



الجامعة المصرية للتعليم الإلكتروني الأهلية
THE EGYPTIAN E-LEARNING UNIVERSITY

EELU

الجامعة المصرية للتعليم الإلكتروني الأهلية

Faculty of Computers and Information Technology Machine Learning Application for Healthcare

By:

- | | |
|---------------------------------|----------------------|
| 1-John Karam William | (ID: 2000366) |
| 2-Tadros Ashraf Malak | (ID: 2000719) |
| 3- Areeg Aboelhamd Ali | (ID: 2000443) |
| 4- Selvana Raeef Youssef | (ID: 2000641) |
| 5-Mariam Sameh Shafeek | (ID: 2001154) |
| 6-Moreen Ayman Aziz | (ID: 2001916) |
| 7-Mayar Abdelraheem | (ID: 2000866) |

Under Supervision of:

Prof. Dr. Walaa El-Hady

Eng. Sabah Mohamed

Lecturer at the Faculty of Computers and
Information Technology
at the Egyptian E-Learning University

Assistant Lecturer in Faculty of Computers
and Information Technology at
The Egyptian E-Learning University



praises and thanks to God, the Almighty, for his blessings throughout my work.

I would like to express our deep gratitude to Prof. Dr. Walaa El-Hady, our project supervisor for her professional guidance and valuable support, to Eng. Sabah Mohamed Morsy for her useful and constructive recommendations on this project through all development process.

Abstract:

Skin cancer is one of the most prevalent forms of cancer globally, and early detection plays a crucial role in improving patient outcomes. This abstract presents Episcan, a mobile application that utilizes deep learning technology to identify seven types of skin cancer diseases. By leveraging the power of artificial intelligence and computer vision, Episcan aims to provide users with a convenient and accessible tool for assessing potential skin cancer risks. Episcan employs a deep learning model trained on a comprehensive dataset of dermatological images, encompassing a diverse range of skin conditions. The deep learning model is designed to accurately classify and differentiate between the following seven types of skin cancer: melanoma, basal cell carcinoma, squamous cell carcinoma, Merkel cell carcinoma, dermatofibrosarcoma protuberans, cutaneous T-cell lymphoma, and Kaposi's sarcoma. The mobile application offers a user-friendly interface, allowing individuals to capture images of suspicious skin lesions using their smartphone's camera. The captured images are then processed by the deep learning model, which analyzes various features, such as asymmetry, border irregularity, color variation, and size, to determine the likelihood of malignancy. Episcan prioritizes user privacy and data security. All

images uploaded to the application are anonymized and stored securely, adhering to stringent privacy guidelines. Additionally, the application encourages users to consult healthcare professionals for definitive diagnoses and treatment recommendations, emphasizing that Episcan serves as a supportive tool rather than a substitute for medical expertise. The development of Episcan represents a significant advancement in the field of dermatology and skin cancer detection. By harnessing deep learning technology and mobile accessibility, the application has the potential to facilitate early detection, promote awareness, and potentially save lives. Further enhancements and validation studies are planned to continually refine the accuracy and performance of the application.

Table of Contents :

Chapter 1: Introduction.....	8
1.1 Introduction.....	9
1.2 Problem definition.....	11
1.3 Project objectives.....	13
1.4 Motivation.....	15
1.5 scope of work.....	16
Chapter 2: Background	17
2.1 Background.....	18
2.2 Artificial intelligence (AI).....	20
2.3 Algorithm search:.....	21
2.4 Why using CNN:.....	22
2.5 Machine learning.....	22
2.6 Machine Learning Object Detectio.....	23
2.7 Front-end.....	24
2.8 Techniques we used in flutter.....	25
2.8.1 What is flutter?.....	25
2.8.2 Advantages of flutter.....	25
2.8.3 Flutter architectural overview.....	27
2.9 Backend.....	38
Chapter 3: Related Work & Related Applications	41
3.1 Introduction.....	42
3.2 Methods of detecting skin cancer.....	44
3.3 Related application.....	45
3.3.1 Mole Scope.....	45
3.3.2 Miiskin.....	47
3.3.3 VisualDx	49
3.3.4 Dermatology.....	51

Chapter 4: Skin cancer Diseases.....	53
4.1 Melanoma.....	54
4.2 Melanocytic naevus.....	57
4.3 Actinic keratosis.....	59
4.4 Basal cell carcinoma.....	61
4.5 Seborrheic keratosis.....	63
4.6 Dermatofibroma.....	65
4.7 Pyogenic granuloma.....	66
Chapter 5: Proposed solution.....	68
5.1 Introduction.....	69
5.2 System Architecture.....	71
5.3 Main feature.....	73
5.4 Mobile application.....	76
5.4.1 Login and Sign up page.....	77
5.4.2 Login page.....	78
5.4.3 Sign up.....	79
5.4.4 Home page.....	80
5.4.5 Disease Information.....	81
5.4.6 Medication Reminder.....	82
5.4.6 Setting Page.....	83
Chapter 6: Code and Result.....	84
6.1 Project Implementatio Code In machine learning.....	89
6.2 Project Implementation Code in Mobile Application.....	104
6.2.1 Sign Up.....	104
6.2.2 Login page	106
6.2.3 Home Page.....	107
6.2.4 Diesase Information.....	110
6.2.5 Medication Reminder.....	112
Chapter 7: Summary, Conclusions and Future Work.....	114
7.1 Conclusion.....	115
7.2 Challenges	115
7.3 Future Work.....	115
Chapter 8: References.	116

Table Of Figures

<i>Figure 1 Object Detection</i>	23
<i>Figure 2 Flutter layer model</i>	28
<i>Figure 3 Widgets</i>	30
<i>Figure 4 Rendering and layout</i>	34
<i>Figure 5 Flutter Layer Model (Web Support)</i>	41
<i>Figure 6 Mole Scope application</i>	45
<i>Figure 7 Miiskin application</i>	47
<i>Figure 8 VisualDx application</i>	49
<i>Figure 9 Dermatology application</i>	51
<i>Figure 10 Anthracnose</i>	56
<i>Figure 11 Bacterial wilt</i>	58
<i>Figure 12 Belly Rot</i>	60
<i>Figure 13 Downy Mildew</i>	63
<i>Figure 14 Gummy Stem Blight</i>	66
<i>Figure 15 Pythium Fruit Rot</i>	68
<i>Figure 17 System Architecture</i>	70
<i>Figure 18 Main feature</i>	73
<i>Figure 19 machine learning (1)</i>	76
<i>Figure 20 machine learning (2)</i>	76
<i>Figure 21 machine learning (3)</i>	76
<i>Figure 22 machine learning (4)</i>	76
<i>Figure 23 main page</i>	77
<i>Figure 24 login</i>	78
<i>Figure 25 register</i>	79
<i>Figure 26 home page</i>	80
<i>Figure 27 Discovery page (1)</i>	81
<i>Figure 27 Discovery page (2)</i>	82
<i>Figure 28 register application</i>	81
<i>Figure 29 flutter register</i>	79
<i>Figure 30 flutter login</i>	78
<i>Figure 31 flutter home (1)</i>	80
<i>Figure 32 flutter home (2)</i>	80
<i>Figure 33 application home</i>	77
<i>Figure 34 registration</i>	104
<i>Figure 35 login</i>	106
<i>Figure 36 home</i>	107
<i>Figure 37 home Arabic</i>	107
<i>Figure 38 result (1)</i>	108
<i>Figure 39 result (2)</i>	110
<i>Figure 40 result (3) Arabic</i>	112

Chapter 1: Introduction

1.1 Introduction

Skin cancer is a major global health concern, with millions of cases diagnosed each year. Early detection of skin cancer is crucial for effective treatment and improved patient outcomes. In recent years, advances in deep learning technology and computer vision have opened new possibilities for automated skin cancer detection systems. This introduction presents a mobile application called Episcan, which utilizes deep learning technology to identify nine types of skin cancer diseases.

Episcan is designed to provide individuals with a convenient and accessible tool for assessing their risk of skin cancer. By harnessing the power of artificial intelligence and computer vision, the application aims to empower users to make informed decisions about their skin health and seek timely medical attention when necessary.

The core of Episcan lies in a sophisticated deep learning model that has been trained on a large dataset of dermatological images. This dataset encompasses a diverse range of skin conditions, including the nine types of skin cancer targeted by the application: melanoma, basal cell carcinoma, squamous cell carcinoma, Merkel cell carcinoma, dermatofibrosarcoma protuberans, cutaneous T-cell lymphoma, and Kaposi's sarcoma. By learning from this extensive dataset, the deep learning model becomes proficient at recognizing the visual patterns and characteristics associated with each specific type of skin cancer.

The development of Episcan represents a significant step forward in the field of dermatology and skin cancer detection. By leveraging deep learning technology and mobile accessibility, the application has the potential to revolutionize how individuals monitor their skin health. It can augment the capabilities of both healthcare professionals and individuals in identifying potential skin cancer risks, potentially leading to earlier detection, increased survival rates, and improved patient outcomes. Through this documentation, we will delve into the methodologies, techniques, and results of our project to identify 9 types of skin cancer diseases using deep learning. We strive to provide a reliable, efficient, and accessible solution that aids patients and doctors in making informed decisions and mitigating the impact of skin cancer diseases.

1.2 Problem definition

The problem addressed by the Episcan mobile application is the lack of accessible and convenient tools for early detection and identification of nine types of skin cancer diseases. Skin cancer is a prevalent and potentially life-threatening condition, and early detection is crucial for effective treatment and improved patient outcomes. However, the visual examination of skin lesions by non-specialists can be challenging and prone to errors, leading to delayed diagnoses and missed opportunities for early intervention.

Traditional methods of skin cancer detection heavily rely on manual examination by dermatologists, which can be time-consuming, expensive, and often requires specialized medical facilities. This creates barriers to timely assessment, particularly for individuals in remote areas or those with limited access to healthcare resources. Furthermore, there is a growing need to raise awareness about skin cancer and empower individuals to take an active role in monitoring their skin health.

The Episcan mobile application addresses these challenges by leveraging deep learning technology and computer vision to provide a user-friendly and accessible tool for identifying the seven types of skin cancer diseases. By harnessing the power of artificial intelligence, the application aims to enable individuals to conduct preliminary assessments of their skin lesions, helping them determine the potential risk of malignancy and make informed decisions about seeking professional medical advice.

1-Enable early detection: By leveraging deep learning models trained on comprehensive dermatological image datasets, Episcan aims to identify visual patterns and characteristics associated with melanoma, basal cell carcinoma, squamous cell carcinoma, Mycosis Fungoides, dermatofibrosarcoma protuberans, cutaneous T-cell lymphoma, and Kaposi's sarcoma. Early detection increases the likelihood of successful treatment and improves patient outcomes.

2-Enhance accessibility: Episcan provides a mobile platform that can be easily accessed by individuals, regardless of their geographical location or proximity to healthcare facilities. By using a smartphone's camera, users can capture images of skin lesions and receive real-time risk assessment scores, empowering them to take immediate action or seek further medical evaluation.

3-Promote awareness and education: Episcan aims to raise awareness about skin cancer and its different types. The application provides users with accompanying information about each identified skin cancer disease, including risk factors, signs, and symptoms, and encourages them to consult healthcare professionals for definitive diagnoses and treatment recommendations.

4-Ensure privacy and data security: Episcan prioritizes user privacy and data security. All uploaded images are anonymized and stored securely, adhering to stringent privacy guidelines to protect the confidentiality of user information.

To overcome these challenges, the Episcan mobile application is the need for accessible and convenient tools for early detection and identification of the nine types of skin cancer diseases. By leveraging deep learning technology, Episcan aims to empower individuals to take proactive steps towards their skin health, promoting early detection, raising awareness, and potentially saving lives.

1.3 Project objectives:

1. Develop a robust deep learning model: The primary objective of the project is to design and train a deep learning model capable of accurately identifying the nine types of skin cancer diseases targeted by the Episcan application. The model should be trained in a comprehensive dataset of dermatological images, ensuring high accuracy and reliability in its predictions.
2. Create a user-friendly mobile application: The project aims to develop a user-friendly mobile application interface that allows individuals to easily capture images of suspicious skin lesions using their smartphone's camera. The application should have an intuitive design, providing a seamless user experience and ensuring accessibility for users with varying levels of technical proficiency.
3. Provide informative and educational content: The Episcan application should accompany the risk assessment score with informative and educational content about the nine types of skin cancer diseases. This content should include details

about risk factors, signs and symptoms, and recommendations for further medical evaluation. The objective is to empower users with relevant information and encourage them to seek professional medical advice.

4. Ensure user privacy and data security: The project should prioritize user privacy and data security. The Episcan application should adhere to strict privacy guidelines, ensuring that all uploaded images are anonymized and stored securely. User information should be protected throughout the application's usage and handling of data.

5. Validate accuracy and performance: The project should conduct rigorous testing and validation to assess the accuracy and performance of the Episcan application. Validation studies should involve comparing the application's predictions with expert dermatologist diagnoses to evaluate its sensitivity, specificity, and overall performance in identifying the s types of skin cancer diseases.

By achieving these project objectives, the Episcan application aims to revolutionize skin cancer detection and empower individuals to take proactive steps towards their skin health.

The proposed solution involves developing an artificial intelligence-powered mobile application that utilizes the device's camera to identify diseases in their early stages. This application aims to prevent the death of the patient by facilitating

timely disease detection. The scope of work includes creating a user-friendly interface, integrating camera functionality, implementing AI algorithms for disease identification, providing accurate and real-time disease diagnosis, and potentially offering recommendations for treatment or prevention.

Thorough testing, performance optimization, and deployment to mobile platforms will be included, along with effective project management and documentation.

1.4 Motivation:

The development of the Episcan mobile application to identify nine types of skin cancer diseases using deep learning technology is driven by several key motivations:

1. Early Detection and Improved Outcomes.
2. Accessibility and Reach.
3. Bridging the Gap in Dermatology Expertise.
4. Advancements in Deep Learning and Computer Vision.
5. Raising Awareness and Education.

The motivations behind the development of the Episcan mobile application are centered around promoting early detection, accessibility, empowerment, and leveraging advancements in deep learning and computer vision. By providing individuals with a user-friendly tool for identifying skin cancer diseases.

1.5 scope of work:

The proposed solution involves developing an artificial intelligence-powered mobile application that utilizes the device's camera to identify 9 types of skin cancer diseases in their early stages. The scope of work includes creating a user-friendly interface, integrating camera functionality, implementing AI algorithms for disease identification, providing accurate and real-time disease diagnosis, and potentially offering recommendations for prevention.

Chapter 2: Background

2.1 Back ground:

AI (Artificial Intelligence) and Machine Learning are two closely related fields that deal with the development and application of intelligent systems that can perform tasks without explicit programming. Here's some background information on AI and Machine Learning:

1-Artificial Intelligence (AI): AI is a broad field that aims to create intelligent machines capable of simulating human-like intelligence. It encompasses various subfields such as machine learning, natural language processing, computer vision, robotics, and more. AI systems can perceive their environment, reason about it, and make decisions

2-Machine Learning (ML): Machine Learning is a subset of AI that focuses on the development of algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. Instead of following predefined rules, machine learning algorithms learn patterns and relationships from training data, allowing them to generalize and make predictions or decisions on new, unseen data.

- Supervised Learning: In supervised learning, the algorithm is trained on labeled data, where the input data is paired with corresponding target labels or outcomes. The algorithm learns to map the inputs to the correct outputs by minimizing the error between its predictions and the actual labels.

- Unsupervised Learning: Unsupervised learning involves training the algorithm on unlabeled data, where there are no predefined target labels. The algorithm discovers patterns, structures, or relationships in the data without explicit guidance. Clustering and dimensionality reduction are common unsupervised learning techniques.

□ Reinforcement Learning: Reinforcement learning involves an agent learning to interact with an environment to maximize a reward signal. The agent learns through trial and error, receiving feedback in the form of rewards or penalties based on its actions. The goal is to learn an optimal policy that maximizes cumulative rewards.`

3-Data and Training: Machine learning algorithms require data for training, validation, and testing. The quality and quantity of the data play a crucial role in the performance of the trained models. Large datasets are often used to train models effectively. Data preprocessing, feature extraction, and data augmentation techniques are applied to enhance the quality and diversity of the training data.

4-Model Evaluation: Machine learning models are evaluated based on their performance on unseen data. Common evaluation metrics include accuracy, precision, recall, F1 score, and others, depending on the nature of the task. Cross validation and holdout validation are commonly used techniques to assess model performance.

Applications: AI and machine learning have applications in various domains, including image and speech recognition, natural language processing, recommendation systems, autonomous vehicles, healthcare, finance, and many more. These technologies have the potential to automate tasks, improve decision-making, and enhance efficiency in a wide range of industries.

2.2 Artificial intelligence (AI):

AI is a bigger concept to creating intelligent machines that can simulate human thinking capability and behavior, whereas machine learning is an application or a subset of AI that allows machines to learn from data without being programmed explicitly.

Artificial intelligence is a technology using which we can create intelligent systems that can simulate human intelligence.

The Artificial intelligence system does not require to be pre-programmed, instead of that, they use such algorithms which can work with their own intelligence. The AI system is concerned about maximizing the chances of success. It involves machine learning algorithms such as Reinforcement learning algorithms and deep learning. AI is being used in multiple places such as Siri, Google, and AlphaGo.

Artificial Intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs. Artificial Intelligence is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.

Knowledge Engineering is an essential part of AI research. Machines and programs need to have bountiful information related to the world to often act and react like human beings. AI must have access to properties, categories, objects, and relations between all of them to implement knowledge engineering. AI initiates common sense, problem-solving, and analytical reasoning power in machines, which is a much more difficult and tedious job.

1. Image Recognition: AI algorithms can be trained to analyze skin cancer images. By training machine learning models on a large dataset of skin cancer images, the algorithm can learn to distinguish between types of skin cancer. Convolutional neural networks (CNNs) are commonly used for image recognition tasks of skin cancer.

2. Data Collection: To train an AI model, a comprehensive dataset of images is required of skin cancer.

The dataset contains 13000 images divided into 9 diseases of skin cancer,

which are

1. **Melanoma**
2. **Melanocytes**
3. **Dermatofibroma**
4. **Actinic keratosis**
5. **Vascular lesions**
6. **Basal cell carcinoma**
7. **Benign keratosis**
8. **Squamous Cell Carcinoma**
9. **Mycosis Fungoides**

Model Training: The extracted features and corresponding disease labels are used to train a machine learning model. This typically involves training a classification algorithm, such as a CNN, using the labeled dataset. The model learns to associate the extracted features with specific diseases, enabling it to classify new images as skin cancer or not

2.3 Algorithm using:

Convolutional Neural Networks (CNNs) are a powerful tool for machine learning, especially in tasks related to computer vision. Convolutional Neural Networks, or CNNs, are a specialized class of neural networks designed to effectively process grid-like data, such as images

Neural networks consist of a group of neural units (neurons) that are organized within a group of layers (layers), which are as follows:

- a. Input (input layer): This layer is part of receiving the input (raw data) and then preparing it for processing in the network that is completed.
- b. Hidden layer (hidden layer): This layer exists between the input layers (input layer) and the output layer, where the units (neuron units) calculate the sum of the weights for the inputs and then prepare them for the remaining layer via cost functions (activation functions).
- c. Output (output layer): This is the last layer in the network and is not allowed to give results. Except that the components of the network are the same as the multilayer perceptron.

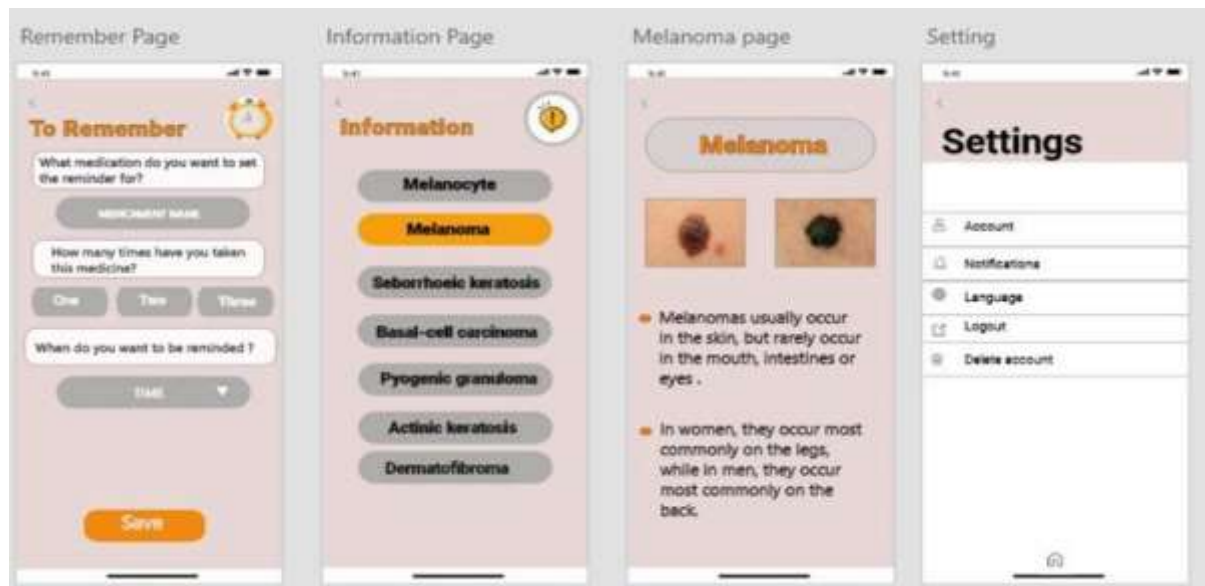
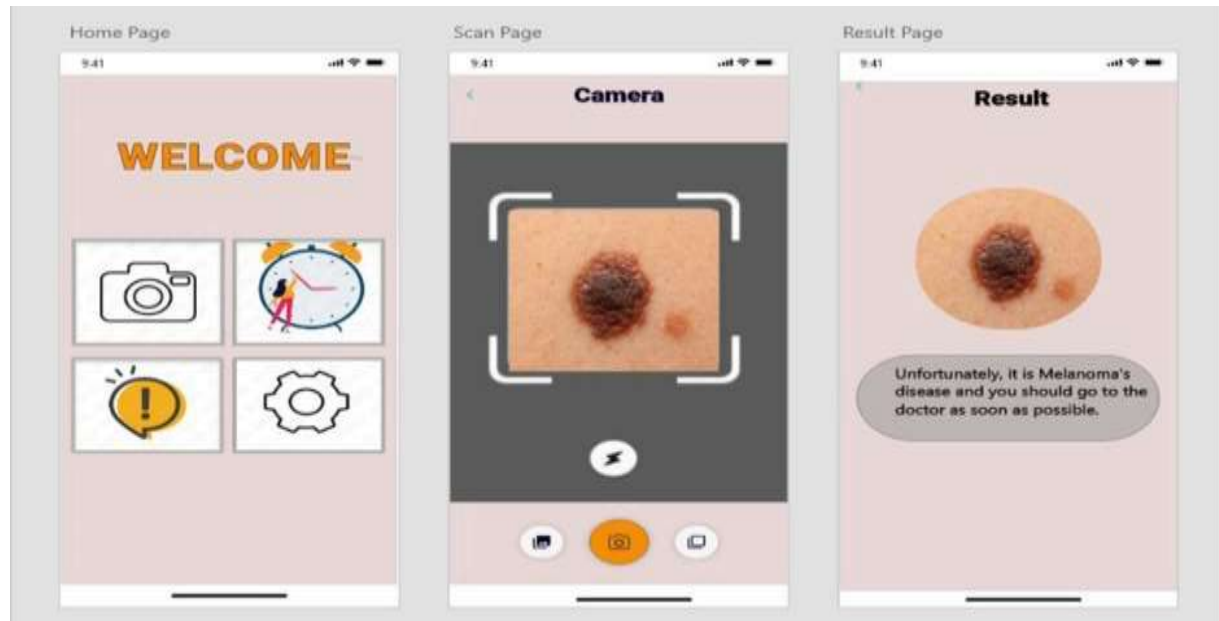
2.4 Why using CNN:

It takes the image as input), it will see a set of pixel values by pixel which is smaller for any image, and the number of these pixels and their values determine the supervision of the distinct image (its size) and the color intensity used in each pixel. This group of pixels is represented by an array of numbers. Write, for example, as follows: 32 x 32 x 3. The first two numbers indicate the dimensions of the image, and the resultant of arriving at the two numbers (1024) is the number of pixels present in the image, while the number 3 indicates the three RGB values. Main channels that use the color of each pixel.

2.5 Machine learning:

- 1- Implement an algorithm to identify and classify skin diseases. Specifically, detect and diagnose 9 types of skin cancer diseases.
- 2- The app offers a feature to provide general information about the seven types of skin cancer, helping patients in better understanding their condition and promoting health awareness.
- 3- Provide real-time analysis of captured images. Ensure quick processing in identification of skin conditions.
- 4- reminding the patient to take medications on a regular basis.
- 5- Present clear results of the skin analysis. Include information on disease detected and recommended actions.

2.6 Machine Learning Object Detection



2.7 Front-end:

The front-end of a web- or mobile application is the part the user interacts with directly. It is usually referred to as the application's "client side." The front end consists of everything that the user sees when interacting with the website or app, such as text colors and styles, photos, graphs and tables, buttons, colors, the navigation menu, and much more. Frontend developers provide the structure, appearance, behavior, and content of everything that appears on browser displays when websites, online applications, or mobile apps are opened. The key focus points of front-end development are responsiveness and performance. A front-end developer must make sure that the site is responsive, meaning that it works properly on devices of all sizes. The application's performance should be always stable, no matter the device is used to access the application. The objective of designing a site is to ensure that when the users open up the site, they see the information in a format that is easy to read and relevant. This is further complicated by the fact that users now use a large variety of devices with varying screen sizes and resolutions thus forcing the designer to take into consideration these aspects when designing the site. They need to ensure that their site comes up correctly in different browsers (cross-browser), different operating systems (cross-platform) and different devices (cross-device), which requires careful planning on the side of the developer. And when using Front End applications on smart phones, we must know the following: Native mobile app developer: i.e., build an app in native languages like kotlin and swift and these apps run on only one platform in Android or IOS. I'm getting more and more switching from java to kotlin. Also, kotlin is somewhat like java in syntax. If you want to build apps for IOS, google swift language learning. Hybrid mobile apps developer: i.e., build hybrid apps and hybrid apps are the apps. that work on both android and iOS platforms.

Do you want to build applications and hybrids using the web and other technologies? Each of these libraries/frameworks is excellent and you can get them from them.

2.8 Techniques we used in flutter:

2.8.1 What is flutter?

Flutter is Google's free and open-source UI framework for creating native mobile applications. Released in 2017, Flutter allows developers to build mobile applications with a single codebase and programming language. This capability makes building both iOS and Android apps simpler and faster.

The Flutter framework consists of both a software development kit (SDK) and their widget-based UI library. This library consists of various reusable UI elements, such as sliders, buttons, and text inputs.

Developers building mobile applications with the Flutter framework will do so Using a programming language called Dart. With a syntax like JavaScript, Dart is typed object programming language that focuses on front-end development

2.8.2 Advantages of flutter:

Although Flutter is a newer cross-platform framework, more and more companies have chosen Flutter over frameworks such as Xamarin, Cordova, and React Native.

Some of the top reasons why development teams choose Flutter include:

- Increased productivity. Using the same codebase for iOS and Android saves both time and resources. Flutter's native widgets also minimize time spent on testing by ensuring there are little to no compatibility issues with different OS versions.`
- Easy to learn. Flutter allows developers to build native mobile applications without needing to access OEM widgets or use a lot of code. This, in addition to Flutter's particularly appealing user interface, makes the mobile app creation process much simpler.
- Easy to learn. Flutter allows developers to build native mobile applications without needing to access OEM widgets or use a lot of code. This, in addition to Flutter's particularly appealing user interface, makes the mobile app creation process much simpler.

- Great performance. Users report that it is difficult to notice the difference between a Flutter app and a native mobile app.
- Cost-effective. Building iOS and Android apps with the same codebase is essentially building two apps for the price of one.
- Available on different IDEs. Developers are free to choose between AndroidStudio and VS Code to edit their code on Flutter.

Great documentation & community. Flutter has many great resources to answer your questions, thanks to its ample documentation with easy-to follow use cases. Flutter users also benefit from community hubs like

Flutter Community and Flutter Awesome for exchanging ideas`

2.8.3 Flutter architectural overview:

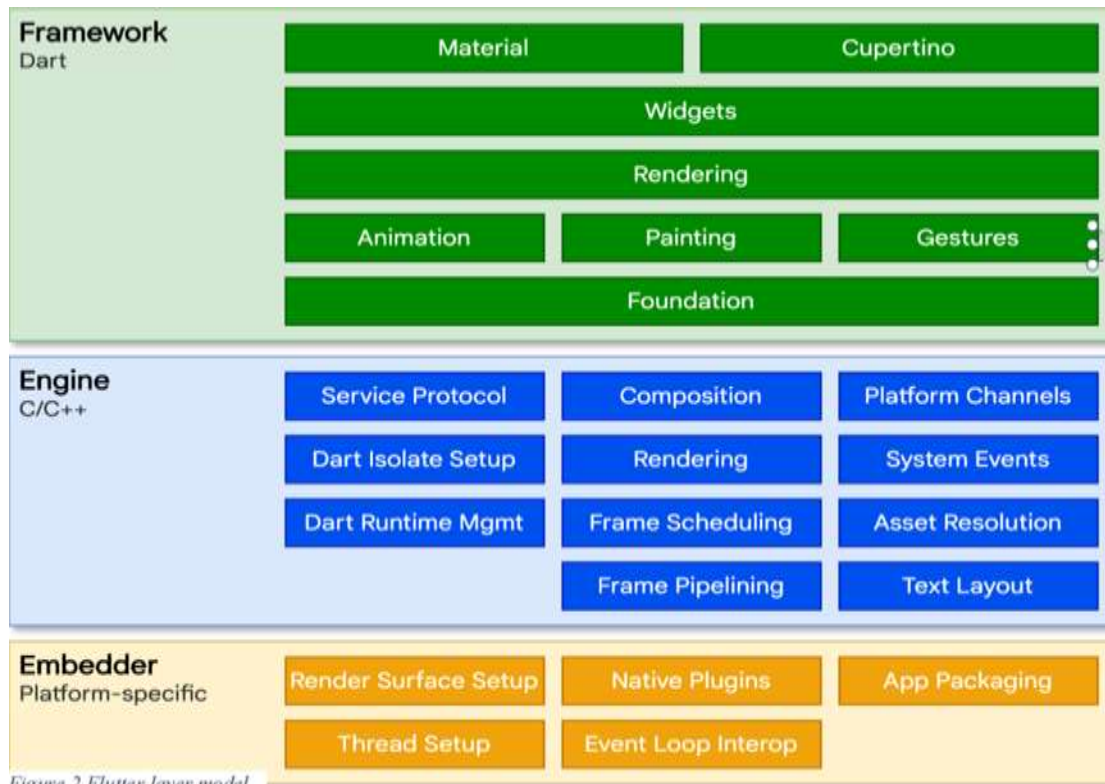
Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services. The goal is to enable developers to deliver high-performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible. During development, Flutter apps run in a VM that offers stateful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code, whether Intel x64 or ARM instructions, or to JavaScript if targeting the web. The framework is open source, with a permissive BSD license, and has a thriving ecosystem of third-party packages that supplement the core library functionality. This overview is divided into several sections:

- 1- The layer model: The pieces from which Flutter is constructed
- 2- Reactive user interfaces: A core concept for Flutter user interface development.
- 3- An introduction to widgets: The fundamental building blocks of Flutter user interfaces.
- 4- The rendering process: How Flutter turns UI code into pixels.
- 5- An overview of the platform embedders: The code that lets mobile and desktop OSes execute Flutter apps.
- 6- Integrating Flutter with other code: Information about different techniques available to Flutter apps.
- 7- Support for the web: Concluding remarks about the characteristics of Flutter in a browser environment.

1. The layer model: The pieces from which Flutter is constructed

Flutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer. No layer has

privileged access to the layer below, and every part of the framework level is designed to be To the underly
ing operating system, Flutter applications are



packaged in the same way as any other native application. A platform-specific embedder provides an entrypt, coordinates with the underlying operating system for access to serviceslike rendering surfaces, optional and replaceable. accessibility, and input; and manages the message event loop. The embedder is written in a language that is appropriate for the platform: currently Java and C++ forAndroid, Objective-C/Objective-C++ for iOS and macOS, and C++ for Windows and Linux. Using the embedder, Flutter code can be integrated into an existing application as a module, or the code may be the entire content of the application. Flutter includes a number of embedders for common target platforms, but other embedders also exist.

At the core of Flutter is the Flutter engine, which is mostly written in C++ and supports the primitives necessary to support all Flutter applications. The engine is responsible for rasterizing composited scenes whenever a new frame needs to be painted. It provides the low-level implementation of Flutter’s core API, including

graphics (through Skia), text layout, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile.

The engine is exposed to the Flutter framework through `dart:ui`, which wraps the underlying C++ code in Dart classes. This library exposes the lowest-level primitives, such as classes for driving input, graphics, and text rendering subsystems.

Typically, developers interact with Flutter through the Flutter framework, which provides a modern, reactive framework written in the Dart language.

It includes a rich set of platforms, layout, and foundational libraries, composed of a series of layers. Working from the bottom to the top, we have:

1-Basic foundational classes

- Building block services such as animation, painting, and gestures that offer commonly used abstractions over the underlying foundation.
- The rendering layer provides an abstraction for dealing with layout. With this layer, you can build a tree of renderable objects. You can manipulate these objects dynamically, with the tree automatically updating the layout to reflect your changes.
- The widgets layer is a composition abstraction. Each render object in the rendering layer has a corresponding class in the widgets layer. In addition, the widgets layer allows you to define combinations of classes that you can reuse. This is the layer at which the reactive programming model is introduced.
- The Material and Cupertino libraries offer comprehensive sets of controls that use the widget layer's composition primitives to implement the Material or iOS design languages.

The Flutter framework is relatively small; many higher-level features that developers

might use are implemented as packages, including platform plugins like camera and webview, as well as platform-agnostic features like characters, http, and animations that build upon the core Dart and Flutter libraries. Some of these packages come from the broader ecosystem, covering services like in-app payments, Apple authentication, and animations.

The rest of this overview broadly navigates down the layers, starting with the reactive paradigm of UI development. Then, we describe how widgets are composed together and converted into objects that can be rendered as part of an application. We describe how Flutter interoperates with other code at a platform level, before giving a brief summary of how Flutter's web support differs from other targets.

2-Reactive user interfaces

On the surface, Flutter is a reactive, pseudo-declarative UI framework, in which the developer provides a mapping from application state to interface state, and the framework takes on the task of updating the interface at runtime when the application state changes. This model is inspired by work that came from Facebook for their own

React framework, which includes a rethinking of many traditional design principles. In most traditional UI frameworks, the user interface's initial state is described once

and then separately updated by user code at runtime, in response to events. One challenge of this approach is that, as the application grows in complexity, the developer needs to be aware of how state changes cascade throughout the entire UI.

There are many places where the state can be changed: the color box, the hue slider, the radio buttons. As the user interacts with the UI, changes must be reflected in every other place. Worse, unless care is taken, a minor change to one part of the user interface can cause ripple effects to seemingly unrelated pieces of code.

One solution to this is an approach like MVC, where you push data changes to the model via the controller, and then the model pushes the new state to the view via the controller. However, this also is problematic, since creating and

updating UI elements are two separate steps that can easily get out of sync. Flutter, along with other reactive frameworks, takes an alternative approach to this problem, by explicitly decoupling the user interface from its underlying state. With React-style APIs, you only create the UI description, and the framework takes care of using that one configuration to both create and/or update the user interface as appropriate.

In Flutter, widgets (kind to components in React) are represented by immutable classes that are used to configure a tree of objects. These widgets are used to

manage a separate tree of objects for layout, which is then used to manage a separate tree of objects for compositing. Flutter is, at its core, a series of mechanisms for efficiently walking the modified parts of trees, converting trees of objects into lower-level trees of objects, and propagating changes across these trees.

A widget declares its user interface by overriding the `build ()` method, which is a function that converts state to UI.

3- Widgets

As mentioned, Flutter emphasizes widgets as a unit of composition. Widgets are the building blocks of a Flutter app's user interface, and each widget is an immutable declaration of part of the user interface.

Widgets form a hierarchy based on composition. Each widget nests inside its parent and can receive context from the parent. This structure carries all the way up to the root widget (the container that hosts the Flutter app typically, `MaterialApp` or `CupertinoApp`), as this trivial example shows:

List Example of a Widget in the preceding code, all instantiated classes are widgets. Apps update their user interface in response to events (such as a user interaction) by telling the framework to replace a widget in the hierarchy with another widget.

The framework then compares the new and old widgets, and efficiently updates the user interface. Flutter has its own implementations of each UI control, rather than deferring to those provided by the system: for example, there is a pure Dart implementation of both the iOS Switch control and the one for the Android equivalent.

This approach provides several benefits:

- **Rendering processing Provides for unlimited extensibility.** A developer who wants a variant of the Switch control can create one in any arbitrary way and is not limited to the extension points provided by the OS.
- **Avoids a significant performance bottleneck** by allowing Flutter to composite the entire scene at once, without transitioning back and forth between Flutter code and platform code.
- **Decouples the application behavior from any operating system dependencies.**

The application looks and feels the same on all versions of the OS, even if the OS changed the implementations of its controls.

List Building widgets

As mentioned earlier, you determine the visual representation of a widget By overriding the `build ()` function to return a new element tree. This tree represents the widget's part of the user interface in more concrete terms. For example, a toolbar widget might have a build function that returns a horizontal layout of some text and various buttons. As needed, the framework recursively asks each widget to build until the tree is entirely described by concrete renderable objects. The framework then stitches together the renderable objects into a renderable object tree.`

A widget's build function should be free of side effects. Whenever the function is asked to build, the widget should return a new tree of widgets¹, regardless of what the widget previously returned. The framework does the heavy lifting work to determine which build methods need to be called based on the render object tree. More information about this process can be found in the Inside Flutter topic. On each rendered frame, Flutter can recreate just the parts of the UI where the state has changed by calling that widget's `build ()` method. Therefore, it is important that build methods should return quickly, and heavy computational work should be done in some asynchronous manner and then stored as part of the state to be used by a build method.

While relatively naïve in approach, this automated comparison is quite effective, enabling high-performance, interactive apps. And the design of the build function simplifies your code by focusing on declaring what a widget is made of, rather than the complexities of updating the user interface from one state to another.

Widget state The framework introduces two major classes of widget: stateful and stateless widgets. Many widgets have no mutable state: they don't have any properties that change over time (for example, an icon or a label). These widgets subclass `Stateless Widget`.

However, if the unique characteristics of a widget need to change based on user interaction or other factors, that widget is stateful. For example, if a widget has a counter that increments whenever the user taps a button, then the value of the counter is the state for that widget. When that value changes, the widget needs to be rebuilt to update its part of the UI. These widgets.`

subclass `Stateful Widget`, and (because the widget itself is immutable) they store mutable state in a separate class that subclasses `State`. Stateful Widgets don't have a build method; instead, their user interface is built through their state object.

Whenever you mutate a `State` object (for example, by incrementing the counter),

you must call `setState()` to signal the framework to update the user interface by calling the State's `build` method again.

Having separate state and widget objects lets other widgets treat both stateless and stateful widgets in the same way, without being concerned about losing state.

Instead of needing to hold on to a child to preserve its state, the parent can create a new instance of the child at any time without losing the child's persistent state. The framework does all the work of finding and reusing existing state objects when appropriate.

4-Rendering processing:

Rendering and layout

This section describes the rendering pipeline, which is the series of steps that Flutter takes to convert a hierarchy of widgets into the actual pixels painted onto a screen.

Flutter's rendering model

You may be wondering: if Flutter is a cross-platform framework, then how can it offer comparable performance to single-platform frameworks?

It's useful to start by thinking about how traditional Android apps work. When drawing, you first call the Java code of the Android framework. The Android system libraries provide the components responsible for drawing themselves to a Canvas object, which Android can then render with Skia, a graphics engine written in C/C++

that calls the CPU or GPU to complete the drawing on the device.

Cross-platform frameworks typically work by creating an abstraction layer over the underlying native Android and iOS UI libraries, attempting to smooth out the inconsistencies of each platform representation. App code is often written in an interpreted language like JavaScript, which must in turn interact with the Java-based

Android or Objective-C-based iOS system libraries to display UI. All this adds overhead that can be significant, particularly where there is a lot of interaction between the UI and the app logic.

By contrast, Flutter minimizes those abstractions, bypassing the system UI widget libraries in favor of its own widget set. The Dart code that paints Flutter's visuals is compiled into native code, which uses Skia for rendering. Flutter also embeds its own copy of Skia as part of the engine, allowing the developer to upgrade their app to stay updated with the latest performance improvements even if the phone hasn't

been updated with a new Android version. The same is true for Flutter on other native platforms, such as iOS, Windows, or macOS. It would be a rare application that drew only a single widget. An important part of any UI framework is therefore the ability to efficiently lay out a hierarchy of widgets, determining the size and position of each element before they are rendered on the screen. The base class for every node in the render tree is Render Object, which defines an abstract model for layout and painting. This is extremely general: it does not commit to a fixed number of dimensions or even a Cartesian coordinate system. (Demonstrated by this example of a polar coordinate system). Each Render Object knows its parents but knows little about its children other than how to visit them and their constraints.

This provides Render Object with sufficient abstraction to be able to handle a variety of use cases.

During the build phase, Flutter creates or updates an object that inherits from Render Object for each Render Object Element in the element tree. Render Objects are primitive: Render Paragraph renders text, Render Image renders an image, and Render Transform applies a transformation before painting its child

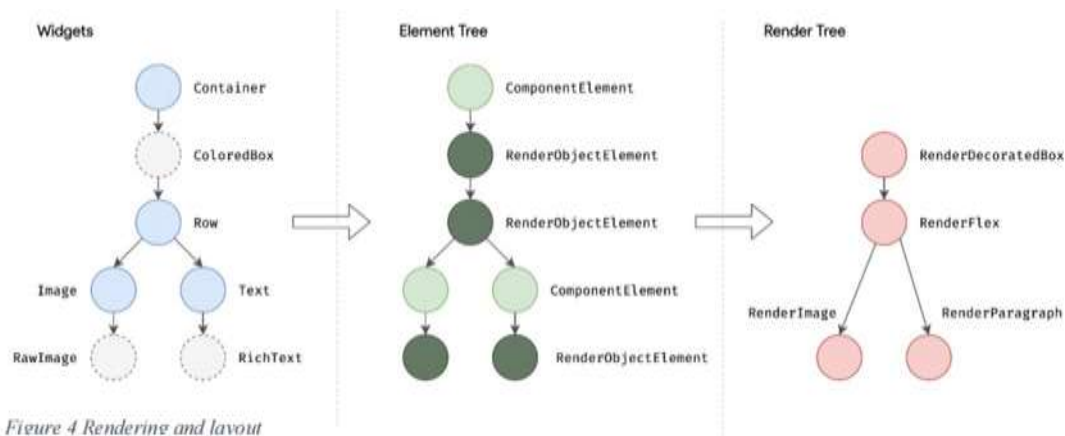


Figure 4 Rendering and layout

Rendering and layout.

Most Flutter widgets are rendered by an object that inherits from the Render Box subclass, which represents a Render Object of fixed size in a 2D Cartesian space. Render Box provides the basis of a box constraint model, establishing a minimum and maximum width and height for each widget to be rendered.

5- Platform embedding

As we've seen, rather than being translated into the equivalent OS widgets, Flutter user interfaces are built, laid out, composited, and painted by Flutter itself. The mechanism for obtaining the texture and participating in the app lifecycle of the underlying operating system inevitably varies depending on the unique concerns of that platform. The engine is platform-agnostic, presenting a stable ABI (Application Binary Interface) that provides a platform embedder with a way to set up and use Flutter.

The platform embedder is the native OS application that hosts all Flutter content, and acts as the glue between the host operating system and Flutter. When you start a Flutter app, the embedder provides the entrypoint, initializes the Flutter engine, obtains threads for UI and rastering, and creates a texture that Flutter can write to. The embedder is also responsible for the app lifecycle, including input gestures (such as mouse, keyboard, touch), window sizing, thread management, and platform messages. Flutter includes platform embedders for Android, iOS, Windows, macOS, and Linux; you can also create a custom platform embedder, as in this worked example that supports remotng Flutter sessions through a VNC-style framebuffer or this worked example for Raspberry Pi.

Each platform has its own set of APIs and constraints. Some brief platform-specific notes:

On iOS and macOS, Flutter is loaded into the embedder as a UI View Controller or NS View Controller, respectively. The platform embedder creates a Flutter Engine, which serves as a host to the Dart VM and your Flutter runtime,

and a Flutter View Controller, which attaches to the Flutter Engine to pass UIKit or Cocoa input events into Flutter and to display frames rendered by the Flutter Engine using Metal or OpenGL.

On Android, Flutter is, by default, loaded into the embedder as an Activity. The view is controlled by a [Flutter View](#), which renders Flutter content either as a view or a texture, depending on the composition and z-ordering requirements of the Flutter content.

- On Windows, Flutter is hosted in a traditional Win32 app, and content is rendered using [ANGLE](#), a library that translates OpenGL API calls to the DirectX 11 equivalents. Efforts are currently underway to also offer a Windows embedder using the UWP app model, as well as to replace

ANGLE with a more direct path, the

GPU via DirectX 12.

6-Integrating with other code

Flutter provides a variety of interoperability mechanisms, whether you're accessing code or APIs written in a language like Kotlin or Swift, calling a native C-based API, embedding native controls in a Flutter app, or embedding Flutter in an existing application.

Platform channels

For mobile and desktop apps, Flutter allows you to call into custom code through a platform channel, which is a mechanism for communicating between your Dart code and the platform-specific code of your host app. By creating a common channel (encapsulating a name and a codec), you can send and receive messages between Dart and a platform component written in a language like Kotlin or Swift. Data is serialized from a Dart type like Map into a standard format, and then deserialized into an equivalent representation in Kotlin (such as HashMap) or Swift (such as Dictionary)

Flutter web support

While the general architectural concepts apply to all platforms that Flutter supports, there are some unique characteristics of Flutter's web support that are worthy of comment.

Dart has been compiling to JavaScript for as long as the language has existed, with a toolchain optimized for both development and production purposes. Many important apps compile from

Dart to JavaScript and run in production today, including the advertiser tooling for Google Ads. Because the Flutter framework is written in Dart, compiling it to JavaScript was relatively straightforward.

However, the Flutter engine, written in C++, is designed to interface with the underlying operating system rather than a web browser. A different approach is therefore required. On the web, Flutter provides a reimplementation of the engine on top of standard browser APIs. We currently have two options for rendering Flutter content on the web: HTML and WebGL. In HTML mode, Flutter uses HTML, CSS,

Canvas, and SVG. To render to WebGL,

Flutter uses a version of Skia compiled to Web Assembly called Canvas Kit. While HTML mode offers the best code size characteristics, Canvas Kit provides the fastest path to the browser's graphics stack and offers somewhat higher graphical fidelity with the native mobile targets⁵.



Figure 5 Flutter Layer Model (Web Support)

The web version of the architectural layer diagram is as follows:

Flutter Layer Model (Web Support)

During development time, Flutter web uses `dartdevc`, a compiler that supports incremental compilation and therefore allows hot restart (although not currently hot reload) for apps. Conversely, when you are ready to create a production app for the web, `dart2js`, Dart's highly optimized production JavaScript compiler is used, packaging the Flutter core and framework along with your application into a minified source file that can be deployed to any web server. Code can be offered in a single file or split into multiple files through deferred imports.

2.9 Backend:

Firebase Authentication for User Registration and SignIn Introduction:

In this chapter, we explore the implementation of Firebase Authentication for user registration and sign-in in your skin cancer disease detection application. Firebase Authentication offers a seamless and secure solution for managing user identities and access control. We will discuss the benefits of using Firebase Authentication, the key features it provides, and step-by-step instructions on integrating it into your application.

Overview of Firebase Authentication:

Firebase Authentication is a service provided by Firebase that simplifies the process of authenticating users in your application. It supports a variety of authentication methods, including email/password, phone number, social media platforms (such as Google, Facebook, Twitter), and more. Firebase Authentication handles user registration, sign-in, and user management, allowing you to focus on building the core functionality of your application.

Setting Up Firebase Authentication:

To start using Firebase Authentication, you need to create a Firebase project and configure the authentication settings. We will guide you through the process of setting up Firebase Authentication in your project, including creating the project, enabling the authentication feature, and configuring the authentication providers you want to support.

User Registration:

User registration is a crucial step for creating user accounts in your application. We will explain how to implement the user registration process using Firebase Authentication's email/password authentication method. This involves creating a registration form, validating user input, securely storing user credentials in Firebase Authentication, and handling registration errors.

User Sign-In:

Once users have registered, they can sign in to access the features of your application. We will demonstrate how to implement the user sign-in process using Firebase Authentication. This includes creating a sign-in form, verifying user credentials, handling authentication errors, and managing user sessions.

Additional Authentication Methods:

Firebase Authentication offers various authentication methods beyond email/password. We will explore the integration of additional authentication methods, such as social media login using platforms like Google, Facebook, or Twitter. This allows users to sign in to your application using their existing social media accounts, simplifying the authentication process.

User Management and Security:

Firebase Authentication provides features for managing user accounts and enhancing security. We will discuss how to implement functionalities such as password reset, email verification, account deletion, and updating user profiles. Additionally, we will explore best practices for securing user data, implementing proper authorization rules, and protecting against common security threats.

Error Handling and Exception Handling:

During the authentication process, various errors and exceptions can occur. We will cover how to handle and display meaningful error messages to users when authentication fails. By providing clear and user-friendly error messages, you can enhance the user experience and help users troubleshoot authentication issues effectively.

Testing and Deployment:

Testing and debugging are critical stages in any application development process. We will provide guidelines on testing the authentication flows in your application to ensure smooth registration and sign-in experiences. Additionally, we will discuss considerations for deploying your application with Firebase Authentication, including security measures and production-ready configuration.

Conclusion:

Implementing Firebase Authentication for user registration and sign-in in your cucumber disease detection application offers numerous advantages, such as simplified authentication workflows, secure user management, and support for various authentication methods. By following the guidelines and instructions outlined in this chapter, you can successfully integrate Firebase Authentication into your application, providing users with a seamless and secure authentication experience.

Chapter 3: Related Work & Related Applications

3.1 Introduction:

In this chapter, we will explore a range of important research papers and studies that focus specifically on skin cancer. These research papers contributed to the understanding, diagnosis and management of various types of skin cancer. By studying these papers, we will gain valuable insights into the identification, characterization, epidemiology and control strategies of skin cancer. This comprehensive overview of current references will highlight the current state of knowledge and developments in the field of skin cancer research.

1- HarshBhatt

This research paper focuses on the use of advanced machine learning techniques in identifying and classifying melanoma skin cancer. Modern methods and techniques in the field of machine learning and their applications in accurate diagnosis and classification of skin cancer are reviewed. The study highlights the use of techniques such as deep neural networks, deep learning and self-classification to improve diagnostic ability and achieve better accuracy in detecting and classifying different types of skin cancer. This study provides a comprehensive understanding of current techniques and advances in machine learning for skin cancer detection and classification, with the aim of developing more precise and personalized management strategies for patients with skin cancer.

2- A.Shah :

This research Focuses on the role of machine learning and deep learning techniques in skin cancer detection. The research aims to review how these technologies can be used to improve diagnosis and detection of skin cancer. Machine learning and deep learning techniques are used to analyze skin images and identify distinct signs of cancerous tumors. Deep models are trained on a large set of previously diagnosed skin images to recognize patterns and similarities that indicate the presence of skin cancer. Studies show that the use of machine learning and deep learning techniques can improve the accuracy of diagnosis and increase the effectiveness of early detection of skin cancer. These technologies can help identify suspected cases more accurately and direct people to the necessary tests to confirm the diagnosis.

3- Kumar

This research focuses on reviewing recent research that uses deep learning techniques to detect skin cancer. Skin cancer is one of the most common types of cancer, and its early diagnosis is crucial for effective treatment.

Research papers that rely on deep learning techniques such as deep neural networks, classification techniques, and feature extraction to improve the accuracy of skin cancer diagnosis are reviewed. The performance of the models used in research is studied and

compared with traditional methods for diagnosing skin cancer. Challenges facing the use of deep learning techniques in this field are also highlighted, such as the need for large amounts of data and the availability of clinical data for training and testing. The results show that using deep learning techniques to detect skin cancer can contribute to improving the accuracy of diagnosis and distinguishing malignant tumors from benign ones. Future directions for research and performance improvement of deep learning models in the field of skin cancer detection are also discussed.

3.2 Methods of detecting skin cancer

By searching through Google Scholar and scientific articles, we have come to the most successful research

- The authors, Harsh Bhatt, Vrunda Shah et al used The “ISIC(2018)/HAM10” dataset, which consists of more than 11,527 images for 8 Types of skin cancer we have reached the level of accuracy 97.70% were using CNNs
- The authors A. Shah, A. Pandya et al used The “ISIC” dataset, we have reached the level of accuracy 92% were using CNNs
- The authors Kumar & Vatsa et al used The “ISIC” dataset, which consists of more than 25,331 images we have reached the level of accuracy 89.6% were using CNN

3.3 Related Application

3.3.1 MoleScope :



Molescope™ is a smartphone application by MetaOptima that aims to reduce the diagnostic time thanks to an application that is able to detect melanomas by tracking moles.

Advantages:

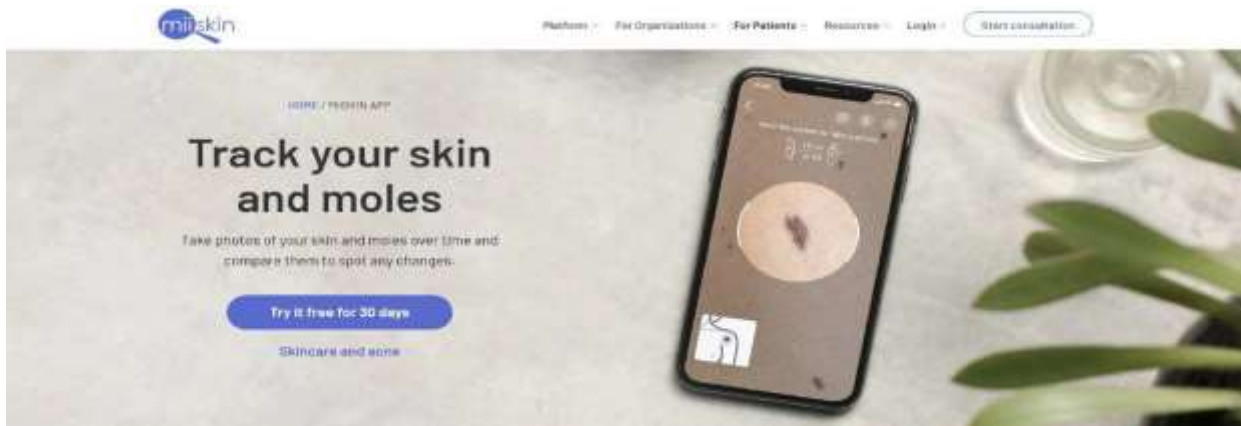
1. Convenience: MoleScope allows users to monitor their moles conveniently from their smartphones, without the need for frequent visits to a dermatologist. This can be particularly useful for individuals with busy schedules or limited access to healthcare services.
2. Early Detection: By regularly monitoring moles and skin changes, MoleScope users may be able to detect potential skin cancer at an early stage. Early detection often leads to better treatment outcomes and prognosis.

3. Educational Resources: The app may provide educational resources and information about skin cancer, helping users understand risk factors, warning signs, and preventive measures.
4. Privacy: Users have control over their own data and images, which may offer a level of privacy and discretion that some individuals prefer over traditional medical examinations.
5. User-Friendly Interface: MoleScope typically features a user-friendly interface, making it accessible to a wide range of users, including those who may not be tech-savvy.

Disadvantages:

1. Accuracy Concerns: While MoleScope can be a helpful tool, it may not always provide accurate assessments of moles or skin conditions. False positives or false negatives are possible, which could lead to unnecessary anxiety or delayed medical treatment.
2. Limited Functionality: The app's capabilities may be limited compared to in-person dermatologist examinations. It may not detect all types of skin abnormalities or provide the comprehensive evaluation that a trained medical professional can offer.
3. Dependence on Technology: Relying solely on an app for skin health monitoring may lead to complacency or a false sense of security. Users should still prioritize regular check-ups with dermatologists for a thorough evaluation.
4. Cost: While the app itself may be free or low-cost, additional services or features may require payment. This could be a barrier for some users, particularly those with limited financial resources.
5. Digital Security Concerns: Transmitting sensitive medical information and images through a mobile app raises concerns about data privacy and security. Users should ensure that the app complies with relevant privacy regulations and employs robust security measures to protect their information.

3.3.2 Miiskin :



Miiskin is a mobile application designed to empower users in monitoring their skin health, with a specific focus on tracking moles and potential indicators of skin cancer.

Advantages:

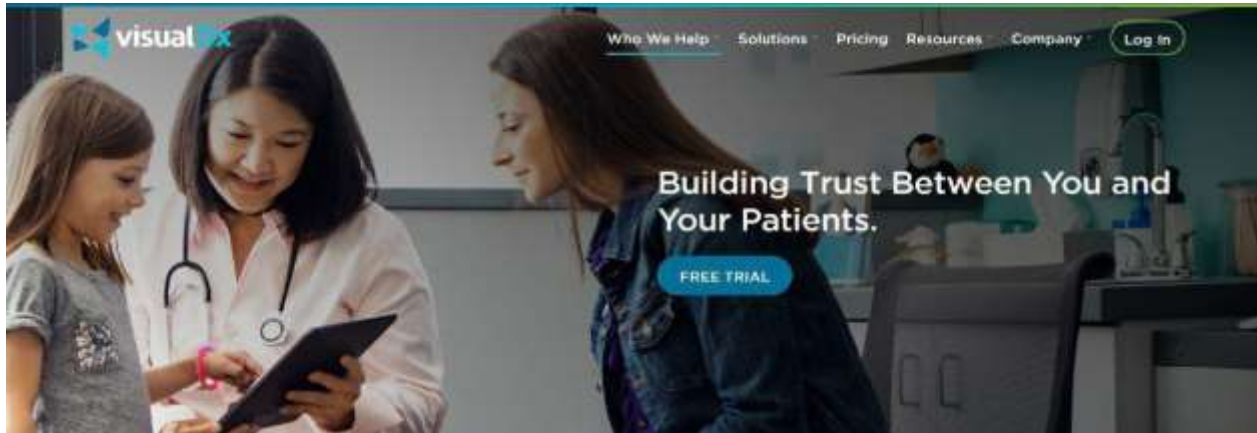
1. **Convenience:** Miiskin provides users with the convenience of monitoring their skin health directly from their smartphones. This accessibility allows for easy and regular tracking of changes in moles and skin conditions.
2. **Early Detection:** Regular use of Miiskin enables users to detect any changes in their moles or skin early on. This early detection can be crucial for identifying potential signs of skin cancer or other skin conditions, leading to prompt medical attention and improved treatment outcomes.
3. **Education and Awareness:** The app offers educational resources and information about skin cancer awareness, risk factors, and preventive measures. This empowers users to become more knowledgeable about their skin health and encourages proactive measures for early detection and prevention.

4. **User-Friendly Interface:** Miiskin typically features a user-friendly interface, making it easy for users to navigate and utilize its various functionalities. This simplicity encourages consistent use and engagement, enhancing the effectiveness of skin monitoring.
5. **Privacy and Security:** Miiskin prioritizes user privacy and data security, allowing users to maintain control over their personal health information and images. This aspect fosters trust and confidence among users, ensuring their sensitive data is handled securely.

Disadvantages:

1. **Accuracy Concerns:** While Miiskin serves as a helpful tool for skin health monitoring, its assessments may not always be completely accurate. False positives or false negatives can occur, potentially leading to unnecessary concern or missed detection of significant changes.
2. **Limited Diagnostic Capability:** The app's diagnostic capabilities are limited compared to professional medical evaluations by dermatologists. It may not detect all types of skin abnormalities or provide a comprehensive assessment of skin conditions, necessitating professional medical consultations for accurate diagnosis and treatment.
3. **Dependency on Technology:** Relying solely on a mobile app for skin monitoring may foster a false sense of security and discourage users from seeking regular medical check-ups. It's important for users to supplement app usage with professional dermatological assessments for thorough evaluation and personalized care.
4. **Cost and Subscription Models:** While Miiskin may offer a free basic version, additional features or premium services may require payment or subscription. This cost factor could be a barrier for some users, especially those with limited financial resources.
5. **Data Privacy Risks:** Transmitting sensitive medical information and images through a mobile app poses potential data privacy risks. Users should ensure that Miiskin complies with relevant privacy regulations and employs robust security measures to safeguard their personal health data.

3.3.3 VisualDx :



VisualDx is a diagnostic clinical decision support system designed to aid healthcare professionals in diagnosing and treating various dermatological conditions.

Advantages:

1. **Comprehensive Database:** VisualDx boasts an extensive database of medical images covering a wide range of dermatological conditions. This comprehensive library allows healthcare professionals to visually compare patient symptoms to known conditions, aiding in accurate diagnosis.
2. **Diagnostic Assistance:** The platform provides diagnostic support by allowing healthcare professionals to input patient symptoms and characteristics, such as skin lesions or rashes, to generate a list of potential diagnoses. This helps clinicians consider a broader range of possibilities and make more informed decisions.
3. **Educational Resource:** VisualDx serves as an educational tool by providing access to detailed information about various dermatological conditions, including symptoms, differential diagnoses, and treatment options. This can help healthcare professionals enhance their knowledge and understanding of dermatology.

4. **Visual Aid for Patient Education:** VisualDx enables healthcare professionals to visually demonstrate dermatological conditions to patients, facilitating better understanding and engagement in their own healthcare. This visual aid can improve patient communication and compliance with treatment plans.
5. **Time Efficiency:** By providing quick access to a vast database of medical images and diagnostic support tools, VisualDx can help healthcare professionals save time in the diagnostic process. This efficiency can lead to faster diagnoses and more timely treatment for patients.

Disadvantages:

1. **Subscription Cost:** VisualDx typically requires a subscription or institutional access, which can be costly for individual healthcare professionals or smaller practices. This cost may serve as a barrier to adoption for some users.
2. **Dependency on Visual Information:** VisualDx relies heavily on visual information provided by healthcare professionals, such as photographs of patient symptoms. While this can be useful, it may not always capture the full clinical context or nuances of a patient's condition, potentially leading to incomplete or inaccurate diagnoses.
3. **Limited Scope of Practice:** While VisualDx covers a wide range of dermatological conditions, it may not be as comprehensive or suitable for diagnosing conditions outside of dermatology. Healthcare professionals should be cautious about relying solely on VisualDx for diagnoses in other medical specialties.
4. **Technical Challenges:** Like any software platform, VisualDx may encounter technical issues such as glitches or downtime, which can disrupt workflow and impact the user experience.
5. **Risk of Overreliance:** There is a risk that healthcare professionals may become overly reliant on VisualDx as a diagnostic tool .

3.3.4 Dermatology A to Z :



Dermatology A to Z is a simple mobile app that offers lots of info about skin problems. It helps you learn about different skin conditions like rashes or acne.

Advantages:

1. **Comprehensive Information:** The application provides comprehensive information about a wide range of dermatological conditions and diseases, enabling users to understand different conditions and their symptoms.
2. **Ease of Access:** Users can access information anytime and anywhere through the application, making it a convenient reference for dermatological medical information.
3. **Patient Education:** The application can help in educating patients about their dermatological conditions, empowering them to utilize the knowledge to better engage with healthcare providers.

Disadvantages:

1. **Limited Interactivity:** Unlike direct interaction with a healthcare provider, the application may lack the ability to provide personalized advice or address specific concerns of individual users.
2. **Reliance on Self-Diagnosis:** There's a risk that users may rely solely on the information provided by the application for self-diagnosis, which can be inaccurate and potentially harmful.
3. **Lack of Real-time Assistance:** In cases where immediate medical attention or consultation is required, the application may not provide real-time assistance or support.

Chapter 4: Skin Cancer Diseases

4.1 Melanoma:

What is melanoma?

Melanoma, which means "black tumor," is the most dangerous type of skin cancer. It grows quickly and has the ability to spread to any organ.

Melanoma comes from skin cells called melanocytes. These cells produce melanin, the dark pigment that gives skin its color. Most melanomas are black or brown in color, but some are pink, red, purple or skin-colored.

About 30% of melanomas begin in existing moles, but the rest start in normal skin. This makes it especially important to pay attention to changes in your skin because the majority of melanomas don't start as moles. However, how many moles you have may help predict your skin's risk for developing melanoma. It's important to know if you're in a high-risk group for developing melanoma skin cancer. Because of the fast growth rate of melanomas, a treatment delay sometimes may mean the difference between life and death. Knowing your risk can help you be extra vigilant in watching changes in your skin and seeking skin examinations since melanomas have a 99% cure rate if caught in the earliest stages. Early detection is important because treatment success is directly related to the depth of the cancerous growth.

What are the symptoms of melanoma?

The first melanoma signs and symptoms often are:

A change in an existing mole.

The development of a new pigmented or unusual-looking growth on the skin. Melanoma doesn't always begin as a mole. It also can happen on otherwise healthy skin.

Melanomas symptoms can happen anywhere on the body. Melanomas most often develop in areas that have had exposure to the sun. This includes the arms, back, face and legs.

Melanomas also can happen in areas that aren't as exposed to the sun. This includes the soles of the feet, palms of the hands and fingernail beds. Melanoma also can happen inside the body. These hidden melanomas are more common in people with brown or Black skin.

Asymmetry: One half does not match the other half.

Border: The edges are not smooth.

Color: The color is mottled and uneven, with shades of brown, black, gray, red or white.

Diameter: The spot is greater than the tip of a pencil eraser (6.0 mm).

Size : a mole may appear to get bigger

Elevation :the mole may develop a raised area

itching or bleeding.

How is melanoma treated?

Your melanoma treatment will depend on the stage of the melanoma and your general health.

Surgery is usually the main treatment for melanoma. The procedure involves cutting out the cancer and some of the normal skin surrounding it. The amount of healthy skin removed will depend on the size and location of the skin cancer.

Typically, surgical excision (removal) of melanoma can be performed under local anesthesia in the dermatologist's office. More advanced cases may require other types of treatment in addition to or instead of surgery.

Treatments for melanoma:

Melanoma Surgery: In the early stages, surgery has a high probability of being able to cure your melanoma. Usually performed in an office, a dermatologist numbs the skin with a local anesthetic and removes the melanoma and margins (healthy surrounding skin).

Lymphadenectomy: In cases where melanoma has spread, removal of the lymph nodes near the primary diagnosis site may be required. This can prevent the spread to other areas of your body.

Metastasectomy: Metastasectomy is used to remove small melanoma bits from organs.

Targeted cancer therapy: In this treatment option, drugs are used to attack specific cancer cells. This “targeted” approach goes after cancer cells, leaving healthy cells untouched.

Radiation Therapy: Radiation therapy includes treatments with high-energy rays to attack cancer cells and shrink tumors.

Immunotherapy: immunotherapy stimulates your own immune system to help fight the cancer

4.2 Melanocytic naevus:

What is a melanocytic naevus?

A melanocytic naevus (American spelling 'nevus'), or mole, is a common precancerous skin lesion due to a local proliferation of pigment cells (melanocytes). It is sometimes called a naevocytic naevus or just 'naevus' (but note that there are other types of naevi). A brown or black melanocytic naevus contains the pigment melanin, so may also be called a pigmented naevus.

A melanocytic naevus can be present at birth (a congenital melanocytic naevus) or appear later (an acquired naevus). There are various kinds of congenital and acquired melanocytic naevi (American spelling 'nevi').

Melanocytic naevus symptoms :

Tan to dark brown, pale pink, and occasionally black in color
Round or oval in shape
Smooth borders

Uniform color throughout

Symmetry (when a line is drawn within them, the two halves have the same appearance)

What is the treatment for melanocytic naevus?

Most melanocytic naevi are harmless and can be safely left alone. They may be removed in the following circumstances:

To exclude cancer

If a naevus is a nuisance: perhaps irritated by clothing, comb or razor
Cosmetic reasons: the mole is unsightly.

Surgical techniques include:

Excision biopsy of a flat or suspicious melanocytic naevus

Shave biopsy of a protruding melanocytic naevus

Electrosurgical destruction

Laser to lessen pigment or remove coarse hair.

4.3 Actinic keratosis

What is actinic keratosis?

Actinic keratosis (AK) is a skin disorder that causes rough, scaly patches of skin. Another name for AK is solar keratosis. AK is a type of precancer, which means that if you don't treat the condition, it could turn into cancer. Without treatment, AK can lead to a type of skin cancer called squamous cell carcinoma.

Actinic keratosis Symptoms

- Rough, dry or scaly patch of skin, usually less than 1 inch (2.5 centimeters) in diameter
- Flat to slightly raised patch or bump on the top layer of skin
- In some cases, a hard, wartlike surface
- Color variations, including pink, red or brown
- Itching, burning, bleeding or crusting

How is actinic keratosis treated ?

Treatment options depend on how many actinic keratoses (AKs) you have and what they look like. Your healthcare provider may recommend removing the skin patches during an office visit.

To remove actinic keratosis, your provider may use:

Chemical peels: A chemical peel is like a medical-grade face mask. Your healthcare applies the peel during an office visit.

The chemicals in the treatment safely destroy unwanted patches in your top layer of skin. In the first.

Few days, the treated area will be sore and red. As the skin heals, you will see a new, healthy layer of skin. **Cryotherapy:** If you have one or two AKs, your provider may use cryotherapy. During this treatment, your provider uses a cold substance such as liquid

Within a few days, these growths will blister and peel off.

Excision: During this treatment, your healthcare provider first numbs skin around your AK. Your provider then scrapes away or cuts out the AKs and stitches the area back together. Usually, you would heal in two to three weeks.

Photodynamic therapy: If you have multiple AKs or AKs that return after treatment, your provider may recommend photodynamic therapy. This treatment uses creams and special light therapy to destroy precancerous skin cells. You will need to stay out of the sun for a few days while the treated skin heals.

4.4 Basal cell carcinoma

What is basal cell carcinoma?

Basal cell carcinoma (BCC) is a type of skin cancer that forms in the basal cells of your skin. Basal cells exist in the lower part of your epidermis, which is the outside layer of your skin. Basal cell carcinoma looks like a small, sometimes shiny bump or scaly flat patch on your skin that slowly grows over time.

What are the symptoms of basal cell carcinoma?

- Lumps, bumps, pimples, scabs or scaly lesions on your skin.
- The lump may appear shinier than the skin around it with tiny visible blood vessels.
- The lump may be slightly see-through (translucent) and close to your normal skin color or white to pink, brown to black or black to blue.
- The lump may grow slowly over time.
- The lump may be itchy or painful.

How is basal cell carcinoma treated?

Your provider will treat basal cell carcinoma by removing cancer from your body. To remove cancer, your treatment options could include:

Electrodessication and curettage:

Scraping off the cancerous lump with a curette and then burning with a special electric needle.

Surgery:

Removing the cancerous lump or lesion with a scalpel (excision or Mohs surgery).

Cryotherapy or cryosurgery:

Freezing the cancerous lump to remove it.

Chemotherapy:

Using powerful medicines to kill cancerous cells in your body.

Photodynamic therapy (PDT):

Applying blue light and a light-sensitive agent to your skin.

Laser therapy:

Using lasers (high-energy beams) to remove cancer instead of using a scalpel.

4.5 Seborrheic keratosis:

What is seborrheic keratosis?

A seborrheic keratosis (seb-o-REE-ik ker-uh-TOE-sis) is a common precancerous skin growth, similar to a mole. Most people will have at least one in their lifetime.

They tend to appear in mid-adulthood and their frequency increases with age. They are harmless and don't require treatment, but you can have them removed if they bother you.

Skin growths like seborrheic keratoses are sometimes also called epidermal tumors. That doesn't mean they're cancer, though. Technically, moles and warts are also epidermal tumors. That just means they are clusters of extra cells on the epiderma, the outer layer of the skin. They aren't considered a risk factor for skin cancer.

seborrheic keratosis symptoms

A seborrheic keratosis grows gradually. Signs and symptoms might include:

- A round or oval-shaped waxy or rough bump, typically on the face, chest, a shoulder or the back
- A flat growth or a slightly raised bump with a scaly surface, with a characteristic "pasted on" look
- Varied size, from very small to more than 1 inch (2.5 centimeters) across

- Varied number, ranging from a single growth to multiple growths
- Very small growths clustered around the eyes or elsewhere on the face, sometimes called flesh moles or dermatosis papulosa nigra, common on Black or brown skin
- Varied in color, ranging from light tan to brown or black
- Itchiness

How is seborrheic keratosis treated?

Medical offices offer several options for removing your seborrheic keratosis:

Cryotherapy: Your healthcare provider will numb the skin and then use liquid nitrogen to freeze the growth. This will cause it to fall off within a few days or weeks.

Electrodesiccation/Curettage: Your healthcare provider will numb the skin and then use a targeted electrocurrent to burn the seborrheic keratosis.

Shave Excision: This is the preferred method when your healthcare provider wants to preserve a sample of the growth to analyze in the lab.

Laser Therapy: Lasers offer an alternative to surgery by burning the growth, sterilizing the wound and sealing the tissue all at once.

Prescription Hydrogen Peroxide: The FDA has recently approved a topical solution of 40% hydrogen peroxide to treat seborrheic keratosis.

4.6 Dermatofibroma:

What is a dermatofibroma?

A dermatofibroma is a common precancerous fibrous nodule usually found on the skin of the lower legs.

A dermatofibroma is also called a cutaneous fibrous histiocytoma.

What are the symptoms of dermatofibroma?

A dermatofibroma usually presents as a solitary firm papule or nodule on a limb.

- A dermatofibroma can occur anywhere on the skin.
- Dermatofibroma size varies from 0.5–1.5 cm diameter; most lesions are 7–10 mm diameter.
- A dermatofibroma is tethered to the skin surface and mobile over subcutaneous tissue.
- Colour may be pink to light brown in white skin, and dark brown to black in dark skin; some appear paler in the centre.
- Dermatofibromas do not usually cause symptoms, but they are sometimes painful, tender, or itchy.

What is the treatment for dermatofibroma?

- Cryotherapy
- shave biopsy
- laser treatments

4.7 Pyogenic granuloma:

What is a pyogenic granuloma?

A pyogenic granuloma (granuloma pyogenicum) is a benign, raised tumor on your skin or mucous membranes. Pyogenic granulomas tend to ooze, and they break and bleed easily.

The name pyogenic granuloma is actually inaccurate. Pyogenic means pus-producing, and a granuloma is a cluster of white blood cells reacting to infection, causing a lump.

But pyogenic granulomas are rarely related to infection, and they don't generally contain white blood cells or pus.

The condition is more accurately called lobular capillary hemangioma, a tumor consisting of abnormal blood vessels

What are a pyogenic granuloma symptoms ?

- A pyogenic granuloma starts as a small, fleshy bump protruding from your skin or mucous membranes. It usually grows quickly, from a few millimeters (the tip of a crayon) to about a half-inch (the tip of a finger).
- They may be pink, red, reddish-brown or purple. They often develop a scaly, white "collar" around the bottom
- At maturity, the growths are often attached to your skin by a stalk-like structure But they can also attach directly to your skin
- The surface of a pyogenic granuloma starts smooth but can become bumpy or crusty.

What is the treatment for pyogenic granuloma?

Your healthcare provider may recommend a medication or procedure to treat pyogenic granulomas.

Topical medications applied to your skin to shrink pyogenic granulomas include:

- Chemicals such as silver nitrate, phenol and trichloroacetic acid (TCA).
- Steroid injections into the lesion
- Cryotherapy, to freeze it away.
- Curettage, to scrape it away, and cautery, to seal the skin with heat.
- Laser treatment to destroy the abnormal tissue.
- Surgical excision, to cut the granuloma out of your skin

4.8 Squamous Cell Carcinoma

What Is Squamous Cell Carcinoma?

Squamous cell carcinoma (SCC) is a very common type of skin cancer that's linked to sun damage. You can get it anywhere that you have skin.

Squamous cells are flat cells that make up the outer layer of your skin. Your body constantly sheds old ones and makes new ones. But if these squamous cells mutate and start to build up, a tumor can form.

SCC is a fairly slow-growing skin cancer. When caught early, it's easy to treat. Unlike some other types of skin cancer, it can spread to your tissues, bones, and nearby lymph nodes over time. If so, treatment can become more complex

Squamous cell carcinoma is a common form of skin cancer. It tends to appear on places that are exposed to UV rays like your head, neck, chest, upper back, ears, lips, arms, legs, and hands. (Photo Credit: iStock/Getty Images)

Types of squamous cell carcinoma

Doctors classify SCC based on where it is in your body.

In situ: This means it's only in the top layer of your skin.

Cutaneous: SCC has been found in other layers of your skin.

Metastatic: If you have this type, SCC has spread from your skin to other parts of your body. While rare, this can happen if SCC isn't found early or goes untreated.

Squamous Cell Carcinoma Symptoms:

If you sunburn easily, SCC is more likely to show up on parts of your body that have been exposed to ultraviolet (UV) rays from the sun or tanning beds, like your head, neck, chest, upper back, ears, lips, arms, legs, and hands. If you have dark skin, SCC is more likely to be found on other parts of your body that aren't in the sun, like the bottoms of your feet or on your penis or vagina.

According to the American Academy of Dermatology, SCC can show up as:

A rough, scaly, red patch of skin

An open sore (with or without raised edges)

A brown spot that looks like a typical age spot

A wart-like growth

A very small horn-like growth

A sore growing in an old scar

A smooth or sore reddish-white patch inside your mouth

A red, brown, or black line beneath a nail

A fingernail or toenail that seems to be shrinking

Although SCC is often red or pink, it can also be:

- Purple
- Gray
- Brown or black (or flecked with these colors)
- Yellow
- White (if it's inside your mouth)

Skin cancer can also show up differently on different types of skin. For instance:

In Asian Americans, it often appears as a round, raised, brown or black growth.

In African Americans, skin cancer often affects areas that don't get sun, like your palms, feet, lower legs, anus, genitals, or nails.

In Latino or Hispanic people, skin cancer typically starts as a skin growth that keeps getting bigger. Or you could have a scaly patch or notice a dark line under or around a nail.

How is squamous cell carcinoma treated?

Treatment for squamous cell carcinoma focuses on removing cancer from your body. Your treatment options vary based on the size, shape and location of your cancer and could include:

Cryosurgery: Freezing the cancer cells to destroy them.

Photodynamic therapy (PDT): Using blue light and light-sensitive agents to remove cancer from your skin.

Curettage and electrodesiccation: Scratching off the cancerous lump with a spoon-like instrument (curette), then burning the area with an electric needle.

Excision: Cutting the cancer out of your skin and stitching your skin back together.

Mohs surgery: Removing layers of skin affected by cancer, most common for facial cancers.

Systemic chemotherapy: Using powerful medicines to destroy cancer cells in your body.

What medications treat squamous cell carcinoma?

If you have invasive squamous cell carcinoma or if treatment to remove your cancer surgically isn't right for you, your healthcare provider could offer medicine to treat your diagnosis. Medicines could include:

Skin creams containing imiquimod or 5-fluorouracil help treat squamous cell carcinoma that's in the top layer of your skin (epidermis).

Cemiplimab-rwlc (Libtayo®) is immunotherapy to treat advanced forms of squamous cell carcinoma.

Pembrolizumab (Keytruda®) is immunotherapy to treat squamous cell carcinoma that isn't treatable with surgery.

4.9 Mycosis Fungoides

What Is Mycosis Fungoides?

Mycosis fungoides is a rare type of cutaneous T-cell lymphoma, which is a form of - It is a slow-growing cancer that primarily affects the skin. It starts with red, scaly patches or plaques on the skin that can resemble other common skin conditions.

Over time, the patches may become larger, thicker, and more tumor-like. Advanced stages can involve internal organs like the lymph nodes, spleen, and blood.

- The cause is not fully understood, but it is thought to involve a genetic mutation in T-cells that leads to their uncontrolled growth and spread.
- Symptoms include the skin lesions, itching, redness, and thickening of the affected skin areas.
- Diagnosis involves skin biopsies to confirm the presence of malignant T-cells.
- Treatment options include topical therapies, phototherapy, chemotherapy, radiation, and immunotherapy, depending on the stage and extent of disease.
- Mycosis fungoides is considered a chronic and incurable disease, but many patients can live for years or even decades with proper management and treatment.
- It is a relatively rare disease, affecting around 4 in 1 million people per year in the United States.

Chapter 5: Proposed Solution

5.1 Introduction:

- Customized Strategies:

Our solution will be tailored to the specific needs and requirements of our stakeholders. We understand that every situation is unique, and we will develop strategies that align with the goals and aspirations of our clients.

- Technology Integration:

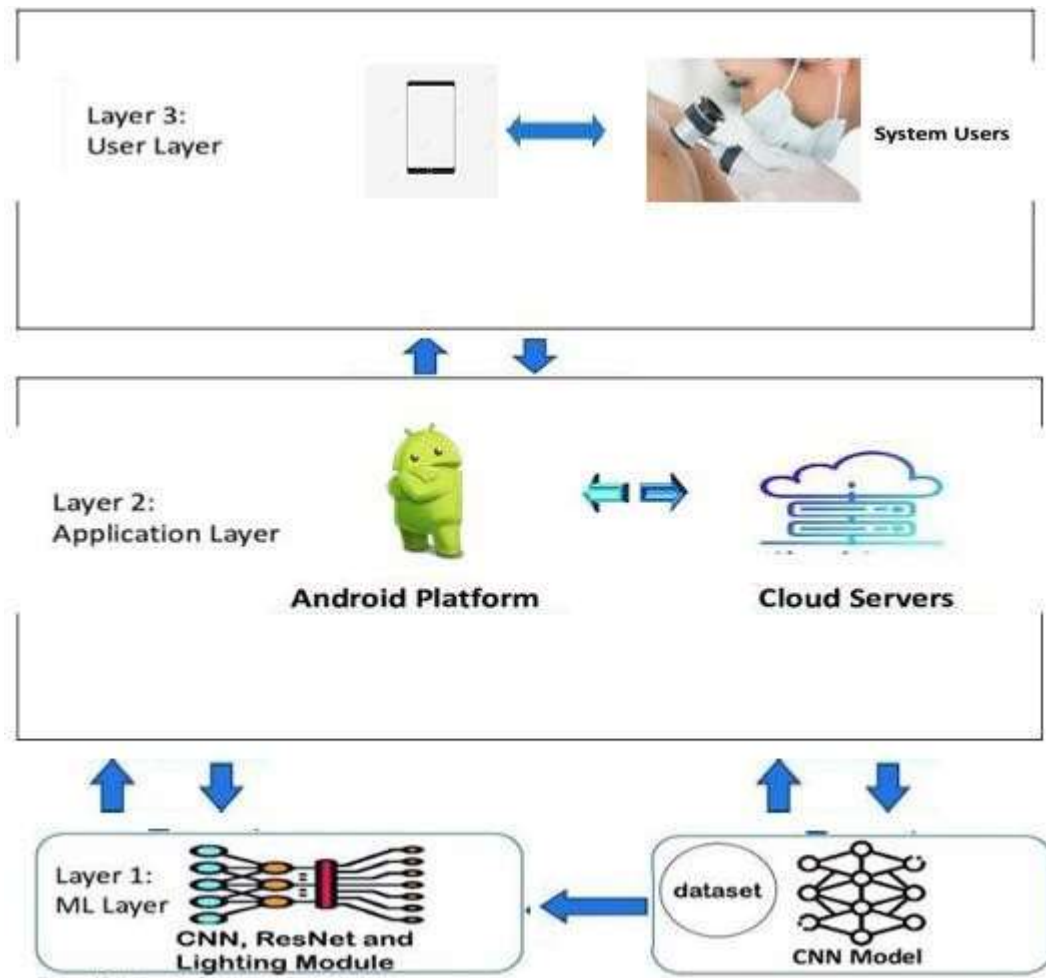
Leveraging the latest advancements in deep learning and artificial intelligence, we will incorporate innovative solutions to enhance efficiency, streamline processes, and maximize results in diagnosing skin cancer.

- Collaborative Partnerships:

We believe in the power of collaboration and will actively seek partnerships with key stakeholders, industry experts, and community leaders. By working together, we can pool resources, share knowledge, and achieve a collective impact.

- Monitoring and Evaluation:

We will establish robust monitoring and evaluation mechanisms to track the progress of our solution implementation. This will ensure transparency, accountability, and continuous improvement throughout the project's lifecycle.



5.2 System Architecture:

The system architecture for skin cancer detection using machine learning and image recognition involves several components working together to provide an accurate and efficient solution. Here is a high-level overview of the system architecture:

1- Data Collection:

The first step is to gather a diverse and comprehensive dataset of skin images. This dataset should include images of healthy skin as well as skin affected by various types of cancer. The dataset serves as the foundation for training the machine learning model.

2-Image Preprocessing:

Once the images are collected, preprocessing techniques are applied to enhance the quality and consistency of the images. Preprocessing may involve operations such as resizing, normalization, and noise reduction to ensure that the images are suitable for analysis.

3-Feature Extraction:

Preprocessed images are then subjected to feature extraction techniques to identify key characteristics and patterns. This step helps in transforming raw image data into a format that can be easily analyzed by the machine learning model.

4- Machine Learning Model Training:

The extracted features are used to train a machine learning model, such as a convolutional neural network (CNN). The model learns to recognize patterns and features associated with different types of skin cancer. Training involves feeding the model with labeled images, allowing it to adjust its internal parameters to accurately classify and identify diseases.

5-Model Evaluation and Validation:

The trained model is evaluated using a separate set of validation images to assess its performance and ensure its generalization capability. This step helps identify any potential issues, such as overfitting or underfitting, and fine-tune the model accordingly.

6-User Interface:

The system incorporates a user interface that allows users (patients or dermatologists) to interact with the disease detection system.

The user interface can be a mobile application or a web-based platform. It provides options for capturing images using the device's camera or uploading images from the gallery.

7- Disease Detection:

When a new image is provided by the user, the image is passed through the trained machine learning model. The model analyzes the image and predicts whether the skin is healthy or affected by a specific type of cancer. The result is displayed to the user, indicating the detected cancer type and its severity if applicable.

8- Recommendations and Remedial Measures:

Along with disease detection, the system may provide recommendations and remedial measures based on the identified type of cancer. This information can include suggested treatments, follow-up actions, or referrals to specialists to help users effectively manage their condition.

5.3 Main feature:

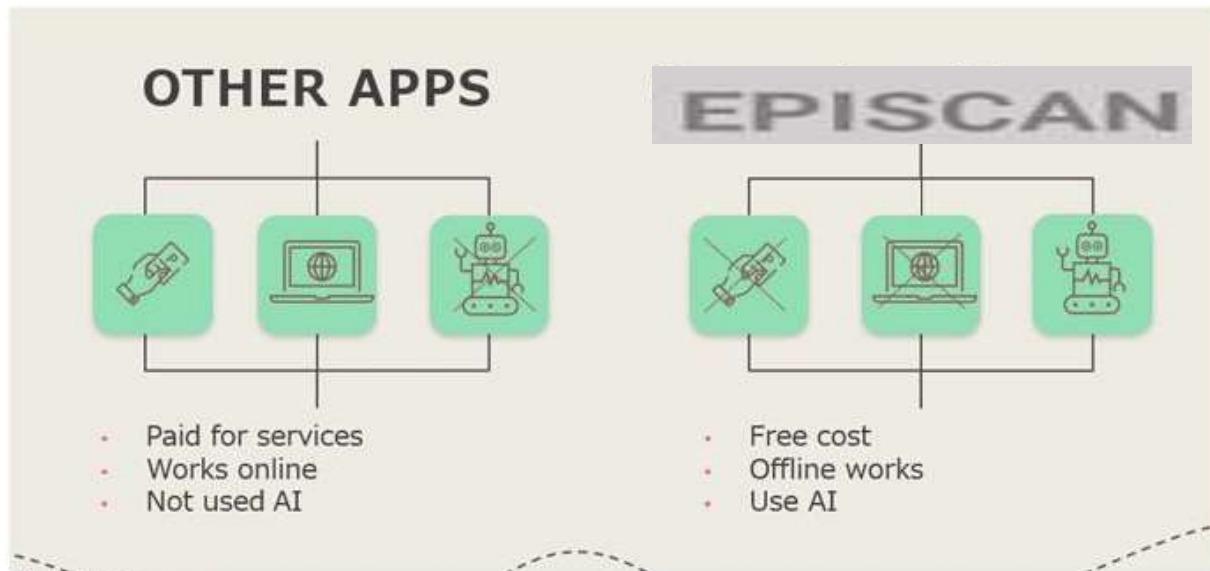


Figure 18 Main feature

After extensive research in the scientific and technological fields, we aimed to create an application for skin cancer detection that would stand out by addressing key limitations observed in existing solutions. Our approach focuses on three main pillars:

1. **Artificial Intelligence (AI):**

- AI is at the core of our application, leveraging advanced machine learning models to accurately diagnose skin cancer. By integrating state-of-the-art AI techniques, our application provides precise and reliable results, setting a new standard in the field of medical diagnostics.

2. **Cost Efficiency:**

- We observed that many similar applications impose high costs on users, which can be a significant barrier to widespread adoption. To overcome this, we developed our application to be completely free of charge. This ensures that everyone, regardless of their financial situation, can benefit from our advanced diagnostic tool.

3. **Offline Functionality:**

- Recognizing that reliable internet connectivity is not always available, especially in rural or remote areas, we designed our application to function offline. This feature ensures that users can access diagnostic services anytime, anywhere, without relying on an internet connection.

Key Features:

- **AI-Powered Diagnosis:** Utilizes convolutional neural networks (CNNs) and other deep learning algorithms to analyze skin images and detect various types of skin cancer with high accuracy.
- **User-Friendly Interface:** Designed for ease of use, allowing users to capture images and receive diagnostic results effortlessly.
- **No Cost to Users:** Completely free to use, removing financial barriers and ensuring accessibility for all.
- **Offline Capability:** Enables users to perform diagnoses without the need for an internet connection, making it ideal for use in remote or underserved areas.

Conclusion:

Our application represents a significant advancement in the field of skin cancer detection, combining cutting-edge AI technology, cost efficiency, and offline functionality. By addressing these critical areas, we aim to provide a highly accessible, reliable, and user-friendly tool that can make a meaningful impact on early cancer detection and treatment.

5.4 Application Main System

This section discusses the implementation details of the main system of the mobile application for skin cancer detection.

User Registration and Login:

Guest Registration: Users can register as guests to quickly access the application's features without creating an account.

Email Registration: Users can register using their email address, with a two-step confirmation for added security.

Google Login: The application supports Google login, allowing users to sign in using their Google account credentials for a seamless experience.

Server Updates:

Code Verification: Upon logging in, the system verifies the user's code and updates the server with the latest data.

Data Synchronization: The server system is immediately updated to reflect any changes or new information, ensuring that users always have access to the most current data.

Main Page:

Application Information: The main page provides users with information about the application and instructions on how to use it.

User Settings: Users have the freedom to update their personal information, such as name and password, at any time through the application settings.

Ease of Use:

User-Friendly Interface: The application is designed to be intuitive and easy to navigate, allowing anyone who downloads it to quickly understand how to use it for skin cancer detection.

Guided Instructions: Clear instructions and prompts guide users through the process of capturing and uploading images for analysis.

Key Features:

AI-Powered Diagnostics: The application leverages advanced AI algorithms to analyze skin images and detect potential cancerous lesions.

Offline Capability: Users can perform diagnostics even without an internet connection, ensuring accessibility in remote areas.

Cost-Free Access: The application is free to use, making it accessible to a wide audience regardless of financial status.

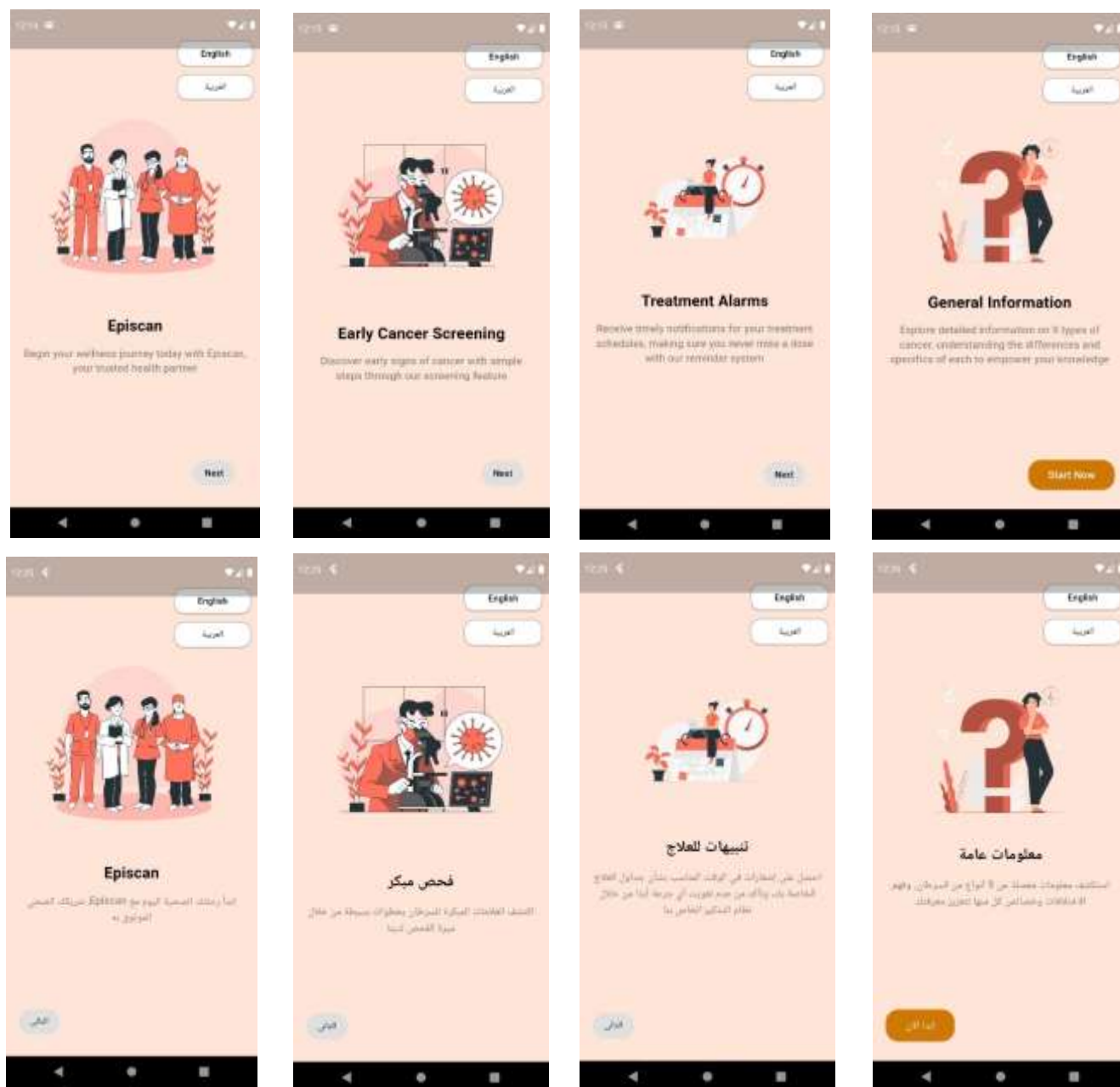
Implementation Details:

Frontend Development: The user interface is developed using Dart and Flutter, providing a seamless experience across both iOS and Android platforms.

Backend Services: The backend services are powered by cloud-based solutions like AWS or Google Cloud, ensuring robust and scalable data processing.

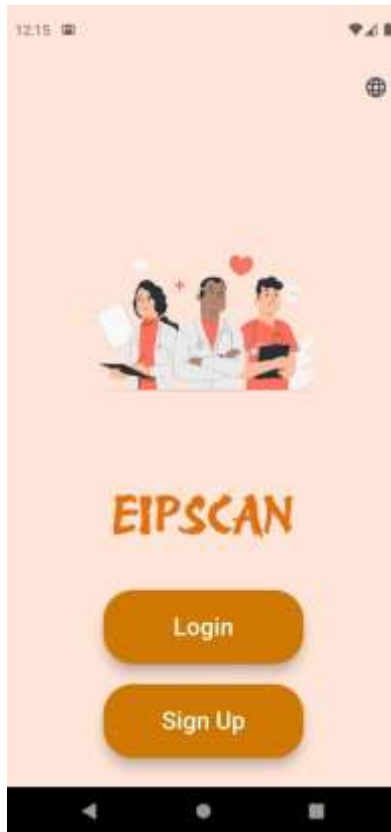
Security Measures: Strong security protocols are implemented to protect user data, including encryption and secure login mechanisms.

5.4 Mobile application



The main system of the application is designed to be user-friendly, secure, and highly efficient, leveraging advanced AI technology to provide accurate skin cancer detection. By offering offline capabilities and being cost-free, the application ensures that users from all backgrounds can benefit from early cancer detection.

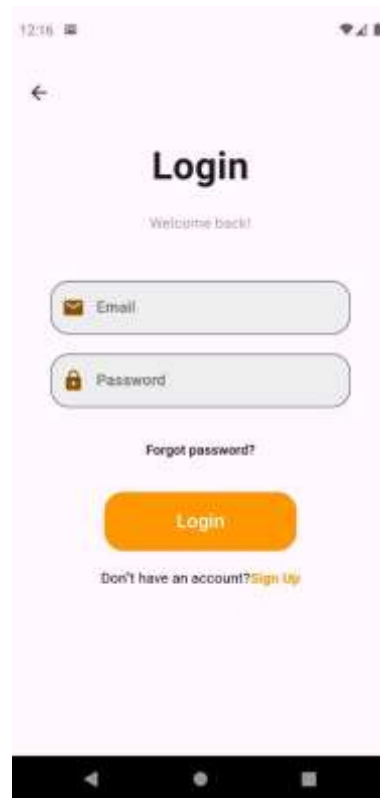
5.4.1 Login and Sign Up Page



The first page after defining the application will consist of two options. The first is if you have an account, you can log in via the login button.

If you do not have an account, you can create one via the sign up button

5.4.2 Login page

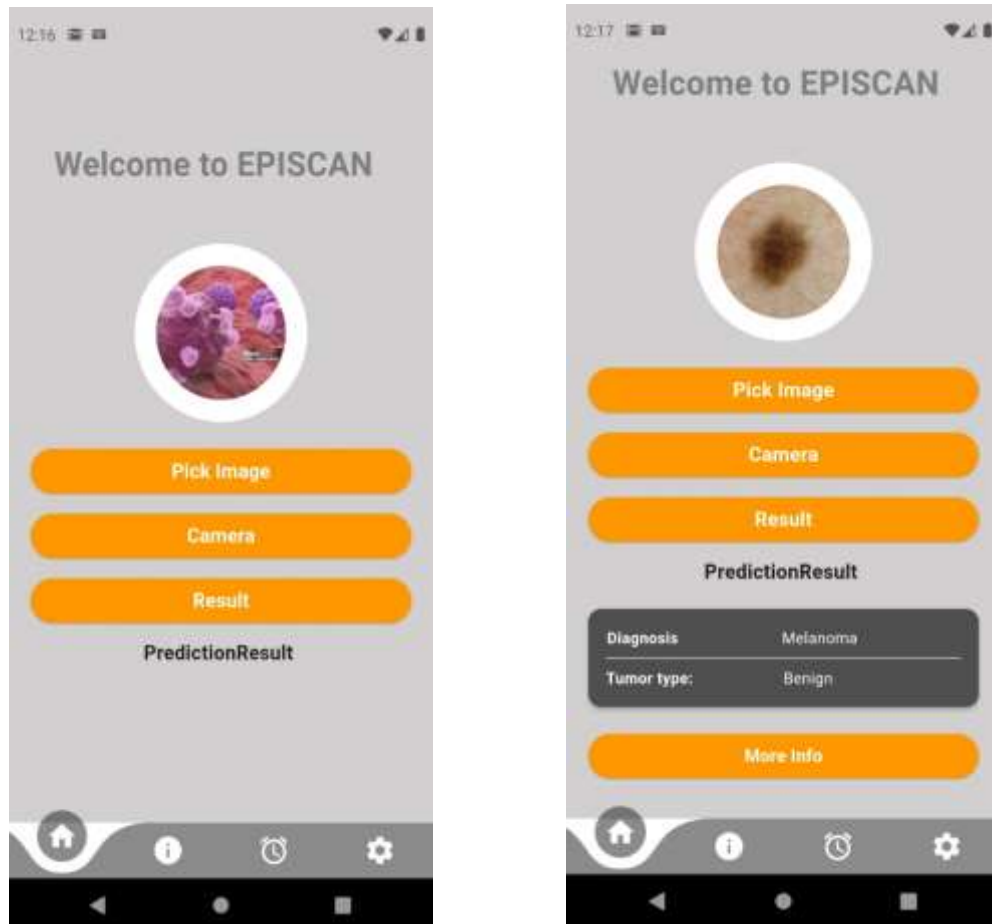


The login page allows the user to log in to the application using his own account that he has previously registered and his computer has been registered in the databa

5.4.3 Sign up

Registration page: The registration page through which the user can register a new account in the application so that he can use the application and log in to the main page. When the registration button is pressed, the data is sent to Firebase and thus he can log in to the application. On this page, the user is asked to enter All of (username - email - phone number - password - reset password).

5.4.4 Home page



Home Page: The home page contains 3 buttons, which are the camera to take a photo of the plant or fruit, the gallery button to take a photo from the phone, and the show result button. Also At the bottom of the home page is a bar containing icons for general information, medication appointment reminders, and app settings.

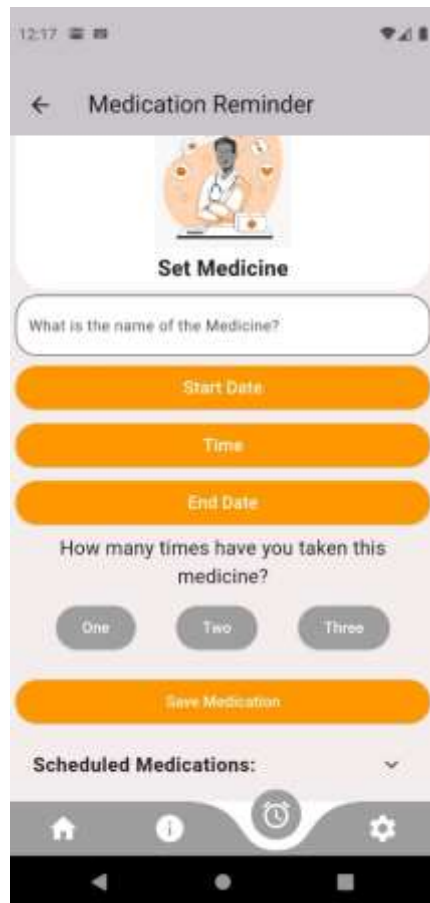
5.4.6 Disease Information



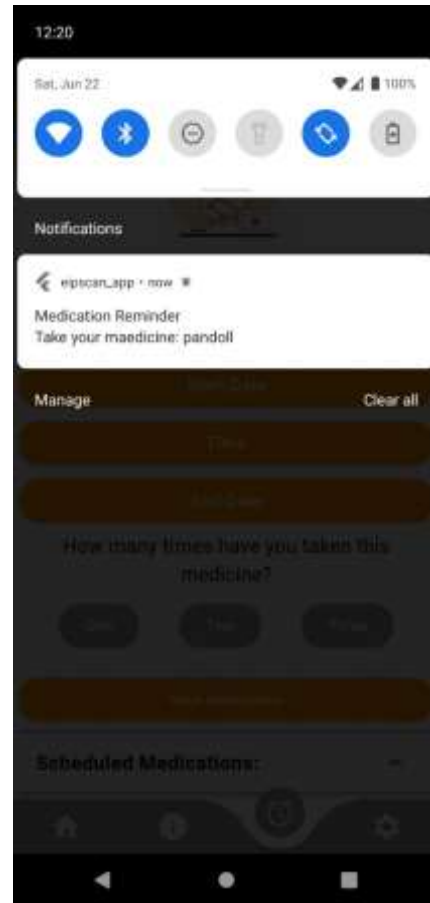
Diagnosing the disease, where the image chosen by the user appears. In the circle, the diagnosis is made when the result button is pressed, and the name of the disease appears as shown.

General information page, which contains information about each of the nine diseases. The disease is defined, The cause of its skin infection, and ways to prevent it.

5.4.7 Medication Reminder

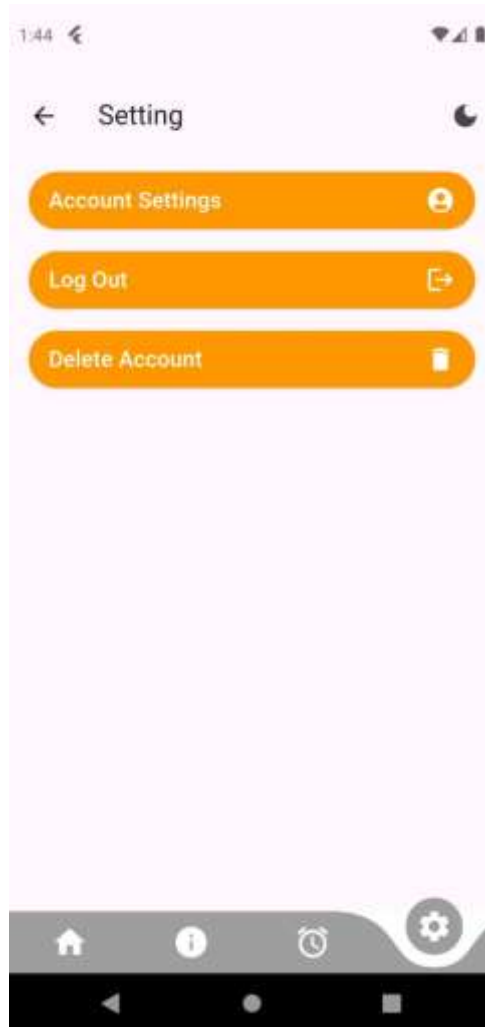


The screenshot shows the 'Medication Reminder' app interface. At the top, there is a back arrow and the title 'Medication Reminder'. Below this is a header with a doctor icon and the text 'Set Medicine'. The form includes a text input field for 'What is the name of the Medicine?'. Below the input field are three orange buttons labeled 'Start Date', 'Time', and 'End Date'. Further down is a section titled 'How many times have you taken this medicine?' with three radio button options: 'One', 'Two', and 'Three'. At the bottom of the form is an orange 'Save Medication' button. Below the form is a section titled 'Scheduled Medications:' with a dropdown arrow. The bottom navigation bar contains icons for home, information, reminders (active), and settings.



The medication appointment reminder page is where the patient registers the name of the medication and chooses the reminder system, whether that it once, twice, or three times a day, and a notification of the appointment will be sent to him.

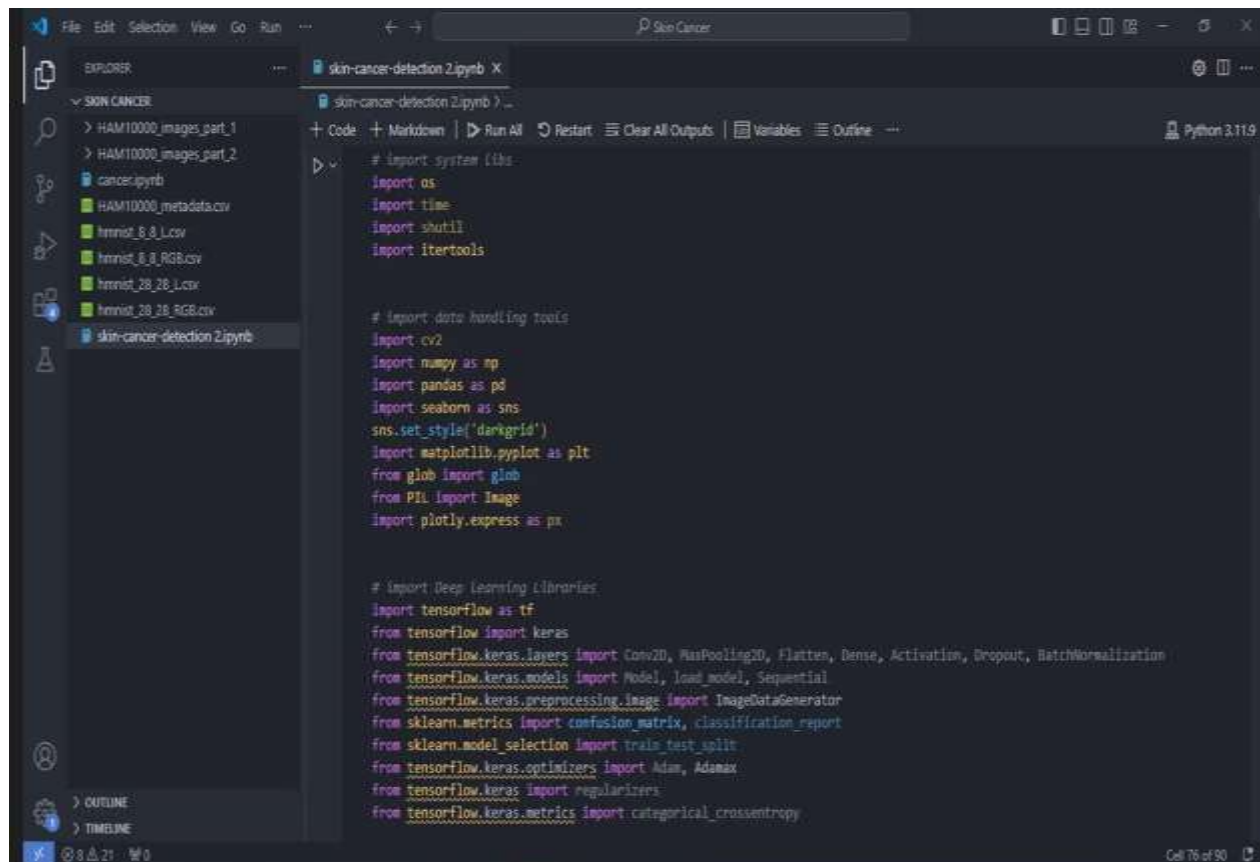
5.4.8 Setting Page:



The settings page contains account settings, changing the language, exiting the application, and also deleting the account.

Chapter 6: Code And Result

6.1 Project Implementation Code In machine learning:



```
# import system libs
import os
import time
import shutil
import itertools

# import data handling tools
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt
from glob import glob
from PIL import Image
import plotly.express as px

# import Deep Learning Libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras.models import Model, load_model, Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras import regularizers
from tensorflow.keras.metrics import categorical_crossentropy
```

1. Importing Libraries

```
import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
    Dense, Activation, Dropout, BatchNormalization

from tensorflow.keras.models import Model, load_model, Sequential

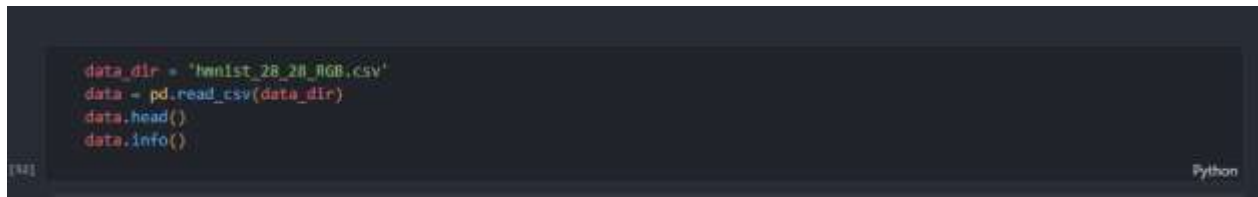
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import confusion_matrix, classification_report
```



```
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras import regularizers
from tensorflow.keras.metrics import categorical_crossentropy
```

This section imports the required libraries and modules for building and training the disease detection model.



```
[12]: data_dir = 'hmnist_28_28_RGB.csv'
      data = pd.read_csv(data_dir)
      data.head()
      data.info()
```

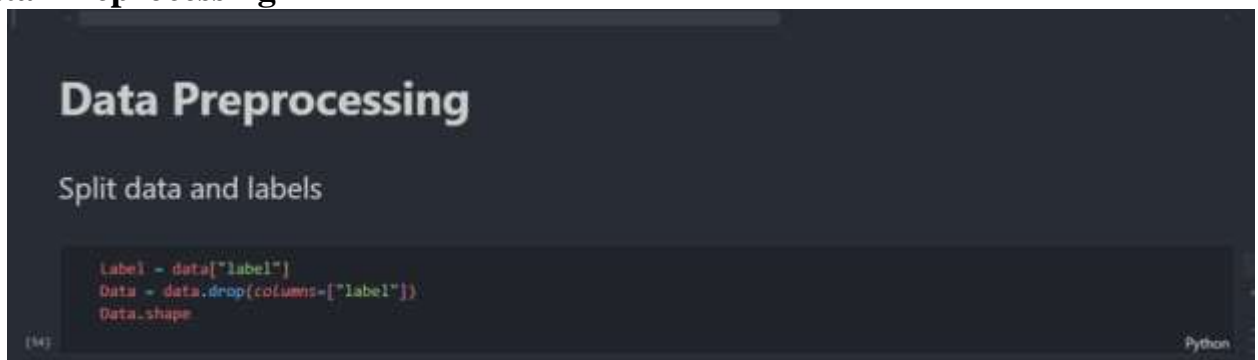
Python

2- Reading the Data

```
data_dir = 'hmnist_28_28_RGB.csv'
data = pd.read_csv(data_dir)
data.head()
data.info()
```

Here it uploads data to and provides a quick overview of the structure and properties of the data

3- Data Preprocessing



Data Preprocessing

Split data and labels

```
[14]: label = data["label"]
      data = data.drop(columns=["label"])
      data.shape
```

Python

- Split data and labels

```
Label=data["label"]
```

```
Data = data.drop(columns=["label"])
```

```
Data.shape
```

Here we divide the dataset into
Label: A variable containing labels/data categories Data:
DataFrame containing only data (images)



```
Handling imbalanced datasets

from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE

[41] Python

from imblearn.over_sampling import RandomOverSampler

oversample = RandomOverSampler()
Data, Label = oversample.fit_resample(Data, Label)
Data = np.array(Data).reshape(-1, 28, 28, 3)
print('Shape of Data :', Data.shape)
Data[1]

[42] Python
... Shape of Data : (46935, 28, 28, 3)

Label = np.array(Label)
Label

[43] Python
```

- Handling imbalanced datasets

```
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import RandomOverSampler oversample
= RandomOverSampler()
Data, Label = oversample.fit_resample(Data, Label)
Data = np.array(Data).reshape(-1, 28, 28, 3) print('Shape
of Data :', Data.shape)
Data[1]
Label = np.array(Label)
Label
```

Here we address the data imbalance using redundant processing techniques, and then reshape the data to suit the requirements of deep models of visualization data learning.

Convert abbreviations to it's words

```
classes = {4: ('nv', 't5grmelanocytic nevi'),
           6: ('mel', 'melanoma'),
           2: ('bkl', 'benign keratosis-like lesions'),
           1: ('bcc', 'basal cell carcinoma'),
           5: ('vasc', 'pyogenic granulomas and hemorrhage'),
           0: ('akiec', 'Actinic keratoses and intraepithelial carcinomae'),
           3: ('df', 'dermatofibroma'),
           7: ('scc', 'Squamous Cell Carcinoma'),
           8: ('mf', 'Mycosis Fungoides')}
```

- Convert abbreviations to their words

```
classes = {4: ('nv', 't5grmelanocytic nevi'),
           6: ('mel', 'melanoma'),
           2: ('bkl', 'benign keratosis-like lesions'),
           1: ('bcc', 'basal cell carcinoma'),
           5: ('vasc', 'pyogenic granulomas and hemorrhage'),
           0: ('akiec', 'Actinic keratoses and intraepithelial carcinomae'),
           3: ('df', 'dermatofibroma'),
           7: ('scc', 'Squamous Cell Carcinoma'),
           8: ('mf', 'Mycosis Fungoides')}
```

Splitting train and test

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(Data, Label, test_size = 0.25, random_state = 49)
print(f'X_train shape: {X_train.shape}\nX_test shape: {X_test.shape}')
print(f'y_train shape: {y_train.shape}\ny_test shape: {y_test.shape}')
```

Here I make a dictionary called classes that helps to classify according to the requirements of the automated mode

- Splitting train and test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Data, Label, test_size
= 0.25, random_state = 49)
print(f'X_train shape: {X_train.shape}\nX_test shape: {X_test.shape}')
print(f'y_train shape: {y_train.shape}\ny_test shape: {y_test.shape}')
```

Here it divides the dataset into two training and test groups at a rate of 75%,25%

Convert labels to categorical types

```
from tensorflow.keras.utils import to_categorical  
  
y_train = to_categorical(y_train)  
y_test = to_categorical(y_test)
```

- Categorical encoding

```
from tensorflow.keras.utils import to_categorical
```

```
y_train = to_categorical(y_train)
```

```
y_test = to_categorical(y_test)
```

Here he does Convert labels to categorical type

Create Image Data Generation

```
datagen = ImageDataGenerator(rescale=(1./255)  
                             ,rotation_range=10  
                             ,zoom_range = 0.1  
                             ,width_shift_range=0.1  
                             ,height_shift_range=0.1)  
  
testgen = ImageDataGenerator(rescale=(1./255))
```

- Data Augmentation

```
datagen = ImageDataGenerator(rescale=(1./255)
```

```
                             ,rotation_range=10
```

```
                             ,zoom_range = 0.1
```

```
                             ,width_shift_range=0.1
```

```
                             ,height_shift_range=0.1)
```

```
testgen = ImageDataGenerator(rescale=(1./255))
```

in this section, an `ImageDataGenerator` object is created for data augmentation. It applies various transformations to the training images to increase the diversity

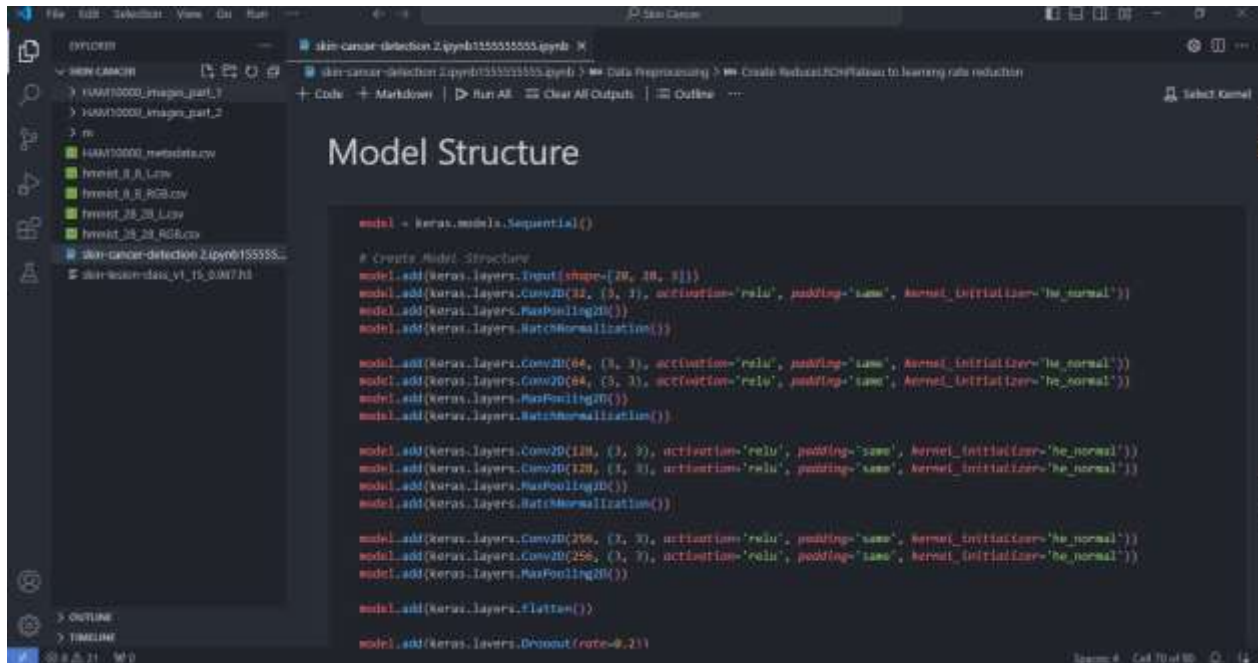
```
from keras.callbacks import ReduceLROnPlateau  
  
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',  
                                             , patience = 2  
                                             , verbose=1  
                                             , factor=0.5  
                                             , min_lr=0.00001)
```

- ReduceLROnPlateau to learning rate reduction

```
from keras.callbacks import ReduceLROnPlateau  
learning_rate_reduction =  
ReduceLROnPlateau(monitor='val_accuracy'  
                  , patience = 2  
                  , verbose=1  
                  , factor=0.5  
                  , min_lr=0.00001)
```

Here, we're using ReduceLROnPlateau to reduce the learning rate during the training process based on the performance of the validation set. Decreasing the learning rate can help improve the training process and avoid excessive increases in the learning rate, ultimately leading to a better and more stable model.

4-model building



```
model = keras.models.Sequential()
```

```
model.add(keras.layers.Input(shape=[28, 28, 3]))
```

```
model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same',  
kernel_initializer='he_normal'))
```

```
model.add(keras.layers.MaxPooling2D())
```

```
model.add(keras.layers.BatchNormalization())
```

```
model.add(keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same',  
kernel_initializer='he_normal'))
```

```
model.add(keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same',  
kernel_initializer='he_normal'))
```

```
model.add(keras.layers.MaxPooling2D())
```

```
model.add(keras.layers.BatchNormalization())
```

```
model.add(keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same',  
kernel_initializer='he_normal'))
```

```
model.add(keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same',
kernel_initializer='he_normal'))
model.add(keras.layers.MaxPooling2D())
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same',
kernel_initializer='he_normal'))
model.add(keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same',
kernel_initializer='he_normal'))
model.add(keras.layers.MaxPooling2D()) model.add(keras.layers.Flatten())
model.add(keras.layers.Dropout(rate=0.2))
model.add(keras.layers.Dense(units=activation='relu',
kernel_initializer='he_normal'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Dense(units=128, activation='relu',
kernel_initializer='he_normal'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Dense(units=64, activation='relu',
kernel_initializer='he_normal'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Dense(units=32, activation='relu',
kernel_initializer='he_normal',
kernel_regularizer=keras.regularizers.L1L2()))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Dense(units=7, activation='softmax',
kernel_initializer='glorot_uniform', name='classifier'))

model.compile(Adamax(learning_rate= 0.001), loss=
'categorical_crossentropy', metrics= ['accuracy'])
model.summary()
```


This code trains the model for 25 epochs using batches containing 128 samples each, while providing validation data to evaluate the model's performance after each epoch. The learning rate reduction callback is used to adjust the learning process during training.

```
def plot_training(history):
    tr_acc = history['accuracy']
    tr_loss = history['loss']
    val_acc = history['val_accuracy']
    val_loss = history['val_loss']
    index_loss = np.argmin(val_loss)
    val_lowest = val_loss[index_loss]
    index_acc = np.argmax(val_acc)
    acc_highest = val_acc[index_acc]

    plt.figure(figsize=(20, 8))
    plt.style.use('fivethirtyeight')
    epochs = [i+1 for i in range(len(tr_acc))]
    loss_label = f'Best epoch= {str(index_loss + 1)}'
    acc_label = f'Best epoch= {str(index_acc + 1)}'

    plt.subplot(1, 2, 1)
    plt.plot(epochs, tr_loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'g', label='Validation loss')
    plt.scatter(index_loss + 1, val_lowest, s=150, c='blue', label=loss_label)
    plt.title('Training and Validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(epochs, tr_acc, 'r', label='Training accuracy')
    plt.plot(epochs, val_acc, 'g', label='Validation accuracy')
    plt.scatter(index_acc + 1, acc_highest, s=150, c='blue', label=acc_label)
    plt.title('Training and Validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.tight_layout()
    plt.show()
```



Show training history

```
def plot_training(hist):
    tr_acc = hist.history['accuracy']
    tr_loss = hist.history['loss']
    val_acc = hist.history['val_accuracy']
    val_loss = hist.history['val_loss']
    index_loss = np.argmin(val_loss)
    val_lowest = val_loss[index_loss]
    index_acc = np.argmax(val_acc)
    acc_highest = val_acc[index_acc]
    plt.figure(figsize= (20, 8))
    plt.style.use('fivethirtyeight')
    Epochs = [i+1 for i in range(len(tr_acc))]
    loss_label = f'best epoch= {str(index_loss + 1)}'
    acc_label = f'best epoch= {str(index_acc + 1)}'

    plt.plot(Epochs, tr_loss, 'r', label= 'Training loss')
    plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')
    plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)
    plt.title("Training and Validation Loss")
    plt.xlabel('Epochs')
    plt.ylabel('Loss')

    plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy')
    plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')
    plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)
    plt.title("Training and Validation Accuracy")
```

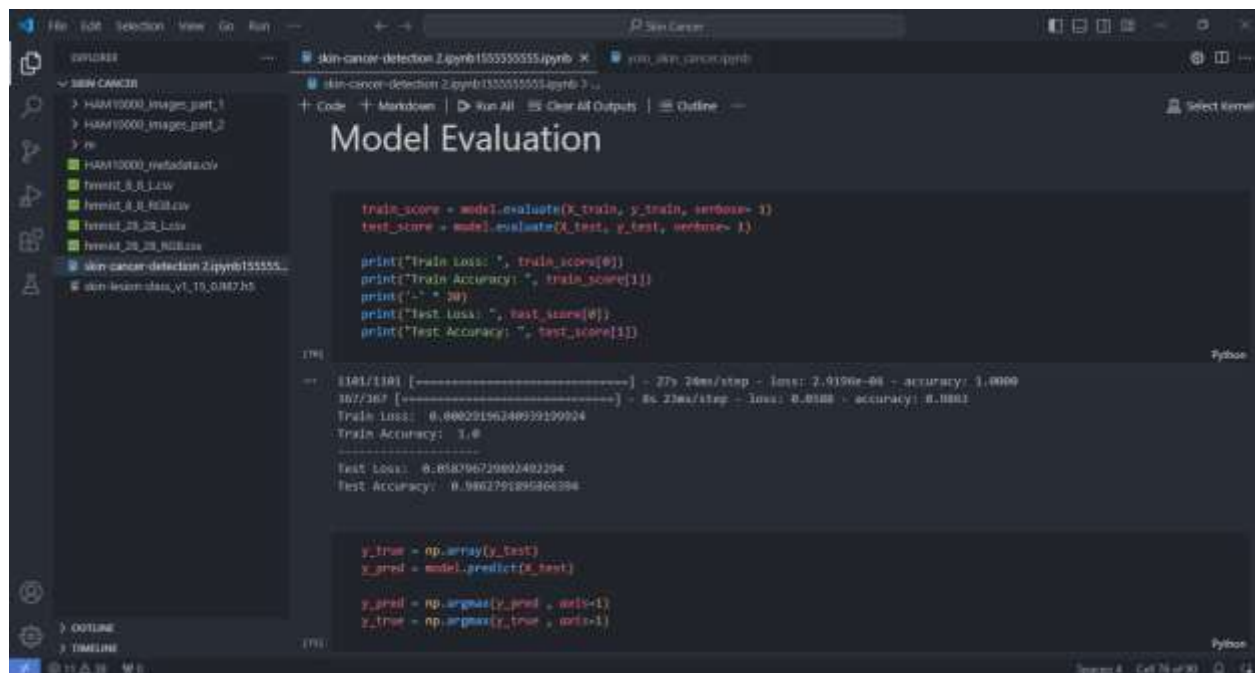
```
plt.ylabel('Accuracy')
```

```
plt.legend()
```

```
plt.tight_layout
```

```
plt.show)(
```

in this section Useful for understanding model performance on a training and verification group, and helps in making decisions about optimization processes and model correction.



```
train_score = model.evaluate(X_train, y_train, verbose= 1)
test_score = model.evaluate(X_test, y_test, verbose= 1)

print("Train Loss: ", train_score[0])
print("Train Accuracy: ", train_score[1])
print('-' * 20)
print("Test Loss: ", test_score[0])
print("Test Accuracy: ", test_score[1])
```

```
1101/1101 [=====] - 27s 24ms/step - loss: 2.9196e-06 - accuracy: 1.0000
1077/1077 [=====] - 8s 23ms/step - loss: 0.0188 - accuracy: 0.9881
Train loss: 0.0002196240939299924
Train Accuracy: 1.0
-----
Test loss: 0.058796720002402294
Test Accuracy: 0.986279189586394
```

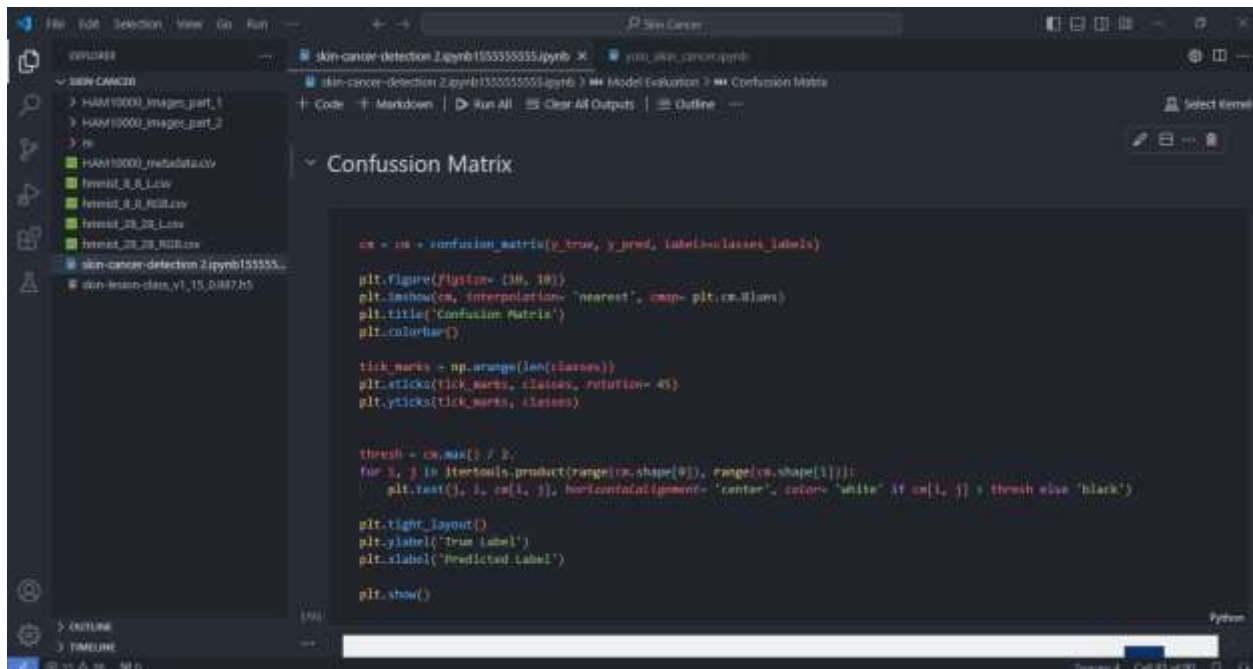
```
y_train = np.array(y_test)
y_pred = model.predict(X_test)

y_pred = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_true, axis=1)
```

6 -Model Evaluation

```
train_score = model.evaluate(X_train, y_train, verbose= 1)
test_score = model.evaluate(X_test, y_test, verbose= 1)
print("Train Loss: ", train_score[0])
print("Train Accuracy: ", train_score[1])
print('-' * 20)
print("Test Loss: ", test_score[0])
print("Test Accuracy: ", test_score[1])
```

in this section ,It's used to measure the model's performance after training and evaluate it on both training and test data.



7 - Confussion Matrix

```
cm = cm = confusion_matrix(y_true, y_pred, labels=classes_labels)
plt.figure(figsize= (10, 10))
plt.imshow(cm, interpolation= 'nearest', cmap= plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation= 45)
plt.yticks(tick_marks, classes)
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j], horizontalalignment= 'center', color= 'white' if cm[i, j] >
thresh else 'black')
```

```
plt.tight_layout()
plt.ylabel('True          Label')
plt.xlabel('Predicted    Label')
plt.show()
```

This code snippet is used to visualize the confusion matrix, which is a performance evaluation technique for classification problems.

Save model

```
model.save('Skin Cancer.h5')
```

[74]

Python

8 – save model

```
model.save('Skin Cancer.h5')
```

This section is provided to predict the name of the category (disease) given to the image Using the trained form. "predict_class_h5" uses the saved .h5 model to make prediction

Skin cancer classification code (benign and malignant)

```
import json
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical, Progbar
from tensorflow.keras.callbacks import Callback

# load COCO JSON format annotations
def load_coco_annotations(json_file):
    with open(json_file, 'r') as f:
        data = json.load(f)
    return data
```

1. Importing Libraries

```
import json
```

```
import os
```

```
import numpy as np
```

```
import tensorflow as tf
```

```
from tensorflow.keras.applications import ResNet50
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
from tensorflow.keras.optimizers import Adam
```

```
from tensorflow.keras.layers import Dense, Flatten, Dropout
```

```
from tensorflow.keras.models import Model
```

```
from sklearn.model_selection import train_test_split
```

```
from tensorflow.keras.utils import to_categorical, Progbar
```

```
from tensorflow.keras.callbacks import Callback
```

This section imports the required libraries and modules for

building and training the disease detection model.

```
# Load COCO JSON format annotations
```

```
def load_coco_annotations(json_file):
```

```
    with open(json_file, 'r') as f:
```

```
        data = json.load(f)
```

```
    return data
```

This function makes it easy to load COCO comments from a JSON file,

enabling the developer to use this data in image processing and object detection tasks.

```
# Preprocess the COCO dataset
def preprocess_coco(data, image_dir, input_shape=(128, 128)):
    images = []
    labels = []
    for annotation in data['annotations']:
        image_id = annotation['image_id']
        category_id = annotation['category_id']

        # Find corresponding image file name
        image_file = next(img['file_name'] for img in data['images'] if img['id'] == image_id)
        image_path = os.path.join(image_dir, image_file)

        # Load and preprocess image
        image = tf.keras.preprocessing.image.load_img(image_path, target_size=input_shape)
        image = tf.keras.preprocessing.image.img_to_array(image)
        image = tf.keras.applications.resnet50.preprocess_input(image)

        images.append(image)
        labels.append(category_id)

    images = np.array(images)
    labels = np.array(labels)

    return images, labels
```

-3Preprocess the COCO dataset

```
def preprocess_coco(data, image_dir, input_shape=(128, 128)):
    images = []
    labels = []
    for annotation in data['annotations']:
        image_id = annotation['image_id']
        category_id = annotation['category_id']

        # Find corresponding image file name
        image_file = next(img['file_name'] for img in data['images'] if img['id'] == image_id)
        image_path = os.path.join(image_dir, image_file)

        # Load and preprocess image
        image = tf.keras.preprocessing.image.load_img(image_path, target_size=input_shape)
        image = tf.keras.preprocessing.image.img_to_array(image)
        image = tf.keras.applications.resnet50.preprocess_input(image)

        images.append(image)
        labels.append(category_id)

    images = np.array(images)
    labels = np.array(labels)

    return images, labels
```

This function is used to pre-process the COCO dataset to train a learning model
We change the size of the image, we process the image, we turn it into a matrix, and we
create a label for each image


```
# Custom callback to suppress detailed output and show progress bar
class TrainingLogger(Callback):
    def on_epoch_begin(self, epoch, Logs=None):
        self.progbar = Progbar(target=self.params['steps'])
        print(f"Epoch {epoch + 1}/{self.params['epochs']}")

    def on_batch_end(self, batch, Logs=None):
        self.progbar.update(batch + 1)

    def on_epoch_end(self, epoch, Logs=None):
        print(f" - loss: {Logs['loss']:.4f} - accuracy: {Logs['accuracy']:.4f}")
```

Custom callback to suppress detailed output and show progress bar

```
class TrainingLogger(Callback):
```

```
    def on_epoch_begin(self, epoch, logs=None):
```

```
        self.progbar = Progbar(target=self.params['steps'])
```

```
        print(f"Epoch {epoch + 1}/{self.params['epochs']}")
```

```
    def on_batch_end(self, batch, logs=None):
```

```
        self.progbar.update(batch + 1)
```

```
    def on_epoch_end(self, epoch, logs=None):
```

```
        print(f" - loss: {logs['loss']:.4f} - accuracy: {logs['accuracy']:.4f}")
```

This code defines the TrainingLogger class that inherits from Callback in the Keras framework. The purpose of this class is to record and print the training progress during each epoch and display basic metrics such as loss and accuracy at the end of each epoch.

```
# Path to train directory
train_dir = '/content/malignant_benign2-1/train'

# Load and preprocess the COCO dataset
train_json = os.path.join(train_dir, '_annotations.coco.json')
train_data = load_coco_annotations(train_json)

X, y = preprocess_coco(train_data, train_dir)

# Split data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Convert labels to one-hot encoding
num_classes = 11
y_train = to_categorical(y_train, num_classes=num_classes)
y_val = to_categorical(y_val, num_classes=num_classes)
```

Path to train directory

train_dir = '/content/malignant_benign2-1/train'

Load and preprocess the COCO dataset

train_json = os.path.join(train_dir, '_annotations.coco.json')

train_data = load_coco_annotations(train_json)

X, y = preprocess_coco(train_data, train_dir)

Split data into training and validation sets

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

Convert labels to one-hot encoding

num_classes = 11

y_train = to_categorical(y_train, num_classes=num_classes)

y_val = to_categorical(y_val, num_classes=num_classes)

This code loads the COCO dataset, prepares it,

and divides it into training and validation sets, with a split of 80%-20%,

enabling the developer to use this data to train a classification model.

```
# Define data augmentation
train_datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_datagen = ImageDataGenerator()
```

Define data augmentation

```
train_datagen = ImageDataGenerator(
```

```
    rotation_range=20,
```

```
    width_shift_range=0.2,
```

```
    height_shift_range=0.2,
```

```
    shear_range=0.2,
```

```
    zoom_range=0.2,
```

```
    horizontal_flip=True,
```

```
    fill_mode='nearest'
```

```
)
```

```
val_datagen = ImageDataGenerator()
```

Data augmentation is a technique used in machine learning and computer vision to increase the size and diversity of the original training data set by creating modified versions of existing images. The goal is to improve the performance of training models by exposing them to more diverse and realistic data.

```

# Define ResNet50 model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
x = base_model.output
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(num_classes, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze the layers of ResNet50 except the top layers
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
batch_size = 32
epochs = 70

train_generator = train_datagen.flow(X_train, y_train, batch_size=batch_size)
val_generator = val_datagen.flow(X_val, y_val, batch_size=batch_size)

# Training with custom callback
model.fit(
    train_generator,
    steps_per_epoch=len(X_train) // batch_size,
    epochs=epochs,
    validation_data=val_generator,

```

Define ResNet50 model

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
```

```
x = base_model.output
```

```
x = Flatten()(x)
```

```
x = Dense(1024, activation='relu')(x)
```

```
x = Dropout(0.5)(x)
```

```
predictions = Dense(num_classes, activation='softmax')(x)
```

```
model = Model(inputs=base_model.input, outputs=predictions)
```

```
# Freeze the layers of ResNet50 except the top layers
```

```
for layer in base_model.layers:
```

```
    layer.trainable = False
```

```
# Compile the model
```

```

model.compile(optimizer=Adam(learning_rate=0.0001),
loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model

batch_size = 32

epochs = 70

train_generator = train_datagen.flow(X_train, y_train, batch_size=batch_size)
val_generator = val_datagen.flow(X_val, y_val, batch_size=batch_size)

# Training with custom callback

model.fit(

    train_generator,

    steps_per_epoch=len(X_train) // batch_size,

    epochs=epochs,

    validation_data=val_generator,

    validation_steps=len(X_val) // batch_size,

    verbose=0, # Suppress the default verbose output

    callbacks=[TrainingLogger()] # Use the custom callback for training logs

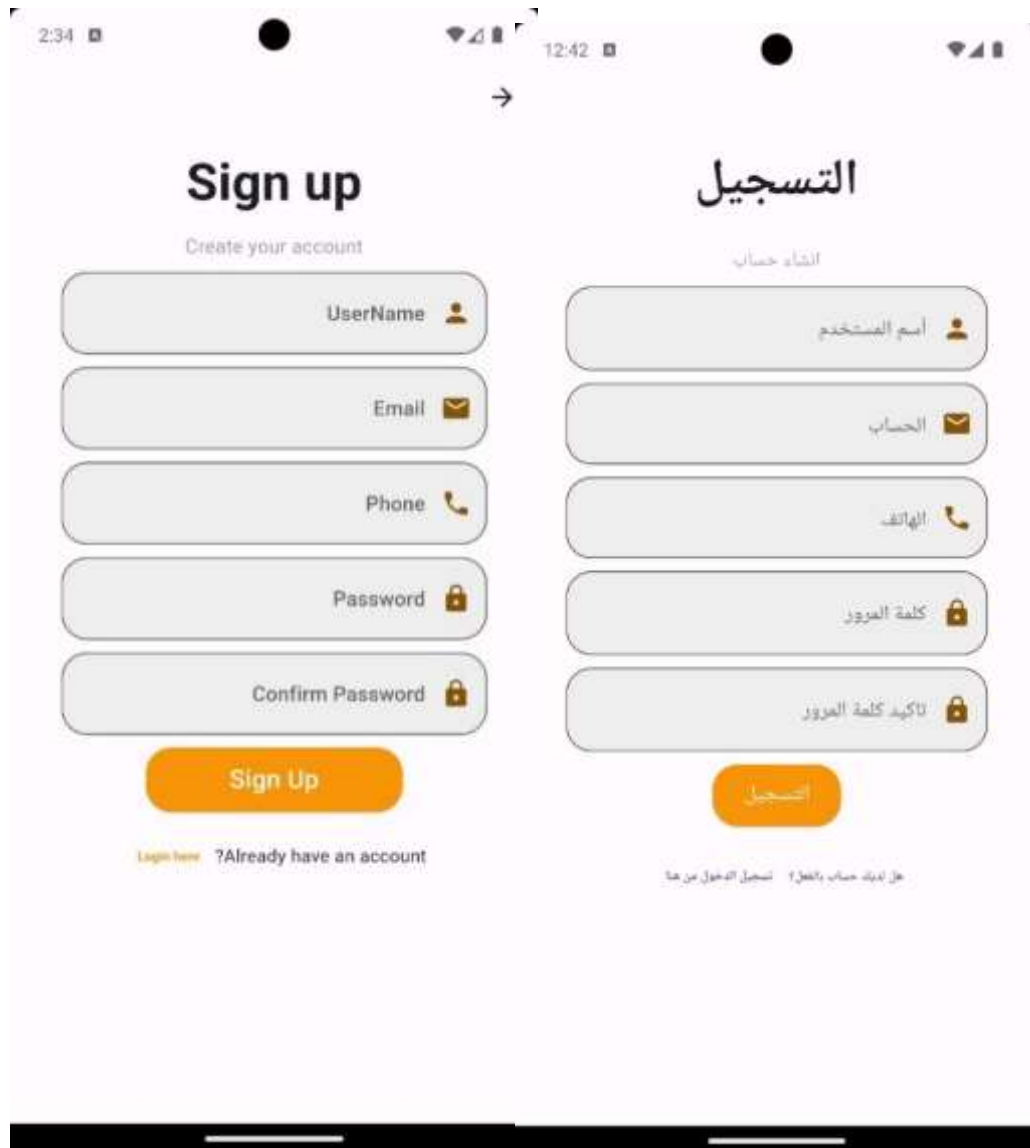
)

```

This excerpt shows how to build and train a ResNet50 model dedicated to the image classification problem using the Data Augmentation technique and freezing the lower layers of the basic model.

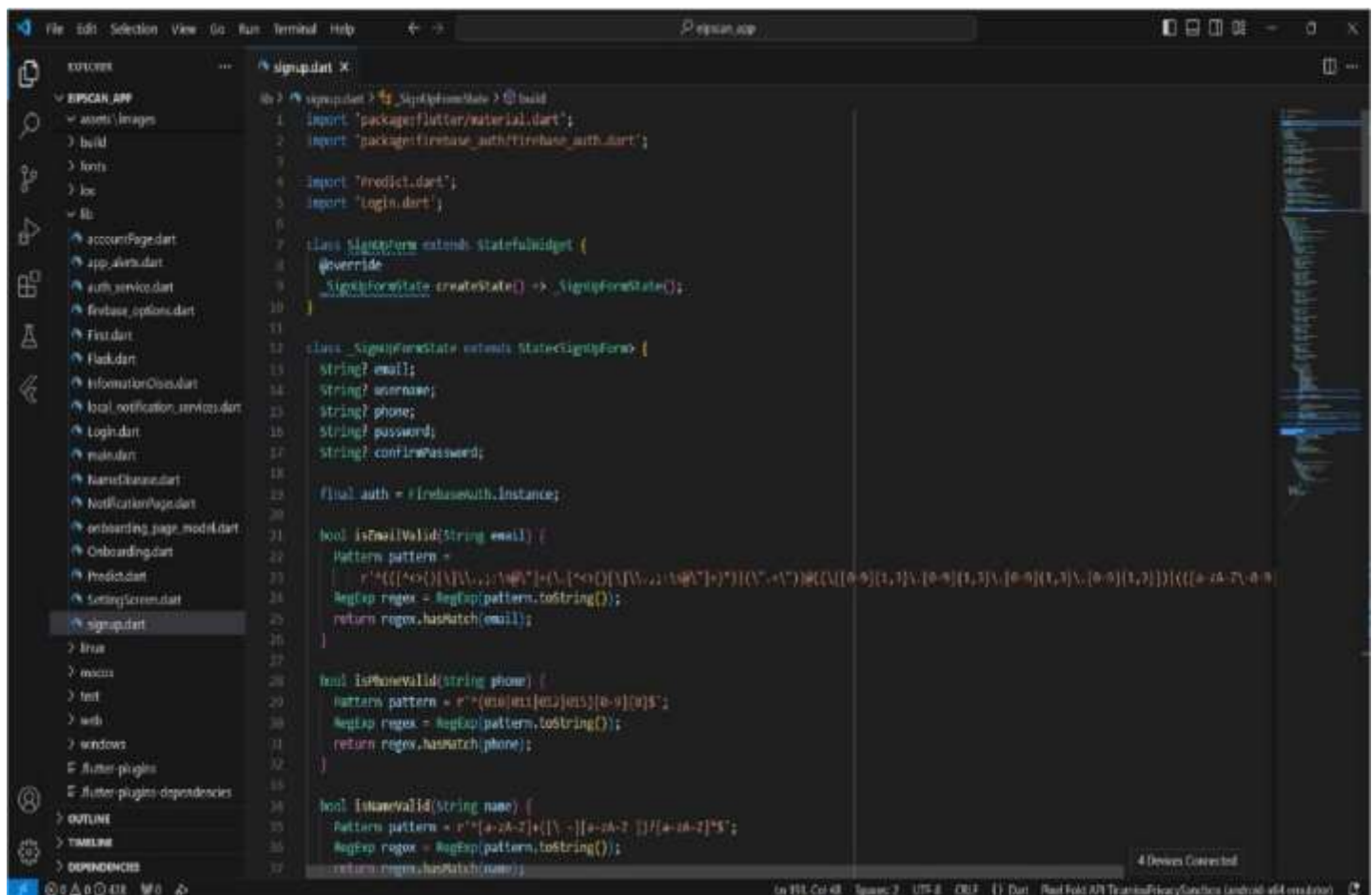
6.2 Project Implementation Code in Mobile Application:

6.2.1 Sign up



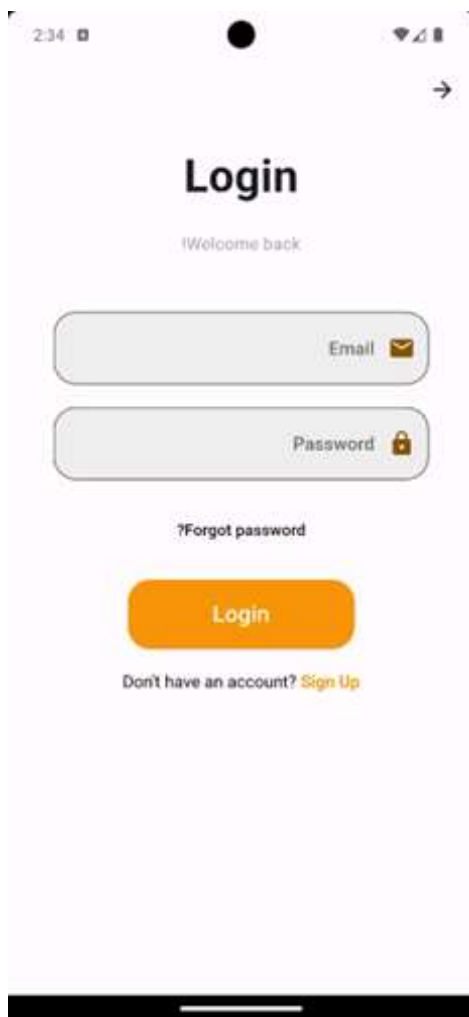
The image displays two side-by-side mobile application screens for user registration. The left screen is in English, titled 'Sign up' with the subtitle 'Create your account'. It features five input fields: 'UserName' (with a person icon), 'Email' (with an envelope icon), 'Phone' (with a phone handset icon), 'Password' (with a lock icon), and 'Confirm Password' (with a lock icon). Below these fields is an orange 'Sign Up' button. At the bottom, there is a link 'Login here' and the text '?Already have an account'. The right screen is in Arabic, titled 'التسجيل' (Registration) with the subtitle 'انشاء حساب' (Create account). It has five corresponding input fields: 'اسم المستخدم' (User Name), 'البريد الإلكتروني' (Email), 'الهاتف' (Phone), 'كلمة المرور' (Password), and 'تأكيد كلمة المرور' (Confirm Password). Below these is an orange 'التسجيل' (Register) button. At the bottom, it says 'هل لديك حساب بالفعل؟' (Do you already have an account?) followed by 'تسجيل الدخول من هنا' (Login from here). Both screens show a status bar at the top with the time (2:34 and 12:42) and a home indicator bar at the bottom.

The user performs the registration process, and here he registers (username, email, phone Password, Confirm Password), and this is where Firebase comes into play When the user finishes registering, Firebase will log this data, and the user can Log in at any time



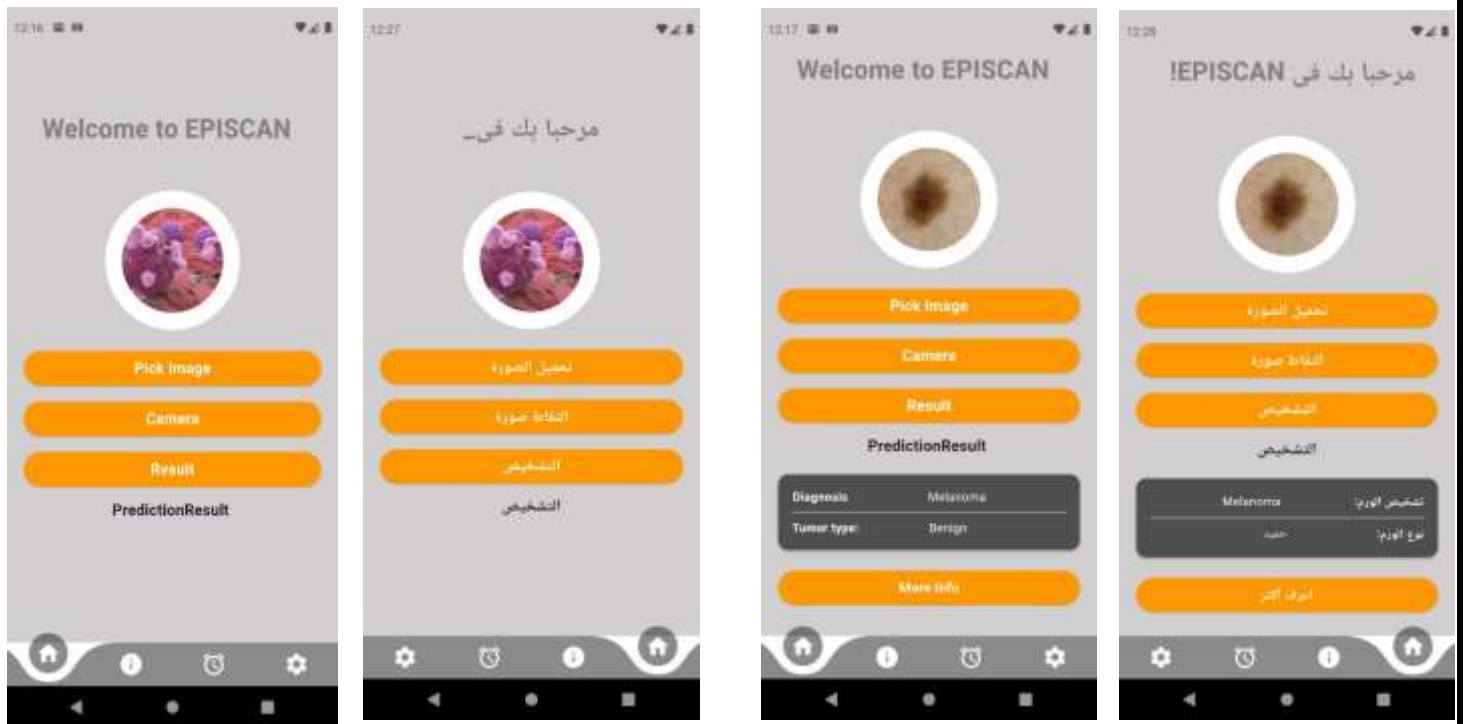
```
1 import 'package:flutter/material.dart';
2 import 'package:firebase_auth/firebase_auth.dart';
3
4 import 'predict.dart';
5 import 'login.dart';
6
7 class SignUpForm extends StatefulWidget {
8   @override
9   SignUpFormState createState() => SignUpFormState();
10 }
11
12 class SignUpFormState extends State<SignUpForm> {
13   String? email;
14   String? username;
15   String? phone;
16   String? password;
17   String? confirmPassword;
18
19   final auth = FirebaseAuth.instance;
20
21   bool isValidEmail(String email) {
22     Pattern pattern =
23       r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]{2,}$';
24     RegExp regex = RegExp(pattern.toString());
25     return regex.hasMatch(email);
26   }
27
28   bool isValidPhone(String phone) {
29     Pattern pattern = r'^(\+91|011|012|015|099)[0-9]{10}$';
30     RegExp regex = RegExp(pattern.toString());
31     return regex.hasMatch(phone);
32   }
33
34   bool isValidName(String name) {
35     Pattern pattern = r'^[a-zA-Z]+([ ]+([a-zA-Z]+)*)?$';
36     RegExp regex = RegExp(pattern.toString());
37     return regex.hasMatch(name);
38   }
```

6.2.2 login:

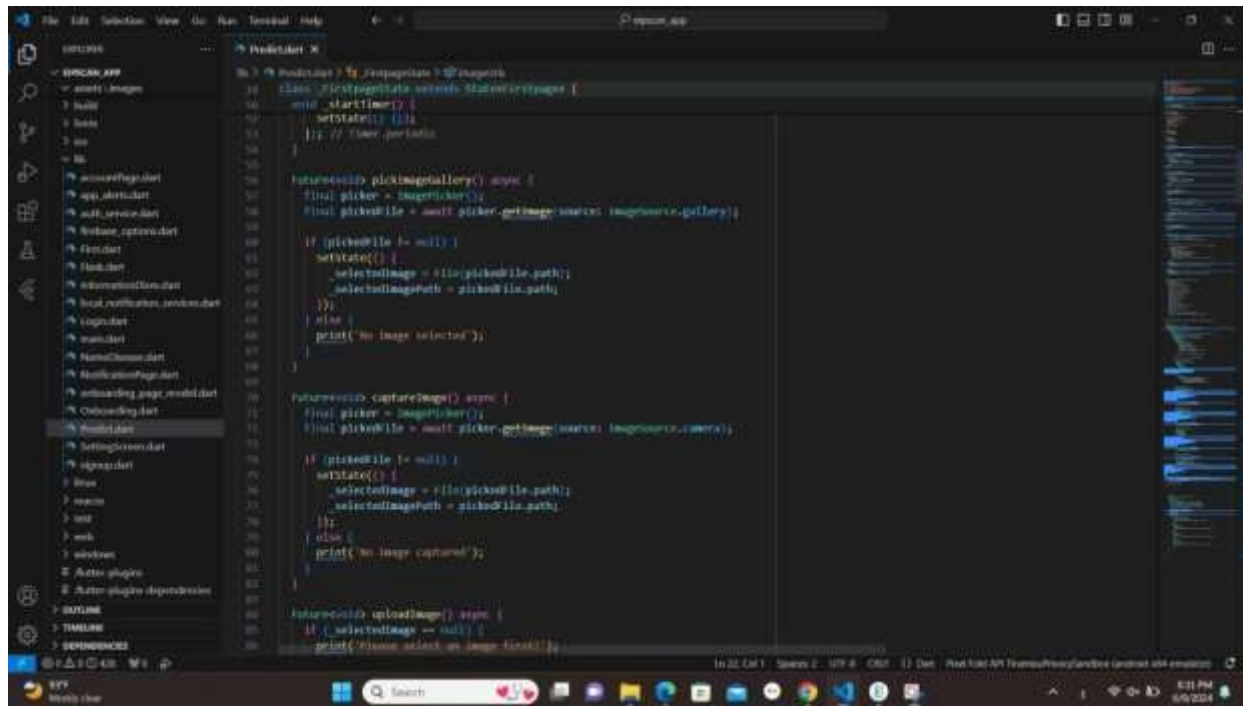


The user logs in using the email and password previously registered on the firebase databas

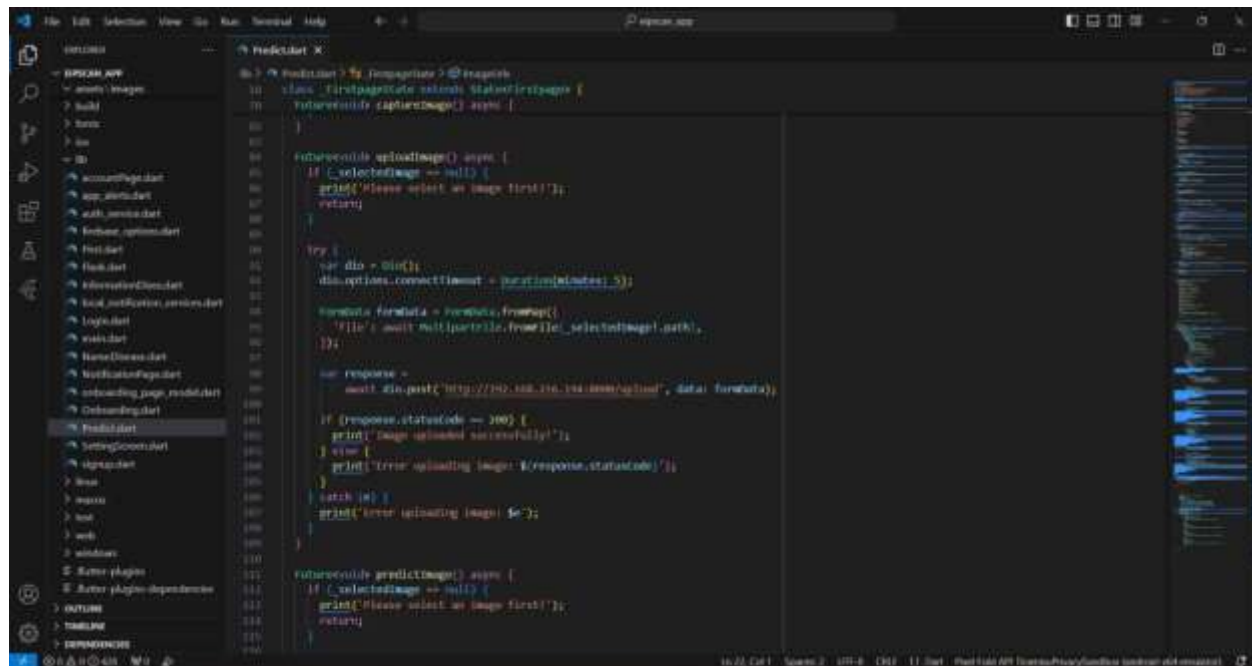
6.2.3 Home:



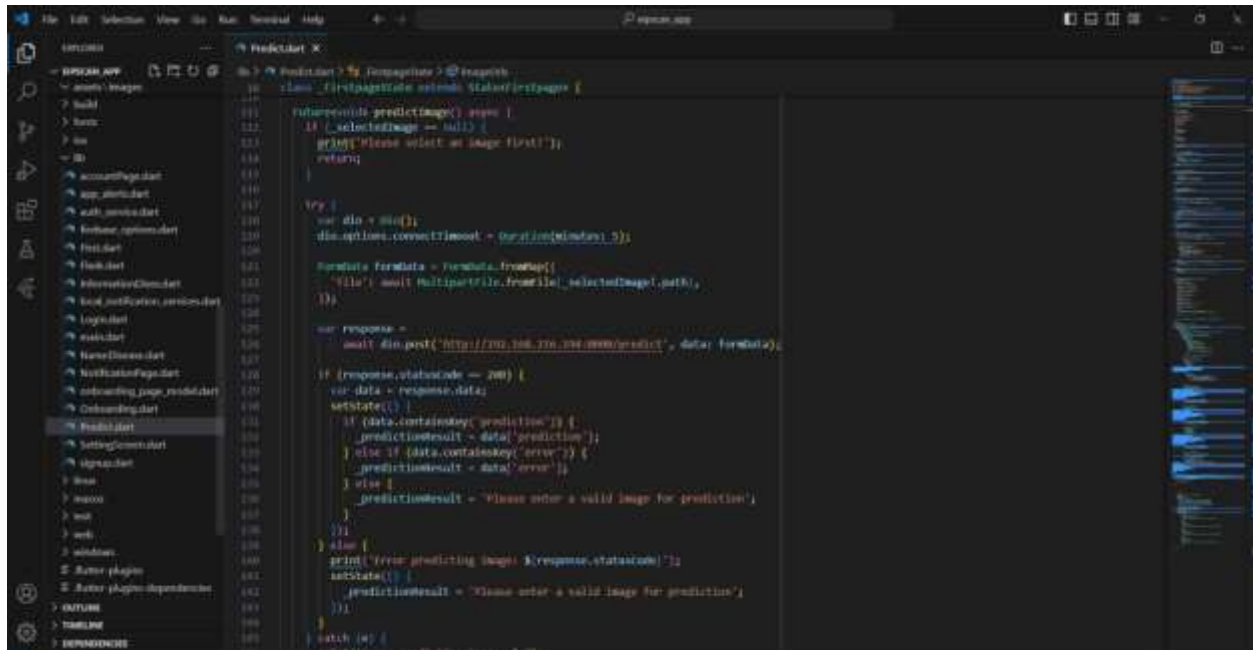
- PickImageGallery , CaptureImage



- UploadImage



- PredictImage

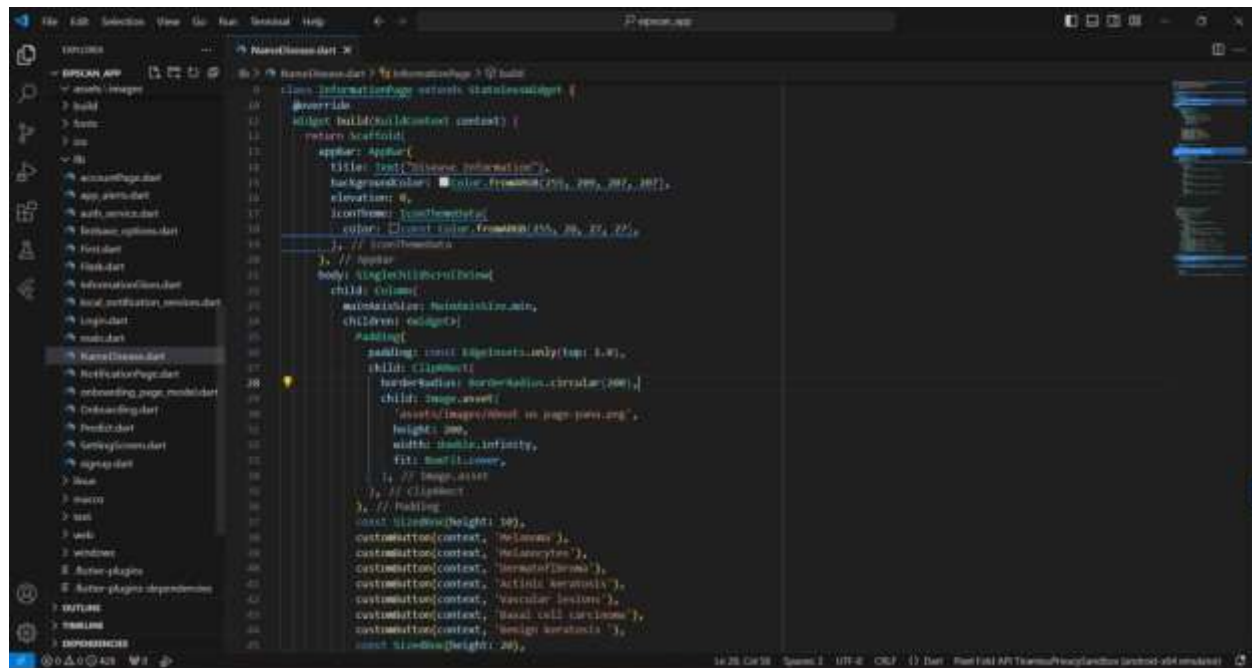


```
10 class _FirstPageState extends State<FirstPage> {
11
12   Future<void> predictImage() async {
13     if (_selectedImage == null) {
14       print('Please select an image first!');
15       return;
16     }
17
18     var dio = Dio();
19     dio.options.connectTimeout = Duration(minutes: 5);
20
21     FormData formData = FormData.fromMap({
22       'file': await MultipartFile.fromFile(_selectedImage!.path,
23       ));
24
25     var response =
26       await dio.post('http://192.168.176.134:8080/predict', data: formData);
27
28     if (response.statusCode == 200) {
29       var data = response.data;
30       setState(() {
31         if (data.containsKey('prediction')) {
32           _predictionResult = data['prediction'];
33         } else if (data.containsKey('error')) {
34           _predictionResult = data['error'];
35         } else {
36           _predictionResult = 'Please enter a valid image for prediction!';
37         }
38       });
39     } else {
40       print('Error predicting image: ${response.statusCode}');
41       setState(() {
42         _predictionResult = 'Please enter a valid image for prediction!';
43       });
44     }
45   }
46 }
47
48 @override
49 void initState() {
50   super.initState();
51 }
```

The code you provided appears to be a Flutter home screen app In the mobile application. It includes a function to select the image from Gallery or camera, and run the machine learning model on the selected image, and Show results for the expected disease.

6.2.4 Disease information





The code you provided appears to be a Flutter implementation of the general information screen. It includes information for 7 skin cancer diseases.

6.2.5 Medication Reminder

4:40

Medication Reminder →



Set Medicine

What is the name of the Medicine

Start Date

Time

End Date

How many times have you taken this medicine

Three Two One

Save Medication

⌵ :Scheduled Medications

⚙️ ⌚ ⓘ 🏠

5:38

تذكير الدواء →



تحديد الدواء

ما هو اسم الدواء؟

تاريخ البدء

الوقت

تاريخ الانتهاء

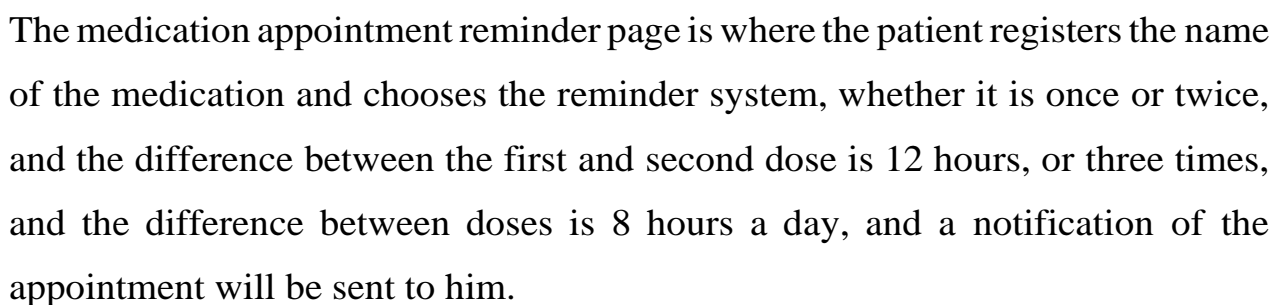
كم مرة تناولت هذا الدواء؟

ثلاثة اثنان واحد

حفظ الدواء

⌵ الأدوية المجدولة:

⚙️ ⌚ ⓘ 🏠



Chapter 7: Summary, Conclusions And Future Work

7.1 Conclusion:

In conclusion, this project successfully addresses the challenge of skin cancer detection by leveraging artificial intelligence (AI) techniques. The developed system, which allows users to take or upload photos of their skin, demonstrates the potential of AI in automating and improving disease detection processes. By applying advanced computer vision algorithms, the system accurately identifies and classifies various types of skin cancer based on visual symptoms. This user-friendly approach empowers individuals to make informed decisions promptly and take proactive measures to manage and mitigate health risks. The project highlights the effectiveness of AI in revolutionizing disease detection in healthcare and paves the way for future advancements in medical diagnostics.

7.2 Challenges:

Developing an AI system for skin cancer detection through photo analysis poses several challenges. Firstly, ensuring the accuracy of disease identification requires a diverse and comprehensive dataset that covers various types of skin cancer, stages, and skin tones. Collecting and labeling such a dataset can be time-consuming and resource-intensive. Secondly, managing variations in lighting, image quality, and background clutter introduces complexities in feature extraction and model training. Preprocessing techniques must be employed to address these challenges. Lastly, integrating the system with user-friendly interfaces that allow for photo capture or upload from the gallery requires careful design to ensure seamless and intuitive user experiences.

7.3 Future Work:

- 1-Integration of Deep Learning Models: Expanding the project's capabilities by integrating more advanced deep learning models could lead to improved disease detection accuracy.
- 2-Using user feedback and testing to identify areas for improvement and help guide the development of new features and improvements to existing ones.
- 3-Integrating the application with other technologies such as augmented reality and virtual assistants to further enhance the functionality and usability.
- 4- Adding new languages and to expand the reach of the application to a wider patients

Chapter 8 : References

Researches

1. Bhatt, Harsh, et al. "State-of-the-art machine learning techniques for melanoma skin cancer detection and classification: A comprehensive review." *Intelligent Medicine* 3.03 (2023): 180-190.
2. Shah, Aarushi, et al. "A comprehensive study on skin cancer detection using artificial neural network (ANN) and convolutional neural network (CNN)." *Clinical eHealth* (2023).
3. Khater, Tarek, et al. "Skin cancer classification using explainable artificial intelligence on pre-extracted image features." *Intelligent Systems with Applications* 20 (2022): 200275.

Dataset

<https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000>

APPS

1. <https://play.google.com/store/apps/details?id=com.molescope>
2. <https://play.google.com/store/apps/details?id=com.miiskin.android>
3. <https://play.google.com/store/apps/details?id=com.rubytribe.skinvision.ac>
4. <https://play.google.com/store/apps/details?id=com.visualdx>

Diseases

1. <https://www.mayoclinic.org/diseases-conditions/melanoma/symptoms-causes/syc-20374884>
2. https://en.wikipedia.org/wiki/Melanocytic_nevus
3. <https://www.mayoclinic.org/diseases-conditions/basal-cell-carcinoma/symptoms-causes/syc-20354187>
4. <https://www.mayoclinic.org/diseases-conditions/seborrheic-keratosis/symptoms-causes/syc-20353878>
5. <https://www.mayoclinic.org/diseases-conditions/actinic-keratosis/symptoms-causes/syc-20354969>
6. <https://en.wikipedia.org/wiki/Dermatofibroma>
7. <https://www.healthline.com/health/pyogenic-granuloma>
8. <https://my.clevelandclinic.org/health/diseases/17480-squamous-cell-carcinoma>
9. <https://my.clevelandclinic.org/health/diseases/21827-mycosis-fungoides>