Ministry of Higher Education
Higher transit institute
Information Systems

Classification of Skin Cancer Using Machine Learning

Using (Machine Learning & Android)

A Thesis Submitted to the Computer Science department in Partial Fulfillment of the Requirements for the award of degree of Bachelors of Science in Computer Science

Prepared by

CS4-07

Supervised by

Prof. Dr. Mohamad Farag

Dr. Magdy Abdulghani Zaid

Dr. Ibrahim Abdul-Latif

4th year computer science

2022-2023

Classification of Skin Cancer Using Machine Learning

Using (Machine Learning & Android)

A Thesis Submitted to the Computer Science department in Partial Fulfillment of the Requirements for the award of degree of Bachelors of Science in Computer Science

Prepared by

CS-07

Mahmoud Ahmad Mahmoud

Mohamed Hassan Mohamed

Eslam Mohamed Ebrahem

Mahmoud Ramadan Hafez

Mohamed Hassan Mousa

Nada Osama Elsayed

Supervised

DR. Magdy Abdelghani

Eng. Asmaa Rizk

Eng. Karim Abu Al-Azm

4th year computer science

2022-2023

Book index

Table Of Content ……..……...I

List Of Figure……..……..……IV

Acknowledgment……..……VI

Abstract ……..…….……VII

Chapter 1: Introduction

1.1. overview.....1

1.2. Introduction 2

1.3. Motivation 2

1.4. Problem statement 3

1.5objective.....4

1.6.schedule…….....4

1.7. thesis structure ……..…….....5

CHAPTER 2: Background and Literature Survey

2.1. Introduction…….....6

2.2. Cancer…….....6

2.3. Anatomy of skin cancer.....8

2.4. Architecture of the skin cancer.....9

2.5. skin neoplasm.....10

2.6. Risk factors of skin cancer.....11

2.7. Staging of skin cancer.....12

2.8. TNM staging.....12

2.9. Overall grade.....12

2.10. Method of assessment of skin abnormality.....13

2.11. Management of a clinically doubtful skin cancer.....13

2.12. Statistics of skin Cancer.....13

2.13. pattern classification techniques.....15

2.14. Statistical Methods.....15

2.15. Structural Methods.....16

Chapter 3: Related Work

3.1. Related Work.....20

CHAPTER 4: System analysis

4.1. pre processing…….....21

4.2. Use case…….....23

4.3. Context Diagram…….....25

4.4. Data flow Diagram (DFD)……..…….....….....27

4.5. Entity Relationship Diagram(ERD)……..……..…….....28

CHAPTER 5: System design

5.1. System Design ……31

5.2. Work of The System…….....32

5.2.1. Database …….....32
5.2.2. Machine learning …….....33
5.2.3. Django Framework …….....34
5.2.4.1. API…….....40
5.2.4.2 .Rest Framework API…..….....42
5.2.4.3. Serialize…….....43
5.2.5. Java…….....44
5.2.5. Android…….....45

CHAPTER 6: Implementation

6.1. Detection Solution…….....53
6.2. Detection Model …….....53
6.3. Loading the Data…….....55
6.4. Data Scaling…….....57
6.5. Data Augmentation…….....60
6.6. Building Deep Learning Model…….....62
6.7. java Code…….....87
6.8. Output …….....95

Chapter 7 : Conclusion and Future Work

7.1 Conclusion.....109
7.2 Future Work.....109

Chapter 8: Reference

8.1. Reference.....110

List Of Figures

1.1 Figure 1.1: Classification of unknown tumor 3

2.1 Figure2.1: Structural changes in cell divisions 7

2.2 Figure2.2: Normal and cancer cell divisions 7

2.3 Figure 2.3: shows benign and malignant tumors and how metastasis takes place.....8

Figure 2.4: Areas of skin 9

2.5: Figure 2.5: Most Common skin Cancer Sites 14

2.6:Figure 2.6: Skin cancer kills 14

2.7Figure 2.7: The statistical pattern recognizer 15

Figure 2.8: The structural pattern recognizer.....16

Figure 2.9: General architecture MLP 18

4.1 pre processing Diagram 22

4.2 Use case.....24

4.3 Context Diagram 26

4.4 Data flow Diagram (DFD) 28

4.5 Entity Relationship Diagram (ERD).....29

5.1 API.....41

5.2 Features of android OS.....45

5.3 Custom ROMs 47

6.1 output 95

6.2 output 96

6.3 output 97

6.4 output 98

6.5 output 99

6.6 output 100

6.7 output 101

6.8 output 102

6.9 output 103

6.10 output 104

6.11 output 105

6.12 output 106

6.13 output 107

6.14 output 108

Acknowledgment

First, regarding Prof. Dr. Mohamed Sayed Farag:

We would like to express our deepest gratitude for your continued support and guidance during our undergraduate studies. Always ready to give advice and help, I spared no effort in helping us achieve this great achievement in my academic life. I wish you all the best in your great work.

Secondly, regarding the project supervisor, Dr. Magdy Abdel-Ghany Zaid:

We are very grateful for your dedication and guidance during the graduation project. You were always available to discuss and answer any questions, and you never hesitated to help us reach positive results. You were the cornerstone of our success in this project, and we will do my best to build on the experiences gained from you on the projects." future. Thank you very much for everything you've done for us.

Third, regarding the engineers supervising the project, Eng. Asmaa Rizk

Eng. Karim Abu Al-Azm:

Thank you so much for your guidance, support and encouragement throughout the graduation project. Your insights, feedback and advice have been invaluable to us. Your dedication and hard work have inspired us to achieve the best possible results. It has been an honor to be under your guidance, and I will always cherish your guidance. I wish you all the best and happiness." in your future endeavours.

Abstract

Skin cancer is a serious and potentially deadly disease that affects millions of people around the world. Early detection and treatment are key to improving patient outcomes and reducing the risk of complications. In this project, we developed a mobile application that uses machine learning algorithms to detect skin cancer in patients. The application is designed for use by both patients and medical professionals, providing a user-friendly interface and powerful diagnostic tools. In addition to the machine learning model, the application includes a range of features to support patient care, such as progress tracking, sun safety tips, and multi-language support. For medical professionals, the application includes decision support tools, clinical data analysis, and a patient referral network. The video chat feature allows doctors to connect with their patients remotely, improving access to care and reducing the need for in-person visits. The application has the potential to revolutionize skin cancer diagnosis and treatment, improving patient outcomes and reducing healthcare costs Skin cancer is a growing public health concern that affects people of all ages and backgrounds. In this project, we developed a mobile application that leverages machine learning algorithms to detect skin cancer in patients quickly and accurately. The application provides users with a range of features, including a skin cancer risk assessment tool, sun safety tips, and personalized recommendations for prevention and early detection. Medical professionals can use the application to access patient records, make diagnoses, and share information with other medical professionals. The application includes a secure messaging feature, enabling medical professionals to communicate with patients and other healthcare providers in real-time. Additionally, the application includes a telemedicine feature, enabling medical professionals to conduct video consultations with patients from anywhere. The application is designed to be user-friendly and accessible, with multi-language support and intuitive navigation. The application has the potential to improve patient outcomes, reduce healthcare costs, and increase access to care for patients with skin cancer.

Chapter 1

INTRODUCTION

1.1. overview

1.2. Introduction

1.3. Motivation

1.4. Problem statement

1.5. objective

1.6. schedule

1.7. thesis structure

Chapter 1: Introduction

1.1 overview:

The skin cancer scan app is designed to help users detect skin cancer at an early stage by providing a machine learning-based classification of skin lesions. The app allows users to take a photo of their skin lesion, which is then analyzed by the machine learning model to provide a prediction of whether the lesion is benign or malignant.

In addition to the skin lesion classification feature, the app also provides a range of other functionalities for both patients and doctors. These include:

- User authentication: Patients and doctors can create an account and log in to access the app's features.

- User profile: Patients and doctors can create and manage their profiles, which include personal information and medical history.

- Skin lesion tracking: Patients can track their skin lesions over time by taking regular photos and comparing the results.

- Video consultation: Patients can connect with doctors via video chat for a remote consultation.

- Appointment booking: Patients can book appointments with doctors and view their appointment history.

- Notification system: The app provides notifications to remind patients about appointments, check-ups, and other important events.

- Educational resources: The app provides educational resources about skin cancer, including information about prevention, detection, and treatment.

Overall, the skin cancer scan app is a comprehensive tool that can help patients and doctors in the early detection and management of skin cancer.

1.2 Introduction:

Medical image processing and analysis have been active fields of research for more than 25 years. Currently, a method-driven modeling approach dominates the field of biomedical image processing as algorithms for registration, segmentation, classification and measurements are developed on a methodological level. The future of medical image processing is, however, seen in task-oriented solutions that are integrated into diagnosis, intervention panning, and therapy and follow-up studies.

In 2001, the Working Group on Medical Image Processing (WG-MIP) was established within the European Federation for Medical Informatics (EFMI) to foster this integration. In particular, the WG-MIP aims at supporting the discussion of how to integrate decision support by means of medical image processing into clinical practice, including the important topics of clinical evaluation, standardization and technology transfer.

Successful technology transfer is based on evidence, i.e. research and approvals. However, evidence in complex domains such as medicine, pharmacy and medical image processing cannot be created without appropriate evaluation methods and validation platforms. The latter are serving as an environment for testing the performance of novel methods and systems in terms of absolute measurement, comprehensive benchmarking and detailed comparison with known and proven methods and systems.

Skin cancer is defined as an abnormal growth of skin cells, and it often occurs when the skin is exposed to harmful sunlight, but skin cancer can occur in different areas of the body that may not normally be exposed to sunlight, such as the genital area and buttocks.

There are three main types of skin cancer:

1. Basal cell carcinoma
2. Squamous cell carcinoma
. Melanoma skin cancer

1.3 Motivation:

Skin cancer develops primarily on areas of skin exposed to the sun, including the scalp, face, lips, ears, neck, arms, chest, and hands, and on the legs in women. But it can affect areas of the skin that rarely see daylight (such as the palms of the hands, under the nails, the spaces between the toes and under the toenails, and the genital area).

Skin cancer affects all skin tones, including dark skin. And when skin cancer occurs in dark-skinned people, it is more likely to occur, usually in areas that are not exposed to sunlight.

the reasons:

Skin cancer is caused by a mutation in DNA. A for healthy skin cells. The mutations cause the cells to grow out of control and as a result a mass of cancerous cells forms.

Much of the damage to DNA in skin cells results from exposure to ultraviolet rays, found in sunlight or in commercial tanning lamps. There are also other factors that may contribute to the risk of skin cancer, such as exposure to toxic substances such as arsenic or having a weakened immune system.

Skin cancer begins in the dermis, the thin layer that provides a protective covering for the skin cells in your body. The dermis contains three main types of cells.

Squamous cells: They are found just under the outer surface of the skin, and act as the lining of the inner skin.

Basal cells: responsible for the production of new skin cells, located under the squamous cells.

Melanocytes: They produce the pigment melanin, which gives skin its natural colour. The secretion of melanin pigment increases when the skin is exposed to the sun, in order to protect the deeper layers of the skin.

UV rays and other possible causes that may cause skin cancer.

1.4 Problem statement:

Given the predefined datasets of different skin tumor samples, the input skin image sample (unidentified tumor) needed to be classified into benign or malignant as shown in Figure 1.1. The main problem with dataset classification is the accuracy results that classify samples into benign or malignant by using different classifiers in order to reach an accurate diagnosis of skin cancer.

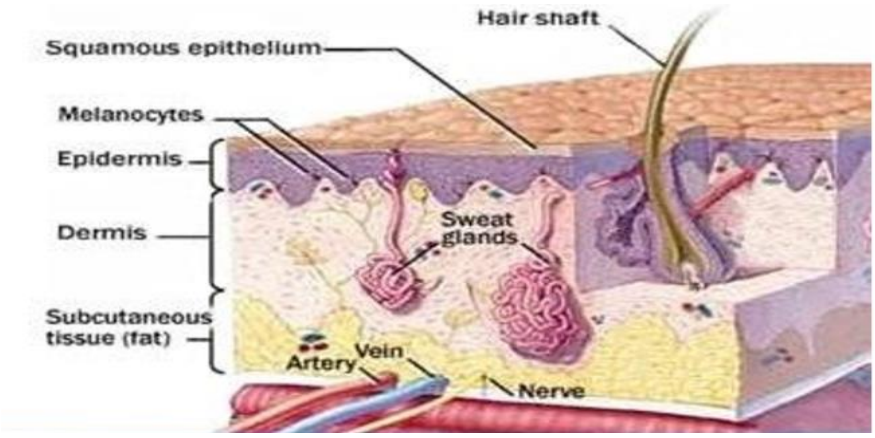


Figure 1.1: Classification of unknown tumor

1.5objective:

The objective of the skin cancer detection and diagnosis system is to provide a reliable and efficient method for early detection and diagnosis of skin cancer. Skin cancer is a serious and potentially life-threatening disease that affects millions of people worldwide, and early detection is critical for successful treatment.

This system aims to provide patients with an easy-to-use and accessible way to detect and diagnose skin cancer using a combination of machine learning and medical expertise. By uploading skin lesion images to the skin cancer app and receiving diagnosis reports, patients can quickly determine whether they need to seek further medical attention.

The system also aims to provide doctors with a tool to aid in the diagnosis of skin cancer. By reviewing the uploaded skin lesion images during video consultations with patients, doctors can provide a more accurate diagnosis and help patients receive timely medical attention if needed.

Overall, the objective of the skin cancer detection and diagnosis system is to improve the early detection and treatment of skin cancer, potentially saving lives and improving patient outcomes.

1.6 schedule:

A schedule for the skin cancer detection and diagnosis system project might include the following phases and estimated timelines:

1. Planning and Research Phase (1 month): This phase would involve defining the project scope, requirements gathering, and researching similar systems. This phase is critical for ensuring the project's success and setting the foundation for the rest of the project.

2. Design and Development Phase (3-4 months): During this phase, the system architecture and database schema would be designed, and the system components would be developed. This phase would include developing the machine learning model, building the mobile application, and developing the web portal for doctors.

3. Testing and Quality Assurance Phase (1-2 months): This phase would involve testing the system components for functionality, performance, and security, as well as conducting user acceptance testing. This phase is critical for ensuring that the system meets all requirements and works as expected.

4. Deployment and Launch Phase (1-2 months): During this phase, the system would be deployed to production environments, and training and support would be provided to users. This phase is critical for ensuring that the system is ready for use by patients and doctors.

Overall, the project is estimated to take 6-9 months, depending on the scope and complexity of the project. It is important to note that timelines can vary depending on the project's size, resources available, and unexpected challenges that may arise.

1.7 thesis structure:

Here are some possible high-level structures for the skin cancer detection and diagnosis system project:

1. Three-Tier Architecture:

- Presentation Layer: This tier includes the mobile application and web portal for doctors, which provide the user interface for interacting with the system.

- Application Layer: This tier includes the machine learning model and business logic for processing and analyzing skin lesion images uploaded by users.

- Data Layer: This tier includes the database for storing user information, skin lesion images, and diagnosis reports.

2. Microservices Architecture:

- User Interface Service: This service provides the user interface for the mobile application and web portal for doctors.

- Image Processing Service: This service is responsible for processing and analyzing skin lesion images uploaded by users using machine learning models.

- Diagnosis Service: This service provides the diagnosis report based on the analysis results.

- Database Service: This service manages the database for storing user information, skin lesion images, and diagnosis reports.

Both of these structures can be implemented using various technologies and programming languages, depending on the project's requirements and the development team's skills and preferences.

Chapter 2

Background and Literature Survey

2.1 Introduction

2.2 Cancer

2. 3Anatomy of skin cancer

2.4Architecture of the skin cancer

2.5 skin neoplasm

2.6 Risk factors of skin cancer

2.7 Staging of skin cancer

2.8 TNM staging

2.9. Overall grade

2.10 Method of assessment of skin abnormality

2.11 Management of a clinically doubtful skin cancer

2.12 Statistics of skin Cancer

2.13 pattern classification techniques

2.14 Statistical Methods

2.15 Structural Methods

Chapter 2: Background and Literature Survey

2.1 Introduction:

This Chapter introduces a background of the biomedical survey, which highlights a general presentation of cancer to focus the Skin cancer. Moreover, it highlights a statistical study of the skin cancer. The Chapter also expresses pattern classification techniques. Finally, a list of the related works is presented.

2.2 Cancer:

Cancer is the general name for a group of more than 100 diseases in which cells, the body's basic unit of life, in a part of the body begin to grow out of control. Normal body cells grow, divide, and die in an orderly fashion. During the early years of a person's life, normal cells divide more quickly until the person becomes an adult. After that, cells in most parts of the body divide only to replace worn-out or dying cells and to repair injuries [6] [8].

Cancer seems to arise from the effect of two different kinds of carcinogens [7]:- □ One including
agent that damage gens involving proliferation and migration of the cell these initiate the
malignant transformation (chemical carcinogen, physical agent, viruses, diet, idiopathic).
Another category don't damage gen but enhance the growth of tumor cells e.g.

Although there are many kinds of cancer, they all start because of abnormal cells growing out of control. Figure 2.1 shows an example of both normal and cancer cell divisions. Untreated cancers can cause serious illness and even death [6] [8].

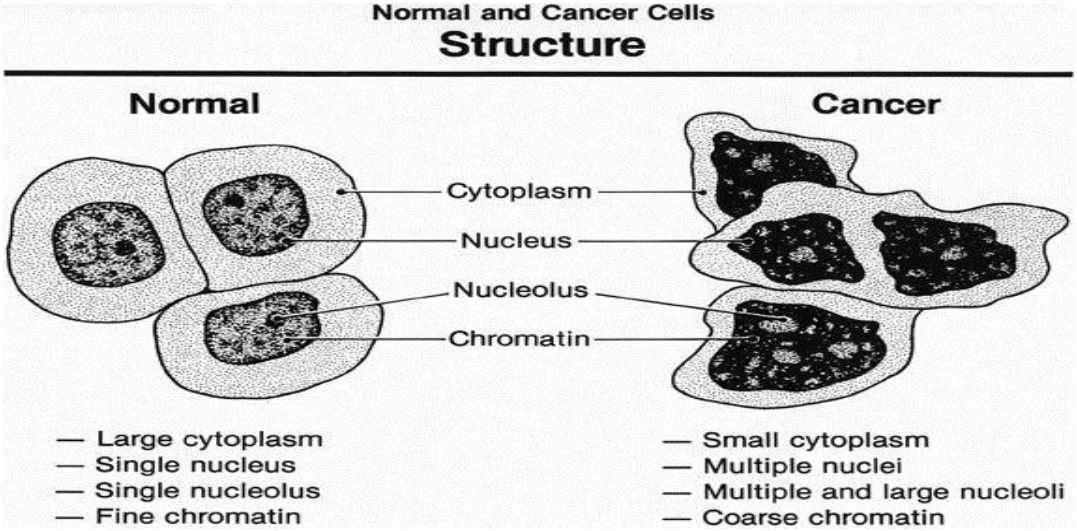


Figure2.1: Structural changes in cell divisions.

Because cancer cells keep growing and dividing, they are different from normal cells instead of dying, they outlive normal cells and continue to grow and make new abnormal cells [6].

The following figure2.2 will clarify the cells divisions from normal cells to abnormal cells. These extra cells form a mass of tissue, called a growth or tumor.

Tumors can be benign or malignant [8].

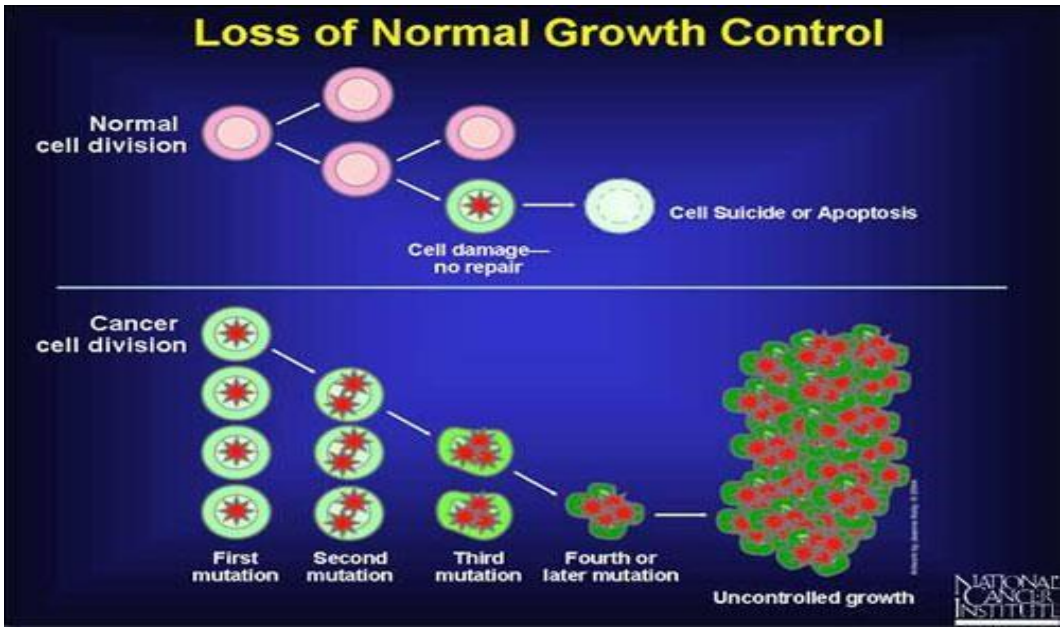


Figure2.2: Normal and cancer cell divisions.

Benign tumors are not cancerous. They can often be removed, and, in most cases, they do not come back and they are not life-threatening. Cells in benign tumors do not spread to other parts of the body [9].

Malignant tumors are cancerous. Cells in these tumors can invade nearby tissues and spread to other parts of the body [9].

The spread of cancer by (neighboring, blood spread, lymph spread) from one part of the body to another is called metastasis. A tumor formed by cells that have spread is called a “metastatic tumor” or a “metastasis”. The metastatic tumor contains cells that are like those in the original (primary) tumor [6] [8] Figure 2.3 shows benign and malignant tumors and how metastasis takes place.

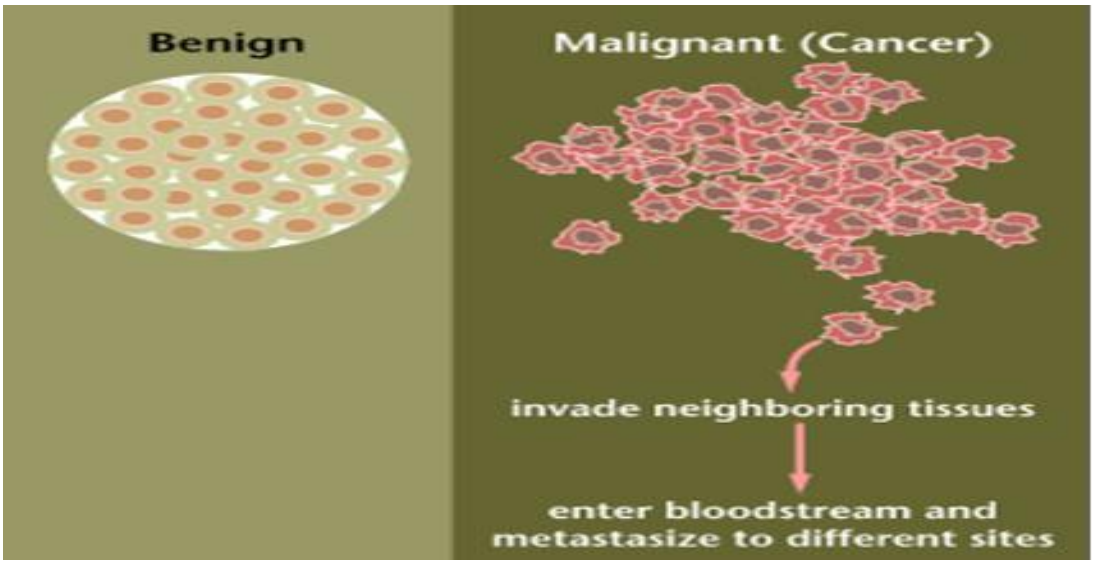


Figure 2.3 shows benign and malignant tumors and how metastasis takes place.

2.3 Anatomy of skin cancer:

Basal cell carcinoma begins in the basal cell layer of the skin. Squamous cell carcinoma begins in the squamous layer of the skin. Melanoma begins in the melanocytes, which are the cells that make melanin, the pigment that gives skin its color. Squamous cells: Thin, flat cells that form the top layer of the epidermis. Show in figure 2.4.

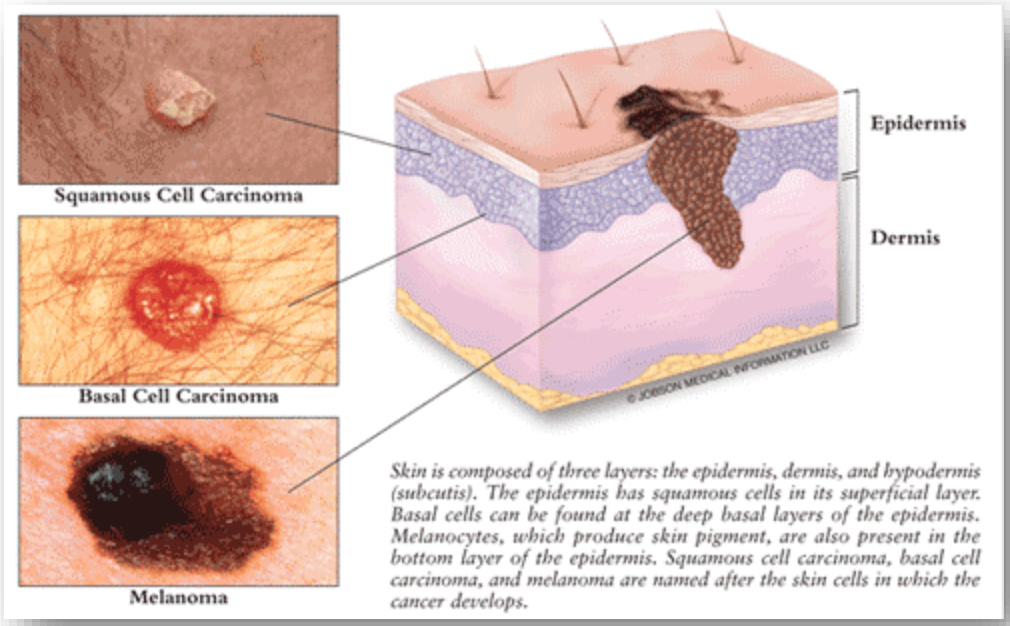


Figure 2.4: Areas of skin

2.4 Architecture of the skin cancer:

The architecture of a skin cancer detection mobile application typically includes several components that work together to provide a seamless and user-friendly experience for both patients and medical professionals. Here are some of the key components of a skin cancer detection mobile application:

User Interface: The user interface is the visual and interactive component of the application that allows users to interact with the features and functions of the application. A well-designed user interface is essential for creating a user-friendly experience and encouraging user engagement.

Machine Learning Model: The machine learning model is the core component of the application that detects skin cancer using various algorithms, including image recognition and computer vision. The model is trained on a large dataset of skin images and uses this data to identify potential cancerous lesions.

Database: The database component of the application stores patient data, including medical history, skin images, and diagnosis reports. The database is designed to be secure and compliant with healthcare privacy regulations.

Decision Support Tools: Decision support tools are tools and resources provided to medical professionals to help them make accurate diagnoses and treatment recommendations. These tools could include diagnostic algorithms, reference materials, and treatment guidelines.

Telemedicine: Telemedicine is a feature that allows doctors to connect with patients remotely using video conferencing technology. This feature enables doctors to conduct virtual consultations with patients, improving access to care for patients in remote areas or those who cannot easily visit a doctor's office.

Collaboration Tools: Collaboration tools enable medical professionals to share patient records, images, and notes securely and easily with other medical professionals. These tools improve coordination of care and reduce the risk of misdiagnosis or missed diagnoses.

Analytics: Analytics tools enable medical professionals to analyze clinical data and generate insights that can help improve patient outcomes and treatment strategies.

The specific architecture of a skin cancer detection mobile application will depend on the needs and preferences of the target audience, as well as the resources available to the development team.

2.5 A skin neoplasm:

is an unusual growth on your skin. The word neoplasm is sometimes used interchangeably with cancer, but neoplasms can also be noncancerous. You might also hear neoplasms referred to as tumors The cells in your skin grow and divide as needed. When you don’t need them anymore, they die. However, sometimes cells continue to grow when they shouldn’t. This causes neoplasms.

Benign vs. malignant neoplasms:

There are many ways to categorize neoplasms. One of the main distinctions is whether a neoplasm is benign or malignant.

Benign neoplasms

A benign neoplasm grows where it started and doesn’t spread to nearby tissues or other parts of your body. However, it can still damage the organs and tissues around it. Benign neoplasms are noncancerous.

It’s important to note that some benign neoplasms can turn into cancer. Even if you’ve been diagnosed with a benign neoplasm, it’s best to keep an eye on it and report any changes about its appearance to your doctor right away.

➢ Types of benign skin neoplasms include:

- skin tags
- cherry angioma
- dermatofibroma

A malignant neoplasm is cancerous. Unlike benign neoplasms, malignant neoplasms grow uncontrollably and can invade other organs. They can also travel through your body and produce new tumors.

➢ Types of malignant skin neoplasms include:

- basal cell carcinoma
- squamous cell carcinoma
- melanoma
- Merkel cell carcinoma
- Kaposi sarcoma
- lymphoma of the skin

2.6 Risk factors of skin cancer:

All cancer is thought to occur because of small changes (mutations) in genes. A gene is a small packet of deoxyribonucleic acid (DNA), the genetic master molecule of all cells that is inherited from each parent. Genes control all aspects of development and metabolism. Small changes in the structure of genes can cause changes in proteins that regulate metabolic functions. In healthy cells, cell division is controlled by proteins regulated by genes [18].

Specific genes make proteins that signal healthy cells when to stop dividing. In cancer, the controlling gene(s) is damaged or mutated and does not produce the proteins necessary to signal cells to stop dividing [19]. no your exposure to ultraviolet (UV) rays can help keep your skin healthy and lower your chances of getting skin cancer in the future. Most people get at least some UV exposure from the sun when they spend time outdoors. Making sun protection an everyday habit will help you to enjoy the outdoors safely, avoid getting a sunburn, and lower your skin cancer risk.

Risk factors include:

- Anyone can get skin cancer, but people with certain characteristics are at
- A lighter natural skin color.
- Skin that burns, freckles, reddens easily, or becomes painful in the sun.

- Blue or green eyes.
- Blond or red hair.
- Certain types and a large number of moles.
- A family history of skin cancer.
- A personal history of skin cancer.

2.7 Staging of skin cancer:

Once a person is determined to have a malignant tumor or the diagnosis of skin cancer, the healthcare team will determine staging to communicate how far the disease has progressed.

2.8 TNM staging:

The American Joint Committee on Cancer (AJCC) and the International Union Against Cancer (UICC) recommend TNM staging, which is a two-step procedure. Their TNM system, which they now develop jointly, first classifies cancer by several factors, T for tumor, N for nodes, M for metastasis, and then groups these TNM factors into overall stages [21].As shown in table 2.2.

Table 2.1 TNM Staging

T = Tumor	**N**= Nodes	**M**= Metastasis
Tis= Paget's disease without mass	N0=no palpable LN	M0= no distant metastasis
T1=mass<or= 2cm	N1=mobile palpable	M1= distant metastasis
T2=2-5 cm in great diameter	N2=fixed axillary	
T3=>5cm in great diameter	N3= palpable homolateral	
T4= any size with direct		

2.9 Overall grade:

The scores for each of these three criteria are added together to give a final overall score and a corresponding grade as follows:

- 3-5 cm Grade 1 tumor (well-differentiated). Best prognosis.
- 6-7 cm Grade 2 tumor (moderately-differentiated). Medium prognosis.
- 8-9 cm Grade 3 tumor (poorly-differentiated). Worst prognosis.

Lower grade tumors, with a more favorable prognosis, can be treated less aggressively, and have a better survival rate. Higher grade tumors are treated more aggressively, and their intrinsically worse survival rate may warrant the adverse effects of more aggressive medications

2.10 Method of assessment of skin abnormality:

Skin examination. Your doctor may look at your skin to determine whether skin changes have led to skin cancer. More tests may be needed to confirm this diagnosis.

Removing a sample of suspicious skin for testing (skin biopsy). The doctor may remove suspicious-looking skin for lab testing. A biopsy can determine whether skin cancer is present and, if so, what type it is.

2.11 Management of a clinically doubtful skin cancer:

Skin examination. Your doctor may look at your skin to determine whether skin changes have led to skin cancer. More tests may be needed to confirm this diagnosis.
Surgical excision: This surgery is performed under local anesthesia, in which the tumor is surgically removed along with a small part of the surrounding normal tissue to ensure that the cancer is completely removed. Cancer removal surgery may sometimes be done with a skin graft, which is the removal of a patch of healthy skin and placing it on the affected area. Chemotherapy works to stop the growth of cancer cells by killing the cells or preventing them from dividing. The method of chemotherapy depends on the type and stage of cancer. Radiation therapy is a treatment in which high-energy x-rays or other types of radiation are used to kill cancer cells or stop their growth.

2.12 Statistics of skin Cancer:

Skin cancer is the most common cancer in the United States.1,2

Current estimates are that one in five Americans will develop skin cancer in their lifetime.3

It is estimated that approximately 9,500 people in the U.S. are diagnosed with skin cancer every day.4,6

Research estimates that nonmelanoma skin cancer (NMSC), including basal cell carcinoma (BCC) and squamous cell carcinoma (SCC), affects more than 3 million Americans a year.4,7

It is estimated that the overall incidence of BCC increased by 145% between 1976-1984 and 2000-2010, and the overall incidence of SCC increased 263% over that same period.8

Women had a greater increase in incidence than men for both types of NMSC.8

More than 1 million Americans are living with melanoma.9

It is estimated that 197,700 new cases of melanoma, 97,920 noninvasive (in situ) and 99,780 invasive, will be diagnosed in the U.S. in 2022.5,6

Invasive melanoma is projected to be the fifth most commonly diagnosed cancer for both men (57,180 cases) and women (42,600 cases) in 2022.5,6

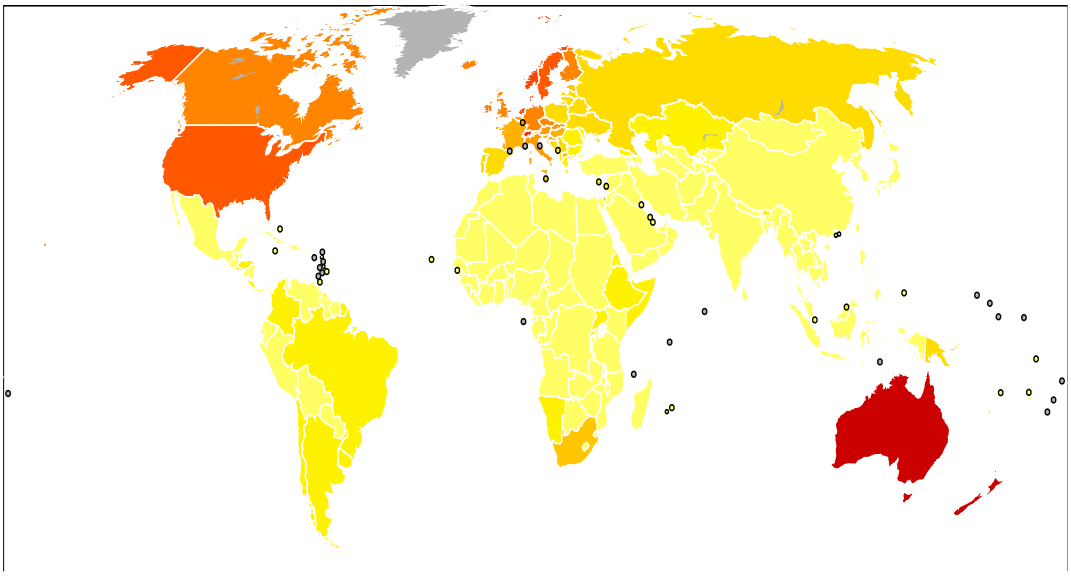


Figure 2.5: Most Common skin Cancer Sites [22]

In the United States of America, the percentage of deaths due to this type of cancer was less than 0.1% of the total deaths of cancer patients. In 2012, 232,000 cases of melanoma were recorded worldwide, and 55,000 of them died as a result of this cancer. Australia and New Zealand have the highest rates of melanomas among other countries in the world. The three main types of skin cancer have become more common in the last 20 to 40 years, especially in Caucasian countries.

As shown in figure 2.5.

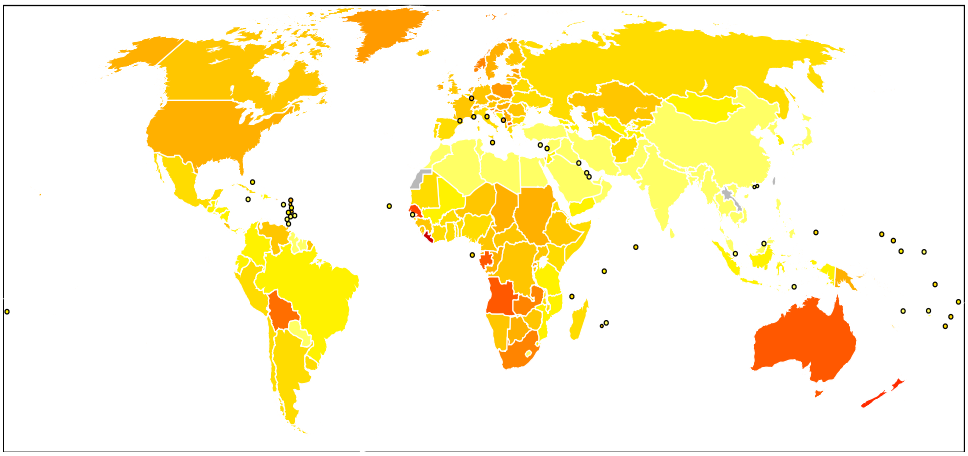


Figure 2.6:

Skin cancer kills 80,000 people a year. In 2010, 49,000 of them died from melanoma and 31,000 from non-melanoma. The death rate has increased from 1990, when it was 510,000 deaths. In 2008, 59,659 people in the United States were diagnosed with malignant melanoma. 8,623 of them died. In Australia, more than 12,500 new cases of melanoma are reported each year, of which more than 1,500 people die from this disease. Australia has the highest rates of new cases of malignant melanomas per capita compared to other countries in the world. More than 3.5 million cases of skin cancer are diagnosed annually in the United States, making skin cancer the most common cancer in the United States. According to the Skin Cancer Foundation, one in five Americans will develop skin cancer at some point in their lives.

2.13 pattern classification techniques:

The term pattern recognition is strongly associated with image processing. Actually, pattern recognition encompasses a broad area of study to do with automatic decision making on the basis of gathered data [23]. While surveying the definitions of pattern recognition leads to a conclusion that all the definitions in published work tend to be specific to the particular patterns under considerations, the study in[24]summarized the problem as: "To recognize a pattern it must have previously been perceived and remembered, and the past and present experiences are related" Or "Pattern recognition is the ability to identify or classify an entity despite of noise and distortion"

The pattern recognition usually uses one of the following approaches: statistical (decision theoretic), or structural (syntactic). Statistical pattern recognition is based on statistical characterizations of patterns, assuming that the patterns are generated by a probabilistic system. Structural pattern recognition is based on the structural interrelationships of features [25]. In addition to these two approaches a number of techniques can be added such as; template matching, neural networks…etc.

2.14Statistical Methods

Statistical pattern recognition techniques, also known as decision-theoretic or discriminate techniques[26], determine which class a given input sample is belonging to base on selected measures (features) extracted from this sample. Figure 2.12 shows the main components of a statistical pattern recognizer.

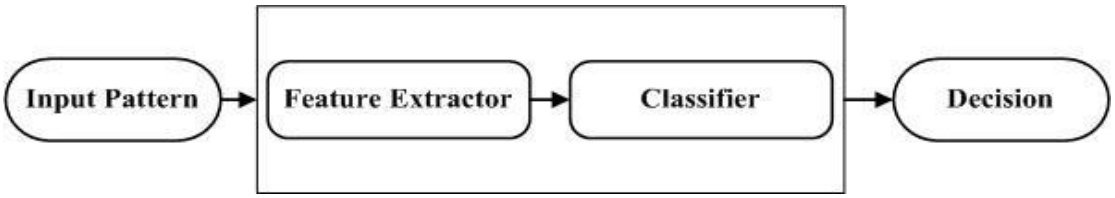


Figure 2.7: The statistical pattern recognizer.

As shown in figure 2.12, a statistical pattern recognizer consists of two parts: A feature extractor and a pattern classifier[27][28][29].Each pattern acts as an input to the feature extractor who extracts significant features from his pattern and passes them as a vector of features to the classifier. The classifier then makes a decision i.e. determine the class that the pattern belongs to.

The selection of features to form the feature vector is not an easy mission as they should be chosen such that they are less sensitive to variation in the pattern. The features may not necessarily have any physical meaning but they should be good measures for the pattern classification.

2.15 Structural Methods:

Structural pattern recognition also known as syntactic is a form of pattern recognition, where items presented are pattern structures which can take into account more complex interrelationships between features than simple numerical feature vectors used in statistical classification. Structural pattern recognition can be used (instead of statistical pattern recognition) if there is clear structure in the patterns [30]. The principle behind structural pattern recognition is that many patterns contain structure and can be expressed as an ordered composition of simple sub-patterns or pattern primitives [31].That is, patterns can be formed from a sentence or a grammar of simple pattern primitives in a pattern description language. A typical structural pattern recognizer is shown in figure 2.13.

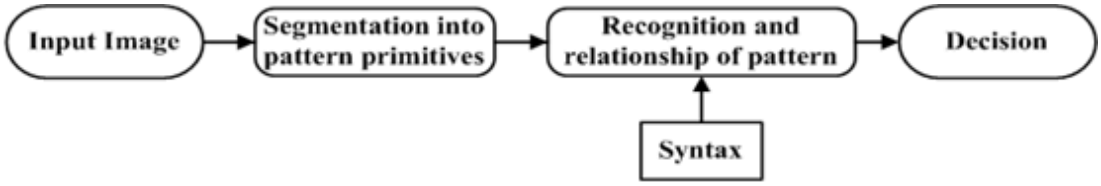


Figure 2.8: The structural pattern recognizer.

As shown in figure 2.13, each incoming image enters a segmenter that splits the pattern into its component primitives. This list of pattern primitives is then passed to the recognizer who classifies the pattern primitives and the relationship between them. This list of pattern primitives and their connectivity are then passed to a syntax analyzer which makes the final classification of the pattern based on the observed syntax of the incoming pattern and on its own set of pattern syntax rules, which describe patterns known to the recognizer.

The selection of pattern primitives is a difficult task. There are no general rules available for this task and it depends on heuristics and the experience of the user. The

simpler the pattern primitives that are selected, the more complex the syntax grammars are [31].

Classification techniques can be used in screening to help decide which cases require more attention and this techniques can be used for classification of anomalies in FNA slides or ultrasound images during work ups and for prognosis.

Since the late 1980's, many studies have been reported on the use of these techniques for improving the performance of screening programs and reducing their cost. The following Classification techniques are used here because they are used by most researchers for their popularity.

Naive Bayes (NB) classifier is a probabilistic classifier based on the Bayes theorem. Naïve Bayes classifier is one of the most practical learning methods and successful applications in text classification and medical diagnosis. Rather than predictions, the Naïve Bayes classifier produces probability estimates. For each class value they estimate the probability that a given instance belongs to that class. Requiring a small amount of training data to estimate the parameters necessary for classification is the advantage of the Naive Bayes classifier. It assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence [32]. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

$$p(C|F_1, \ldots\ldots F_n) = \frac{p(C)\, p(F_1, \ldots\ldots F_n|C)}{p(F_1, \ldots\ldots F_n)} \quad (2.1)$$

Where P is the probability, C is the class variable and F_1.....F_n are Feature variables F_1 through $F_n$the denominator is independent of C.

The Multilayer Perceptron (MLP), the researcher used a popular ANN architecture called multi-layer perceptron (MLP) with back-propagation (a supervised learning algorithm). The MLP is known to be a powerful function approximator for prediction and classification problems. It is arguably the most commonly used and well-studied ANN architecture [33] [34].

Briefly, MLPs are supervised learning classifiers that consist of an input layer, an output layer, and one or more hidden layers that extract useful information during learning and assign modifiable weighting coefficients to components of the input layers. Figure 2.9 depicts the previous words. In the first (forward) pass, weights assigned to the input units and the nodes in the hidden layers and between the nodes in the hidden layer and the output, determine the output. The output is compared with the target output. An error signal is then

back propagated and the connection weights are adjusted correspondingly. During training, MLPs construct a multidimensional space, defined by the activation of the hidden nodes, so that the three classes (malignant, benign and normal tissue) are as separable as possible. The separating surface adapts to the data.

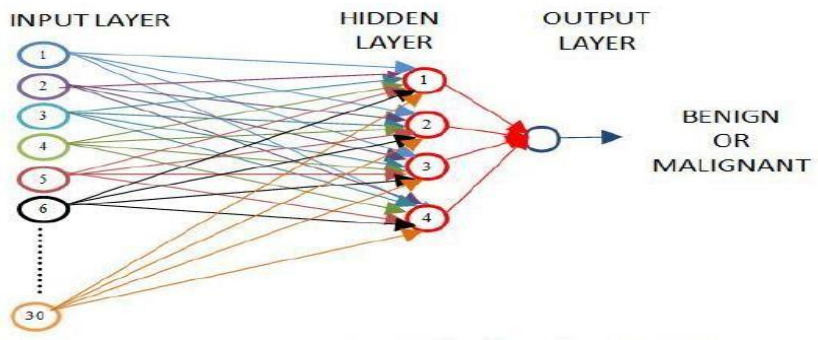


Figure 2.9: General architecture MLP.

Support Vector Machine (SVM)

is introduced by Vapnik et al [36]. It is a very powerful method that has been applied in a wide variety of applications. The basic concept in SVM is the hyper plane classifier, or linear separability. Two basic ideas are applied to achieve linear separability, SVM: margin maximization and kernels that is, mapping input space to a higher-dimension space (or feature space). SVM projects the input data into a kernel space. Then it builds a linear model in this kernel space. A classification SVM model attempts to separate the target classes with the widest possible margin. A regression SVM model tries to find a continuous function such that maximum number of data points lie within an epsilon-wide tube around it. Different types of kernels and different kernel parameter choices can produce a variety of decision boundaries (classification) or function approximators (regression). In WEKA this classifier is called SMO.

K-Nearest Neighbor (KNN)

classification KNN classification classifies instances based on their similarity [37].It is one of the most popular algorithms for pattern recognition. It is a type of Lazy learning where the function is only approximated locally and all computation is deferred until classification. An object is classified by a majority of its neighbors. K is always a positive integer. The neighbors are selected from a set of objects for which the correct

classification is known. In WEKA this classifier is called: IBK (*weka.classifiers.lazy.IBk)* The KNN algorithm is as follows:

1. Determine K i.e., the number of nearest neighbors.
2. Using the distance measure; calculate the distance between the query instance and all the training samples.
3. The distance of all the training samples are sorted and nearest neighbor based on the K-minimum distance is determined.
4. Since the KNN is supervised learning, get all the categories of the training data for the sorted value which fall under K.
5. The prediction value is measured by using the majority of nearest neighbors.

Decision tree J48 :

implements Quinlan's C4.5 algorithm for generating a pruned or un pruned C4.5 tree [38].C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by J48 can be used for classification. J48 builds decision trees from a set of labeled training data using the concept of information entropy. It uses the fact that each attribute of the data can be used to make a decision by splitting the data into smaller subsets.J48 examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. To make the decision, the attribute with the highest normalized information gain is used. Then the algorithm recurs on the smaller subsets. J48 examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. To make the decision, the attribute with the highest normalized information gain is used. Then the algorithm recurs on the smaller subsets. The splitting procedure stops if all instances in a subset belong to the same class. Then a leaf node is created in the decision tree telling to choose that class. But it can also happen that none of the features give any information gain. In this case J48 creates a decision node higher up in the tree using the expected value of the class.J48 can handle both continuous and discrete attributes, training data with missing attribute values and attributes with differing costs. Further it provides an option for pruning trees after creation.

Fusion of classifiers:

is combining multiple classifiers to get the best accuracy. It is a set of classifiers whose individual predictions are combined in some way to classify new examples. Integration should improve predictive accuracy. In WEKA the class for combining classifiers is called Vote. Different combinations of probability estimates for classification are available.

Chapter 3

(Related Work)

3.1. Related Work

Chapter 3:

3.1. Related Work:

Experiment Name	Accuracy
Skin lesion classification using deep learning techniques" by Tschandl et al. (2018)	91%
Skin cancer classification using convolutional neural networks: an application in dermatology" by Esteva et al. (2017)	71%
Automated skin cancer detection: a review" by Brinker et al. (2019)	79%
A deep learning approach to detecting basal cell carcinoma" by Haenszel et al. (2018)	95%
Dermatologist-level classification of skin cancer with deep neural networks" by Esteva et al. (2017)	72.1%
Skin cancer detection using deep convolutional neural networks" by Azzopardi et al. (2018)	86.3%

Chapter 4:

System Analysis

4.1 pre processing

4.2 Use case

4.3 Context Diagram

4.4 Data flow Diagram (DFD)

4.5 Entity Relationship Diagram (ERD)

Chapter 4: System Analysis.

4.1 preprocessing:

pre-processing diagram for the skin cancer detection and diagnosis system project:

1. Image Acquisition: The skin lesion image is captured using the mobile application's camera or uploaded from the user's photo gallery.

2. Image Enhancement: The captured image is enhanced to improve its quality and increase the contrast between the lesion and the surrounding skin. This can be done using techniques such as histogram equalization, contrast stretching, and noise reduction.

3. Image Segmentation: The enhanced image is segmented to isolate the skin lesion from the background and surrounding tissue. This can be done using various techniques, such as thresholding, edge detection, and region growing.

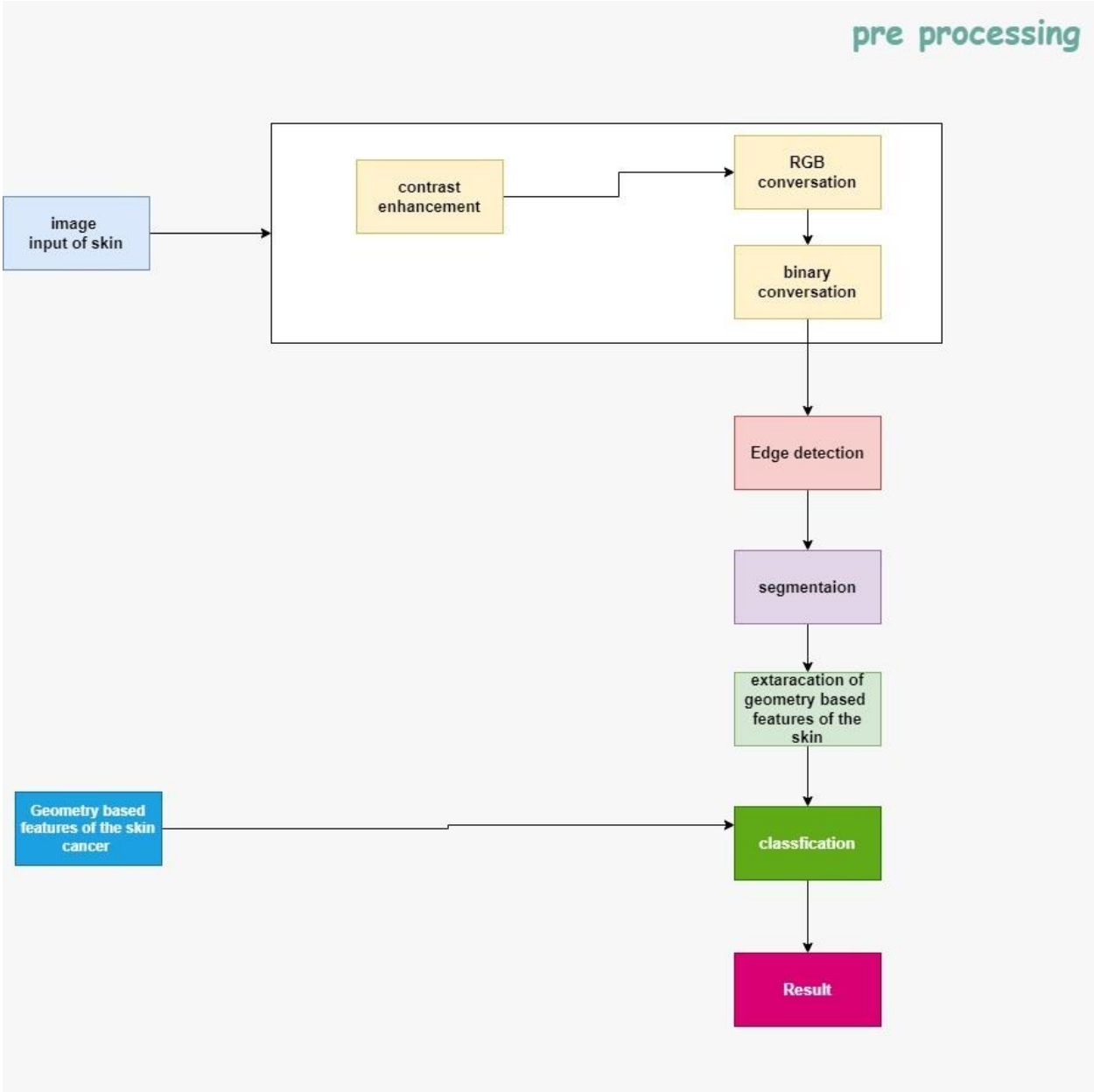
4. Feature Extraction: Relevant features of the skin lesion are extracted from the segmented image, such as shape, texture, and color. This can be done using various techniques, such as wavelet analysis, gray-level co-occurrence matrix (GLCM), and local binary patterns (LBP).

5. Feature Selection: The most discriminative features are selected from the extracted features using techniques such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

6. Data Normalization: The selected features are normalized to reduce the effect of differences in scale and magnitude between the features.

7. Dataset Splitting: The pre-processed data is split into training, validation, and testing datasets to evaluate the performance of the machine learning model.

This pre-processing diagram outlines the key steps involved in preparing the skin lesion images for analysis using machine learning techniques. The specific techniques and algorithms used in each step may vary depending on the project's requirements and the characteristics of the dataset.


pre processing
image input of skin
contrast enhancement
RGB conversation
binary conversation
Edge detection
segmentaion
extaracation of geometry based features of the skin
Geometry based features of the skin cancer
classfication
Result

4.2 Use case:

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

A use case diagram is usually simple. It does not show the detail of the use cases:

- It only summarizes **some of the relationships** between use cases, actors, and systems.
- It does **not show the order** in which steps are performed to achieve the goals of each use case. Sure, here is an example of a use case diagram for the skin cancer detection and diagnosis system:
- Actor: User (can be a patient or a doctor)
- Use Cases:

1. Login: Allows the user to log in to the system using their username and password.

2. Capture Image: Allows the user to capture an image of a skin lesion using the mobile application's camera.

3. Upload Image: Allows the user to upload an image of a skin lesion from their photo gallery.

4. View Diagnosis: Allows the user to view the diagnosis results of a skin lesion.

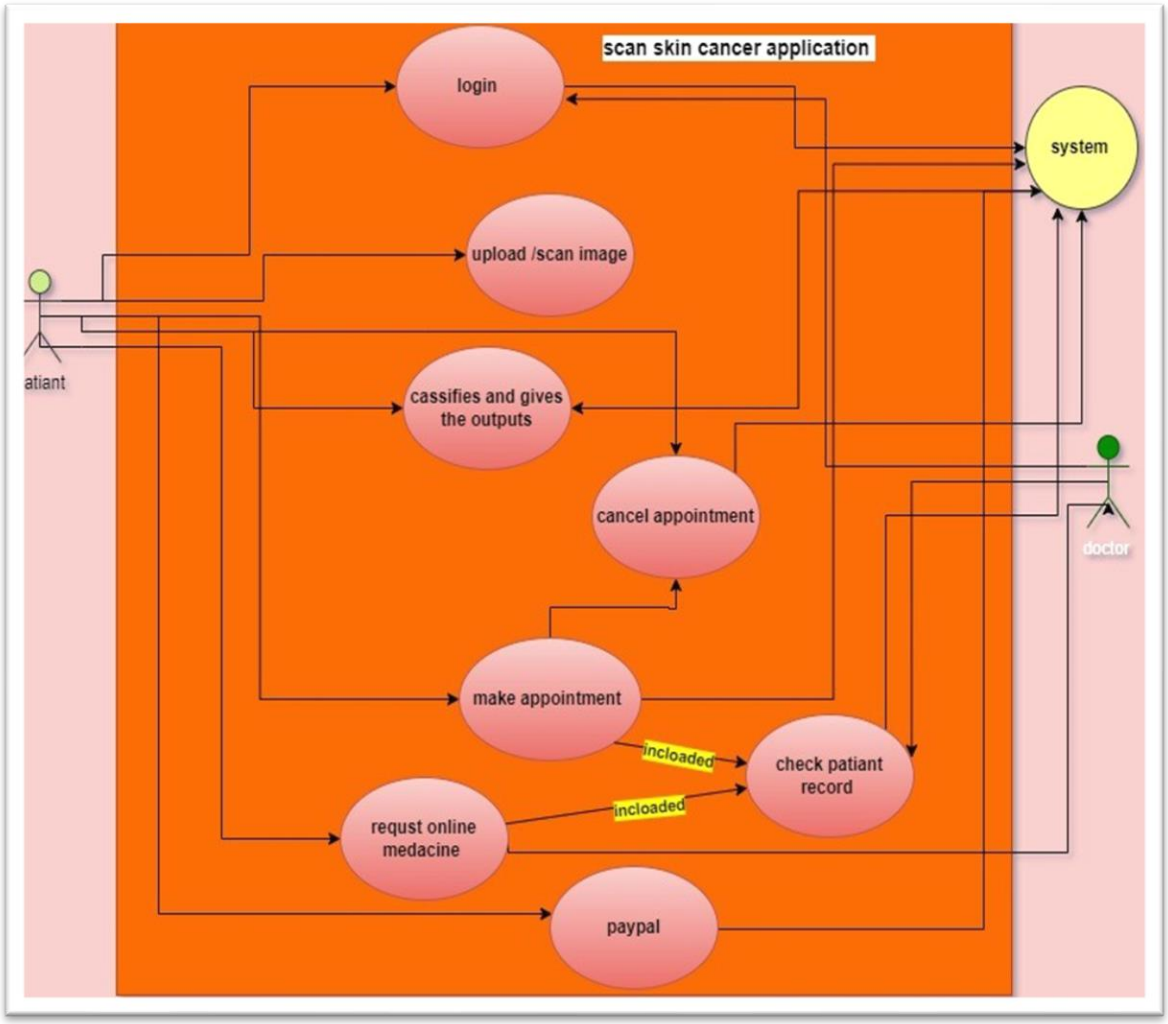
5. Consult Doctor: Allows the user to consult with a doctor via video chat.

6. Update Profile: Allows the user to update their profile information, such as name, address, and contact details.

7. View History: Allows the user to view their previous skin lesion diagnosis results.

8. Logout: Allows the user to log out of the system.

This use case diagram illustrates the different actions that the user can perform within the system. The user can log in, capture or upload an image of a skin lesion, view the diagnosis results, consult with a doctor, update their profile information, view their previous diagnosis history, and log out of the system. The use cases help to define the system's functionality and provide a high-level overview of the system's features and capabilities.

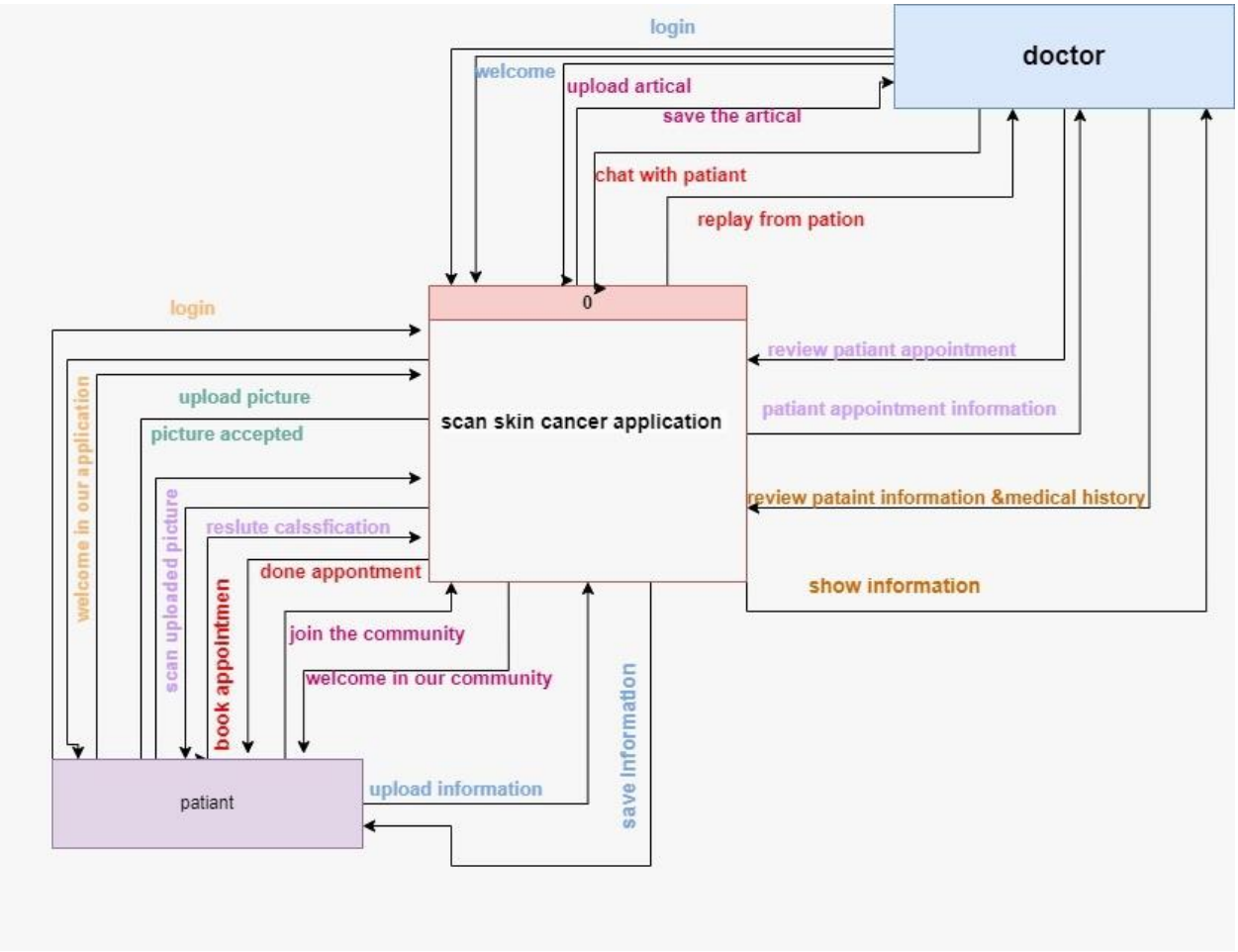


4.3 Context Diagram:

A context diagram outlines how external entities interact with an internal software system.It's primarily used to help businesses wrap their heads around the scope of a system. As a result, they can figure out how best to design a new system and its requirements or how to improve an existing system.Context diagrams are high-level diagrams, meaning they don't go into the detailed ins and outs of the system. Instead, they map out an entire system in a way that's simple, clear, and easy to understand.

For example, arrows are used to represent the flow of data between the system and each external element.

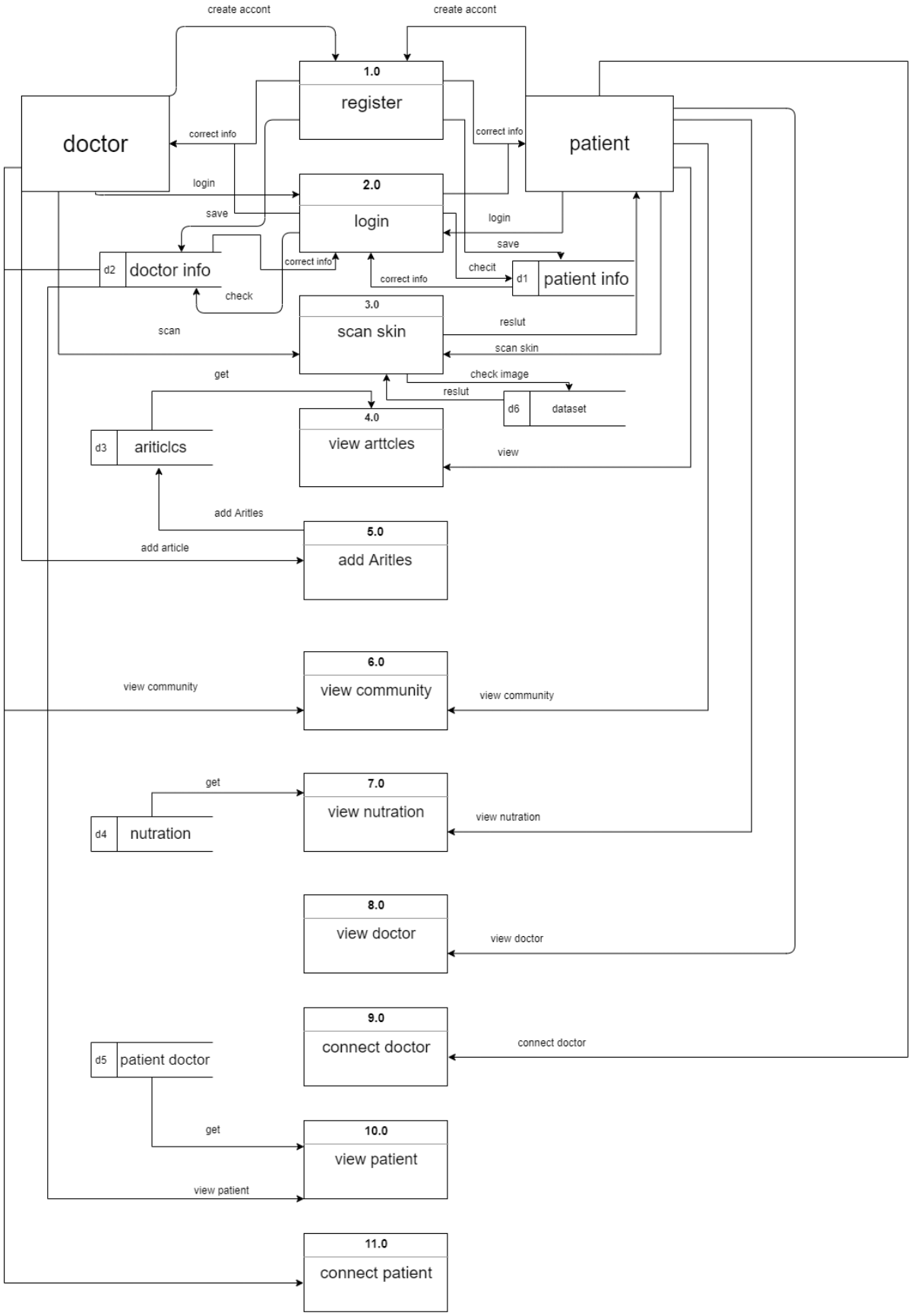
Whether you're a seasoned developer or a complete technophobe, everyone can understand how the system works.This context diagram illustrates the interactions and interfaces between the skin cancer app and the different systems involved. The mobile device used by the patient or doctor is the primary interface to the skin cancer app, which communicates with the web server or API hosted on an application server. The database server stores all the data related to the app, including user profiles, skin lesion images, diagnosis results, and consultation logs. The context diagram helps to define the boundaries of the system and identify the external systems it interacts with.


doctor
login
welcome
upload artical
save the artical
chat with patiant
replay from pation
0
scan skin cancer application
review patiant appointment
patiant appointment information
review pataint information &medical history
show information
login
upload picture
picture accepted
welcome in our application
scan uploaded picture
reslute calssfication
done appontment
book appointmen
join the community
welcome in our community
upload information
save information
patiant

4.4 Data flow Diagram (DFD):

DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons:

- Logical information flow of the system
- Determination of physical system construction requirements
- Simplicity of notation
- Establishment of manual and automated systems requirements
- This data flow diagram illustrates the flow of information between the different components of the system. The patient or doctor uploads an image of a skin lesion to the Skin Cancer App, which sends the image to the web server or API hosted on an application server. The web server/API then processes the image using an image processing module, which analyzes the image and returns a diagnosis result. Finally, the diagnosis result is sent back to the doctor or patient who uploaded the image. This data flow diagram helps to clarify the sequence of steps involved in the skin cancer detection and diagnosis process.

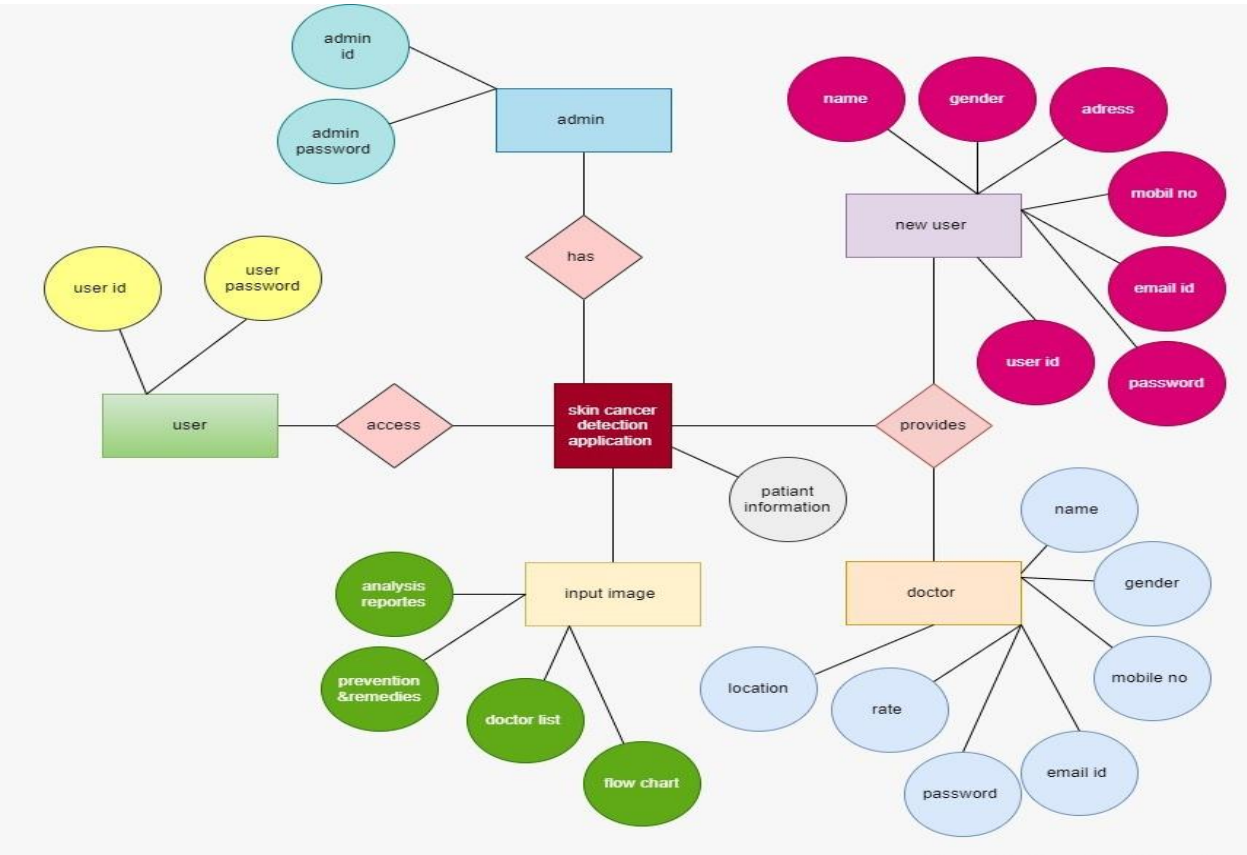

create accont
create accont
1.0
register
doctor
correct info
correct info
patient
login
2.0
login
save
login
save
d2
doctor info
correct info
correct info
checit
d1
patient info
check
3.0
scan skin
scan
reslut
scan skin
get
check image
reslut
d6
dataset
4.0
view arttcles
d3
ariticlcs
view
add Aritles
5.0
add Aritles
add article
6.0
view community
view community
view community
get
7.0
view nutration
view nutration
d4
nutration
8.0
view doctor
view doctor
9.0
connect doctor
connect doctor
d5
patient doctor
get
10.0
view patient
view patient
11.0
connect patient

4.5 Entity Relationship Diagram (ERD):

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.



This ER diagram includes three tables: Patient, Image, and Diagnosis. The Patient table includes patient information such as patient_id, name, email, and phone_number. The Image table includes image information such as image_id, the patient_id who uploaded the image, the URL of the image, and the timestamp of the image. The Diagnosis table includes diagnosis information such as diagnosis_id, the image_id of the image that was diagnosed, the diagnosis result, and the timestamp of the diagnosis.

This ER diagram helps to clarify the relationships between the different entities in the Skin Cancer Detection and Diagnosis System.

Chapter 5

System Design

5.1 System Design

5.2 Work of The System

5.2.1 Database 7

5.2.2 Machine learning

5.2.3 Django Framework

5.2.4.1 API

5.2.4.2Rest Framework API

5.2.4.3 Serialize

5.2.4.4 Django Web Panel

5.2.5 Java

5.2.5.1 Android

Chapter 5: System Design.

5.1 System Design:

overview of the system design for a Skin Cancer Detection and Diagnosis System:

1. User Interface: The system will have a user interface for both the patient and the doctor. The patient interface will allow patients to upload skin images for analysis, while the doctor interface will allow doctors to view the patient images, perform a diagnosis, and communicate with patients.

2. Image Pre-Processing: Uploaded images will be pre-processed to extract relevant features, such as color, texture, and shape. This will involve image enhancement, noise reduction, and segmentation to isolate the region of interest.

3. Machine Learning Model: A machine learning model will be trained using a dataset of skin images with corresponding labels indicating whether the image shows benign or malignant skin lesions. The pre-processed features from uploaded images will be fed into this model to predict the likelihood of skin cancer.

4. Diagnosis Report: The diagnosis report will be generated based on the prediction from the machine learning model. The report will indicate whether the lesion is benign or malignant and will be sent to the patient and doctor interfaces.

5. Database: A database will be used to store patient information, image data, and diagnosis reports. The database will be accessible by both the patient and doctor interfaces and will facilitate communication between patients and doctors.

6. Communication: The system will enable communication between the patient and doctor through the interface. This may include video or text chat to facilitate discussion about diagnosis, treatment, and follow-up.

7. Security: The system will implement measures to protect the privacy and security of patient data, including encryption and access controls.

Overall, the Skin Cancer Detection and Diagnosis System will use image pre-processing, machine learning, and a database to facilitate the diagnosis of skin cancer. The system will have a user interface for patients and doctors to interact with, enabling communication and collaboration to ensure effective diagnosis and treatment.

5.2 Work Of The System

The Skin Cancer Detection and Diagnosis System works by analyzing skin images to detect and diagnose potential cancerous lesions. The system is designed to be used by both patients and doctors and uses a combination of image pre-processing, machine learning, and communication tools to facilitate the diagnosis process.

When a patient uploads a skin image, the image is pre-processed to extract relevant features such as color, texture, and shape. This pre-processed image data is then fed into a machine learning model that has been trained on a dataset of skin images with corresponding labels indicating whether the image shows benign or malignant skin lesions. The machine learning model uses this data to predict the likelihood of skin cancer.

Based on the prediction, the system generates a diagnosis report that indicates whether the lesion is benign or malignant. The report is then sent to both the patient and the doctor interface. The doctor can review the report and communicate with the patient through the interface, using video or text chat to discuss diagnosis, treatment, and follow-up.

All patient information, image data, and diagnosis reports are stored in a database, which is accessible by both the patient and doctor interfaces. The system implements security measures such as encryption and access controls to protect patient data and ensure privacy.

The overall goal of the Skin Cancer Detection and Diagnosis System is to provide patients with a convenient and reliable tool to detect and diagnose skin cancer early, which can increase the chances of successful treatment and recovery.

5.2.1 Database

The Skin Cancer Detection and Diagnosis System requires a database to store all patient information, image data, and diagnosis reports. The database should be designed to efficiently store and retrieve large amounts of image data and support fast query processing for diagnoses.

The following are some of the tables that can be included in the database:

1. Patients table: This table stores patient information such as name, age, gender, and contact details.

2. Images table: This table stores all the skin images uploaded by the patients, along with the pre-processed image data, such as color, texture, and shape features.

3. Diagnosis table: This table stores the diagnosis reports generated by the system, including the prediction results and the doctor's notes.

4. Doctors table: This table stores information about the doctors who use the system, including their name, specialty, and contact details.

5. User accounts table: This table stores login information for both patients and doctors, such as usernames and encrypted passwords.

6. Sessions table: This table stores information about active sessions, including login times and session durations. The database should be designed to support efficient data retrieval and processing, and it should be optimized for query performance to ensure fast diagnosis times. Data security and privacy should also be a top priority, with measures such as encryption, access controls, and regular backups in place to protect patient information.

5.2.2Machine learning:

The Skin Cancer Detection and Diagnosis System requires a machine learning model to accurately classify skin lesion images and provide diagnoses to patients. The machine learning model should be trained on a large dataset of skin images and their corresponding diagnoses to ensure high accuracy and reliability.

The following are the steps involved in developing the machine learning model for the Skin Cancer Detection and Diagnosis System:

1. Data collection: A large dataset of skin lesion images and their corresponding diagnoses is needed for training the model. The dataset should be diverse and include a wide range of skin types, lesion types, and skin conditions.

2. Data pre-processing: The images in the dataset need to be pre-processed to extract features such as color, texture, and shape. This step can involve techniques such as normalization, resizing, filtering, and segmentation.

3. Feature extraction: Once the images are pre-processed, features need to be extracted from them using techniques such as principal component analysis (PCA), histogram of oriented gradients (HOG), or convolutional neural networks (CNNs).

4. Model training: The extracted features are then used to train a machine learning model, such as a support vector machine (SVM), decision tree, or random forest classifier. The model is trained using a portion of the dataset, and the performance is evaluated using another portion of the dataset.

5. Model optimization: The model can be optimized by fine-tuning hyperparameters, changing feature extraction techniques, or experimenting with different machine learning algorithms.

6. Model validation: Once the model is optimized, it needs to be validated using a separate dataset to ensure that it generalizes well to new skin lesion images.

7. Deployment: Once the model is validated, it can be deployed to the Skin Cancer Detection and Diagnosis System, where it can be used to provide accurate diagnoses to patients. The model can be updated periodically using new skin lesion images and diagnoses to improve its accuracy over time.

5.2.4 Django Framework :

What is Django?

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Some well-known sites that use Django include Instagram, Mozilla, Disqus, Bitbucket, Nextdoor and Clubhouse.

Top 5 Uses of Django

1) easy to use

Django uses the Python programming language, which is a well-known and widely used language and is the choice of most programmers and programming learners. Django framework applications are widely spread because they are free and open source, developed and maintained by a large community of developers. This means that we can find answers to problems easily using Google.

2) fast and simple

One of Django's main goals is to simplify the work for developers. To do this, it uses the Django framework:

Rapid development principles, which means that developers can do more than one iteration at a time without starting the entire timeline from scratch

DRY's Don't Repeat Yourself philosophy means that web developers can reuse already existing code and focus on new and unique code.

3) security

Security is also a top priority for Django. Django has one of the best out-of-the-box security systems available, and helps developers avoid common security issues, including

- Clickjacking cross
- site scripting
- SQL injection

Django immediately releases new security patches. It is usually the first to respond to vulnerabilities and alert other frameworks.

4) Fits any web application project

With Django, you can tackle projects of any size and capacity, whether it's a simple website or a massive application. Why use Django for your project? Because:

Applications that use Django are scalable, so you can build applications that handle lots of visitors as well as large amounts of information.

Also, Django is cross-platform, which means that your project is not based on Mac, Linux or PC.

Django supports most well-known databases, for example, MySQL, PostgreSQL, which allows the developer to choose the appropriate database system for the project or even more than one database system at the same time.

5) Excellent explanation and documentation

Django offers one of the best documentations compared to other open-source frameworks. The site contains plenty of practical examples and tutorials, not just an explanation of functions.

Advantages of Django:

1. Easy way to make URL links

A clean and neat URL scheme is an important detail of any high-quality web application. Django allows you to design URLs the way you want without any restrictions.

2. Django supports object relational mapping or ORM

It is one of the most powerful features of Django, which enables the developer to treat your database, as you would SQL. In fact, ORM in Django is just a way to generate SQL commands to query your database using Python code, manipulate it, and get results without having to write a single SQL command.

3. data migration process

Migrations are Django's way of publishing changes you make to forms (adding a field, deleting a form, etc.). They're designed to be mostly automated, but you'll need to know when and when migrations are being made and what common issues you might encounter.

4. form language

Template language makes it easy to manipulate variables, functions, and loops within a template for displaying data, as it is a way to write logic for entering HTML pages.

5. MVT System - Form View Form

Django's MVT provides a unique way to separate database, logic, and templates.

MVT is a software design pattern. Where the View represents the controller to receive requests from the user and deal with them in a logical way, the form represents a gateway to dealing with databases, and the template represents the HTML pages that display the data.

6. temporary storing

The main distinguishing feature of dynamic websites is that they are dynamic. Meaning that every time a user requests a page, the web server performs all kinds of calculations - from database queries to push-ups to render to the visitor, but in return this process requires a lot of computation and consumes a lot of material from the server especially on huge sites with high traffic a lot.

Caching something is saving the result of an expensive computation so that the server doesn't have to do the computation next time when the same page is requested.

Django provides caching at more than one level, such as caching at the level of databases, file systems, and even at the level of memory.

Disadvantages of Django:

While Django is an amazing framework, there are a few cons that may or may not be a problem for you. First, the URL specifying with regular expressions is not an easy task to accomplish, at least for beginners. It also feels a little bloated for small projects, and some people find it quite populated with big projects since models, for example, are all included in a single file. Template errors fail silently by default, so if you don’t know that, you might waste a lot of time trying to figure out what’s wrong with the application, or even worse, you might not even know that your application has a problem. It’s also a strongly opinionated framework, which gives it a monolithic feeling. There’s one popular and advised way of doing things and you’re supposed to follow it.

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc).

Internally, while it provides choices for almost any functionality you might want (e.g., several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Django can be used for building the backend of a mobile application as well. Here are some steps to create a mobile app backend using Django:

1) Install Django: First, you need to install Django by running the following command in the terminal:

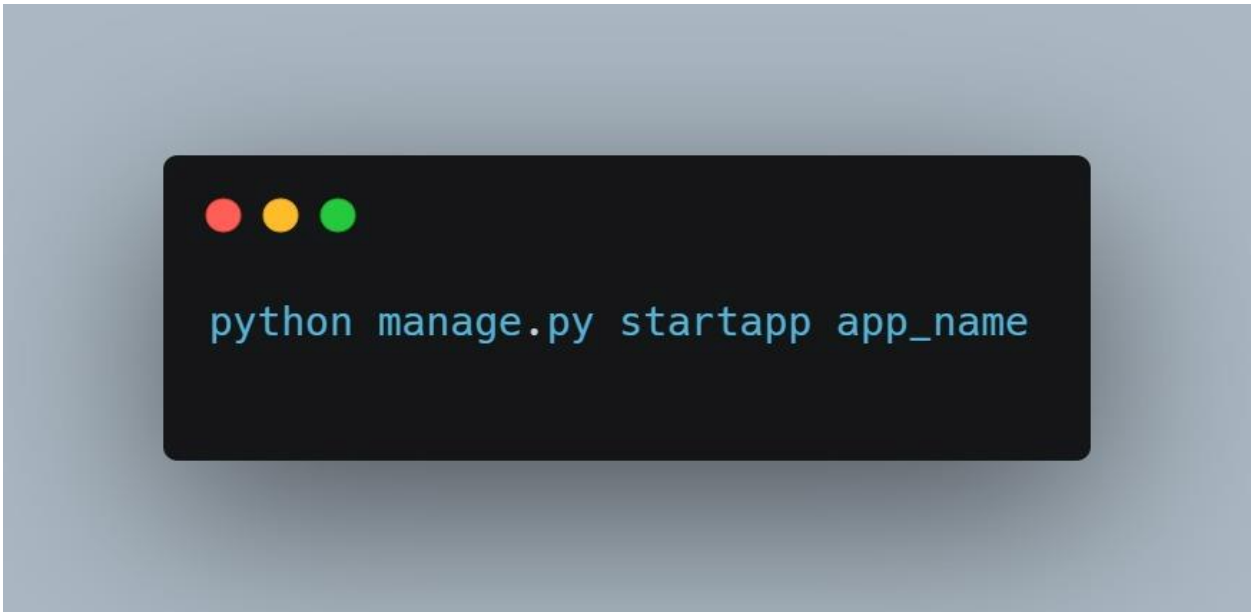
```
pip install django
```

2)Create a new Django project: You can create a new Django project by running the following command in the terminal:

```
django-admin startproject project_name
```

This will create a new directory with the name of your project

3) Create a new Django app: Next, you can create a new Django app within your project by running the following command:



This will create a new directory with the name of your app.

4) Define the models: In Django, you define the models using Python classes that inherit from the django.db.models.Model class. You can define the models for your application based on the requirements of the mobile app.

5. Create APIs: Django provides a built-in module called `django-rest-framework` that makes it easy to create RESTful APIs for your mobile app. You can define the APIs by creating a new view for each API endpoint and mapping it to a URL.

6. Implement authentication: To secure your API endpoints, you can implement authentication using `django-rest-framework` or other authentication modules.

7. Test the APIs: Once you have implemented the APIs, you can test them using tools like Postman or curl.

By following these steps, you can create a robust backend for your mobile application using Django.

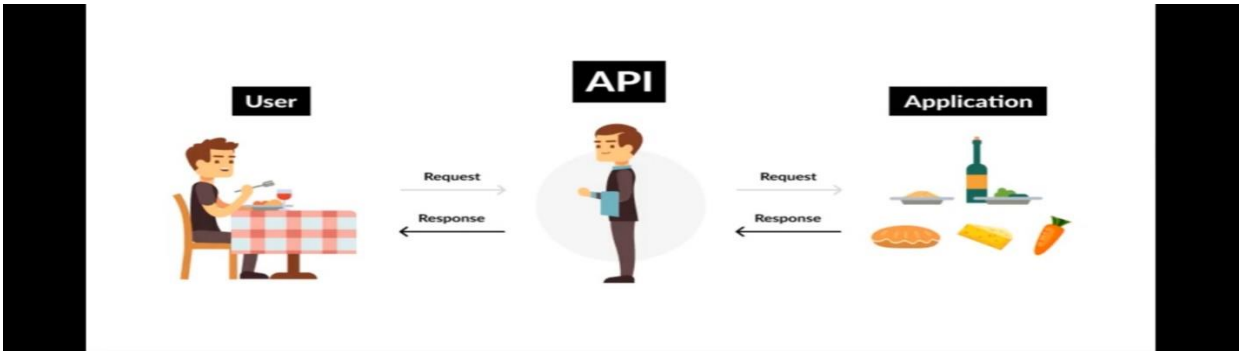
5.2.4.1 API :

An application programming interface (API) is a connection between computers or between computer programs. It is a type of software interface, offering a service to other pieces of software.[1] A document or standard that describes how to build or use such a connection or interface is called an *API specification*. A computer system that meets this standard is said

to implement or expose an API. The term API may refer either to the specification or to the implementation.

In contrast to a user interface, which connects a computer to a person, an application programming interface connects computers or pieces of software to each other. It is not intended to be used directly by a person (the end user) other than a computer programmer who is incorporating it into the software. An API is often made up of different parts which act as tools or services that are available to the programmer. A program or a programmer that uses one of these parts is said to *call* that portion of the API. The calls that make up the API are also known as subroutines, methods, requests, or endpoints. An API specification *defines* these calls, meaning that it explains how to use or implement them.

One purpose of APIs is to hide the internal details of how a system works, exposing only those parts a programmer will find useful and keeping them consistent even if the internal details later change. An API may be custom-built for a particular pair of systems, or it may be a shared standard allowing interoperability among manysystem



To build the API for this app, you could use a framework such as Django REST framework, which provides a powerful and flexible toolkit for building APIs. Here's a high-level overview of the steps involved in building the API:

1. Define the API endpoints: Start by defining the endpoints for the API. For example, you might have an endpoint for uploading an image of a skin lesion, an endpoint for retrieving the diagnosis results, and an endpoint for sending a message to the doctor.

2. Implement the endpoints: Using Django REST framework, you can define the views for each endpoint, along with the serializers that will be used to convert the data to and from JSON format.

3. Secure the API: It's important to secure the API to prevent unauthorized access. You can use authentication and authorization mechanisms provided by Django REST framework to ensure that only authorized users can access the API.

4. Test the API: Before deploying the API, it's important to thoroughly test it to ensure that it's working correctly. You can use tools such as Postman or curl to test each endpoint and verify that the responses are correct.

5. Deploy the API: Once the API is tested and working correctly, you can deploy it to a production server. You can use cloud services such as AWS or Heroku to host the API, or you can deploy it on your own server.

Overall, building an API for this app is an important part of the development process, as it allows the mobile app to communicate with the back-end server and provide users with the diagnosis results and other functionality.

5.2.4.2 Rest Framework API :

For this app, you could consider using Django REST framework (DRF) for building your RESTful API. Django REST framework is a powerful and flexible toolkit for building Web APIs, and it's built on top of the Django web framework, which makes it easy to integrate with your existing Django application.

With DRF, you can quickly build APIs that support multiple data formats, including JSON, XML, and more. It also provides a wide range of features out of the box, including support for authentication, throttling, pagination, and filtering, which makes it easy to build robust and scalable APIs.

To get started with DRF, you would first need to install it in your Django project using pip. Then, you would define your API endpoints using DRF's class-based views and serializers. You can customize the behavior of your API views by overriding DRF's built-in methods, such as get(), post(), and put().

DRF also provides a web-based interface called the browsable API, which allows developers to interact with your API endpoints directly from their browser. This can be a useful tool for testing and debugging your API during development.

Overall, DRF is a great choice for building RESTful APIs in Django, and it has a large and active community that provides support and resources to help you get started.

5.2.4.3 Serializer :

Here's an example of a serializer that could be used for the Skin Cancer Detection app

```
from rest_framework import serializers
from .models import SkinImage

class SkinImageSerializer(serializers.ModelSerializer):
    class Meta:
        model = SkinImage
        fields = ('id', 'image', 'diagnosis', 'confidence')
```

In this example, the serializer is named `SkinImageSerializer` and it is used to convert data from the `SkinImage` model to a format that can be used by the API. The serializer includes four fields: `id`, `image`, `diagnosis`, and `confidence`.

- `id` is the unique identifier for each image.

- `image` is the actual image data, stored as a file.

- `diagnosis` is the result of the skin cancer diagnosis, which will be one of several possible categories (e.g. "melanoma", "basal cell carcinoma", "squamous cell carcinoma", etc.).

- `confidence` is a value between 0 and 1 that represents how confident the algorithm is in its diagnosis.

Once this serializer is defined, it can be used in views to handle incoming requests and generate responses with the appropriate data.

5.2.5 Java:

Java is a general-purpose, class-based, object-oriented programming language designed for having less implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc.

Java is a multi-platform, object-oriented, and network-centric language. It is among the most used programming language. Java is also used as a computing platform.

It is considered as one of the fast, secure, and reliable programming languages preferred by most organizations to build their projects.

What is Java used for?

Here are some important Java applications:

Is is used for developing Android Apps Helps you to create Enterprise Software Wide range of Mobile java Applications Scientific Computing Applications Use for Big Data Analytics Java Programming of Hardware devices Used for Server-Side Technologies like Apache, JBoss, GlassFish, etc. History of Java Programming Language Here are important landmarks from the history of the Java language. The Java language was initially called OAK.

Originally, it was developed for handling portable devices and set-top boxes. Oak was a massive failure.

In 1995, Sun changed the name to “Java” and modified the language to take advantage of the burgeoning www (World Wide Web) development business.

Later, in 2009, Oracle Corporation acquired Sun Microsystems and took ownership of three key Sun software assets: Java, MySQL, and Solaris.

5.2.5.1 Android:

Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a partnership of developers known as the Open Handset Alliance and commercially sponsored by Google. It was disclosed in November 2007, with the first commercial Android device, the HTC Dream, launched in September 2008.

It is free and open-source software. Its source code is Android Open Source Project (AOSP), primarily licensed under the Apache License. However, most Android devices dispatch with additional proprietary software pre-installed, mainly Google Mobile Services (GMS), including core apps such as Google Chrome, the digital distribution platform Google Play and the associated Google Play Services development platform.

About 70% of Android Smartphone runs Google's ecosystem, some with vendor-customized user interface and some with software suite, such as TouchWiz and later *One UI* by Samsung, and *HTC Sense*.

Competing Android ecosystems and forks include Fire OS (developed by Amazon) or Lineage OS. However, the "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use android branding.

Features of Android Operating System

Below are the following unique features and characteristics of the android operating system, such as:

1. Near Field Communication (NFC):

Most Android devices support NFC, which allows electronic devices to interact across short distances easily. The main goal here is to create a payment option that is simpler than carrying cash or credit cards, and while the market hasn't exploded as many experts had predicted, there may be an alternative in the works, in the form of Bluetooth Low Energy (BLE).

2. Infrared Transmission:

The Android operating system supports a built-in infrared transmitter that allows you to use your phone or tablet as a remote control.

3. Automation:

The Tasker app allows control of app permissions and also automates them.

4. Wireless App Downloads:

You can download apps on your PC by using the Android Market or third-party options like AppBrain. Then it automatically syncs them to your Droid, and no plugging is required.

5. Storage and Battery Swap:

Android phones also have unique hardware capabilities. Google's OS makes it possible to upgrade, replace, and remove your battery that no longer holds a charge. In addition, Android phones come with SD card slots for expandable storage.

6. Custom Home Screens:

While it's possible to hack certain phones to customize the home screen, Android comes with this capability from the get-go. Download a third-party launcher like ***Apex, Nova***, and you can add gestures, new shortcuts, or even performance enhancements for older-model devices.

7. Widgets:

Apps are versatile, but sometimes you want information at a glance instead of having to open an app and wait for it to load. Android widgets let you display just about any feature you choose on the home screen, including weather apps, music widgets, or productivity tools that helpfully remind you of upcoming meetings or approaching deadlines.

8. Custom ROMs:

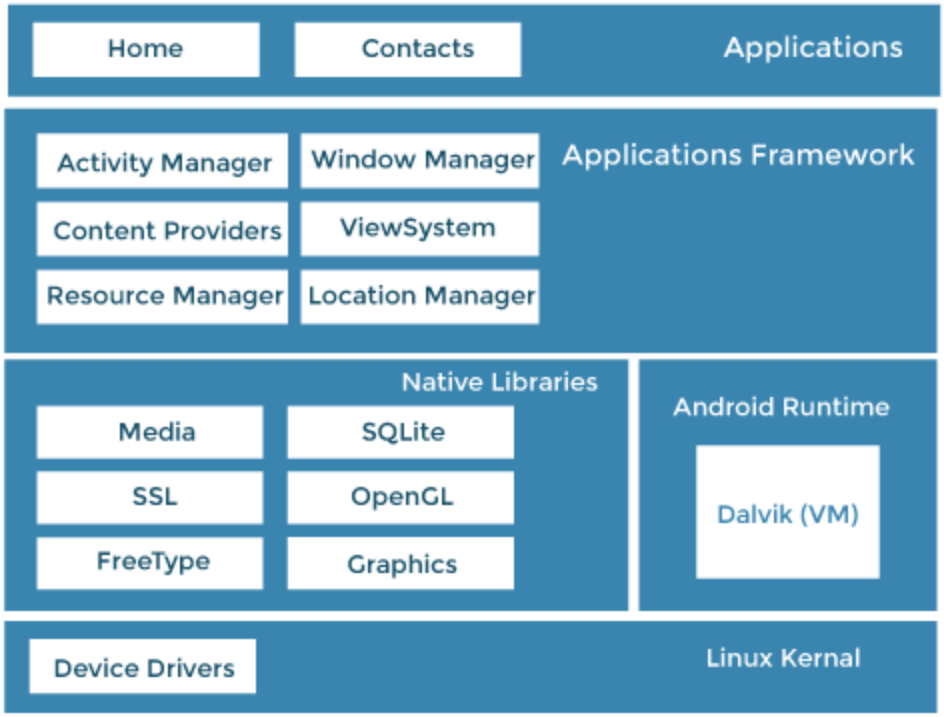
Because the Android operating system is open-source, developers can twist the current OS and build their versions, which users can download and install in place of the stock OS. Some are filled with features, while others change the look and feel of a device. Chances are, if there's a feature you want, someone has already built a custom ROM for it.

Architecture of Android OS

The android architecture contains a different number of components to support any android device needs. Android software contains an open-source Linux Kernel with many C/C++ libraries exposed through application framework services.

Among all the components, Linux Kernel provides the main operating system functions to Smartphone and Dalvik Virtual Machine (DVM) to provide a platform for running an android application. An android operating system is a stack of software components roughly divided into five sections and four main layers, as shown in the below architecture diagram.

- Applications
- Application Framework
- Android Runtime
- Platform Libraries
- Linux Kernel



1. Applications:

An application is the top layer of the android architecture. The pre-installed applications like camera, gallery, home, contacts, etc., and third-party applications downloaded from the play store like games, chat applications, etc., will be installed on this layer.

It runs within the Android run time with the help of the classes and services provided by the application framework.

2. Application framework:

Application Framework provides several important classes used to create an Android application. It provides a generic abstraction for hardware access and helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.

It includes different types of services, such as activity manager, notification manager, view system, package manager etc., which are helpful for the development of our application according to the prerequisite.

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications. The Android framework includes the following key services:

Activity Manager: Controls all aspects of the application lifecycle and activity stack.

Content Providers: Allows applications to publish and share data with other applications.

Resource Manager: Provides access to non-code embedded resources such as strings, colour settings and user interface layouts.

Notifications Manager: Allows applications to display alerts and notifications to the user.

View System: An extensible set of views used to create application user interfaces.

3. Application runtime:

Android Runtime environment contains components like core libraries and the Dalvik virtual machine (DVM). It provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), Dalvik Virtual Machine (DVM) is a register-based virtual machine designed and optimized for Android to ensure that a device can run multiple instances efficiently.

It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

4. Platform libraries:

The Platform Libraries include various C/C++ core libraries and Java-based libraries such as Media, Graphics, Surface Manager, OpenGL, etc., to support Android development.

app: Provides access to the application model and is the cornerstone of all Android applications.

content: Facilitates content access, publishing and messaging between applications and application components.

database: Used to access data published by content providers and includes SQLite database, management classes.

OpenGL: A Java interface to the OpenGL ES 3D graphics rendering API.

os: Provides applications with access to standard operating system services, including messages, system services and inter-process communication.

text: Used to render and manipulate text on a device display.

view: The fundamental building blocks of application user interfaces.

widget: A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.

WebKit: A set of classes intended to allow web-browsing capabilities to be built into applications.

media: Media library provides support to play and record an audio and video format.

surface manager: It is responsible for managing access to the display subsystem.

SQLite: It provides database support, and FreeType provides font support.

SSL: Secure Sockets Layer is a security technology to establish an encrypted link between a web server and a web browser.

5. Linux Kernel:

Linux Kernel is the heart of the android architecture. It manages all the available drivers such as display, camera, Bluetooth, audio, memory, etc., required during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the other android architecture components. It is responsible for the management of memory, power, devices etc. The features of the Linux kernel are:

Security: The Linux kernel handles the security between the application and the system.

Memory Management:

It efficiently handles memory management, thereby providing the freedom to develop our apps.

Process Management:

It manages the process well, allocates resources to processes whenever they need them.

Network Stack:

It effectively handles network communication.

Driver Model:

It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

Android Applications

Android applications are usually developed in the Java language using the Android Software Development Kit. Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play, SlideME, Opera Mobile Store, Mobango, F-droid or the Amazon Appstore.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

Android Emulator

The Emulator is a new application in the Android operating system. The Emulator is a new prototype used to develop and test android applications without using any physical device.

The android emulator has all of the hardware and software features like mobile devices except phone calls. It provides a variety of navigation and control keys. It also provides a screen to display your application. The emulators utilize the android virtual device configurations. Once your application is running on it, it can use services of the android platform to help other applications, access the network, play audio, video, store, and retrieve the data.

Advantages of Android Operating System

We considered every one of the elements on which Android is better as thought about than different platforms. Below are some important advantages of Android OS, such as:

Android Google Developer:

The greatest favourable position of Android is Google. Google claims an android operating system. Google is a standout amongst the most trusted and rumoured item on the web. The name Google gives trust to the clients to purchase Android gadgets.

Android Users:

Android is the most utilized versatile operating system. More than a billion individuals clients utilize it. Android is likewise the quickest developing operating system in the world. Various clients increment the number of applications and programming under the name of Android.

Android Multitasking:

The vast majority of us admire this component of Android. Clients can do heaps of undertakings on the double. Clients can open a few applications on the double and oversee them very. Android has incredible UI, which makes it simple for clients to do multitasking.

Google Play Store App:

The best part of Android is the accessibility of many applications. Google Play store is accounted for as the world's largest mobile store. It has practically everything from motion pictures to amusements and significantly more. These things can be effortlessly downloaded and gotten to through an Android phone.

Android Notification and Easy Access:

Without much of a stretch, one can access their notice of any SMS, messages, or approaches their home screen or the notice board of the android phone. The client can view all the notifications on the top bar. Its UI makes it simple for the client to view more than 5 Android notices immediately.

Android Widget:

Android operating system has a lot of widgets. This gadget improves the client encounter much and helps in doing multitasking. You can include any gadget relying on the component you need on your home screen. You can see warnings, messages, and a great deal more use without opening applications.

Disadvantages of Android Operating System

We know that the Android operating system has a considerable measure of interest for users nowadays. But at the same time, it most likely has a few weaknesses. Below are the following disadvantages of the android operating system, such as:

Android Advertisement pop-ups:

Applications are openly accessible in the Google play store. Yet, these applications begin demonstrating tons of advertisements on the notification bar and over the application. This promotion is extremely difficult and makes a massive issue in dealing with your Android phone.

Android require Gmail ID:

You can't get to an Android gadget without your email ID or password. Google ID is exceptionally valuable in opening Android phone bolts as well.

Android Battery Drain:

Android handset is considered a standout amongst the most battery devouring operating systems In the android operating system, many processes are running out of sight, which brings about the draining of the battery. It is difficult to stop these applications as the lion's share of them is system applications.

Android Malware/Virus/Security:

Android gadget is not viewed as protected when contrasted with different applications. Hackers continue attempting to take your data. It is anything but difficult to target any Android phone, and each day millions of attempts are done on Android phones

Chapter 6

Implementation

6.1 Detection Solution

6.2 Detection Model

6.3 Loading the Data

6.4 Data Scaling

6.5 Data Augmentation

6.6 Building Deep Learning Model

6.7 java Code

6.8 Output

Chapter 6:

6.1 Detection Solution:

There are several motivations for using AI solutions in the detection and diagnosis of skin cancer:

1. Improved Accuracy:

AI algorithms can analyze medical images more accurately and consistently than human doctors, potentially reducing the rate of missed or misdiagnosed skin cancers.

2. Increased Efficiency:

AI solutions can analyze images much more quickly than humans, allowing for more patients to be evaluated in a shorter amount of time. This can help reduce wait times for appointments and improve patient outcomes.

3. Increased Access:

AI solutions can potentially increase access to skin cancer screening and diagnosis for people who may not have easy access to dermatologists or other healthcare providers.

4. Reduced Healthcare Costs:

By reducing the need for unnecessary biopsies and follow-up appointments, AI solutions may help reduce healthcare costs associated with skin cancer diagnosis and treatment.

Overall, AI solutions have the potential to improve the accuracy, efficiency, and accessibility of skin cancer detection and diagnosis, which can lead to better outcomes for patients and reduce the burden on the healthcare system.

6.2 Detection Model :

Importing Libraries We Used In Our Model

This is a Python code snippet that imports several libraries and frameworks, initializes some variables and clears the current Keras session. Here is a breakdown of what each line does:

- import NumPy as np imports the NumPy library, which is used for numerical computing in Python. It is often used for working with arrays and matrices.

- import pandas as pd imports the Pandas library, which is used for data manipulation and analysis in Python. It is often used for working with structured data and tabular data.

- import matplotlib.pyplot as plt imports the Matplotlib library, which is used for data visualization in Python. It is often used for creating plots and charts.

- import TensorFlow as tf imports the TensorFlow library, which is a popular open-source framework for building and training machine learning models.

- from TensorFlow import keras imports the Keras API from TensorFlow, which provides a high-level interface for building and training machine learning models.

- from tensorflow.keras import layers imports the layers module from the Keras API in TensorFlow. This module contains a variety of predefined layers that can be used to build neural networks.

- from tensorflow.keras.models import Sequential imports the Sequential model from the Keras API in TensorFlow. This is a simple type of neural network that consists of a sequence of layers.

- from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten imports specific types of layers from the Keras API in TensorFlow. These layers are commonly used in convolutional neural networks (CNNs), which are used for image recognition and classification.

- from tensorflow.keras.metrics import BinaryAccuracy, Precision, Recall imports specific metrics from the Keras API in TensorFlow. These metrics are used to evaluate the performance of machine learning models.

- import warnings imports the warnings module, which is used to suppress warning messages that may be displayed during the execution of the code.

- warnings.filterwarnings("ignore") turns off warning messages that may be displayed during the execution of the code.

- tf.keras.backend.clear_session() clears the current Keras session, which removes any previously defined models or layers from memory. This is often done to ensure that the memory is freed up before starting to build a new model

In[5]

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense,
Flatten
from tensorflow.keras.metrics import BinaryAccuracy, Precision, Recall
import warnings
warnings.filterwarnings("ignore")
tf.keras.backend.clear_session()
```

6.3 Loading the Data:

This code snippet uses the Keras API in TensorFlow to load image datasets from directories. The image_dataset_from_directory() function is used to create a dataset object from the images in the specified directories. Here is a breakdown of what each line does:

- test_data = keras.utils.image_dataset_from_directory("/kaggle/input/skin-cancer-malignant-vs-benign/test") loads the image dataset from the "/kaggle/input/skin-cancer-malignant-vs-benign/test" directory into the test_data variable. The images in this directory are assumed to be organized into subdirectories based on their class labels (e.g., "malignant" or "benign"). The image_dataset_from_directory() function automatically splits the images into batches and creates a labeled dataset that can be used to train a machine learning model.

- train_data = keras.utils.image_dataset_from_directory("/kaggle/input/skin-cancer-malignant-vs-benign/train") loads the image dataset from the "/kaggle/input/skin-cancer-malignant-vs-benign/train" directory into the train_data variable. This directory contains the training data, which is used to train the machine learning model.

Note that the image_dataset_from_directory() function has several optional parameters that can be used to customize how the images are loaded and preprocessed. For example, you can specify the size of the images, the batch size, and whether to shuffle the data before each epoch. These parameters can be used to fine-tune the performance of the machine learning model.

In [6]:

```
test_data = keras.utils.image_dataset_from_directory("/kaggle/input/skin-cancer-malignant-vs-
benign/test")
train_data =  keras.utils.image_dataset_from_directory("/kaggle/input/skin-cancer-malignant-vs-
benign/train")
```

- Found 660 files belonging to 2 classes.
- Found 2637 files belonging to 2 classes.
- Plotting a few Example Images

This code snippet visualizes a batch of images from the training data using Matplotlib. Here is a breakdown of what each line does:

- batch = train_data.as_numpy_iterator().next() converts the train_data object (which is a TensorFlow dataset object) into a NumPy array using the as_numpy_iterator() method. The next() method is then used to get the first batch of images from the dataset.

- fig, ax = plt.subplots(3, 5, figsize=(15,10)) creates a 3x5 grid of subplots using the subplots() function in Matplotlib. The figsize parameter specifies the size of the figure in inches.

- ax = ax.flatten() flattens the array of subplots into a 1-dimensional array, so that they can be easily accessed using a loop.

- for idx, img in enumerate(batch[0][:15]): iterates over the first 15 images in the batch, using the enumerate() function to get both the index (idx) and the image itself (img).

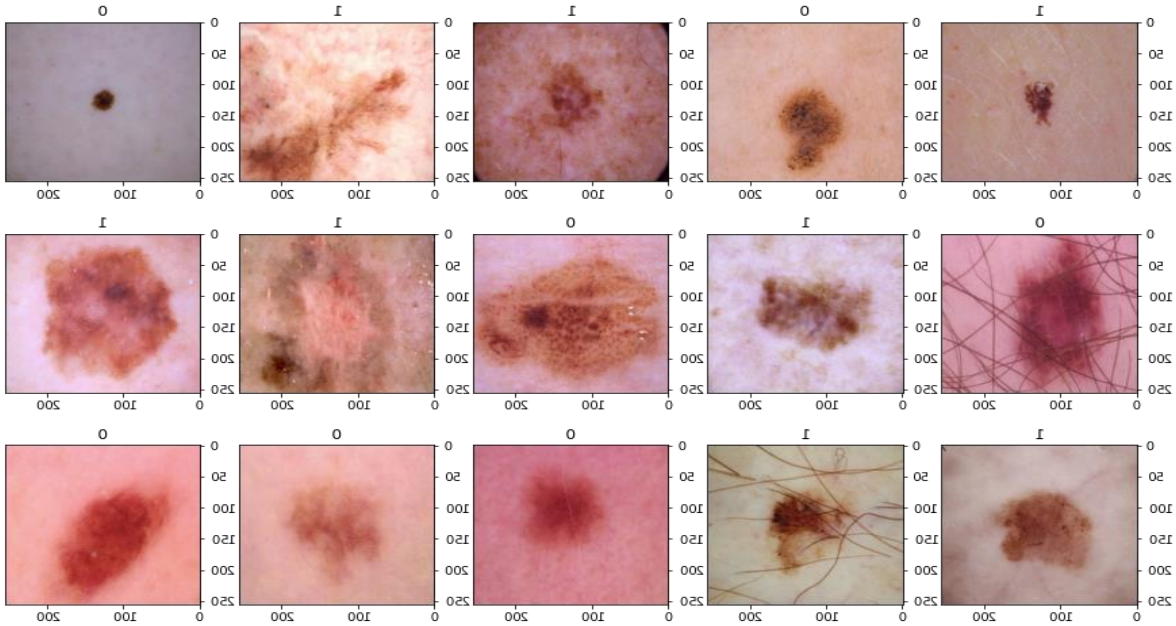
- ax[idx].imshow(img.astype(int)) displays the image on the idxth subplot using the imshow() function in Matplotlib. The astype(int) method is used to convert the image to an integer data type, which is required by Matplotlib.

- ax[idx].title.set_text(batch[1][idx]) sets the title of the idxth subplot to the label of the corresponding image in the batch (batch[1][idx]).

The result is a grid of 15 images with their corresponding labels, which can be used to visually inspect the quality of the training data and check if the labels are correct.

In [7]:

```
batch = train_data.as_numpy_iterator().next()
fig, ax = plt.subplots(3, 5, figsize=(15,10))
ax = ax.flatten()
for idx, img in enumerate(batch[0][:15]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```



6.4 Data Scaling:

This code snippet pre-processes the image data in the train_data and test_data datasets by normalizing the pixel values. Here is a breakdown of what each line does:

- train_data = train_data.map(lambda x,y: (x/255, y)) applies a lambda function to each element of the train_data dataset using the map() method. The lambda function takes two arguments x and y, which represent the input image and its corresponding label, respectively. Inside the lambda function, the image is divided by 255 to scale the pixel

values to the range [0,1]. This is a common normalization technique used in deep learning to improve the performance of the model.

- test_data = test_data.map(lambda x,y: (x/255, y)) applies the same normalization process to the test_data dataset. This ensures that the test images are pre-processed in the same way as the training images, which is important for accurate evaluation of the model's performance.

After this pre-processing step, the pixel values of all the images in the datasets will be in the range [0,1] instead of [0,255]. This normalization step can help the machine learning model converge faster and reduce the risk of overfitting to the training data.

In [8]:

```
train_data = train_data.map(lambda x,y: (x/255,
y))
test_data = test_data.map(lambda x,y: (x/255, y))
```

This code snippet gets a batch of images and their corresponding labels from the train_data dataset. Here is a breakdown of what each line does:

1. train_data is a TensorFlow dataset object that contains the training images and their corresponding labels.
2. train_data.as_numpy_iterator() returns an iterator that can be used to iterate over the elements of the train_data dataset as NumPy arrays.
3. next() is used to get the first element of the iterator, which corresponds to a batch of images and their corresponding labels.

batch is assigned to the batch of images and labels obtained from the previous step.

After executing this code snippet, the batch variable will contain a tuple with two elements: a NumPy array of images with shape (batch_size, image_width, image_height, num_channels) and a NumPy array of labels with shape (batch_size,). The batch_size is a hyperparameter that determines the number of images per batch, while the image_width, image_height, and num_channels correspond to the dimensions of the images in pixels and the number of color channels (e.g., 3 for RGB images). This batch of images and labels can be used to train a machine learning model using the TensorFlow framework.

In [9]:

```
batch = train_data.as_numpy_iterator().next()
```

This code snippet prints the minimum and maximum values of the batch of images obtained from the train_data dataset after normalizing their pixel values. Here is a breakdown of what each line does:

- print("Minimum value of the scaled data:", batch[0].min()) prints a message to the console indicating that the minimum value of the scaled data is going to be printed, followed by the actual minimum value of the image data in the batch variable, which is obtained using the .min() method.
- print("Maximum value of the scaled data:", batch[0].max()) prints a message to the console indicating that the maximum value of the scaled data is going to be printed, followed by the actual maximum value of the image data in the batch variable, which is obtained using the .max() method.

These two lines of code are used to verify that the pre-processing step of normalizing the pixel values of the images in the train_data dataset was successful. Since the images were divided by 255 in the normalization step, their pixel values should now be in the range [0,1]. Therefore, the minimum value should be close to 0 and the maximum value should be close to 1. By printing these values to the console, we can ensure that the pixel values of the images are in the expected range and that the normalization step was successful.

In [10]:

```
print("Minimum value of the scaled data:", batch[0].min())
print("Maximum value of the scaled data:", batch[0].max())
```

- Minimum value of the scaled data: 0.0
- Maximum value of the scaled data: 1.0

6.5 Data Augmentation:

This code snippet gets a batch of images and their corresponding labels from the train_data dataset. Here is a breakdown of what each line does:

1. train_data is a TensorFlow dataset object that contains the training images and their corresponding labels.
2. train_data.as_numpy_iterator() returns an iterator that can be used to iterate over the elements of the train_data dataset as NumPy arrays.
3. next() is used to get the first element of the iterator, which corresponds to a batch of images and their corresponding labels.

batch is assigned to the batch of images and labels obtained from the previous step.

After executing this code snippet, the batch variable will contain a tuple with two elements: a NumPy array of images with shape (batch_size, image_width, image_height, num_channels) and a NumPy array of labels with shape (batch_size,). The batch_size is a hyperparameter that determines the number of images per batch, while the image_width, image_height, and num_channels correspond to the dimensions of the images in pixels and the number of color channels (e.g., 3 for RGB images). This batch of images and labels can be used to train a machine learning model using the TensorFlow framework.

In [11]:

```
batch = train_data.as_numpy_iterator().next()
```

This code snippet defines a data augmentation pipeline using the Keras Sequential API, and applies the pipeline to a batch of images to generate augmented images. Here is a breakdown of what each line does:

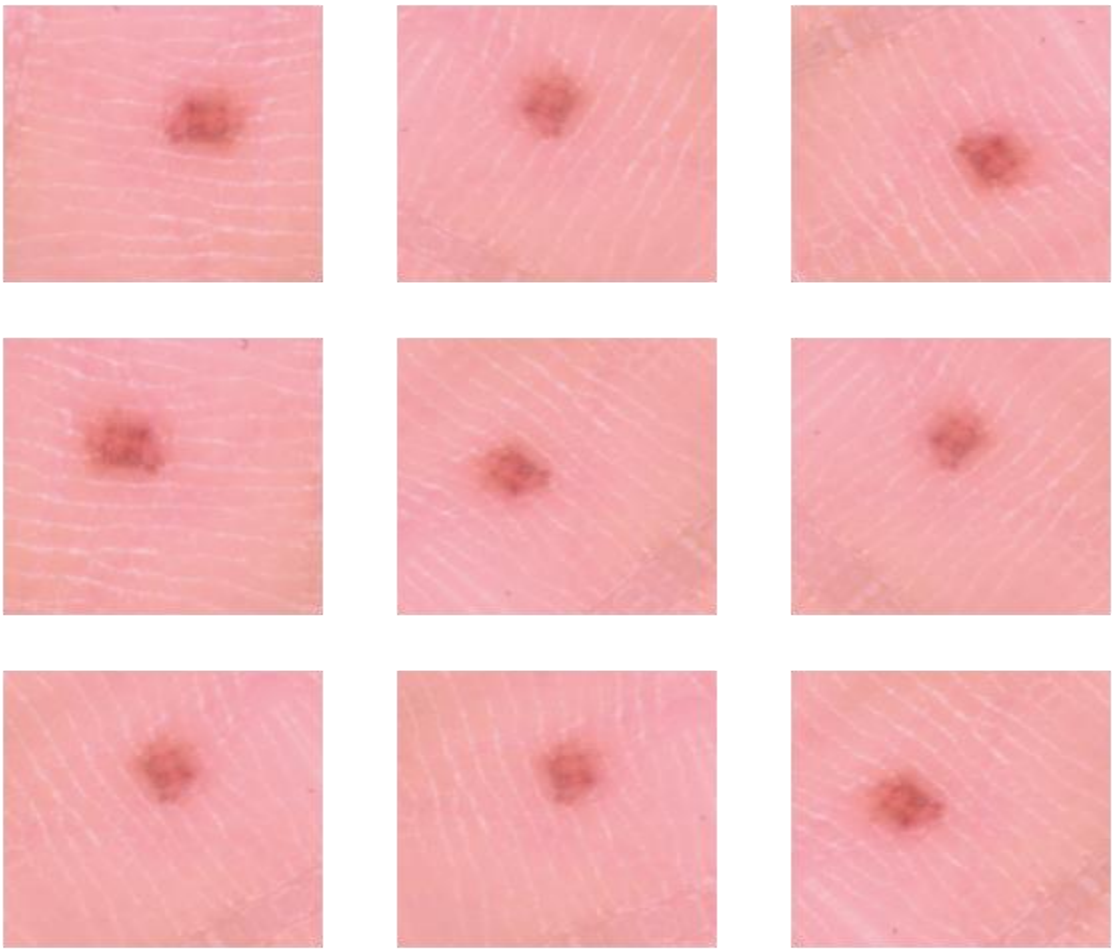
- data_augmentation =Sequential([...]) defines a Keras Sequential model called data_augmentation that sequentially applies several image augmentation techniques to an input image. The augmentation techniques used in this example include horizontal and vertical flipping, random zooming, random contrast adjustment, and random rotation.

- image = batch[0] assigns the first image in the batch variable (which was previously obtained from the train_data dataset) to a new variable called image.
- plt.figure(figsize=(10, 10)) creates a new figure with a size of 10x10 inches using the Matplotlib library.
- for i in range(9): starts a loop that will iterate 9 times.
- augmented_image = data_augmentation(image) applies the data augmentation pipeline defined in data_augmentation to the image variable to generate an augmented image.
- ax = plt.subplot(3, 3, i + 1) creates a new subplot with a grid of 3 rows and 3 columns, and assigns it to the ax variable.
- plt.imshow(augmented_image[0]) displays the augmented image generated in the current iteration of the loop.
- plt.axis("off") removes the axis labels and ticks from the current subplot.

By executing this code snippet, a figure containing 9 augmented images will be displayed, each one generated by applying the data augmentation pipeline defined in data_augmentation to the same input image. This is a useful technique to increase the size and diversity of the training dataset, which can help improve the performance of a machine learning model by reducing overfitting.

In [12]:

```
data_augmentation = Sequential([
    layers.RandomFlip("horizontal_and_vertical", input_shape=
(256, 256, 3)),
    layers.RandomZoom(0.1),
    layers.RandomContrast(0.1),
    layers.RandomRotation(0.2)
])

image = batch[0]


plt.figure(figsize=(10, 10))
for i in range(9):
    augmented_image = data_augmentation(image)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_image[0])
    plt.axis("off")
```



6.6 Building Deep Learning Model:

This code defines a Convolutional Neural Network (CNN) using the Keras Sequential API to classify images of skin lesions as benign or malignant. Here's what each line does:

- model = Sequential([...]) creates a new Keras Sequential model called model, which is initialized as an empty neural network.
- data_augmentation is a data augmentation pipeline defined in a previous code snippet.
- Conv2D(16, (3,3), 1, activation="relu", padding="same") creates a Convolutional layer with 16 filters, a kernel size of (3,3), a stride of 1, ReLU activation, and same padding. This layer convolves the input image with 16 different filters to extract features.
- MaxPooling2D() creates a MaxPooling layer that performs down-sampling along the spatial dimensions of the output feature map, by taking the maximum value of each non-overlapping rectangular patch.

- The network architecture continues with more Conv2D and MaxPooling2D layers, with increasing number of filters, kernel sizes, and stride.
- Flatten() creates a Flatten layer that flattens the output of the last convolutional layer into a 1D array.
- Dense(128, activation="relu") creates a fully connected Dense layer with 128 units and ReLU activation. This layer receives the flattened output of the previous layer as input, and applies a linear transformation followed by a non-linear activation function to generate a feature vector.
- Dense(1, activation="sigmoid") creates a final Dense layer with 1 unit and sigmoid activation, which outputs a probability value between 0 and 1 indicating the predicted probability of the input image belonging to the positive class (i.e., malignant).

By stacking the different types of layers in the desired order, this model defines a CNN architecture that takes an input image, applies several convolutional and pooling layers to extract features, flattens the output into a 1D array, passes it through a fully connected layer to further process the features, and generates a binary prediction of the input image.

In [13]:

```
model = Sequential([
    data_augmentation,
    Conv2D(16, (3,3), 1, activation="relu", padding="same"),
    Conv2D(16, (3,3), 1, activation="relu", padding="same"),
    MaxPooling2D(),
    Conv2D(32, (5,5), 1, activation="relu", padding="same"),
    Conv2D(32, (5,5), 1, activation="relu", padding="same"),
    MaxPooling2D(),
    Conv2D(16, (3,3), 1, activation="relu", padding="same"),
    Conv2D(16, (3,3), 1, activation="relu", padding="same"),
    MaxPooling2D(),

    Flatten(),
    Dense(128, activation="relu"),
    Dense(1, activation="sigmoid")
])
```

This code compiles a neural network model with the binary cross-entropy loss function, Adam optimizer, and three evaluation metrics: precision, recall, and accuracy.

- loss='binary_crossentropy': This sets the loss function to binary cross-entropy, which is commonly used for binary classification problems.
- optimizer='adam': This sets the optimizer to Adam, which is a popular optimizer used in deep learning.
- metrics=[tf.keras.metrics.Precision(), tf.keras.metrics.Recall(), "acc"]: This sets the evaluation metrics to be used during training and testing. Precision and recall are useful metrics for evaluating binary classification models, while "acc" represents the overall classification accuracy. The tf.keras.metrics module provides several pre-defined evaluation metrics that can be used directly in the metrics argument.

In [14]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=[tf.keras.metrics.Precision(), tf.keras.metrics.Recall(), "acc"]
```

This code prints a summary of the neural network model that has been defined previously. The summary method provides a convenient way to inspect the structure of the model, including the number of layers, the shape of the output tensors at each layer, and the number of trainable parameters.

The output of model.summary() typically includes the following columns:

- Layer: The name of the layer in the model.
- Output Shape: The shape of the output tensor produced by the layer.
- Param #: The number of trainable parameters in the layer.
- Connected to: The name of the layer(s) that this layer is connected to.

The summary also includes a line at the bottom showing the total number of trainable parameters in the model. This information is useful for estimating the computational resources required to train the model, and can be helpful in optimizing the model architecture.

In [15]:

```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 256, 256, 3)	0
conv2d (Conv2D)	(None, 256, 256, 16)	448
conv2d_1 (Conv2D)	(None, 256, 256, 16)	2320
max_pooling2d (MaxPooling2D)	(None, 128, 128, 16)	0
conv2d_2 (Conv2D)	(None, 128, 128, 32)	12832
conv2d_3 (Conv2D)	(None, 128, 128, 32)	25632
max_pooling2d_1 (MaxPooling2	(None, 64, 64, 32)	0
conv2d_4 (Conv2D)	(None, 64, 64, 16)	4624
conv2d_5 (Conv2D)	(None, 64, 64, 16)	2320
max_pooling2d_2 (MaxPooling2	(None, 32, 32, 16)	0

flatten (Flatten) (None, 16384) 0

_____

dense (Dense) (None, 128) 2097280

_____

dense_1 (Dense) (None, 1) 129

==========

Total params: 2,145,585

Trainable params: 2,145,585

Non-trainable params: 0

_____

This code trains the previously defined neural network model (model) using the fit method in TensorFlow.

The fit method trains the model on the training data for a specified number of epochs. In this case, the number of epochs is set to 100. During training, the model will update its weights and biases to minimize the loss function specified in the compile method. The train_data argument is the training dataset that the model will use to learn the patterns in the input data. The validation_data argument is the test dataset that the model will use to evaluate its performance after each epoch.

The fit method returns a History object that contains information about the training process, such as the loss and metrics (e.g., accuracy, precision, recall) values computed for each epoch on both the training and validation data. This object is assigned to the variable history so that we can later access and visualize the training and validation metrics.

In [16]:

```
history = model.fit(train_data, epochs=100, validation_data=test_data)
```

Epoch 1/100

83/83 [==========] - 22s 154ms/step - loss: 0.6474 - precision: 0.6002 - recall: 0.4027 - acc: 0.6071 - val_loss: 0.5228 - val_precision: 0.6726 - val_recall: 0.8767 - val_acc: 0.7500

Epoch 2/100

83/83 [==========] - 8s 91ms/step - loss: 0.5408 - precision: 0.6649 - recall: 0.8354 - acc: 0.7342 - val_loss: 0.4822 - val_precision: 0.6713 - val_recall: 0.9667 - val_acc: 0.7697

Epoch 3/100

83/83 [==========] - 8s 87ms/step - loss: 0.4801 - precision: 0.6765 - recall: 0.9031 - acc: 0.7600 - val_loss: 0.4599 - val_precision: 0.6791 - val_recall: 0.9733 - val_acc: 0.7788

Epoch 4/100

83/83 [==========] - 7s 84ms/step - loss: 0.4589 - precision: 0.6914 - recall: 0.9114 - acc: 0.7751 - val_loss: 0.3987 - val_precision: 0.7136 - val_recall: 0.9300 - val_acc: 0.7985

Epoch 5/100

83/83 [==========] - 7s 84ms/step - loss: 0.4661 - precision: 0.7055 - recall: 0.8647 - acc: 0.7747 - val_loss: 0.4044 - val_precision: 0.6831 - val_recall: 0.9700 - val_acc: 0.7818

Epoch 6/100

83/83 [==========] - 8s 92ms/step - loss: 0.4299 - precision: 0.7219 - recall: 0.8914 - acc: 0.7948 - val_loss: 0.4341 - val_precision: 0.8040 - val_recall: 0.8067 - val_acc: 0.8227

Epoch 7/100

83/83 [==========] - 8s 85ms/step - loss: 0.4427 - precision: 0.7123 - recall: 0.8438 - acc: 0.7744 - val_loss: 0.4204 - val_precision: 0.7216 - val_recall: 0.9333 - val_acc: 0.8061

Epoch 8/100

83/83 [==========] - 7s 85ms/step - loss: 0.4286 - precision: 0.7340 - recall: 0.8830 - acc: 0.8017 - val_loss: 0.4293 - val_precision: 0.7794 - val_recall: 0.7300 - val_acc: 0.7833

Epoch 9/100

83/83 [==========] - 8s 87ms/step - loss: 0.4128 - precision: 0.7420 - recall: 0.8697 - acc: 0.8036 - val_loss: 0.3659 - val_precision: 0.7432 - val_recall: 0.9167 - val_acc: 0.8182

Epoch 10/100

83/83 [==========] - 7s 85ms/step - loss: 0.4004 - precision: 0.7555 - recall: 0.8647 - acc: 0.8115 - val_loss: 0.3556 - val_precision: 0.7568 - val_recall: 0.9233 - val_acc: 0.8303

Epoch 11/100

83/83 [==========] - 8s 84ms/step - loss: 0.3918 - precision: 0.7607 - recall: 0.8739 - acc: 0.8180 - val_loss: 0.3577 - val_precision: 0.7893 - val_recall: 0.8867 - val_acc: 0.8409

Epoch 12/100

83/83 [==========] - 7s 85ms/step - loss: 0.3771 - precision: 0.7766 - recall: 0.8855 - acc: 0.8324 - val_loss: 0.3784 - val_precision: 0.8211 - val_recall: 0.7800 - val_acc: 0.8227

Epoch 13/100

83/83 [==========] - 8s 93ms/step - loss: 0.3773 - precision: 0.7589 - recall: 0.8705 - acc: 0.8157 - val_loss: 0.3793 - val_precision: 0.7577 - val_recall: 0.9067 - val_acc: 0.8258

Epoch 14/100

83/83 [==========] - 7s 83ms/step - loss: 0.3720 - precision: 0.7835 - recall: 0.8647 - acc: 0.8301 - val_loss: 0.3435 - val_precision: 0.7639 - val_recall: 0.9167 - val_acc: 0.8333

Epoch 15/100

83/83 [==========] - 8s 87ms/step - loss: 0.3685 - precision: 0.7654 - recall: 0.8805 - acc: 0.8233 - val_loss: 0.3529 - val_precision: 0.7682 - val_recall: 0.9167 - val_acc: 0.8364

Epoch 16/100

83/83 [==========] - 8s 90ms/step - loss: 0.3538 - precision: 0.7841 - recall: 0.8830 - acc: 0.8366 - val_loss: 0.3723 - val_precision: 0.7254 - val_recall: 0.9333 - val_acc: 0.8091

Epoch 17/100

83/83 [==========] - 7s 83ms/step - loss: 0.3454 - precision: 0.7955 - recall: 0.8805 - acc: 0.8430 - val_loss: 0.3769 - val_precision: 0.7712 - val_recall: 0.9100 - val_acc: 0.8364

Epoch 18/100

83/83 [==========] - 7s 85ms/step - loss: 0.3683 - precision: 0.7816 - recall: 0.8672 - acc: 0.8297 - val_loss: 0.3830 - val_precision: 0.7354 - val_recall: 0.9267 - val_acc: 0.8152

Epoch 19/100

83/83 [==========] - 8s 85ms/step - loss: 0.3563 - precision: 0.7992 - recall: 0.8680 - acc: 0.8411 - val_loss: 0.3507 - val_precision: 0.7668 - val_recall: 0.8767 - val_acc: 0.8227

Epoch 20/100

83/83 [==========] - 8s 92ms/step - loss: 0.3427 - precision: 0.7943 - recall: 0.8839 - acc: 0.8434 - val_loss: 0.3404 - val_precision: 0.7812 - val_recall: 0.8567 - val_acc: 0.8258

Epoch 21/100

83/83 [==========] - 8s 86ms/step - loss: 0.3416 - precision: 0.7947 - recall: 0.8764 - acc: 0.8411 - val_loss: 0.3584 - val_precision: 0.7644 - val_recall: 0.8867 - val_acc: 0.8242

Epoch 22/100

83/83 [==========] - 7s 84ms/step - loss: 0.3353 - precision: 0.7939 - recall: 0.8755 - acc: 0.8403 - val_loss: 0.3408 - val_precision: 0.7834 - val_recall: 0.8800 - val_acc: 0.8348

Epoch 23/100

83/83 [==========] - 8s 90ms/step - loss: 0.3373 - precision: 0.8006 - recall: 0.8755 - acc: 0.8445 - val_loss: 0.3616 - val_precision: 0.7427 - val_recall: 0.9333 - val_acc: 0.8227

Epoch 24/100

83/83 [==========] - 7s 84ms/step - loss: 0.3388 - precision: 0.7934 - recall: 0.8981 - acc: 0.8476 - val_loss: 0.3423 - val_precision: 0.7827 - val_recall: 0.8767 - val_acc: 0.8333

Epoch 25/100

83/83 [==========] - 7s 83ms/step - loss: 0.3107 - precision: 0.8208 - recall: 0.8764 - acc: 0.8570 - val_loss: 0.3650 - val_precision: 0.7562 - val_recall: 0.9200 - val_acc: 0.8288

Epoch 26/100

83/83 [==========] - 8s 87ms/step - loss: 0.3415 - precision: 0.7958 - recall: 0.8922 - acc: 0.8472 - val_loss: 0.3593 - val_precision: 0.7927 - val_recall: 0.8667 - val_acc: 0.8364

Epoch 27/100

83/83 [==========] - 8s 93ms/step - loss: 0.3239 - precision: 0.7979 - recall: 0.8805 - acc: 0.8445 - val_loss: 0.3719 - val_precision: 0.7444 - val_recall: 0.8933 - val_acc: 0.8121

Epoch 28/100

83/83 [==========] - 7s 83ms/step - loss: 0.3227 - precision: 0.7961 - recall: 0.8906 - acc: 0.8468 - val_loss: 0.3488 - val_precision: 0.7771 - val_recall: 0.8833 - val_acc: 0.8318

Epoch 29/100

83/83 [==========] - 7s 84ms/step - loss: 0.3365 - precision: 0.8101 - recall: 0.8555 - acc: 0.8434 - val_loss: 0.3833 - val_precision: 0.7842 - val_recall: 0.8600 - val_acc: 0.8288

Epoch 30/100

83/83 [==========] - 7s 83ms/step - loss: 0.3405 - precision: 0.7808 - recall: 0.8780 - acc: 0.8328 - val_loss: 0.3456 - val_precision: 0.7730 - val_recall: 0.8967 - val_acc: 0.8333

Epoch 31/100

83/83 [==========] - 8s 90ms/step - loss: 0.3186 - precision: 0.8025 - recall: 0.8964 - acc: 0.8529 - val_loss: 0.3504 - val_precision: 0.7829 - val_recall: 0.8533 - val_acc: 0.8258

Epoch 32/100

83/83 [==========] - 7s 82ms/step - loss: 0.3144 - precision: 0.8169 - recall: 0.8906 - acc: 0.8597 - val_loss: 0.3699 - val_precision: 0.7405 - val_recall: 0.9133 - val_acc: 0.8152

Epoch 33/100

83/83 [==========] - 7s 83ms/step - loss: 0.3088 - precision: 0.8102 - recall: 0.8881 - acc: 0.8548 - val_loss: 0.3500 - val_precision: 0.7755 - val_recall: 0.8867 - val_acc: 0.8318

Epoch 34/100

83/83 [==========] - 8s 90ms/step - loss: 0.3395 - precision: 0.7997 - recall: 0.8872 - acc: 0.8479 - val_loss: 0.3764 - val_precision: 0.7225 - val_recall: 0.9633 - val_acc: 0.8152

Epoch 35/100

83/83 [==========] - 7s 84ms/step - loss: 0.3211 - precision: 0.8039 - recall: 0.8906 - acc: 0.8517 - val_loss: 0.3423 - val_precision: 0.7547 - val_recall: 0.9333 - val_acc: 0.8318

Epoch 36/100

83/83 [==========] - 7s 85ms/step - loss: 0.3189 - precision: 0.7977 - recall: 0.8830 - acc: 0.8453 - val_loss: 0.3619 - val_precision: 0.7421 - val_recall: 0.9400 - val_acc: 0.8242

Epoch 37/100

83/83 [==========] - 7s 85ms/step - loss: 0.3103 - precision: 0.8253 - recall: 0.8881 - acc: 0.8639 - val_loss: 0.3569 - val_precision: 0.7771 - val_recall: 0.9067 - val_acc: 0.8394

Epoch 38/100

83/83 [==========] - 7s 84ms/step - loss: 0.3080 - precision: 0.8080 - recall: 0.8964 - acc: 0.8563 - val_loss: 0.3322 - val_precision: 0.7933 - val_recall: 0.8700 - val_acc: 0.8379

Epoch 39/100

83/83 [==========] - 9s 98ms/step - loss: 0.3107 - precision: 0.8224 - recall: 0.9014 - acc: 0.8669 - val_loss: 0.3519 - val_precision: 0.7507 - val_recall: 0.9333 - val_acc: 0.8288

Epoch 40/100

83/83 [==========] - 8s 87ms/step - loss: 0.3185 - precision: 0.8173 - recall: 0.8855 - acc: 0.8582 - val_loss: 0.3640 - val_precision: 0.7381 - val_recall: 0.9300 - val_acc: 0.8182

Epoch 41/100

83/83 [==========] - 8s 91ms/step - loss: 0.3099 - precision: 0.8192 - recall: 0.8822 - acc: 0.8582 - val_loss: 0.3360 - val_precision: 0.8049 - val_recall: 0.8800 - val_acc: 0.8485

Epoch 42/100

83/83 [==========] - 7s 84ms/step - loss: 0.3111 - precision: 0.8091 - recall: 0.8889 - acc: 0.8544 - val_loss: 0.3324 - val_precision: 0.7838 - val_recall: 0.8700 - val_acc: 0.8318

Epoch 43/100

83/83 [==========] - 8s 86ms/step - loss: 0.2976 - precision: 0.8096 - recall: 0.8989 - acc: 0.8582 - val_loss: 0.3232 - val_precision: 0.7787 - val_recall: 0.9033 - val_acc: 0.8394

Epoch 44/100

83/83 [==========] - 7s 83ms/step - loss: 0.3122 - precision: 0.8036 - recall: 0.8855 - acc: 0.8498 - val_loss: 0.3548 - val_precision: 0.7798 - val_recall: 0.8733 - val_acc: 0.8303

Epoch 45/100

83/83 [==========] - 8s 91ms/step - loss: 0.2982 - precision: 0.8100 - recall: 0.8972 - acc: 0.8578 - val_loss: 0.3283 - val_precision: 0.7957 - val_recall: 0.8700 - val_acc: 0.8394

Epoch 46/100

83/83 [==========] - 7s 82ms/step - loss: 0.3070 - precision: 0.8137 - recall: 0.8830 - acc: 0.8551 - val_loss: 0.3181 - val_precision: 0.7813 - val_recall: 0.8933 - val_acc: 0.8379

Epoch 47/100

83/83 [==========] - 8s 87ms/step - loss: 0.3054 - precision: 0.8174 - recall: 0.8939 - acc: 0.8612 - val_loss: 0.3509 - val_precision: 0.7727 - val_recall: 0.9067 - val_acc: 0.8364

Epoch 48/100

83/83 [==========] - 7s 84ms/step - loss: 0.2980 - precision: 0.8176 - recall: 0.8872 - acc: 0.8589 - val_loss: 0.3522 - val_precision: 0.7913 - val_recall: 0.8467 - val_acc: 0.8288

Epoch 49/100

83/83 [==========] - 7s 83ms/step - loss: 0.2942 - precision: 0.8199 - recall: 0.8822 - acc: 0.8586 - val_loss: 0.3372 - val_precision: 0.7708 - val_recall: 0.8967 - val_acc: 0.8318

Epoch 50/100

83/83 [==========] - 8s 87ms/step - loss: 0.2861 - precision: 0.8261 - recall: 0.9006 - acc: 0.8688 - val_loss: 0.3428 - val_precision: 0.8117 - val_recall: 0.8333 - val_acc: 0.8364

Epoch 51/100

83/83 [==========] - 7s 82ms/step - loss: 0.2962 - precision: 0.8236 - recall: 0.8855 - acc: 0.8620 - val_loss: 0.3840 - val_precision: 0.7585 - val_recall: 0.8900 - val_acc: 0.8212

Epoch 52/100

83/83 [==========] - 8s 92ms/step - loss: 0.2953 - precision: 0.8310 - recall: 0.8914 - acc: 0.8684 - val_loss: 0.3348 - val_precision: 0.7619 - val_recall: 0.9067 - val_acc: 0.8288

Epoch 53/100

83/83 [==========] - 7s 85ms/step - loss: 0.2921 - precision: 0.8318 - recall: 0.8881 - acc: 0.8677 - val_loss: 0.3256 - val_precision: 0.7591 - val_recall: 0.9033 - val_acc: 0.8258

Epoch 54/100

83/83 [==========] - 7s 82ms/step - loss: 0.2845 - precision: 0.8246 - recall: 0.8956 - acc: 0.8661 - val_loss: 0.3371 - val_precision: 0.7785 - val_recall: 0.8433 - val_acc: 0.8197

Epoch 55/100

83/83 [==========] - 8s 91ms/step - loss: 0.2839 - precision: 0.8249 - recall: 0.9014 - acc: 0.8684 - val_loss: 0.3358 - val_precision: 0.7892 - val_recall: 0.8733 - val_acc: 0.8364

Epoch 56/100

83/83 [==========] - 7s 84ms/step - loss: 0.2824 - precision: 0.8280 - recall: 0.8931 - acc: 0.8673 - val_loss: 0.3701 - val_precision: 0.7801 - val_recall: 0.8633 - val_acc: 0.8273

Epoch 57/100

83/83 [==========] - 8s 92ms/step - loss: 0.2954 - precision: 0.8271 - recall: 0.8872 - acc: 0.8646 - val_loss: 0.4198 - val_precision: 0.7202 - val_recall: 0.9267 - val_acc: 0.8030

Epoch 58/100

83/83 [==========] - 8s 92ms/step - loss: 0.2886 - precision: 0.8228 - recall: 0.8997 - acc: 0.8665 - val_loss: 0.3498 - val_precision: 0.7726 - val_recall: 0.8833 - val_acc: 0.8288

Epoch 59/100

83/83 [==========] - 7s 85ms/step - loss: 0.2916 - precision: 0.8306 - recall: 0.8805 - acc: 0.8642 - val_loss: 0.3722 - val_precision: 0.7373 - val_recall: 0.9167 - val_acc: 0.8136

Epoch 60/100

83/83 [==========] - 8s 85ms/step - loss: 0.3121 - precision: 0.8192 - recall: 0.8931 - acc: 0.8620 - val_loss: 0.3315 - val_precision: 0.7598 - val_recall: 0.9067 - val_acc: 0.8273

Epoch 61/100

83/83 [==========] - 7s 83ms/step - loss: 0.2821 - precision: 0.8250 - recall: 0.8939 - acc: 0.8658 - val_loss: 0.3392 - val_precision: 0.7655 - val_recall: 0.9033 - val_acc: 0.8303

Epoch 62/100

83/83 [==========] - 8s 89ms/step - loss: 0.2871 - precision: 0.8236 - recall: 0.8931 - acc: 0.8646 - val_loss: 0.3381 - val_precision: 0.7872 - val_recall: 0.8633 - val_acc: 0.8318

Epoch 63/100

83/83 [==========] - 7s 85ms/step - loss: 0.2799 - precision: 0.8354 - recall: 0.8822 - acc: 0.8677 - val_loss: 0.3318 - val_precision: 0.8000 - val_recall: 0.8667 - val_acc: 0.8409

Epoch 64/100

83/83 [==========] - 7s 82ms/step - loss: 0.2650 - precision: 0.8444 - recall: 0.9064 - acc: 0.8817 - val_loss: 0.3451 - val_precision: 0.8165 - val_recall: 0.8600 - val_acc: 0.8485

Epoch 65/100

83/83 [==========] - 7s 84ms/step - loss: 0.2881 - precision: 0.8234 - recall: 0.9039 - acc: 0.8684 - val_loss: 0.3327 - val_precision: 0.7867 - val_recall: 0.9100 - val_acc: 0.8470

Epoch 66/100

83/83 [==========] - 7s 83ms/step - loss: 0.2790 - precision: 0.8353 - recall: 0.8981 - acc: 0.8733 - val_loss: 0.3208 - val_precision: 0.7958 - val_recall: 0.8833 - val_acc: 0.8439

Epoch 67/100

83/83 [==========] - 7s 83ms/step - loss: 0.2758 - precision: 0.8333 - recall: 0.8897 - acc: 0.8692 - val_loss: 0.3159 - val_precision: 0.8232 - val_recall: 0.8533 - val_acc: 0.8500

Epoch 68/100

83/83 [==========] - 8s 86ms/step - loss: 0.2719 - precision: 0.8336 - recall: 0.8956 - acc: 0.8714 - val_loss: 0.3271 - val_precision: 0.8129 - val_recall: 0.8833 - val_acc: 0.8545

Epoch 69/100

83/83 [==========] - 8s 92ms/step - loss: 0.2737 - precision: 0.8414 - recall: 0.8956 - acc: 0.8760 - val_loss: 0.3284 - val_precision: 0.8012 - val_recall: 0.8733 - val_acc: 0.8439

Epoch 70/100

83/83 [==========] - 7s 84ms/step - loss: 0.2772 - precision: 0.8400 - recall: 0.8906 - acc: 0.8733 - val_loss: 0.3200 - val_precision: 0.7839 - val_recall: 0.9067 - val_acc: 0.8439

Epoch 71/100

83/83 [==========] - 7s 84ms/step - loss: 0.2778 - precision: 0.8285 - recall: 0.9039 - acc: 0.8714 - val_loss: 0.3922 - val_precision: 0.7190 - val_recall: 0.9467 - val_acc: 0.8076

Epoch 72/100

83/83 [==========] - 8s 90ms/step - loss: 0.2768 - precision: 0.8267 - recall: 0.8964 - acc: 0.8677 - val_loss: 0.3264 - val_precision: 0.8247 - val_recall: 0.8467 - val_acc: 0.8485

Epoch 73/100

83/83 [==========] - 7s 86ms/step - loss: 0.2657 - precision: 0.8534 - recall: 0.8947 - acc: 0.8824 - val_loss: 0.3368 - val_precision: 0.8268 - val_recall: 0.8433 - val_acc: 0.8485

Epoch 74/100

83/83 [==========] - 7s 85ms/step - loss: 0.2664 - precision: 0.8458 - recall: 0.8981 - acc: 0.8794 - val_loss: 0.3329 - val_precision: 0.7953 - val_recall: 0.8933 - val_acc: 0.8470

Epoch 75/100

83/83 [==========] - 7s 84ms/step - loss: 0.2650 - precision: 0.8397 - recall: 0.8972 - acc: 0.8756 - val_loss: 0.3304 - val_precision: 0.8031 - val_recall: 0.8700 - val_acc: 0.8439

Epoch 76/100

83/83 [==========] - 8s 96ms/step - loss: 0.2795 - precision: 0.8358 - recall: 0.8972 - acc: 0.8733 - val_loss: 0.3229 - val_precision: 0.7903 - val_recall: 0.8667 - val_acc: 0.8348

Epoch 77/100

83/83 [==========] - 8s 85ms/step - loss: 0.2622 - precision: 0.8462 - recall: 0.8872 - acc: 0.8756 - val_loss: 0.3412 - val_precision: 0.7727 - val_recall: 0.9067 - val_acc: 0.8364

Epoch 78/100

83/83 [==========] - 7s 85ms/step - loss: 0.2505 - precision: 0.8580 - recall: 0.9140 - acc: 0.8923 - val_loss: 0.3518 - val_precision: 0.8092 - val_recall: 0.8767 - val_acc: 0.8500

Epoch 79/100

83/83 [==========] - 7s 84ms/step - loss: 0.2478 - precision: 0.8529 - recall: 0.8964 - acc: 0.8828 - val_loss: 0.3368 - val_precision: 0.7749 - val_recall: 0.8833 - val_acc: 0.8303

Epoch 80/100

83/83 [==========] - 8s 89ms/step - loss: 0.2555 - precision: 0.8467 - recall: 0.9181 - acc: 0.8874 - val_loss: 0.3762 - val_precision: 0.8273 - val_recall: 0.7667 - val_acc: 0.8212

Epoch 81/100

83/83 [==========] - 7s 83ms/step - loss: 0.2700 - precision: 0.8524 - recall: 0.8881 - acc: 0.8794 - val_loss: 0.3464 - val_precision: 0.7855 - val_recall: 0.9033 - val_acc: 0.8439

Epoch 82/100

83/83 [==========] - 7s 82ms/step - loss: 0.2513 - precision: 0.8574 - recall: 0.9089 - acc: 0.8900 - val_loss: 0.3435 - val_precision: 0.7874 - val_recall: 0.8767 - val_acc: 0.8364

Epoch 83/100

83/83 [==========] - 7s 85ms/step - loss: 0.2499 - precision: 0.8565 - recall: 0.9073 - acc: 0.8889 - val_loss: 0.3441 - val_precision: 0.7874 - val_recall: 0.8767 - val_acc: 0.8364

Epoch 84/100

83/83 [==========] - 7s 84ms/step - loss: 0.2671 - precision: 0.8565 - recall: 0.8972 - acc: 0.8851 - val_loss: 0.3381 - val_precision: 0.7917 - val_recall: 0.8867 - val_acc: 0.8424

Epoch 85/100

83/83 [==========] - 7s 85ms/step - loss: 0.2598 - precision: 0.8521 - recall: 0.8997 - acc: 0.8836 - val_loss: 0.3182 - val_precision: 0.7994 - val_recall: 0.8767 - val_acc: 0.8439

Epoch 86/100

83/83 [==========] - 7s 84ms/step - loss: 0.2409 - precision: 0.8610 - recall: 0.9056 - acc: 0.8908 - val_loss: 0.3611 - val_precision: 0.7842 - val_recall: 0.8600 - val_acc: 0.8288

Epoch 87/100

83/83 [==========] - 8s 90ms/step - loss: 0.2680 - precision: 0.8445 - recall: 0.9073 - acc: 0.8821 - val_loss: 0.3430 - val_precision: 0.7735 - val_recall: 0.8767 - val_acc: 0.8273

Epoch 88/100

83/83 [==========] - 8s 84ms/step - loss: 0.2450 - precision: 0.8635 - recall: 0.9089 - acc: 0.8934 - val_loss: 0.3588 - val_precision: 0.7723 - val_recall: 0.8933 - val_acc: 0.8318

Epoch 89/100

83/83 [==========] - 7s 85ms/step - loss: 0.2467 - precision: 0.8648 - recall: 0.9031 - acc: 0.8919 - val_loss: 0.3371 - val_precision: 0.7844 - val_recall: 0.8733 - val_acc: 0.8333

Epoch 90/100

83/83 [==========] - 8s 92ms/step - loss: 0.2568 - precision: 0.8573 - recall: 0.8981 - acc: 0.8859 - val_loss: 0.3493 - val_precision: 0.7975 - val_recall: 0.8400 - val_acc: 0.8303

Epoch 91/100

83/83 [==========] - 7s 83ms/step - loss: 0.2646 - precision: 0.8451 - recall: 0.9023 - acc: 0.8805 - val_loss: 0.3541 - val_precision: 0.7903 - val_recall: 0.8667 - val_acc: 0.8348

Epoch 92/100

83/83 [==========] - 7s 84ms/step - loss: 0.2461 - precision: 0.8506 - recall: 0.9131 - acc: 0.8878 - val_loss: 0.3579 - val_precision: 0.7901 - val_recall: 0.8533 - val_acc: 0.8303

Epoch 93/100

83/83 [==========] - 8s 86ms/step - loss: 0.2549 - precision: 0.8557 - recall: 0.8964 - acc: 0.8843 - val_loss: 0.3299 - val_precision: 0.7988 - val_recall: 0.8600 - val_acc: 0.8379

Epoch 94/100

83/83 [==========] - 8s 92ms/step - loss: 0.2406 - precision: 0.8713 - recall: 0.9106 - acc: 0.8984 - val_loss: 0.3502 - val_precision: 0.7847 - val_recall: 0.8867 - val_acc: 0.8379

Epoch 95/100

83/83 [==========] - 9s 98ms/step - loss: 0.2316 - precision: 0.8687 - recall: 0.9123 - acc: 0.8976 - val_loss: 0.3548 - val_precision: 0.8076 - val_recall: 0.8533 - val_acc: 0.8409

Epoch 96/100

83/83 [==========] - 7s 82ms/step - loss: 0.2589 - precision: 0.8562 - recall: 0.9056 - acc: 0.8881 - val_loss: 0.3569 - val_precision: 0.7920 - val_recall: 0.8633 - val_acc: 0.8348

Epoch 97/100

83/83 [==========] - 8s 86ms/step - loss: 0.2477 - precision: 0.8536 - recall: 0.9106 - acc: 0.8885 - val_loss: 0.3408 - val_precision: 0.8158 - val_recall: 0.8267 - val_acc: 0.8364

Epoch 98/100

83/83 [==========] - 8s 89ms/step - loss: 0.2487 - precision: 0.8546 - recall: 0.8989 - acc: 0.8847 - val_loss: 0.3444 - val_precision: 0.7938 - val_recall: 0.8600 - val_acc: 0.8348

Epoch 99/100

83/83 [==========] - 7s 84ms/step - loss: 0.2302 - precision: 0.8641 - recall: 0.9081 - acc: 0.8934 - val_loss: 0.3571 - val_precision: 0.7959 - val_recall: 0.8967 - val_acc: 0.8485

Epoch 100/100

83/83 [==========] - 7s 84ms/step - loss: 0.2452 - precision: 0.8575 - recall: 0.9198 - acc: 0.8942 - val_loss: 0.3357 - val_precision: 0.8316 - val_recall: 0.8233 - val_acc: 0.8439

Results:

This code evaluates the performance of the trained model on the test dataset using three metrics: binary accuracy, recall, and precision.

First, BinaryAccuracy(), Recall(), and Precision() are initialized as objects to measure these metrics.

Then, for each batch in the test dataset, the model predicts the output probabilities for the input images using model.predict(X). These predicted probabilities yhat are then compared to the actual labels y to compute the metrics. ** bin_acc.update_state(y, yhat) updates the binary accuracy metric with the actual and predicted labels for the current batch. Similarly, recall.update_state(y, yhat) and precision.update_state(y, yhat) update the recall and precision metrics, respectively.

Finally, the results of the metrics are printed using bin_acc.result().numpy(), recall.result().numpy(), and precision.result().numpy(). These methods return the current value of the metric, which is stored as a state in the corresponding metric object.

In [17]:

```
bin_acc = BinaryAccuracy()
recall = Recall()
precision = Precision()

for batch in test_data.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X)
    bin_acc.update_state(y, yhat)
    recall.update_state(y, yhat)
    precision.update_state(y, yhat)

print("Accuracy:", round(bin_acc.result().numpy(),2)," % ", "\nRecall:", recall.result().numpy(),
"\nPrecision:", round(precision.result().numpy(),2)," 
```

- Accuracy: 0.84 %
- Recall: 0.8233333
- Precision: 0.83 %
- Saving Model

This code saves the trained model as an h5 file with the name "skin_cancer.h5". This file can be used later for inference or to continue training the model. Saving the model allows us to reuse it without having to retrain it from scratch, which can be time-consuming and computationally expensive. The saved model file can be loaded later using the load_model() function provided by the Keras API.

In [18]:

Testing Model

```
model.save("skin_cancer.h5")
```

This code creates a batch of data from the test_data dataset by using the as_numpy_iterator() method to obtain an iterator over the dataset and then calling the next() function to get the first batch of data.

By default, as_numpy_iterator() returns batches of data as tuples of numpy arrays (x, y), where x is the batch of input images, and y is the batch of corresponding labels. Therefore, after executing this code, batch will be a tuple containing a batch of input images and their corresponding labels.

In [19]:

```
batch = test_data.as_numpy_iterator().next()
```

batch[0] contains a batch of images, and batch[0][15] accesses the 16th image in the batch (Python indexing starts from 0). This code returns the pixel values of the 16th image in the batch as a NumPy array.

In [20]:

```
batch[0][15]
```

Out[20]:

```
array([[[0.8901961 , 0.54509807, 0.6       ],
        [0.8901961 , 0.5514706 , 0.60318625],
        [0.8928922 , 0.5502451 , 0.5985294 ],
        ...,
        [0.904902  , 0.61960787, 0.6669118 ],
        [0.89705884, 0.62107843, 0.6610294 ],
        [0.9098039 , 0.627451  , 0.67058825]],

       [[0.8933824 , 0.5387255 , 0.60318625],
        [0.8907935 , 0.54509807, 0.60637254],
        [0.888511  , 0.54825366, 0.59733456],
        ...,
        [0.9076899 , 0.6208027 , 0.6657169 ],
        [0.9106005 , 0.61848956, 0.66361827],
        [0.9129902 , 0.627451  , 0.67058825]],

       [[0.8860294 , 0.5372549 , 0.6012255 ],
        [0.88065255, 0.5326746 , 0.60003066],
        [0.88227636, 0.53246015, 0.59453124],
        ...,
        [0.9039522 , 0.6202359 , 0.6598805 ],
        [0.90563726, 0.6136795 , 0.66320467],
        [0.90563726, 0.6166667 , 0.66519606]],

       ...,

       [[0.82892156, 0.5487745 , 0.5139706 ],
        [0.8237439 , 0.5416054 , 0.50978863],
        [0.81665134, 0.5407935 , 0.49938726],
        ...,
        [0.8244945 , 0.57909006, 0.5576593 ],
        [0.8205576 , 0.5859988 , 0.5550858 ],
        [0.8129902 , 0.58161765, 0.55269605]],

       [[0.82058823, 0.5539216 , 0.50441176],
        [0.814614  , 0.5415748 , 0.49923408],
        [0.821492  , 0.5383885 , 0.5003983 ],
        [0.8244945 , 0.57909006, 0.5576593 ],
        [0.8205576 , 0.5859988 , 0.5550858 ],
        [0.8129902 , 0.58161765, 0.55269605]],

       [[0.82058823, 0.5539216 , 0.50441176],
        [0.814614  , 0.5415748 , 0.49923408],
        [0.821492  , 0.5383885 , 0.5003983 ],
        ...,
        [0.8106618 , 0.5718903 , 0.54952514],
        [0.81790745, 0.5833946 , 0.55574447],
        [0.82009804, 0.5857843 , 0.5615196 ]],

       [[0.80784315, 0.5411765 , 0.49803922],
        [0.81740195, 0.5443627 , 0.49803922],
        [0.81691176, 0.53161764, 0.4872549 ],
        ...,
        [0.8186275 , 0.5764706 , 0.55808824],
        [0.8137255 , 0.57941175, 0.5615196 ],
        [0.8392157 , 0.5921569 , 0.5647059 ]]],
dtype=float32)
```

batch[1] contains the corresponding labels for the batch of images, and batch[1][15] accesses the label of the 16th image in the batch (Python indexing starts from 0). This code returns the label of the 16th image in the batch.

In [21]

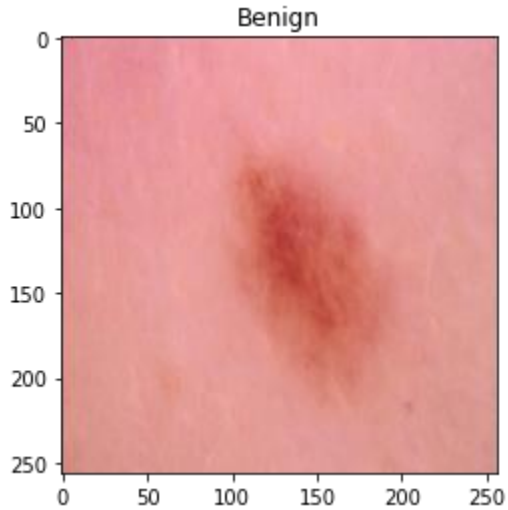
```
batch[1][15]
```

Out[21]:

```
This code displays an image and its label (whether it is malignant or benign).

- img is a variable that stores the 16th image in the batch of test data.

- label is a variable that stores the label (1 for malignant and 0 for benign) of the 16th image in the
batch.

- plt.imshow() displays the image.

- plt.title() sets the title of the plot based on the label. If the label is 1, it sets the title as
"Malignant" and if the label is 0, it sets the title as "Benign".

- plt.show() displays the plot.
```

In [22]:

```
img, label = batch[0][15], batch[1][15]
plt.imshow(img)
if label==1:
    plt.title("Malignant")
else:
    plt.title("Benign")
plt.show()
```



In this code, model.predict() is used to obtain the model's prediction for the input image img. However, img has a shape of (256, 256, 3), while the model expects a batch of images with shape (batch_size, 256, 256, 3). Therefore, np.expand_dims(img, 0) is used to add an extra dimension to the input image, so that its shape becomes (1, 256, 256, 3), which can be fed into the model. The resulting y_hat variable contains the predicted probability that the input image is malignant.

In [23]:

```
y_hat = model.predict(np.expand_dims(img,
0))
```

y_hat contains the model's prediction for the given input image img. The model.predict method is used to get the model's output (i.e., prediction) for a given input. Here, the np.expand_dims function is used to expand the dimensions of the input image from (256, 256, 3) to (1, 256, 256, 3), which is the shape expected by the model.

Since we are predicting for only one input image, we get a single output, which is a floating-point number between 0 and 1. This represents the model's prediction for the probability that the input image belongs to the positive class (i.e., malignant skin cancer).

In [24]:

```
y_hat
```

Out[24]:

```
array([[0.00128437]],
dtype=float32)
```

This code checks the predicted probability of the input image being malignant or benign. If the predicted probability is less than 0.5, the code prints "Benign", otherwise it prints "Malignant".

The y_hat variable contains the predicted output probabilities of the image being malignant. The if statement checks if the predicted probability is less than 0.5. If it is, the code prints "Benign" because the model is binary classification and assumes that the probability of the input image belonging to the first class (benign) is greater than the probability of belonging to the second class (malignant) if it is less than 0.5. If the predicted probability is greater than or equal to 0.5, the code prints "Malignant" because the model assumes that the probability of the input image belonging to the second class (malignant) is greater than or equal to the probability of belonging to the first class (benign) if it is greater than or equal to 0.5.

In [25]:

```
if y_hat < 0.5:
    print("Benign")
else:
 print("Malignant")
```

6.7 java Code

```
<?xmi version="1.0" encoding="utf-8"?>
<manifest xmIns: android="http: // schemas. android.com/apk/res/android"
xmins: tools="http://schemas .android. com/tools">
<uses-permission android: name="android .permission.INTERNET" / >
<uses-permission android: name="android •permission.READ_EXTERNAL_STORAGE" />
<uses-permission android: name-"android.permission .WRITE EXTERNAL STORAGE" />
<application
android: allowBackup="true"
android: dataExtractionRules-"@mI/data_extraction_rules"
android: fullBackupContent="@mI/backup_rules"
android: icon="@mipmap/ic_launcher"
android:label="skin cancer'
android: roundIcon-"@mipmap/ic_launcher_round"
android: supportsRti="true"
android:theme="@style/Theme .SkinCancer"
tools:targetApi="31"›
```

```java
package co.boldbrand. skincancer;
import ...
public class MainActivity extends AppompatActivity implements View. OnClickListener {
TextView logoTextView,registerMeTextView;
@override
protected void onCreate (Bundle savedInstanceState) {
super.onCreate (savedInstanceState);
setContentView (R.layout .activity_main);
logoTextView=findViewById(R.id.Logo);
registerMeTextView=findViewById(R.id.register_me);
registerMeTextView.setOnClickListener(this);

double angleInRadians = Math. toRadians (45) ;
double length = logoTextView.getPaint () •getTextSize();

double endX = Math. sin (angleInRadians) * length;
double endY = Math. cos (angleInRadians) * length;}

Shader textShader = new LinearGradient ((float) 0, (float) 0, (float) endX, (float)
newint[]{getColor (R.color.start_color), getColor (R.color.end_color)},
new float []{0, 1}, Shader. TileMode .CLAMP);
logoTextView.getPaint().setShader (textShader) ;
}
@Override
public void onClick(View v) {
int id=v.getId();
if(id==R.id.register me){
startActivity (new Intent ( packageContext: this, SignUpActivity .class));
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://scne
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFF"
    tools:context=".SignUpActivity">
    <TextView
        android:id="@+id/logo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="24dp"
        android:text="SC"
        android:fontFamily="@font/palatte_mosaic"
        android:textSize="32sp"
```

```
import java.1o.serializable;
public class SkinLesion implements Serializable, Comparable<SkinLesion> {
private String lesionType;
private String probability;

public void setLesionType(String lesionType) {
this.lesionType = lesionType;
}
public void setProbability(String probability) {
this.probability = probability;
}
public String getLesionType() {
return lesionType;
}
public String getProbability() {
return probability;
}
public SkinLesion() {
this.lesionType = lesionType;
this.probability = probability;
}
@Override
public int compareTo(SkinLesion s) §
return Float.valueof(this.getProbability ()) .compareTo(Float.valueof
(s.getProbability()));
}
```

```java
import android.content. Context;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.List;

public class SkinLesionAdapter extends ArrayAdapter<SkinLesion> {

public skinLesionAdapter(@NonNull Context context, List <skinLesion› skinLesions) {
super (context, 0, skinLesions);
}
@NonNull
@Override
public View getview(int position, @Nullable View convertview, @NonNull ViewGroup parent) {
View listItemView = convertView;
if (listItemView == null) {
listItemview = LayoutInflater.from(getContext ()) . inflate(
R.layout.activity_listview, parent, false);
}
SkinLesion skinLesion = getItem(position);
String lesion = skinLesion. getlesionType( );
TextView lesionTextView = (TextView) listItemView.findViewById(R.id.lesion);
lesionTextView.setText(lesion);
String probability = skinlesion.getProbability( );
probability = probability + "%";
TextView probabilityTextView = (TextView) listitemview.findViewById(R.id.probability);
probabilityTextView.setText(probability);
return listItemView;
   }
}
```

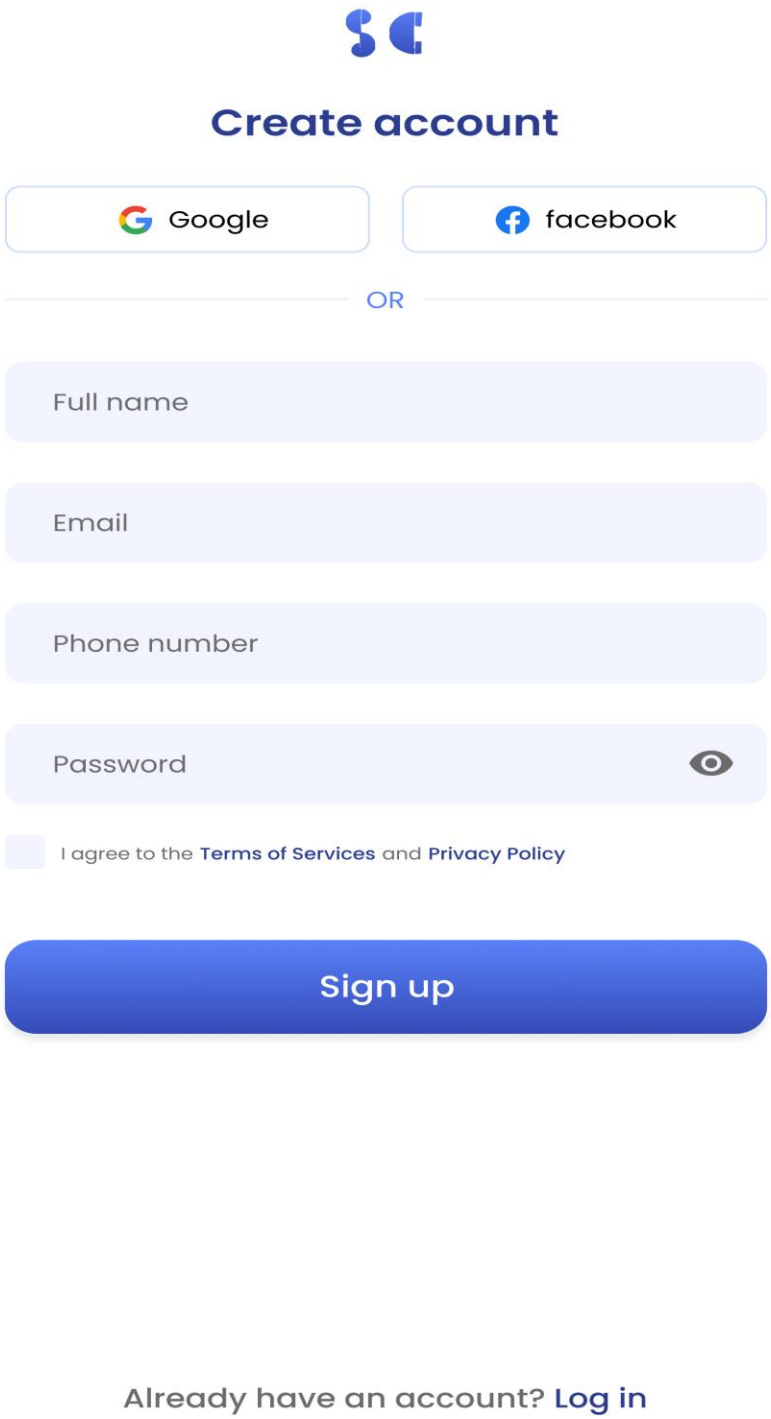
```
import android.view.MenuItem;
import android.view.ViewGroup;
import android.widget.ListView;
import android.widget. TextView;

import java.util.ArrayList;
import java.util.List;

public class ResultActivity extends AppCompatActivity {
private skinLesionAdapter mAdapter;
private TextView mEmptystateTextView;

@override
protected void onCreate Bundle savedInstancestate) {
super.onCreate (savedInstancestate);
setContentView(R.layout.activity result);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
setTitle("Detection Result");

Intent intent = getIntent ();
List‹SkinLesion> skinLesions = (ArrayList‹SkinLesion>) intent.getSerializableExtra("skinLesions");

ListView skinLesionsListView = (ListView) findViewById(R.id.list);
mEmptystateTextView = (TextView) findViewById(R.id.empty_view);

LayoutInflater inflater = getLayoutInflater () ;
ViewGroup header = (ViewGroup) inflater.inflate(R.layout.listview_header, skinlesionsListView, false);
skinLesionsListView.addHeaderView(header, null, false);
skinLesionsListView.setEmptyView(mEmptyStateTextView);
mAdapter = new skinLesionAdapter (this, new ArrayList<SkinLesion>());
skinLesionsListView.setAdapter(mAdapter);

mAdapter .addAll (skinLesions);
}
@override
public boolean onOptionsItemSelected(MenuItem item) {
onBackPressed0;
return true;
  }

}
```

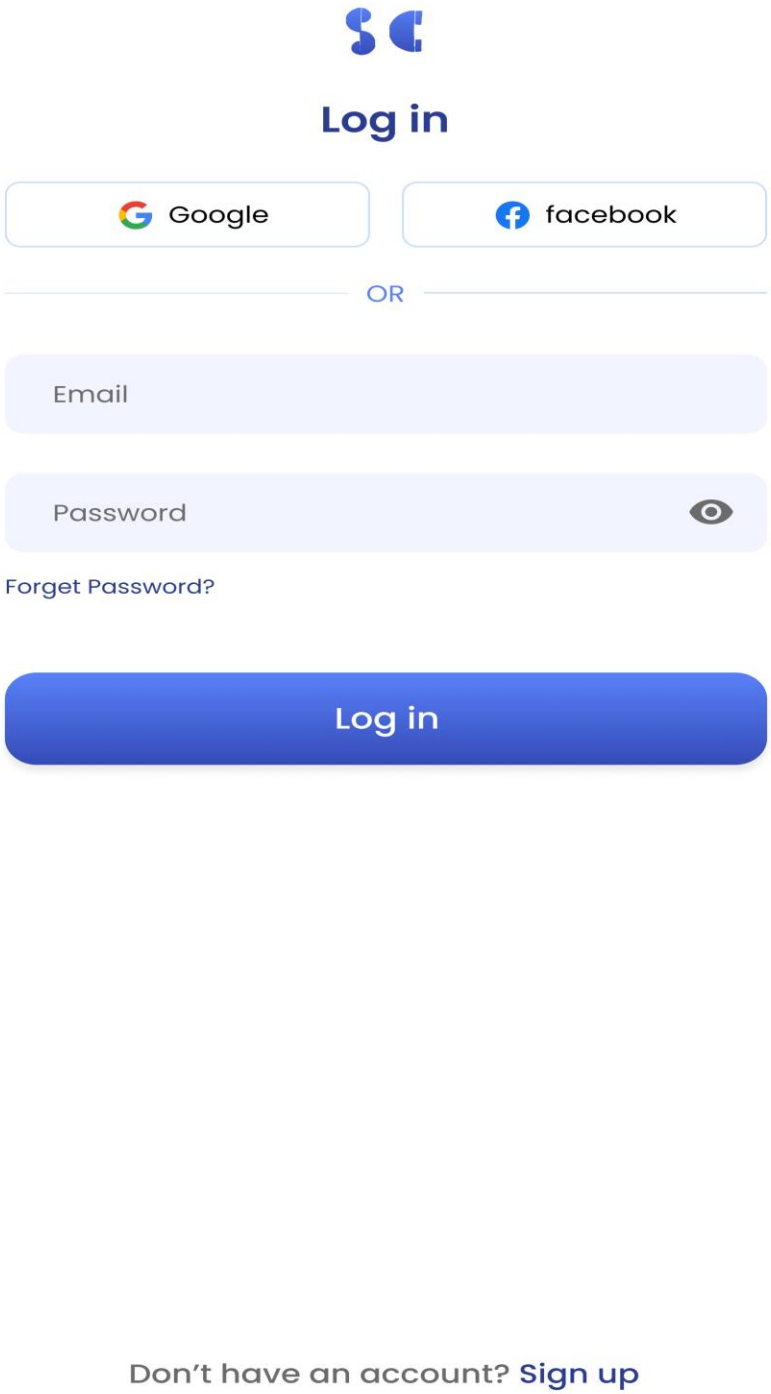
```
//Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
//setSupportActionBar(toolbar);

inputEmail = (EditText) findViewById(R.id.email);
inputPassword = (EditText) findviewById(R.id.password);
//progressBar = (ProgressBar) findViewById(R.id.progressBar);
signup = (Textview) findViewById(R.id.signup);
login = (TextView) findViewById(R.id.login);
//btnReset = (Button) findViewById(R.id.btn reset password);

Get Firebase auth instance
auth = FirebaseAuth.getInstance);

signup.setOnclickListener(new View.onclickListener() {
@Override
public void onClick(View v) {
startActivity (new Intent (LoginActivity.this, signupActivity.class));
}
});
/*btnReset.setOnclickListener(new View.onclickListener () {
@override
public void onclick(View v) {
startActivity(new Intent (LoginActivity.this, ResetPasswordActivity.class));
});*/
login.setOnClickListener (new View.OnClickListener ) {
@Override
public void onclick(View v) {
String email = inputEmail.getText().tostring();
final String password = inputPassword.getText () .tostring();
if (TextUtils.isEmpty(email)) {
Toast.makeText (getApplicationContext (), "Enter email address!", Toast. LENGTH_SHORT) . show();
return;
}
if (TextUtils.isEmpty(password)) {
Toast. makeText(getAnplicationConteyt()."Enter naccwordi" Toast IENGTH SHORT). show ();
```

```
Toast.makeText (getApplicationContext (),"Enter email address!", Toast.LENGTH_SHORT) . show();
return;
}
if (TextUtils.isEmpty(password)) {
Toast.makeText (getApplicationContext (), "Enter password!",Toast. LENGTH _SHORT). show();
return;
}
//progressBar.setvisibility(View.VISIBLE);
pd. setMessage("Signing in");
pd.setCancelable(false);
pd. show() ;

//authenticate user
auth.signInwithEmailAndPassword(email,password)
•addonCompleteListener(LoginActivity.this,new OnCompleteListener<AuthResult>() {
@Override
public void onComplete(@NonNull Task<AuthResult> task) {
// If sign in fails, display a message to the user. If sign in succeeds
// the auth state listener will be notified and logic to handle the
// signed in user can be handled in the listener.
//progressBar.setvisibility(View.GONE);
pd.dismiss ();
if (!task.isSuccessful()) {
// there was an error
if (password.length () < 6) {
inputPassword.setError(getString(R.string.minimum_password));
} else {
Toast.makeText (LoginActivity.this, getstring(R.string.auth_failed), Toast. LENGTH_LONG) . show();
} else {
Intent intent = new Intent (LoginActivity.this, NavigationActivity.class);
startActivity(intent);
finish();
            }
          }
      });
    }
  });
}
```

```
import android.app.ProgressDialog;
import android.content. Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.view.windowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget. Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class LoginActivity extends AppCompatActivity {

private EditText inputEmail, inputPassword;
private TextView login;
private TextView signup;
private FirebaseAuth auth;
private ProgressDialog pd;
private Button btnReset;

@Override
protected void onCreate(Bundle savedInstanceState) ‹
super. onCreate(savedInstancestate);

//Get Firebase auth instance

auth = FirebaseAuth.getInstance);
pd=new ProgressDialog (LoginActivity.this);
if (auth.getCurrentUser () != null) {
startActivity(new Intent (LoginActivity.this, NavigationActivity.class));
finish();
}

//set the view now
setcontentView(R.layout.activity_login);
retWindow(
setFlags(WindowManager.lavoutParams.FLAGFULISCREENWindowManager.lavoutParams.FlAG FULISCREEN)
```

6.8 output


Create account
Google
facebook
OR
Full name
Email
Phone number
Password
I agree to the Terms of Services and Privacy Policy
Sign up
Already have an account? Log in


Log in
Google
facebook
OR
Email
Password
Forget Password?
Log in
Don't have an account? Sign up

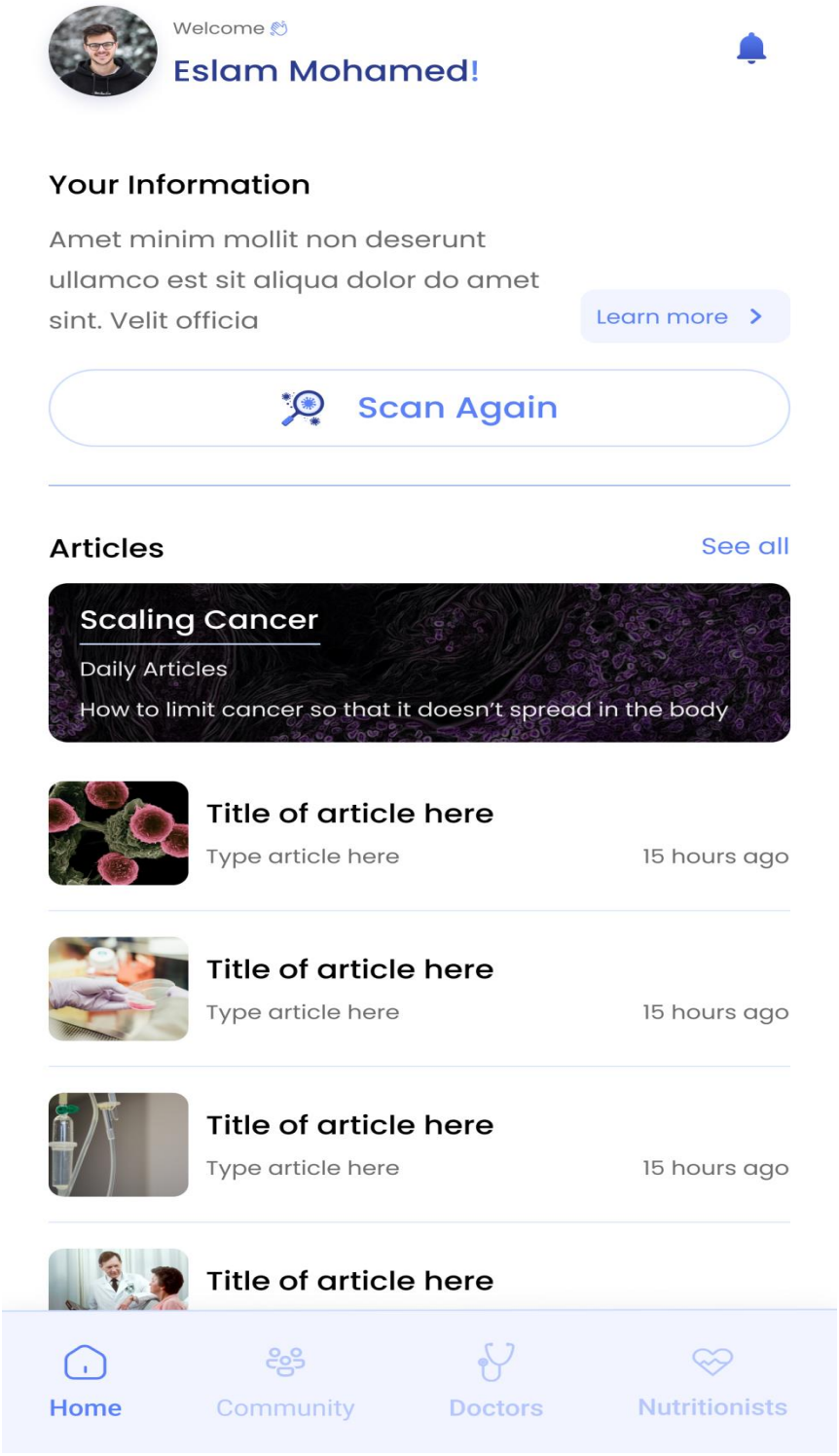
←

Forget password

Please enter the phone number you used when logging in to send code verification

Phone number

Send


Welcome
Eslam Mohamed!
Your Information
Amet minim mollit non deserunt ullamco est sit aliqua dolor do amet sint. Velit officia
Learn more
Scan Again
Articles
See all
Scaling Cancer
Daily Articles
How to limit cancer so that it doesn't spread in the body
Title of article here
Type article here
15 hours ago
Title of article here
Type article here
15 hours ago
Title of article here
Type article here
15 hours ago
Title of article here
Home
Community
Doctors
Nutritionists

←

Your information

Amet minim mollit non deserunt ullamco est sit aliqua dolor do amet sint. Velit officia

Age	32
Malignant/Benign	Malignant
Location	cairo
Date of enter	22/2/2023

History prescription

Dr. Hassan Mohamed
22/1/2023 >

Dr. Hassan Mohamed
22/1/2023 >

Dr. Hassan Mohamed
22/1/2023 >

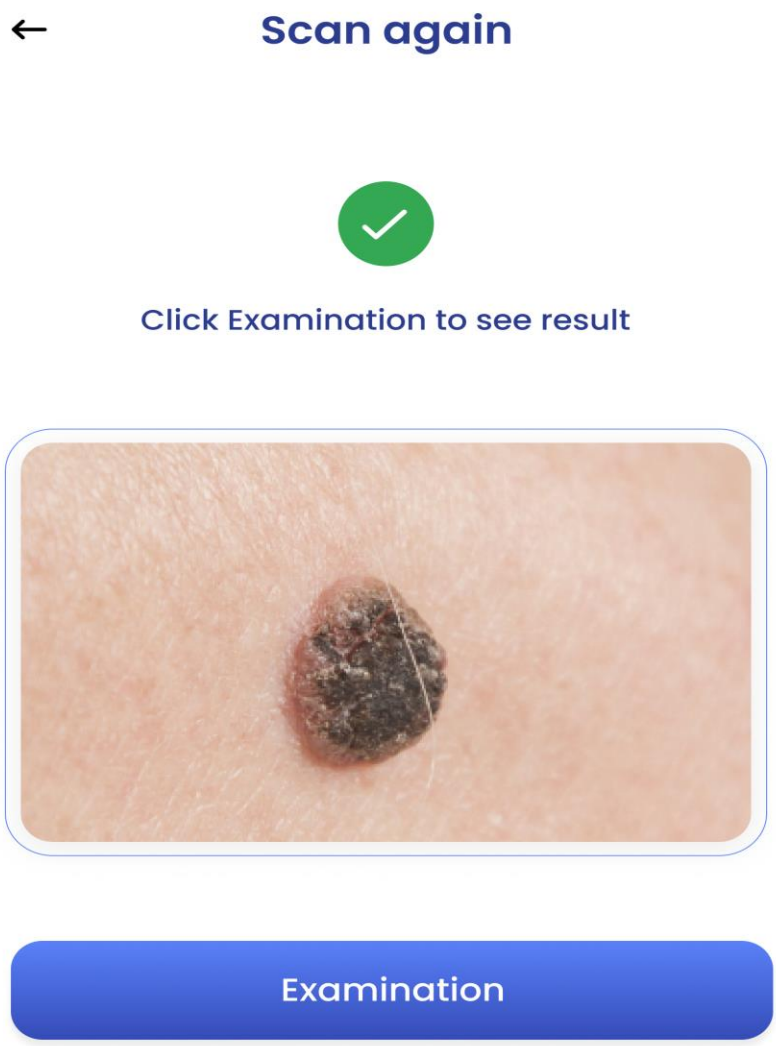
← Skin scan

Disease identification

Lets take photo of spot from your skin to can detect if you have cancer disease

Take photo for spot

Next


Scan again
Click Examination to see result
Examination

Find doctor

Online consultation

Dr. Kareem Ahmed

Specializes in skin cancer

4.5

Chat

Top doctors

Dr. Hassan Mohamed

250 EGP

Specializes in skin cancer

Cairo, Egypt | 8:00Am - 6:00Pm | 4.5

Nearby doctors

Dr. Ahmed Mohamed

350 EGP

Specializes in skin cancer

Cairo, Egypt | 8:00Am - 6:00Pm | 4.9

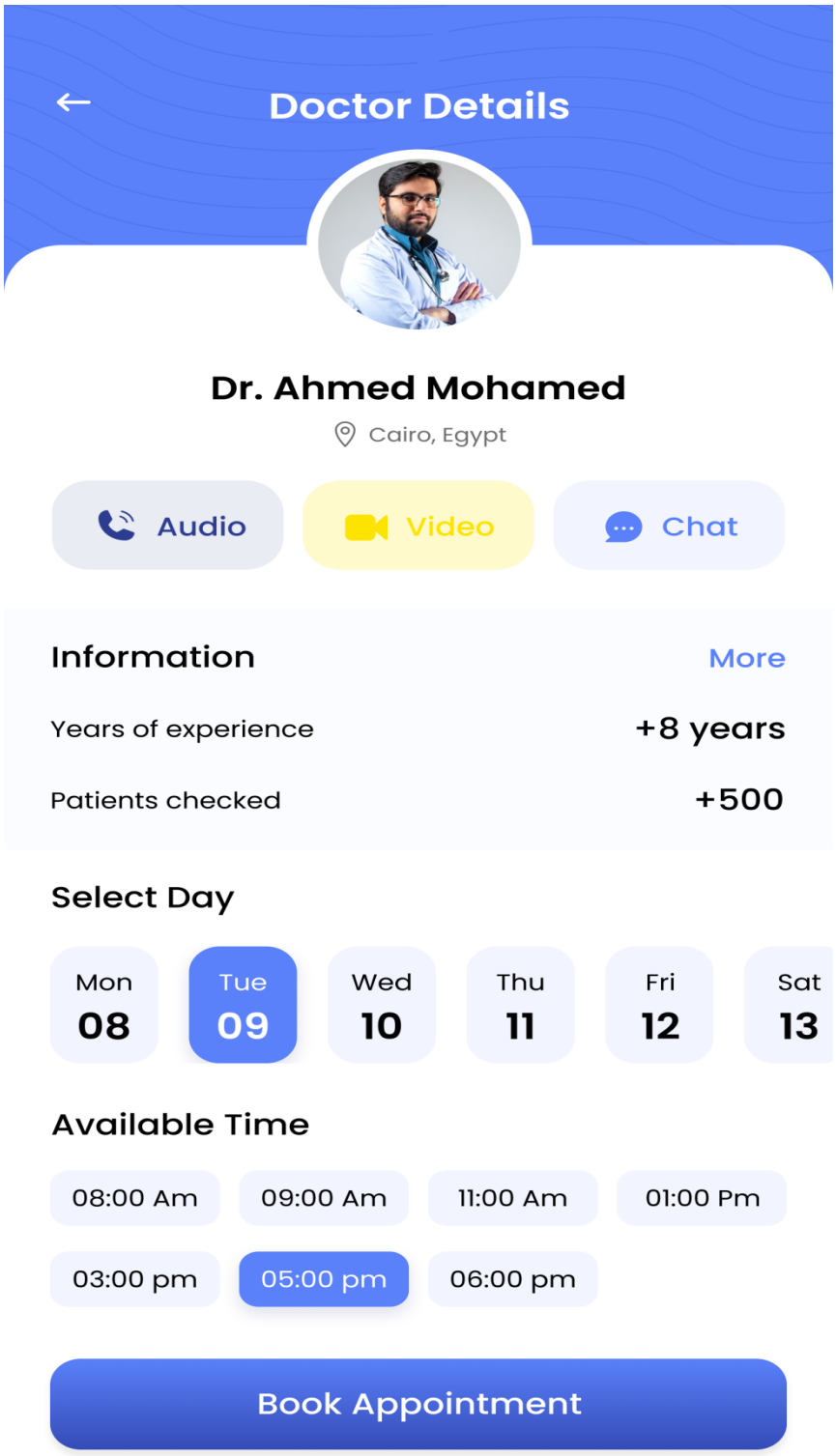
Dr. Eslam Mohamed

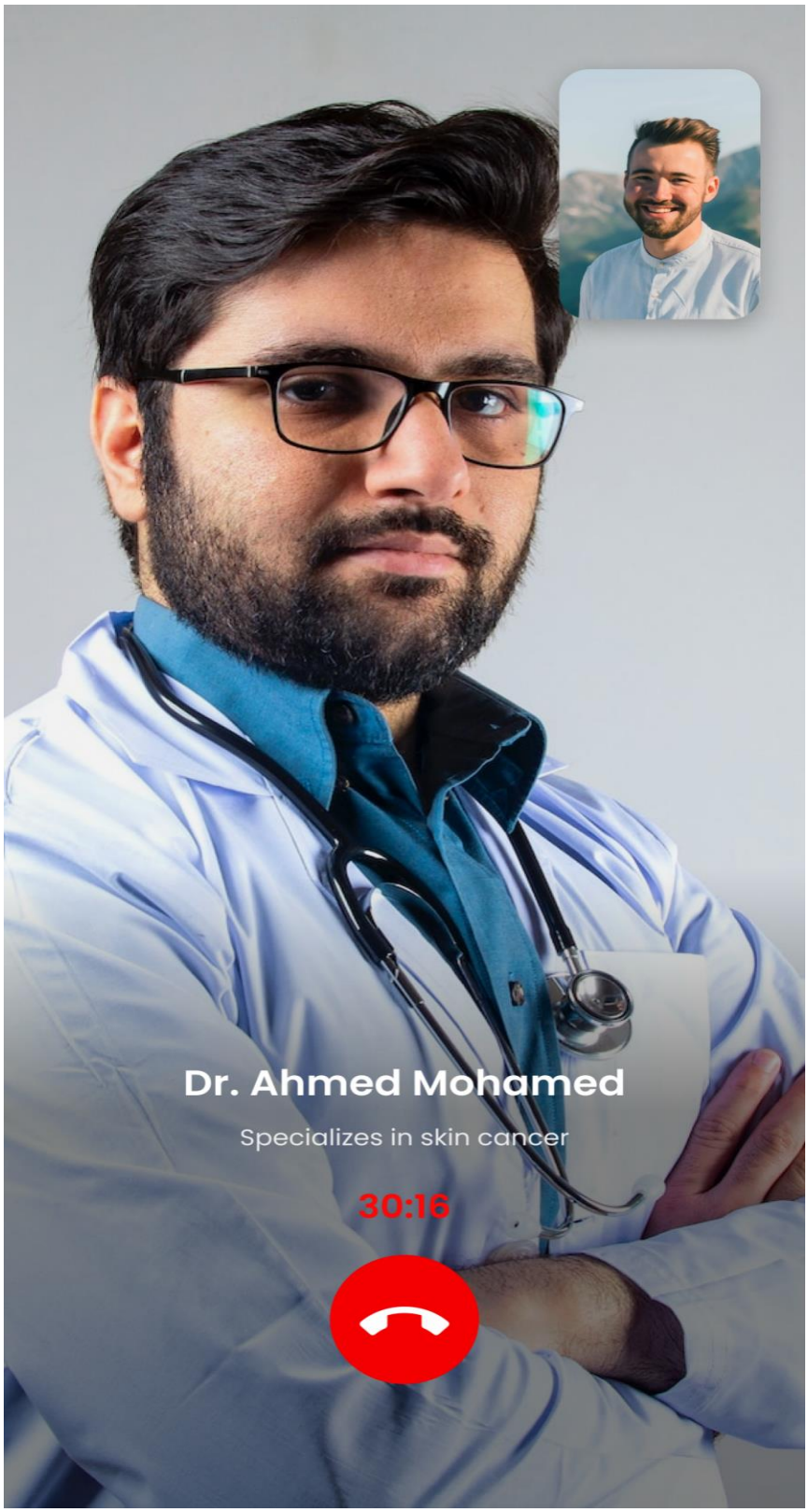
250 EGP

Specializes in skin cancer

Giza, Egypt | 8:00Am - 6:00Pm | 4.5

Home | Community | Doctors | Nutritionists


Doctor Details
Dr. Ahmed Mohamed
Cairo, Egypt
Audio
Video
Chat
Information
More
Years of experience
+8 years
Patients checked
+500
Select Day
Mon 08
Tue 09
Wed 10
Thu 11
Fri 12
Sat 13
Available Time
08:00 Am
09:00 Am
11:00 Am
01:00 Pm
03:00 pm
05:00 pm
06:00 pm
Book Appointment


Dr. Ahmed Mohamed
Specializes in skin cancer
30:16

Nutritionists

Skin cancer patients should aim to eat a balanced and varied diet that includes plenty of fruits, vegetables, whole grains, lean proteins, and healthy fats. Some specific foods and nutrients that may be particularly beneficial for skin cancer patients include:

Staying hydrated

Is important for overall health, and may also help keep the skin healthy and supple

Antioxidant

vitamin C, vitamin E, and beta-carotene, berries, citrus fruits, leafy greens, nuts, and seeds.

Omega-3 fatty acids

salmon, tuna, and mackerel, walnuts, flaxseeds, and chia seeds.

Vitamin D

fatty fish, egg yolks, and fortified foods such as milk or

Home Community Doctors Nutritionists

Please take care to drink the daily portion to protect your skin

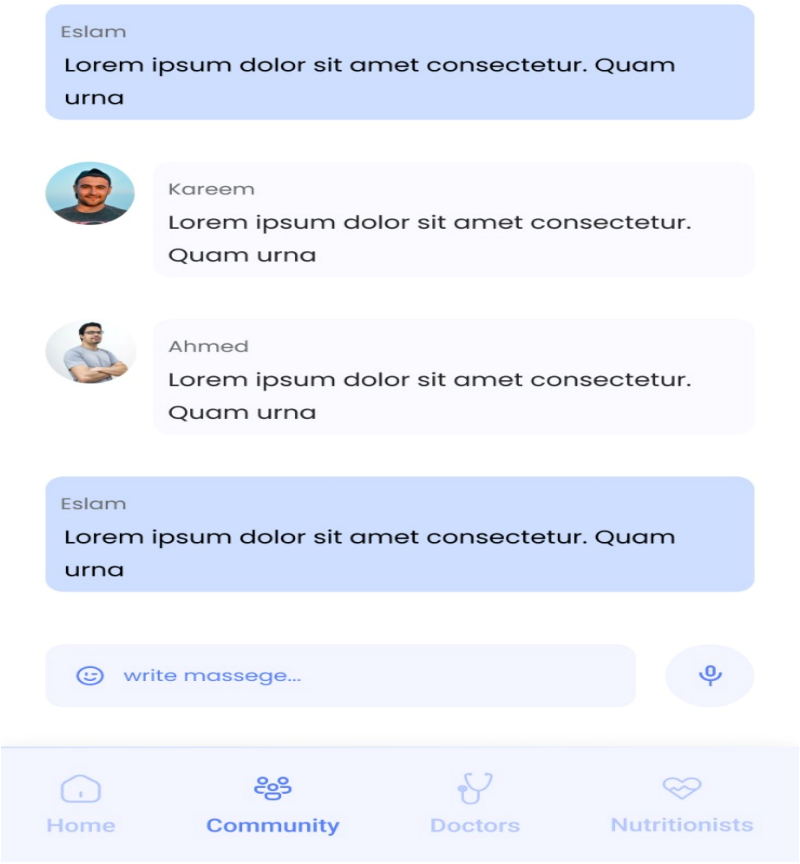
-4 glasses

The amount of water a skin cancer patient should drink each day can vary depending on factors such as their age, gender, weight, activity level, and overall health status. However, a general guideline is to aim for at least 8-10 glasses (64-80 ounces) of water per day.

It's important for skin cancer patients to stay hydrated to support their overall health and well-being, but they should also be mindful of any fluid restrictions they may have due to their medical condition or treatment. Additionally, they may need to adjust their fluid intake based on factors such as sweating, vomiting, or diarrhea.

It's always best for skin cancer patients to consult with their healthcare provider or a registered dietitian to determine the appropriate amount of water and other fluids they should be consuming each day.

Chat


Eslam
Lorem ipsum dolor sit amet consectetur. Quam urna
Kareem
Lorem ipsum dolor sit amet consectetur. Quam urna
Ahmed
Lorem ipsum dolor sit amet consectetur. Quam urna
Eslam
Lorem ipsum dolor sit amet consectetur. Quam urna
write massege...
Home
Community
Doctors
Nutritionists

Chapter 7

(Conclusion and Future Work)

7.1 Conclusion

7.2 Future Work

Chapter 7

7.1 Conclusion

It can be concluded that the skin scanning system was successful. This system consists of a learning machine model, a server, and a phone application. It is easy to use and there is no cost to use it. The patient is allowed to examine the spots on his skin, whether they are benign or malignant. It also allows the patient to communicate with doctors via chat. The patient can also book with the doctor through the application, and the doctor can also add articles about skin cancer and the method of preventing it, as well as adding useful food for cancer patients

They met as follows:

• Creation of a skin scan system to be used via smartphone

• Design and implementation of a skin cancer screening program at no final cost

• Easy to use application.

7.2 Future work

There are some recommendations for future works. some of them:

1. Examination of burns and their degrees can be added, and burn doctors can be added.

2. Skin care can be added, as well as articles about cosmetics and their harm to the skin.

Chapter 8

(Reference)

8.1. Reference

Chapter 8

8.1. Reference:

1. https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000
2. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.
3. Codella, N. C., Rotemberg, V., Tschandl, P., & Celebi, M. E. (2018). Deep learning, sparse coding, and SVM for melanoma recognition in dermoscopy images. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 611-619). Springer, Cham.
4. Guo, Y., Feng, Y., & Yin, Y. (2020). Skin cancer detection using image processing techniques: A review. Journal of Healthcare Engineering, 2020.
5. Tschandl, P., Codella, N., Akay, B. N., Argenziano, G., Braun, R. P., Cabo, H., ... & Halpern, A. (2018). Comparison of the accuracy of human readers versus machine-learning algorithms for pigmented skin lesion classification: an open, web-based, international, diagnostic study. The Lancet Oncology, 19(7), 9-78.
6. Wang, W., Yang, Y., Wang, W., & Gao, Y. (2019). Deep convolutional neural network-based computer-aided diagnosis of skin lesion using differential evolution algorithm. Journal of Medical Systems, 43(12), 373.
7. Sharma, A., Verma, A., & Gupta, S. K. (2021). Skin Cancer Detection Using Machine Learning Techniques: A Review. Journal of Medical Systems, 45(5), 1-13.
8. Siegel, R. L., Miller, K. D., & Jemal, A. (2021). Cancer statistics, 2021. CA: A Cancer Journal for Clinicians, 71(1), 7-33.
9. American Cancer Society. (2021). Skin Cancer Facts & Statistics. Retrieved from https://www.cancer.org/cancer/skin-cancer/about/key-statistics.html.