload very large cine; force long run
Expected Result	HTTP 504/408 with friendly text; no partial writes; app offers *Retry*
Result	HTTP 504/408 with friendly text; no partial writes; app offers *Retry*
Status	Pass

Table 6.32: Network drop during upload

Test Case ID	E07
Testable Functionality	Client retry/back-off and idempotency
Initial State	Upload in progress
Input	Kill Wi-Fi/cellular mid-transfer, then restore
Expected Result	Graceful failure toast; user can Retry; no duplicate jobs on resume
Result	Graceful failure toast; user can *Retry*; no duplicate jobs on resume
Status	Pass

6.6 System Evaluation

The AICardia system was evaluated as a complete pipeline to measure segmentation accuracy, cardiac metric computation, disease classification performance, and overall usability within the mobile application. All experiments were conducted on unseen patients from the ACDC dataset for an unbiased evaluation. As illustrated in Figure 6.4 below, the LV segmentation (FCDenseNet) is already very good, resulting in smooth and consistent cavity contours for basal, mid, and apical slices. On the test set, it obtained a

Dice score of 0.914, a Soft-IoU of 0.850, and an accuracy of 0.9934, which allows us to conclude that the cavity extraction is robust for subsequent volume analysis.

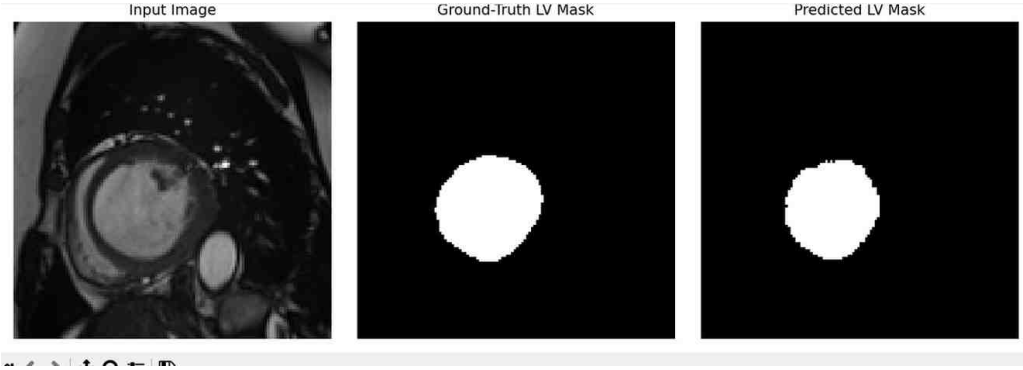

Figure 6.4: LV segmentation visual sanity checks.

In Figure 6.5 below, the myocardium segmentation model demonstrated a strong result, accurately capturing the thin ventricular wall with consistent ring-shaped boundaries that can be used for volume and thickness measurements. It obtained a Dice score of 0.840 and a Soft-IoU of 0.7723, which aligns with expected values for thin myocardial structures.

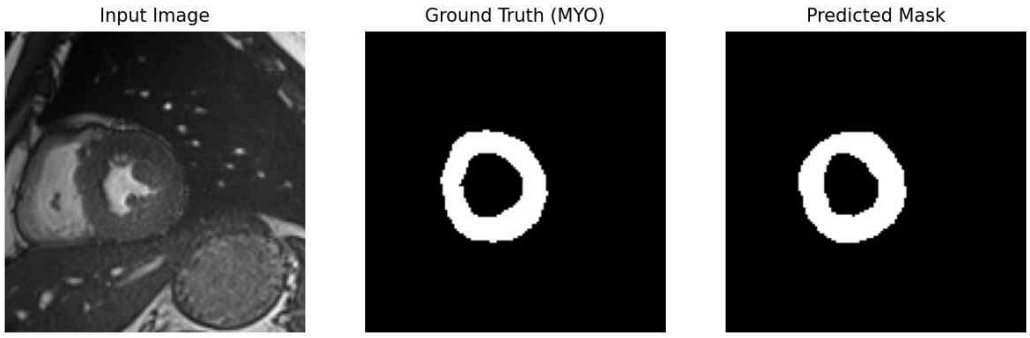

Figure 6.5: Myocardium segmentation visual sanity checks.

Using these segmentations, the system computed end-diastolic (ED) and end-systolic (ES) volumes and derived the ejection fraction (EF) from the LV volume–time curve. Across all test patients, ED/ES identification was stable, and the resulting EF values fell within clinically reasonable ranges.

For diagnostic evaluation, as shown in the Figure 6.6 below, the stacked classifier was tested across all four categories (DCM, MINF, HCM, NOR). It achieved an overall accuracy of 85%, with strong performance in distinguishing low-EF conditions (DCM vs. MINF) and good separation between structural categories (HCM vs. NOR). Confidence values ranged from 0.51 to 0.99, showing that the model reported uncertainty appropriately.

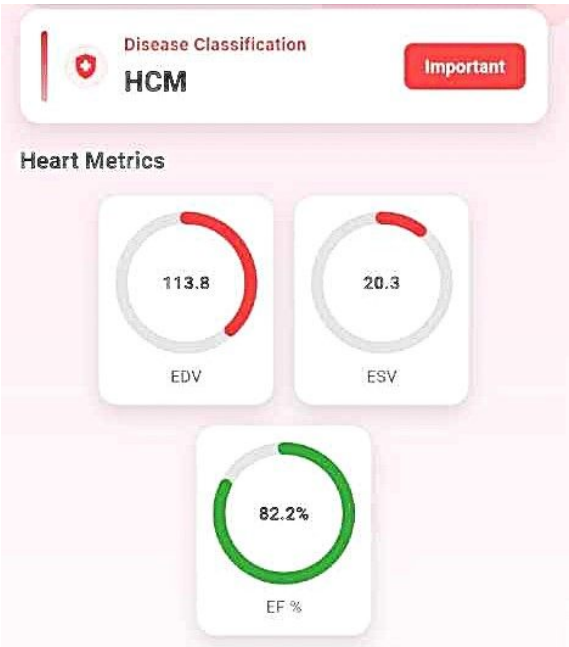

Figure 6.6: Classification Validity.

As shown in table 6.33, our numbers are therefore more conservative and more reproducible. To limit overfitting, the model choice was based on validation behaviour (early stopping and checkpoints), and the classifier used a held-out split in its own feature space.

Table 6.33: validation/test metrics

Component	**Metric**	**Result**
LV segmentation	Dice / Soft-IoU	0.914 / 0.850
	Accuracy / Loss	0.9934 / −0.4711
MYO segmentation	Dice / Soft-IoU	0.840 / 0.7723
	Accuracy / Loss	0.9893 / −0.3648
Classifier	Overall accuracy	85%

A patient level review was also completed, summarizing each subject's true diagnosis, predicted label, classification correctness, functional metrics (EDV, ESV, EF), and model confidence. Most incorrect predictions occurred in borderline structural cases where HCM and NOR display similar wall-thickness features, consistent with observations in existing studies.

The mobile application was evaluated in terms of usability and stability. Users are able to register, log in, upload MRI scans, and see results through a neat and simplified interface. EDV, ESV, and EF are presented using circular gauges, the LV volume curve is displayed clearly, and the predicted diagnosis is shown in an easily interpretable format.

Overall, the system demonstrated strong segmentation accuracy, dependable functional measurements, solid classification performance, and a user-friendly workflow. These

results indicate that AICardia functions effectively as an AI-assisted cardiac MRI analysis tool suitable for clinical decision support and research use.

Chapter 7

Conclusions

AICardia set out to turn a raw cine cardiac MRI into clear, clinician friendly answers on a phone. The system has that end to end access, the Flutter app uploads a study (NIfTI/DICOM), the FastAPI backend segments the left ventricle (LV) and myocardium (MYO), derives EDV/ESV/EF, wall thickness and mass, and a stacked classifier labels the case as DCM, HCM, MINF, and NOR. We kept the workflow honest, no test and train mixing and no reliance on ground truth ED/ES timings at inference.

7.1 What We Achieved

7.1.1 An end to end, working pipeline

A single, deterministic pipeline connects the mobile client, the inference service, and the trained models. From "file pick" to diagnosis, the app shows progress, returns metrics, and renders a volume time curve and a final label with confidence.

7.1.2 Methodological integrity

Unlike many implementations that rely on provided ED/ES frames or curated masks, AICardia computes ED/ES from the model's own per frame segmentations. We also enforced strict split hygiene as test data never informed training or tuning.

7.1.3 Model performance

With the ACDC dataset and our 128×128 preprocessing, the LV segmenter (FCDenseNet) achieved Dice 0.914, Soft-IoU 0.850, accuracy 0.9934, and combined BCE+Dice loss -0.4711

on the held-out test set. The MYO model reached Dice 0.840, Soft-IoU 0.7723, accuracy 0.9893, and loss -0.3648. The stacked classifier attained 85% test accuracy. These figures were obtained without using ground truth ED/ES at inference.

7.1.4 A simple mobile experience

Registration and login are straightforward. Uploading a cine triggers server processing and the results screen highlights EDV, ESV, EF and key MYO metrics, including a clear diagnosis chip. Along with the recent analyses are listed for quick review.

7.1.5 Resilience and safety

Bad inputs fail safely with clear messages, temporary files are cleaned, retries and idempotent resubmissions avoid duplicate work. CORS is restricted in production and artifacts load defensively at server start.

7.2 What Was Hard

At the start we planned to classify conditions using LV metrics alone. Early experiments showed that was not enough. As some conditions need structural cues from the myocardium. MYO segmentation proved harder than LV. The thin ring is easy to miss, and early versions were unstable across slices. After several rounds of training (edge-aware loss, stronger post-processing, better crop/resize) we restored the full four class pipeline with reliable thickness and mass estimates. We also accepted that loading 4D NIfTI with $\sim$50 frames adds latency; this shaped our performance targets and UI feedback.

7.3 Limitations

- **Generalization.** Performance can vary across scanners and protocols; local fine tuning is likely to help.
- **Latency.** End to end time is bounded by 4D file I/O and per frame inference. Very large studies load and process more slowly.
- **Scope.** PACS/RIS integration, DICOM C-STORE, and formal clinical validation are outside this version.
- **Regulatory.** AICardia is a research prototype, not a medical device. The results must be reviewed by clinicians.

7.4 Future Directions

- **Models.** Explore 2.5D/3D or temporal networks for smoother contours and add calibrated uncertainty and per class thresholds.
- **Data and learning.** Active learning from clinician edits and cross-site training with strict de-identification.
- **Vast Dataset.** Training on different types of datasets and formats of CINE MRI.

7.5 Lessons Learned

7.5.1 Technical

Working with medical images means respecting headers and spacing, also small mistakes there ripple into all metrics. Per-frame LV segmentation is a reliable anchor for ED/ES, and stacking simple classical models on well-chosen features can perform competitively. Packaging with FastAPI and joblib keeps deployment practical.

7.5.2 Process

Clear split hygiene, repeatable seeds, and contract tests (for JSON shape and value ranges) save time later. Honest reporting with no shortcut use of ground truth timings, and no test leakage builds trust even when numbers look modest.

7.6 Recommendations

For the next team, some recommendations are as follows:

- While selecting the dataset, choose the large one or multiple datasets.
- Ship with per class decision thresholds and a “low confidence” flag to guide review.
- Expand data diversity and document any site-specific preprocessing.
- Keep user guidance simple like what to upload, typical processing time, and how to read the charts.

7.7 Closing Note

AICardia now turns a cine MRI into interpretable LV function, myocardium structure, and a four class diagnosis on a phone. The journey from a LV only idea to a robust LV + MYO system taught us when extra modelling is essential and how to keep the pipeline dependable.

While there is more to build like speed, integrations, and clinical studies. We have a solid, transparent foundation that clinicians can try, question, and improve with us.

Appendix A

User Manual

Welcome to AICardia! This User Manual helps clinicians turn a cine cardiac MRI into clear heart metrics and a suggested diagnosis on mobile. This guide walks through account setup, uploading a study, and reading results.

A.1 System Requirements

- **Device:** Android phone (Android 9.0 or later).
- **Network:** Stable Wi-Fi or mobile data (uploads are larger for .nii/.nii.gz).
- **Files:** 4D cine MRI in .nii, .nii.gz, .dcm, or a zipped study.

A.2 Getting Started

A.2.1 Create an Account

Open the app and register with your name, email, and a password as shown in figure A.1. Inline checks keep entries valid and a privacy link is provided.

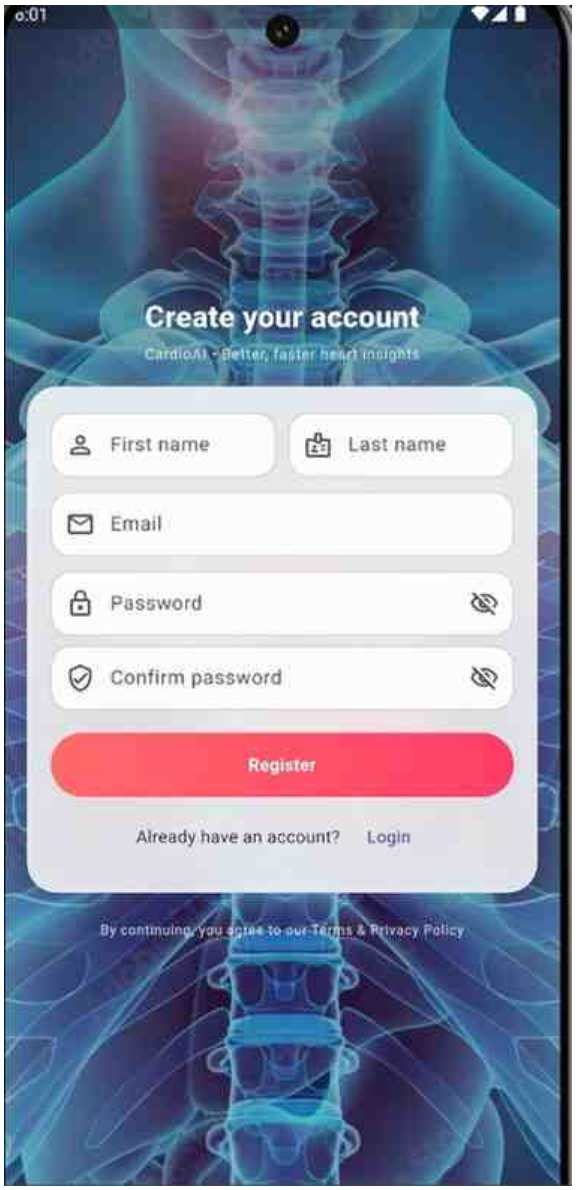

Figure A.1: Registration screen. "Create your AICardia account".

A.2.2 Log In

Return to the app and sign in using your email and password as shown in figure A.2. Quick links let you reset a password or create a new account.

Figure A.2: Login screen with secure access for returning users.

A.3 Using the App

A.3.1 Home / Dashboard

The home card greets you and shows the main action like "Upload Cardiac MRI" as shown in figure A.3. Supported formats are listed with quick tips.

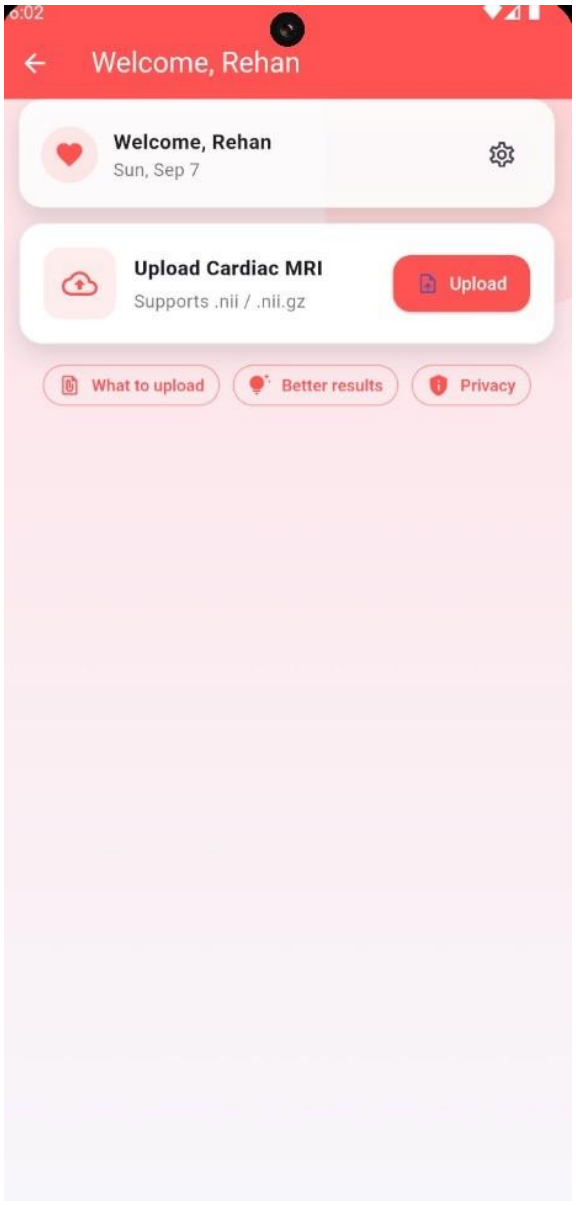

Figure A.3: Home/Dashboard, start an analysis from “Upload Cardiac MRI”.

A.3.2 Upload in Progress

After choosing a file, the app shows the filename and timestamp while the study uploads and the server starts analysis as shown in figure A.4.

Figure A.4: Upload status, selected file and submission time.

A.3.3 Diagnosis & Key Metrics

When processing completes, AICardia shows as shown in figure A.5 the predicted condition with a priority tag, plus EDV, ESV, EF, thickness (P95), and mass.

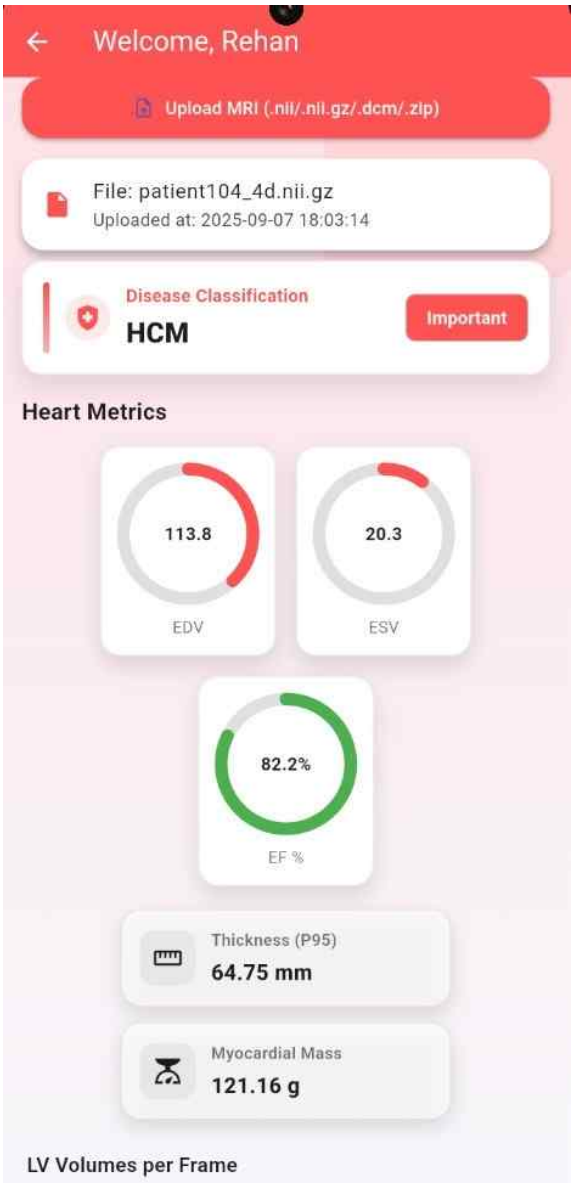

Figure A.5: Results, diagnosis chip and core heart metrics.

A.3.4 LV Volume Curve & History

A line chart plots as shown in figure A.6 shows LV volume per frame (ED/ES are visible at the extrema). Recent analyses are saved with an EF badge.

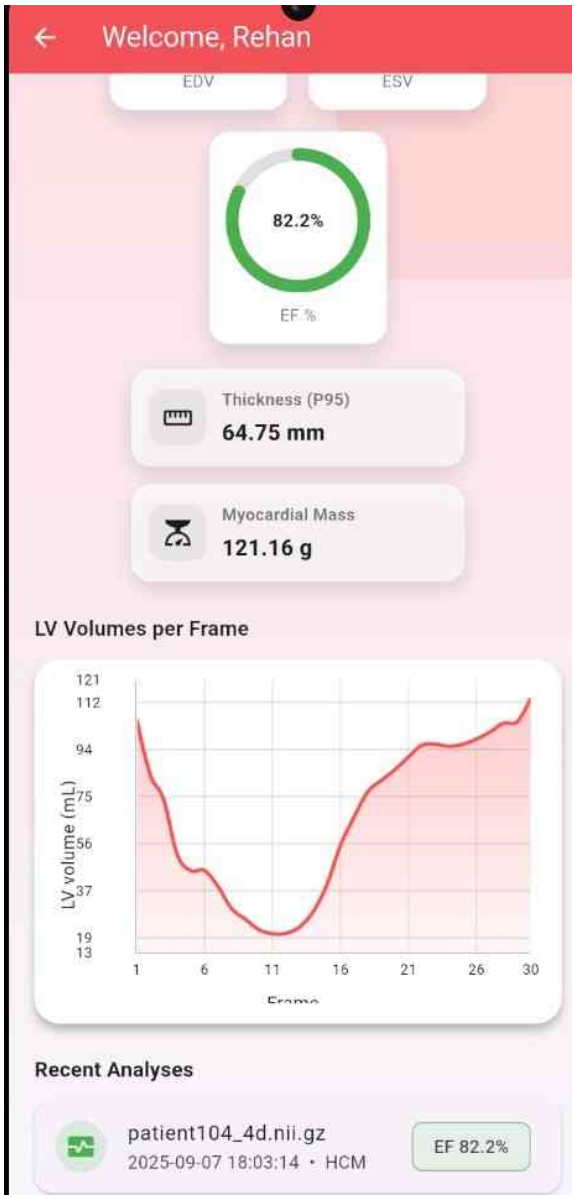

Figure A.6: LV volume curve and recent analyses list.

A.4 Tips & Troubleshooting

- **Slow upload?** .nii/.nii.gz cines often contain ∼50+ frames; prefer Wi-Fi or zip the study.
- **Unsupported file** or missing headers? the app will show a clear message, try another export or format.
- **Connection drop:** the upload retries automatically, you can cancel and resubmit if needed.

A.5 Privacy

Files are sent only for analysis and are cleaned up after processing.

References

[1] M. Irshad, M. Yasmin, M. I. Sharif, M. Rashid, and S. Kadry. A novel light u-net model for left ventricle segmentation using mri. *Mathematics*, 11:3245, 2023. Cited on pp. vii, 9, and 16.

[2] K. I. Chibueze, A. F. Didiugwu, N. G. Ezeji, and N. V. Ugwu. A cnn-based model for heart disease detection. Preprint / Online, 2024. CNN on MRI with additional features (incl. caffeine intake); trained on Enugu State University CAD Kaggle dataset (~90,000 samples); reported 94% accuracy. Cited on pp. vii, 1, 5, 9, 10, and 16.

[3] Rajpurkar, Pranav, Hannun, Awni Y., Masoumeh Haghpanahi, Codie Bourn, and Andrew Y. Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*, 2017. 64,121 ECG records from 29,163 patients; confirms cardiologist-level performance. Cited on pp. vii, 10, 11, and 17.

[4] Hyeonhoon Lee, Hyun-Lim Yang, Ho Geol Ryu, Chul-Woo Jung, Youn Joung Cho, Soo Bin Yoon, Hyun-Kyu Yoon, and Hyung-Chul Lee. Machine learning for real-time cardiac arrest prediction, 2018. In-hospital prediction using heart rate and EHR data from an NHS Foundation Trust; early identification of at-risk patients. Cited on pp. vii, 1, 11, 12, and 17.

[5] Muhammad Ali Muzammil, Saman Javid, Azra Khan Afridi, Rupini Siddineni, Mariam Shahabi, Muhammad Haseeb, F N U Fariha, Satesh Kumar, Sahil Zaveri, and Abdulqadir J Nashwan. Artificial intelligence enhanced electrocardiography for accurate detection of heart disease. *J Electrocardiol*, 2024. Analysis of ECG (IV/EV leads), large datasets, noise/missing data handling, and interoperability. Cited on pp. vii, 12, 13, and 17.

[6] PhD) Satoshi Kodera (MD, FJCC) Hiroshi Akazawa (MD, PhD, FJCC) Hiroyuki Morita (MD, PhD, and FJCC) Issei Komuro (MD, PhD. Prospects for cardiovascular medicine using artificial intelligence. *J Cardiol*, 2021. Review across ECG, echo, CT, etc.; covers data quality, robustness, and clinical integration. Cited on pp. vii, 13, 14, and 18.

[7] Tomofumi Nakamura and Tetsuo Sasano. Artificial intelligence and cardiology: Current status and perspective. *J Cardiol*, 2021. Reviews AI in ECG/electrophysiology; interpretability and trust; predicting demographics from ECG. Cited on pp. vii, 14, and 18.

[8] Alexander Schepart, Arianna Burton, Larry Durkin, Allison Fuller, Ellyn Charap, Rahul Bhambri, and Faraz S Ahmad 3. Artificial intelligence enabled tools in cardiovascular medicine. *Cardiovascular Digital Health Journal*, 2023. Qualitative study (cardiologists and health IT administrators); barriers include trust and interoperability; calls for standardized guidelines. Cited on pp. vii, 15, and 19.

[9] Victor Chang, Vallabhanent Rupa Bhavani, Qianwen Xu, and Alamgir Hossain. An artificial intelligence model for heart disease detection using machine learning algorithms. *Healthcare Analytics*, 2022. Random Forest on external databases (age, cholesterol, BP, etc.); 83% accuracy; Python-based healthcare application. Cited on pp. vii, 15, 16, and 20.

[10] Olivier Bernard et al. Deep learning for cardiac mri: Automated cardiac diagnosis challenge. In *STACOM Workshop, MICCAI*, pages 1–13, Quebec City, Canada, 2017. Cited on pp. 1 and 53.

[11] Simon Jégou, Michal Drozdzal, David Vázquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1175–1183, 2017. Cited on pp. 6, 7, 55, and 57.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. Cited on p. 6.

[13] Wenjia Bai, Matthew Sinclair, Giacomo Tarroni, and et al. Automated cardiovascular magnetic resonance image analysis with fully convolutional networks. *Journal of Cardiovascular Magnetic Resonance*, 20(1):65, 2018. Published 2018 Sep 14. Cited on p. 8.

[14] Cedars–Sinai Medical Center. Cedars–sinai ecg ai (presto): Predicting sudden cardiac arrest from ecg patterns. Ongoing research project / clinical AI initiative. AI models trained on hospital patient records (PRESTO) to predict SCA; focuses on subtle ECG patterns, explainability, and privacy. Cited on pp. 11 and 17.

[15] Flutter Team. Flutter: Build apps for mobile, web and desktop. https://docs.flutter.dev/, 2025. Documentation. Cited on pp. 51 and 60.

[16] Darcy Mason and the pydicom contributors. pydicom: Working with dicom files in python. https://pydicom.github.io/, 2021. Documentation. Cited on p. 51.

[17] Matthew Brett, Michael Hanke, et al. Nibabel: Access a variety of neuroimaging file formats. https://nipy.org/nibabel/, 2019. Documentation. Cited on pp. 51 and 52.

[18] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous systems. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016. Cited on p. 52.

[19] Sebastián Ramírez. Fastapi documentation. https://fastapi.tiangolo.com/, 2019. Accessed: 2025-09-10. Cited on p. 52.