|  |
| --- |
|  |
| Skin Cancer Detection  Application |
|  |

Content:

Chapter 01. Introduction 3

Chapter 02. Background 7

2.1 Mobile Application: 7

2.2 Machine Learning 8

Chapter 03. Literature Review 13

*3.1 15*

*3.2 15*

*3.3 15*

*3.4 15*

*3.5 15*

***Chapter 04. Implementation*** *17*

***Data Gathering:*** *17*

***Conclusion:*** *41*

***Reference:*** *41*

******

**CHAPTER 1**

**Introduction**

# ***Chapter 01. Introduction***

Skin cancer is one of the most common forms of cancer, affecting millions of people worldwide. Early detection is crucial for effective treatment and improved outcomes. Introducing the revolutionary Skin Cancer Detection App, an AI-powered solution that empowers individuals to take charge of their skin health. This state-of-the-art mobile application harnesses the power of cutting-edge technology to provide users with a comprehensive platform for skin analysis, personalized risk assessment, and seamless communication with dermatologists.

Developed by a team of medical experts, computer scientists, and healthcare innovators, the Skin Cancer Detection App aims to revolutionize the way people approach skin health. By combining advanced AI algorithms with a user-friendly Flutter-based interface, the app offers a convenient and accessible solution for individuals to take proactive steps in managing their skin conditions, from benign moles to potential melanomas.

At the heart of the Skin Cancer Detection App lies a powerful AI engine that leverages advanced machine learning algorithms to analyze the user's skin in unprecedented detail. By utilizing computer vision and deep learning techniques, the app can accurately detect and classify a wide range of skin conditions, from benign moles to potentially malignant lesions.

The AI-powered skin analysis process begins with the user capturing high-quality images of their skin using their smartphone's camera. The app's sophisticated image processing algorithms then analyze these photos, comparing them to a vast database of skin images and medical records to identify any concerning patterns or irregularities.

Through this advanced AI-driven analysis, the Skin Cancer Detection App can provide users with a comprehensive risk assessment, highlighting areas of concern and offering personalized guidance on the next steps to take. Whether it's a recommendation to consult a dermatologist or simple steps to monitor a mole, the app empowers users to take proactive measures in managing their skin health.

The Skin Cancer Detection App is built upon the powerful Flutter framework, a cross-platform technology that enables the seamless development of high-performance mobile applications for both iOS and Android devices. By leveraging the flexibility and efficiency of Flutter, the app delivers a consistent and engaging user experience across a wide range of smartphones and tablets.

The Flutter-based architecture of the Skin Cancer Detection App boasts several key advantages. Firstly, it ensures a fast and responsive user interface, with smooth animations and instant feedback, creating a visually stunning and intuitive experience for users. Additionally, the cross-platform nature of Flutter allows the app to be deployed on both iOS and Android platforms simultaneously, without the need for separate codebases, thereby reducing development time and costs.

Furthermore, the Skin Cancer Detection App's Flutter-based design enables effortless integration with a wide range of native device features, such as the camera, location services, and secure data storage. This allows the app to fully leverage the capabilities of modern smartphones, delivering a seamless and comprehensive solution for skin cancer detection and management.

By embracing the power of Flutter, the Skin Cancer Detection App provides users with a cutting-edge, user-friendly platform that combines the best of mobile technology with the latest advancements in AI-powered skin analysis. This strategic choice of technology ensures that the app remains at the forefront of innovation, delivering a responsive, efficient, and adaptable solution for individuals seeking to take control of their skin health.

Capture Your Skin: The Skin Cancer Detection App makes it easy for users to capture high-quality images of their skin using their smartphone's camera. With clear instructions and intuitive guidance, the app helps users frame their skin and ensure the photos are properly focused and well-lit, providing the AI engine with the best possible data to analyze.

AI-Powered Analysis: Once the skin images are captured, the app's advanced AI algorithms spring into action. Using state-of-the-art computer vision and deep learning techniques, the app meticulously examines the skin for any signs of abnormalities, comparing the user's photos against a vast database of medical imagery and dermatological records. This comprehensive analysis is the backbone of the app's skin cancer detection capabilities.

Personalized Insights: Based on the AI-powered analysis, the Skin Cancer Detection App provides users with personalized insights and risk assessments. The app clearly identifies any areas of concern, such as suspicious moles or lesions, and offers detailed information about the potential nature of these skin conditions – whether they are benign, potentially malignant, or require further medical attention. These personalized insights empower users to take proactive steps in managing their skin health.

Benign Skin Conditions: Many skin growths and changes are considered benign, meaning they are not cancerous and pose no immediate threat to one's health. These include common moles, freckles, skin tags, and age spots. While benign skin conditions may sometimes be cosmetically undesirable, they typically do not require extensive treatment beyond monitoring for any changes. The Skin Cancer Detection App can help users identify these harmless skin features and provide guidance on when to seek medical advice, reassuring individuals that not every skin change is a cause for concern.

Malignant Skin Conditions: On the other hand, malignant skin conditions, such as melanoma, basal cell carcinoma, and squamous cell carcinoma, are forms of skin cancer that require prompt medical attention. The Skin Cancer Detection App's advanced AI analysis can detect the early signs of these potentially life-threatening skin conditions, allowing users to take proactive steps towards diagnosis and treatment. By leveraging the app's personalized risk assessment, users can be empowered to seek timely medical intervention and increase their chances of successful treatment outcomes.

Normal Skin Conditions: In addition to benign and malignant skin conditions, the Skin Cancer Detection App can also identify normal skin features that do not require any special attention. This includes common blemishes, rashes, and other temporary skin changes that may be caused by environmental factors, allergies, or minor injuries. By providing users with a clear understanding of their skin's condition, the app helps individuals distinguish between harmless skin changes and those that warrant further medical evaluation, ensuring they seek the appropriate level of care and avoid unnecessary worry or intervention.

Personalized Guidance: The Skin Cancer Detection App seamlessly connects users with board-certified dermatologists, providing them with the opportunity to receive personalized guidance and expert medical advice. Through the app's secure in-app chat feature, users can share their skin analysis results, ask questions, and discuss any concerns they may have about their skin health. This direct line of communication with dermatologists empowers users to make informed decisions and take proactive steps in managing their skin conditions.

Comprehensive Evaluations: For users who require a more in-depth evaluation, the Skin Cancer Detection App can facilitate virtual consultations with dermatologists. During these online appointments, users can have their skin thoroughly examined by medical professionals, who can then provide a comprehensive assessment, diagnosis, and personalized treatment recommendations. This level of access to specialized care helps ensure that any potential skin issues are detected early and addressed appropriately, ultimately improving patient outcomes.

Seamless Referrals: In instances where the AI-powered analysis or virtual consultation indicates the need for further medical attention, the Skin Cancer Detection App can seamlessly refer users to the most suitable dermatologists in their local area. The app's extensive network of healthcare providers allows it to match users with experienced, board-certified specialists who can provide in-person examinations and treatment plans tailored to their individual skin health needs. This streamlined referral process ensures that users receive the appropriate level of care without added stress or confusion.

without added stress or confusion.******

**BackGround**

**CHAPTER 2**

# ***Chapter 02. Background***

# ***2.1 Mobile Application:***

Our skin cancer detection app utilizes Flutter, an open-source mobile application development framework developed by Google. By harnessing Flutter’s versatility, we deliver a seamless user experience across various devices, including iOS and Android smartphones, tablets, and desktop computers. The cross-platform capabilities of Flutter enable us to maintain a single codebase, simplifying deployment across different operating systems.

The key to Flutter's cross-platform prowess lies in its use of the Dart programming language, a fast, object-oriented, and type-safe language that is well-suited for building modern, scalable applications. Dart's ability to compile to both native machine code and high-performance JavaScript enables our app to deliver a truly native-like experience on every device, with smooth animations, responsive layouts, and lightning-fast performance. This ensures that our users, regardless of their device of choice, can enjoy a cohesive and intuitive interface that seamlessly adapts to their needs." Using the Flutter framework it was possible to build an accompanying application with which the model could be presented to the general public."[1]

In the dynamic landscape of Flutter development, a robust set of tools and libraries empowers developers to create immersive and feature-rich applications. At the forefront, the Image Picker widget stands as a pivotal component, enabling users to seamlessly select images from their device's gallery or capture new ones using the camera. This widget facilitates a user-friendly experience, essential for applications reliant on visual content.

Similarly, the URL Launcher plugin serves as a gateway to external resources, allowing Flutter apps to seamlessly navigate users to web pages, emails, phone calls, and other applications. This integration enhances app functionality by providing seamless access to external content and services, enriching the user experience.

In the realm of state management, the BLoC (Business Logic Component) pattern emerges as a powerful paradigm for managing application state and facilitating data flow. By decoupling business logic from UI components, BLoC enables maintainable, testable, and scalable Flutter applications. This architectural approach fosters code organization and separation of concerns, paving the way for robust and maintainable codebases.

To bring dynamic textual content to life, Flutter developers turn to the AnimatedText widget, which adds flair and interactivity to text elements. With support for various animations such as fading, scaling, and sliding, AnimatedText enhances user engagement and visual appeal, making Flutter applications more captivating and interactive.

Finally, Dio emerges as a versatile HTTP client for Flutter, offering a straightforward and flexible solution for making network requests. Dio's intuitive API, support for interceptors, and robust error handling capabilities make it a preferred choice for fetching data from remote servers and integrating with RESTful APIs. With Dio, Flutter developers can effortlessly manage network requests, handle responses, and ensure a smooth user experience in their applications.

In the realm of Flutter development, Firebase emerges as a pivotal backend solution, offering a comprehensive suite of services that streamline various aspects of app development. Firebase Authentication stands as the cornerstone, facilitating seamless user authentication through multiple methods such as email/password, social sign-ins, and phone number verification. Integrating Firebase Authentication into Flutter applications ensures robust user management and authentication workflows, fostering secure user interactions.

Complementing Firebase Authentication, Firebase Cloud Storage serves as a reliable repository for storing and retrieving user-generated content, including images, videos, and documents. This cloud-based storage solution offers scalability, security, and ease of integration, empowering Flutter developers to implement efficient file storage mechanisms without the hassle of managing complex server infrastructures.

At the heart of Firebase integration lies Firebase Core, which provides the fundamental framework for initializing and configuring Firebase services within Flutter applications. By incorporating Firebase Core, developers gain streamlined access to a plethora of Firebase functionalities, including Analytics for tracking user engagement and behavior patterns. This foundational layer ensures seamless integration of Firebase services, allowing developers to focus on crafting exceptional user experiences while Firebase handles the backend intricacies. Together, Firebase Authentication, Firebase Cloud Storage, and Firebase Core form a robust ecosystem within Flutter, empowering developers to build secure, scalable, and feature-rich applications with unparalleled ease and efficiency.

# ***2.2 Machine Learning***

At the heart of our skin cancer detection app lies a powerful artificial intelligence (AI) system that leverages advanced image classification techniques to analyze user-submitted skin lesion images. By integrating state-of-the-art deep learning models, we have developed a robust and accurate solution capable of differentiating between benign, premalignant, and malignant skin conditions with a high degree of precision.

The foundation of our AI-powered image classification engine is a convolutional neural network (CNN) architecture, specifically the MobileNetV3 model, which has been optimized for efficient deployment on mobile devices. This lightweight and high-performing model allows us to execute the complex image analysis algorithms directly on the user's smartphone, ensuring fast and responsive results without the need for a constant internet connection or server-side processing.

The training of our CNN" Early detection using smartphones is carried out by giving the smartphone the ability to recognize objects with skin cancer characteristics. The convolution neural network (CNN) is often used in disease detection and classification."[2] model was a meticulous process, where we carefully curated a diverse dataset of skin lesion images, encompassing a wide range of conditions, including melanoma, basal cell carcinoma, squamous cell carcinoma, and benign growths. By leveraging the power of transfer learning, we were able to fine-tune a pre-trained MobileNetV3 model, enabling it to learn the unique visual patterns and characteristics associated with different skin cancer types and other skin conditions.

The trained model is then seamlessly integrated into the skin cancer detection app, allowing users to simply capture an image of their skin lesion using their device's camera. The AI engine then instantly analyzes the image, compares it against the trained dataset, and provides the user with a detailed classification result, indicating the likelihood of the lesion being benign, premalignant, or malignant. This rapid feedback allows users to make informed decisions about seeking further medical advice or monitoring the lesion over time, ultimately promoting early detection and improved health outcomes.

*Choosing MobileNetV3 for Efficient Image Analysis:*

When selecting the deep learning model to power the image classification capabilities of our skin cancer detection app, we carefully evaluated several state-of-the-art architectures. Ultimately, we chose to integrate the MobileNetV3 model due to its exceptional performance, efficiency, and suitability for mobile deployment.

MobileNetV3 is the latest iteration of the popular Mobile Net family of convolutional neural network models, developed by researchers at Google. This innovative architecture has been specifically designed to deliver high-accuracy results while maintaining a compact model size and lightning-fast inference times, making it an ideal fit for our skin cancer detection app running on users' smartphones.

Unlike larger, more computationally intensive models, MobileNetV3 leverages a series of efficient building blocks, including depth-wise separable convolutions, squeeze-and-excitation modules, and novel activation functions. These optimizations allow the model to achieve state-of-the-art performance on a wide range of mobile vision tasks, including image classification, object detection, and semantic segmentation, all while remaining highly efficient and compact.

By integrating MobileNetV3 into our app, we can ensure that users receive accurate and timely skin cancer detection results without the need for a constant internet connection or server-side processing. The model's ability to run directly on the user's device, without the latency and privacy concerns associated with cloud-based solutions, is a key advantage that aligns with our goal of providing a seamless and user-centric experience.

*Training the CNN Model for Skin Cancer Detection:*

Data Collection and Curation: The foundation of our robust skin cancer detection model lies in the careful curation of a diverse dataset of skin lesion images. Our team meticulously compiled a comprehensive collection of images spanning a wide range of skin conditions, including melanoma, basal cell carcinoma, squamous cell carcinoma, and various benign growths. This diverse dataset ensures that our convolutional neural network (CNN) model can learn the unique visual patterns and characteristics associated with different skin cancer types, as well as non-cancerous lesions, to provide accurate and reliable classifications.

Transfer Learning and Fine-tuning: To leverage the power of deep learning while maintaining efficient performance on mobile devices, we employed a transfer learning approach in training our skin cancer detection model. Starting with a pre-trained MobileNetV3 architecture, we fine-tuned the model's weights and parameters using our curated dataset of skin lesion images. This process allowed us to harness the robust feature extraction capabilities of the MobileNetV3 model, while tailoring the final layers to the specific task of skin cancer classification. By leveraging transfer learning, we were able to achieve state-of-the-art accuracy without the need to train a CNN model from scratch, which would have been computationally intensive and resource-intensive.

Model Optimization and Evaluation: Once the CNN model was trained and fine-tuned, we conducted a rigorous evaluation process to ensure its accuracy, reliability, and suitability for real-world deployment on mobile devices. This involved extensive testing on held-out validation and test sets, where the model's performance was measured across various metrics, such as classification accuracy, precision, recall, and F1-score. Additionally, we optimized the model's size and computational requirements to ensure lightning-fast inference times and efficient resource utilization on users' smartphones, without compromising the quality of the skin cancer detection results.

*Providing Patients with Accurate Scan Results:*

Instant Feedback on Skin Lesions: One of the key features of our skin cancer detection app is its ability to provide users with immediate feedback on the potential malignancy of their scanned skin lesions. Once a user captures an image of a suspicious growth or mole, our advanced AI-powered image classification engine swiftly analyzes the visual characteristics and compares them against our extensive database of skin conditions. Within seconds, the app delivers a detailed report, indicating the likelihood of the lesion being benign, premalignant, or malignant.

Empowering Patients to Seek Medical Advice: By equipping users with this valuable information, our skin cancer detection app empowers individuals to make informed decisions about their health and take the necessary steps to seek professional medical advice. The app's clear and actionable results help users understand the urgency of their condition, whether it requires immediate attention from a dermatologist or simply regular monitoring over time. This early detection capability is crucial, as it can significantly improve the chances of successful treatment and recovery, ultimately saving lives.

Ongoing Skin Health Monitoring: Our skin cancer detection app also provides users with the ability to monitor their skin health over time, allowing them to track the progression or changes in their skin lesions. Users can easily capture multiple images of the same area and compare the results, enabling them to detect any concerning changes that may require further medical evaluation. This functionality encourages regular self-examination and proactive management of skin health, fostering a culture of prevention and early intervention that can significantly improve patient outcomes.

*Image Processing and Augmentation Libraries:*

os: This library is used for interacting with the operating system, allowing you to perform actions like creating directories, listing files in a directory, and managing paths.

PIL: Image Opening and Saving: You’re using Image. open() to load images and image. save() to save them after conversion or processing.

Image Conversion: The convert() method is used to change image modes, for example, converting RGBA images to RGB.

cv2 (OpenCV): Image Reading: cv2.imread() loads an image from a file.

Image Resizing: cv2.resize() changes the size of an image to the specified dimensions.

Image Transformation: Functions like cv2.flip() and cv2.rot90() are used for flipping and rotating images, respectively.

Image Enhancement: Techniques like Gaussian blur (cv2.GaussianBlur()) and histogram equalization (cv2.equalizeHist()) are applied to improve image quality.

matplotlib.pyplot: Although not explicitly used in the code you provided, this library is often used for visualizing images and results during the development of image processing applications.

numpy: This library is essential for numerical computing in Python. It’s used for handling arrays and matrices, which are fundamental in image processing and machine learning tasks.

tensorflow: You’ve upgraded TensorFlow, which suggests you’re using deep learning models. TensorFlow provides the backend for Keras, which is a high-level neural networks API.

Convolutional Neural Network (CNN) for Skin Cancer Detection: A CNN is a deep learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and differentiate one from the other.

The preprocessing steps you’ve implemented are crucial for a CNN to work effectively. By converting images to a uniform size and enhancing their features, you’re ensuring that the CNN can learn from the most relevant information.

*Dataset Preparation:*

<https://www.kaggle.com/datasets/fanconic/skin-cancer-malignant-vs-benign>

split folders: This library is used to split your dataset into training, validation, and testing sets, which is a standard practice in machine learning to evaluate the performance of your model.

In addition to splitting the dataset into training, validation, and testing sets using split folders, we incorporated two additional folders, "normal" and "undefined," to enhance the dataset's diversity and robustness. These folders contain samples that represent normal and undefined instances, enriching the model's understanding of various scenarios it might encounter in real-world applications. By integrating these folders into the original dataset, we aim to prevent overfitting by providing the model with a more comprehensive range of data points to learn from. This adaptation ensures that the model can generalize effectively across different classes and scenarios, ultimately leading to improved performance and reliability during evaluation.

*Model Performance:*

Achieving 96% accuracy is an excellent result, but it’s also important to look at other metrics like precision, recall, and the confusion matrix to understand how well your model is performing across all classes (undefined, normal, benign, malignant).

# ***Chapter 03. Literature Review***

**CHAPTER 3**

**Literature Review**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **NO** | **Paper** | **Year** | **Steps** | **Algorithms** | **Accuracy** | **Dataset** |
| **1** | A New CNN-Based Deep Learning Model Approach for Skin Cancer Detection and Classification | 2023 | 1. Evaluate input images in different color spaces 2. Use segmentation methods to isolate ROI 3. Classify images with CNN-based architecture | CNN, MatConvNet | 85.9% - 86.3% | BCC and Akiec skin lesions |
| **2** | Advanced Skin Cancer Detection Using Deep Learning | 2023 | Comparison of DL models for  automatic skin cancer detection | CNNs (ResNetv2, VGG16, EfficientNet-B5, EfficientNet-B7) | Up to 84.22% | Not specified |
| **3** | Deep Convolutional Neural Network with TensorFlow and Keras to Classify Skin Cancer Images | 2023 | System development for skin cancer detection | CNN, TensorFlow, Keras Not specified | 85% | HAM10000 |
| **4** | Skin Cancer Detection and Classification Using Neural Network | 2024 | Neural network application for skin cancer classification | CNN | 87% | Not specified |
| **5** | Skin Cancer Detection: A Review Using Deep Learning Techniques | 2023 | Review of deep learning approaches for skin cancer detection | ANN, CNN, KNN, GAN | 80% | Not specified |

***3.1*** A New CNN-Based Deep Learning Model Approach for Skin Cancer Detection and Classification

This paper presents a decision support system based on convolution-based deep learning methods for distinguishing between Basal Cell Carcinoma (BCC) and Actinic Keratosis (Akiec). It involves evaluating input images in different color spaces, using segmentation methods like Geodesic Active Contour and Chan and Vese, and classifying images with the CNN-based MatConvNet-1.0-beta15 architecture. The proposed model achieved an accuracy and F1 score of 85.9% - 86.3%1.[3]

***3.2*** Advanced Skin Cancer Detection Using Deep Learning

This research compares the effectiveness of several deep learning models for automatic skin cancer detection using pre-trained CNN models like ResNetv2, VGG16, EfficientNet-B5, and EfficientNet-B7. The study found that EfficientNet-B7 provided the highest F1-score, reaching 84.22%2.[4]

***3.3*** Deep Convolutional Neural Network with TensorFlow and Keras to Classify Skin Cancer Images

The objective of this paper is to develop a system based on Deep Convolutional Neural Networks with TensorFlow and Keras that can detect skin cancer. The system was tested using the HAM10000 database, which consists of dermatoscopic images across 7 different classes of skin cancer3.[5]

***3.4*** Skin Cancer Detection and Classification Using Neural Network

This paper discusses the use of neural networks for the detection and classification of skin cancer. It emphasizes the potential of CNNs in medical image analysis and the importance of accurate classification in the diagnosis process4.[6]

***3.5*** Skin Cancer Detection: A Review Using Deep Learning Techniques

The paper provides a comprehensive review of classical deep learning approaches, such as artificial neural networks (ANN), convolutional neural networks (CNN), Kohonen self-organizing neural networks (KNN), and generative adversarial neural networks (GAN) for skin cancer detection5.[7]

***A logo of a pink cell

Description automatically generated***

**Implementation**

**CHAPTER 4**

### ***Chapter 04. Implementation***

Introducing our revolutionary skin cancer detection app, a cutting-edge fusion of Flutter technology and artificial intelligence. Designed to empower users with the ability to perform skin scans from the comfort of their own homes, our app revolutionizes the way we approach skin health.

With the prevalence of skin cancer on the rise, early detection is more critical than ever. Our app leverages the power of AI algorithms trained on vast datasets of dermatological images to provide accurate and timely assessments of skin abnormalities. By simply capturing a photo of the skin lesion in question, users receive instant feedback on whether it is benign, malignant, or normal.

The seamless integration of Flutter ensures a user-friendly experience across Android platforms, allowing for widespread accessibility. Our app prioritizes user privacy and security, ensuring that sensitive medical information remains confidential.

By democratizing access to skin cancer detection, our app empowers individuals to take proactive control of their health. Whether at home or on the go, users can confidently monitor their skin health with the guidance of advanced AI technology. Welcome to the future of skin cancer detection—where innovation meets convenience, all in the palm of your hand.

### ***Data Gathering:***

Gather a comprehensive dataset to train and validate a machine learning model for a skin cancer detection app, distinguishing between benign and malignant lesions.

Dataset Size and Diversity:

We collect a large and diverse dataset comprising images of both benign and malignant skin lesions.

Sources:

We research organizations to obtain high-quality, labeled images. Utilize publicly available datasets, such as the Kaggle:

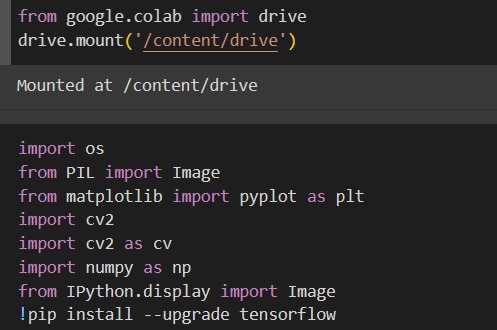
<https://www.kaggle.com/datasets/nodoubttome/skin-cancer9-classesisic>

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

Our Dataset:

<https://www.kaggle.com/datasets/fanconic/skin-cancer-malignant-vs-benign>

In addition to splitting the dataset into training, validation, and testing sets using split folders, we incorporated two additional folders, "normal" and "undefined," to enhance the dataset's diversity and robustness.



These lines are used to mount your Google Drive to the Colab environment, allowing you to access files from your Drive within Colab.os: Provides a way of using operating system dependent functionality.

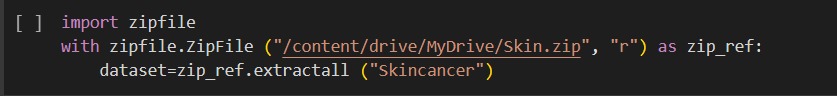
PIL.Image: Used for opening, manipulating, and saving many different image file formats.

matplotlib.pyplot: A collection of functions that make matplotlib work like MATLAB, used for creating static, interactive, and animated visualizations in Python.

cv2: OpenCV library for computer vision tasks.

numpy: Fundamental package for scientific computing with Python.

IPython.display.Image: Displays an image in the IPython notebook.The last line is a command that runs in the shell (noted by the !) which upgrades the TensorFlow library to the latest version using pip, Python’s package installer.



st a ZIP file.with zipfile.ZipFile("/content/drive/MyDrive/skin.zip", "r") as zip\_ref: This line opens the ZIP file located at /content/drive/MyDrive/skin.zip in read mode ("r") as a ZipFile object. The with statement ensures that the file is properly closed after its suite finishes, even if an error is raised at some point. dataset = zip\_ref.extractall("skincancer"): This line extracts all the contents of the ZIP file into a directory named "skincancer". The variable dataset is assigned the result of the extraction, although in this context, it’s not used later in the code.

A screen shot of a computer code

Description automatically generated

*Purpose:* The function extract images is designed to transfer image files from a specified source directory to a target directory. (applied to the rest of the folders (malignant, normal, undefined)

*Functionality:*

Import Statements: The os module is used for interacting with the operating system, and the shutil module provides functions for file operations.

Creating Target Folder: It checks if the target folder exists and creates it if it doesn’t.

Iterating Over Files: The function iterates over all files in the source directory.

Identifying Images: It checks if a file is an image by looking at its extension. The extensions considered as images are .png, .jpg, .jpeg, and .gif.

Moving Images: If a file is an image, it is moved to the target directory.

Example Usage: At the end of the snippet, there’s an example of how to use the function with specific paths for the source and target directories.

A screenshot of a computer screen

Description automatically generated

Import Statements: The os module is used for interacting with the operating system, and PIL.Image is used for opening, manipulating, and saving many different image file formats.

Path Definitions: path is the directory containing the original images, and jpg\_folder is the directory where the converted JPEG images will be saved.

Folder Creation: The script checks if the jpg\_folder exists and creates it if it doesn’t.

Image Conversion Loop: The script loops through each file in the path directory, checks if the file is an image with a specific extension, and then processes and saves it as a JPEG image in the jpg\_folder.

Image Processing: If an image is in RGBA format (which includes a transparency channel), it is converted to RGB format before saving, as JPEG does not support transparency.

Saving Images: Each image is saved with a .JPEG extension in the jpg\_folder.

(applied to the rest of the folders (malignant, normal, undefined)

A screenshot of a computer program

Description automatically generated

*Purpose:* The script resizes all image files in a specified folder to 224x224 pixels, which is a common size for input into neural networks. (applied to the rest of the folders (malignant, normal, undefined)

*Functionality:*

Import Statements: The cv2 module from OpenCV is used for image processing tasks, and the os module is used for interacting with the operating system.

Path Definitions: folder\_path is the directory containing the original images, and new\_folder\_path is where the resized images will be saved.

Resizing Parameters: The width and height variables set the dimensions for the resized images.

Folder Creation: The script checks if the new\_folder\_path exists and creates it if it doesn’t.

Image Processing Loop: The script iterates over each file in the folder\_path directory, checks if the file is an image, and then resizes and saves it in the new\_folder\_path.

*Purpose:* The function augment\_images enhances the dataset by creating augmented versions of the original images. This is crucial for training robust machine learning models, especially when the dataset is limited. (applied for all folders)

*Functionality:*

Import Statements: The script imports the necessary libraries for file handling and image processing.

Folder Paths: It defines the paths for the input and output folders containing the benign skin cancer images.

Augmentation Parameters: The script sets parameters for rotation and flipping the images.

A computer screen with text on it

Description automatically generatedImage Processing: For each image, the script applies rotations and flips, then saves the augmented images to the output folder.

Horizontal Flip: This part of the code applies a horizontal flip to the image. It uses the OpenCV library’s flip function with the flip code 1 to flip the image around the y-axis.

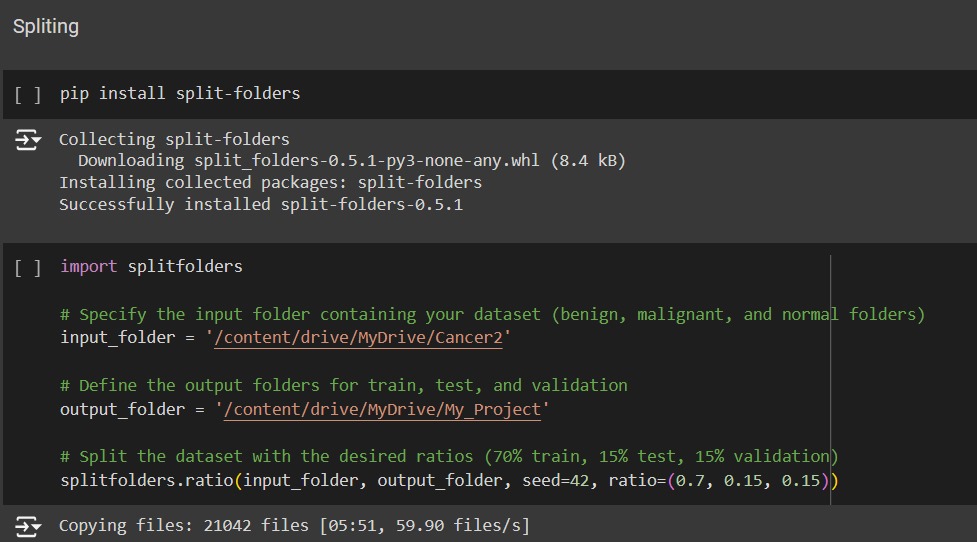
Vertical Flip: Similarly, a vertical flip is applied using the flip code 0, which flips the image around the x-axis.

Saving Augmented Images: The augmented images are saved in the specified output\_folder. The filenames are constructed to include the type of augmentation applied for easy identification.

Function Call: The augment\_images function is called with the paths to the input and output folders as arguments. This function will process all images in the input folder and save the augmented versions in the output folder.

Success Message: After the augmentation process is completed, a message is printed to confirm that the augmented images have been successfully saved.A screen shot of a computer program

Description automatically generated



Installation: The split-folders package is installed using pip. This package is used for splitting large datasets into training, validation, and test sets.

Importing Library: The splitfolders library is imported to use its functionality.

Dataset Paths: input\_folder is the path to the dataset that contains images of benign, malignant, and normal skin conditions. output\_folder is the path where the split datasets will be saved.

Splitting the Dataset: The ratio function from the splitfolders library is called to split the dataset into training, testing, and validation sets with the specified ratios. The seed parameter ensures that the split is reproducible.

*A screenshot of a computer program

Description automatically generatedPurpose:* The script converts color images from the "benign" folder into grayscale images and saves them in the “skincancer” folder. Grayscale images are often used in skin cancer detection because they reduce computational complexity and focus on the texture and shape of skin lesions.(applied for all folders)

*Functionality:*

Import Statements: The script imports os for directory operations, cv2 for image processing, and numpy for handling arrays.

Path Definitions: It defines the paths to the source and destination folders.

Image Processing Loop: The script reads each image, checks if it’s loaded correctly, converts it to grayscale, and saves it in the destination folder.

Error Handling: If an image fails to load, an error message is printed.

A screenshot of a computer

Description automatically generated

*Purpose:* The script is used to remove noise from grayscale images in the “benign” folder. Noise reduction is an important preprocessing step in image analysis, especially for medical images, as it helps in enhancing the quality of the images for better feature extraction and classification by the detection model. (apply for all folders)

*Functionality:*

Import Statements: The script imports os for directory operations and cv2 (OpenCV) for image processing.

Path Definitions: It defines the paths to the source folder containing grayscale images and the destination folder where denoised images will be saved.

Folder Creation: The script checks if the destination folder exists and creates it if necessary.

Image Processing Loop: The script reads each image, applies a Gaussian blur to denoise the image, and saves the denoised image in the destination folder.

Gaussian Blur: The cv2.GaussianBlur function applies a Gaussian blur with a kernel size of (5,5) and a standard deviation of 0. This helps in smoothing the image and reducing noise.

A screenshot of a computer

Description automatically generated

*Purpose:* The script enhances the contrast of denoised images using histogram equalization. This technique improves the visibility of features in medical images, which can be beneficial for the accuracy of skin cancer detection algorithms. (applied for all folders)

*Functionality*:

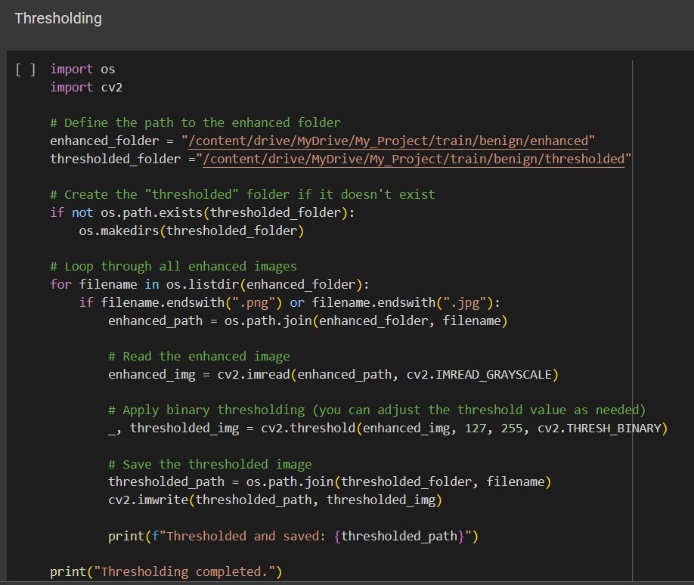
Import Statements: The script imports os for directory operations and cv2 (OpenCV) for image processing.

Path Definitions: It defines the paths to the source folder containing denoised images and the destination folder where enhanced images will be saved.

Folder Creation: The script checks if the destination folder exists and creates it if necessary.

Image Processing Loop: The script reads each denoised image, applies histogram equalization to enhance the contrast, and saves the enhanced image in the destination folder.

Histogram Equalization: The cv2.equalizeHist function is used to apply histogram equalization to the grayscale images.

**

*Purpose:* The script applies binary thresholding to the enhanced images. This technique is used to create a binary image from a grayscale image where all pixels that are above a certain threshold are set to the maximum value (white), and all other pixels are set to the minimum value (black).

*Functionality:*

Import Statements: The script imports os for directory operations and cv2 (OpenCV) for image processing.

Path Definitions: It defines the paths to the source folder containing enhanced images and the destination folder where thresholded images will be saved.

Folder Creation: The script checks if the destination folder exists and creates it if necessary.

Image Processing Loop: The script reads each enhanced image, applies binary thresholding with a threshold value of 127, and saves the thresholded image in the destination folder.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

Model Setup: The MobileNetV3Small model is loaded without its top layer to serve as the base model for feature extraction. The base model’s weights are frozen to prevent them from being updated during training.

Top Layer Addition: A new top layer is added to perform the classification task. It consists of a global average pooling layer followed by a dense layer with 512 units and ReLU activation, a dropout layer for regularization, and a final dense layer with 4 units and softmax activation corresponding to the four classes.

Model Compilation: The model is compiled with the Adam optimizer and categorical crossentropy loss function, which is suitable for multi-class classification tasks.

Training Data: The training data should be loaded using an appropriate method, which might include data augmentation or preprocessing steps.

Model Training: The model is trained on the loaded data. This part of the code will include specifying the number of epochs, batch size, and other training parameters.

Model Saving: After training, the model is saved for future use, such as making predictions on new data.

*Purpose:* This script extracts the final accuracy metric from the history object, which contains data about the model’s performance during training.

*Functionality:*

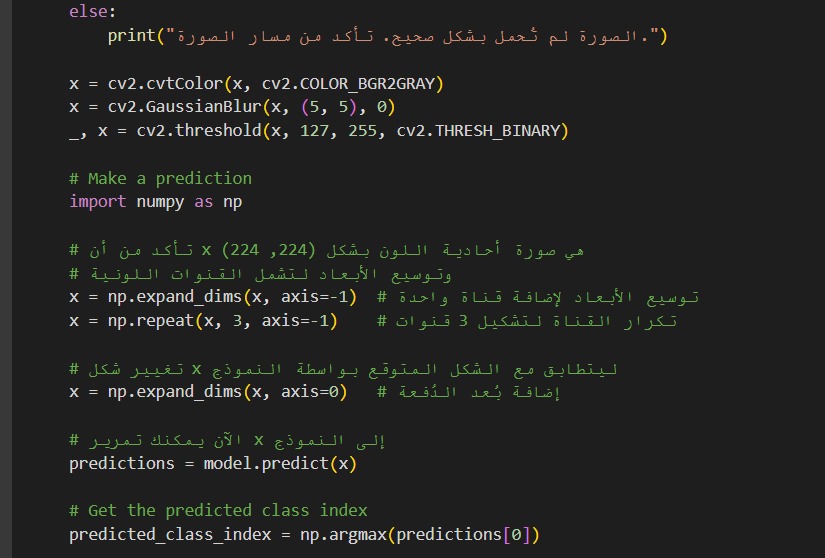
The history.history['accuracy'] retrieves the list of accuracy values recorded after each epoch during model training.

The [-1] index is used to access the last value in this list, which corresponds to the final accuracy after the last training epoch.

The print statement then outputs this final accuracy that equal to 96%



A computer screen shot of a program code

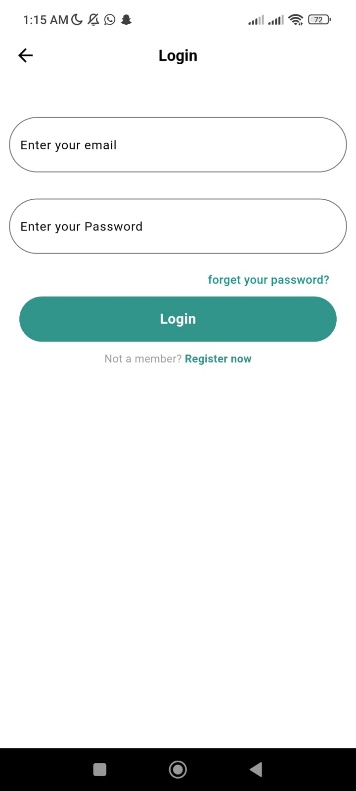
Description automatically generated

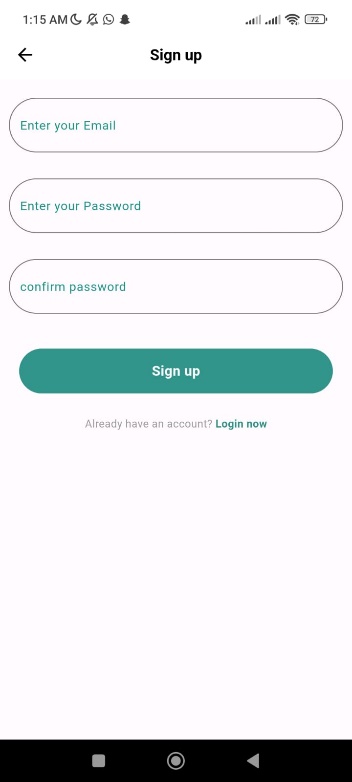
This start page disappears after seconds and that content app's name and move you to Login page or Signup page. As the following Figure:



**

The Logn-in and signup page includes a form where user can enter their login credentials, such as an email address and password. The form also includes options for users to reset their password or create a new account if they don't have one. As the following Figure:





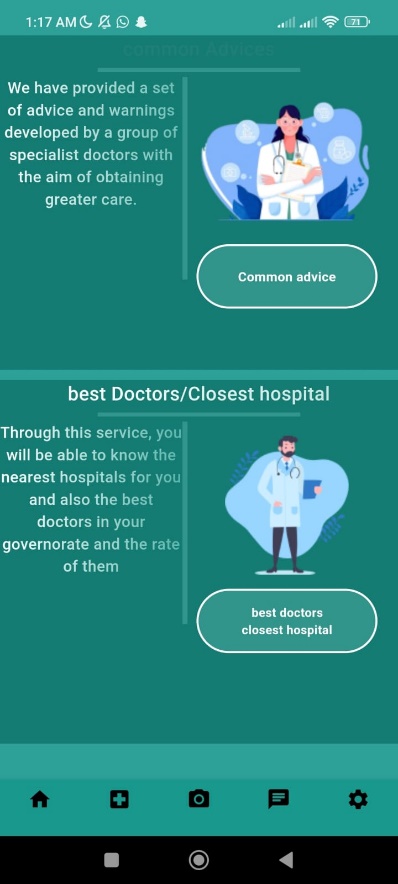
A screen shot of a computer screen

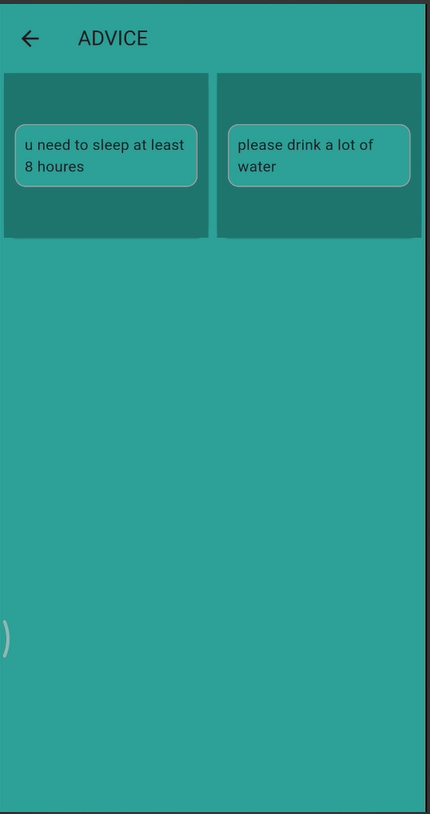
Description automatically generated



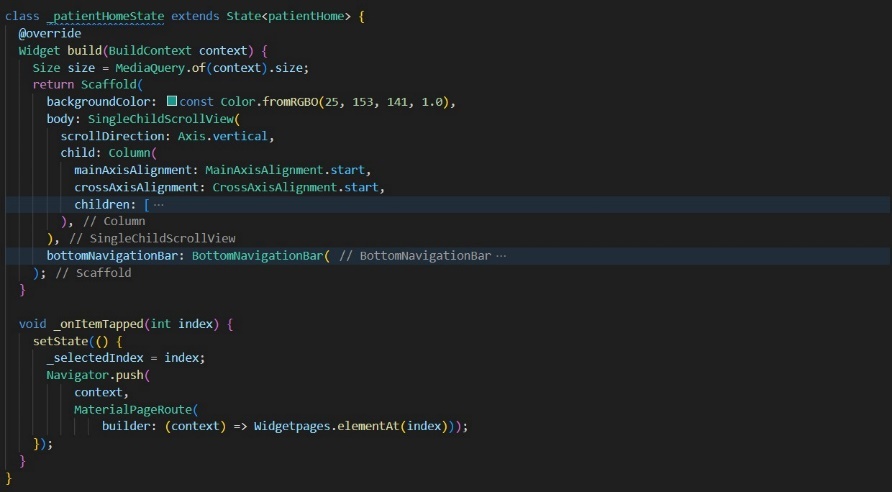
The main page for patients contains a part that includes advice provided by some doctors who reviewed the application and other, and the other part is about suggestions for some best hospitals and doctors to help the patient. As the following Figure:



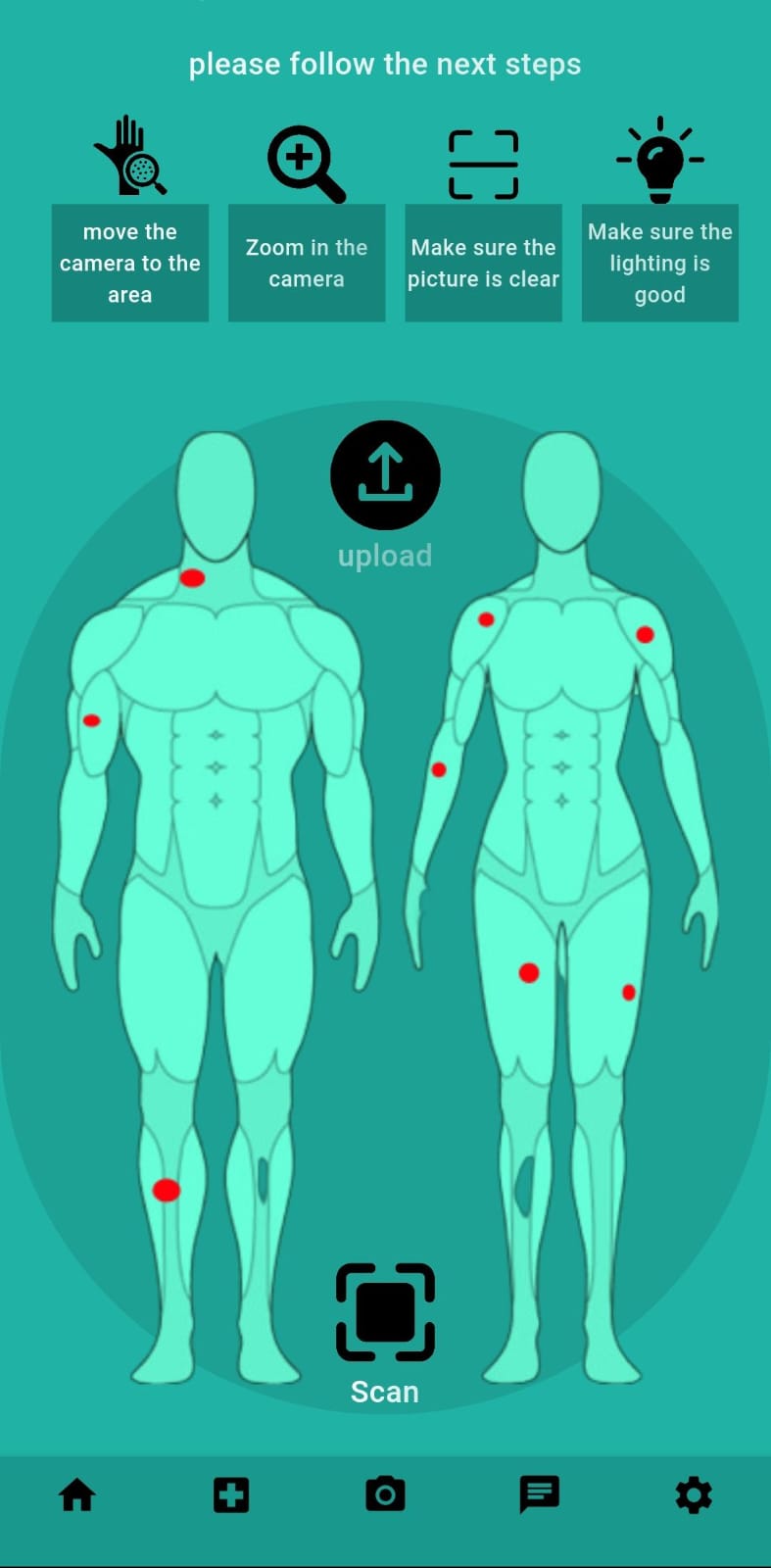


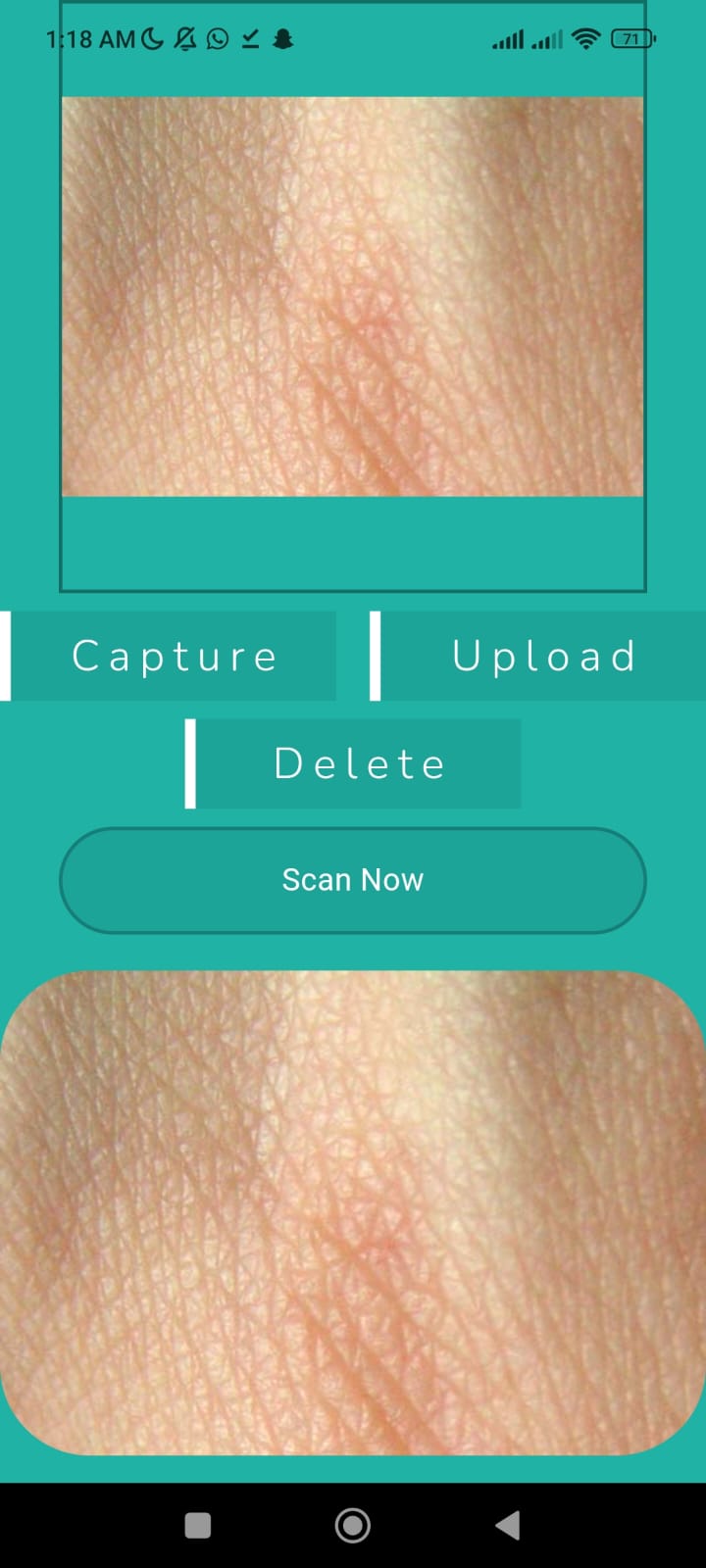






The gallery page or capture page allows uploading pictures of the patient to be examined. The examination is performed, and the result is shown to the patient to know the type of tumor: benign(low), malignant(high), or healthy. As the following Figure:





A screen shot of a computer

Description automatically generatedA screen shot of a computer screen

Description automatically generated

A computer screen shot of a program

Description automatically generated

A screen shot of a computer program

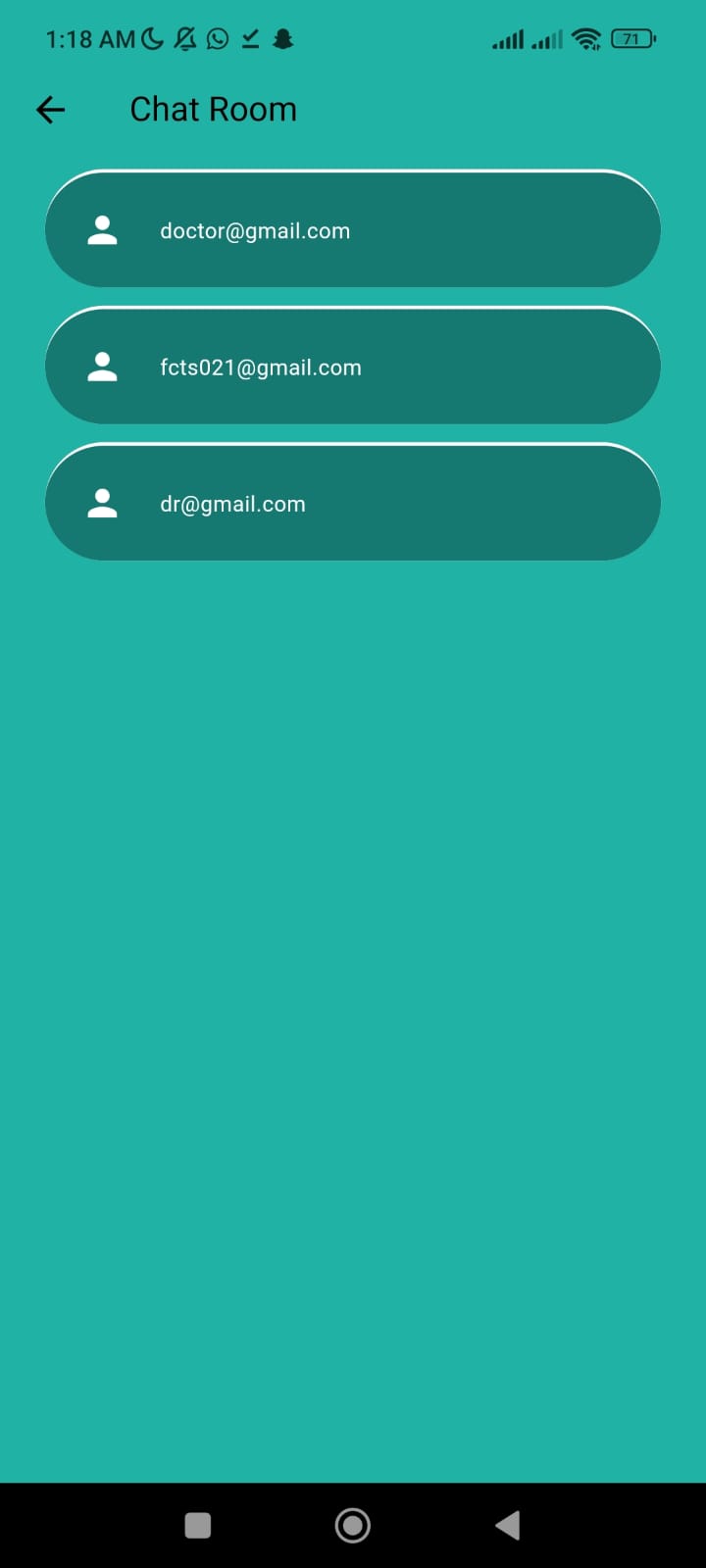
Description automatically generated

A black rectangular with colorful text

Description automatically generated

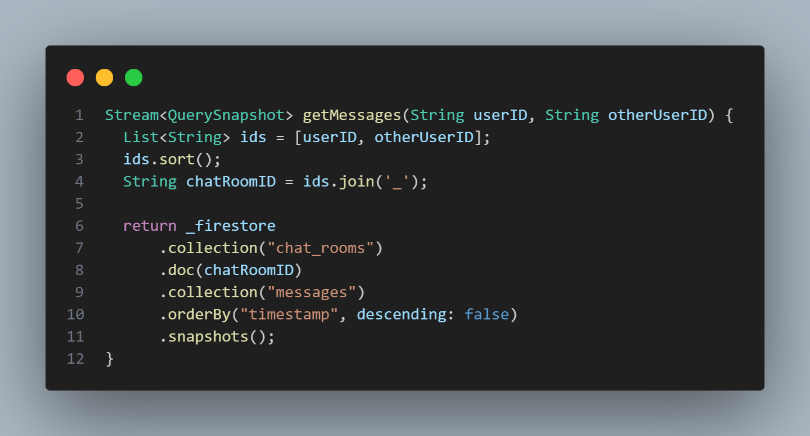
The chatting page is for the patients if they want to know more information about their case from a professional doctor. They pick a doctor based on their history and send them an image of their skin lesion. As the following Figure:







A screen shot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generatedA screen shot of a computer code

Description automatically generated

This page is to test the user experience and see if the patient has any comment that he would like to share with us. As the following Figure:

A screenshot of a phone

Description automatically generated



In this page, the user can control their account such as delete, update and logout. As the following Figure:

A screenshot of a phone

Description automatically generatedA screenshot of a login screen

Description automatically generatedA screenshot of a video game

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

The doctor's home page contains conversations with patients, following them up, and the possibility of giving advice. As the following Figure:

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA group of people in medical uniforms

Description automatically generated

A screen shot of a computer screen

Description automatically generated



#### **Conclusion:**

In conclusion, the development and implementation of our skin cancer detection app mark a significant stride in the intersection of healthcare and technology. Through harnessing the power of artificial intelligence and machine learning algorithms, we have created a tool that empowers individuals to take control of their skin health with unprecedented ease and accuracy.

Our app not only streamlines the process of early detection but also serves as a crucial educational resource, raising awareness about the importance of regular skin checks and sun protection measures. By providing users with timely and reliable assessments of their skin lesions, we aim to facilitate early intervention and ultimately save lives.

Moreover, the widespread accessibility of our app ensures that individuals from all walks of life can benefit from its capabilities, bridging gaps in healthcare access and empowering underserved communities to prioritize their skin health.

##### ***Reference:***

[1] Länsberg, S., & Manfredsson, A. (2022). Detection of malignant melanomas using neural networks. Odr.chalmers.se. <https://odr.chalmers.se/items/b4aac4e4-13a9-4b1e-8d73-556d0d720050>

[2] Hartanto, C. A., & Wibowo, A. (2020). Development of Mobile Skin Cancer Detection using Faster R-CNN and MobileNet v2 Model. 2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE). <https://doi.org/10.1109/icitacee50144.2020.9239197>

[3] Halit Çetiner, & Sedat Metlek. (2023). A New CNN-Based Deep Learning Model Approach for Skin Cancer Detection and Classification. Studies in Computational Intelligence, 177–199. <https://doi.org/10.1007/978-3-031-46341-9_7>

[4] Alzamel, M., Seba Alhejaili, Alhumaidhi, F., Joud Alismail, Lama Almubarak, Halah Altammami, Iliopoulos, C., & Lim, Z. (2023). Advanced Skin Cancer Detection Using Deep Learning. Communications in Computer and Information Science, 267–278. <https://doi.org/10.1007/978-3-031-34204-2_23>

[5] Amine, A. (2020). Deep Convolutional Neural Network with TensorFlow and Keras to Classify Skin Cancer Images. Scalable Comput. Pract. Exp. <https://www.academia.edu/89566773/Deep_Convolutional_Neural_Network_with_TensorFlow_and_Keras_to_Classify_Skin_Cancer_Images>

[6] Hermosilla, P., Soto, R., Vega, E., Suazo, C., & Ponce, J. (2024). Skin Cancer Detection and Classification Using Neural Network Algorithms: A Systematic Review. Diagnostics, 14(4), 454. <https://doi.org/10.3390/diagnostics14040454>

[7] Dildar, M., Akram, S., Irfan, M., Khan, H. U., Ramzan, M., Mahmood, A. R., Alsaiari, S. A., Saeed, A. H. M., Alraddadi, M. O., & Mahnashi, M. H. (2021). Skin Cancer Detection: A Review Using Deep Learning Techniques. International Journal of Environmental Research and Public Health, 18(10), 5479. <https://doi.org/10.3390/ijerph18105479>