A blue circle with white text

Description automatically generated

**Early Diagnosis and Detection of**

**Knee Osteoarthritis and determining its severity degree based on deep neural network**

(**Knee Osteoarthritis**)

**Team members:**

**1- Mario Ashraf yaqoub 20-00355 5- Sara Bakhet Falltous 20-00006**

**2- Thomas Hany Nagy 20-01193 6- Merna Samir Fouad 20-00200**

**3- Youssef Mohsen thabet 20-00440 7- Roudia Abdelrhman 20-01585**

**4- Yassa akram Roshdy 20-01457**

**Under Supervision of:**

**DR. Khaled Wassif**

### Professor and Head of Computer Science Department

#### Faculty of Computers and Artificial Intelligence,

#### Cairo University

**T.A Zienab Mostafa**

Teacher Assistant in Computer and Information Technology Egyptian E-Learning University

# DEDICATION

*“ This project is dedicated to our beloved parents who have stood by us throughout our entire educational journey ”.*

# ACKNOLEDGMENTS

*Firstly, we would like to thank* Dr: Khaled wassif

*for his support, guidance, and advice throughout the year.*

*We also owe a debt of gratitude to* Eng: Zienab Mostafa

*for his dedication, keen interest, and efforts with us and technical support.*

*Lastly, we would like to thank our parents and families for the moral, spiritual, emotional, and financial support*

# ABSTRACT

Knee Osteoarthritis accounts for more than 80% cases of arthritis impacting life quality of individuals. Huge number of people Suffer form arthritis causes pain, mobility limitation, affecting independence and quality of life It is an irreversible disease that the only cure is the replacement of the knee, being important to diagnose it at early stages to prevent its progression. **Knee Osteoarthritis** aims to improve the detection of Knee Osteoarthritis at all stages based on Kellgren-Lawrence scale using machine learning models

The main idea of the project is to build an application consisting of two models and another feature that can classification and early detection of knee osteoarthritis, so that we get a clear and understandable result for these processes. It also helps user and doctor to easily detection of Knee osteoarthritis

At the end of this project, we will present an application that can be used by any user with simple and easy steps and get a result that is understandable to all users, ranging from novice user to doctors and researchers.

# Table of content:

**Contents**

DEDICATION 2

ACKNOLEDGMENTS 3

ABSTRACT 4

Table of content 5

Lists of figures 6

Chapter 1: Introduction 8

1.1. Introduction 9

1.2. Background 12

1.3. Deep learning 14

1.4. Preprocessing 16

1.4.1. Image preprocessing 13

1.5. Overview of Data Set 17

1.6. Proposed model 17

1.7. Software development life cycle 19

Chapter 2: Analysis and design 27

3.1. Functional and non functional 28

3.2. Use case actor 29

3.3. Fully dressed use case 33

3.4. Use case diagram 36

3.5. Sequence diagram 37

Chapter 4: GUI interface 41

# Lists of figures:

Figure1 anatomy of knee…………………………………………………………..… 10

Figure2 show joint space…………………………………………………………….. 12

Figure3 show neural network………………………………………………………… 15

Figure4 System Development Life Cycle ……………………….……………..……. 19

Figure5 Waterfall model…………………………………………………………..…. 21

Figure6 Agile model……………………………………………………………....…. 22

Figure7 V-Shaped model…………………………………………………………….. 22

Figure8 Spiral model………………………………………………………………….23

Figure9 Iterative model…………………………………………………………..…. 24 Figure10 Big Bang model………………………………………………………..…... 24

Figure11 parallel model…………………………………………………………...…. 25

Figure12 Use Case ………………………………………………………………..…. 29

Figure13 Use Case for (Registration) ………………...…………………………..…. 30

Figure14 Use Case for (Login) ……………….............…………………………..…. 30

Figure15 Use Case for (Determining KOA X-ray) ………………...……...……..…. 31

Figure16Use Case for (Choose doctor) ………………...………………………..…. 31

Figure17 Use Case for (Answer the Questionnaire) ……………………………..…. 32

Figure18Use Case for (Changing data for user) …………….…………………..…. 32

Figure19 Use Case for (Contact with a doctor) ………………...………………..…. 33

Figure20 Use Case for (Do Exercise) ………………...………………..…………… 33

Figure21 Use Case for (Follow up with Patient) ………………...………….……… 34

Figure22 Use Case for (Add Diagnosis) ………………...……………………….…34 Figure23Use Case for (Make a Report) ………………...………….……………… 35

Figure24 Use Case for (Use Case Diagram) ………………...……………………... 36

Figure25 System Sequences for (Registration) ………………......………………… 37

Figure26 System Sequences for (Login) ……………………........………………… 37

Figure27 System Sequences for (Upload Image) ………………......…………….… 37

Figure28 System Sequences for (Contact with Doctor) ………………......…………39

Figure29 System Sequences for (Make Exercise) ………………......…………….… 40

Figure30 Candidate Conceptual Classes ………………………......…………………58

Figure31 The first page for app that patient , doctor ………………......………….… 63

Figure32 sign up for patient for first time …………………………......………….… 63

Figure33 login page to the patient that already has an account……………..…….… 64

Figure34 the page about knee osteoarthritis……………………….………..…….… 64

Figure35The page to upload x-ray to classify…………………….….……..…….… 66

Figure36 The page to choose doctor……………………..………………….…….… 71

Figure37The page to make appointment…………………………………………….. 71

Figure38 The page to view time table………….……..………………….…….….… 72

Figure39 The page to confirm appointment………………….…..………….…….… 72

Figure40 The page successfully done.……………….……….…..………….…….… 73 Figure41 The page to chat with doctors.……………….…………………….…….… 74

# Chapter 1

## Introduction

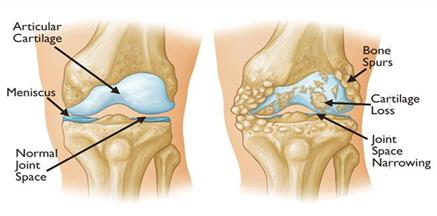
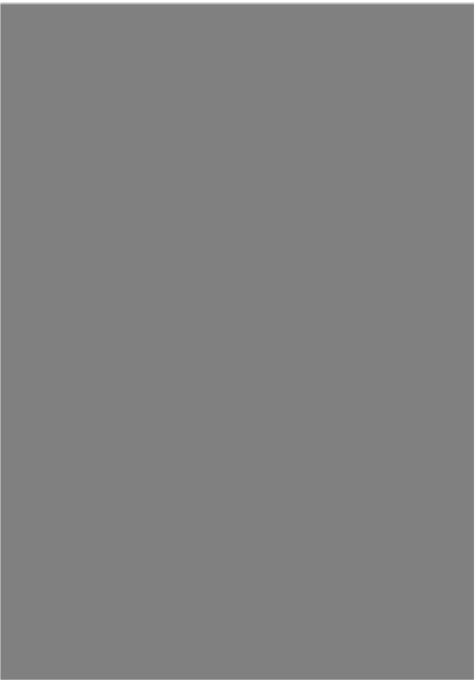
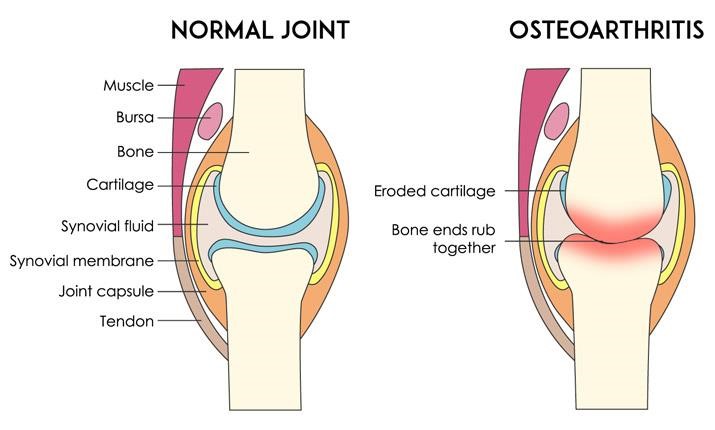
**CH 1: Introduction**

Knee osteoarthritis (OA), also known as degenerative joint disease, is typically the result of wear and tear and progressive loss of articular cartilage. It is most common in the elderly. Knee osteoarthritis can be divided into two types, primary and secondary. Primary osteoarthritis is articular degeneration without any apparent underlying reason. Secondary osteoarthritis is the consequence of either an abnormal concentration of force across the joint as with post-traumatic causes or abnormal articular cartilage, such as rheumatoid arthritis (RA).

Osteoarthritis is typically a progressive disease that may eventually lead to disability. The intensity of the clinical symptoms may vary for each individual. However, they typically become more severe, more frequent, and more debilitating over time. The rate of progression also varies for each individual. Common clinical symptoms include knee pain that is gradual in onset and worse with activity, knee stiffness and swelling, pain after prolonged sitting or resting, and pain that worsens over time. Treatment for knee osteoarthritis begins with conservative methods and progresses to surgical treatment options when conservative treatment fails. While medications can help slow the progression of RA and other inflammatory conditions, no proven disease-modifying agents for the treatment of knee osteoarthritis currently exist.

Knee osteoarthritis is classified as either primary or secondary, depending on its cause. Primary knee osteoarthritis is the result of articular cartilage degeneration without any known reason. This is typically thought of as degeneration due to age as well as wear and tear. Secondary knee osteoarthritis is the result of articular cartilage degeneration due to a known reason

Figure1 show the anatomy of knee



Risk Factors for Knee OA

Modifiable

* Articular trauma
* Occupation – prolonged standing and repetitive knee bending
* Muscle weakness or imbalance
* Weight
* Health – metabolic syndrome

Non-modifiable

* Gender - females more common than males
* Age
* Genetics
* Race

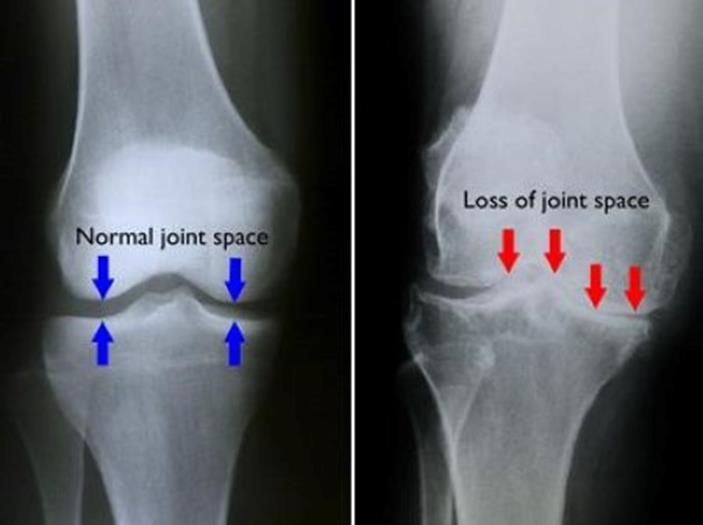
Knee osteoarthritis is the most common type of arthritis diagnosed, and its prevalence will continue to increase as life expectancy and obesity rises. Depending on the source, roughly 13% of women and 10% of men 60 years and older have symptomatic knee osteoarthritis. Among those older than 70 years of age, the prevalence rises to as high as 40%. The prevalence of knee osteoarthritis in males is also lower than in females. Interestingly, not everyone who demonstrates radiographic findings of knee osteoarthritis will be symptomatic. One study found that only 15% of patients with radiographic findings of knee OA were symptomatic. Not factoring in age, the incidence of symptomatic knee osteoarthritis is roughly 240 cases per 100,000 people per year.

Articular cartilage is composed primarily of type II collagen, proteoglycans, chondrocytes, and water. Healthy articular cartilage constantly maintains an equilibrium between each of the components so that any degradation of cartilage is matched by synthesis. Healthy articular cartilage is thus maintained. In the process of osteoarthritis, matrix metalloproteases (MMPs), or degradative enzymes, are overexpressed, disrupting the equilibrium and resulting in an overall loss of collagen and proteoglycans. In the early stages of osteoarthritis, chondrocytes secrete tissue inhibitors of MMPs (TIMPs) and attempt to increase the synthesis of proteoglycans to match the degradative process. However, this reparative process is not enough. The loss in equilibrium results in a decreased amount of proteoglycans despite increased synthesis, increased water content, the disorganized pattern of collagen, and ultimately loss of articular cartilage elasticity. Macroscopically these changes result in cracking and fissuring of the cartilage and ultimately erosion of the articular surface Although knee osteoarthritis is closely correlated with aging, it is important to note that knee osteoarthritis is not simply a consequence of aging but rather its own disease. This is supported by the differences seen in cartilage with both osteoarthritis and aging.

Furthermore, the enzymes responsible for cartilage degradation are expressed in higher amounts in knee osteoarthritis, whereas they are at normal levels in the normal aging cartilage

**Cartilage Changes in Aging**

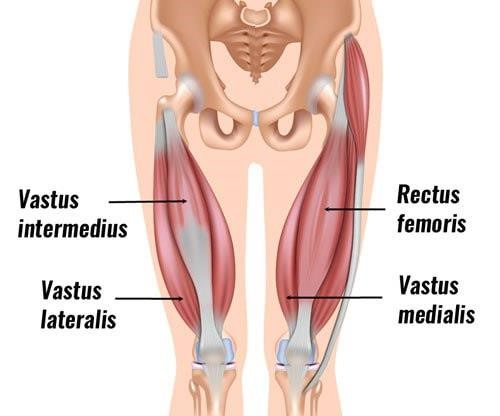
* Water content – decreased
* Collagen – same
* Proteoglycan content – decreased
* Proteoglycan synthesis – same
* Chondrocyte size – increased
* Chondrocyte number – decreased
* Modulus of elasticity – increased



**Figure2 show joins space**

(**Left**) In this x-ray of a normal knee, the space between the bones indicates healthy cartilage (arrows). (**Right**) This x-ray of an arthritic knee shows severe loss of joint space.

1.2. Background: Biologically:



What does the quadriceps muscle do?

The anatomy of the quadriceps muscle acts to bend the hip joint and straighten the knee joint. Additional to creating movement at these joints, the quadriceps muscle also functions to stabilize the knee joint during movement .Joint stability during movement is compromised with weak quadriceps muscles, increasing the risk of damage to knee joint structures .The quadriceps muscle also acts to absorb load and stresses applied to the knee joint . Hence, weakness in the quadriceps muscles could result in abnormal loading and subsequent structural damage to joint structures, including menisci, ligaments, cartilage and bone

Is quadriceps muscle strength related to knee pain?

Although knee pain may result from multiple factors including both biological and/or psychosocial, studies suggest that quadriceps muscle weakness can contribute to worsening knee pain. Contrarily, greater quadriceps muscle strength has been shown to prevent further damage to knee joint structures. It is also alluded to in research that a certain amount of quadriceps muscle strength is necessary to protect the knee from adverse loading. Therefore, quadriceps muscle strength is shown to be strongly associated with knee pain and disability. Interestingly, both the strength of the muscle contraction as well as the mass of the muscle contributes to the overall strength of the quadriceps muscle.

The role of the quadriceps muscle in mediating risk for knee osteoarthritis (OA) is a common subject of investigation. The quadriceps muscle is a principal contributor to knee joint stability and provides shock absorption for the knee during ambulation. Clinically, weakness of the quadriceps muscle is consistently found in patients with knee OA. Research has shown that higher quadriceps muscle strength is associated with a reduced risk for incident symptomatic knee OA. However, there is limited evidence to suggest that quadriceps muscle plays a significant role in the incidence of radiographic knee OA. In addition, greater quadriceps muscle strength is associated with a lower risk for progression of bio femoral joint space narrowing and cartilage loss in women. This article summarizes knowledge of the relationship between quadriceps muscle strength and risk for knee OA.

Is quadriceps muscle strength related to knee injuries?

A common finding in research is that reduced quadriceps muscle strength is a common clinical finding and risk factor among individuals with or at risk of knee osteoarthritis .Additionally, the progression of knee osteoarthritic changes has been associated with quadriceps muscle strength. Weakness in the quadriceps muscle is also a risk factor for non-contact anterior cruciate ligament injuries as well as patella femoral pain syndrome

**1.3. Deep Learning**

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It’s achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks.

The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.

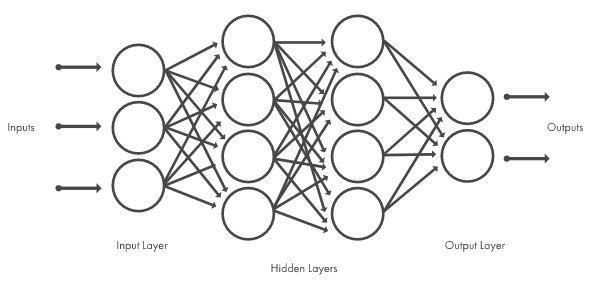


Figure 3 neural network

The model used:

## EfficientNetB5:

EfficientNetB5 is a convolutional neural network architecture that is part of the EfficientNet family, developed by Google AI. The EfficientNet models are designed to achieve high performance in image classification tasks with significantly lower computational costs compared to previous models. The EfficientNet family includes a range of models from B0 to B7, with B5 being one of the mid-sized variants.

### Key Features of EfficientNetB5:

1. **Compound Scaling**: EfficientNet models use a method called compound scaling, which uniformly scales all dimensions of the network—depth, width, and resolution—using a set of fixed scaling coefficients. This approach aims to balance the trade-offs between these dimensions and improve efficiency and performance.
2. **Baseline Network**: EfficientNetB0 serves as the baseline model, and it was developed using a neural architecture search (NAS). The other models, including EfficientNetB5, are scaled versions of this baseline.
3. **Performance**: EfficientNetB5 achieves higher accuracy than many previous models while using fewer parameters and less computational power. It offers a good balance between model size and accuracy.
4. **Architecture Details**:
   * **Depth (number of layers)**: EfficientNetB5 has more layers compared to smaller variants like B0, contributing to its ability to capture more complex features.
   * **Width (number of channels)**: It increases the number of channels in each layer, allowing for more feature maps and better feature extraction.
   * **Resolution (input image size)**: The input resolution is higher in EfficientNetB5, allowing it to capture more detailed information from images.

### Specifications of EfficientNetB5:

* **Input Image Size**: 456 x 456 pixels
* **Parameters**: Approximately 30 million
* **FLOPs**: Around 9.9 billion

### Advantages:

* **Efficiency**: Offers a good trade-off between computational efficiency and accuracy.
* **Scalability**: The model can be scaled up or down to meet different computational constraints and performance requirements.
* **Performance**: Achieves state-of-the-art accuracy on standard benchmarks like ImageNet.

### Applications:

EfficientNetB5 is commonly used in various image classification tasks in areas such as medical imaging, object detection, and other computer vision applications where high accuracy and efficiency are crucial.

In summary, EfficientNetB5 is a powerful and efficient convolutional neural network model, designed to deliver high performance in image classification tasks while maintaining computational efficiency.

**1.4. Processing of data:**

**1.4.1. Image processing**

The aim of pre-processing is to improve the quality of the image so that we can analyze it in a better way. By preprocessing we can suppress undesired distortions and enhance some features which are necessary for the particular application we are working for. Those features might vary for different applications.

Feature extractionextracts the useful information or image properties from the segmented images finally; this information is used in the classification system for training and testing purpose. The classification system is usually supported by intelligent classifier, such as neural network and support vector machine

**Techniques of image processing:**

Since the image database consists of both digital photo. These images are obtained from different source and the size of the images is non-standard. The x-ray images usually contain noise and air bubbles. These feature that is not part of the KOA and would reduce the accuracy of the border detection. In order to overcome these problems, the first step to do is apply some image processing techniques to the images.

**1.Data augmentation:** is a set of techniques used to increase the amount of data in a machine learning model by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It helps smooth out the machine learning model and reduce the over fitting of data.

1. **Image Scaling:** Image scaling techniques are applied due to the lack of same and standard size of images. Since the x-ray images may be gathered from different sources and sizes, the first step is to resize the images to have the fixed width pixels but variable size of height.
2. **Color Space Transformation:** Since color information plays an inevitable role in KOA detection systems, so we need to extract the more corresponding color of images for further processing. Generally, the common color spaces include RGB, HSV, HSI, gray scale and CIE-XY.
3. **Image Restoration:** Image Restoration is defined as the procedure to recover the degraded image from a blurred and noisy one. It can restore the degraded images in different ways. The image degradation can happen by various defects such as imperfection of imaging system, bad focusing, motion and etc. which make an image usually noisy or blur. Since the corrupted images lead to fault detection, hence, it is essential to know about noises present in an image to select the most appropriate diagnosing algorit

### 1.5. Overview of Data Set

**Knee Osteoarthritis imaging data**

Data Collection The research was conducted using information from a dataset provided by Osteoarthritis Initiative (OAI) our research utilizes only data from the baseline. As it was mentioned earlier, the classification is based on Lawrence-Kellgren metric that rank knee osteoarthritis severity from 0 to 4, this metric was provided by radiologist in the study.

This dataset contains knee X-ray data for both knee joint detection and knee KL grading. The Grade descriptions are as follows:

* **Grade 0:** Healthy knee image.
* **Grade 1**(Doubtful): Doubtful joint narrowing with possible osteophyte lipping
* **Grade 2**(Minimal): Definite presence of osteophytes and possible joint space narrowing
* **Grade 3** (Moderate): Multiple osteophytes, definite joint space narrowing, with mild sclerosis.
* **Grade 4** (Severe): Large osteophytes, significant joint narrowing, and severe sclerosis.

**Proposed model**

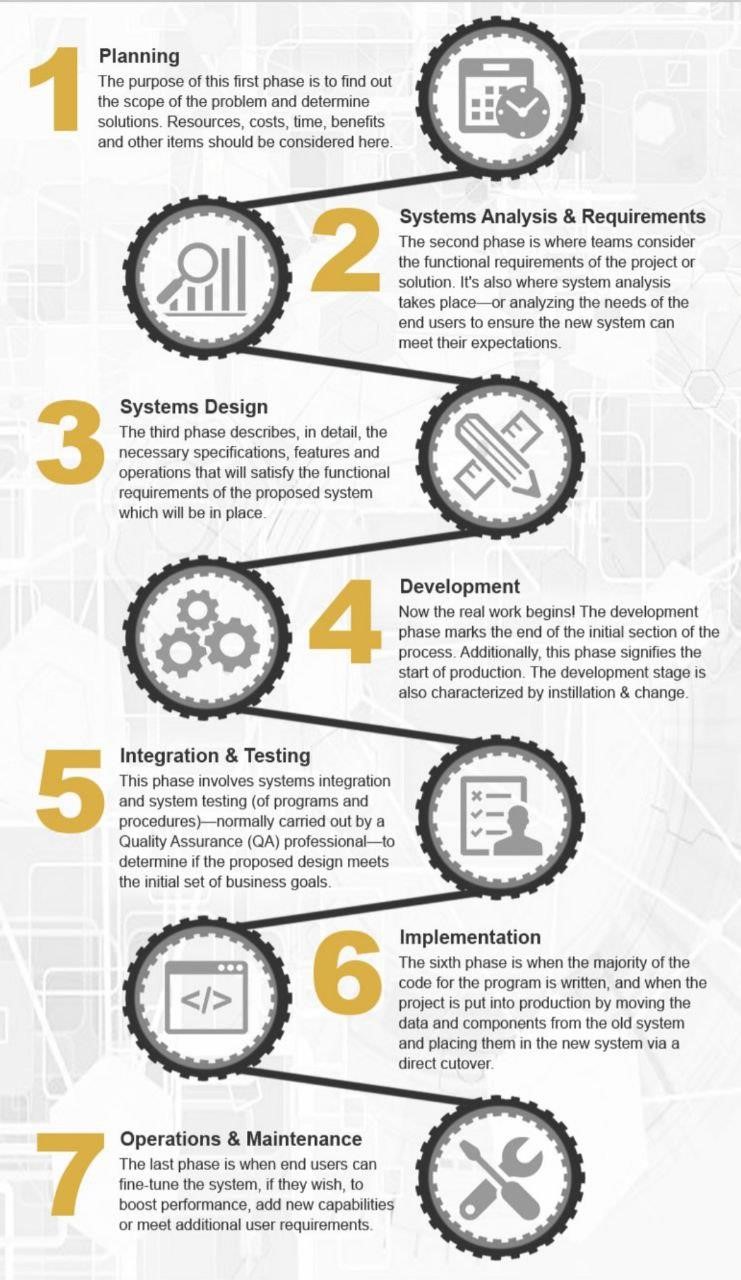
A diagram of a diagram

Description automatically generated

**1.6. System Development Life Cycle**

System Development Life Cycle (SDLC) is a conceptual model which includes policies and procedures for developing or altering systems throughout their life cycles ,

The systems development life cycle concept applies to a range of hardware and software configurations, as a system can be composed of hardware only, software only, or a combination of both. There are usually six stages in this cycle:

* + requirements
  + design
  + implementation
  + testing
  + deployment
  + operations  maintenance

#### 1.6.1 Feasibility Study or Planning

* Define the problem and scope of existing system.
* Overview the new system and determine its objectives.
* Confirm project feasibility and produce the project Schedule.
* During this phase, threats, constraints, integration and security of system are also considered.
* A feasibility report for the entire project is created at the end of this phase.

#### 1.6.2 Analysis and Specification

* Gather, analyze, and validate the information.
* Define the requirements and prototypes for new system.
* Examine the information needs of end-user and enhances the system goal.
* A Software Requirement Specification (SRS) document, which specifies the software, hardware, functional, and network requirements of the system, is prepared at the end of this phase.

#### 1.6.3 System Design

* Includes the design of application, network, databases, user interfaces, and system interfaces.
* Transform the SRS document into logical structure, which contains detailed and complete set of specifications that can be implemented in a programming language.
* Create a contingency, training, maintenance, and operation plan.
* Review the proposed design. Ensure that the final design must meet the requirements stated in SRS document.
* Finally, prepare a design document which will be used during next phases.

**1.6.4. Implementation**  Implement the design into source code through coding.

* Combine all the modules together into training environment that detects errors and defects.
* A test report which contains errors is prepared through test plan that includes test related tasks such as test case generation, testing criteria, and resource allocation for testing.
* Integrate the information system into its environment and install the new system.

#### 1.6.5. Maintenance/Support

* Include all the activities such as phone support or physical on-site support for users that is required once the system is installing.
* Implement the changes that software might undergo over a period of time, or implement any new requirements after the software is deployed at the customer location.
* It also includes handling the residual errors and resolve any issues that may exist in the system even after the testing phase.
* Maintenance and support may be needed for a longer time for large systems and for a short time for smaller systems.

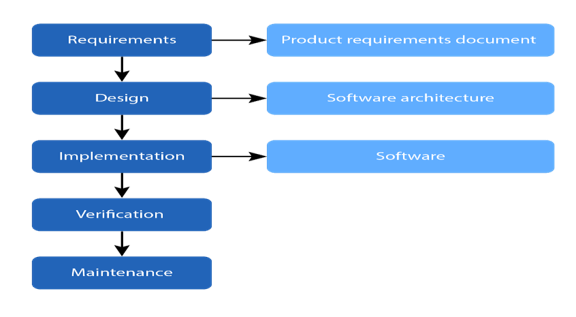
**1.7. Sex Main SDLC Models: Which One Is The Best?**

One of the fundamental notions of SDLC is the model. An SDLC model serves as the basis of development and should be adapted to the team, project, and product requirements.

There are different models, each of them having its strengths and weaknesses.

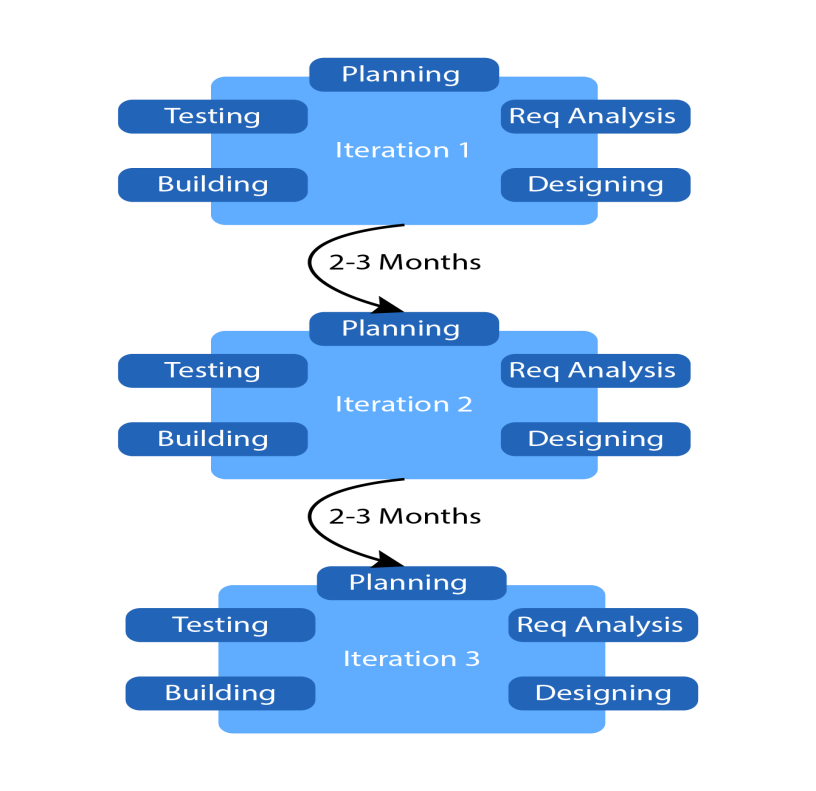
**1.7.1.Waterfall model**

The waterfall model is the earliest and the simplest of all the SDLC methodologies. The whole software development process is separated into phases, each phase beginning only when the previous one is completed. Every stage has its own project plan and relies on the information from the previous stage. The main SDLC stages of this model include gathering and analyzing the requirements, system design, implementation, testing, deployment, and launch. Such a model is widely used in software development as it is simple to plan and manage. The process is also strictly documented and the processes and outcomes of each stage are clearly predefined.



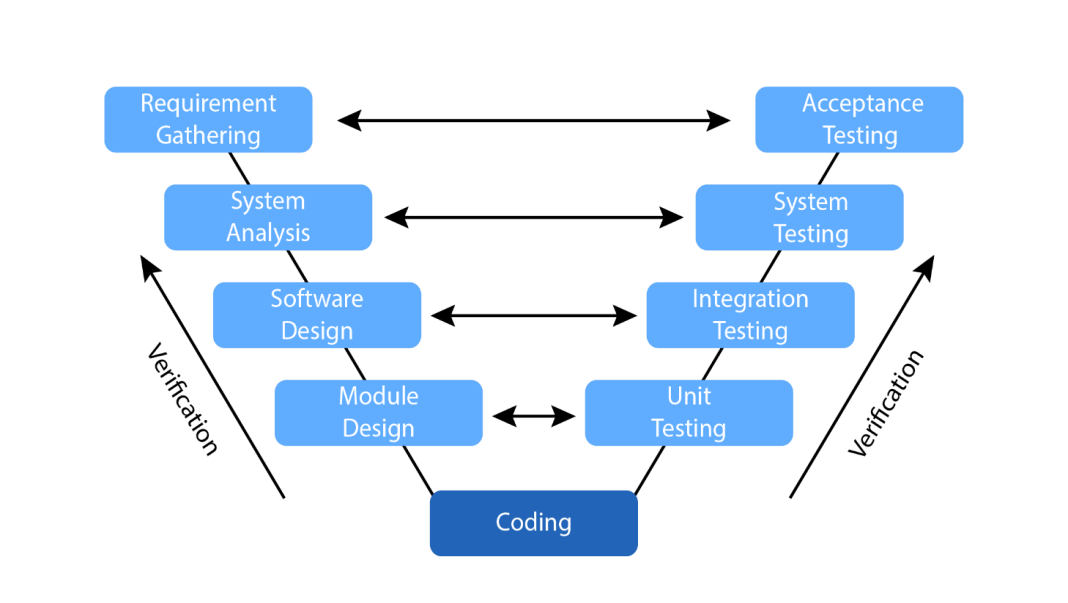
##### 1.7.2 Agile model

This software development life cycle model is considered a very practical one. It divides the project into cycles with short time limits which enables the quick delivery of a running product. At the end of each cycle, the product is tested. This SDLC methodology is the most widely used alternative of the waterfall one. Being a cycle model, it enhances the interactions between different development team members. It is also highly dependent on the interactions between the development company and the customer. The agile model is based on the adaptive approach that doesn’t require thorough planning. The developers adjust the process to the changing product requirements.



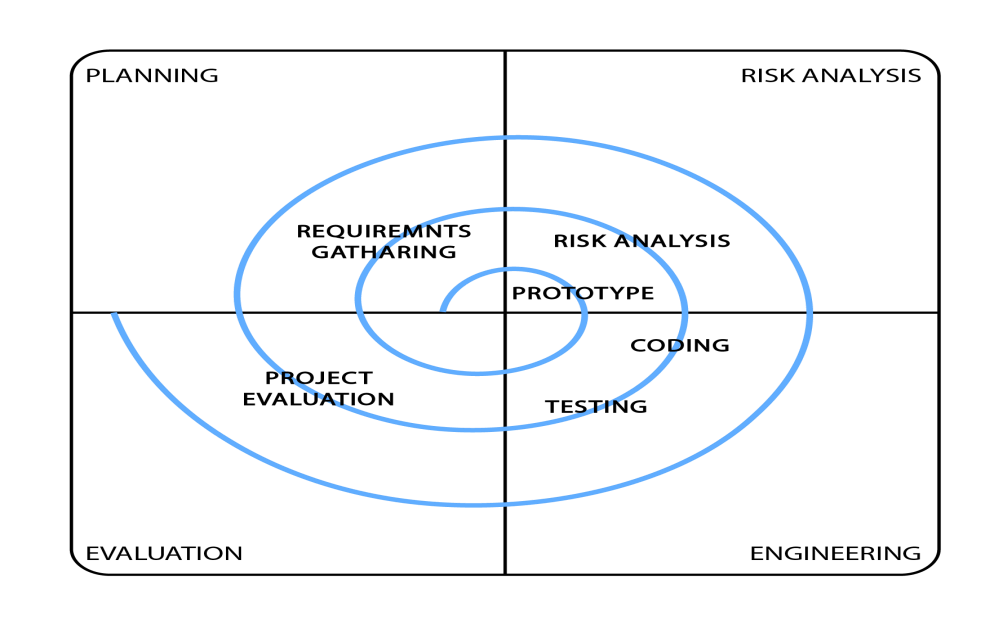
##### 1.7.3. V-shaped model

V-shaped SDLC model is an extended waterfall model, in which the testing (verification) and development (validation) stages go parallels, the testing stages correspond to certain development stages. This model has a strict plan and like in a waterfall model, the next stage doesn’t begin until the previous one is finished. According to this model, testing is done hierarchically and every stage has a certain deliverable. Errors are easy to detect at the early stages.



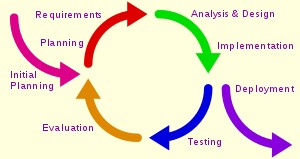
##### 1.7.3 Spiral model

According to the spiral model, the software development life cycle is divided into repetitive architecture and prototyping stages. The main problem of this SDLC model is to find the right moment to move on to the following stage. Every stage encompasses such phases as: planning, risk analysis, engineering and testing, and evaluation. These stages are repeated multiple times until the product is ready. For now, it is one of the most flexible methodologies that gives developers much freedom to create a highly personalized product. Shifting to the next stage is done even if work on the previous one is not completed.



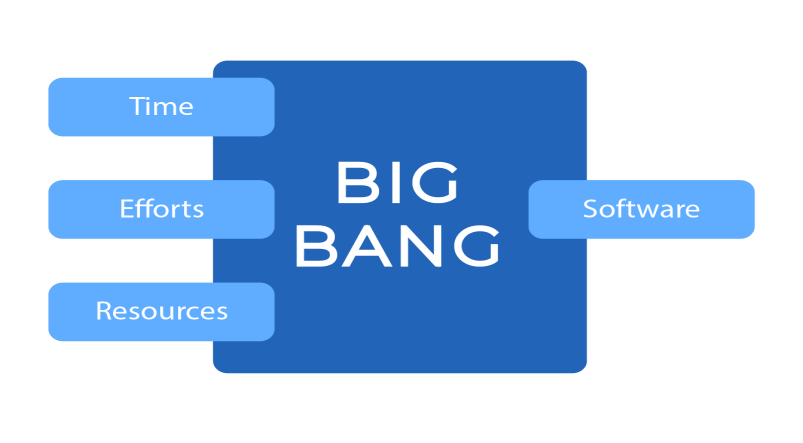
#### 1.7.3. Iterative Model

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.



##### 1.7.4. Big Bang model

This model is something out-of-the-box in software development. Its highlight is that there isn’t any specific process developers have to follow. There is the only stage - bang, which means starting the project immediately. Even the customer requirements are not clearly defined according to this model. As a rule, there are one or two developers working on the project. It may seem chaotic, but it works for small projects by saving time, energy, and cost on the initial stages.

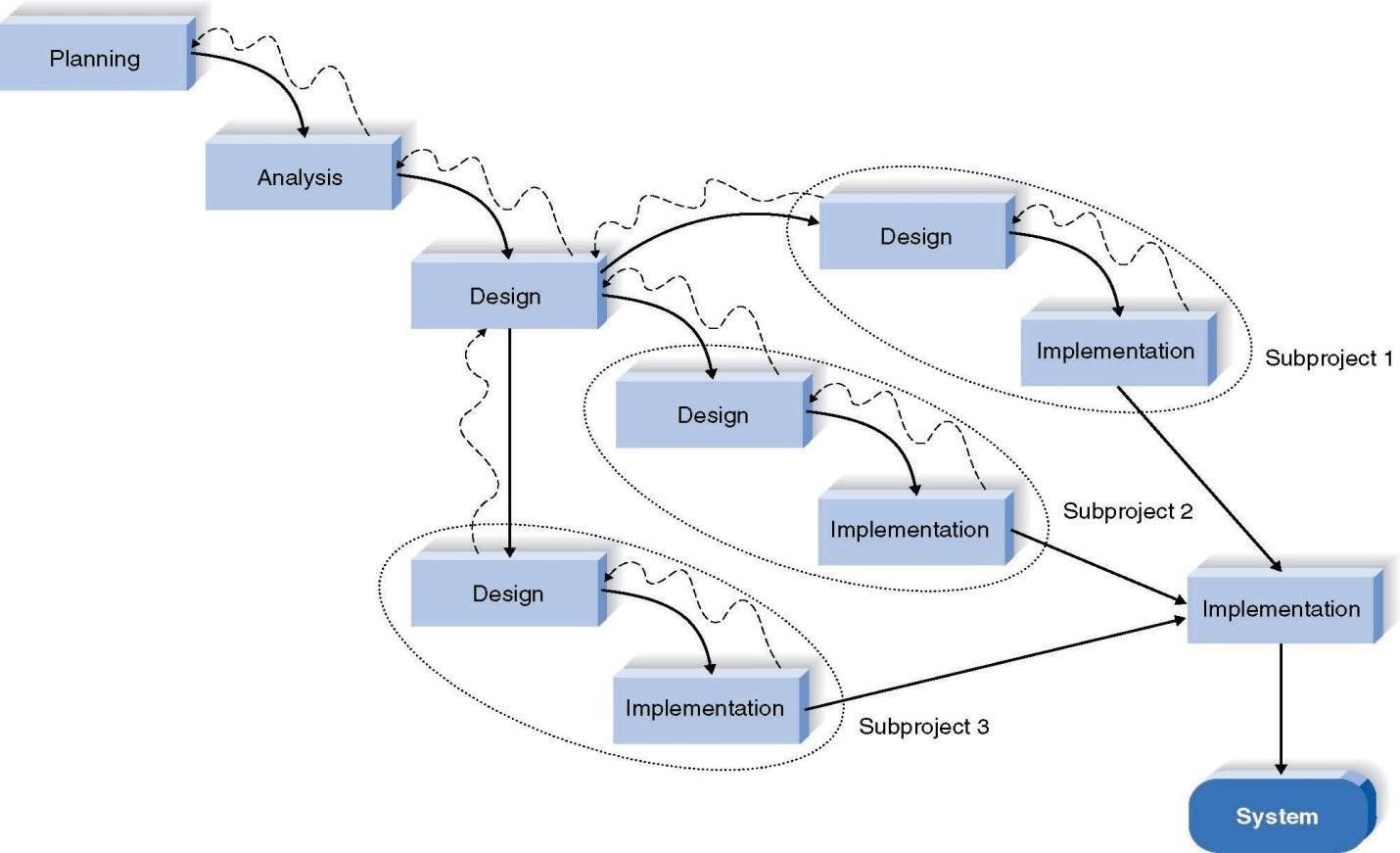


So after what has been explained we have chosen the **Iterative model** because the Iterative model will be most suitable when:

* The project requirements are clear, unambiguous, and detailed  The product definition is stable

**1.7.5. Parallel model**

Parallel development methodology attempts to address the problem of long delays between the analysis phase and the delivery of the system. Instead of doing design and implementation in sequence, it performs a general design for the whole system design and implementation in sequence, it performs a general design for the whole system and the system is delivered.



**Advantages of the parallel model:**

* It can reduce the time to deliver a system.
* There is less chance of changes in the business environment causing rework.

Our project working of parallel model, system developers divide a project into subprojects. Subprojects are integrated and merged into the final product. Subprojects are developed at the same time before being integrated into the final product.

**So we chose the parallel model.**

**Chapter 2**

Analysis and Design

**CH 2: Analysis Phase**

**3.1 As is system**

**In case of patient:**

* Patient faces a problem of how to detect knee osteoarthritis
* There are difficulties to present results of X-ray knee in an easy way
* Some of patients face a problem in find doctors and follow up with them.

**In case of doctors:**

* There are difficulties to find application about knee which have many features.
* There are difficulties to find an application which help to predict the Knee Osteoarthritis and its severity.
* It’s difficult to follow up with the patient with easy way.
* It’s difficult to early detection of knee osteoarthritis and analysis EMG signal
* There are many ways of known KOA but they need an effort and time to do a certain process.

**3.2 To be system**

**In case of patients:**

* The system provides knowledge to deal with knee Osteoarthritis.
* The system offers a real time prediction of KOA.
* The system offers easy way to contact with doctor and follow up.
* The user can insert the x-ray or EMG signal and get the results directly.

**In case of doctors:**

* The system combines features with different properties and different operation.
* The system helps the doctor to detect and early detection of KOA.
* The system provides a visualization tool that predicts the knee osteoarthritis.
* The systems allow that searching about doctors and make an appointment.
* It’s easy to use our application and get a clear result in a simple way.

**Functional requirement:**

1. Registration of patients, doctors
2. Enter the personal data
3. System asks the patient to uploading the image of the X ray of the patient Knee (4) System asks the patient to uploading the file that contain the EMG signal of knee musclues
4. System Preprocessing the image by removing the unwanted parts of the image
5. Post-processing the image that sharping and contrast the image

Scaling the image by resize the image

1. Color space Transformation by extracting the color of the image for processing
2. Restoration the image by removing noise & Blur from the image
3. system detect the image of the X ray if the patient has knee osteoarthritis.
4. system takes an appointment for the patient with a doctor
5. ChatBot with Patients

**Non-functional requirement:**

**(1)Security:** Assures that all data inside the system or its part will be protected against malware attacks or unauthorized access.

**(2)Availability:** The system is available for use 24 hours per day

**(3)Reliability:** The system or its element would run without a failure for a given period under predefined conditions. Traditionally, it’s expressed as a probability percentage. For example, if the system has 90 percent reliability for a month, this means that during this month, under normal usage conditions, there’s an 90 percent chance that the system won’t experience critical failure.

**(4)Recoverability:** The ability to recover from a crash or a failure in the system and returning to full operations.

**(5)Flexibility:**  The ease with which the software can be modified to adapt to different environments, configurations, and user expectations.

**(6)Capacity:** The system needs to support at least 1000 people at once

**3.3.Use case:**

|  |  |
| --- | --- |
| Actors | Goals |
| Patient | * Registration * Login * Determining Knee OA form x-ray image * Progression knee OA from EMG signal * Choose doctor * Make an appointment * Answer the questionnaire * Changing data of user * Contact with a doctor * Choose available time * Do exercise |
| Doctor | * Login * Registration * Contact with patient * Follow up with the patient * Ask the patient some questions * Determine available time * Make a report * Add diagnosis |

#### 3.3.1. Tables for Use Case: 1. Registration

|  |  |
| --- | --- |
| Use Case Name id | registration |
| Actor(s)Name | Patient &doctor |
| Trigger | User must enter the required data |
| Pre-condition | Open the application |
| Basic Bath | 1. Open the application 2. Enter your name 3. Enter gmail 4. Enter password |
| Alternative Bath | * If there are no internet the app sends to user “check your internet connection” * If user don’t enter the login page entries the system send to user” please enter the forgotten entry” |
| Post-condition | Registration is successfully |

1. **Login**

|  |  |
| --- | --- |
| Use Case Name id | login |
| Actor(s)Name | Patient &doctor |
| Trigger | User must enter the app and login with gmail and Password |
| Pre-condition | Open the application |
| Basic Bath | 1. Open the application 2. Enter your gmail 3. Enter user password |
| Alternative Bath | * If there are no internet the app sends to user “check your internet connection” * If user don’t enter the login page entries the system send to user” please enter the forgotten entry” |
| Post-condition | Login is successfully |

1. **Determining KOA the x-ray image**

|  |  |
| --- | --- |
| Use Case Name id | Upload x-ray |
| Actor(s)Name | Patient |
| Trigger | After login the system will ask the patient to Upload his/her knee x-ray |
| Pre-condition | 1. Open the application 2. Login 3. Upload your knee x-ray |
| Basic Bath | 1. Open the application 2. The system asks the patient to upload knee x-ray |
| Alternative Bath | If the patient does not have a picture of his knees , the system asks him to take a picture of his x-ray knee in the parallel position and upload it on the Application |
| Post-condition | The patient knows if he has KOA and it`s severity |

1. **Choose doctor**

|  |  |  |
| --- | --- | --- |
| Use Case Name id | choosing doctor | |
| Actor(s)Name | Patient | |
| Trigger | The patient choose the doctor from  “doctors” in menu | |
| Pre-condition | 1. | Open the application |
|  | 2. | Choose “doctors” from menu |
|  | 3. | Select the doctor who will follow your state |
| Basic Bath | 1. | Open the application |
|  | 2. | Enter to the menu and choose “doctors” |
|  | 3. | Choose your doctor from the list |
| Post-condition | * The patient will contact with the doctor * The doctor can follow his/her patient state perfectly | |

1. **Answer the questionnaire**

|  |  |
| --- | --- |
| Use Case Name id | answer the questionnaire |
| Actor(s)Name | patient |
| Trigger | system classifies patient state from his answers |
| Pre-condition | 1. Open the app 2. Doing registration |
| Basic Bath | After registration the system ask the patient to answer question |
| Alternative Bath | The system does backup for data to avoid missing it |
| Post-condition | Know the patient state |

1. **Changing data of user**

|  |  |  |
| --- | --- | --- |
| Use Case Name id | Change user data | |
| Actor(s)Name | user | |
| Trigger | The user can change his/her data from settings Of the app | |
| Alternative Bath |  | The user can change his/her(information) after contacting with the doctor |
| Post-condition | The user can change the data | |

1. **Contact with a doctor**

|  |  |
| --- | --- |
| Use Case Name id | Contact with a doctor |
| Actor(s)Name | Patient |
| Trigger | Contact with doctor to explain |
| Pre-condition | Open the application |
| Basic Bath | * Open the application * Choose doctors from doctor list * Choose your doctor |
| Alternative Bath | * The patient can chat with the doctor |
| Post-condition | The patient contact with doctor Successfully |

1. **Do exercise**

|  |  |
| --- | --- |
| Use Case Name id | Do exercise |
| Actor(s)Name | Patient |
| Trigger | The patient can less the severity of knee OA By doing some exercise given by the doctor |
| Pre-condition | Open the application |
| Basic Bath | * Open the application * List of videos in home screen |
| Alternative Bath | * The doctor put his phone number on the application so the patient can contact with him by the phone * The patient can chat with the doctor” |
| Post-condition | the patient choose the most suitable exercise |

**9.Follow up with the patient**

|  |  |
| --- | --- |
| Use Case Name id | Follow up with the patient |
| Actor(s)Name | Doctor |
| Trigger | A doctor communicates with his patient and follows up on any developments in his condition in order to work with him |
| Pre-condition | * Open the application  * Choose the patient * Contact with the patient |
| Basic Bath | 1. Open the application 2. Choose the patient 3. Contact with the patient The |
| Alternative Bath | The doctor asks patient to send report |
| Post-condition | The doctor follow up patient`s state successfully |

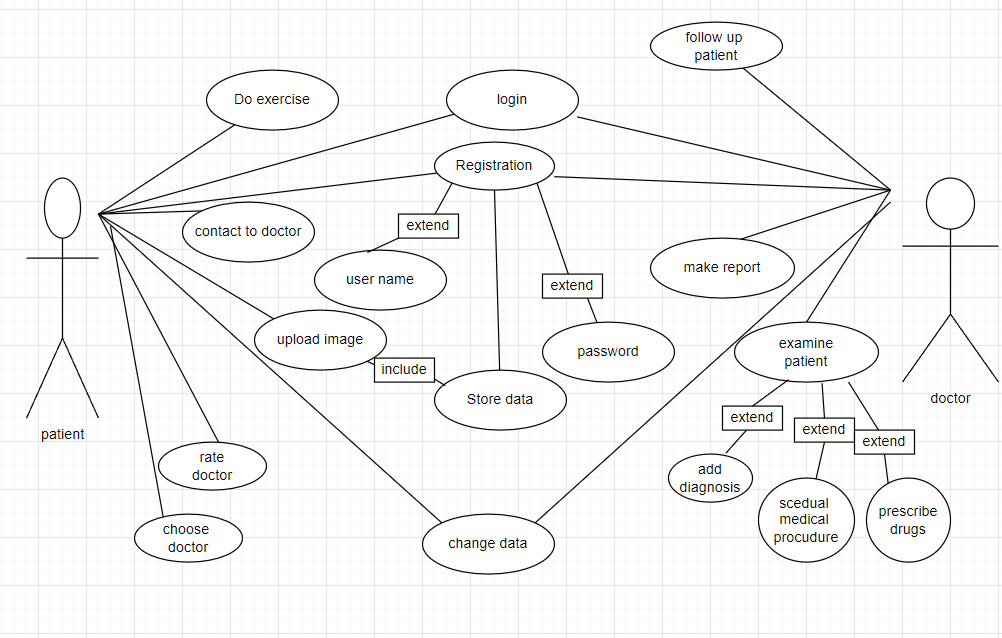
**10. Add diagnosis**

|  |  |
| --- | --- |
| Use Case Name id | Add diagnosis |
| Actor(s)Name | doctor |
| Trigger | A doctor communicates with his patient and adds diagnosis. |
| Pre-condition | Open the application |
| Basic Bath | * Open the application * Choose the patient |
| Alternative Bath | Contact with patient by chat |
| Post-condition | Update the state of the patient |

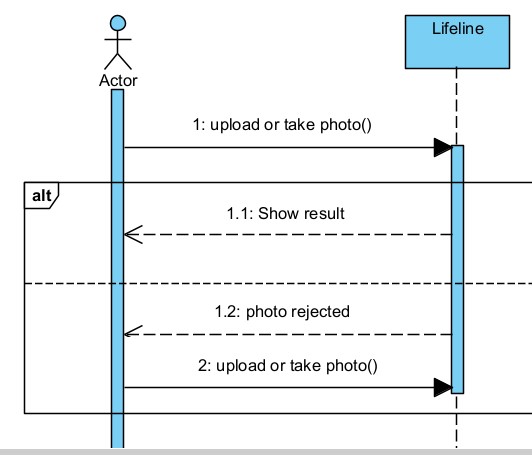
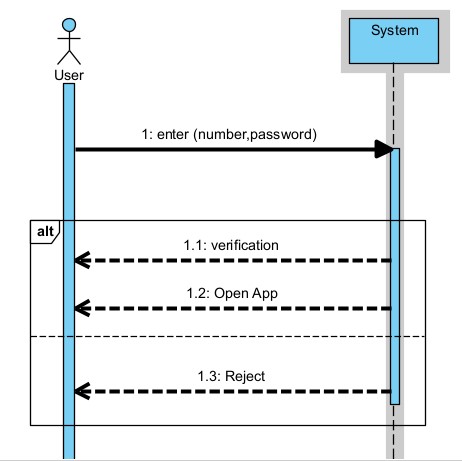
**11.Make a report**

|  |  |
| --- | --- |
| Use Case Name id | Make a report |
| Actor(s)Name | doctor |
| Trigger | Doctor make a report for the patient state That include the state of knee and medicine for his state |
| Pre-condition | * Open the app * Choose the patient from patient list |
| Basic Bath | Make a report from report sign that exist in front of the detected patient |
| Alternative Bath | The doctor add notes for patient to do |
| Post-condition | The patient know his state from that report and what medicine he will take. |

**3.3.2. Use case diagram**



#### 3.3.3.System sequence diagram (SSD)

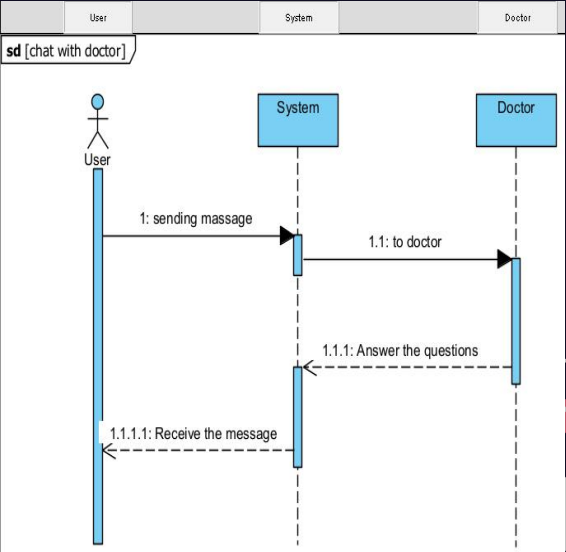


**Figure23 system sequence for (login) Figure24 system sequence for (upload image)**

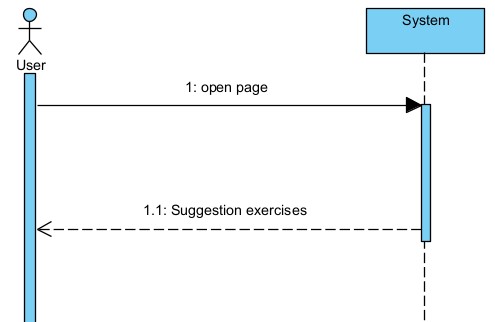
صورة تحتوي على نص, لقطة شاشة, خط, موازِ

تم إنشاء الوصف تلقائياً

**Figure25 system sequence for (upload X-Ray signal)**

****

**Figure25 system sequence for (contact with doctor)**



**Figure27 system sequence for (make exercises)**

**We user MVVM Architecture to provide clean code.**

**Auth feature :**

**Flutter Code :**

**Auth cubit:**

import 'dart:io';  
import 'package:cloud\_firestore/cloud\_firestore.dart';  
import 'package:firebase\_auth/firebase\_auth.dart';  
import 'package:firebase\_storage/firebase\_storage.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:save\_knee\_23/constants.dart';  
import 'package:save\_knee\_23/features/auth/view\_models/login\_data.dart';  
import 'package:save\_knee\_23/features/home/view\_models/user\_model.dart';  
import 'package:save\_knee\_23/features/home/view\_models/video\_model.dart';  
  
import '../view\_models/register\_data.dart';  
  
part 'auth\_state.dart';  
  
class AuthCubit extends Cubit<AuthState> {  
 AuthCubit() : super(AuthInitial());  
  
 static AuthCubit *get*(BuildContext context) => BlocProvider.*of*(context);  
  
 CollectionReference videosCollection =  
 FirebaseFirestore.*instance*.collection('videos');  
 CollectionReference doctorCollection =  
 FirebaseFirestore.*instance*.collection(kDoctorsCollection);  
 CollectionReference userCollection =  
 FirebaseFirestore.*instance*.collection('users');  
 final storage = FirebaseStorage.*instance*;  
 bool? isLoading = false;  
 late UserModel userModel;  
 List<UserModel> doctorsList = [];  
 List<VideoModel> videosList = [];  
  
 Future<void> userRegister(RegisterData registerData) async {  
 emit(AuthLoading());  
 try {  
 final userCredential = await FirebaseAuth.*instance* .createUserWithEmailAndPassword(  
 email: registerData.email, password: registerData.password);  
 final uId = userCredential.user!.uid;  
 final userModel = UserModel(name: registerData.name, uId: uId);  
 await userCollection.doc(uId).set(userModel.toJson());  
 await getUser();  
 emit(AuthSuccess());  
 } on FirebaseAuthException catch (e) {  
 if (e.code == 'weak-password') {  
 emit(AuthFailure(errMessage: 'weak-password'));  
 } else if (e.code == 'email-already-in-use') {  
 emit(AuthFailure(errMessage: 'email already in use'));  
 } else {  
 emit(AuthFailure(errMessage: 'authentication error'));  
 }  
 } catch (e) {  
 emit(AuthFailure(errMessage: 'something went Wrong'));  
 }  
 }  
  
 Future<void> userLogin(LoginData loginData) async {  
 emit(AuthLoading());  
 try {  
 final userCredential = await FirebaseAuth.*instance* .signInWithEmailAndPassword(  
 email: loginData.email, password: loginData.password);  
 final uId = userCredential.user!.uid;  
 final docSnapshot = await userCollection.doc(uId).get();  
 userModel = UserModel.fromJson(docSnapshot);  
 emit(AuthSuccess());  
 } on FirebaseAuthException catch (e) {  
 if (e.code == 'user-not-found') {  
 emit(AuthFailure(errMessage: 'user-not-found'));  
 } else if (e.code == 'wrong-password') {  
 emit(AuthFailure(errMessage: 'wrong-password'));  
 } else {  
 emit(AuthFailure(errMessage: 'authentication error'));  
 }  
 } catch (e) {  
 emit(AuthFailure(errMessage: 'something went wrong'));  
 }  
 }  
  
 Future<void> uploadImages(File imageFile) async {  
 emit(ImageInitial());  
 try {  
 final Reference storageRef = storage.ref().child('users');  
 final UploadTask uploadTask =  
 storageRef.child('${imageFile.hashCode}').putFile(imageFile);  
 final TaskSnapshot snapshot = await uploadTask;  
 final String downloadUrl = await snapshot.ref.getDownloadURL();  
 final uId = FirebaseAuth.*instance*.currentUser!.uid;  
 await userCollection.doc(uId).update({'profileImage': downloadUrl});  
 await getUser();  
 emit(ImageLoaded());  
 } catch (e) {  
 emit(ImageFailure(errMessage: e.toString()));  
 }  
 }  
  
 getDoctorsList() {  
 doctorCollection.snapshots().listen((event) {  
 doctorsList.clear();  
 for (var doc in event.docs) {  
 doctorsList.add(UserModel.fromJson(doc));  
 }  
 emit(DoctorsListLoaded());  
 });  
 }  
  
 getVideosList() {  
 videosCollection.snapshots().listen((event) {  
 videosList.clear();  
 for (var doc in event.docs) {  
 videosList.add(VideoModel.fromJson(doc));  
 }  
 });  
 }  
  
 Future<void> getUser() async {  
 final uId = FirebaseAuth.*instance*.currentUser!.uid;  
 final docSnapshot = await userCollection.doc(uId).get();  
 userModel = UserModel.fromJson(docSnapshot);  
 }  
  
 Future<bool> getAuthStatus() async {  
 final uId = FirebaseAuth.*instance*.currentUser!.uid;  
 final docSnapshot = await userCollection.doc(uId).get();  
 userModel = UserModel.fromJson(docSnapshot);  
 bool isUser = FirebaseAuth.*instance*.currentUser != null;  
 getDoctorsList();  
 return isUser;  
 }  
}

part of 'auth\_cubit.dart';  
  
abstract class AuthState {}  
class AuthInitial extends AuthState {}  
class AuthLoading extends AuthState {}  
class GetUserSuccess extends AuthState {}  
class AuthSuccess extends AuthState {}  
class AuthFailure extends AuthState {  
 final String errMessage;  
 AuthFailure({required this.errMessage});  
  
}  
class ImageInitial extends AuthState {}  
class ImageLoading extends AuthState {}  
  
class ImageLoaded extends AuthState {  
}  
class ImageFailure extends AuthState {  
 final String errMessage;  
  
 ImageFailure({required this.errMessage});  
}  
class DoctorsListLoaded extends AuthState {}

**chat feature**

**chat cubit :**

import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:cloud\_firestore/cloud\_firestore.dart';  
import 'package:firebase\_auth/firebase\_auth.dart';  
import 'package:save\_knee\_23/constants.dart';  
import 'package:save\_knee\_23/features/chat/view\_models/message\_model.dart';  
  
part 'chat\_state.dart';  
  
class ChatCubit extends Cubit<ChatState> {  
 ChatCubit() : super(ChatInitial());  
 List<MessageModel> messagesList = [];  
 CollectionReference doctorCollection =  
 FirebaseFirestore.*instance*.collection(kDoctorsCollection);  
 final String uId = FirebaseAuth.*instance*.currentUser!.uid;  
  
 static ChatCubit *get*(BuildContext context) => BlocProvider.*of*(context);  
  
 CollectionReference messagesCollection =  
 FirebaseFirestore.*instance*.collection(kMessagesCollection);  
  
 Future<void> sendMassage(  
 String message, String docId, String senderId) async {  
 final DocumentReference groupMessageDocRef =  
 messagesCollection.doc(docId).collection(kMessagesCollection).doc();  
 final messageId = groupMessageDocRef.id;  
 final messageModel = MessageModel(  
 message: message,  
 date: DateTime.now(),  
 uId: uId,  
 messageId: messageId,  
 senderId: senderId);  
 await groupMessageDocRef.set(messageModel.toJson());  
 emit(SendMessageSuccess());  
 }  
  
 void getMessages(String docId) {  
 messagesCollection  
 .doc(docId)  
 .collection(kMessagesCollection)  
 .snapshots()  
 .listen(  
 (querySnapshot) {  
 if (querySnapshot.docs.isEmpty) return;  
 messagesList = querySnapshot.docs  
 .map((doc) => MessageModel.fromJson(doc.data()))  
 .toList();  
 emit(GetMessagesSuccess());  
 },  
 );  
 }  
}

part of 'chat\_cubit.dart';  
  
abstract class ChatState {}  
  
class ChatInitial extends ChatState {}  
  
class SendMessageSuccess extends ChatState {}  
class GetMessagesSuccess extends ChatState {}

**message model:**

import 'package:cloud\_firestore/cloud\_firestore.dart';  
  
class MessageModel {  
 final String message;  
 final DateTime date;  
 final String uId;  
 final String messageId;  
 final String senderId;  
  
 MessageModel({  
 required this.message,  
 required this.date,  
 required this.uId,  
 required this.messageId,  
 required this.senderId,  
 });  
  
 factory MessageModel.fromJson(Map<String, dynamic> json) {  
 return MessageModel(  
 message: json['message'] ?? '',  
 date: (json['date'] as Timestamp).toDate().toLocal(),  
 uId: json['uId'] ?? '',  
 messageId: json['messageId'] ?? '',  
 senderId: json['senderId'] ?? '',  
 );  
 }  
  
 Map<String, dynamic> toJson() {  
 return {  
 'message': message,  
 'date': Timestamp.fromDate(date),  
 'uId': uId,  
 'messageId': messageId,  
 'senderId': senderId,  
 };  
 }  
}

**chart bubble :**

import 'package:flutter/material.dart';  
import 'package:save\_knee\_23/features/chat/view\_models/message\_model.dart';  
  
class ChatBubble extends StatelessWidget {  
 final MessageModel messageModel;  
  
 const ChatBubble({  
 super.key,  
 required this.messageModel,  
 });  
  
 @override  
 Widget build(BuildContext context) {  
 return Align(  
 alignment: Alignment.*centerLeft*,  
 child: Container(  
 padding: const EdgeInsets.only(  
 left: 32,  
 right: 32,  
 top: 28,  
 bottom: 28,  
 ),  
 margin: const EdgeInsets.symmetric(  
 horizontal: 10,  
 vertical: 5,  
 ),  
 decoration: const BoxDecoration(  
 borderRadius: BorderRadius.only(  
 topRight: Radius.circular(32),  
 topLeft: Radius.circular(32),  
 bottomRight: Radius.circular(32),  
 ),  
 color: Color(0xff274460)),  
 child: Text(  
 messageModel.message,  
 style: const TextStyle(color: Colors.*white*, fontSize: 13),  
 ),  
 ),  
 );  
 }  
}

**Chat view body :** import 'package:firebase\_auth/firebase\_auth.dart' hide User;  
import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:save\_knee\_23/core/utils/assets.dart';  
import 'package:save\_knee\_23/features/chat/chat\_cubit/chat\_cubit.dart';  
import 'package:save\_knee\_23/features/home/view\_models/user\_model.dart';  
  
import 'chat\_bubble.dart';  
  
class ChatViewBody extends StatefulWidget {  
 final UserModel? doctorModel;  
  
 const ChatViewBody({super.key, this.doctorModel});  
  
 @override  
 State<ChatViewBody> createState() => \_ChatViewBodyState();  
}  
  
class \_ChatViewBodyState extends State<ChatViewBody> {  
 TextEditingController controller = TextEditingController();  
  
 @override  
 void didChangeDependencies() {  
 ChatCubit.*get*(context).getMessages(widget.doctorModel?.uId ?? '');  
 super.didChangeDependencies();  
 }  
  
 @override  
 void deactivate() {  
 ChatCubit.*get*(context).messagesList.clear();  
 super.deactivate();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 var chat = ChatCubit.*get*(context);  
 return Scaffold(  
 backgroundColor: Colors.*white*,  
 appBar: AppBar(  
 backgroundColor: const Color(0xff274460),  
 title: Row(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: [  
 Padding(  
 padding: const EdgeInsets.only(right: 10),  
 child: Text(  
 'chat with ${widget.doctorModel!.name}',  
 ),  
 ),  
 widget.doctorModel!.profileImage != null &&  
 widget.doctorModel!.profileImage!.isNotEmpty  
 ? CircleAvatar(  
 radius: 20,  
 backgroundImage:  
 NetworkImage(widget.doctorModel!.profileImage!),  
 )  
 : const CircleAvatar(  
 radius: 20,  
 backgroundImage: AssetImage(AssetsData.*doctorImage*),  
 )  
 ],  
 ),  
 ),  
 body: BlocConsumer<ChatCubit, ChatState>(  
 listener: (\_, state) {  
 if (state is SendMessageSuccess) {}  
 },  
 builder: (\_, state) {  
 return Column(  
 children: [  
 Expanded(  
 child: ListView.builder(  
 reverse: true,  
 itemCount: chat.messagesList.length,  
 itemBuilder: (\_, index) => ChatBubble(  
 messageModel: chat.messagesList[index],  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: TextField(  
 controller: controller,  
 decoration: InputDecoration(  
 hintText: 'Send Message',  
 suffixIcon: IconButton(  
 icon: const Icon(  
 Icons.*send*,  
 ),  
 onPressed: () {  
 final message = controller.text;  
 if (controller.text.isNotEmpty) {  
 controller.clear();  
 chat.sendMassage(  
 message,  
 widget.doctorModel?.uId ?? '',  
 FirebaseAuth.*instance*.currentUser?.uid ?? '');  
 debugPrint("Messages: ${chat.messagesList.length}");  
 }  
 },  
 ),  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10),  
 ),  
 ),  
 ),  
 ),  
 ],  
 );  
 },  
 ),  
 );  
 }  
}

**Chat view:**

import 'package:flutter/material.dart';  
import 'package:save\_knee\_23/features/chat/views/widgets/chat\_view\_body.dart';  
import 'package:save\_knee\_23/features/home/view\_models/user\_model.dart';  
  
class ChatView extends StatelessWidget {  
 final UserModel? doctorModel;  
  
 const ChatView({super.key, this.doctorModel});  
  
 @override  
 Widget build(BuildContext context) {  
 return ChatViewBody(doctorModel: doctorModel);  
 }  
}

**home feature :**

**home cubit :**

import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:save\_knee\_23/features/home/views/widgets/home\_view\_body.dart';  
import 'package:save\_knee\_23/screens/xray\_screen.dart';  
  
import '../views/chat\_list\_view.dart';  
  
part 'home\_state.dart';  
  
class HomeCubit extends Cubit<HomeState> {  
 HomeCubit() : super(HomeInitial());  
  
 static HomeCubit *get*(BuildContext context) => BlocProvider.*of*(context);  
  
 // CollectionReference userCollection =  
 // FirebaseFirestore.instance.collection('users');  
 // final storage = FirebaseStorage.instance;  
 // UserModel? userModel;  
 //  
 // Future<void> uploadImages(File imageFile) async {  
 // emit(HomeInitial());  
 // try {  
 // final Reference storageRef = storage.ref().child('users');  
 // final UploadTask uploadTask =  
 // storageRef.child('${imageFile.hashCode}').putFile(imageFile);  
 // final TaskSnapshot snapshot = await uploadTask;  
 // final String downloadUrl = await snapshot.ref.getDownloadURL();  
 // final uId = FirebaseAuth.instance.currentUser!.uid;  
 // await userCollection.doc(uId).update({'profileImage': downloadUrl});  
 // await getUser();  
 // emit(ImageLoaded());  
 // } catch (e) {  
 // emit(ImageFailure(errMessage: e.toString()));  
 // }  
 // }  
 //  
 // Future<void> getUser() async {  
 // final uId = FirebaseAuth.instance.currentUser!.uid;  
 // final docSnapshot = await userCollection.doc(uId).get();  
 // userModel = UserModel.fromJson(docSnapshot);  
 // }  
 List<Widget> screens = [  
 const HomeViewBody(),  
 XRayScreen(),  
 const ChatListView(),  
 ];  
 int currentIndex = 0;  
  
 void changeBottomNav(index) {  
 currentIndex = index;  
 emit(ChangeBottomNavState());  
 }  
}

part of 'home\_cubit.dart';  
  
abstract class HomeState {}  
  
class HomeInitial extends HomeState {}  
class ImageLoading extends HomeState {}  
class ImageLoaded extends HomeState {  
}  
class ImageFailure extends HomeState {  
 final String errMessage;  
  
 ImageFailure({required this.errMessage});  
}  
class ChangeBottomNavState extends HomeState {}

**user model :**

class UserModel {  
 late final String name;  
 String? profileImage;  
 String? uId;  
  
 UserModel({required this.name, required this.uId, this.profileImage = ''});  
  
 UserModel.fromJson(json) {  
 name = json['name'] as String;  
 profileImage = json['profileImage'] as String?;  
 uId = json['uId'] as String?;  
 }  
  
 Map<String, dynamic> toJson() {  
 return {  
 'name': name,  
 'profileImage': profileImage,  
 'uId': uId,  
 };  
 }  
}

**home widgets :**

**chat list view :**

import 'package:flutter/material.dart';  
import 'package:go\_router/go\_router.dart';  
import 'package:save\_knee\_23/core/utils/app\_router.dart';  
import 'package:save\_knee\_23/core/utils/assets.dart';  
import 'package:save\_knee\_23/features/home/view\_models/user\_model.dart';  
  
class ChatListItem extends StatelessWidget {  
 const ChatListItem({super.key, required this.docModel});  
  
 final UserModel docModel;  
  
 @override  
 Widget build(BuildContext context) {  
 return Row(  
 children: [  
 docModel.profileImage != null && docModel.profileImage!.isNotEmpty  
 ? CircleAvatar(  
 backgroundColor: Colors.*grey*,  
 radius: 40,  
 backgroundImage: NetworkImage(docModel.profileImage!),  
 )  
 : const CircleAvatar(  
 backgroundColor: Colors.*grey*,  
 radius: 40,  
 backgroundImage: AssetImage(AssetsData.*doctorImage*),  
 ),  
 const SizedBox(  
 width: 10,  
 ),  
 Column(  
 children: [  
 Text(  
 'Dr. ${docModel.name}',  
 style: TextStyle(fontSize: 18),  
 ),  
 Text('Physiotherapist')  
 ],  
 ),  
 const Spacer(),  
 Padding(  
 padding: const EdgeInsets.only(right: 6),  
 child: IconButton(  
 onPressed: () {  
 GoRouter.*of*(context).push(AppRouter.*kChatView*, extra: docModel);  
 },  
 icon: const Icon(Icons.*chat*)),  
 )  
 ],  
 );  
 }  
}

**appbar :**

import 'dart:io';  
import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:image\_picker/image\_picker.dart';  
import 'package:save\_knee\_23/constants.dart';  
import 'package:save\_knee\_23/core/utils/assets.dart';  
import '../../../auth/auth\_cubit/auth\_cubit.dart';  
  
class CustomHomeViewAppBar extends StatefulWidget {  
 const CustomHomeViewAppBar({super.key});  
  
 @override  
 State<CustomHomeViewAppBar> createState() => \_CustomHomeViewAppBarState();  
}  
  
class \_CustomHomeViewAppBarState extends State<CustomHomeViewAppBar> {  
 File? imageFile;  
  
 @override  
 Widget build(BuildContext context) {  
 double height = MediaQuery.*of*(context).size.height;  
 double width = MediaQuery.*of*(context).size.width;  
 var auth = BlocProvider.*of*<AuthCubit>(context);  
 return Stack(  
 children: [  
 Container(  
 height: height \* 0.15,  
 width: width,  
 decoration: const BoxDecoration(  
 borderRadius: BorderRadius.only(  
 bottomRight: Radius.circular(20),  
 bottomLeft: Radius.circular(20)),  
 color: kPrimaryColor,  
 ),  
 child: Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 23),  
 child: Row(  
 mainAxisAlignment: MainAxisAlignment.spaceBetween,  
 children: [  
 Column(  
 mainAxisAlignment: MainAxisAlignment.spaceBetween,  
 children: [  
 Opacity(  
 opacity: .9,  
 child: Text(  
 'Hi ${auth.userModel.name}',  
 style: const TextStyle(  
 fontSize: 20,  
 color: Colors.*white*,  
 ),  
 )),  
 const Text(  
 'Find Your Doctor',  
 style: TextStyle(  
 fontSize: 20,  
 color: Colors.*white*,  
 ),  
 ),  
 ],  
 ),  
 BlocBuilder<AuthCubit, AuthState>(  
 buildWhen: (\_, currentState) =>  
 currentState is ImageInitial ||  
 currentState is ImageFailure ||  
 currentState is ImageLoaded,  
 builder: (context, state) {  
 return Row(  
 children: [  
 auth.userModel.profileImage != null &&  
 auth.userModel.profileImage!.isNotEmpty  
 ? CircleAvatar(  
 backgroundColor: kPrimaryColor,  
 backgroundImage: NetworkImage(  
 auth.userModel.profileImage!,  
 ),  
 radius: 40,  
 )  
 : const CircleAvatar(  
 backgroundImage: AssetImage(  
 AssetsData.*kneeImage*,  
 ),  
 radius: 40,  
 ),  
 IconButton(  
 onPressed: () async {  
 await pickImage(auth);  
 },  
 icon: const Icon(  
 Icons.*add\_a\_photo*,  
 size: 16,  
 ))  
 ],  
 );  
 },  
 ),  
 ],  
 ),  
 ),  
 )  
 ],  
 );  
 }  
  
 Future<void> pickImage(AuthCubit auth) async {  
 try {  
 final ImagePicker picker = ImagePicker();  
 final XFile? xFile = await picker.pickImage(  
 source: ImageSource.gallery,  
 );  
 if (xFile != null) {  
 File xFilePathToFile(XFile xFile) {  
 return File(xFile.path);  
 }  
  
 imageFile = xFilePathToFile(xFile);  
 if (imageFile != null) {  
 auth.uploadImages(imageFile!);  
 }  
 }  
 } catch (e) {  
 ScaffoldMessenger.*of*(context)  
 .showSnackBar(const SnackBar(content: Text('please pick an image')));  
 }  
 }  
}

**doctor container:**

import 'package:flutter/material.dart';  
import 'package:go\_router/go\_router.dart';  
import 'package:save\_knee\_23/core/utils/app\_router.dart';  
import 'package:save\_knee\_23/core/utils/assets.dart';  
  
import '../../view\_models/user\_model.dart';  
  
class DoctorContainer extends StatelessWidget {  
 const DoctorContainer(  
 {super.key, required this.doctorModel, required this.color});  
  
 final UserModel doctorModel;  
 final Color color;  
  
 @override  
 Widget build(BuildContext context) {  
 double height = MediaQuery.*of*(context).size.height;  
 return AspectRatio(  
 aspectRatio: 6 / 3.8,  
 child: Container(  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(16),  
 color: color.withOpacity(0.2)),  
 child: Row(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 doctorModel.profileImage != null &&  
 doctorModel.profileImage!.isNotEmpty  
 ? Expanded(  
 flex: 3,  
 child: Container(  
 height: height,  
 decoration: BoxDecoration(  
 borderRadius:  
 const BorderRadius.all(Radius.circular(16)),  
 color: color,  
 image: DecorationImage(  
 fit: BoxFit.cover,  
 image: NetworkImage(doctorModel.profileImage!))),  
 ),  
 )  
 : Expanded(  
 flex: 3,  
 child: Container(  
 height: height,  
 decoration: BoxDecoration(  
 borderRadius:  
 const BorderRadius.all(Radius.circular(16)),  
 color: color,  
 image: const DecorationImage(  
 fit: BoxFit.cover,  
 image: AssetImage(AssetsData.*doctorImage*))),  
 ),  
 ),  
 Expanded(  
 flex: 2,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 'Dr.${doctorModel.name}',  
 style: const TextStyle(fontSize: 20),  
 ),  
 Container(  
 decoration: BoxDecoration(  
 color: color.withOpacity(0.3),  
 borderRadius: BorderRadius.circular(16)),  
 child: IconButton(  
 padding: EdgeInsets.*zero*,  
  
 // icon thatttttttttttttttttt onPressed  
 onPressed: () {  
 GoRouter.*of*(context)  
 .push(AppRouter.*kChatView*, extra: doctorModel);  
 },  
 icon: Icon(Icons.*message\_outlined*, color: color),  
 ),  
 )  
 ],  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

**ExcVidList :**

import 'package:flutter/material.dart';  
import 'package:save\_knee\_23/constants.dart';  
import 'package:url\_launcher/url\_launcher.dart';  
  
import 'fetch\_subnails.dart';  
  
class ExcVidList extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return ListView.builder(  
 scrollDirection: Axis.horizontal,  
 itemCount: urlList.length,  
 itemBuilder: (BuildContext context, int index) {  
 return InkWell(  
 onTap: () {  
 Uri url = Uri.*parse*(urlList[index]);  
 launchUrl(url);  
 },  
 child: Padding(  
 padding: EdgeInsets.all(2),  
 child: ClipRRect(  
 borderRadius: BorderRadius.circular(10),  
 child: Stack(  
 alignment: Alignment.*center*,  
 children: [  
 ColorFiltered(  
 colorFilter:  
 const ColorFilter.mode(Colors.*white70*, BlendMode.hue),  
 child: Image.network(  
 getYoutubeThumbnail(urlList[index]),  
 fit: BoxFit.fitHeight,  
 ),  
 ),  
 Icon(  
 Icons.*play\_circle\_outline\_sharp*,  
 color: Color.fromRGBO(255, 255, 255, 30),  
 size: 30,  
 )  
 ],  
 ),  
 ),  
 ),  
 );  
 },  
 );  
 }  
}

**fetch subnails:**

String getYoutubeThumbnail(String videoUrl) {  
 String id = videoUrl.substring(videoUrl.length - 11);  
 return 'https://img.youtube.com/vi/$id/0.jpg';  
}

**Home view body :**

import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:save\_knee\_23/features/home/views/widgets/vid\_list.dart';  
import '../../../auth/auth\_cubit/auth\_cubit.dart';  
import 'ExcVidList.dart';  
import 'custom\_home\_view\_app\_bar.dart';  
import 'doctor\_container.dart';  
  
class HomeViewBody extends StatefulWidget {  
 const HomeViewBody({super.key});  
  
 @override  
 State<HomeViewBody> createState() => \_HomeViewBodyState();  
}  
  
class \_HomeViewBodyState extends State<HomeViewBody> {  
 final List<Color> colorList = [  
 const Color(0xffd4a373),  
 const Color(0xffbc6c25),  
 const Color(0xffccd5ae),  
 const Color(0xfff4a261),  
 const Color(0xff99582a),  
 const Color(0xff457b9d),  
 const Color(0xff9e2a2b),  
 const Color(0xff6c584c),  
 ];  
  
 @override  
 Widget build(BuildContext context) {  
 var auth = AuthCubit.*get*(context);  
 double height = MediaQuery.*of*(context).size.height;  
 return BlocBuilder<AuthCubit, AuthState>(  
 buildWhen: (\_, currentState) =>  
 currentState is ImageInitial ||  
 currentState is ImageFailure ||  
 currentState is ImageLoaded ||  
 currentState is DoctorsListLoaded,  
 builder: (\_, state) {  
 return Column(  
 children: [  
 Expanded(  
 child: ListView(  
 physics: const NeverScrollableScrollPhysics(),  
 children: [  
 const CustomHomeViewAppBar(),  
 SizedBox(  
 height: height \* .06,  
 ),  
 VidList(  
 height: MediaQuery.*of*(context).size.height \* .2,  
 label: 'Continue Excercising',  
 hasSeeAllButton: false,  
 tile: ExcVidList()),  
 const Padding(  
 padding: EdgeInsets.only(left: 10, bottom: 7),  
 child: Text(  
 'Choose Your Doctor',  
 textAlign: TextAlign.start,  
 style:  
 TextStyle(fontSize: 20, fontWeight: FontWeight.*bold*),  
 ),  
 ),  
 SizedBox(  
 height: height \* 0.2,  
 child: ListView.builder(  
 scrollDirection: Axis.horizontal,  
 itemCount: auth.doctorsList.length,  
 itemBuilder: (\_, index) {  
 final colorIndex = index % colorList.length;  
 return Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 10),  
 child: DoctorContainer(  
 color: colorList[colorIndex],  
 doctorModel: auth.doctorsList[index],  
 ),  
 );  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 ],  
 );  
 },  
 );  
 }  
}

**vidList :** import 'package:flutter/material.dart';  
  
class VidList extends StatelessWidget {  
 final double height;  
 final String label;  
 final bool hasSeeAllButton;  
 final Widget tile;  
  
 VidList(  
 {required this.height,  
 required this.label,  
 required this.hasSeeAllButton,  
 required this.tile});  
  
 @override  
 Widget build(BuildContext context) {  
 return SizedBox(  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceBetween,  
 children: [  
 Text(  
 label,  
 style: TextStyle(fontWeight: FontWeight.*w600*, fontSize: 20),  
 ),  
 TextButton(  
 onPressed: () {},  
 child: hasSeeAllButton ? Text('see all >') : Text('')),  
 ],  
 ),  
 SizedBox(  
 height: height,  
 child: tile,  
 )  
 ],  
 ),  
 );  
 }  
}

**chat list view :**

import 'package:flutter/material.dart';  
import 'package:save\_knee\_23/features/auth/auth\_cubit/auth\_cubit.dart';  
import 'package:save\_knee\_23/features/home/views/widgets/chat\_list\_View.dart';  
  
import '../view\_models/user\_model.dart';  
  
class ChatListView extends StatelessWidget {  
 const ChatListView({super.key, this.docModel});  
  
 final UserModel? docModel;  
  
 @override  
 Widget build(BuildContext context) {  
 var auth = AuthCubit.*get*(context);  
 return Scaffold(  
 body: ListView.builder(  
 itemCount: auth.doctorsList.length,  
 itemBuilder: (context, index) {  
 return Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 6),  
 child: ChatListItem(docModel: auth.doctorsList[index]),  
 );  
 },  
 ));  
 }  
}

**home view :** import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:icon\_broken/icon\_broken.dart';  
import 'package:save\_knee\_23/features/home/home\_cubit/home\_cubit.dart';  
  
class HomeView extends StatelessWidget {  
 const HomeView({super.key});  
  
 static String *id* = 'homeView';  
  
 @override  
 Widget build(BuildContext context) {  
 final homeCubit = HomeCubit.*get*(context);  
 return BlocBuilder<HomeCubit, HomeState>(  
 buildWhen: (\_, currentState) => currentState is ChangeBottomNavState,  
 builder: (\_, state) {  
 return SafeArea(  
 child: Scaffold(  
 bottomNavigationBar: BottomNavigationBar(  
 currentIndex: homeCubit.currentIndex,  
 onTap: (index) {  
 homeCubit.changeBottomNav(index);  
 },  
 items: const [  
 BottomNavigationBarItem(  
 label: 'Home',  
 icon: Icon(  
 IconBroken.*Home*,  
 ),  
 ),  
 BottomNavigationBarItem(  
 label: 'X-Ray',  
 icon: Icon(IconBroken.*Activity*),  
 ),  
 BottomNavigationBarItem(  
 label: 'Chats',  
 icon: Icon(  
 IconBroken.*Chat*,  
 ),  
 ),  
 ],  
 ),  
 body: homeCubit.screens[homeCubit.currentIndex],  
 ),  
 );  
 },  
 );  
 }  
}

**splash feature :**

import 'package:flutter/material.dart';  
import 'package:go\_router/go\_router.dart';  
import 'package:save\_knee\_23/core/utils/app\_router.dart';  
import 'package:save\_knee\_23/core/utils/assets.dart';  
  
class SplashViewBody extends StatefulWidget {  
 const SplashViewBody({super.key});  
  
 @override  
 State<SplashViewBody> createState() => \_SplashViewBodyState();  
}  
  
class \_SplashViewBodyState extends State<SplashViewBody>  
 with SingleTickerProviderStateMixin {  
 late AnimationController animationController;  
 late Animation<Offset> slidingAnimationLogo;  
  
 @override  
 void initState() {  
 super.initState();  
 initSlidingAnimation();  
 navigatingToHomeView();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 body: Column(  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
 Text('Save Your Knee'),  
 SizedBox(  
 height: MediaQuery.*of*(context).size.height \* .3,  
 ),  
 Center(  
 child: SlideTransition(  
 position: slidingAnimationLogo,  
 child: Image.asset(  
 'assets/images/knee.jpg',  
 fit: BoxFit.cover,  
 ),  
 ),  
 ),  
 const SizedBox(  
 height: 4,  
 ),  
 const Spacer(),  
 const Text('Loading.. please wait patiently'),  
 const CircularProgressIndicator(),  
 const SizedBox(  
 height: 20,  
 )  
 ],  
 ),  
 );  
 }  
  
 void initSlidingAnimation() {  
 animationController = AnimationController(  
 vsync: this,  
 duration: const Duration(seconds: 1),  
 );  
  
 slidingAnimationLogo =  
 Tween<Offset>(begin: const Offset(1, 0), end: Offset.*zero*)  
 .animate(animationController);  
 animationController.forward();  
 slidingAnimationLogo.addListener(() {  
 setState(() {});  
 });  
 }  
  
 void navigatingToHomeView() {  
 Future.delayed(  
 const Duration(seconds: 2),  
 () {  
 GoRouter.*of*(context).push(AppRouter.*kLoginView*);  
 },  
 );  
 }  
}

**x-ray result screen :**

import 'dart:io';  
  
import 'package:flutter/material.dart';  
import 'package:flutter\_screenutil/flutter\_screenutil.dart';  
  
import '../constants.dart';  
import '../widgets/custom\_appbar.dart';  
  
class XRayResultScreen extends StatefulWidget {  
 final File? \_image;  
 final String result;  
  
 const XRayResultScreen(this.\_image, this.result, {super.key});  
  
 @override  
 State<XRayResultScreen> createState() => \_XRayResultScreenState();  
}  
  
class \_XRayResultScreenState extends State<XRayResultScreen> {  
 @override  
 Widget build(BuildContext context) {  
 return SafeArea(  
 child: Scaffold(  
 appBar: PreferredSize(  
 preferredSize: Size.fromHeight(60.h),  
 child: CustomAppBar(  
 title: 'Your Result',  
 onPressed: () => Navigator.*pop*(context),  
 ),  
 ),  
 body: Container(  
 height: MediaQuery.*of*(context).size.height,  
 width: MediaQuery.*of*(context).size.width,  
 alignment: Alignment.*center*,  
 decoration: BoxDecoration(  
 image: DecorationImage(  
 image: AssetImage('assets/images/test\_bg.jpg'),  
 fit: BoxFit.fill),  
 ),  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 crossAxisAlignment: CrossAxisAlignment.center,  
 children: [  
 Spacer(flex: 3),  
 Container(  
 decoration: BoxDecoration(  
 border: Border.all(width: 5, color: kHomeScreenColor),  
 ),  
 child: Image.file(  
 widget.\_image!,  
 width: 180.w,  
 height: 180.h,  
 fit: BoxFit.fill,  
 ),  
 ),  
 Spacer(flex: 2),  
 Container(  
 height: 80.h,  
 width: MediaQuery.*of*(context).size.width \* .75,  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(25.0).w,  
 gradient: LinearGradient(  
 colors: [kHomeScreenColor, Colors.*black*])),  
 clipBehavior: Clip.antiAliasWithSaveLayer,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Text(  
 widget.result,  
 textAlign: TextAlign.center,  
 style: TextStyle(  
 fontSize: 20.sp,  
 color: Colors.*white*,  
 fontWeight: FontWeight.*bold*),  
 ),  
 ),  
 ),  
 Spacer(flex: 1),  
 ],  
 ),  
 ),  
 ),  
 );  
 }  
}

**x-ray screen :**

import 'dart:io';  
  
import 'package:flutter/material.dart';  
import 'package:flutter\_screenutil/flutter\_screenutil.dart';  
import 'package:image\_picker/image\_picker.dart';  
import 'package:save\_knee\_23/widgets/bottom\_sheet.dart';  
  
import '../constants.dart';  
import '../services/xray\_classifier.dart';  
import '../widgets/custom\_appbar.dart';  
import 'xray\_result\_screen.dart';  
  
class XRayScreen extends StatefulWidget {  
 @override  
 State<XRayScreen> createState() => \_XRayScreenState();  
}  
  
class \_XRayScreenState extends State<XRayScreen> {  
 File? \_image;  
 ImagePicker imagePicker = ImagePicker();  
 late String result;  
  
 \_imgFromCamera() async {  
 XFile? pickedFile = await imagePicker.pickImage(source: ImageSource.camera);  
 \_image = File(pickedFile!.path);  
 result = await doImageClassification(\_image);  
 setState(() {  
 \_image;  
 });  
 }  
  
 \_imgFromGallery() async {  
 XFile? pickedFile =  
 await imagePicker.pickImage(source: ImageSource.gallery);  
 \_image = File(pickedFile!.path);  
 result = await doImageClassification(\_image);  
 setState(() {  
 \_image;  
 });  
 }  
  
 @override  
 void initState() {  
 super.initState();  
 loadModelFiles();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return SafeArea(  
 child: Scaffold(  
 appBar: PreferredSize(  
 preferredSize: Size.fromHeight(60.h),  
 child: CustomAppBar(  
 title: 'Your X-ray',  
 onPressed: () => Navigator.*pop*(context),  
 ),  
 ),  
 body: Container(  
 height: MediaQuery.*of*(context).size.height,  
 width: MediaQuery.*of*(context).size.width,  
 alignment: Alignment.*center*,  
 decoration: const BoxDecoration(  
 image: DecorationImage(  
 image: AssetImage('assets/images/test\_bg.jpg'),  
 fit: BoxFit.fill),  
 ),  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 crossAxisAlignment: CrossAxisAlignment.center,  
 children: [  
 const Spacer(flex: 3),  
 \_image == null  
 ? CircleAvatar(  
 radius: 100.r,  
 backgroundColor: kHomeScreenColor,  
 child: CircleAvatar(  
 radius: 95.r,  
 backgroundImage:  
 const AssetImage('assets/images/bg\_circle.png'),  
 ))  
 : Container(  
 decoration: BoxDecoration(  
 border: Border.all(width: 5, color: kHomeScreenColor),  
 ),  
 child: Image.file(  
 \_image!,  
 width: 180.w,  
 height: 180.h,  
 fit: BoxFit.fill,  
 ),  
 ),  
 const Spacer(flex: 2),  
 Container(  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(25.r),  
 gradient: LinearGradient(  
 colors: [kHomeScreenColor, Colors.*black*])),  
 clipBehavior: Clip.antiAliasWithSaveLayer,  
 child: ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: kHomeScreenColor,  
 fixedSize: Size(220.w, 40.h)),  
 onPressed: () {  
 showModalBottomSheet(  
 context: context,  
 builder: (context) => CustomBottomSheet(  
 context,  
 galleryFunction: \_imgFromGallery,  
 // cameraFunction: \_imgFromCamera,  
 ));  
 },  
 child: Text(  
 'Choose X-ray Image',  
 style: TextStyle(color: Colors.*white*, fontSize: 15.sp),  
 )),  
 ),  
 const Spacer(flex: 1),  
 Container(  
 decoration: BoxDecoration(  
 color: kHomeScreenColor,  
 borderRadius: BorderRadius.circular(25.r),  
 ),  
 clipBehavior: Clip.antiAliasWithSaveLayer,  
 child: ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: kHomeScreenColor,  
 fixedSize: Size(100.w, 15.h),  
 ),  
 onPressed: () {  
 \_image == null  
 ? ScaffoldMessenger.*of*(context).showSnackBar(  
 const SnackBar(  
 content: Text("Please upload an image"),  
 ),  
 )  
 : Navigator.*push*(context,  
 MaterialPageRoute(builder: (context) {  
 return XRayResultScreen(\_image, result);  
 }));  
 },  
 child: Text(  
 'Next',  
 style: TextStyle(color: Colors.*white*, fontSize: 20.sp),  
 )),  
 ),  
 const Spacer(flex: 1),  
 ],  
 ),  
 ),  
 ),  
 );  
 }  
}

**constants:**

import 'package:flutter/material.dart';  
import 'package:flutter\_screenutil/flutter\_screenutil.dart';  
import 'dart:ui';  
  
Color kLogoDrk = const Color(0xFF222641);  
Color kLogoLit = const Color(0xFF4942ce);  
const kPrimaryColor = Color(0xff40688F);  
const String kDoctorsCollection = 'doctors';  
const String kMessagesCollection = 'messages';  
Color kHomeScreenColor = const Color(0xff0D235C);  
Color kButtonColor = const Color(0xff3E1E82);  
Color kBackgroundColor = const Color(0xffDBF3FF);  
  
TextStyle kButtonTextStyle = TextStyle(fontSize: 18.sp, color: Colors.*white*);  
  
BoxDecoration kGradientDecoration = const BoxDecoration(  
 gradient: LinearGradient(  
 begin: Alignment.*topRight*,  
 end: Alignment(0, 0),  
 colors: [  
 Color(0xFFfba867),  
 //Color(0xFFff8a65),  
 Color(0xFFc5cae9),  
 // Color(0xFFfba867),  
 Color(0xFF7986cb),  
 Color(0xFFc5cae9),  
 ],  
 ),  
);  
  
BoxDecoration kGradientDecoration2 = const BoxDecoration(  
 gradient: LinearGradient(  
 begin: Alignment.*bottomRight*,  
 end: Alignment.*topLeft*,  
 colors: [  
 Color(0xFFc5cae9),  
 Color(0xffD4CFC8),  
 // Color(0xFF7986cb),  
 Color(0xFFdcf4ff),  
 Color(0xffced5e8),  
 Color(0xFFc5cae9),  
 Color(0xFFc5cae9),  
 ]));  
  
class RPSCustomPainter extends CustomPainter {  
 @override  
 void paint(Canvas canvas, Size size) {  
 Paint paint0 = Paint()  
 ..color = const Color.fromRGBO(255, 255, 255, 1)  
 ..style = PaintingStyle.fill  
 ..strokeWidth = 1.0;  
  
 Path path0 = Path();  
 path0.moveTo(0, 0);  
 path0.lineTo(0, size.height \* 0.6666667);  
 path0.quadraticBezierTo(  
 size.width \* 0.45, size.height, size.width \* 0.7, size.height \* 0.7);  
 path0.quadraticBezierTo(  
 size.width, size.height \* 0.35, size.width \* 2 / 3, 0);  
 path0.lineTo(0, 0);  
 path0.close();  
  
 canvas.drawPath(path0, paint0);  
 }  
  
 @override  
 bool shouldRepaint(covariant CustomPainter oldDelegate) {  
 return true;  
 }  
}  
  
//SCRATCH DATA  
class OldDoctorClass {  
 final String name;  
 final String department;  
 final double rate;  
 final String imgPath;  
 final bool isFav;  
 final int salary;  
  
 OldDoctorClass(  
 {required this.name,  
 required this.department,  
 required this.rate,  
 required this.imgPath,  
 required this.isFav,  
 required this.salary});  
}  
  
List<String> urlList = [  
 'https://youtu.be/ILgVZvYM1WI',  
 'https://youtu.be/U\_WjSAsyh-8',  
 'https://youtu.be/y6NUWq\_AEvI',  
 'https://youtu.be/hXa9QIEGQPg',  
 'https://youtu.be/wN\_\_vnhknJw',  
 'https://youtu.be/raSJqKjjUv0',  
 'https://youtu.be/QPFRYUlXR2I',  
 'https://youtu.be/OexLzyW8tXM',  
 'https://youtu.be/sDz7uY5JdYs',  
 'https://youtu.be/v4U6YE\_BX08',  
 'https://youtu.be/7SpA\_0jLC0g',  
 'https://youtu.be/CCwMevUe7lc',  
];  
  
List<OldDoctorClass> drList = [  
 OldDoctorClass(  
 name: 'Dr. Fillerup Garb',  
 department: 'Medicine Specialist',  
 rate: 3.7,  
 imgPath:  
 'https://as2.ftcdn.net/v2/jpg/02/82/72/13/1000\_F\_282721302\_ASa8MKXhTukl1TnIxGl56eiOQkrvK5zL.jpg',  
 isFav: false,  
 salary: 25),  
 OldDoctorClass(  
 name: 'Dr. Blessing Ross',  
 department: 'Dentist',  
 rate: 4,  
 imgPath:  
 'https://as2.ftcdn.net/v2/jpg/02/69/98/91/1000\_F\_269989145\_sP2OHMejODs0zrlkLWpB2zADAmTwljkT.jpg',  
 isFav: true,  
 salary: 18),  
 OldDoctorClass(  
 name: 'Dr. Crick Bong',  
 department: 'Oncologist',  
 rate: 3.5,  
 imgPath:  
 'https://as2.ftcdn.net/v2/jpg/01/70/97/49/1000\_F\_170974984\_vTtMiYxyEyKgshyQqcdFLjQ0WkQFULEG.jpg',  
 isFav: false,  
 salary: 30),  
 OldDoctorClass(  
 name: 'Dr. Strain Oliver',  
 department: 'Oncologist',  
 rate: 4.8,  
 imgPath:  
 'https://as1.ftcdn.net/v2/jpg/01/32/94/46/1000\_F\_132944601\_TzKWWNgwnBBsShz9UwWiUjTpoydpB5cV.jpg',  
 isFav: true,  
 salary: 35),  
 OldDoctorClass(  
 name: 'Dr. Lachinet Hades',  
 department: 'Medicine Specialist',  
 rate: 2.9,  
 imgPath:  
 'https://as2.ftcdn.net/v2/jpg/02/46/36/57/1000\_F\_246365778\_4X1uXrRyBoFq79wr3flAMv3GDGdSjA6e.jpg',  
 isFav: false,  
 salary: 40),  
];  
  
List<String> availableTimes = [  
 '01:00 PM',  
 '02:00 PM',  
 '03:00 PM',  
 '04:00 PM',  
 '05:00 PM',  
 '06:00 PM',  
 '07:00 PM',  
 '08:00 PM',  
 '09:00 PM',  
 '10:00 PM',  
 '11:00 PM',  
 '12:00 AM',  
 '08:00 AM',  
 '09:00 AM',  
 '10:00 AM',  
 '11:00 AM',  
 '12:00 PM',  
];  
  
List<String> remindTimes = [  
 '10 minutes',  
 '15 minutes',  
 '20 minutes',  
 '25 minutes',  
 '30 minutes',  
 '35 minutes',  
 '40 minutes',  
 '45 minutes',  
 '50 minutes',  
 '55 minutes',  
 '1 hour',  
 '1.5 hour'  
];

**image picker function :**

import 'dart:io';  
  
import 'package:flutter/material.dart';  
import 'package:flutter\_screenutil/flutter\_screenutil.dart';  
import 'package:image\_picker/image\_picker.dart';  
  
import '../constants.dart';  
  
class MyImagePicker extends StatelessWidget {  
 late ImagePicker imagePicker;  
 File? \_image;  
  
 MyImagePicker(this.\_image);  
  
 \_imgFromCamera(File? \_image) async {  
 XFile? pickedFile = await imagePicker.pickImage(source: ImageSource.camera);  
 File image = File(pickedFile!.path);  
 //result = await doImageClassification(\_image);  
 \_image = image;  
 }  
  
 \_imgFromGallery(File? \_image) async {  
 XFile? pickedFile =  
 await imagePicker.pickImage(source: ImageSource.gallery);  
 File image = File(pickedFile!.path);  
 //result = await doImageClassification(\_image);  
 \_image = image;  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Container(  
 height: 200.h,  
 width: MediaQuery.*of*(context).size.width,  
 decoration: BoxDecoration(  
 gradient: LinearGradient(  
 colors: [Color(0xff101010), kHomeScreenColor, Colors.*black*])),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.center,  
 children: [  
 Padding(  
 padding: EdgeInsets.only(top: 12),  
 child: Container(  
 height: 24.h,  
 width: 180.w,  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(25.r),  
 gradient: LinearGradient(colors: [  
 Colors.*black*,  
 Color(0xff0D235C),  
 Colors.*black87* ])),  
 child: Center(  
 child: Text(  
 'Please Choose Image',  
 style: TextStyle(color: Colors.*white*, fontSize: 18.sp),  
 ))),  
 ),  
 Row(  
 children: [  
 Column(  
 children: [  
 InkWell(  
 onTap: () {  
 \_imgFromGallery(\_image);  
 Navigator.*pop*(context);  
 },  
 child: Container(  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(40.0).w,  
 ),  
 clipBehavior: Clip.antiAliasWithSaveLayer,  
 child: Padding(  
 padding: EdgeInsets.only(top: 20, left: 60),  
 child: Image(  
 image: AssetImage('assets/images/gallery.png'),  
 height: 64.h),  
 ),  
 )),  
 Padding(  
 padding: EdgeInsets.only(left: 65, top: 8),  
 child: Text(  
 "Gallery",  
 style: TextStyle(fontSize: 18.sp, color: Colors.*white*),  
 ),  
 ),  
 ],  
 ),  
 Column(  
 children: [  
 InkWell(  
 onTap: () {  
 \_imgFromCamera(\_image);  
 Navigator.*pop*(context);  
 },  
 child: Container(  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(25.0).w,  
 ),  
 clipBehavior: Clip.antiAliasWithSaveLayer,  
 child: Padding(  
 padding: EdgeInsets.only(top: 32, left: 125),  
 child: Image(  
 image: AssetImage('assets/images/camera.jpg'),  
 height: 48),  
 ),  
 )),  
 Padding(  
 padding: EdgeInsets.only(left: 125, top: 16),  
 child: Text(  
 "Camera",  
 style: TextStyle(fontSize: 18.sp, color: Colors.*white*),  
 ),  
 ),  
 ],  
 ),  
 ],  
 ),  
 ],  
 ),  
 );  
 }  
}

**Chapter 4**

UI interface

