SkinAI: Deep Learning for Skin Disease Classification

BY

Lau Jian Yi

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS) (INFORMATION SYSTEMS ENGINEERING)

Faculty of Information and Communication Technology

(Kampar Campus)

FEBRUARY 2026

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Information Systems (Honours) (Information Systems Engineering) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Dr Sayed Ahmad Zikri Bin Sayed Aluwee, for his ongoing support, motivation and important insights during the advancement of this final year project. The knowledge given and guidance have been essential in assisting me with my research on creating an AI model for classifying skin diseases.

I also appreciate the Faculty of Computer Technology of University Tunku Abdul Rahman for offering me the essential resources and knowledge to complete this work. I am also very grateful to my classmates who provided their insights and gave valuable feedback.

Moreover, I extend my heartfelt gratitude to my family and friends for their suggestions, support and love throughout this project's duration. This work would have been impossible without their moral support.

Finally, I sincerely acknowledge everyone who played a direct or indirect role in the success of this project. Their support and encouragement will always be valued and remembered.

ABSTRACT

Skin diseases are among the most prevalent health concerns worldwide, affecting millions annually and posing challenges in timely diagnosis and effective treatment. Access to dermatological expertise remains unevenly distributed, particularly in underserved and rural areas, where delays in diagnosis can lead to advanced complications and increased healthcare costs. Leveraging advancements in artificial intelligence (AI) and deep learning, this project introduces "SkinAI", a mobile-based application for skin disease classification.

This study highlights the importance of developing an AI-powered solution that bridges the gap between technology and real-world application. By integrating a deep learning- based model into a user-friendly mobile application, "SkinAI" empowers individuals and healthcare professionals with an accessible diagnostic tool for early skin disease detection. The proposed system aims to classify common skin conditions using a robust dataset, ensuring stable accuracy and adaptability across diverse skin types and diseases including eczema, psoriasis, basal cell carcinoma and viral infection. The contributions of this project extend to advancing AI in medical imaging, enhancing healthcare accessibility, and promoting cost-effective diagnostic solutions. "SkinAI" not only facilitates rapid skin condition analysis but also provides users with intuitive interaction, enabling non- technical users to leverage cutting-edge AI technologies effortlessly.

Area of Study: Deep Learning, Mobile Application

Keyword: Skin Disease, Medical Imaging, Deep Learning, Diagnostic Tools, Healthcare Accessibility, Artificial Intelligence, Skin Analysis, Skin Disease Classification

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\%$	Percentage
x	Multiple

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
ML	Machine Learning
DL	Deep Learning
ReLU	Rectified Linear Unit
MBConv	Mobile Inverted Bottleneck Convolution
KNN	K-Nearest Neighbors
SVM	Support Vector Mechanic
GLCM	Gray-Level Cooccurrence Matrix
GPU	Graphics Processing Unit
MOH	Ministry of Health
RGB	Red Blue Green
GAP	Global Average Pooling
5G	Artificial Intelligence
API	Convolutional Neural Network
CPU	Central Processing Unit
GPIO	General Purpose Input Output
IOT	Internet of Things
IP	Internet Protocol
RAM	Random Access Memory
TTA	Test-Time Augmentation

Chapter 1
Introduction

In this chapter, we present the background and motivation of our research, our contributions to the field, and the outline of the thesis. …….

1.1 Background Information

Skin is the covering layer of human body's surface, and it is the largest organ in human body. The main function of skin is to provide protection and receives sensory stimuli from external environment, mitigating the harmful impact on human body [1]. In human daily life, there are so many factors that will inadvertently or unintentionally affect the status of the skin, such as smoking, poor diet, poor-quality sleep, stress and anxiety, a sedentary lifestyle and sun exposure [2]. These conditions not only weaken the integrity of skin but can also lead to diseases that endanger human life.

Skin diseases are common in Malaysia and impose a notable public health burden. According to the National and hospital-based research, it indicates that eczema impacts approximately 13.4% of Malaysian children, making one of the highest figures noted in Asia [3]. Furthermore, the rates of having psoriasis in Johor Bahru increased from 0.27% to 0.51% from 2010 to 2020 [4], while 12.3% of restaurant workers in Peninsular Malaysia were found to have suspected occupational skin diseases [5]. Taken together, these researches highlight the importance of resources that can prompt identification of skin diseases whatever through online or offline channels. These data emphasize the necessity for different solutions to facilitate prompt and dependable disease identification. In recent, artificial intelligence (AI) development, especially deep learning, has demonstrated considerable potential in medical imaging that can provide automated solutions in assisting diagnosis of skin diseases.

Convolutional neural networks (CNNs) was one of the deep learning methods that profoundly impacted the field of image detection and classification. By progressively modifying the weight across various layers, CNNs can extract hierarchical features from images and make them particularly efficiency at recognizing skin conditions through clinical images. This project is aimed to develop an AI-based

application that can leverage deep learning methods like CNNs for accurate skin disease classification, connecting dermatological knowledge with practical healthcare demands.

1.2 Problem Statement and Motivation

Skin diseases affect millions worldwide, yet access to dermatology expertise remains unevenly distributed. The underprivileged population has restricted access to dermatology specialty care due to the socioeconomic status, living area and dermatologist location distribution [6]. In many rural areas, patients often experience delays in diagnosis or obtain inaccurate assessments due to the shortage of specialists. These difficulties may result in a decline in quality of life and higher medical expenses. In the case of contagious conditions, this may further escalate into public health risks. The need of effective diagnostic techniques with high accuracy of detection is required to increase the sizeable section of the population served while ensuring the diagnostic results are correct at the same time. Although recent advancements in AI show promise for medical image classification, the existing skin diseases classification models face limitations in Malaysia due to the variable accuracy, incompatible skin disease classes and insufficient practical usability for broader implementation in healthcare environments.

As an individual with an allergy-prone skin type, the past experiences further motivated the development of AI-based skin disease classification application that is more accessible, accurate and user-friendly. Due to the high expenses of medical consultations and the time required to have a call on specialist care, these often create uncertainty for patients about whether a skin disease warrants a dermatologist's visit. Moreover, most existing AI-based solution model are mainly intended for research and do not have user-friendly interfaces for non-technical individuals like patients. As a result, non-technical practitioners might be unsure about how to effectively engage with these systems, such as submitting an appropriate skin image for classification. Thus, this consider as a compelling reason to create "SkinAI" application that combines cutting-edge deep learning methods for accurate classification of prevalent skin diseases while also offering an easy-to-use platform. This proposed system would ultimately connect advanced AI technologies with practical dermatological requirements.

1.3 Project Objectives

The first objective of this project is to create and train deep learning models that can reliably classify four types of common skin diseases with consistent accuracy. The skin diseases classes include eczema, psoriasis, basal cell carcinoma and viral infections such as warts and molluscum. By utilizing CNNs and transfer learning, the aim of the project is to provide a dependable model that give reliable and consistent classification results. The used dataset also guarantees that the project stay aligns to the need of patients from regions around the world.

The second objective is to examine and compare three different deep learning algorithms such as MobileNetV2, ResNet50 and EfficientNetB2, to determine the best model for mobile use. This evaluation will enable the project to validate the selection of the most effective algorithm and offer perspectives on the trade-offs related to accuracy, efficiency and computational expense. In this way, it ensures that the final chosen model is both precise and fine-tuned for immediate application on smartphones with limited resources and environment.

The third objective centers on incorporating the chosen AI model into a mobile application, featuring an easy-to-use and accessible interface. This application aims to enhance the accessibility of AI-powered skin disease detection to public. In doing so, the application is expected to narrow the gap between advanced AI technologies and real healthcare requirements.

1.4 Project Scope and Direction

The scope of this project is to design an AI and mobile-based application named “SkinAI”. This application integrated a deep learning model to identify four common types of skin diseases. By analysing and comparing 3 different deep learning algorithms, a final working prototype that includes pre-trained CNNs, techniques for transfer learning and methods for images preprocessing to attain precise classification results will be chosen. However, the scope of this project is limited to disease classification only. Beyond classification, the model will not cover any complex medical diagnostics or medical image segmentation result.

The working prototype will further design into a mobile application with a simple and user-friendly interface to facilitate easy interaction for public. This

application will be able to run on common smartphone devices, ensuring broad usability and practicality. Users can either register a new account or continue in anonymous mode to access the application. They can upload images for classification and review their classification history. However, beyond the skin disease classification function, the application does not cover any other advance functions such as online medical consultations and medicine prescription features.

1.5 Impact, Significant and Contribution

Firstly, this project aims to contribute significantly to the field of medical image processing and AI-driven healthcare solutions. The core contribution is the exploration of deep learning methods to improve the efficiency and accuracy for medical image classification in the field of skin disease. By refining the existing algorithms and testing them against challenging datasets, it advances the technical capabilities of AI applications in medical field. The fast analysis and identification of skin disease will help people get a basic understanding of their skin threats and it assists people to make informed decisions about seeking professional dermatological care.

Secondly, another important contribution is the accessibility of people in healthcare. The lack of an intuitive user interface in many current AI models for skin disease medical picture classification renders them unsuitable for non-technical. In this project, the integration of deep learning model and intuitive application will be explored and provide convenience to public. With the help of this project, people could be able to utilize cutting-edge technologies without requiring specific technical knowledge and simply detect skin disease by himself even at home.

Lastly, this project seeks to address societal challenges by developing a scalable and reasonably priced solution for early skin disease classification. Many people, especially those in underserved communities, lack access to dermatologists or affordable healthcare. By focusing on skin disease classification using deep learning, contribution of development an AI-based tool can benefit a wider audience, offering a second opinion that is cost-effective, time-saving, and easy to use.

1.6 Report Organization

This report has been divided into seven chapters: Chapter 1, Introduction, Chapter 2, Literature Review, Chapter 3, System Methodology, Chapter 4, System Design, Chapter 5, System Implementation, Chapter 6, System Evaluation and Discussion, and Chapter 7, Conclusion and Recommendation.

Chapter 1 presents the background of the project, problem statement and motivation, project objectives, project scope and direction, project significance and contribution, and organization of the report. Chapter 2 discusses the associated concepts, technologies, and past works associated with the classification of skin diseases, deep learning, CNN models, and mobile-based healthcare applications.

Chapter 3 covers the system methodology, such as data set preparation, preprocessing, the model training strategy, evaluation metrics, system architecture, use case diagram, activity diagram and verification plan .Chapter 4 provides the system design of the SkinAI system such as the system block diagram, system component specifications, software component design and component interaction operations.

Chapter 5 covers the system implementation, such as the development environment, the integration of the TensorFlow Lite model, the implementation of the Android application, screenshots of system operation and problems faced during the implementation. Chapter 6 assesses and discusses the final system by providing model testing results, Android application testing, project challenges, system limitations, and objectives evaluation.

This chapter is the final chapter of the SkinAI project, which summarizes the entire development, performance, contributions of the project, limitations, and recommendations to develop the project further.

Chapter 2
Literature Review

2.1 Key Concept

2.1.1 Machine Learning

Machine Learning (ML) is a subfield of artificial intelligence (AI) that allows computers to identify patterns in data and make predictions without explicitly programming [7]. ML methods are typically categorized into supervised, unsupervised and reinforcement learning. While supervised learning is the most prevalent for classification tasks like medical diagnosis.

2.1.2 Deep Learning

Deep Learning (DL) is a specialized branch of ML. It leverages artificial neural networks with many layers, which also known as deep architectures, to learn complex hierarchical representation of data [8]. The difference is ML methods require manual feature extraction while DL models automatically extract features directly from raw data. This feature makes DL particularly more suitable for task such as image recognition and medical image analysis. General DL algorithms including convolutional neural networks, recurrent neural networks and transformer networks.

2.1.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of deep learning model specifically developed for the analysis of visual information [9]. CNNs consist of multiple essential layers:

1. Input image: Capturing visuals and text as pixels
2. Convolutional layers: A small filter moves across the images and detects features like edges or corners.

3. Activation Function (ReLU, Rectified Linear Units): Triggers the network by permitting only positive signals to go through.
4. Pooling layers: Reduce dimensionality while preserving important features.
5. Fully Connected layers: Combine extracted features for classification.
6. Output Layer: Produces the final result.

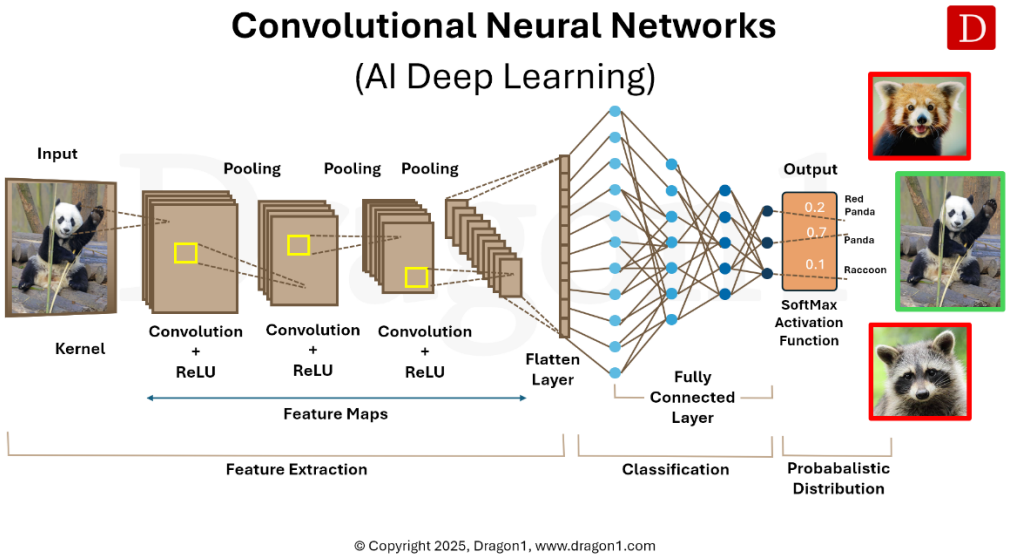


Figure 2.1 Convolutional Neural Network flow diagram

A CNN is made up of several layers that collaborate together to progressively gain a deeper understanding of an image. This enabled the network to initially learn basic features such as lines, then gradually more intricate shapes such as eyes and objects. Therefore, CNNs work much more efficiently and accurately in image classification and computer version.

2.1.4 MobileNetV2

MobileNetV2 is a lightweight 53 layers CNN model with an input size of 224 x 224 [10]. It is specially designed for mobile and embedded vision applications due to the low latency and low power consumption. By implementing the depth wise separable convolutions concept, it separated the filter and process the features efficiently. This reduces the number of parameters needed to perform precision training compared to

traditional CNNs. In addition, mobileNetV2 utilized inverted residual blocks with linear bottlenecks which features are expanded into a wider layer to capture richer patterns and then they are compressed back into smaller dimension using a linear activation to avoid losing import information.

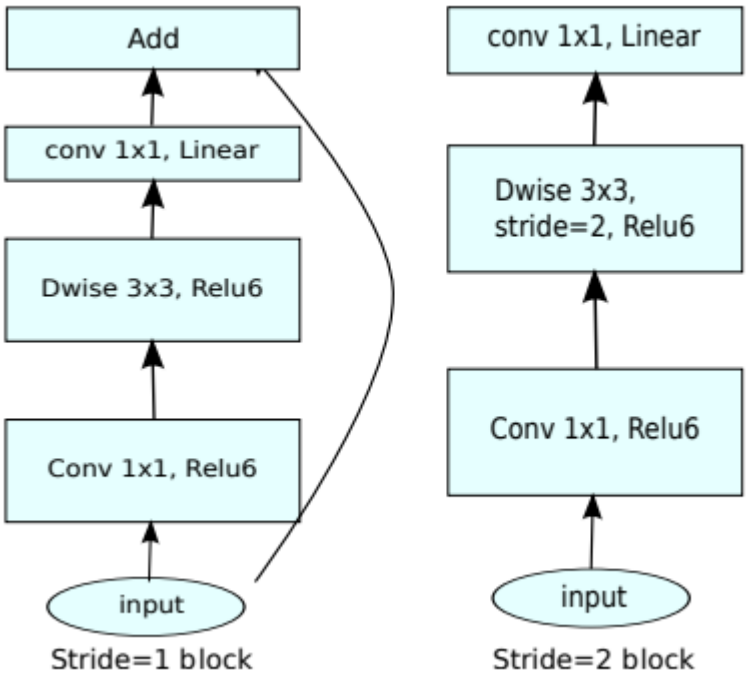


Figure 2.2 MobileNetV2 architecture diagram

These features help MobileNetV2 to achieves a strong balance between speed and accuracy, making it ideal for limited computational resources [11].

2.1.5 ResNet50

ResNet50 is one of the most widely used versions in the family of Residual Network family. It consists of 50 layers with an input size 224 x 224, each structured into a series of bottleneck residual blocks. ResNet50 pioneered the concept of skipping connections to address the vanishing gradient problem in deep networks. This concept allows the input data to bypass one and more layers, making the gradients flow more directly during training. Thus, the ultra-deep architecture will be more feasible and accurate [12].

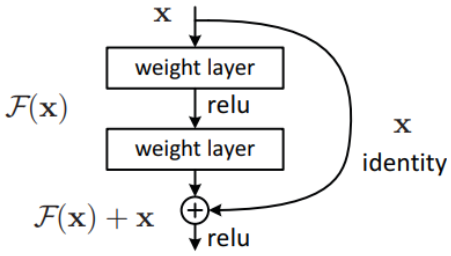


Figure 2.3 ResNet50 architecture diagram

Resnet acts as a solid baseline model for various image recognition tasks because of its effective feature extraction and impressive accuracy. However, the performance is directly proportional to the computational expenses.

2.1.6 EfficientNetB2

EfficientNetB2 is the scaled-up variant within the EfficientNet family. The architecture of EfficientNetB2 is based on Mobile Inverted Bottleneck Convolution (MBConv) blocks which similar to those found in MobileNetV2. EfficientNetB2 uses an input size of 260 x 260 which is larger than the baseline EffcientNetB0 that uses 224 x 224, allowing it to capture finer spatial details in images. Unlike traditional CNNs that scale by only increasing depth and weight, EffcientNet employs a compound scaling method. This method scales the depth, width and input resolution in a coordinated manner, enabling the network to attain high precision while remaining efficient [13].

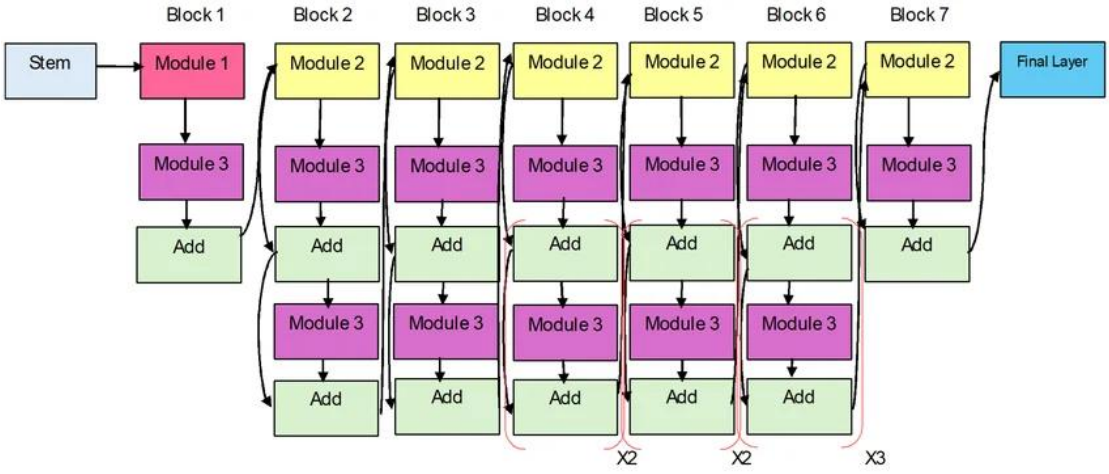


Figure 2.4 EfficientNetB2 architecture diagram

EfficeintNetB2 has demonstrated strong performance in transfer learning scenarios. It delivers accurate classification with minimal computational costs when pre-trained on large datasets like ImageNet and fine-tuned on smaller and domain-specific datasets. This trade-off between accuracy and efficiency makes EfficientNetB2 a viable option for skin disease classification, especially for applications that may be deployed on resource-limited or mobile devices.

2.1.7 Transfer Learning

Transfer learning is a machine learning approach where knowledge acquired from one task is used to enhances model performance on another related task [14]. Since training CNNs from scratch requires massive labeled datasets and high computational resources, transfer learning addresses this challenge by reusing knowledge from pre-trained models that have been trained on large datasets such as ImageNet. This approach speeds up the development process, minimized computational requirements and greatly improves the model performance.

2.2 Previous work on skin diseases classification

There are few ways to classify the skin diseases where it depends on traditional or modern technologies. With the rapid development of the artificial intelligence and deep learning technologies, the field of automated skin disease classification has been greatly improved. These methods provide patient a valuable support for dermatologists while ensuring a stable accuracy in diagnosing various skin conditions at the same time. This section examines relevant research and categorizes them based on their assessment criteria, dataset, feature extraction methods and classification algorithms.

2.2.1 Physical Examination with Clinical Expertise

Firstly, dermatologists would assess the patient's medical record to discover the possibility of genetic disease. After that, a full skin examination including examination of scalp, nails and mucous membranes will be carried out to assist dermatologists to determine the type of skin condition [15]. By observing the physical characteristics of

the skin lesions such as color, shape, size and location of the abnormality, dermatologists will have a preliminary diagnosis on the type of skin disease. In cases where is serious conditions or diagnosis cannot be make based just on the appearance of the skin lesion, advance tests should be conducted to further investigate on it.

This hands-on examination requires experienced dermatologists to accurately distinguish between different skin diseases because it is challenging given the overlap in symptoms across various conditions such as color, texture and other features. However, the approach of making diagnoses based on experience and knowledge falls well short of providing patients with the medical resources they need as the dermatologists' growth rate and the incidence rate of skin disease is mismatch at the time.

2.2.2 Histopathology

Histopathology is the cornerstone in traditional skin disease diagnosis, especially to detecting cancerous lesions. This technique involved taking a biopsy which is a small piece of skin from the affected skin area and examining it under a microscope [15]. In the process, dermatologists will numb a small area of skin using a scalpel or punch tools and sent to a laboratory for examination after removing a piece of skin.

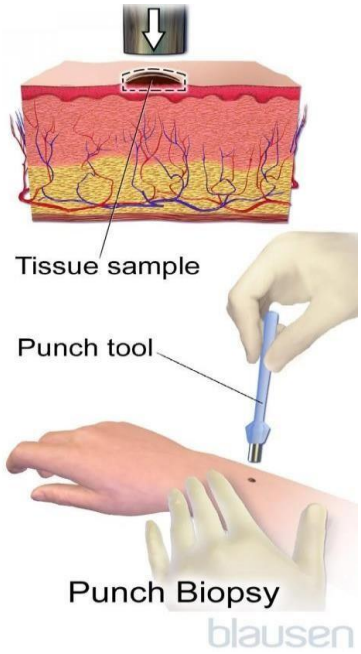


Figure 2.5 Histopathological examination

Although this method provides a high accuracy, histopathological examination is time-consuming and requires the results to be interpreted by qualified pathologists. It takes 4 to 5 days to generate the result report which requires a large amount of time, and it may affect the patient getting cure on time. Besides, histopathological examination can be costly and invasive due to the need for laboratory investigations.

2.2.3 Skin Diseases Detection Using Image Processing in Machine Learning

Jagdish etal. [16] proposed a model to detect and classify skin diseases by applying advanced image processing and machine learning techniques. This study is only focusing on two types of skin diseases, Basal and Squamous cell diseases, and the investigation process was divided to four stages which is image preprocessing, segmentation of image using fuzzy clustering, feature extraction, and classification. Data size is 50 images of sample, collected from 50 patients from the hospital.

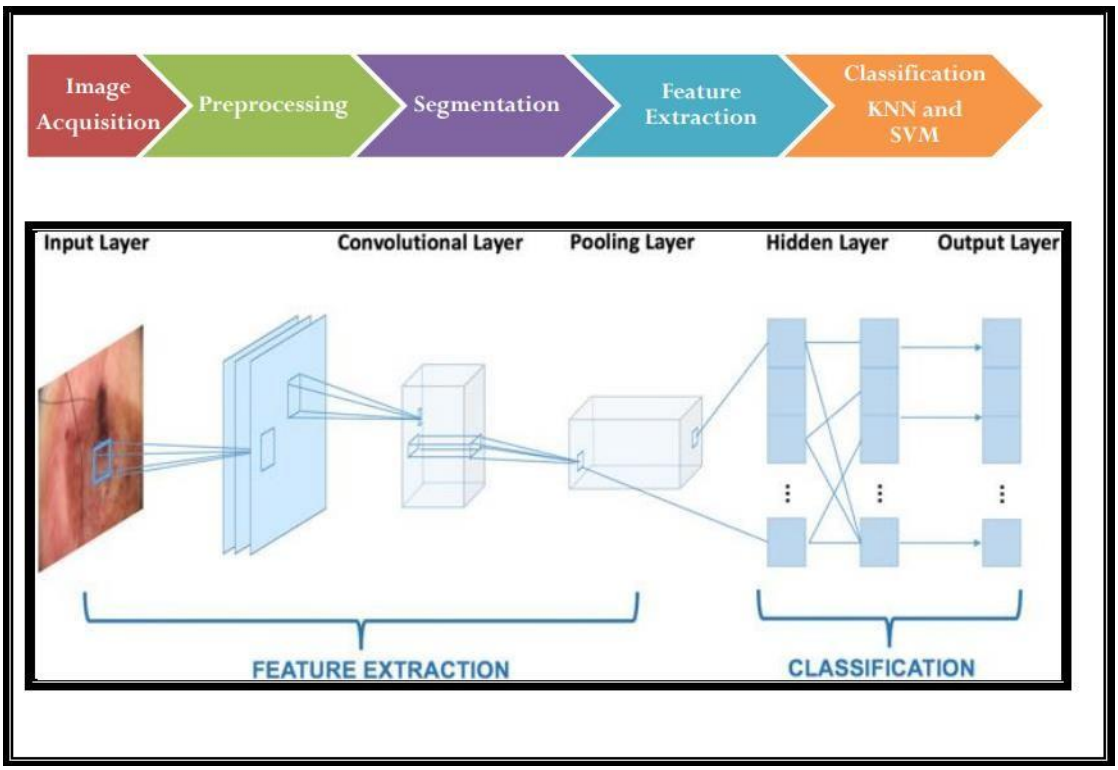


Figure 2.6 Review model 1 progress cycle

The methodology is the highlighted point of the study. By employing a step-by-step process, Jagdish started with image preprocessing which is a method to remove

background noise from images by using median filter. Then, they use advanced techniques like fuzzy clustering for segmentation and wavelet transform for feature extraction have further underscores the methodological rigor. Moreover, a comparative analysis of K-Nearest Neighbors (KNN) and Support Vector Mechanics (SVM) was conducted to offer a valuable insight into their relative effectiveness in the context of skin disease detection. As a result, KNN demonstrated higher accuracy with performance of 91.2% which indicated as a reliable result for practical applications.

Algorithm	Feature Extraction	Wavelet-based SYMLET Analysis						
		sym2	sym3	sym4	sym5	sym6	sym7	sym8
KNN Classification	R _Color	77	67	87	93	76	54	75
	G _Color	87	65	65	67	88	76	48
	B _Color	68	88	85	77	64	53	64
	Mean_ Value	66	86	65	65	65	85	65
	SD_Value	85	92	73	54	63	76	81
	Entropy	54	66	82	88	66	86	56
	Ellipticity	73	55	76	64	75	96	79
	Intensity	45	76	46	88	74	75	46
	Coefficient with Correlation	65	54	72	64	56	74	56
Overall Accuracy Using KNN Classification								
		sym2	sym3	sym4	sym5	sym6	sym7	sym8
Basal Disease Accuracy (%)		87%	40%	88%	91.2%	77%	40%	78%
Squamous Disease Accuracy (%)		34%	89%	50%	39%	54%	84%	60%
	Feature Extraction	sym2	sym3	sym4	sym5	sym6	sym7	sym8
SVM Classification	R _Color	57	20	86	53	32	34	56
	G _Color	53	65	53	75	64	24	74
	B _Color	86	77	54	73	53	46	69
	Mean_ Value	85	43	68	32	46	57	67
	SD_Value	74	65	67	75	47	85	54
	Entropy	29	64	25	75	46	56	53
	Ellipticity	83	85	53	56	49	53	45
	Intensity	63	54	85	68	52	86	76
	Coefficient with Correlation	76	84	36	85	85	84	87
Overall Accuracy Using SVM Classification								
		sym2	sym3	sym4	sym5	sym6	sym7	sym8
Basal Disease Accuracy (%)		77%	50%	79%	85 %	83%	76%	56%
Squamous Disease Accuracy (%)		34%	79%	60%	69%	54%	83%	60%

Table 2.1 Result of review model 1

Despite its strengths, the major limitation of this study is the used of small dataset size which consists of only 50 images. The limited sample size will raise concerns about the generalizability of the result, especially when it applies to bigger datasets or diverse populations. Furthermore, the reliance on machine learning algorithms such as KNN and SVM is effective for structured data but not fully exploit

to hierarchical features present in medical image. By using machine learning, the study will need to develop extensive manual feature engineering which is both time-consuming and prone to human error.

2.2.4 Skin Disease Detection Using Convolutional Neural Networks (CNNs)

T.Shanthi, R. S. Sabeenian and R.Anand [17] introduced a study about developing a computer vision system for automatic diagnosis different type of skin diseases. By utilizing an 11-layer CNNs architecture such as convolution layer, rectified linear unit layer (ReLU), pooling layer and fully connected layer, it validated on DermNet database images to classify common skin disease such as acne, keratosis, eczema herpeticum and urticaria. Each of the image classes contain around 30 to 60 different samples. As a result, the model demonstrated high classification accuracy ranging from 98.6% to 99.04%, showing the potential deep learning techniques in dermatological diagnosis.

Classification of skin disease images		**Predicted class**				**Per class accuracy**
		Acne	**Keratosis**	**Eczema herpeticum**	**Urticaria**	
Actual class	Acne	12	2	0	0	85.7
	Keratosis	1	24	0	1	92.3
	Eczema herpeticum	1	0	14	0	93.3
	Urticaria	0	0	1	13	92.8

Table 2.2 Result of review model 2

The significant strength of the study is the usage of CNNs layer. It proven the effectiveness in automatically extracting hierarchical features from images without the need for manual feature engineering which belongs to machine learning. As CNNs are deep learning models which capable of automatically learning important features from raw images, it overcomes the limitations of using traditional texture-based analysis such

as Gabor filters and Gray Level Co-occurrence Matrix (GLCM). CNNs adapt to differentiate the variation in skin tone, texture and lesion appearance without requiring manual intervention. Therefore, CNNs is exploited to better suited for tasks like skin disease classification compared to machine learning methods to handle. Besides, DermNet is a publicly reliable database platform which it enhances the reproducibility and credibility of the result.

While the limitation of the study is the computational requirements. Training CNNs is computationally expensive, especially for deeper layer of model. A powerful GPUs and suitable environment must be set as a goof barrier for some researches or institutions. It also takes significant processing time for training the deep learning machine. Besides, domain expertise is still needed for the pre-processing steps in order to make an image enhancement or augmentation for higher accuracy.

2.2.5 Skin Disease Detection Using Deep Learning and Ensemble Learning

Anabik Pal et al. [18] explores an ensemble deep learning approach to classify skin disease using dermo copy images. The main approach of this study is to combine three-state-of-the-art CNN architecture including ResNet50, DenseNet121 and MobileNet, to classify seven types of common skin disease. The dataset used is available from ISIC 2018 website with 10,015 training images and 193 validation images which is considered as high valuable dataset that can provide large insights. Lastly, majority voting technical will be used to represent as the final result of skin disease detection test. In this study, it is proven that ensemble methods performed a better performance than individual models which 0.775 accuracy was achieved. When evaluated separately, ResNet50 attained an accuracy of 0.773 but demanded significant computational resources, DenseNet121 reached a moderate accuracy of 0.698 while MobileNet recorded the lowest accuracy at 0.597.

EXPERIMENTAL RESULT

Sl No.	Method	Score
1	ResNet-50	0.773
2	DenseNet-121	0.698
3	MobileNet	0.597
6	Ensemble (1,2)	0.769
5	Ensemble (2,3)	0.715
4	Ensemble (1,3)	0.771
7	Ensemble (1,2,3)	**0.775**

Table 2.3 Result of review model 3

The strength of this study lies in the use of advanced diagnostic models like ensemble learning, transfer learning, state-of-the-art architecture that leverage deep learning to identify skin disease with higher accuracy.

However, the ensemble method is time-consuming and computationally expensive due to the need of training multiple models and combining their outputs. This raised the problem of heavy system and memory needed. Therefore, although the method demonstrated high accuracy, it might not be appropriate for use in resource-limited environments like mobile application, where efficient and lightweight models are favored.

2.2.6 Skin Disease Detection Using CNN in Mobile-based

Maduranga etal. [19] presented an artificial intelligence based mobile application to detect seven types of skin diseases. By using CNNs, MobileNet and transfer learning technologies, different skin disease on the HAM10000 dataset can be classify with an accuracy of 85%.

	Precision	Recall	F1-score	support
akiec	0.73	0.46	0.57	69
bcc	0.83	0.67	0.74	93
bkl	0.65	0.77	0.71	228
df	0.81	0.61	0.69	28
mel	0.73	0.69	0.71	226
nv	0.94	0.95	0.94	1338
vasc	0.95	0.86	0.90	21
Micro avg	0.87	0.87	0.87	2003
Macro avg	0.80	0.72	0.75	2003
Weighted avg	0.87	0.87	0.86	2003

Table 2.4 Result of review model 4

On the other hand, a mobile application interface is built to integrate with the model for quick and accurate action. This approach brings convenient for general practitioners, allowing them to quickly diagnose skin diseases even using smart phone.

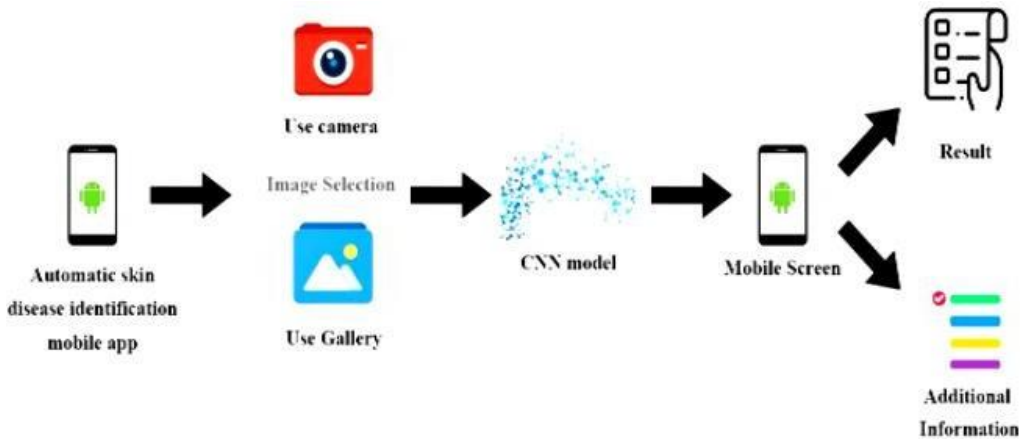


Figure 2.7 Mobile app progress cycle of review model 4

Although the research displays encouraging results, certain limitations can be recognized. In order to have a well fit in mobile application environment, this research chose MobileNet as the favored model compared to other options. The lack of presenting a thorough comparison with other deep learning models and robust evidence tables to back the selection raised concerns about whether MobileNet was the best architecture for achieving a balance between accuracy and efficiency in mobile applications. This could greatly reduce the reliability of the research result. Furthermore, while the dataset comprises seven skin diseases, these diseases may not correspond with the most frequent or urgent skin diseases found in Malaysia. In this way, it restricts the local relevance of the solution since healthcare requirements can vary by region.

2.3 Fact Finding

The global shortage of dermatologists is a popular public health challenge whose impact extends far beyond any single country. The shortage is most happened in sub-Saharan Africa where these countries have fewer than one dermatologist per million people. This causes large segments of the population cannot access specialized care in time. Not only that, Asia which accounted for the largest share of the global population, there are still many countries that have fewer than one dermatologist per 100,000 people

and most specialists are highly concentrated in major cities only [20]. Based on the research, the Global Burden of Disease Study 2021 ranked skin diseases as one of the top ten causes of non-fatal disease burden. However, many resource limited regions are continuing facing severe shortages of dermatologists, dermatology training and underdeveloped health infrastructure, limiting public access to diagnosis and consult medications [21].

This global mismatch showed that patients in rural areas are frequently experiencing delayed diagnoses or they often rely on general practitioners that does not have professional dermatological training to manage complex skin conditions. Malaysia represents one example of this broader global trend. The official Ministry of Health (MOH) reported an uneven distribution of dermatologists across multiple areas, with certain states facing shortages of specialists and depending on contracted dermatologists [22]. This deficit frequently results in extended wait times, especially in rural and semi-urban regions. Furthermore, a study conducted in the Petaling District showed a notable link between individual awareness of skin infections and their quality of life, suggesting that insufficient public awareness is a major obstacle to timely diagnosis and successful treatment [23].

To tackle this problem, a simple mobile application could aid in encouraging early diagnosis because younger people generally favor easy digital options rather than spending time waiting in hospitals.

2.4 Data Collection

Skin diseases represent one of the most significant health burdens globally. The Global Burden of Disease Study 2021 reported a global incidence of 2 billion prevalent cases and 41.9 million disability-adjusted life years (DALYs) due to skin diseases with incidence and prevalence more than 35% higher than in 1990 [24].

Among the overall burden of skin disease, several common conditions deserve particular attention, including skin cancer, eczema, psoriasis and viral infections. Basal Cell Carcinoma (BCC) is the most common type of skin cancer. It makes up about 80% of all non-melanoma skin cancer cases and the number of cases is increasing by around 3–10% each year [25]. Eczema is also a very common skin disease. It affects around

11% of children and about 6% of adults worldwide [26]. In general, up to 20% of children and 10% of adults may experience eczema at some point, showing the infectivity of this disease [27]. Psoriasis is a long-term condition caused by the immune system and is recognised as a serious noncommunicable disease. According to the World Psoriasis Day consortium, 2 to 3% of the total population which approximately 125 million people around the world have affected by psoriasis [28]. In additional, viral skin infections that include warts and molluscum are also a very common skin disease. In 2021, there were about 84.7 million new cases worldwide with over 130 million total cases [29]. All of this evidence shows that these four skin diseases are common and affect many people worldwide.

In addition, technological advancements further strengthen the justification for AI-based approaches. Reliable platforms such as Kaggle and GitHub provide free access to comprehensive datasets that support deep learning model development. For instance, the HAM10000 dataset, which contains more than 10,000 images of skin conditions, has become a standard benchmark for training CNNs in dermatology [30]. Moreover, both lightweight and heavy deep learning architectures have been developed to balance computational cost and accuracy, making them suitable for different environments. This indicates that the resources required for training and deploying mobile AI applications in dermatology are already sufficient and accessible.

2.5 Critical Remark for Previous Work

From the literature review, traditional methods like physical exams and histopathology offer high precision but are constrained by their invasive nature expense and long processing durations. This renders them impractical for rapid screening in resources limited areas. The goal of this “SkinAI” proposed project will overcome this shortage by emphasizing lightweight and rapid AI models suitable for mobile deployment to achieve faster outcomes.

Approaches based on machine learning like Jagdish et al. showed that combining feature engineering with classifiers such as KNN and SVM can reach notable accuracy levels. Nonetheless, the research dependence on a limited dataset and manual feature extraction restricts both scalability and resilience. To address this issue, the proposed project will utilize extensive publicly datasets from Kaggle and apply

transfer learning with CNNs architecture. In this way, it minimized the requirement for manual feature extraction and enhancing generalizability.

On the other hand, CNN based studies like Shanthi et al. demonstrated the effectiveness of deep learning in feature extraction and attaining exceptionally high precision. However, the deep learning model consists 11 layers of CNNs only which the small datasets was trained on superficial of CNN architectures. This could raised issues regarding scalability and performance in real world scenarios. The proposed project is differed by testing more advanced architectures like ResNet, MobileNet which have deeper CNN layers and can demonstrate greater effectiveness in managing intricate image classification challenges.

The CNN based research conducted by Anabik Pal at al. emphasized the benefits of ensemble learning. By comparing the final accuracy of different combinations of algorithms, it proven that ensemble learning achieved greater accuracy than standalone models. Individual models such as MobileNet, ResNet and DenseNet did not produce equally high accuracy levels. Even with this benefit, the considerable time and computational expenses of ensemble methods render them less appropriate for mobile apps. To achieve a balance between precision while minimizing the strain on mobile device resources, the proposed project focuses on the analysis of lightweight but efficient single-model designs by utilizing the layers and weights of CNNs architectures.

Finally, the research by Maduranga et al. effectively showcased the possibility of incorporating a CNN model into a mobile application and it was closely matches the goals of the proposed project. However, the research did not offer comparative findings on MobileNet model with other deep learning models. Moreover, the dataset utilized contained skin diseases that might not reflect the most prevalent skin diseases in Malaysia. To overcome these limitations, the proposed project intends to carry out a comparison between different deep learning models while concentrating on a more specific set of common skin diseases pertinent to the local environment in Malaysia.

Chapter 3
System Methodology

This chapter presents the system methodology and design used to develop the AI-based mobile application for skin disease classification, SkinAI. It explain clearly how the deep learning model and Android application were planned and structured before implementation.

3.1 Project Methodology

The general methodology of this project is structured and organized into a pipeline that links development of deep learning models and implementation of an Android mobile application. The initial stage is concerned with the development of deep learning models, whereas the second stage is concerned with the integration of Android applications. During the model development stage, the skin disease dataset is prepared, preprocessed, augmented, and used in training the three CNN models, including MobileNetV2, ResNet50, and EfficientNetB2. Training of these models is done through transfer learning and compared using the same method of evaluation to ensure fairness.

The training and testing of the models are followed by the process of selecting the best model according to the accuracy of the classification and the ability to deploy the models in a mobile setting. The chosen model is then translated into the TensorFlow Lite and embedded into the Android application. During the application phase, the user is able to log-in with Google sign-in or anonymous, accept the terms and the disclaimer, select or capture a skin image, perform a classification, view the result and save scan records using Cloudinary and Firebase Firestore.

3.1.1 Dataset Preparation

The dataset used in this project was obtained from Kaggle [31]. There is total of four types skin diseases, including eczema, psoriasis, basal cell carcinoma and viral infection like warts and molluscum (See Appendix A)

These four classes were selected because they represent common types of skin-related conditions like inflammatory skin diseases, cancer-related lesions, and infection-related skin

conditions. In this project, the data of the four diseases was set up to undergo supervised classification of images where each image is allocated to either of the four target classes.

Skin Disease Classes	**Number of Images**
Eczema	1,677
Psoriasis	2,055
Basal Cell Carcinoma	1,600
Viral Infections	2,103
Total	**7,435**

Table 3.1 Dataset Distribution

The dataset was maximally expanded to improve model learning and classification performance. The final dataset contains a total of 7,435 images with eczema class containing 1,677 images, psoriasis contains 2,055 images, basal cell carcinoma contains 1,600 images, and viral infections contain 2,103 images.

Dataset Split	**Percentage (%)**	**Number of Images**
Training Set	70	5,204
Validation Set	15	1,115
Testing Set	15	1,116
Total	**100**	**7,435**

Table 3.2 Dataset Separation

The dataset was divided into training, validation, and testing sets using a 70:15:15 ratio. The training set consists of 5,204 images, the validation set consists of 1,115 images, and the testing set consists of 1,116 images. In this project, training dataset was used to train the CNN models and validation dataset was used to track the performance of the model in training. Testing dataset was not used during training but was used to test the final model after it finished

finalizing. The data separation progress is significant because it enables the model performance to be measured using unseen data, which gives a more accurate pointer to the model generalization.

3.1.2 Dataset Preprocessing and Augmentation

Data preprocessing was performed to make sure that all the images were compatible with the input requirements of the CNN models. Thus, all three models were fed with this input size in order to have consistency and fairness during model comparison.

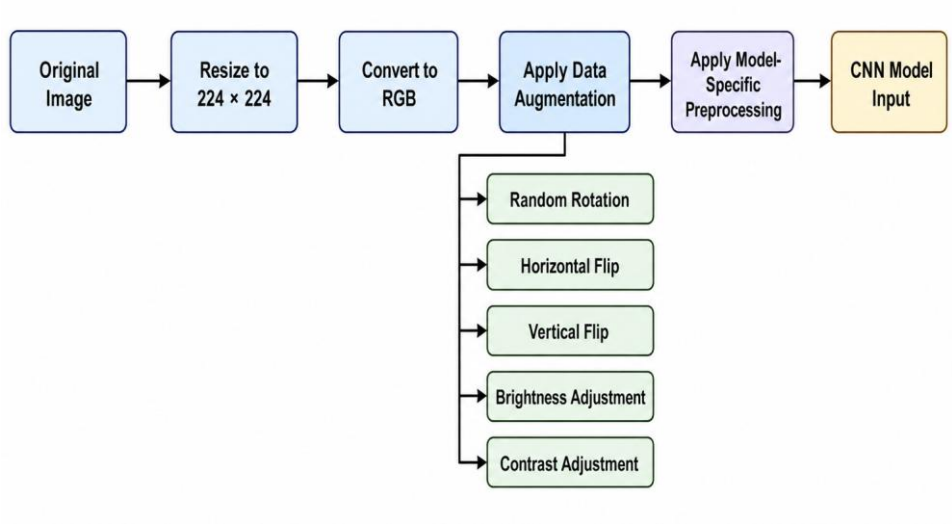


Figure 3.1 Data Preprocessing and Augmentation Pipeline

The figure above shows the preprocessing and augmentation pipeline used in this project. All the images were scaled to 224 x 224 pixels and transformed into RGB format before being used as model input. After that, data augmentation was done on the training data. This is intended to make images more varied and to decrease the probability of overfitting. The augmentation techniques that were applied in this project were random rotation, horizontal flipping, vertical flipping, brightness adjustment and contrast adjustment. Random rotation was done on 0°, 90, 180 or 270 degrees of rotation. Maximum delta of brightness was set to 0.15 and contrast adjustment was set to 0.85 to 1.15.

After augmentation, model-specific preprocessing is applied according to the selected CNN architecture. For example, MobileNetV2 used the MobileNetV2 preprocessing function while ResNet50 and EfficientNetB2 will also use their respective preprocessing function. This is to

make sure that the pixel values are converted to the desired format of each pretrained model. Overall, this pipeline helps improve model generalization and reduces the risk of overfitting.

3.1.3 Transfer Learning of CNN Models

In this project, transfer learning was implemented to enhance the performance of the pretrained models and eliminate the necessity to train CNN models from scratch. Deep CNN model training is expensive, both in terms of data and computational capacity. Thus, pretrained CNN models were used as backbone feature extractors. These pretrained models were trained with ImageNet weights and so they had already learned on the general image concepts like edges, textures, shapes, and patterns from a large-scale image dataset.

This project compared three CNN architectures that were trained, including MobileNetV2, ResNet50, and EfficientNetB2. During the initial phase of the project, EfficientNetB0 was taken into consideration because of its small architecture and ability to run on a mobile platform. Nevertheless, as the development phase proceeded, EfficientNetB2 was added as the refined EfficientNet candidate to enhance the eventual model comparison. Thus, the last model comparison is MobileNetV2, ResNet50, and EfficientNetB2.

A custom classification head was connected to each CNN backbone. The head of the classification was to map the extracted image features into four output classes which are eczema, psoriasis, basal cell carcinoma and viral infection. The last output layer employed the use of Softmax activation to generate probability scores of the four disease classes.

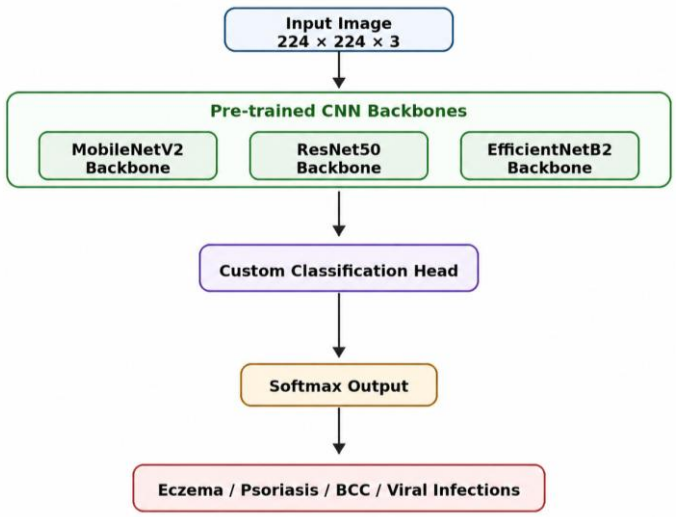


Figure 3.2 Transfer Learning Approach

The above figure illustrates the transfer learning approach used in this project. The input image is first passed into a pretrained CNN backbone which will extract the visual features of the skin image. Then, the extracted features are passed into a custom classification head that uses Softmax to calculate the probability scores and maps them into four categories of skin diseases. The design of the classification head is the same in all the candidate models to facilitate fair comparison. This will enable the models to reuse the ImageNet knowledge that has been previously trained while adapting the last layers to fit the skin disease classification problem.

3.1.4 Training Strategy

The CNN models were trained using a three-stage training strategy. This strategy was designed to slowly tune the pretrained models to the skin disease data. The candidate models were put through the same training strategy so that there was fair comparison.

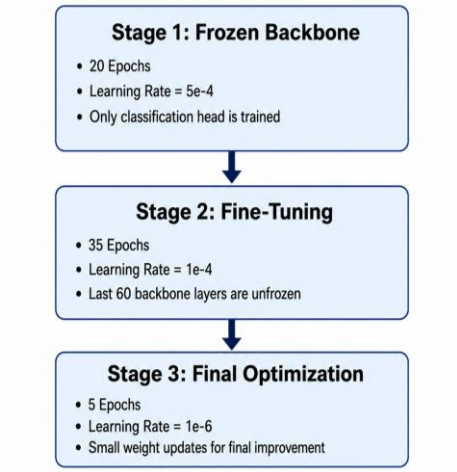


Figure 3.3 Three-stage Training Strategy

The above figure shows the three-stage training strategy used in this project.

In stage 1, CNN backbone was frozen and the only part that was trained was the custom classification head. This enabled the classification head to learn the theory of mapping the extracted features on the four skin disease classes without modifying the pretrained backbone weight. This stage was trained over 20 epochs with a learning rate of 5e-4.

In stage 2, the last 60 layers of the CNN backbone were unfrozen for fine-tuning. This enabled the model to fine-tune some of the pretrained feature extraction layers so that it can fit

the skin disease dataset. This stage was trained with a lower learning rate of 1e-4 to prevent significant updates on the pretrained weights and it was trained with 35 epochs.

In stage 3, final optimization was done with a very small learning rate of 1e-6 in 5 epochs. This step enabled the model to take minor changes in the weights that had been learned without destabilizing the training process. There was a total of 60 training epochs will be taken for each model training.

3.1.5 Overfitting Prevention Techniques

One of the key issues with CNN-based medical image classification is overfitting. It arises when the model is trained on training data in a manner that gives it a good fit but fails to perform well on unseen data. As the image of skin diseases can be different based on lighting, angle, background, skin color and the manifestation of the disease, the model should be trained to generalize outside the training set. Various methods were used to minimize overfitting and enhance the generalization of the models.

Technique	**Setting**	**Purpose**
Dropout	• 0.5 in classification head	• Reduces reliance on specific neurons
L2 Regularization	• 1e-4 (0.0001)	• Penalizes overly complex weights
Label Smoothing	• 0.1	• Reduces overconfidence in prediction
Early Stopping	• Patience = 5 epochs	• Stops training when validation performance stops improving
ReduceLROnPlateau	• Factor = 0.5 • Patience = 3 epochs	• Reduces learning rate when progress slows
Class Weights	• Balanced from training data	• Reduces effect of class imbalance

Table 3.3 Overfitting Prevention Techniques

The above table is a summary of the overfitting prevention techniques in this project. The complexity of models and overconfident predictions were minimized with dropout, L2 regularization and label smoothing. While the training process was controlled with early stopping and ReduceLROnPlateau. The imbalance of classes was also reduced by applying balanced class weights to eliminate the influence of class imbalance when training.

3.1.6 Test Time Augmentation

Test Time Augmentation is also referred to as TTA which was used in the evaluation phase of the model. In TTA, there are multiple versions of the same test image applied to the model and the results of the prediction are averaged to create the final output.

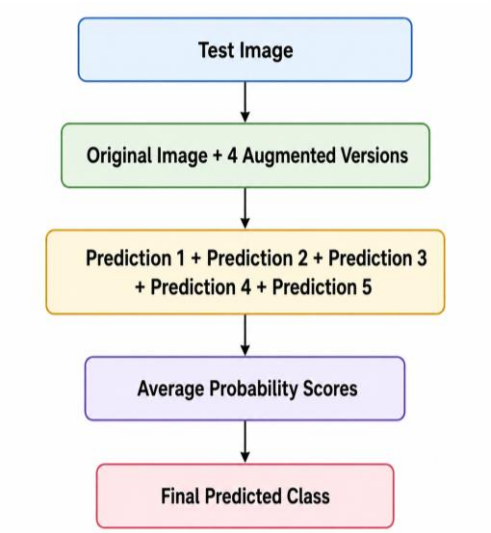


Figure 3.4 Test Time Augmentation Process

The figure above shows the TTA process used during model evaluation. The system produces predictions on the original image and four augmented versions of the test image rather than producing a prediction of only one version of the test image. The average of the five prediction outputs is obtained to produce the overall probability score. This technique enhances the accuracy of prediction since the final prediction result is not as influenced by slight changes in image position or look.

3.1.7 Evaluation Metrics

The performance of the trained models was evaluated using several evaluation metrics. The overall percentage of correctly classified images was measured by accuracy. Nonetheless, accuracy cannot be a sufficient criterion since the performance in each disease classification can vary. Thus, precision, recall, and F1-score were utilized as well.

Metric	Description	Purpose
Accuracy	• Percentage of total correct predictions	• Measures overall model correctness
Precision	• Correct positive predictions over total predicted positives	• Measures reliability of predicted class
Recall	• Correct positive predictions over total actual positives	• Measures ability to detect each class
F1-score	• Harmonic mean of precision and recall	• Measures balanced class performance
Confusion Matrix	• Actual classes compared with predicted classes	• Identifies class-level misclassification
Model Size	• File size of trained or converted model	• Determines mobile deployment suitability
Inference Time	• Time required to generate prediction	• Determines mobile application responsiveness

Table 3.4 Evaluation Metrics

The table above shows the measures that were applied to compare the candidate models and justify the choice of the model. In addition to classification measures, model size was also considered since the final system will be deployed to the mobile platform. A highly accurate but large file size model might not be practical to use in a mobile environment. Thus, the model comparison took into account not only the performance of classifications but also the mobile suitability.

3.2 System Design Diagram

The development of SkinAI has two key stages. The first stage is the deep learning model development phase. It involves gathering, preprocessing, and augmenting the images of skin diseases and using them to feed three different CNN models. The second stage is the deployment to a mobile application phase, where the selected trained model is converted to TensorFlow Lite format and deployed into an Android application.

3.2.1 System Architecture Diagram

The system architecture diagram shows the overall system design pattern for the proposed AI-based skin disease classification application. It consists of three interconnected working units that work together to support the entire workflow of the system.

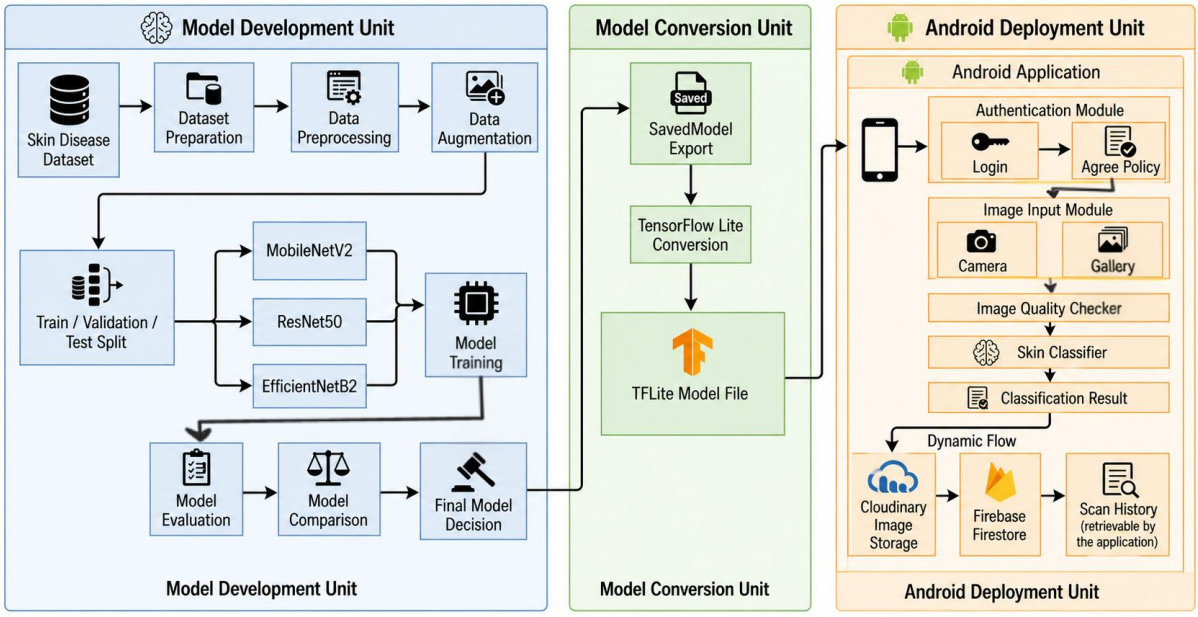


Figure 3.5 System Architecture Diagram

The complete architecture pipeline starts with the input dataset. A large dataset of skin disease image data is provided for training, validation and testing. Before feeding the training models, the images are resized, converted to RGB format, preprocessed and augmented to enhance model generalization. By using transfer learning technique, three different architecture CNN models were trained, which are MobileNetV2, ResNet50 and EfficientNetB2. The models are then compared in terms of their performance using the same metrics. After the evaluation stage, the most suitable model is selected for deployment based.

The selected model will convert to TensorFlow Lite version and be deployed into the Android app. In Android application, users must first access the application through Google sign-in or anonymous login. After that, users are required to accept the terms and condition policy before they can enter the main application. This ensures that users understand the system is only a preliminary screening tool and not a replacement for professional medical diagnosis. Then, users have the option to upload or take a photo of a skin lesion. The uploaded image is first tested by the image quality checker, followed by classification and the result will be shown on the user interface. Meanwhile, the image and result can be saved in the Cloudinary and the Firebase Firestore database for history retrieval.

3.2.2 Use Case Diagram and Description

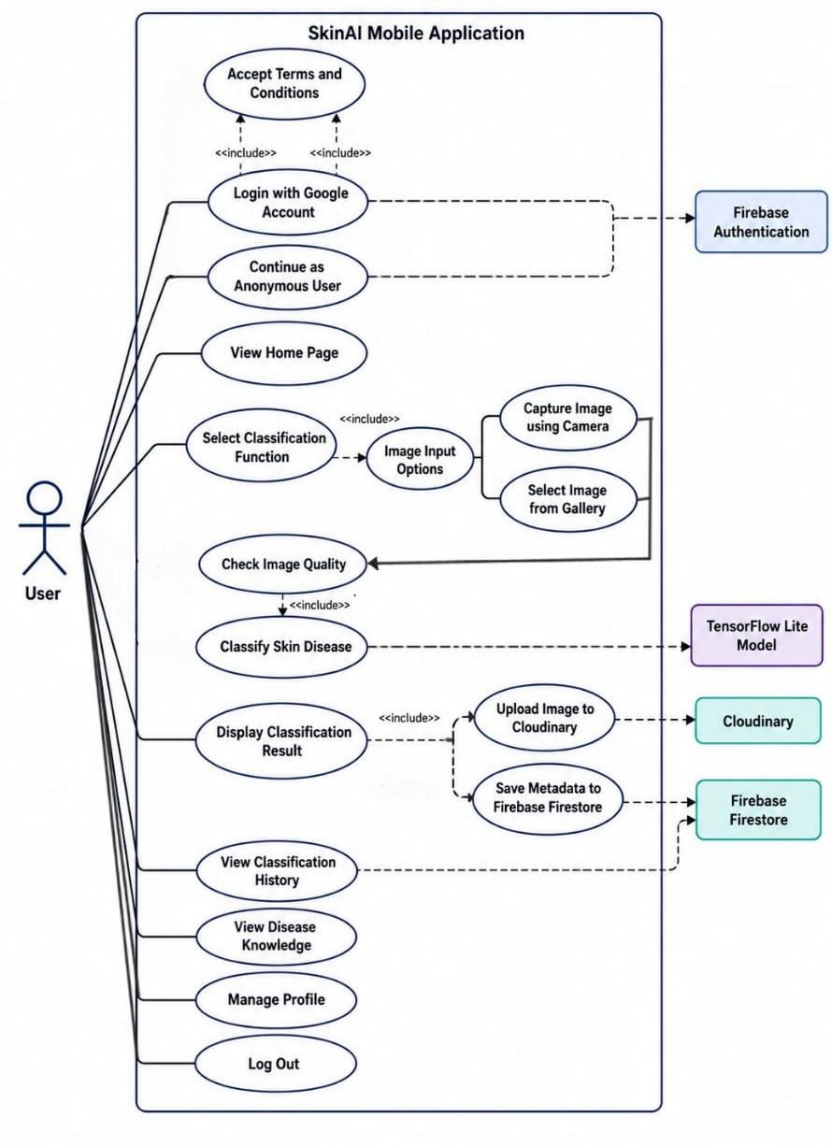


Figure 3.6 Use case diagram

The use case diagram above describes the main interactions between users and the SkinAI application. The main actor of the system is users who need to accept the terms and conditions before accessing the main functions of the application. This is necessary as SkinAI is aimed at using as an initial screening tool but not as an alternative to professional medical diagnosis.

After that, users can access the application either by logging in using a Google account or anonymously. The difference between these two login methods is that a Google account enables the scan history to be saved and sync across different devices, while anonymous login is a temporary access method that only keep records for the current session. Once users have been authenticated, they can access several main functions, including skin disease classification, viewing classification history, reading skin disease knowledge, managing their profile, and logging out from the application.

For the classification function, users can either choose an image in the gallery or take a new image with the camera to perform skin classification function. The image quality checker will then check the quality of the uploaded image to double confirm that it can be subjected to classification. For case that the image is successful in the quality checking process, it will be sent to the TensorFlow Lite classifier to make the prediction of the skin disease category. After that, the result of classification and confidence can be displayed to the user. The system also automatically stores the scan result by uploading the image to Cloudinary and saving the related metadata in Firebase Firestore.

Item	**Description**
Use Case Name	Login and Accept Terms
Actor	User
Precondition	The user has launched the SkinAI application.
Main Flow	The user opens the application. The system displays the login page. The user chooses either Google sign-in or anonymous login. After successful authentication, the system displays the terms and conditions page. The user accepts the terms and proceeds to the home page.

Alternative Flow	If authentication fails, the system remains on the login page and prompts the user to try again. If the user does not accept the terms, the system does not proceed to the home page.
Postcondition	The user is authenticated and allowed to access the main functions of the application.

Table 3.5 Use Case Description of Login and Term Acceptance

Item	**Description**
Use Case Name	Classify Skin Disease
Actor	User
Precondition	The user has logged in, accepted the terms and conditions, and entered the home page.
Main Flow	The user selects the classification function. The user captures an image using the camera or selects an image from the gallery. The system loads the image and performs image quality checking. If the image passes the checking process, the image is passed to the TensorFlow Lite classifier. The classifier returns the predicted disease class and confidence score. The result is displayed to the user.
Alternative Flow	If the image is blurry, too dark, too bright, or does not contain sufficient skin area, the system displays an error message and asks the user to provide another image.
Postcondition	The user views the classification result.

Table 3.6 Use Case Description of Classify Skin Disease

Item	**Description**
Use Case Name	View Scan History
Actor	User
Precondition	The user has logged in and has at least one saved scan result.
Main Flow	The user opens the history page. The system retrieves scan records from Firebase Firestore. The records are displayed in a list with disease result, confidence score, date, and related image.

Alternative Flow	If no scan history exists, the system displays an empty history message.
Postcondition	The user can review previous scan results.

Table 3.7 Use Case Description of View Scan History

Three use cases tables above are based on the completed Android system. They include the scenarios that show how the user interact with SkinAI, and all the alternatives' situations are covered.

In short, the use case diagram and tables summarizes the functional requirements of the completed SkinAI Android system. The main use cases include authentication, terms acceptance, image input, image quality checking, classification, result display, scan history, disease knowledge, profile management, Cloudinary image storage, Firebase Firestore record storage, and logout.

3.2.3 Activity Diagram

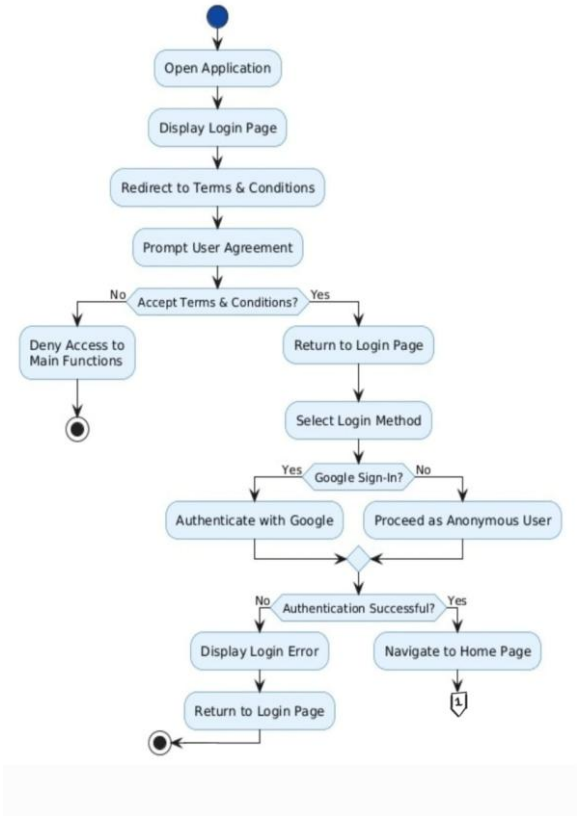


Figure 3.7 Activity Diagram Authentication

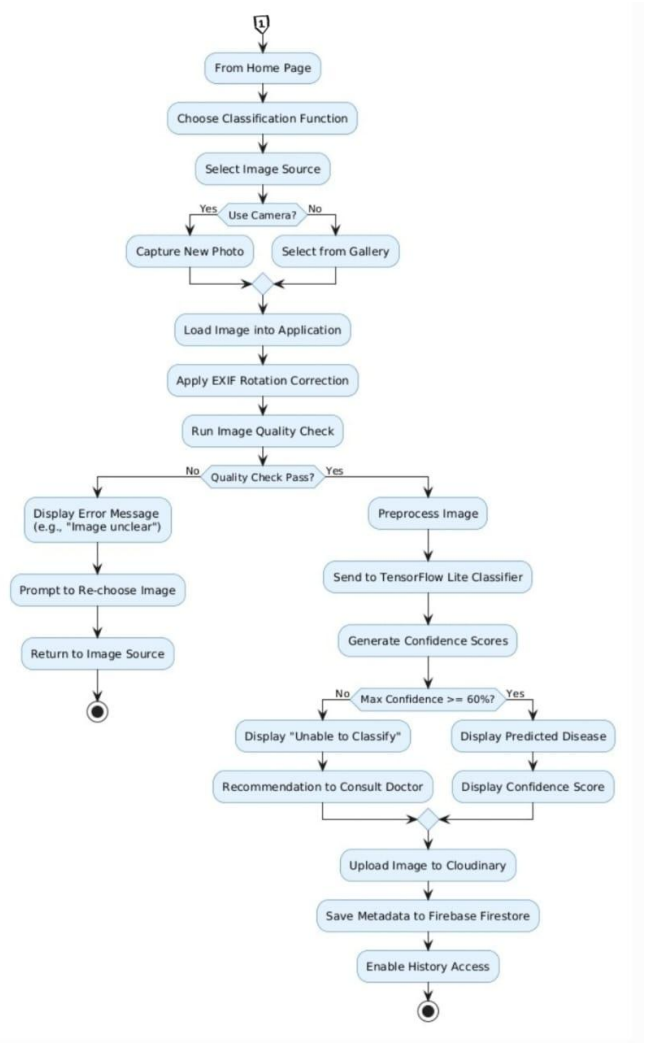


Figure 3.8 Activity Diagram Classification

The activity diagram illustrates the step-by-step workflow of the SkinAI classification process. The process starts when users open the application. Users will first enter the login page and redirect to the term and conditions page to prompt for agreement of using this application. For situation that users do not accept the terms and conditions, the application will not allow access to the main functions. In contract, once users agreed and checked on the term and condition box, they will be redirect back to login page and given options of using Google sign-in or anonymous login.

After successful authentication, users will be directed to the home page and can choose to perform classification function. Users are allowed to take a new picture with the help of the camera or choose an existing picture in the gallery. The captured or chosen image is then loaded into the application and the system will handle the image orientation by applying EXIF rotation correction.

Next, the image quality checker will determine the appropriateness of the image to be classified. The checking procedure considers the clearness of the image, the level of brightness, and the presence of a sufficient amount of skin in the image. In case of failure in the quality checking procedure by the image, the system gives users an error message and redirects users to the process of rechoosing the image. Once the image passes through a quality checking process, it is preprocessed and sent to the TensorFlow Lite classifier.

The classifier creates predicted confidence scores for each of the skin disease categories that it can classify. In case that the maximum score of confidence is less than 60%, then the system will show an "unable to classify" response to avoid presenting a low-confidence prediction. Conversely, if the confidence score is 60% or above, the system will show the predicted disease category and confidence score to users. After the result is displayed, the system will automatically upload the image to Cloudinary and save the metadata of the classification to Firebase Firestore so that the result can be accessed again later through the history function.

3.3 Verification Plan

The verification plan was drawn up to make sure that the trained deep learning model and the Android application works properly before final assessment. The verification process includes model performance, compatibility with the TensorFlow Lite, compatibility with Android applications, as well as the usability of the user interface.

To verify the models, the trained models are tested on the reserved testing data that is not accessed during training or verification. To measure model performance evaluation metrics like accuracy, precision, recall, F1-score, and confusion matrix are used. Test Time Augmentation is also used in testing to enhance the stability of prediction. The models are then contrasted in terms of classification performance, size of the models and appropriateness in mobile implementation.

In the case of TensorFlow Lite verification, after conversion to their equivalent model, it is checked to ensure that it can be loaded and executed in Android application. The input size, output classes, label order and TensorFlow Lite running compatibility is checked to ensure that the trained model and the mobile application do not conflict.

To verify Android application, the key functions of the SkinAI application are tested, which are authentication, terms acceptance, image input, image quality checking, classification, result display, cloud image upload and scan history storage. The user interface will also be inspected to enable users to navigate the application easily and realize that the system is merely a pre-screenship tool and not a substitute to professional medical diagnosis. Chapter 6 gives the detailed testing procedures, classification results, application screenshots and the performance evaluation.

The detailed testing procedures, classification results, application screenshots, and performance evaluation are presented in Chapter 6.

Chapter 4
System Design

This chapter presents the system methodology and design used to develop the AI-based mobile application for skin disease classification, SkinAI. It will describe clearly the overall system architecture, user interaction flow, activity flow and all technique used in this project.

4.1 System Block Diagram

The system block diagram is used to describe the overall structure of SkinAI by decomposing the application into various functional layers. It illustrate the relationship among the user interface, image processing, machine learning model, and the backend storage components to support the entire process of skin disease classification.

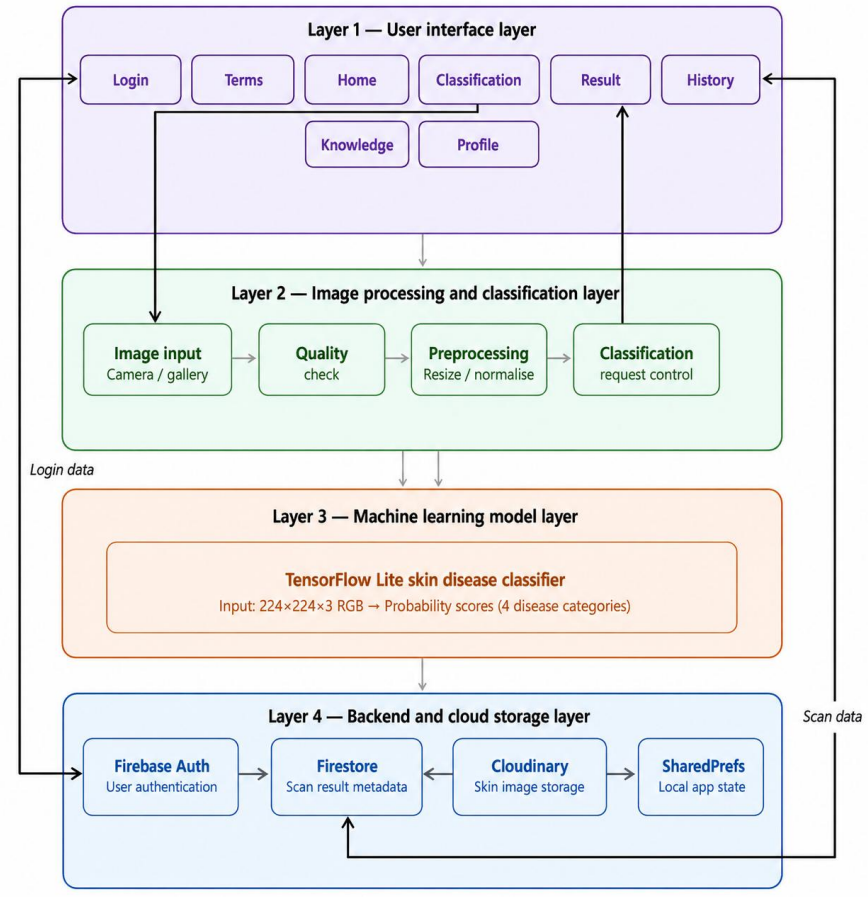


Figure 4.1 System Block Diagram

The system block diagram of SkinAI as shown above is an organization that breaks down the application into four main functional layers. The screens that users interact with are in the user interface layer. It include features like login, terms and disclaimer, home, classification, result, history, knowledge and profile screens. The task of this layer is to navigate and present information to users.

The image processing and classification layer is the intermediate processing layer between the user interface and the machine learning model. It handles image input from camera or gallery, performs image quality checking, image preprocessing and controls the classification request before passing the image to the model. Furthermore, the TensorFlow Lite skin disease classifier that is included in the machine learning model layer accepts a 224 x 224 x 3 RGB image and provides probability scores for the four supported disease categories.

The backend and cloud storage layer offer supportive services for authentication, storage of results, storage of images, and managing local preferences. In this way, Firebase Authentication is used to log a user in while Firebase Firestore is used to store metadata of scan results. Cloudinary is also used to store the uploaded skin image and SharedPreferences stores simple local application states. The block diagram demonstrates that each component has a certain role to play and the final application can be implemented, maintained and tested in a modular fashion by separating the system into these layers.

4.2 System Component Specification

This section outlines the key components that are needed to design and implement SkinAI. As SkinAI is a software-based AI mobile application, the system components primarily include software modules, model components, Android application components, and backend services.

4.2.1 Model Component Specification

The model component of SkinAI is the deep learning model that classifies images of skin diseases. Under the final system, the chosen model is implemented in the TensorFlow Lite format so that it can be executed as part of the Android application. The model part collaborates with the image preprocessing module and the SkinClassifier class to produce prediction results.

The model takes an input image in 224x224x3, RGB format. Thus, the chosen picture needs to be resized and transformed into the appropriate input format prior to inference. The model generates four probability scores which include eczema, psoriasis, basal cell carcinoma and viral infections. The application then picks the class that is most likely to be the greatest result as the forecasted outcome. A 60% confidence threshold is used to minimize the possibility of showing low-confidence predictions.

Specification	**Description**
Model Type	CNN-based transfer learning model
Deployment Format	TensorFlow Lite
Input Size	224 × 224 × 3 RGB image
Output	Probability scores for four disease classes
Class 1	Eczema
Class 2	Psoriasis
Class 3	Basal Cell Carcinoma
Class 4	Viral Infections
Confidence Threshold	60%
Android Model Location	app/src/main/assets/
TensorFlow Lite Version	2.14.0
Label File	labels.txt

Table 4.1 Model Component Specification

The table above outlines the model specifications to integrate Android. It makes sure that the input image format, the output labels, the confidence threshold and the model file location are aligned with the implementation of the classifier.

4.2.2 Android Application Component Specification

The Android application is comprised of several activities and Java classes.

Activity	Purpose
MainActivity	• Displays the home screen • Display knowledge carousel
LoginActivity	• Handles Google sign-in and anonymous login
TermsActivity	• Displays terms and medical disclaimer
ClassificationActivity	• Allows users to select an image through camera or gallery
ResultActivity	• Displays classification result and confidence score
HistoryActivity	• Displays previous scan records
ProfileActivity	• Displays user profile information
KnowledgeDetailActivity	• Displays detailed disease information

Table 4.2 Android Activities and Functions

The table above show that all activities have a designated screen or function in the application. The application will start with authentication and terms acceptance and then proceed to allow access to the main functions. Once in the home screen, users can categorize images, browse the history of past scans, read disease knowledge, and manage profile information.

Class	Purpose
SkinClassifier	• Loads the TensorFlow Lite model • Performs inference
ImageQualityChecker	• Checks blur, brightness, and skin pixel ratio
FirebaseHelper	• Handles Firestore scan record operations
CloudinaryHelper	• Handles image upload to Cloudinary
SharedPreferencesHelper	• Stores local application preferences
ScanResult	• Represents the scan result data model
KnowledgeCarouselAdapter	• Handles disease knowledge carousel display
HistoryAdapter	• Displays scan records in the history list

Table 4.3 Android Java Classes and Functions

In addition to Android activities, there are several Java classes used to support machine learning inference, image checking, cloud storage, database operations, local preferences and list display. The table above shows the importance roles that Java classes play in applications.

4.2.3 Backend and Storage Component Specification

Several backend and storage components are involved in this project to provide supporting services for authentication, scan history management, image storage, and local application state management.

Component	**Purpose**
Firebase Authentication	• Supports Google sign-in and anonymous login
Firebase Firestore	• Stores scan result metadata and history records
Cloudinary	• Stores uploaded skin images
SharedPreferences	• Stores local application preferences or simple states
Internet Permission	• Allows the application to connect to Firebase and Cloudinary

Table 4.4 Backend and Storage Components

The table above show the backend and storage modules used in SkinAI. Google sign-in and anonymous login are done using Firebase Authentication while Firebase Firestore stored scan result metadata such as predicted disease, confidence score, image URL, timestamp and user ID. The uploaded skin images are stored in Cloudinary and the resulting image URL is then stored in Firebase Firestore together with the scan Metadata. This isolates image storage from structured metadata storage and provides a more efficient history retrieval. Simple local storage like application states or preferences is achieved with the help of SharedPreferences. Internet access is necessary since Firebase and Cloudinary rely on network connectivity.

4.3 Software Components Design

This section describes the design of the main software components that support image classification, image quality checking, user interface navigation, and cloud-based history storage.

4.3.1 TensorFlow Lite Classifier Design

The TensorFlow Lite classifier is designed to allow the trained skin disease classification model to run inside the Android application. The classifier component is responsible for loading the ".tflite" model from the Android assets folder, preparing the input image, running inference, and returning the predicted disease class with its confidence score.

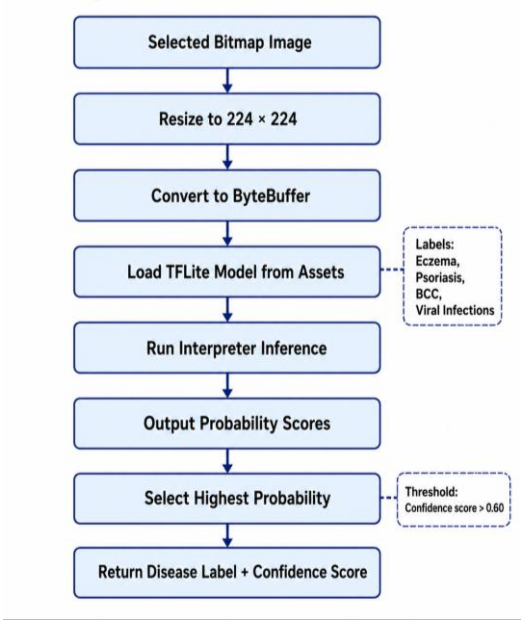


Figure 4.2 TFLite Inference Design

The figure above shows the TensorFlow Lite classifier design. When the user chooses or takes an image, the image is made into a Bitmap object by the classifier. The image is then transformed into a ByteBuffer format and fed into the TensorFlow Lite interpreter. The interpreter executes the model and returns probability scores of the four categories of diseases. The class of the disease with the highest probability is chosen as a predicted outcome and shown alongside the confidence score. This architecture enables the trained model to be directly implemented in the Android application.

4.3.2 Image Quality Checker Design

Image quality checker will be used to avoid the classification of unsuitable images. Since low quality of images can influence the reliability of model predictions, the application verifies that the selected image has a high quality before it is sent to the classifier. This will ensure minimization of chances of coming up with untrustworthy predictions based on hazy photos, dimly lit photos or photos which lack enough skin area. The checker mainly conducts four major checks.

1. Check 1: Blur Detection. This is based on Laplacian variance algorithm used to detect the sharpness of an image. A variance value of 10 and above means that the image is in focus enough to be classified.

2. Check 2: Brightness Detection. This is used to measure the average brightness of the picture. The range of values between 30 and 240 depicts that there is acceptable lighting. Darker values are not allowed to be below 30 and the brightest values are not allowed to be above 240.

3. Check 3: Suspicious Image Detection. This check analyzes the image to look for patterns that could indicate the image is a graphic, QR code or non-photographic content. This check looks at dark pixel ratio which should be lower than 40% and color variety which should have equal or more than 20 unique colors.

4. Check 4: Skin Pixel Ratio Detection. An HSV color space analysis is used to detect flesh-tone pixels. A range of at least 0.01% but less than 32 to confirm the presence of skin but not false positives due to graphics or photos containing too much skin-colored material.

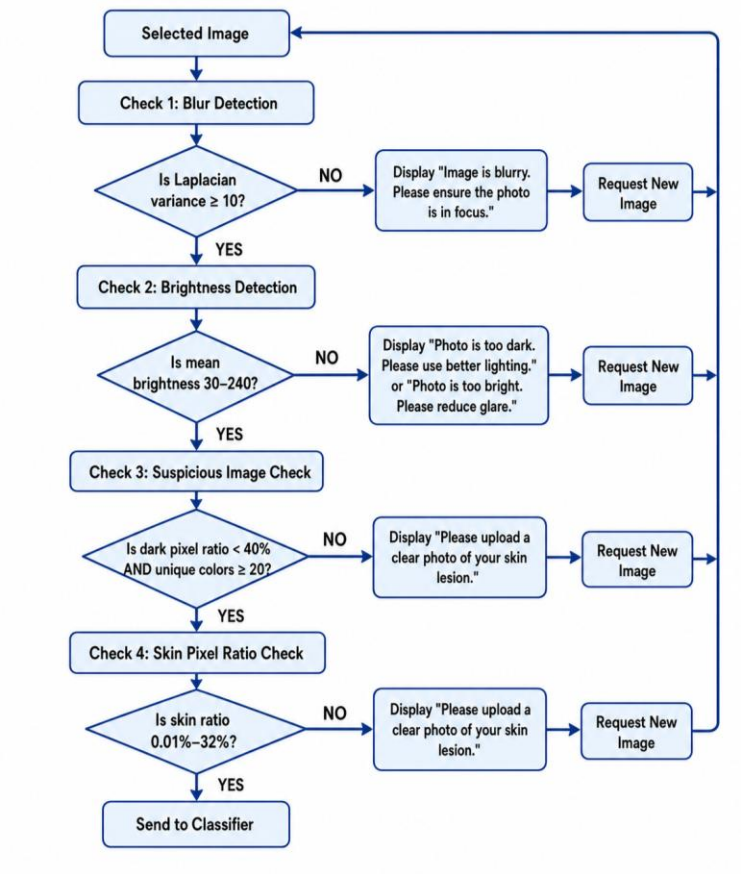


Figure 4.3 Image Quality Checking Process

The design of the image quality checker is indicated in figure above. The selected image must pass four sequential checking stages before it is classified. The process will be terminated as soon as the first failure to make sure that only quality pictures reach the classifier. In the event of failure of any checking stage the application will issue a suitable error message to prompt the user to enter another image. Upon passing all the checks, the application then goes to transmit the image to the classifier where the disease is predicted.

4.3.3 Android User Interface Design

Android user interface is meant to offer a transparent and straightforward workflow to common users. As SkinAI is supposed to be a preliminary screen, the interface should be able to direct users through the process of making a classification without necessitating technical expertise. The main screens include the login, terms and conditions, home, classification, result, history, knowledge and profile.

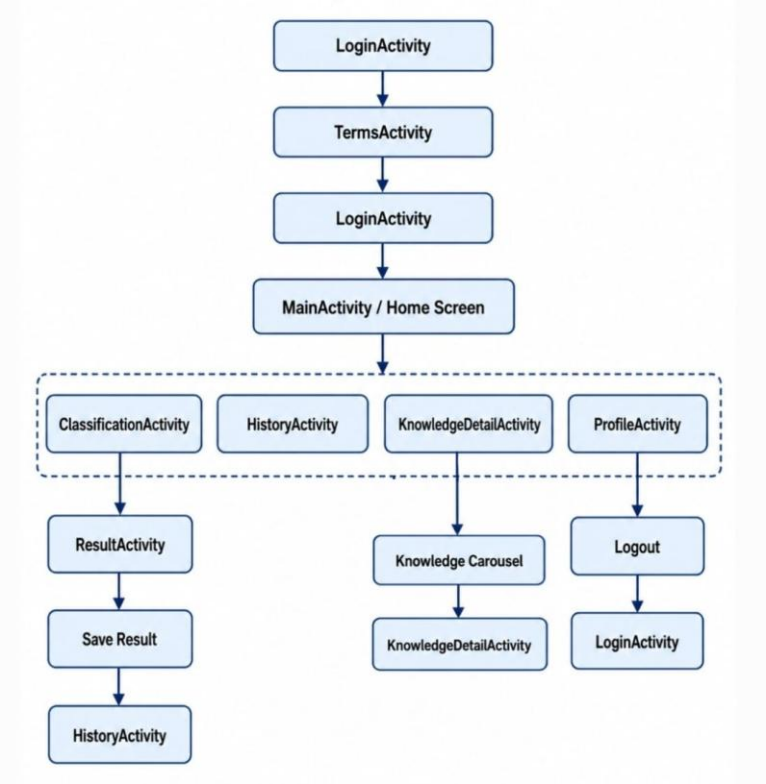


Figure 4.4 User Interface Navigation Structure

The above figure shows the Android user interface navigation design of SkinAI. The user is introduced to the application via the login screen and terms screen. On accepting the terms, the user can choose the desired login method and get redirected to the home screen. The classification, history, knowledge of diseases, and profile functions can be accessed by the user through the home screen and navigation bar. The classification functionality results in the result screen upon processing of an image by the classifier. The results saved may be later accessed via the history screen. The home screen provides users with the knowledge carousel that can lead the user to information about the disease on the knowledge detail page. This system of navigation is aimed at maintaining the application simple and comprehensible to the general users.

4.3.4 Database and Cloud Storage Design

The database and cloud storage model segregates the image storage and scan metadata storage. Cloudinary was responsible for storing uploaded skin images and Firebase Firestore store structured scan information. The design does not permit large image files to be directly stored in Firestore and enables the application itself to access scan records effectively.

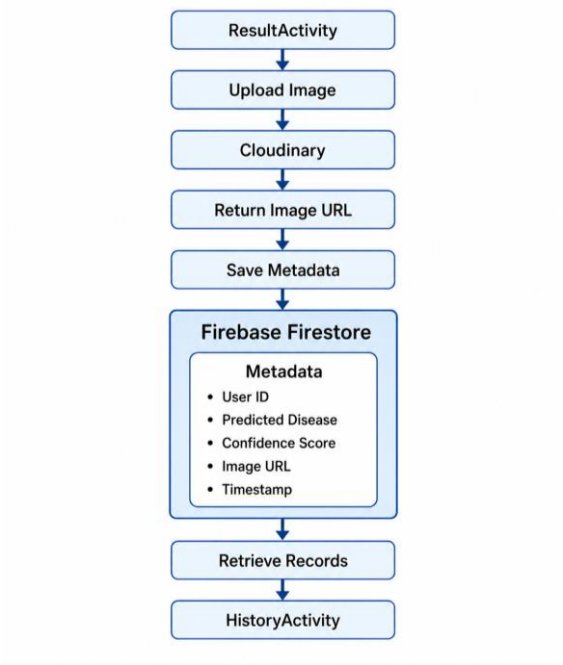


Figure 4.5 Database and Cloud Storage Design

Figure above shows the database and cloud storage design of SkinAI. By saving a scan result, the skin image is uploaded to Cloudinary. Once upload has been done, Cloudinary provides an image URL. This URL along with the predicted disease, confidence score, timestamp, and user information are stored in Firebase Firestore in the application. As the user opens the history page, the application will fetch the stored scan records on Firestore and present them in the history list. This is designed to enable image files and scan Metadata to be handled independently and effectively.

4.4 System Components Interaction Operations

This section describes the interaction of the key elements of SkinAI in the course of system operation. It highlights the key system operations that involve interaction of the Android activities, helper classes, the TensorFlow Lite classifier, Firebase, and Cloudinary.

4.4.1 Image Classification Operation

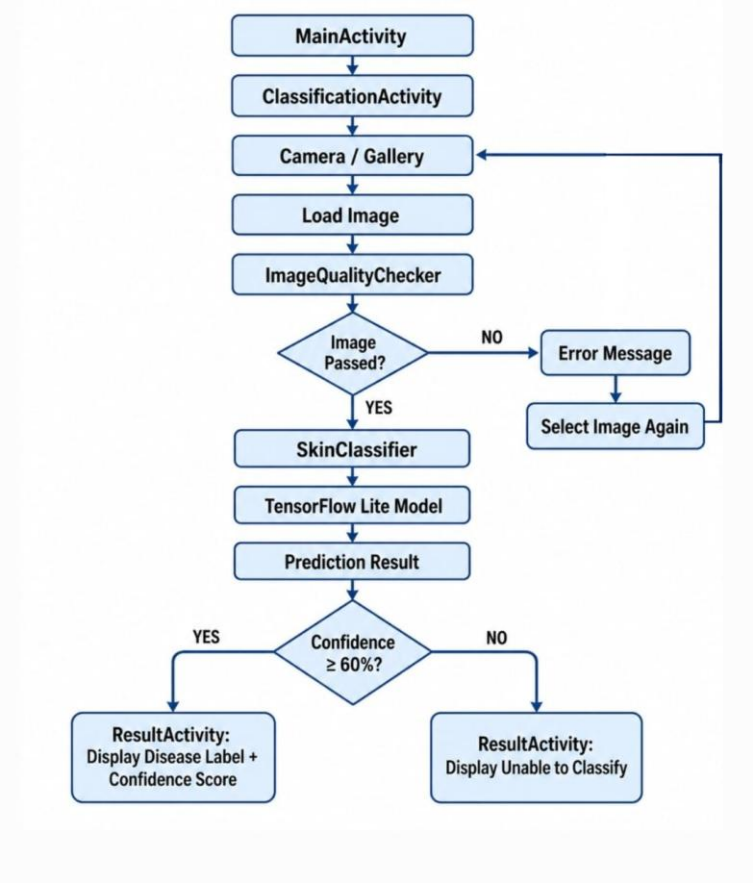


Figure 4.6 Image Classification Operation

The figure above represents the process of the image classification in SkinAI. The process starts with the home screen and requires the user to supply an image of their skin using the camera or gallery. The selected image is first checked by the image quality checker. In case the image is inappropriate, the application will deliver an error and request the user to supply another image. In case the image passes the quality checking process, the image is sent to SkinClassifier component. The TensorFlow Lite model then infers and gives out the probability score of the four disease categories. The system will pick the most probable class and will then use a 60 percent confidence margin before the system will be used to show the final result. When the maximum confidence score is below 60%, the system shows an “unable-to-classify” result. In case confidence score is 60% or higher, the predicted disease label and confidence score are sent back to the result screen and shown to the user. This algorithm makes sure that only high-quality images are forwarded to the classifier and that images with low confidence are not presented as valid classification outcomes.

4.4.2 Result Storage and History Retrieval Operation

Result storage and history retrieval operation explain how the results of the classification are stored and subsequently collected to display to the user.

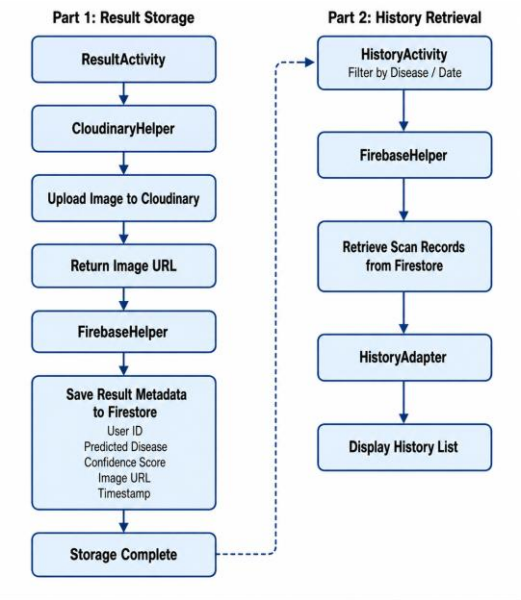


Figure 4.7 Result Storage and History Retrieval Operation

The result storage and history retrieval operation is presented in the above figure. Once the image has been classified, the chosen image is uploaded to Cloudinary via CloudinaryHelper. After a successful upload, Cloudinary will give an image URL. The application then provides FirebaseHelper to store the result metadata into Firebase Firestore. Upon the user visiting the history page, the application reads the saved records in Firestore and displays them using the history adapter. The history records might contain the expected disease, a level of confidence, date, and image. This operation enables users to view past classification outcome and have a record of past scans.

Chapter 5
System Implementation

This chapter presents the implementation of SkinAI which show how the proposed design was implemented into a working Android application. The implementation covers the development environment setup, model implementation, TensorFlow Lite conversion, Android model integration, Android application implementation, system operation screenshots, and implementation issues encountered during development.

5.1 Development Environment Setup

The development environment setup includes the hardware and software tools used to develop, train, convert, integrate, and test the SkinAI system.

5.1.1 Hardware Setup

The hardware involved in this project is a development laptop and an Android device.

Description	**Specifications**
Laptop Model	Lenovo IdeaPad 5
Processor	AMD Ryzen 7 5700U with Radeon Graphics
Operating System	Windows 11
Graphic	AMD Radeon™ Graphic
Memory	8GB
Storage	477GB
Android Testing Device Model	Samsung Galaxy S23 Ultra

Table 5.1 Hardware setup

Report writing, management of project files, Android Studio development, integration of TensorFlow Lite, and system testing were done using the development laptop. Even though

training the deep learning model was primarily done using Google Colab, the development laptop was still needed to work with project files and develop the Android application.

The finished mobile application was tested with the help of Android testing device. The testing procedure was to check image selection, TensorFlow Lite model loading, displaying classification results, Firebase connection, Cloudinary upload and retrieving scan history.

5.1.2 Software Setup

The software involved in this project include a combination of programming languages, machine learning framework and mobile developments tools.

Description	**Specifications**
Operating System	Windows 11
Model Training Platform	Google Colab
Mobile Development Tool	Android Studio
Programming Language/Environment	• Python (AI model training) • Java 11 (Mobile application development)
Frameworks / Libraries	• TensorFlow/Keras • TensorFlow Lite 2.14.0 • Glide 4.16.0
Backend Service	• Firebase Authentication • Firebase Firestore • Cloudinary
Build System	Gradle

Table 5.2 Software specifications

The table above provides the list of the software technologies involved in the creation of SkinAI. Windows 11 was the operating system of choice to be developed with, and Google Colab was used as the model training environment because it supports cloud-based deep

learning experimentation. The dataset processing, model training and evaluation were done using Python. The CNN models were developed and trained using TensorFlow and Keras, then deployed to run on-device using the Android application and TensorFlow Lite 2.14.0.

For the mobile application, Android Studio was used as the main development environment and Java 11 was used to implement application logic such as user interface functions. Firebase Authentication was employed to support Google login and anonymous login method while Firebase Firestore was employed to store scan history and related result Metadata. The cloud storage was done using Cloudinary whereby skin images uploaded were stored. Images were loaded and displayed in the Android application with Glide 4.16.0. Lastly, Gradle will be used to manage project dependencies and the Android build process.

5.2 Model Implementation

Implementation of the model included training the candidate CNN models, preparing the chosen model to be converted to a TensorFlow Lite model and integrating the converted model into the Android application. The detailed model testing results and final model selection justification are presented in Chapter 6. Thus, the section is concerned with only the implementation steps involved in preparing and deploying the model in the Android application.

5.2.1 CNN Model Training Implementation

CNN architecture like MobileNetV2, ResNet50, and EfficientNetB2 were implemented and trained to compare their results. MobileNetV2 was chosen as the lightweight baseline model since it is applicable in mobile and embedded applications. ResNet50 was used as a more in depth comparison model owing to its high feature extraction ability. EfficientNetB2 was adopted as the finalized EfficientNet candidate since it offers more capable architecture yet is still efficient. In this way, all the models were introduced with the pretrained ImageNet weights and linked to a custom classification head to four-class skin disease classification. The 4 classes of output were eczema, psoriasis, basal cell carcinoma and viral infections.

To have a fair comparison of the models, they were implemented using the identical general training pipeline. The images were scaled to 224 224 pixels, transformed into RGB, preprocessed by the respective model specific preprocessing function, and trained using

transfer learning. All candidate models were taken through the same three-stage training strategy as was used in Chapter 3.

After training and evaluation, one of the models was selected and ready to be deployed on Android. Chapter 6 shows the detailed comparison between MobileNetV2, ResNet50, and EfficientNetB2.

5.2.2 TensorFlow Lite Conversion Implementation

The chosen CNN model was then converted into TensorFlow Lite format to ensure that the model could be run as a part of the Android application. The transition to TensorFlow Lite was required as the model files of the standard Keras model cannot be directly executed in an Android application.

Two conversion procedures have been tried in implementation. The first method made direct Keras model export and optimization with TensorFlow Lite. This approach created a smaller model file of about 8.5 MB. The converted model however failed in Android run time due to incompatibility of unknown "FULLY_CONNECTED" opcode version. Hence this conversion technique was not deployed to a final deployment.

The second method involved SavedModel to TensorFlow Lite conversion. In this approach, the trained model was initially saved as a SavedModel file then converted to TensorFlow Lite format using supported TensorFlow Lite built-in operations. This method was able to be used with TensorFlow Lite 2.14.0 and run in the Android application. Therefore, the SavedModel to TensorFlow Lite conversion method was used for the deployed model.

Conversion Method	**Model Size**	**Compatibility**	**Result**
Direct Keras + Optimize.DEFAULT	~ 8.5 MB	Not compatible with TensorFlow Lite 2.14.0	Failed
SavedModel to TensorFlow Lite	30.8 MB	Compatible with TensorFlow Lite 2.14.0	Working

Table 5.3 TensorFlow Lite Conversion Comparison

The TensorFlow Lite model file deployed in the Android application was "efficientnetb2_skin_disease.tflite". The model takes as input 224 × 224 × 3 RGB image input and the output as probability scores of four disease classes. Though the deployed model file is indicated here to enable a clear picture on the implementation, the performance results and reasons why this model was selected are discussed in Chapter 6.

5.2.3 Android Model Integration

The converted TensorFlow Lite model file which is represented by the file named as ".tflite" model file was placed in the Android project assets folder. A labels.txt file was also included that contained the names of the classes in the appropriate order of output. The four labels in the application were eczema, psoriasis, basal cell carcinoma and viral infections.

Android application loads the TensorFlow Lite model and uses SkinClassifier class to make an inference. When an image is chosen or taken, it is resized to 224 x 224 pixels and converted to the desired input format which then passed through the TensorFlow Lite interpreter. The scores on output probability are computed by the interpreter with the aim of determining the disease class with the highest confidence score.

Component	**Implementation**
Model File	efficientnetb2_skin_disease.tflite
Model Location	app/src/main/assets/
Label File	labels.txt
Classifier Class	SkinClassifier
Input Size	224 × 224 × 3 RGB
Output	Four disease probability scores
Confidence Threshold	60%
TensorFlow Lite Version	2.14.0

Table 5.4 Android Model Integration

5.3 Android Application Implementation

Android Studio was used to implement the Android application in Java. This implementation was in accordance with the system design as given in Chapter 4. The application gives an end-to-end user authentication flow to image classification and scan history.

5.3.1 Authentication and Terms Implementation

The implementation of authentication and terms acceptance was done with the combination of LoginActivity, TermsActivity, Firebase Authentication and SharedPreferences helper. On startup of the application, the system examines whether there is already an authentication session. In case of no active authentication session, users will be sent to the login screen.

Two methods of login were incorporated in the LoginActivity which includes Google sign-in and anonymous logins. But both the login buttons are inactive once the login screen was initially shown. The user should agree to the terms and conditions by clicking the checkbox offered on the home login screen. The Terms and conditions details are shown in TermsActivity where users can read the application terms first before accepting it.

Once the terms and conditions checkbox has been checked, the Google sign-in and anonymous login buttons will be enabled. The user has the option of either method of logging in. Google sign-in enables users to access the application with the help of a Google account and scan history will be syn across different devices. Anonymous login enables users to use the application without creating a full account, but the scan history will only be recorded on that current session. The two authentication methods are also managed by Firebase Authentication.

After the successful authentication, the user will be redirected to the MainActivity that shows the main application page. This application will make sure that the users are fully aware of the terms and the disclaimer before being permissible to log in and use the main application functions. This is significant as SkinAI is intended as a preliminary screening device and not as a substitute to professional medical diagnosis.

5.3.2 Image Input Implementation

The input in ClassificationActivity was implemented as image input. The app offers two image input methods which is camera capture and gallery pick. To acquire camera input, the application relies upon the native camera intent to take a new picture. For Gallery input, the application enables the user to pick an existing image in the device gallery. Image capture guidelines are also shown in the classification page to guide the user on the way to give an appropriate skin image, such as using clear lighting, avoiding blurry images and ensuring that the area of the skin is in view. This assists in enhancing quality and accuracy of the uploaded picture prior to classification.

Once the user chooses or takes a picture, the picture URI is loaded into the application. The picture is then transformed into a bitmap to be further processed. EXIF rotation processing was introduced to fix image orientation prior to checking or classifying the image. This action is significant since the pictures taken with a mobile camera could have rotation metadata, and improper rotation can have an impact on the image quality check and classification.

The loaded image is shown in the classification screen to allow the user to verify the image selected before proceeding. That is followed by the chosen bitmap being submitted to the image quality checking process. When the image is accepted, the procedure of classification moves on to the result stage.

5.3.3 Image Quality Checker Implementation

The image quality checker was set up in the ImageQualityChecker class. It was triggered after the user selected or captured an image in ClassificationActivity. Before sending the image to the classifier, the app checked the selected bitmap to see if it was suitable for classification.

The implementation checks five conditions that are blur level, brightness level, dark pixel ratio, unique color variation, and skin pixel ratio. It rejected unclear images based on blur and it turned down images that were too dark or too bright based on brightness. The dark pixel ratio check helped filter out images with too much dark background. Unique color checks identified graphic-like or non-photo images, while skin pixel ratio checks ensured that enough skin-like area was present in the image.

If the image fails any of the quality checks, ClassificationActivity displayed an error message. It asked the user to select or capture another image. Only images that passed the quality checks moved on to the result and classification stage.

Check	Threshold	Error Message
Blur Detection	Laplacian variance ≥ 10	"Image is blurry. Please ensure the photo is in focus."
Brightness Detection	30 ≤ mean brightness ≤ 240	"Photo is too dark. Please use better lighting." / "Photo is too bright. Please reduce glare."
Suspicious Image Check	Dark pixel ratio < 40% AND unique colors ≥ 20	"Please upload a clear photo of your skin lesion."
Skin Pixel Ratio	0.01% ≤ skin ratio ≤ 32%	"Please upload a clear photo of your skin lesion."

Table 5.5 Image Quality Checking Implementation

This implementation helped reduce unsuitable image inputs before TensorFlow Lite inference was performed.

5.3.4 Classification Result Implementation

The classification result function worked through the interaction among ClassificationActivity, ResultActivity, and SkinClassifier. Once an image went through the quality checking process, it was sent to ResultActivity. The SkinClassifier class loaded the TensorFlow Lite model from the assets folder and prepared the image for inference.

During this step, the image was resized to 224 × 224 pixels and converted into the input format needed by the TensorFlow Lite interpreter. The interpreter produced four probability scores for eczema, psoriasis, basal cell carcinoma, and viral infections. The application chose the class with the highest score as the predicted category.

A 60% confidence threshold was used before showing the final result. If the highest confidence score was below 60%, the result screen displayed an unable-to-classify message. If

the confidence score was 60% or higher, the result screen showed the predicted disease label and confidence score. This approach helped avoid displaying low-confidence predictions as reliable classification results.

5.3.5 History and Storage Implementation

The history and storage function was built using CloudinaryHelper, FirebaseHelper, ScanResult, HistoryActivity, and HistoryAdapter. After generating the classification result, the selected image was uploaded to Cloudinary. Once the upload succeeded, Cloudinary provided an image URL

Next, the image URL was stored in Firebase Firestore along with the scan metadata. The stored metadata included the predicted disease, confidence score, timestamp, image URL, and user ID. The ScanResult model class represented each scan record in the application.

When the user accessed the history screen, HistoryActivity retrieved the saved scan records from Firestore. It then passed the data to HistoryAdapter, which displayed the scan records in a list. Each historical item showed the stored result information and the corresponding image. This setup allowed users to save and review their previous classification results.

5.3.6 Disease Knowledge and Profile Implementation

The disease knowledge function was built using MainActivity, KnowledgeCarouselAdapter, and KnowledgeDetailActivity. In MainActivity, a knowledge carousel shows the four disease categories: eczema, psoriasis, basal cell carcinoma, and viral infections. Users can browse disease cards from the home screen.

When a user selects a disease card, the application opens KnowledgeDetailActivity. This activity provides general information about the chosen disease category. This function aims to raise basic awareness about the supported skin disease classes. Besides, additional disclaimer content will be shown to indicate that this is for general educational purposes and should not be considered professional medical advice.

The profile function was developed using ProfileActivity. This activity displays user-related information and account options. For users who sign in with Google, the profile page

shows account-related details. For anonymous users, the profile page indicates that they are using anonymous access. Users can also log out from the profile screen, allowing them to exit the current session and return to the login process. This implementation supports additional application functions beyond classification and improves the overall usability of SkinAI.

5.4 System Operation

This section presents the operation of the implemented SkinAI application using screenshots. The screenshots are included to demonstrate the completed user interface and show how users interact with the main functions of the application.

5.4.1 Login and Terms Screens

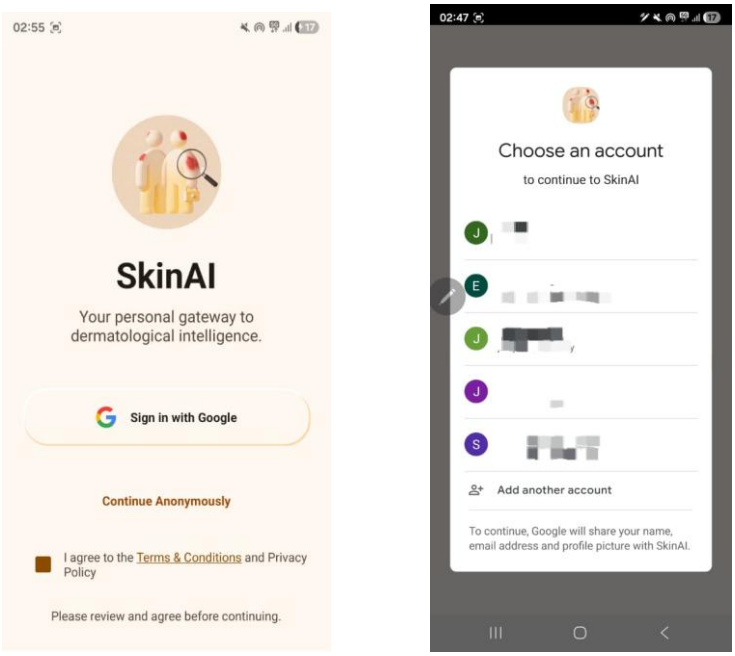


Figure 5.1 Login Pages

Figure 5.1 shows the login screen of SkinAI. This screen is shown when the users start using the application without an active authentication session. The login screen has two authentication methods, which include Google sign-in and anonymous login. The login buttons are not enabled until the user checks the terms and conditions check box. This is in order to make sure that the users are aware of the application disclaimer before getting into the main application.

Figure 5.2 Terms and Condition Page

Figure 5.2 shows the terms and conditions screen. This screen shows the application disclaimer and informs the users that SkinAI is not meant to be a diagnostic tool. Users are notified that the result of using the application should not be discussed as a professional medical diagnosis. Once the user has read and accepted the terms, he can go back to the login screen and continue with authentication.

5.4.2 Home Screen

Figure 5.3 Home Page

Figure 5.3 shows the home screen of SkinAI after successful authentication. The main application functions such as skin disease classification, scan history, disease knowledge, and profile management are provided in the home screen. It also has a disease knowledge carousel to show information cards of the supported types of the skin disease. Such a design enables users to be able to access the major functions of the application from a single central screen.

5.4.3 Image Selection Screen

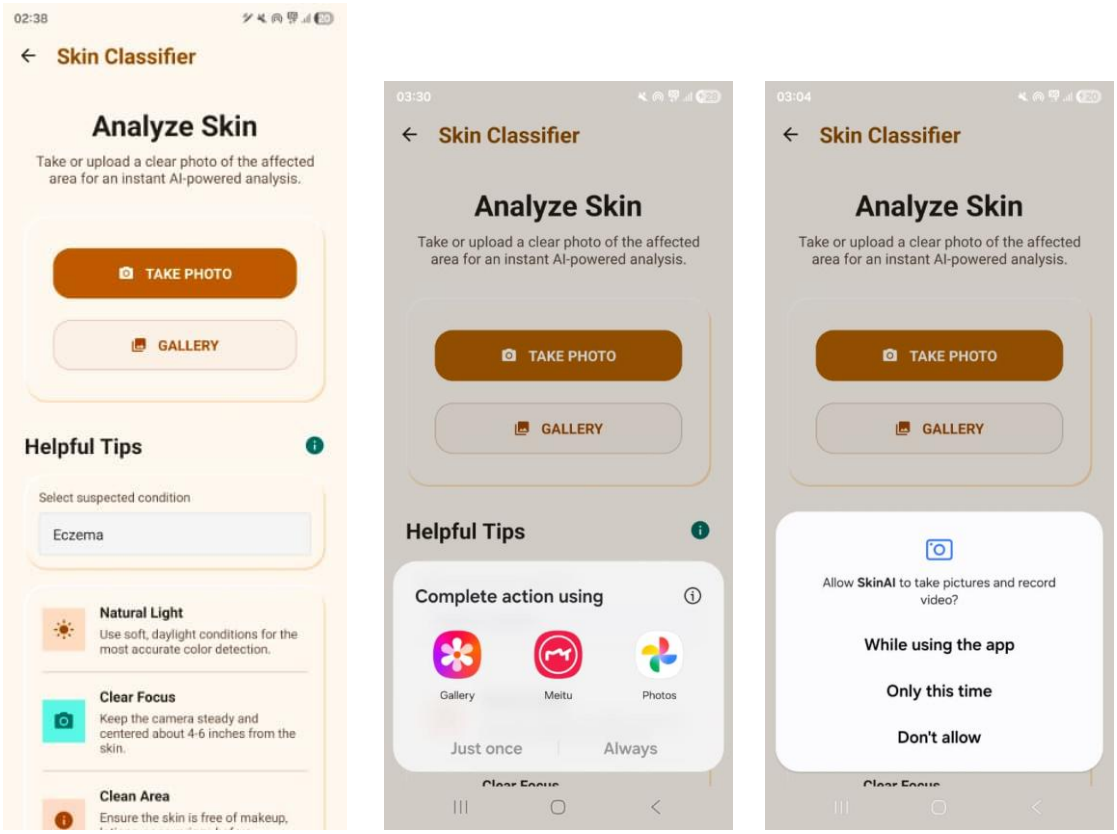


Figure 5.4 Image Selection Pages

Figure 5.4 shows the image selection screen used for skin disease classification. The user can take a new photo with the help of the camera or choose an old photo in the gallery. Image capture instructions are also available in the screen so that user can submit an appropriate image such as one that is well lit, not blurry and the area of skin is visible. These instructions aid in enhancing the quality of the image uploaded prior to classification.

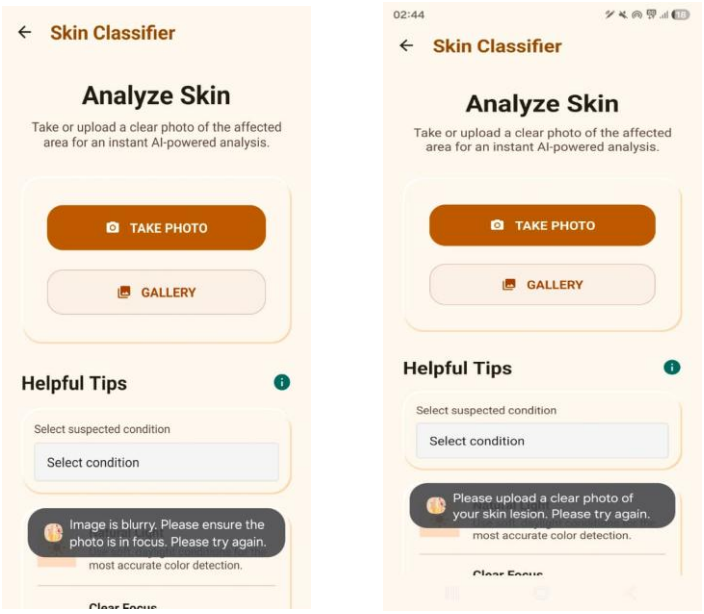


Figure 5.5 Image Quality Errors

Figure 5.5 shows examples of the image quality error message. In case the image chosen fails in the image quality checking process the application will display an error message and will not transmit the image to the classifier. The error can be either due to blur, poor lighting, excessive dark background, invalid graphic-like image or inadequate area of the skin. This is followed by requesting the user to give another image which is more appropriate to be classified.

5.4.4 Classification Result Screen

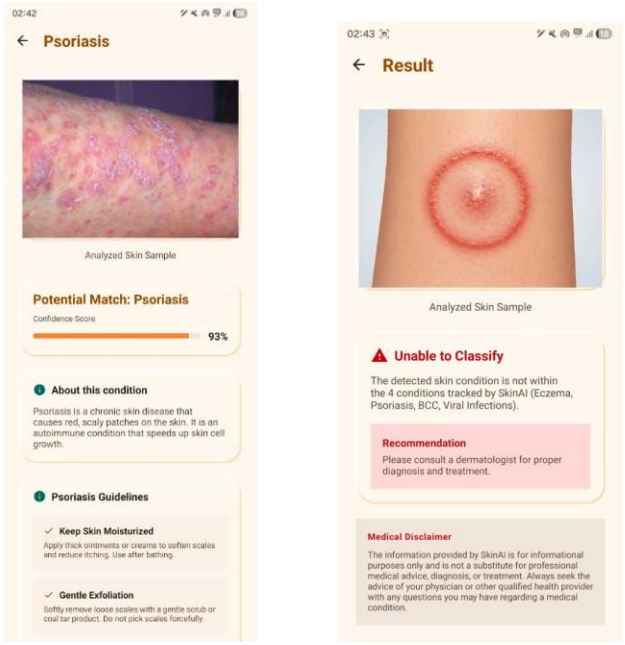


Figure 5.6 Classification Results Pages

Figure 5.6 shows the classification result screen. Once the chosen picture has gone through quality check and has been represented by the TensorFlow Lite classifier, the result screen will display the predicted skin disease type and confidence score. When the confidence score falls below the threshold, the application will not show a low-confidence prediction but will instead show an unable-to-classify result. In addition, the pertinent guidelines and management recommendations will be displayed based on the skin disease under classification. This assists the user to understand what should be done in case they have not thought of consulting a doctor. The result screen also reminds users that the prediction is of initial screening and should not be considered a medical diagnosis that is confirmed.

5.4.5 History Screen

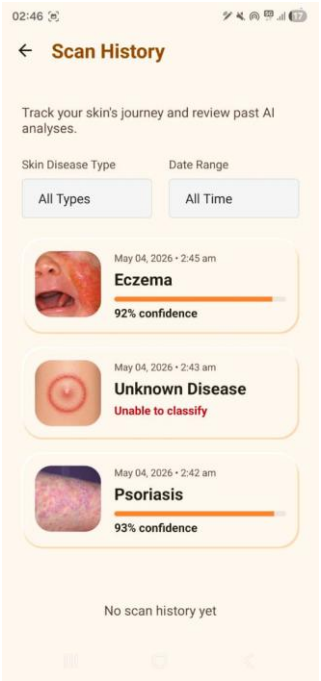


Figure 5.7 Scan History Page

Figure 5.7 shows the scan history screen. This screen shows the history of scan records that the user has saved. The predicted disease, score of confidence, scan date, and image related to the scan may be included in each record. The history feature enables users to view past classification results and track their past scan information. It can also be used to filter the desired history result that a user wants to view, by using the searching bar.

5.4.6 Knowledge and Profile Screens

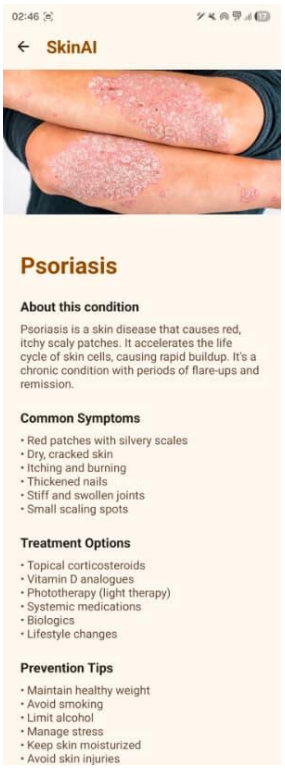


Figure 5.8 Disease Knowledge Pages

Figure 5.8 shows the disease knowledge screen. There is general information in this screen about the chosen category of skin disease. The knowledge function assists users to comprehend the four types of supported diseases that include eczema, psoriasis, basal cell carcinoma, and viral infections. The information is availed as awareness and educational assistance..

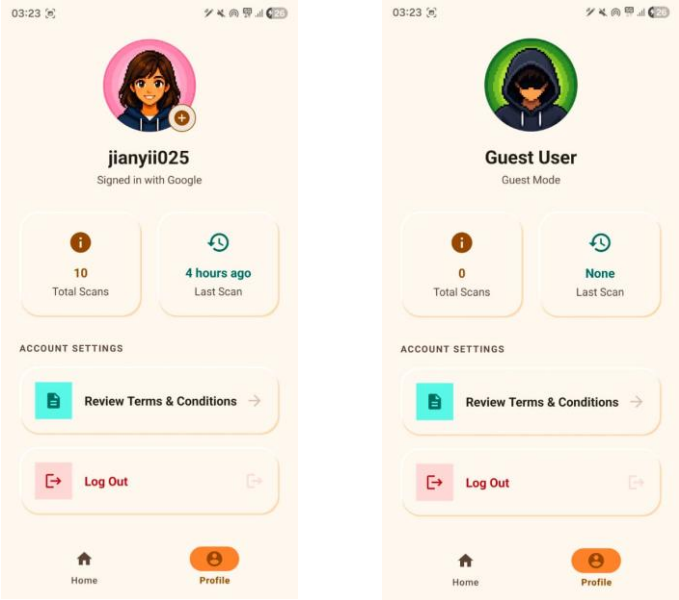


Figure 5.9 Profile Pages

Figure 5.9 shows the profile screen. This is the user-related information and account options that are shown in this screen. Users can recognize the type of operation they have either Google login or anonymous login. The profile screen can also be used to logout, so that the user can end the ongoing session and go back to the flow of logins.

5.5 Implementation Issues and Challenges

5.5.1 TensorFlow Lite Conversion Compatibility Issue

A key problem was experienced during the process of converting the model into TensorFlow Lite format. The approach of directly converting the Keras model while optimizing it for TensorFlow Lite resulted in a smaller model of about 8.5 MB in size. Nevertheless, it did not work on the Android platform since it utilized an unsupported FULLY_CONNECTED opcode version.

The problem was solved using the other approach of converting the model from the SavedModel format to TensorFlow Lite. Even though it resulted in a bigger model file of about 30.8 MB, it was supported by TensorFlow Lite 2.14.0 and could run properly within the Android app. This shows that mobile deployment requires not only a small model size but also runtime compatibility.

5.5.2 Image Quality Handling

Quality of the images was another aspect to be dealt with since the images that the users will upload could be of various qualities. For example, differences lighting conditions, angle of capturing the photos, background, skin visibility or even non-photo content. This could compromise the accuracy of classifications.

Thus, approach that the system evaluates image blurness, image brightness, percentage of dark pixels in the image, uniqueness of colors, and skin pixel percentage prior to passing on the image to the classifier was carried out. When unsuitable, the system gives the user an error message prompting the user to select another image.

5.5.3 Classification Result Implementation

Firebase and Cloudinary integration required the application to communicate with external cloud services. Internet permission was needed for authentication, database storage, and image upload. The application also had to ensure that the image upload to Cloudinary finished before saving the image URL in Firestore. Otherwise, the scanned information will not be stored properly if one of the processes fails.

To manage this process, CloudinaryHelper uploaded the image and returned the image in URL format. Meanwhile, FirebaseHelper saved the scan metadata into Firestore. This separation made the storage process clearer and easier to maintain. Moreover, proper Firebase and Cloudinary configuration was also required to prevent connection or storage errors during result saving.

5.5.4 Model Size and Mobile Deployment Trade-Off

Model size is an important factor because SkinAI is designed for Android deployment. A smaller model is usually better for mobile applications since it needs less storage and memory. However, the smaller optimized TensorFlow Lite conversion failed due to compatibility issues.

The final deployed model was larger because the compatible SavedModel conversion method was used. Although the file size increased, the model functioned properly in the

Android application. The larger model guarantees compatibility and inference time stayed under two seconds on mid-range devices at the same time.

5.6 Concluding Remark

In this chapter, the application of the SkinAI system was introduced. The development environment setup outlined the hardware and software tools that were used in the project. All the importance information such as the implementation of the candidate CNN models, the conversion of the selected model to the form of TensorFlow Lite and the integration of the model into the Android application were included in the model implementation section.

The Android application implementation involved authentication and terms acceptance, taking of images by the camera and gallery, checking of image quality to ensure only the appropriate images are classified, display of a classification result with confidence scores, history and storage features using Firebase and Cloudinary, disease knowledge section with detailed information about the four skin conditions and profile management functions. The system operation portion showed the necessary screen shots to show the finished application workflow.

The issues and challenges in implementation were discussed, including compatibility with TensorFlow Lite conversion, Android preprocessing consistency, handling image quality, Firebase and Cloudinary integration, and model size trade-offs. Like any application based on machine learning, the system has limitations, which are addressed in the following chapter.

In summary, SkinAI application has been successfully deployed with all the intended features operational and ready to be tested. The second chapter introduces the system evaluation and discussion which includes model testing results, final model comparison, Android application testing, result discussion, system limitations and objectives evaluation.

Chapter 6
System Evaluation and Discussion

This chapter introduces the assessment and the discussion of SkinAI system. The assessment will concentrate on two key components, which are deep learning model assessment and Android applications test. This chapter is meant to find out whether or not the developed system meets the project objectives.

6.1 Evaluation Setup and Metrics

In this section, the evaluation setup will be described to evaluate the trained models and the developed Android application. The evaluation focuses on two main areas: model performance evaluation and Android application testing.

6.1.1 Model Evaluation Metrics

In this chapter, the model evaluation makes use of the evaluation metrics presented in Chapter 3. These metrics include accuracy, precision, recall, F1-score, confusion matrix, model size, and inference time. The overall correctness of the model was measured using accuracy, whereas the performance of the model at the class level was measured by use of precision, recall, and F1-score. Patterns of misclassification of the four disease classes were identified using the confusion matrix.

In addition to the performance of classification, model size and inference time were also taken into consideration since the chosen model needs to be appropriate to run on Android. A model that is highly accurate but lacks mobile suitability may not be viable to a mobile application. Hence, the last model choice was based on the predictive performance and its portability.

6.1.2 Android Application Testing Criteria

Android application testing was carried out to ensure that the application features implemented worked as expected. The test included the primary workflow starting with the login process up to the classification and scan history retrieval. As the target users could consist of the general users and non-technical users, the application must also consider providing clear navigation, readable messages and appropriate error handling.

Testing Area	**Purpose**
1. Authentication Testing	Verifies Google login and anonymous login
2. Terms Acceptance Testing	Ensures users accept the disclaimer before using the application
3. Image Input Testing	Verifies camera and gallery image selection
4. Image Quality Testing	Verifies blur, brightness, dark background, unique colour, and skin area checking
5. Classification Testing	Verifies TensorFlow Lite prediction output
6. Result Display Testing	Verifies disease label and confidence score display
7. Cloud Storage Testing	Verifies image upload to Cloudinary
8. Database Testing	Verifies scan result saving and retrieval in Firebase Firestore
9. User Interface Testing	Verifies navigation, readability, and usability of application screens

Table 6.1 Android Application Testing Criteria

6.2 Model Testing Results

6.2.1 MobileNetV2 Result

MobileNetV2 was trained and evaluated as the lightweight baseline model. The final MobileNetV2 model achieved a Test Time Augmentation accuracy of 85.48% and a macro F1-score of 0.86. After SavedModel to TensorFlow Lite conversion, the model size was 9.83 MB, making it the smallest deployed TensorFlow Lite model among the three evaluated models.

	Precision	*Recall*	*F1-score*	*Support*
Eczema	0.80	0.81	0.80	251
Psoriasis	0.78	0.83	0.81	309
Basal Cell Carcinoma	1.00	1.00	1.00	240
Viral Infections	0.87	0.80	0.83	316
Accuracy			0.85	1116
Macro avg	0.86	0.86	0.86	1116
Weighted avg	0.86	0.85	0.86	1116

Table 6.2 Classification report of MobileNetV2

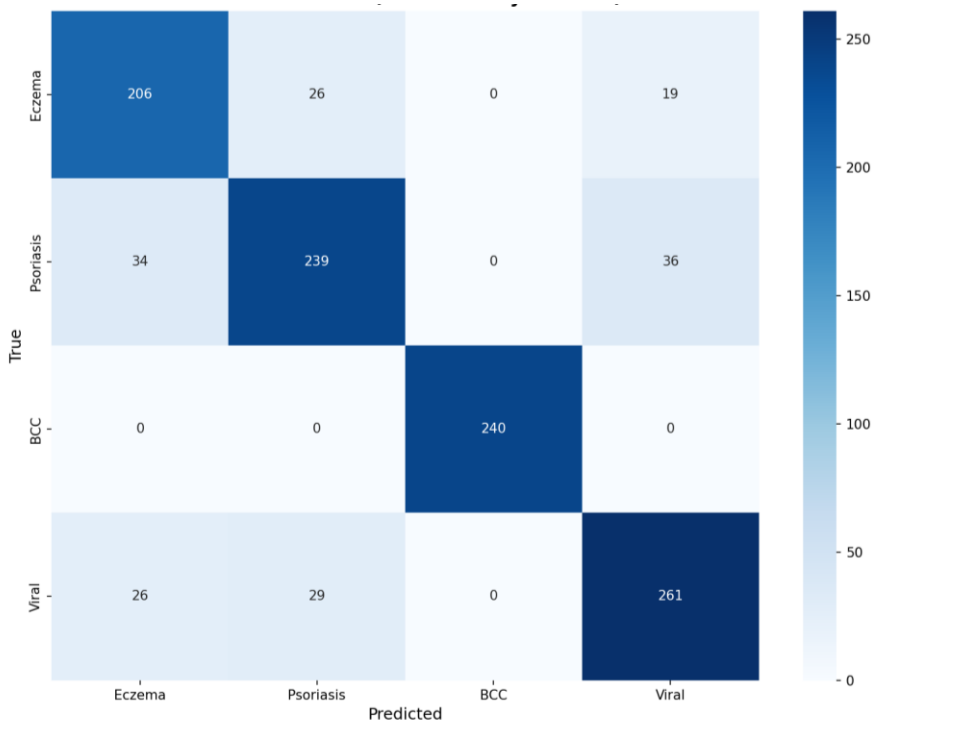


Figure 6.1 Confusion Matrix of MobileNetV2

Figure 6.1 shows the confusion matrix of MobileNetV2. The model was able to classify the 206 eczema images, 239 psoriasis images, 240 basal cell carcinoma images and 261 viral

infection images correctly. The best result was obtained with basal cell carcinoma as all 240 testing images were properly identified. This indicates that MobileNetV2 could tell basal cell carcinoma apart distinctly when compared to the other classes.

Nevertheless, the confusion table also reveals that MobileNetV2 was more confused with the eczema, psoriasis, and viral infections. Indicatively, some eczema images were wrongly classified as either psoriasis or viral infections, or some psoriasis images were wrongly classified as either eczema or viral infections. This is in agreement with the classification report where eczema and psoriasis obtained lower F1-scores than basal cell carcinoma.

Overall, MobileNetV2 achieved reasonable classification accuracy and is the smallest deployed TensorFlow Lite model. Nevertheless, it was the least accurate of the three models tested. Thus, even though MobileNetV2 is more appropriate when lightweight deployment is the key consideration, its lower performance in terms of class-level performance made it less suitable as the final model of SkinAI.

6.2.2 ResNetV2 Result

ResNet50 was trained and evaluated as a deeper CNN comparison model. It was included because residual connections allow deeper models to learn strong image features. The final ResNet50 model achieved a Test Time Augmentation accuracy of 88.89% and a macro F1-score of 0.89. After SavedModel to TensorFlow Lite conversion, the model size was 91.75 MB, making it the largest deployed model among the three evaluated models.

	Precision	*Recall*	*F1-score*	*Support*
Eczema	0.86	0.83	0.84	251
Psoriasis	0.85	0.83	0.84	309
Basal Cell Carcinoma	1.00	1.00	1.00	240
Viral Infections	0.87	0.91	0.89	316
Accuracy			0.89	1116

Macro avg	0.89	0.89	0.89	1116
Weighted avg	0.89	0.89	0.89	1116

Table 6.3 Classification report for ResNet50

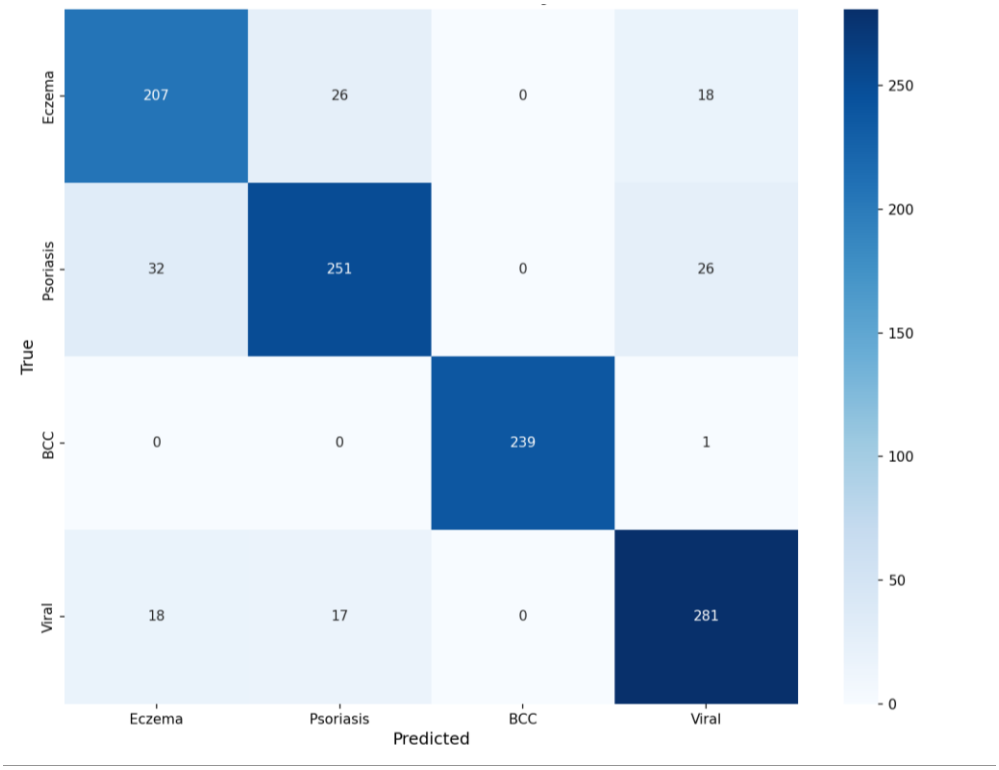


Figure 6.2 Confusion Matrix of ResNet50

Figure 6.2 shows the confusion matrix of ResNet50. The model was able to classify correctly 207 eczema images, 251 psoriasis images, 239 basal cell carcinoma images, and 281 images of viral infections. ResNet50 was better than MobileNetV2 in the classification of psoriasis and viral infections. It was found that the images of viral infection were on average better classified with the increase in the number of correctly classified images to 281.

Similar to MobileNetV2, ResNet50 demonstrated good performance in the presence of basal cell carcinoma where only one basal cell carcinoma image was incorrectly classified as a viral infection. Nevertheless, eczema and psoriasis were still misclassified with each other and with viral infections. This implies that even with a more complex CNN framework, the visual similarity between inflammatory skin diseases would still be a challenge.

In general, ResNet50 was more accurate than MobileNetV2 and provided better performance on the class level. Nonetheless, its deployed TensorFlow Lite model size of 91.75 MB is far bigger than the other models. As SkinAI is expected to be deployed on Android, this

large model size means ResNet50 would not be appropriate to use in the final mobile application, even though it has good classification performance.

6.2.3 EfficientNetB2 Result

EfficientNetB2 was trained and evaluated as the refined EfficientNet candidate. It achieved the highest Test Time Augmentation accuracy among the three models, with 89.25% TTA accuracy. It also achieved the best macro F1-score of 0.90. After SavedModel to TensorFlow Lite conversion, the deployed model size was 30.8 MB.

	Precision	*Recall*	*F1-score*	*Support*
Eczema	0.84	0.86	0.85	251
Psoriasis	0.85	0.83	0.84	309
Basal Cell Carcinoma	1.00	1.00	1.00	240
Viral Infections	0.89	0.89	0.89	316
Accuracy			0.89	1116
Macro avg	0.90	0.90	0.90	1116
Weighted avg	0.89	0.89	0.89	1116

Table 6.4 Classification report for EfficientNetB2

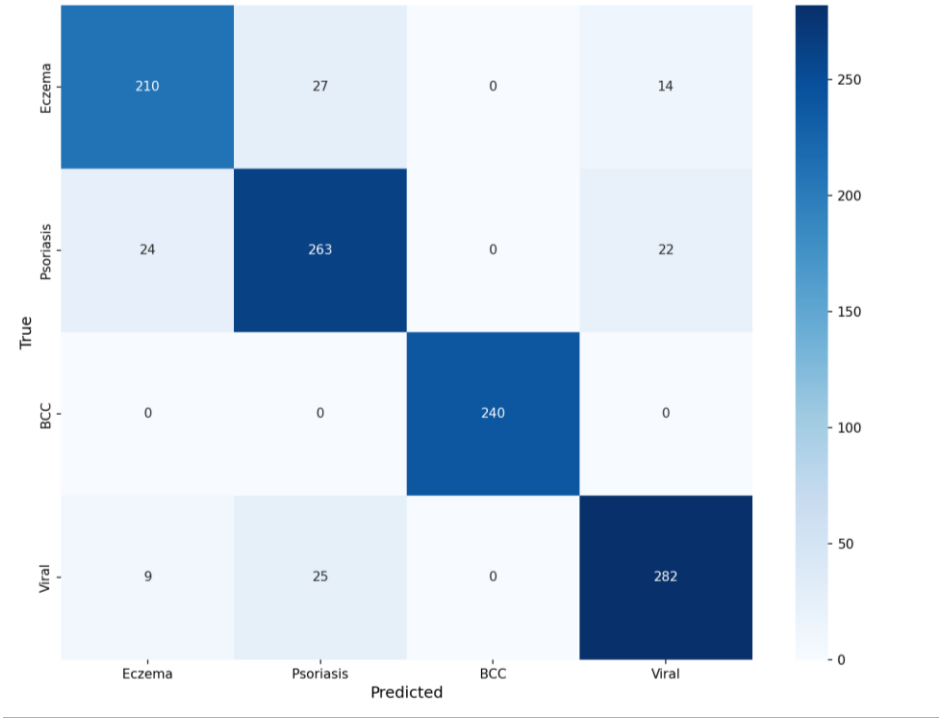


Figure 6.3 Confusion Matrix of EfficientNetB2

Figure 6.3 shows the confusion matrix of EfficientNetB2. The model accurately predicted 210 images of eczematous lesions, 263 images of psoriasis lesions, 240 images of basal cell carcinoma lesions, and 282 images of viral infection lesions. EfficientNetB2 made the most correct predictions of eczema, psoriasis, and viral infections. Basal cell carcinoma also was classified perfectly with all 240 testing images correctly predicted.

The confusion matrix shows that EfficientNetB2 minimized a few misclassification points in comparison with other models. Indicatively, the classification of psoriasis increased to 263 correct classifications, which is higher as compared to MobileNetV2 and ResNet50. Classification of viral infection was also very good with 282 correct classifications. Nevertheless, there was still a mix-up between eczema, psoriasis and viral infections. This means that the model still finds it difficult to differentiate visually similar skin conditions, particularly when the symptoms of redness, scaling, inflammation and abnormal texture overlap.

Overall, EfficientNetB2 attained the highest overall classification performance in comparison with the other two models. It achieved the best TTA accuracy and the best macro F1-score yet has a significantly smaller deployed model size than ResNet50. Whereas it was bigger than MobileNetV2, it offered better classification performance and a more balanced

trade-off between accuracy and deployment feasibility. Hence, EfficientNetB2 has been chosen as the ultimate model of SkinAI.

6.2.4 Final Model Comparison

According to the classification reports and confusion matrices, all three models were very strong in the classification of basal cell carcinoma. This class showed close to perfect or ideal classification performance in the models that were tested. Eczema, psoriasis and viral infection had more misclassification due to the fact that these conditions may have similar visual appearances.

Model	**TTA Accuracy**	**Macro F1-score**	**SavedModel TFLite Size**	**Mobile Suitability**
MobileNetV2	85.48%	0.86	9.83 MB	Good
ResNet50	88.89%	0.89	91.75 MB	Not suitable
EfficientNetB2	89.25%	0.90	30.8 MB	Selected

Table 6.5 Final Model Testing Comparison

The smallest deployed model was MobileNetV2, which had the lowest TTA accuracy. ResNet50 offered higher performance and class-level accuracy, but the deployed TensorFlow Lite model size was too large to be deployed in a mobile environment. EfficientNetB2 achieved the best overall performance and provided a better balance between classification accuracy, macro F1-score, and deployed model size.

Therefore, **EfficientNetB2 was selected as the final model** because it provided the best balance between classification performance and Android deployment suitability.

6.3 Android Application Testing

Android application testing was conducted to verify whether the implemented SkinAI application overall worked as expected.

6.3.1 Functional Testing

Functional testing was performed to verify the core application functions. The test cases covered the main user workflow from launching the application to viewing saved scan history.

Test Case	Expected Result	Status
1. Launch application	Login screen is displayed	Pass
2. Login with Google account	User is authenticated successfully	Pass
3. Continue anonymously	User enters the application anonymously	Pass
4. Accept terms and disclaimer	User proceeds to home screen	Pass
5. Open home screen	Main functions are displayed	Pass
6. Select image from gallery	Selected image is loaded	Pass
7. Capture image using camera	Captured image is loaded	Pass
8. Run classification	Disease label and confidence score are displayed	Pass
9. Apply confidence threshold	Low-confidence prediction shows unable-to-classify result	Pass
10. Save scan result	Result metadata is stored in Firebase Firestore	Pass
11. Upload image	Image is uploaded to Cloudinary	Pass
12. View history	Previous scan records are displayed	Pass
13. Open disease knowledge	Disease information is displayed	Pass
14. Open profile	User profile information is displayed	Pass

Table 6.6 Functional Testing Test Cases

The functional testing results show that the main application functions worked as expected. Users were able to access the application, provide images, obtain classification results, save results, and retrieve previous scan records.

6.3.2 Image Quality Testing

The quality testing of the image was done to determine whether the system had the capability to reject inappropriate images prior to classification. This is significant since bad image quality can decrease the reliability of predictions. The conditions that were tested were clear skin images, blurry images, poor lighting images, images with excessive dark background, graphic or non-photo images, non-skin images, and valid skin images.

Test Input	Expected Result	Status
1. Clear skin image	Image passes quality checking	Pass
2. Blurry image	Image unclear message is displayed	Pass
3. Too dark image	Poor lighting message is displayed	Pass
4. Too bright image	Poor lighting message is displayed	Pass
5. Image with excessive dark background	Unsuitable image message is displayed	Pass
6. Graphic or QR-code-like image	Invalid image message is displayed	Pass
7. Non-skin image	Not a skin image message is displayed	Pass
8. Image with sufficient skin region	Image proceeds to classification	Pass

Table 6.7 Image Quality Testing

The image quality checker came in handy to avoid sending inappropriate images to the classifier. This enhances the accuracy of the classification process since more suitable image inputs get into the model.

6.3.3 Firebase and Cloudinary Testing

To check whether scan records were able to be saved and retrieved in their proper locations. Firebase and Cloudinary testing was done. The image was then uploaded to Cloudinary, and the result URL of the image was stored in Firebase Firestore along with the result metadata. The history page then used to retrieve the saved data in Firestore.

Test Case	Expected Result	Status
1. Upload scan image to Cloudinary	Image URL is returned	Pass
2. Save disease result to Firestore	Result metadata is stored	Pass
3. Save confidence score	Confidence score is stored with result	Pass
4. Save timestamp	Date and time are stored	Pass
5. Save user ID	Scan record is linked to the user	Pass
6. Retrieve scan history	Saved records are displayed	Pass
7. Display scan image in history	Image loads correctly from stored URL	Pass

Table 6.8 Firebase and Cloudinary Testing

The testing results show that the backend and storage functions worked correctly. The uploaded image was stored in Cloudinary and the structured scan metadata were stored in Firebase Firestore. This enabled scan records to be retrieved and show up on the history page.

6.3.4 User Interface Testing

To make sure that the screens of the application were clear and easy to use, user interface testing was conducted. The test was aimed at the ability of the users to move between screens, comprehend the messages being shown, and to complete the classification process without technical knowledge.

Screen / Function	Expected Result	Status
1. Login screen	Login options are visible and understandable	Pass
2. Terms screen	Disclaimer is readable and must be accepted	Pass
3. Home screen	Main functions are accessible	Pass
4. Classification screen	Camera and gallery options are clear	Pass
5. Error message display	Error message is understandable	Pass
6. Result screen	Disease result and confidence score are readable	Pass
7. History screen	Previous scan records are displayed clearly	Pass

8. Knowledge screen	Disease information is readable	Pass
9. Profile screen	User information and account option are displayed	Pass

Table 6.9 User Interface Testing

The results of the user interface testing reveal that the application offers a clear workflow to the general users. The application screens facilitate the main classification process and give the appropriate feedbacks when the user input is not appropriate.

6.4 Discussion and Project Challenges

This section is about the model performance, mobile deployment challenge, and system limitations found based on the evaluation results. It is an important discussion since the classification accuracy alone should not be used to determine the final model selection. Since SkinAI is intended to be a mobile application, the model chosen should also be realistic to run on Android and should be able to support a useful application development workflow.

6.4.1 Model Performance Discussion

The results of the model testing indicate that EfficientNetB2 has the highest overall performance compared with the other two tested models. MobileNetV2 was the lowest performing model in terms of TTA accuracy and the size of the deployed TensorFlow Lite model was the smallest of all three models. ResNet50 had good classification performance, but its deployed TensorFlow Lite model size was very large in comparison to the other models. EfficientNetB2 had the highest TTA accuracy and optimum macro F1-score and a more practical deployed size model compared to ResNet50.

The confusion matrices also indicate that the easiest class to classify is basal cell carcinoma. The three models performed very well in this class with EfficientNetB2 and MobileNetV2 correctly classifying all their 240 images of basal cell carcinoma. This could be due to the fact that basal cell carcinoma images have more visually distinct lesion patterns than do inflammatory skin disease images.

Conversely, there were higher misclassification in eczema, psoriasis and viral infection. Eczema and psoriasis were at times confused as such since they may exhibit similar visual characteristics that include redness, scaling, inflammation, and irregular skin texture. Certain

images of viral infections were also not correctly identified as eczema or psoriasis, which is an indication that visual similarity between skin disease types is not an easy matter.

Overall, EfficientNetB2 was the most suitable in terms of balancing classification capabilities and mobile application applicability. MobileNetV2 even though smaller in size had lower accuracy and class-level performance. ResNet50 was a highly performing deployed model, but its extremely large, deployed model size rendered it less practical to deploy on Android. Thus, EfficientNetB2 turned out to be the most appropriate model to implement the final SkinAI application.

6.4.2 Mobile Deployment Challenges

The implementation of the chosen model into Android application was done with the help of TensorFlow Lite. The process of model conversion generated, however, a critical deployment challenge. The direct optimized TensorFlow Lite conversion method created a smaller model file but failed in Android execution due to an unsupported version of the operator FULLY_CONNECTED.

To solve this issue, the SavedModel to TensorFlow Lite conversion method was used. This approach resulted in a larger model file, although it is compatible with the TensorFlow Lite version 2.14.0. The sizes of final deployed TensorFlow Lite models were 9.83 MB in the case of MobileNetV2, 91.75 MB in the case of ResNet50, and 30.8 MB in the case of EfficientNetB2. Even though EfficientNetB2 had increased in size after being converted to SavedModel, it was still significantly smaller than ResNet50 and could be used in the Android application.

This challenge demonstrates that the mobile deployment is not necessarily based solely on model accuracy or original trained model size. The runtime compatibility, support of TensorFlow Lite operators, inference speed, and storage requirement should also be taken into account. In this project, having a compatible TensorFlow Lite model was more significant than having a smaller model that could not run properly. Thus, the conversion method SavedModel was chosen to be sure that the final model could be loaded and successfully executed in the Android application.\

6.4.3 System Limitations

There are a number of limitations to SkinAI. First, the system can only support four types of diseases which are eczema, psoriasis, basal cell carcinoma, and viral infections. Given an image of a different skin condition, the model can still be used to classify the image into one of the four supported classes. This can cause a misinterpretation in case the user gets the output as a confirmed diagnosis wrong.

Second, the system heavily relies on image quality. Despite the image quality checker which allows one to reject unsuitable images, the performance of the classifier can still be influenced by lighting condition, camera angle, background, clarity of image, and visible skin area. The image quality checker eliminates inappropriate input, but it does not ensure that all accepted images are clinically ideal to classify them.

Third, the data was obtained in the public sources and may not be fully representative of all real-life clinical situations, local Malaysian clinical images, other skin tones, and other imaging environments. This restricts the ability of generalizing the model when used beyond the testing dataset.

Fourth, the system has not been clinically validated by dermatologists. Thus, SkinAI can be used solely as an initial screening and awareness tool. The users are advised to seek medical care to have appropriate diagnosis and treatment. This is in line with the objective of SkinAI as a screening application and not a substitute to the professional medical diagnosis.

6.5 Objectives Evaluation

This section evaluates whether the project objectives stated in Chapter 1 were achieved.

6.5.1 Objective 1 Evaluation

The initial goal of this project was to create and train deep learning models that would be able to classify four types of skin diseases, that is, eczema, psoriasis, basal cell carcinoma, and viral infections. This objective was achieved.

Three CNN-based transfer learning models were generated and trained in this project using the ready skin disease data. These models included MobileNetV2, ResNet50 and EfficientNetB2. The dataset was split into training, validation and testing sets where the testing

set was held aside during the process of training and validation to assess the performance of the model on the unseen images.

The accuracy, precision, recall, F1-score, and confusion matrix were used to evaluate the trained models. The results showed that all three models were able to classify the four selected skin disease categories with acceptable performance. Of the three models, EfficientNetB2 may be considered the most successful with the the highest Test Time Augmentation accuracy of 89.25% and the best macro F1-score of 0.90. Hence, the initial objective was met since the project was able to create and train CNN models that would be able to classify the four targeted skin disease categories.

6.5.2 Objective 2 Evaluation

The second aim of this project was to compare MobileNetV2, ResNet50, and EfficientNetB2 to identify the most suitable model to use on mobile devices. This objective was achieved.

To have a fair comparison, the three models were trained and evaluated on the same dataset and similar training strategy. The comparison was based on classification performance and appropriateness of mobile deployment, including Test Time Augmentation accuracy, macro F1-score, confusion matrix performance, and TensorFlow Lite model size.

The MobileNetV2 has an 85.48% TTA accuracy and a macro F1-score of 0.86 with a SavedModel TensorFlow Lite size of 9.83 MB. ResNet50 had a TTA accuracy of 88.89% and a macro F1-score of 0.89, but had a SavedModel TensorFlow Lite size of 91.75 MB, so is not as suitable to run on mobile. EfficientNetB2 had the best TTA accuracy of 89.25% and the highest macro F1-score of 0.90 with a SavedModel TensorFlow Lite size of 30.8 MB.

Since EfficientNetB2 offered the most optimal balance between classification and deployment capabilities. Thus, EfficientNetB2 was chosen as the last model of SkinAI and the second goal was reached.

6.5.3 Objective 3 Evaluation

The third goal of the project was to make the choice of AI model integrated into a mobile application with a convenient interface. This goal was achieved.

The chosen EfficientNetB2 was transformed into the TensorFlow Lite format and implemented into the Android application. It has been implemented in Android Studio with Java and has the necessary functions to compose a complete workflow of skin disease screening. The functions are Google sign in, anonymous login, acceptance of terms and conditions, capture of camera image, selection of gallery image, checking of image quality, display of classification results, save scan results, Cloudinary image upload, Firebase Firestore history storage, disease knowledge display, and profile management.

The Android application testing showed that the main functions worked as expected. Users could log in, accept the terms and disclaimer, provide an image of their skin, receive feedback on image quality, view the classification result, save the scan result, access previous history records, access disease knowledge, and manage his or her profile. Thus, the third goal was fulfilled since the chosen AI model was properly embedded in a working and convenient to use Android application.

6.6 Concluding Remark

In this chapter, the assessment and discussion of SkinAI system were presented. The results of the model testing revealed that EfficientNetB2 was the most appropriate model to be deployed to Android when compared to ResNet50, as it showed the best overall results with high TTA accuracy and macro F1-score, and was also much more appropriate to be deployed to Android when compared to ResNet50. Thus, EfficientNetB2 was chosen as the final model which would be used in the application.

During the Android application testing, the core functions used worked as expected and they included authentication, terms acceptance, image input, image quality checking, classification, result display, cloud storage, history retrieval, disease knowledge, and profile management. Other limitations in the discussion included supported disease scope, dependency on image quality, limitations in data sets and unproven clinical results. In general, the assessment revealed that the project goals were met. In the following chapter, the project is concluded and recommendations given on how the project can be improved in future.

Chapter 7
System Conclusion and Recommendations

This chapter is the final chapter of the SkinAI project, which summarizes the entire development, performance, contributions of the project, limitations, and recommendations to develop the project further.

7.1 Conclusion

SkinAI was developed as an Android-based application for preliminary skin disease SkinAI is an Android-based application that has been created to screen the preliminary stages of skin diseases. The system enables users to provide skin images via the camera or gallery, performs extensive image quality checks, including blur detection, brightness analysis, dark pixel ratio, and skin pixel ratio validation. It classifies the image using a TensorFlow Lite model, displays the predicted disease category and confidence score and stores scan records with Cloudinary to store images and Firebase Firestore to store metadata. Other features included in the application are user authentication using Google Sign-In and anonymous access, viewing scan history, disease knowledge carousel, and user profile management.

Three CNN-based transfer learning models were trained and evaluated in this project, including MobileNetV2, ResNet50, and EfficientNetB2. All models were trained on four-class skin diseases classification which are Eczema, Psoriasis, Basal Cell Carcinoma (BCC) and Viral Infections. According to the final evaluation results, EfficientNetB2 has the most balanced score in terms of classification performance like approximately 89% accuracy with Test Time Augmentation and mobile deployment suitability in terms of model compatibility and inference speed. Consequently, EfficientNetB2 was chosen as the ultimate model and effectively implemented into the Android application with the help of TensorFlow Lite.

Generally, the project objectives were met. The project was able to design and test three CNN models to classify four-class skin disease, implement strong image quality checking solutions to guarantee quality inputs, and package the chosen EfficientNetB2 model into a fully functional Android app with cloud-based storage and user management. It is noteworthy that

SkinAI must not be considered as a substitute to professional medical diagnosis, but as a preliminary screening and awareness tool. Users should always seek the advice of qualified healthcare professionals to get the right medical advice and treatment.

7.2 Recommendations

There are a number of ways in which future work can be improved. To begin with, it is possible to extend the dataset by adding more high-quality and dermatologist-approved images of skin diseases. To enhance model generalization, a bigger and more variable dataset can assist in improving model generalization, particularly in classes that are visually similar, like eczema and psoriasis. More variation in skin tone, lighting condition, camera quality, and the real life image settings should also be added to future datasets.

Second, the system can be supplemented with additional categories of skin diseases. At the moment SkinAI supports only four classes, it is possible to add more common skin diseases to the application to make it more useful and comprehensive. A class that is unknown or not supported may also be added to minimize the risk of trying to put an unsupported disease in one of the existing categories.

Third, the model could be optimized even more to be deployed in the mobile environment. Even though the final TensorFlow Lite model was functioning properly, the deployed model size can be still improved. Future research can investigate post-training quantization, float16 quantization, or other TensorFlow Lite optimization techniques to achieve a smaller model size at the cost of an acceptable model size and compatibility.

Fourth, the quality checker of the image can be enhanced. The present-day checker takes into consideration blur, brightness, dark background, unique colour variation, and skin pixel ratio. The model can be made more concentrated on the affected skin region rather than the surrounding background by adding more advanced skin region detection or segmentation to the model.

Finally, in future work, clinical validation should be taken into consideration. The model is to be tested with the clinical images which are reviewed by a dermatologist and only then proceeds to be used in any real healthcare setting. This would enhance the accuracy of the system and guarantee the application to be safe and responsible as a pre-screening tool

REFERENCES

[1]
W. Montagna, "Human skin | anatomy," *Encyclopædia Britannica*. 2019. Available: https://www.britannica.com/science/human-skin

[2]
ifaadmin, "Lifestyle choices that cause skin damage," *Artisan Aesthetic Clinics*, Mar. 01, 2019. https://artisanclinics.com/blog/lifestyle-choices-that-cause-skin-damage/

[3]
Y.-Y. Goh, F. Keshavarzi, and Y. L. Chew, "Prevalence of Atopic Dermatitis and Pattern of Drug Therapy in Malaysian Children," *Dermatitis*, vol. 29, no. 3, pp. 151–161, May 2018, https://pubmed.ncbi.nlm.nih.gov/29762208/

[4]
S. E. Choon *et al.*, "Incidence and prevalence of psoriasis in multiethnic Johor Bahru, Malaysia: a population-based cohort study using electronic health data routinely captured in the Teleprimary Care (TPC®) clinical information system from 2010 to 2020: Classification: Epidemiology," *The British Journal of Dermatology*, Jul. 2022, https://pmc.ncbi.nlm.nih.gov/articles/PMC9804555

[5]
M. H. Ahmad Fuad *et al.*, "Prevalence and associated factors of suspected occupational skin diseases among restaurant workers in peninsular Malaysia: secondary data analysis of Registry for Occupational Disease Screening (RODS)," *BMJ Open*, vol. 14, no. 8, p. e079877, Aug. 2024, https://pmc.ncbi.nlm.nih.gov/articles/PMC11331978

[6]
D. Duniphin, "Limited Access to Dermatology Specialty Care: Barriers and Teledermatology," *Dermatology Practical & Conceptual*, vol. 13, no. 1, p. e2023031, Jan. 2023, doi: https://doi.org/10.5826/dpc.1301a31.

[7]

REFERENCES

IBM, “What Is Machine learning?,” *IBM*, Sep. 22, 2021. https://www.ibm.com/think/topics/machine-learning

[8]
Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, https://pubmed.ncbi.nlm.nih.gov/26017442/

[9]
M. Paauwe, “Convolutional Neural Networks Explained with Examples,” *Dragon1.com*, 2025. https://www.dragon1.com/terms/convolutional-neural-networks-definition (accessed Sep. 09, 2025).

[10]
Yashvi Chandola, J. Virmani, H.S. Bhadauria, and P. Kumar, “Lightweight end-to-end Pre-trained CNN-based computer-aided classification system design for chest radiographs,” *Elsevier eBooks*, pp. 167–183, Jan. 2021, https://www.sciencedirect.com/topics/computer-science/mobilenetv2

[11]
M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” Mar. 2019. Available: https://arxiv.org/pdf/1801.04381

[12]
K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015. Available: https://arxiv.org/pdf/1512.03385

[13]
M. Tan and Q. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” Sep. 2020. Available: https://arxiv.org/pdf/1905.11946

[14]
IBM, “Transfer Learning,” *Ibm.com*, Feb. 12, 2024. https://www.ibm.com/think/topics/transfer-learning

[15]
"Diagnosis of Skin Disorders - Skin Disorders," *MSD Manual Consumer Version*. https://www.msdmanuals.com/home/skin-disorders/biology-of-the-skin/diagnosis-of-skin-disorders

[16]
Jagdis etal., Cruz-Vargas J.A.D.L., Camacho M.E.R.
Advance study of skin diseases detection using image processing methods Nat. Volatiles Essent. Oils J., 9 (1) (2022), pp. 997-1007

[17]
T. Shanthi, R. S. Sabeenian, and R. Anand, "Automatic diagnosis of skin diseases using convolution neural network," *Microprocessors and Microsystems*, vol. 76, p. 103074, Jul. 2020, doi: https://doi.org/10.1016/j.micpro.2020.103074.

[18]
A. Pal, S. Ray, and U. Garain, "Skin disease identification from dermoscopy images using deep convolutional neural network," *arXiv.org*, 2018. https://arxiv.org/abs/1807.09163

[19]
Maduranga M., Nandasena D.
Mobile-based skin disease diagnosis system using convolutional neural networks (CNN)
I.J. Image Graphics Signal Process., 3 (2022), pp. 47-57

[20]
admin, "Dermatologist Statistics Worldwide: Global Access (Data)," *Dermatologist London - Dermatology Clinic UK (Skin Doctor)*, Feb. 10, 2026. https://www.london-dermatology-centre.co.uk/blog/dermatologist-statistics-worldwide/

[21]
A. B. Wilder-Smith *et al.*, "Strengthening dermatology capacity and competency in resource-limited settings: A viewpoint," *PLOS Neglected Tropical Diseases*, vol. 19, no. 11, p. e0013720, Nov. 2025, doi: https://doi.org/10.1371/journal.pntd.0013720.

[22]
The Ministry of Health Malaysia. "MOH Malaysia Dermatology Services Operational Policy", Aug 2016, Available: https://www.moh.gov.my/moh/resources/Polisi/Dermatology_Services_Operational_Policy_2016.pdf?utm_source=chatgpt.com

[23]
Jamunarani Appalasamy, "A study on awareness of skin infection among adults in Petaling district, Malaysia," *ResearchGate*, Jan. 07, 2016. https://www.researchgate.net/publication/336305325_A_study_on_awareness_of_skin_infection_among_adults_in_Petaling_district_Malaysia (accessed Sep. 08, 2025)

[24]
D. Li, S. Fan, H. Zhao, J. Song, W. Li, and X. Xu, "Global, regional and national burden of skin and subcutaneous diseases: a systematic analysis of the Global Burden of Disease Study 2021," *International Health*, Jun. 2025, doi: https://doi.org/10.1093/inthealth/ihaf070.

[25]
K. Chen and X. Liu, "Global burden of skin cancer and its subtypes: a comprehensive analysis from 1990 to 2021 with projections to 2040," *Frontiers in Public Health*, vol. 13, Sep. 2025, doi: https://doi.org/10.3389/fpubh.2025.1610661.

[26]
"Atopic Dermatitis Has High Global Burden," *Clinical Advisor*, Oct. 15, 2024. https://www.clinicaladvisor.com/news/atopic-dermatitis-has-high-global-burden/

[27]
S. M. Langan *et al.*, "Trends in eczema prevalence in children and adolescents: A Global Asthma Network Phase I Study," *Clinical & Experimental Allergy*, vol. 53, no. 3, pp. 337–352, Feb. 2023, doi: https://doi.org/10.1111/cea.14276.

[28]
National Psoriasis Foundation, "Psoriasis Statistics," *www.psoriasis.org*, Dec. 21, 2022. https://www.psoriasis.org/psoriasis-statistics/

[29]
D. Li, M. Chen, W. Li, X. Xu, and Q. Li, “Global burden of viral skin diseases from 1990 to 2021: a systematic analysis for the global burden of disease study 2021,” *Frontiers in Public Health*, vol. 13, Feb. 2025, doi: https://doi.org/10.3389/fpubh.2025.1464372.

[30]
Z. Kwan, S.-M. Wong, S. Robinson, L. L. Tan, and R. Ismail, “Pattern of skin diseases among patients attending an outpatient dermatology clinic in a tertiary hospital in urban Malaysia,” Australasian Journal of Dermatology, vol. 58, no. 4, pp. e267–e268, Jun. 2017, doi: https://pubmed.ncbi.nlm.nih.gov/28660702/

[31]
I. Hossain, “Skin diseases image dataset,” *Kaggle.com*, 2021. https://www.kaggle.com/datasets/ismailpromus/skin-diseases-image-dataset/discussion?sort=undefined (accessed Sep. 10, 2025).

APPENDIX

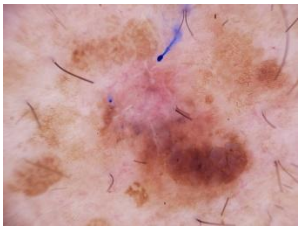
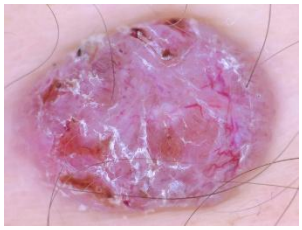
Appendix A-1 Sample of Skin Disease Dataset



Sample image of Eczema from the dataset



Sample image of Psoriasis from the dataset

Sample image of Basal Cell Carcinoma from the dataset

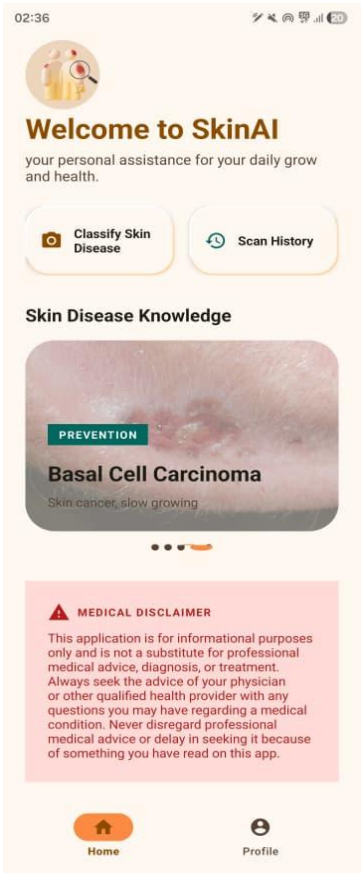


Sample image of viral infection from dataset.

Appendix B-1

Full View of Application

Term Page



Home Page

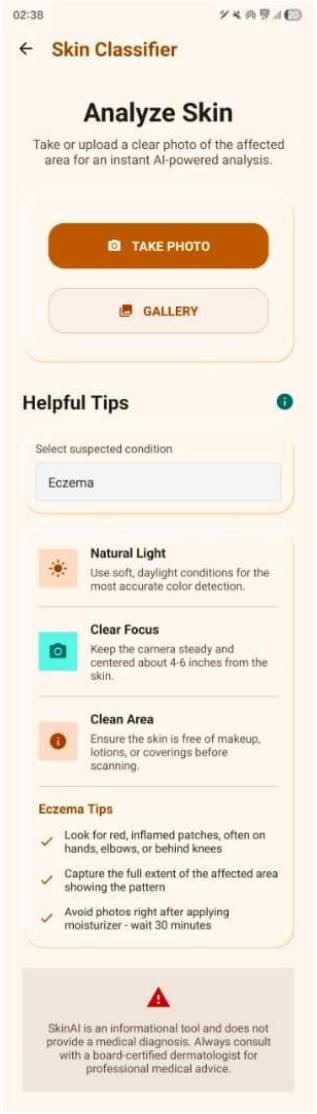
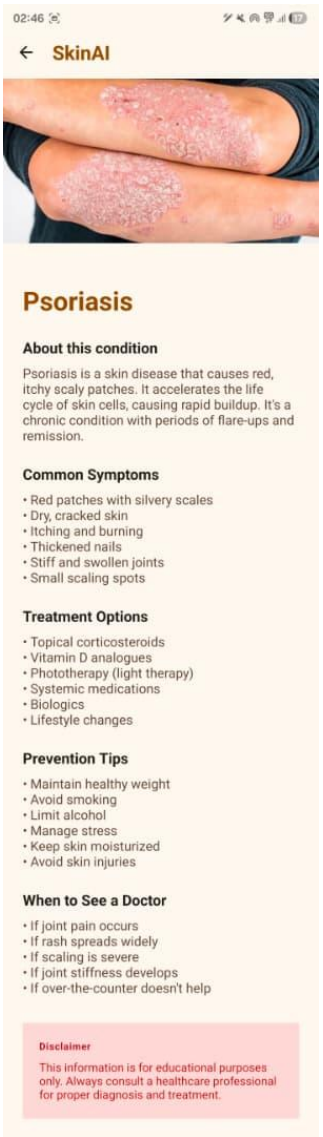


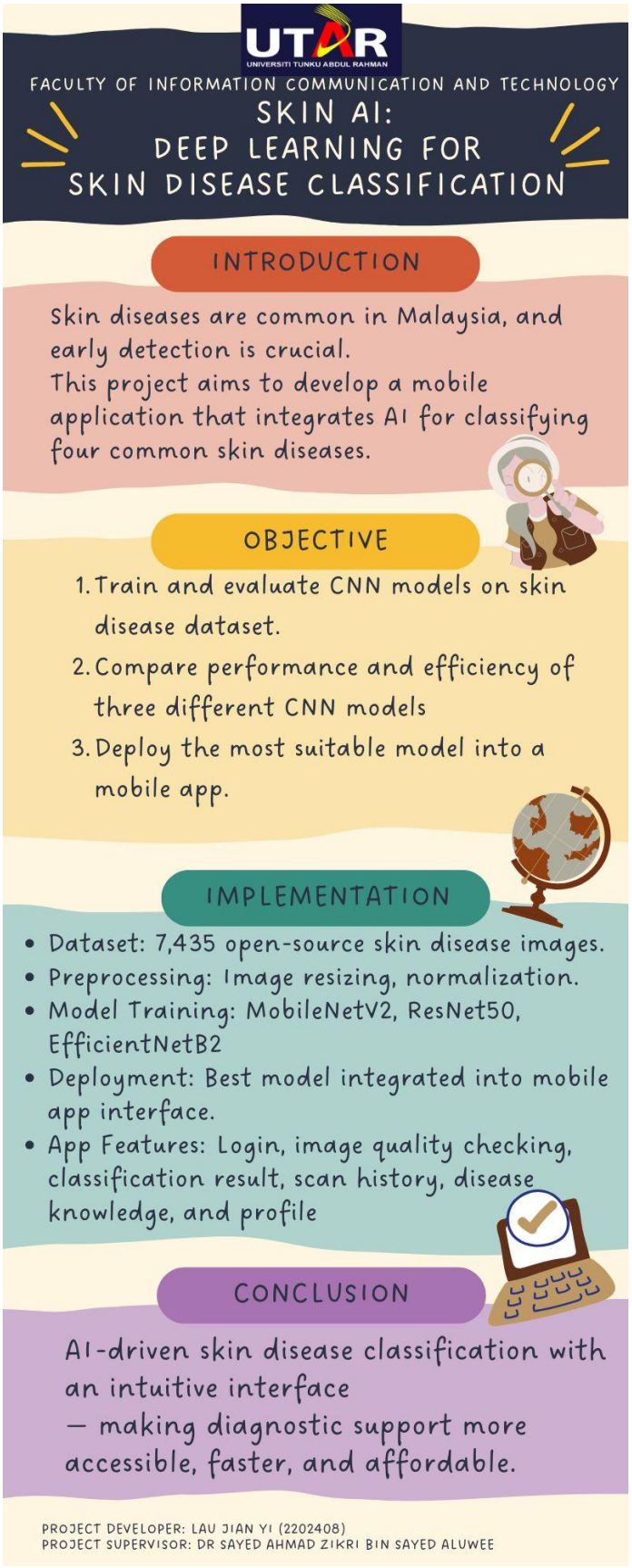
Image Input Page

Classification Result Page



Knowledge Details Page

POSTER


UTAR
UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION COMMUNICATION AND TECHNOLOGY
SKIN AI:
DEEP LEARNING FOR
SKIN DISEASE CLASSIFICATION
INTRODUCTION
Skin diseases are common in Malaysia, and early detection is crucial.
This project aims to develop a mobile application that integrates AI for classifying four common skin diseases.
OBJECTIVE
1. Train and evaluate CNN models on skin disease dataset.
2. Compare performance and efficiency of three different CNN models
3. Deploy the most suitable model into a mobile app.
IMPLEMENTATION
• Dataset: 7,435 open-source skin disease images.
• Preprocessing: Image resizing, normalization.
• Model Training: MobileNetV2, ResNet50, EfficientNetB2
• Deployment: Best model integrated into mobile app interface.
• App Features: Login, image quality checking, classification result, scan history, disease knowledge, and profile
CONCLUSION
AI-driven skin disease classification with an intuitive interface
— making diagnostic support more accessible, faster, and affordable.
PROJECT DEVELOPER: LAU JIAN YI (2202408)
PROJECT SUPERVISOR: DR SAYED AHMAD ZIKRI BIN SAYED ALUWEE