

Senior Design Project

CSE499B

‘HealthMate’: Utilizing Artificial Intelligence for Sustaining Fast and accurate Healthcare System

Submitted by:

Name	ID
Md. Shofiul Islam	1712698642
Jaynab Sultana	1721182042
Anika Tahsin Meem	1721810642

Faculty Advisor

Dr. Shahnewaz Siddique

Assistant Professor

Electrical & Computer Engineering, North South University

Summer 2021

DECLARATION

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Students' name & Signature

1. **Md. Shofiul Islam**

2. **Jaynab Sultana**

3. **Anika Tahsin Meem**

APPROVAL

The Senior Design project entitled “HealthMate-Utilizing Artificial Intelligence for Sustaining Fast and Accurate Healthcare System” by Md. Shofiul Islam (ID#1712698642), Jaynab Sultana (ID#1721182042), and Anika Tahsin Meem (ID #1721810642) is approved in partial fulfillment of the requirement of the Degree of Bachelor of Science in Computer Science and Engineering on and has been accepted as satisfactory.

Supervisor’s Signature

Dr. Shahnewaz Siddique

Assistant Professor

Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh.

Department Chair’s Signature

Dr. Rezaul Bari

Associate Professor

Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh.

ACKNOWLEDGMENT

First and foremost, we would want to express our gratitude to the Almighty for providing us with the strength to carry out our duties and complete the report. Dr. Shahnewaz Siddique, a distinguished member of our faculty, has our heartfelt gratitude. We appreciate our supervisor, Dr. Shahnewaz Siddique sir, for his encouragement, inspiration, and timely advice. Our supervisor, Dr. Shahnewaz Siddique sir, has provided us with a lot of encouragement, suggestions, and very constructive criticism, which has really aided us in exploring the idea for our project. This report was created to provide you a hands-on experience while also providing you with a theoretical understanding. We also want to express our heartfelt gratitude to all of the teachers who have helped us complete the project with perfect understanding by giving us with technical expertise and spiritual support. It is critical that we express our gratitude to our respected faculty member Dr. Shahnewaz Siddique for his undivided attention and assistance in achieving this goal. Also, we owe a debt of gratitude to North South University's ECE department for offering us with a course like CSE 499, which allowed us to devote ourselves fully to this project and see it through to completion.

ABSTRACT

The report presents the design and the implementation of an artificial intelligence-based healthcare system “HealthMate”. It is categorized as a web-based application. The application is built for caring patient and predicting some fatal disease. The application will play an important role between doctors and patients. Patients able to meet doctors through video chat and doctor can give prescription to the user through online. The healthcare system will support people with less amount of time. The application will be efficient system to detect disease and support the patient with doctor. Since it is artificial intelligence-based system, it is more efficient than a medical specialist. Machine learning and deep learning algorithm is used for model training and testing. We get encouraging results. Then, the system is built using the model. This system will become reliable and acceptable through it’s service and performance measure. In medical service, it will contribute beside the medical specialist to diagnose and detect simple to large diseases.

Table of Content

		Page No.
	Chapter 1: Introduction	9
1.1	Introduction	10
1.2	Project Details	10
1.3	Project Goals	11
1.4	Background study of the project	11-14
1.5	Summary	14
	Chapter 2: Methodology	15
2.1	Introduction	16
2.2	Methods and Methodology	16-22
2.3	Summary	22
	Chapter 3: Technical Details	23
3.1	Introduction	24
3.2	Problem Statement	24
3.3	Problem with the existing systems	24
3.4	Proposed Solution	24
3.5	Flowchart of the “HealthMate”	25

3.6	The architecture of the “HealthMate”	
3.7	Summary	
	Chapter 4:Result and analysis	
4.1	Introduction	36
4.2	Result	36-47
4.3	Limitations	47
4.4	Discussion	47
4.5	Summary	47
5.1	Introduction	49
5.2	Working plan	49
	Chapter 6: Design Impact	51
6.1	Introduction	52
6.2	Environmental Impact	52
6.3	Economic Impact	52
6.4	Social Impact	52
6.5	Sustainability	52-53
6.6	Income Generation	53
6.7	Benefit	53
6.8	Summary	53
	Chapter 7: Conclusion	54-55
	Bibliography	56-58

	Code	58-84
--	------	-------

List of Figures

Figure No.	Figure caption	Page No.
1	AI in healthcare market size	12
2	AI background	13
3	The process of melanoma detection	17
4	Creating Model of any disease	19
5	DenseNet Architecture	20
6	Inception Module	21
7	Random Forest Architecture	22
8	Flowchart of the Fast and accurate healthcare system	27
9	Architecture of the Fast and accurate healthcare system.	28
10	Home page of the 'HealthMate' system	29
11	Query Submission form.	30
12	Admin login form.	30
13	Admin portal.	31
14	Password change option	31
15	Doctor portal.	32
16	Patient portal.	32-33
17	Online video chat.	34
18	Training and Validation loss for Autoencoder	37
19	Training and validation.	38
20	Pneumonia Accuracy	39
21	Pneumonia loss.	39
22	Malaria Accuracy	40
23	Malaria loss graph.	41
24	Confusion matrix Table	42
25	Training and Validation loss	42
26	Training and Validation accuracy	42
27	Melanoma Accuracy	43
28	Melanoma loss	43
29	Confusion matrix Table	44

30	Breast Cancer Accuracy	45
31	Breast Cancer Accuracy	45
32	The working plan of our project part-1.	50
33	The working plan of our project part-2.	51

CHAPTER 1

Introduction

1.1 Introduction

“Utilizing Artificial Intelligence for Sustaining Fast and Accurate Healthcare System” is a computer program that will be classified as a web-based system that will forecast simple to complex diseases based on image processing and symptoms and prescribe the users via the internet. The major goal of the system is to allow consumers and diagnostic centers to use the online system to avoid diagnosing errors that require too much time and money. The system is designed to be user-friendly, allowing even those with limited technical knowledge to utilize it. Long-distance people will profit from the system since they won't have to waste time traveling to town or waiting for a long-distance medical diagnosis. They can find out what their disease's prognosis is through the healthcare system.

1.2 Project Details

In the “Fast and accurate healthcare system,” there are three sorts of user modules:

1. Admin

The administrator's task is to keep an eye on the entire system and manage it. The admin can control many other things of the system. They are allowing-

- To make any type of changes through the whole system
- To control the user access
- To add or delete or update the user, disease, appointment.

6. Doctors Profile Module

There will be certain tasks in the case of the doctor profile module:

- i. Fixing/editing/updating the appointment schedule;
- ii. Uploading personal photos with accurate doctor information; and monitoring and prescribing patients via the internet.
- iii. By using his or her profile in the system, the doctor can provide information to the patient and communicate with him or her.
- iv. Can be able to check whether he has an appointment or appointment request.
- v. Using log out option Doctors can exit from this module.

7. Patient Profile Module

In the system, the patient module is responsive. The system's primary focus is on them. They can create their account in the system. They can make an appointment request online. They can check the disease by uploading their skin lesion or photographs of x-ray abnormalities. After completing his task, the user can log off of the system.

1.3 Project Goals

With our project, we aim to focus on a few primary objectives. The following are the details:

- i. The system for real-time detection:** The primary goal is to detect any disease early and thoroughly. The device can assist patients in detecting disorders and connecting with doctors in real-time for proper monitoring.
- ii. Accuracy:** The system will be able to forecast disease with high accuracy utilizing a convolutional neural networking model based on deep image analysis. Our main goal is to acquire this level of precision. To achieve and compare high accuracy to diagnose disease, different models such as Inceptionv3, ResNet50, DenseNet121, VGG16, and VGG19 are utilized.
- iii. Avoid wasting time and money:** Another goal is to save time and money. To save time and money, patients can schedule appointments with medical specialists online.
- iv. Simple User Interface:** The system has been designed to be basic to use. The user interface has been maintained simply without being overly complicated.

1.4 Background study of the project

Patients', doctors', and hospital managers' lives are made easier by artificial intelligence, which performs activities normally performed by people in a fraction of the time and at a fraction of the expense. The AI sector, which was valued at around \$600 million in 2014 and is expected to reach \$150 billion by 2026, is one of the world's fastest-growing sectors [1]. In the modern healthcare sector, artificial intelligence (AI) is the most powerful technology. The latest successful applications of artificial intelligence (AI) in the healthcare sector have reached their full potential due to the increasingly increasing accessibility of healthcare medical data and the advancements of big data diagnostic techniques. Potential artificial intelligence (AI) strategies can disengage healthcare-appropriate knowledge hidden in the massive amount of data with the aid of essential medical queries, which can maintain healthcare decision-making. Modern healthcare technology has spread to many innovative startups around the world, allowing people to live healthier and longer lives. The advancements were initially driven by the advent of mobility and software, which enabled the health sector to

digitize many pen-and-paper-based processes and operations that are currently slowing service delivery. Computer software has become much more sophisticated and autonomous in recent years. Machine learning (ML) and artificial intelligence (AI) are used to discuss these new skills. These emerging abilities are addressed under the umbrellas of machine learning (ML) and artificial intelligence (AI), both of which are speeding up the pace of healthcare improvement. Machine learning (ML) and artificial intelligence (AI) applications in healthcare have enabled the industry to address some of the most pressing issues in specific domains.

The use of machine-learning algorithms and software, or artificial intelligence (AI), to simulate human cognition in the study, presentation, and comprehension of complex medical and healthcare data is referred to as artificial intelligence in healthcare. Fig 1 shows U.S. AI background in healthcare market size, from 2016 to 2028.

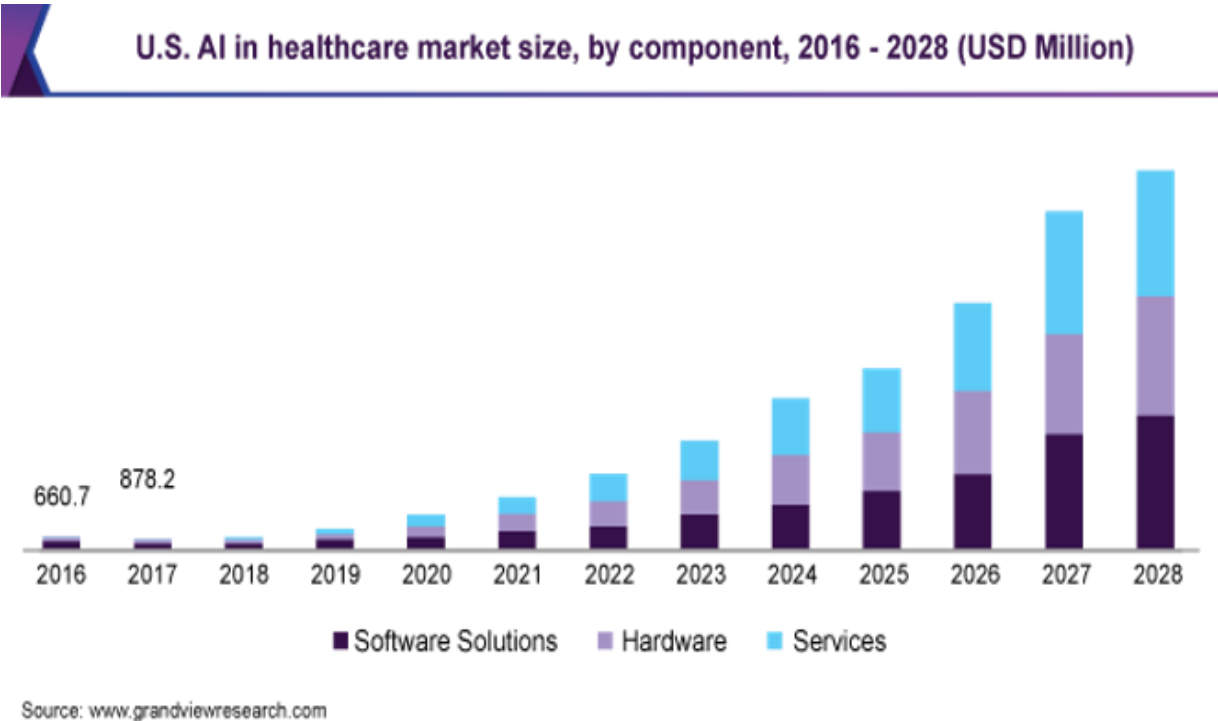


Figure 1: AI in healthcare market size

From this figure 1, we can see that it is growing at a mass rate. There is 3 sections AI is divided into, which is Software solutions, Hardware, and Services. Fig 2 shows the AI background.

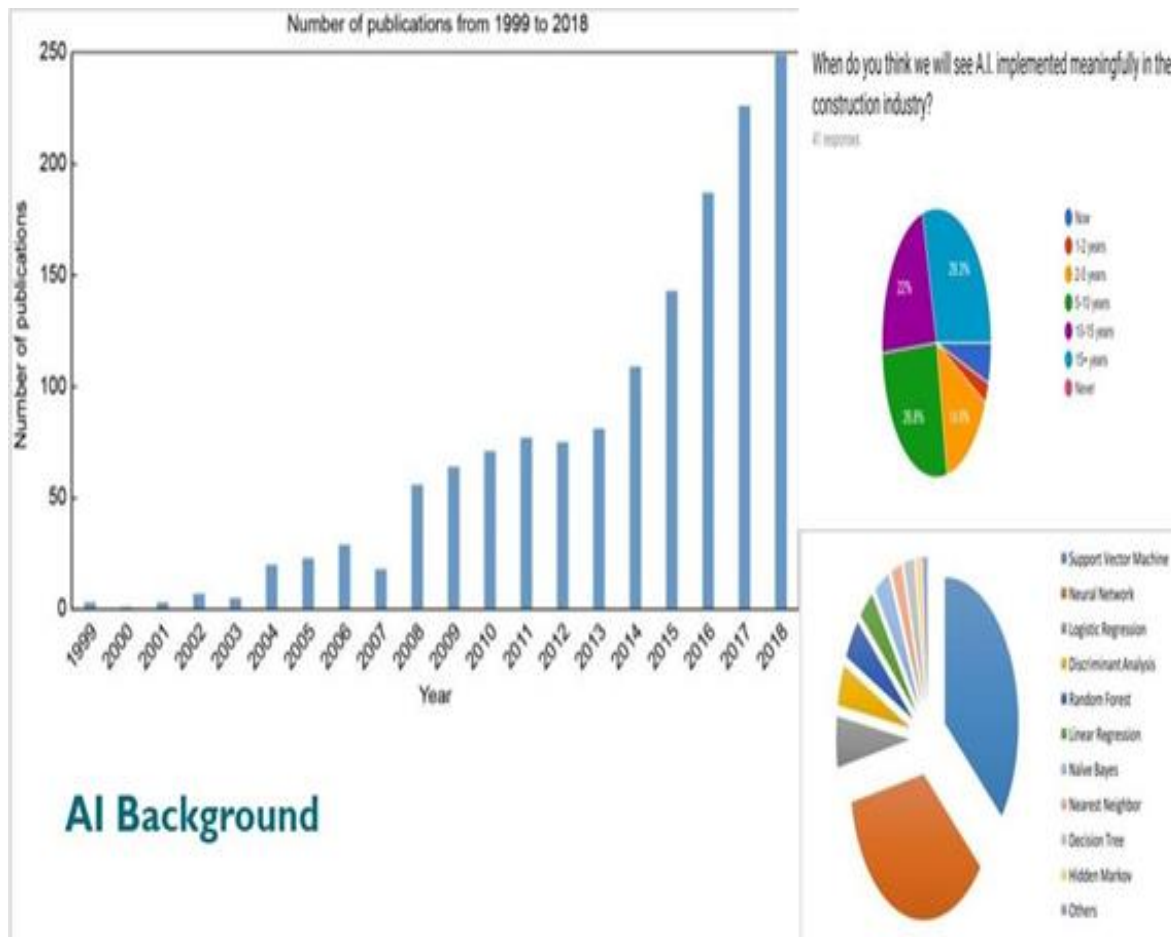


Figure 2: AI Background

Here using AI, there are multiple publications from 1999 to 2018. As the demand growing and simultaneously the publications. Also, the different types of model applying rate increased. We can see from the figures. Supervised learning produces more clinically relevant findings than unsupervised learning; thus, supervised learning is used most frequently in AI applications in healthcare. (It's worth noting that unsupervised learning can be utilized in the preprocessing stage to reduce dimensionality or identify subgroups, making the subsequent supervised learning phase more efficient.) Linear regression, logistic regression, naive Bayes, decision tree, closest neighbor, random forest, discriminant analysis, support vector machine (SVM), and neural network are examples of relevant approaches. The prevalence of various supervised learning approaches in medical applications is seen in Figure 2, with SVM and neural networks being the most common.

AI is described as the ability of computer algorithms to make educated guesses based solely on input data. General, the first problem-solving software or expert method, was developed during research in the 1960s and

1970s. It was planned for organic chemistry applications, but it served as the foundation for a later scheme. Medical records mining [2], robotics-assisted surgeries [3], medical management and supporting hospital operations, clinical data interpretation [4], clinical trial participation [5,6], image-based diagnosis [7], preliminary diagnosis [8], virtual nursing [9], and connected health care devices [10] have all shown significant promise. In addition to these uses, current initiatives to regulate the use of AI in the medical domain [11] and large investments in AI research [12] suggest that AI could become a crucial tool to help decision-making in the medical domain soon. M. M. Kamruzzaman is employed with Architecture of Smart Health Care System Using Artificial Intelligence [13]. PathAI, which was co-founded by Khosla and CEO Andy Beck in 2016, uses computer vision and deep learning to detect and diagnose cancer. PathAI, in particular, employs convolutional neural networks (CNNs) in the field of pathology. Buoy Health is a symptom checker powered by artificial intelligence. Artificial Intelligence is being used by BETH ISRAEL DEACONESS MEDICAL CENTER to diagnose potentially fatal diseases. Zebra Medical Vision is an AI-enabled helper for radiologists that receives imaging images and analyzes them automatically for various clinical abnormalities it has studied. The results are forwarded to radiologists, who evaluate the assistant's reports when establishing a diagnosis [14]. HAMS supports the OASIS EDXL-HAVE standard, which allows for the open and interoperable sharing of data [18]. MYCIN is regarded as one of the most important early applications of artificial intelligence in medicine. MYCIN, as well as other systems like INTERNIST-1 and CASNET, were unable to achieve this goal. MYCIN and other systems such as INTERNIST- 1 and CASNET, on the other hand, were not widely used by clinicians. The microcomputer and new levels of network access proliferated in the 1980s and 1990s [15]. Another one, in 2011 by Aao Ni, his goal is to generate a functioning prototype that has the potential to improve in-clinic doctor-patient communication, with a focus on information exchange. Demonstrate how projection, as well as picture and video capture, can be used to support critical communication activities during physical therapy visits in a flexible way. AnatOnMe adds to the overall agenda by enabling a vital component of effective clinical information exchange [16]. During this time, researchers and developers realized that AI systems in healthcare needed to be built to accommodate the lack of perfect data and rely on physicians' expertise. Fuzzy set theory, Bayesian networks, and artificial neural networks have all been used in the development of intelligent computing systems in healthcare.

1.5 Summary

This chapter gave us insight into the modules that we have in the proposed system. This chapter provided a clear picture of how this system is effective to use in diagnostic centers with the help of the elaboration of the project goals.

CHAPTER 2

Methodology

2.1 Introduction

In this chapter, we discuss the Methodology which we need to implement the system

2.2 Methods and Methodology

In figure 3, we represent our work through a block diagram of the system and an overview of the model of convolutional neural networking. Here, we can see that inceptionv3 and vgg16 model are consisted of ImageDataGenerator, Inception V3/VGG16 model, Average_pooling2d, Batch_normalization_273, conv2d, Activation_275, Dense_2, Flatten_2. A deep learning algorithm and machine learning algorithm are used to predict diseases. The methods we follow in this “Fast and healthcare system” are given below-

- 1) We collect a dataset from Kaggle which is used in this project.
- 2) After collecting the dataset, we get noise and the different sizes of images. So, we preprocess the data for removing noise and resize images.
- 3) Then we used a convolutional neural networking model (VGG16, VGG19, INCEPTIONV3, ResNET50, DenseNet121) or machine learning.
- 4) The preprocessed data are used for training and testing with the CNN model.
- 5) Then we check the accuracy.

We use categorical cross-entropy, Adam optimizer, SoftMax classifier.

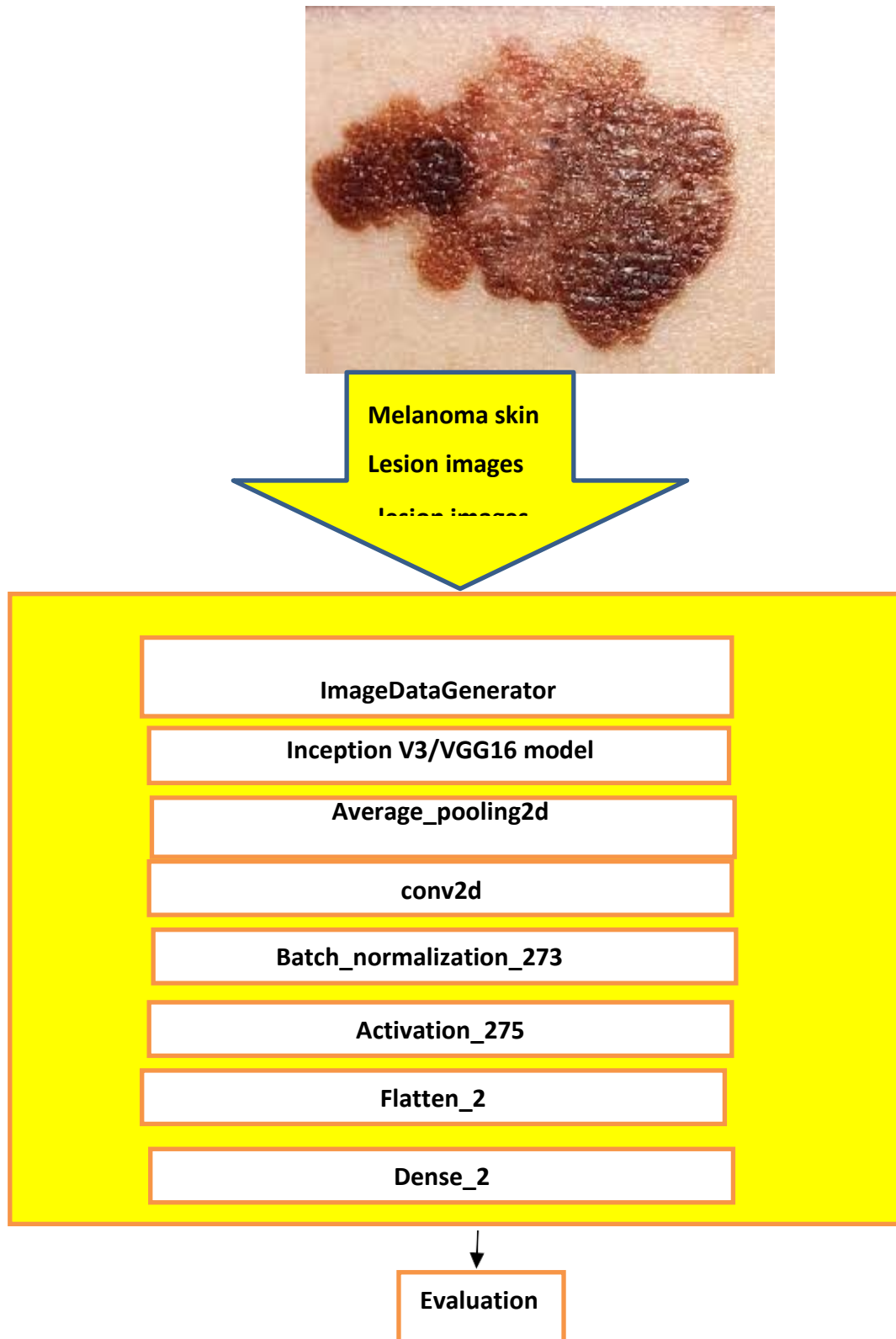


Figure 3: The process of melanoma detection.

In the coding part, the batch size was 32, experimental runtime was GPU. We use 10 epochs, 50 epochs in our coding in the case of the hypermeter. This section contains the methods and materials parts that are adapted to implement the goal of the system. This system aims to predict multiple diseases like Pneumonia, Covid19, Cancer, Diabetes, Melanoma, and so on. Using or uploading their particular diseases X-ray images or MRI images to our system any patient or user can detect their diseases in their primary stage. AI can 'learn' features from a massive volume of healthcare data using complex algorithms, and then use the findings to aid clinical practice. It could also have learning and self-correcting capabilities to enhance accuracy depending on the input. Physicians can benefit from AI systems that provide up-to-date medical information from journals, textbooks, and clinical practices to help them provide effective patient care. Furthermore, an AI system can aid in the reduction of diagnostic and treatment errors, which are unavoidable in human clinical practice. Furthermore, an AI system collects usable data from a huge patient population to aid in developing real-time conclusions for health risk alerts and prediction. For creating any kind of diseases model, we collected a dataset of particular diseases like images or CSV data. Here for creating any kind of off-model we used different types of models like Random Forest, Decision Tree, SVM, CNN, Resnet, Inception, Autoencoder, etc. Data preprocessing will be getting ready for arranging the rough data and making it suitable for a machine learning and deep learning illustrate. It is the essential and crucial step though making a machine learning illustrate. Then data augmentation, data augmentation methods in CNN utilizing Tensorflow and Keras. But a few times as of late any procedure: Images resizing. The preeminent commonly utilized image broadening strategies with code cases and representation of pictures after extension. From here onwards, data will be alluded to like pictures. We are going to be utilizing Tensorflow or OpenCV composed in Python in all our cases. After that, the feature is extracted. In machine learning and measurements, classification could be an administered learning approach in which the computer program learns from the input information and make the show which has been utilized for evaluate the performance, result from analysis. Figure 4 shows the creating model procedure for a particular disease:

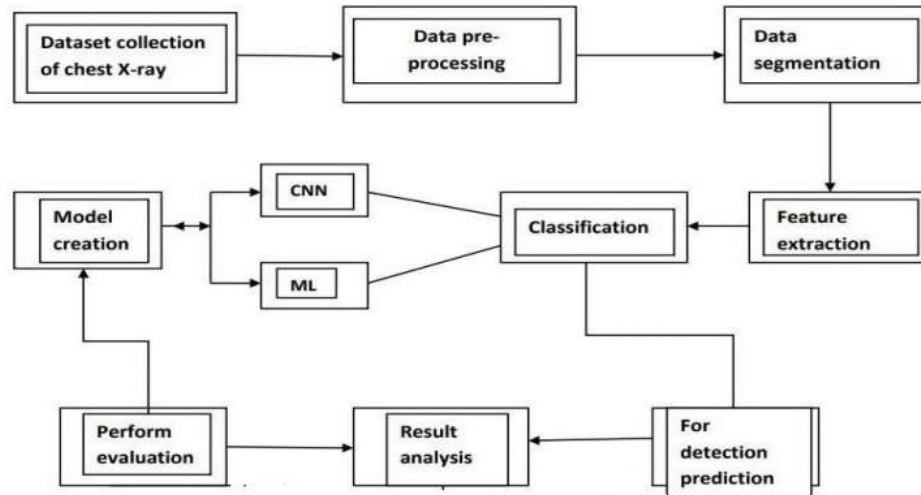


Figure 4: Creating Model of any disease

This image processing technique helps in object detection based on the color, size, and shape of images. Different benchmark CNN models have been embraced in our proposed work. They have been trained individually to make independent predictions. Then the models are combined, using the new method of weighted average assembling technique, to predict a class value. This modern proposed assembling strategy is anticipated to create the expectation more robust. Our proposed work comprises of three pre-trained model Autoencoder and CNN models such as DenseNet, Resnet50, and Inception model. We have used Keras and TensorFlow based on given parameters to train the model. At that point run the prepared models on the test pictures and select lesson name 0 or 1 based on the weighted normal gathering of the 3 models. Whereas partitioning the pictures into preparing and testing, guarantee that there's no persistent cover i.e., distinctive pictures of the same quiet isn't display in both preparing and testing datasets.

DenseNet Architecture

First of all, a DenseNet's convolution creates a greater number of highlight maps. The number of yields includes maps of a layer is characterized as the development rate. DenseNet has lower

require of wide layers since as layers are thickly associated there's little excess within the learned highlights. Fig 4 shows the architecture of the CNN model DenseNet.

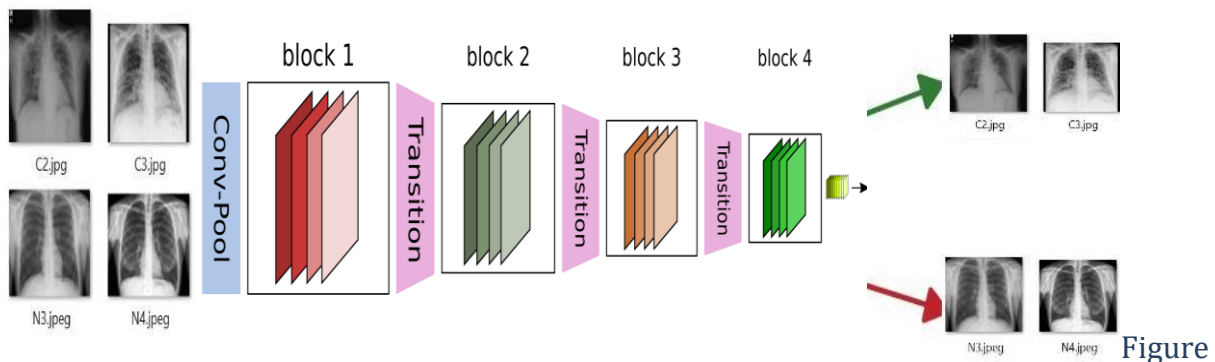


Figure 5: DenseNet Architecture

Densely connected Convolutional Systems, DenseNets, are the following step on the way to keep expanding the profundity of profound convolutional systems. Conventional feed-forward neural systems interface the yield of the layer to the following layer after applying a composite of operations. We have as of now seen that ordinarily, this composite incorporates a convolution operation or pooling layers, a group normalization, and an activation work. DenseNets are separated into Dense Blocks, where the measurements of the highlight maps remain steady inside a piece, but the number of channels changes between them. These layers between them are called Move Layers and pay attention to the downsampling applying a bunch normalization, a 1x1 convolution, and 2x2 pooling layers. Within the new deeper level speaking to the primary Thick Layer inside the primary Thick Square, able to see how this behavior of including 32 times the number of layers is accomplished. We perform as the creators propose a 1x1 convolution with 128 filters to diminish the highlight maps estimate and perform a more costly 3x3 convolution (keep in mind to incorporate the cushioning to guarantee the measurements stay steady) with this chosen 32 number of include maps of development rate. Then, the input volume and the result of the two operations (which are the same for each Thick Layer inside each Thick Piece) are concatenated.

Inception Module:

The paper proposes a modern sort of engineering – Google Net or Initiation v1. It is essentially a convolutional neural organize (CNN) which is 27 layers profound. 1×1 Convolutional layers sometimes recently applying another layer, which is primarily utilized for dimensionality reduction. Fig 5 shows the architecture of the CNN model Inception Module.

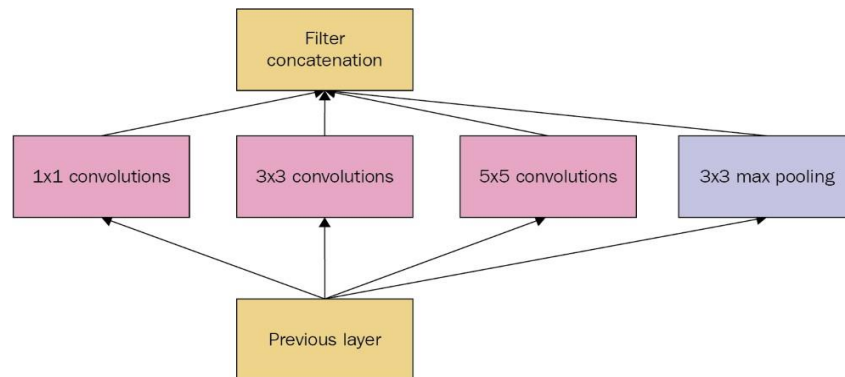


Figure 6: Inception Module

Beginning Modules are consolidated into convolutional neural systems (CNNs) as a way of diminishing computational cost. As a neural net bargain with an endless cluster of pictures, with wide variety within the included picture substance, moreover known as the salient parts, they have to be outlined fittingly. The foremost disentangled adaptation of a beginning module works by performing a convolution on input with not one, but three distinctive sizes of channels (1×1 , 3×3 , 5×5). Too, max pooling is performed. At that point, the coming about yields are concatenated and sent to the following layer. By organizing the CNN to perform its convolutions on the same level, the arrangement gets dynamically more extensive, not more profound.

Random Forest Architecture

The machine learning algorithm we used was the Random Forest algorithm. Random Forest is a versatile and user-friendly software technique that produces excellent results without the use of super parameters in the majority of cases. Fig 6 shows the architecture of the Random Forest Module.

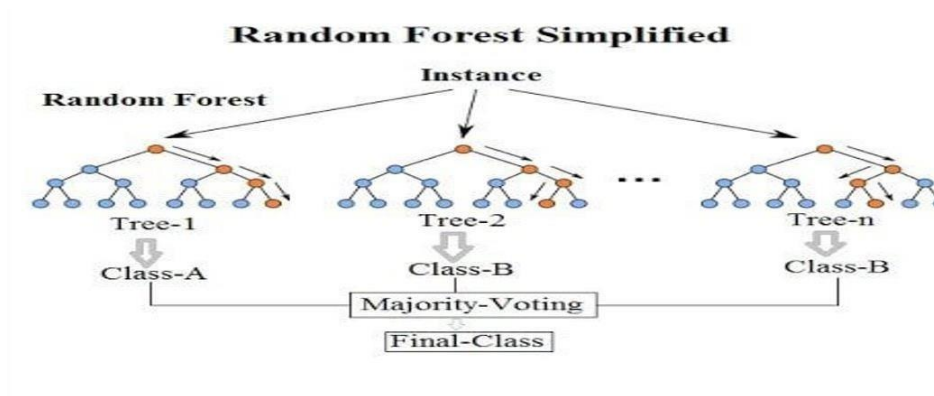


Figure 7: Random Forest Architecture

It's simple to use and can be used to classify and predict data. Random Forest is a learning algorithm that is supervised. To put it simply: The random tree generates and merges a variety of options.

2.3 Summary

In this chapter, we get an idea about the methods and methodology.

CHAPTER 3

Technical Details

3.1 Introduction

In this chapter, we will discuss the working flow diagram, technical design, system architecture of “HealthMate”. This section represents the problem with the existing similar system, proposed solution.

3.2 Problem statement

Bangladesh's total population is expected to reach 164.69 million by 2020. This is beyond the healthcare system's capacity. In Bangladesh, bringing a huge population under healthcare is a major difficulty. Furthermore, many people in this large population die as a result of diseases that are not discovered early enough. Bangladesh's healthcare system is completely unregulated. The difficulty is also caused by political unpredictability and a lack of a well-defined regulatory mechanism. Another issue in hospitals is an unmanageable patient load. Because of a lack of sufficient information about modern machines and equipment, diagnose errors occur. An ineffective governance framework; Another issue in the healthcare system is poor management and institutional capacity in the Ministry of Health and Family Welfare, as well as inefficient distribution of public resources.

3.3 Problem with the existing systems

Doctors use tests to identify the lesion. X-rays, CT scans, and positron emission tomography (PET) scans are used to diagnose melanoma. X-rays emit a lot of radiation. As a result, it has negative consequences on our bodies, such as vomiting, bleeding, fainting, and hair loss. Abdominal discomfort, diarrhea, nausea or vomiting, and constipation are all possible side effects of a CT scan. Nausea, vomiting, headaches, itching, flushing, allergic reactions, and a minor rash are all symptoms of a PET scan. So, these are the drawbacks of the current systems. The existing systems take a long time to operate. People must also wait for the results of the investigation. These are intricate systems. As a result, they desire a simpler system. We can also observe the present paper-based system, which takes a long time to set up and is difficult to save and retrieve data.

3.4 Proposed Solution

Our recommended approach is to employ a web-based system called "Fast and Accurate Healthcare System." This method allows doctors and patients to communicate online. This system requires less time. The patient can meet with a doctor and can share their problem through video chat. They can get prescriptions online. The patient can check diseases by uploading lesion pictures.

3.5 Flowchart of the “HealthMate”

In figure 8, the flowchart illustrates the whole process of the healthcare system. Before login “HealthMate”, any user must register first. The next job is to visit the website. The whole working steps for designing the site is given below:

Initial Task: First, We create a Django app using `django-admin`. (Django-admin startproject Smarthospital). Then, inside the Django project, we have created a Django app name hospital by using the “`python manage.py startapp hospital`” command. After that, we need an admin to maintain all apps we have inside our Django project. So we created an admin using the “`python manage.py createsuperuser`” command. After executing this command it will ask for some pieces of information like admin name and password.

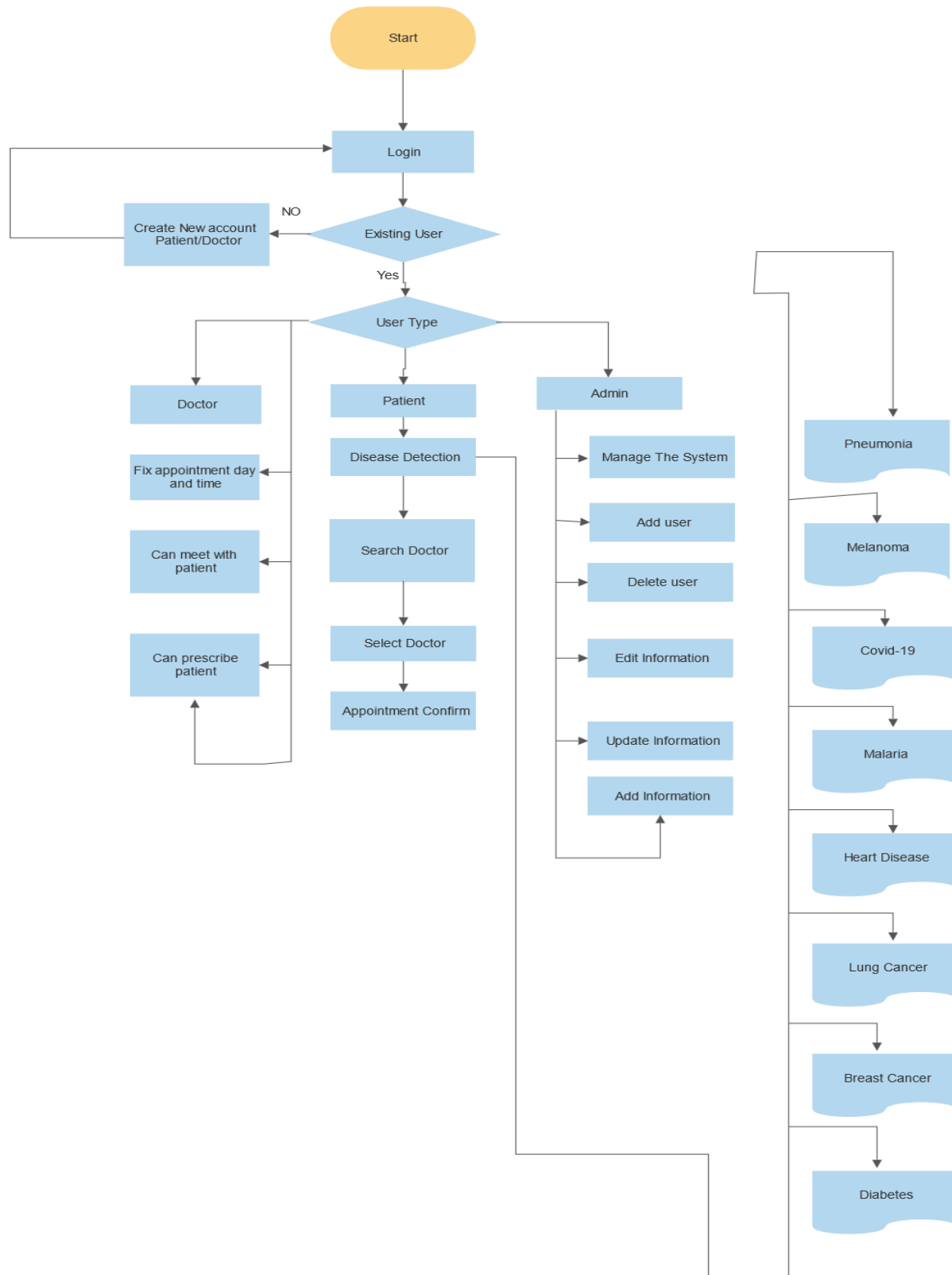
After Implementing all functionalities: If a user wants to use our system, he/she needs an account in our system. To get an account he/she needs to go to the Sign-Up section where he/she will get a form to enter his/her information such as name, email, password. If a doctor wants to create an account in our system, the doctor also needs to give his/her basic information like a general user, in addition to that the doctor must give the hospital name where he is currently working. After creating an account in our system, a user/doctor can now log in to our system by using his/her email and password. After login, the person will redirect to the patient dashboard if he/she login as a patient, or to the doctor dashboard if the user login as a doctor.

Patient Dashboard: Patient dashboard will allow the patient to make an appointment with a doctor by clicking on the “Make an appointment” button. After clicking on the “Make an appointment” button patient will see a form where he/she can select a doctor from the doctor’s list. Also, This form will allow the patient to pick the time when he/she want to consult with the doctor. Patient dashboard also has the disease prediction button. If a patient click on the “Disease pradiction” button, the patient will redirect to a form where he/she can select the disease want to predict, and after giving the infected image if the user press on submit it will give the result of how much percent our system sure that that person is infected or not.

Doctor Dashboard: After login as a doctor, the user will see the doctor dashboard. On the right side of the doctor dashboard, the doctor will see appointment requests and appointment lists with the patient's name and time. From the appointment requests, a doctor can accept or reject appointment requests. If the doctor rejects the request it will remove from the appointment request list, if he/she accepts the request, the request will add to the appointments list with the chat option enable. The doctor dashboard also has the “Disease Prediction” button that allows the doctor to determine the patient’s condition.

Admin Dashboard: We add admin using Django “createsuperuser” option so that our system allows the admin to access all the information the system has. If a person login as an admin he/she will redirect to the Django admin dashboard where he/she can add a new doctor, patient, and another admin. The admin will see all the databases our system has

Figure 8: Flowchart of the Fast and accurate healthcare system



3.6 The architecture of the “HealthMate”

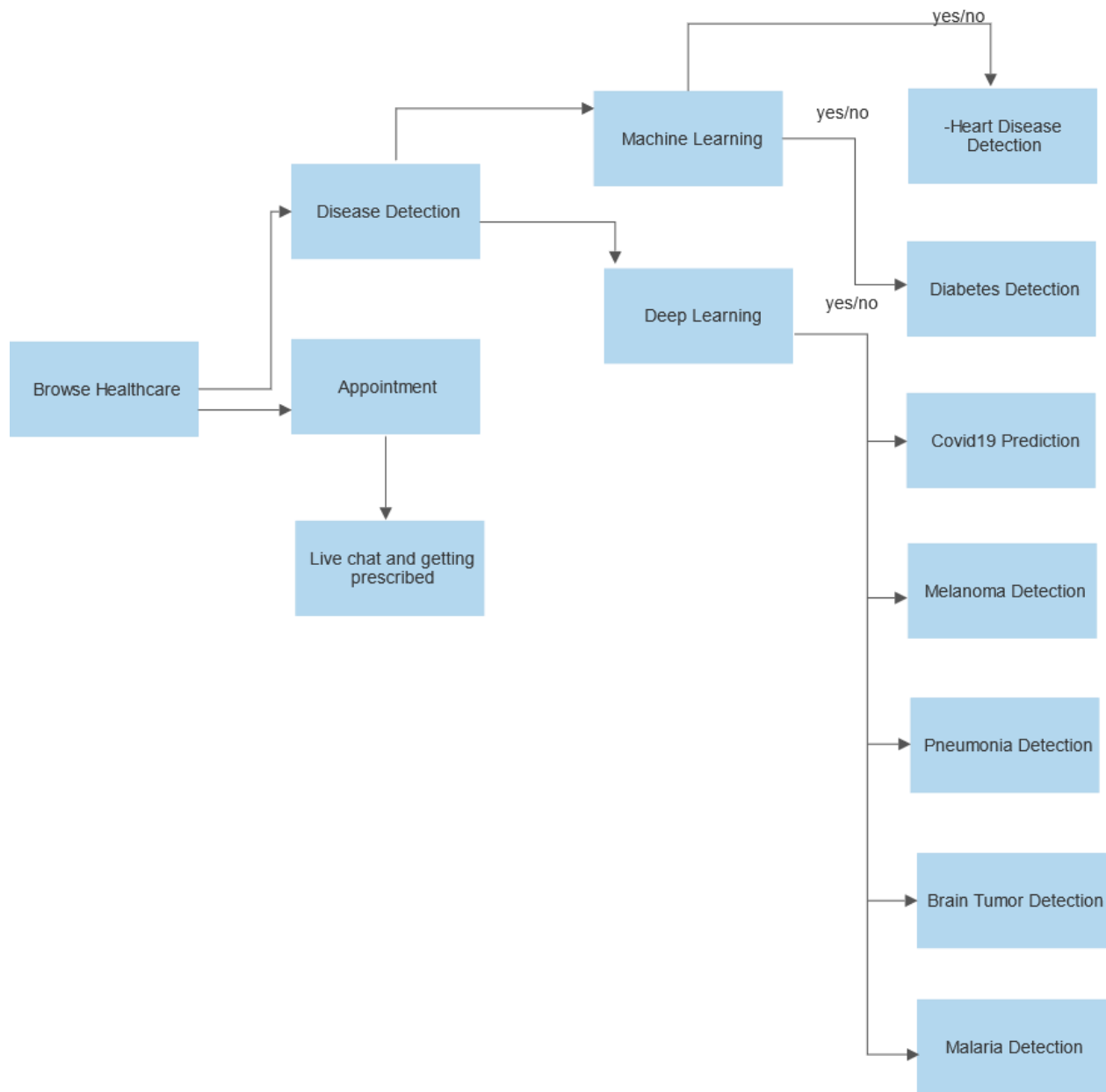


Figure 9: Architecture of the Fast and accurate healthcare system.

In figure 7, after browsing for the system, we get two options-1) disease detection, 2)appointment then we use machine learning or deep learning model to detect disease. A Fast

and accurate Healthcare System is a computer program that enables users to predict a simple disease to large disease through image processing and symptoms and prescribed the users through online. Our Healthcare System are typically designed to: Connect doctors and patients through online, Predict and diagnose disease with high accuracy, Guide the patient to correct care based on the diagnosis and prediction. For creating the model, we used machine learning, deep learning, neural network.

System Framework

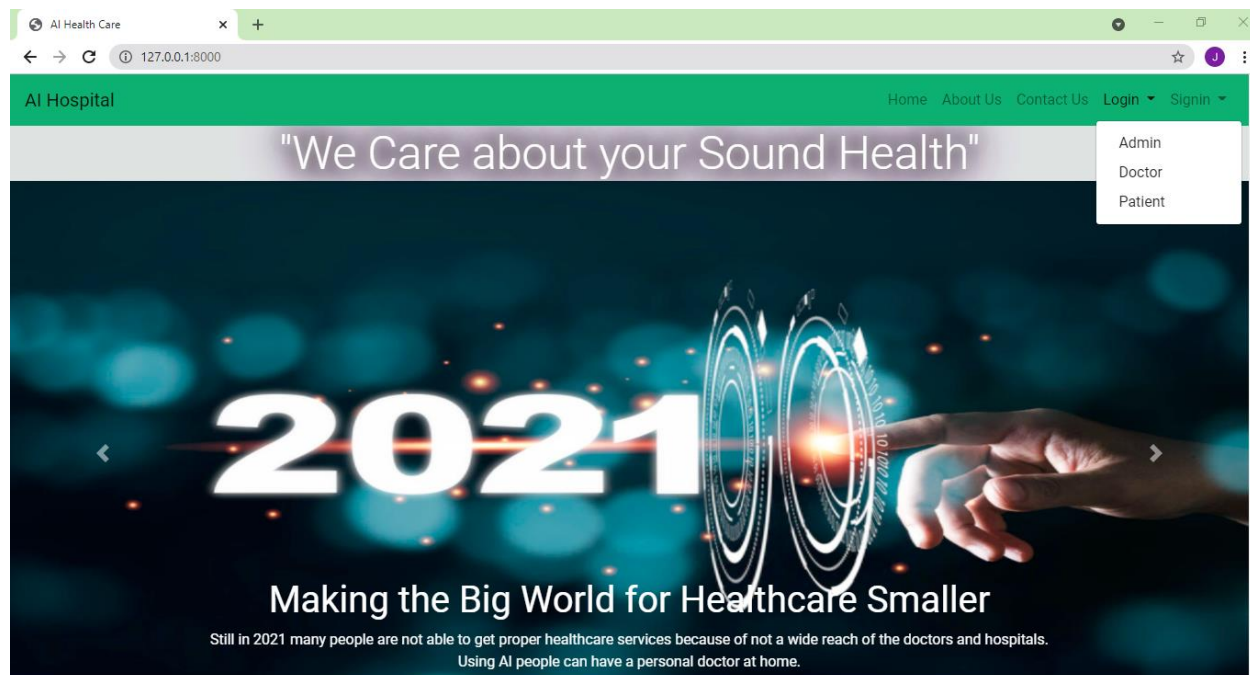


Figure 10: Home page of the 'HealthMate' system.

Figure 10 represents the homepage of our healthcare system. The homepage is designed in such a way that it is easy to see what diseases can be detected through this site. When we scroll down, we will see the diseases including here. In the navigation bar we can the options-“Home, About us, Contact us, log in, Sign In”. There are three modules. These three modules will get access through the login option; before login, the doctor and patient need to sign in.

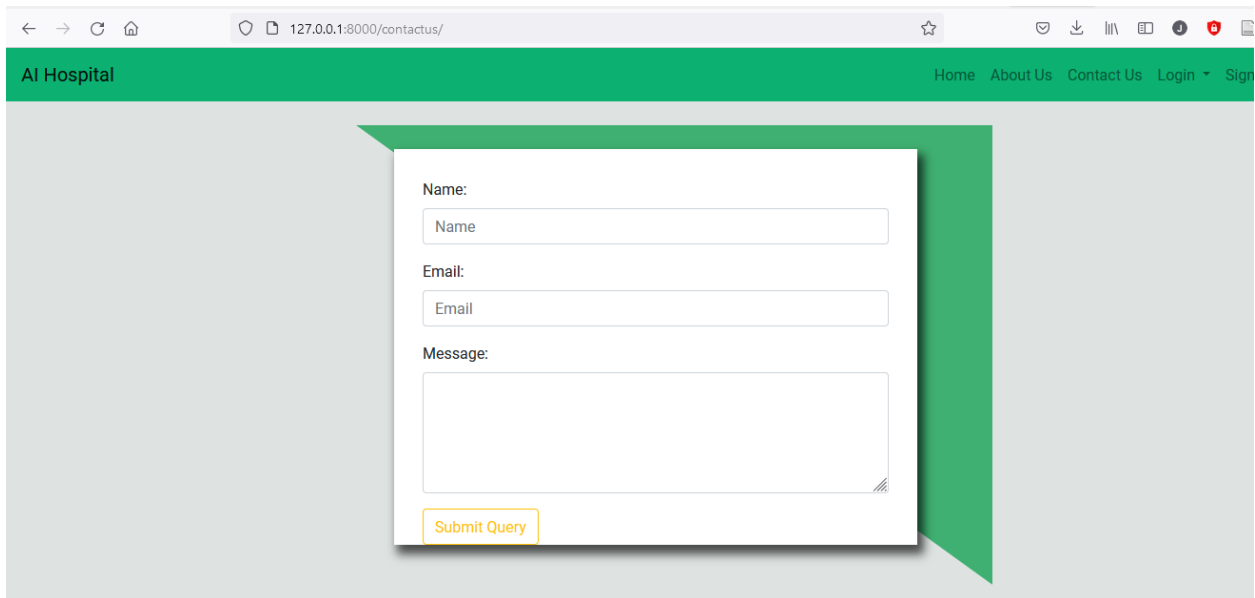
A screenshot of a web browser displaying the 'AI Hospital' contact form. The browser's address bar shows '127.0.0.1:8000/contactus/'. The page has a green header with 'AI Hospital' on the left and navigation links 'Home', 'About Us', 'Contact Us', 'Login', and 'Sign' on the right. The main content area is light gray. A white form box is centered, containing three input fields: 'Name' (with placeholder text 'Name'), 'Email' (with placeholder text 'Email'), and 'Message' (a larger text area). Below these fields is an orange 'Submit Query' button.

Figure 11: Query Submission form.

Whether the patient face any problem or he/she has any queries or suggestion, then using his/her name or email address, he/she can write messages and submit it to the system. Admin can see the message.

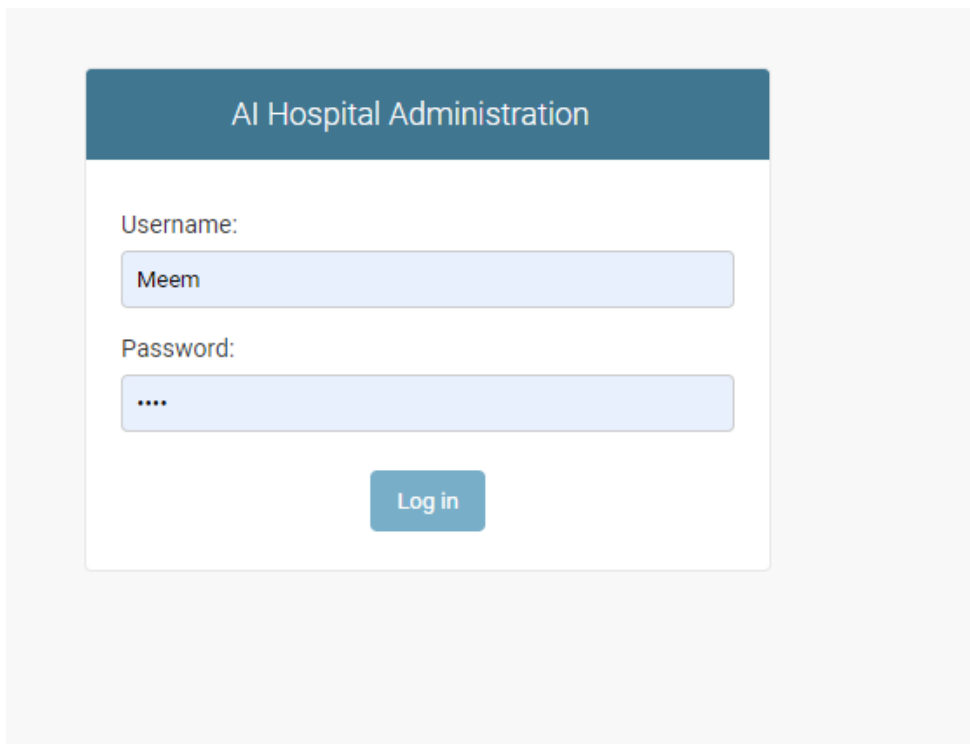
A screenshot of the 'AI Hospital Administration' login form. The form has a dark blue header with the text 'AI Hospital Administration'. Below the header, there are two input fields: 'Username' (containing the text 'Meem') and 'Password' (containing four dots). Below these fields is a blue 'Log in' button.

Figure 12: Admin login form.

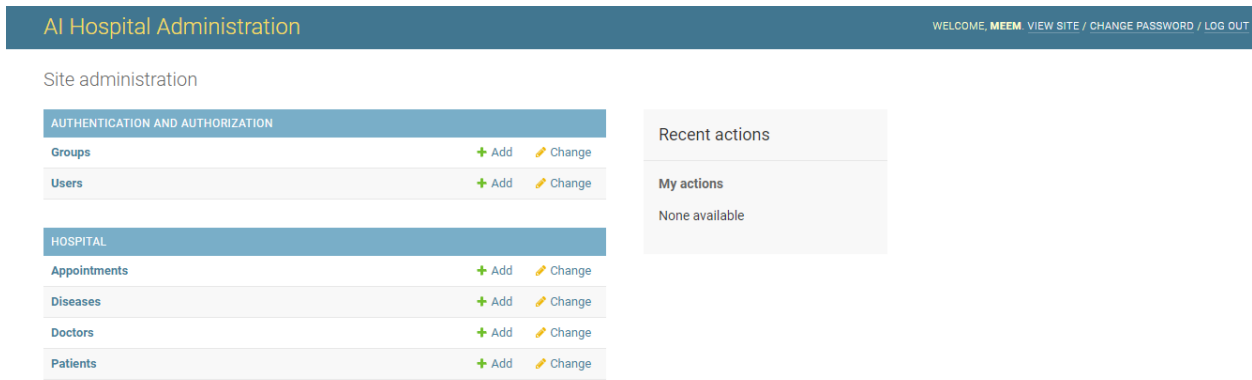


Figure 13: Admin portal.

Admin needs to give username and password to log in to the admin portal. After login admin can add or change users, diseases, appointments. He can see the user detail from the database. Figure12 presents the admin login form and Figure 3 represents the admin portal.

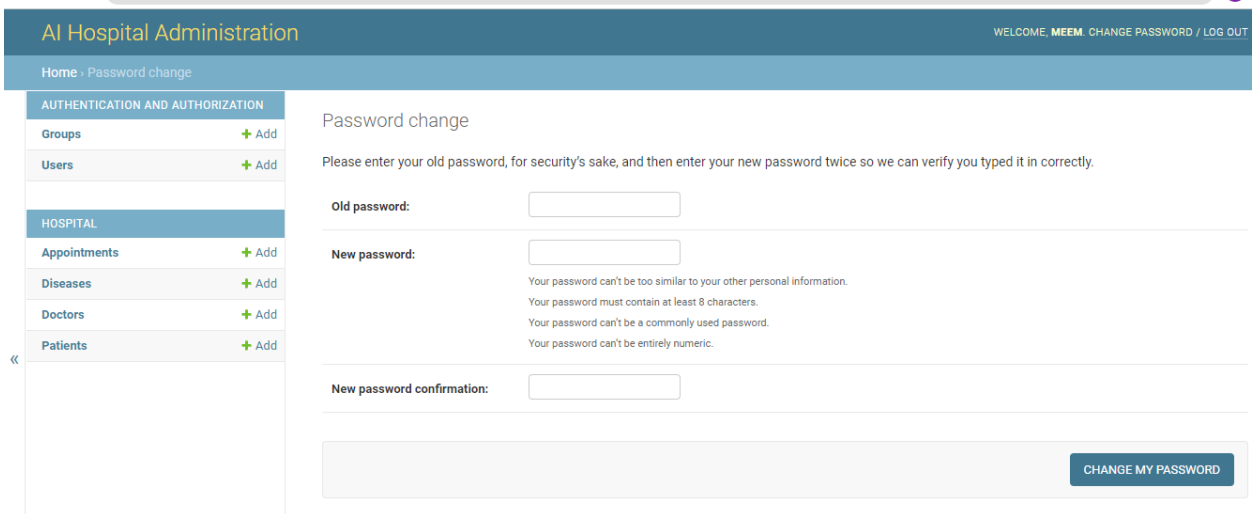


Figure 14: Password change option.

Figure 14 shows that the admin can change the password. In the above right corner, we see the “CHANGE PASSWORD” option. Admin will be able to change his password by using the option and he can exit from this page by logging out.

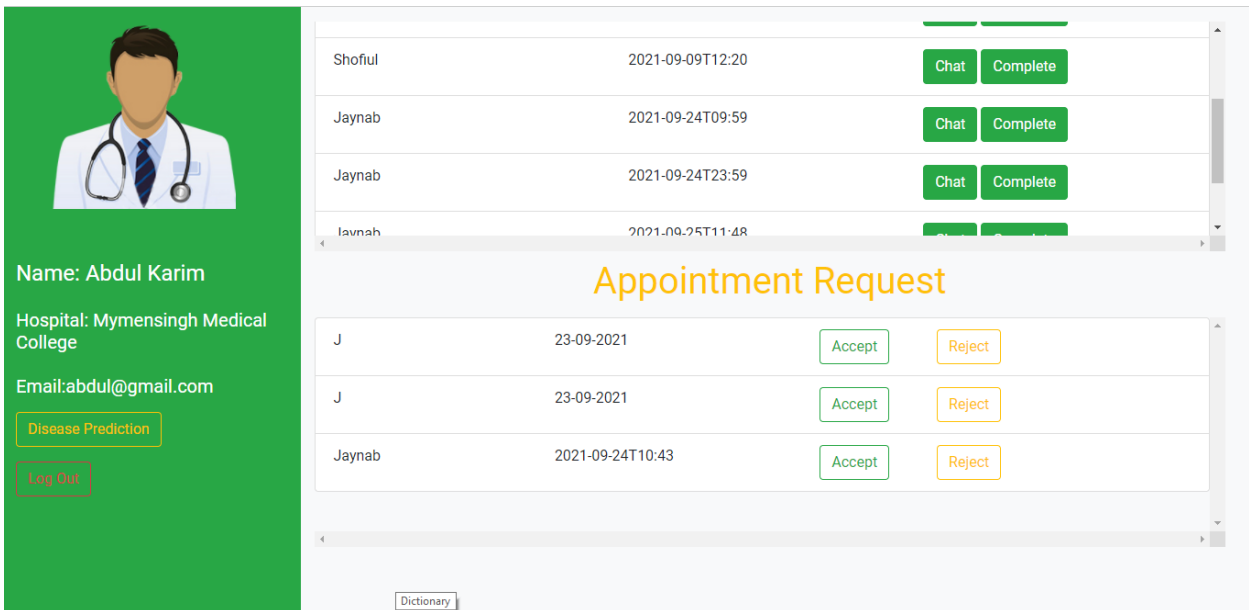
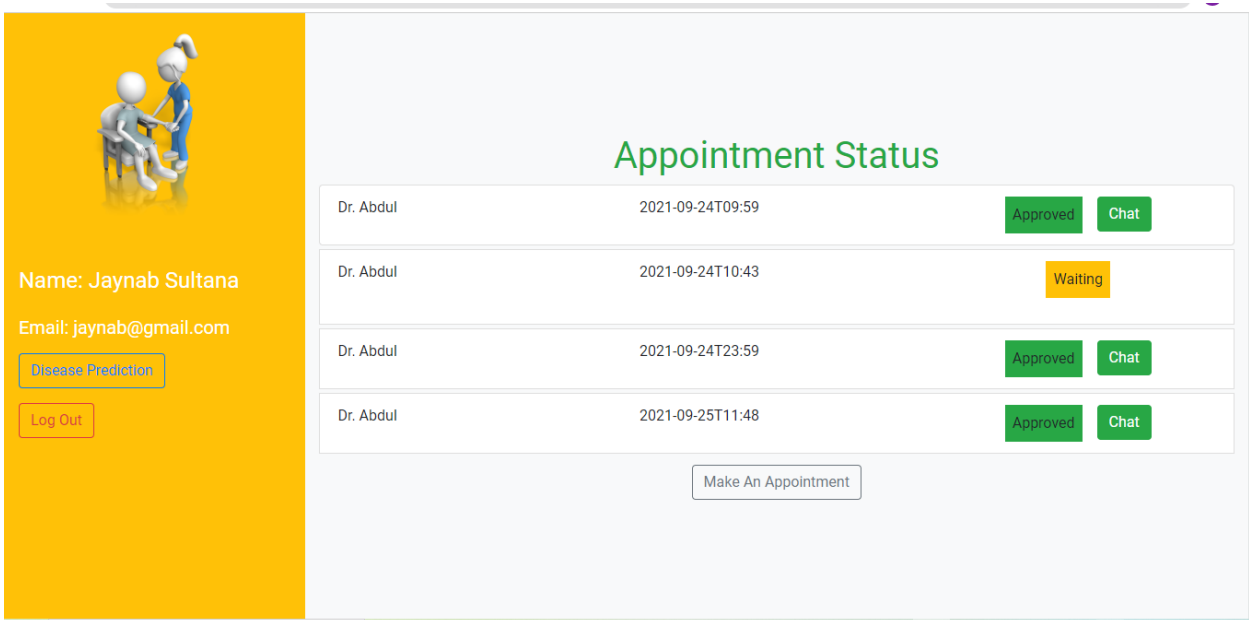


Figure 15: Doctor portal.

Figure 15 represents, in the doctor portal, the doctor can see the appointment request from a patient. When a doctor will accept the request, then the patient will be able to meet with the doctor. Using the disease prediction option doctor can detect disease by using an image file. Then, using the “Log out” option he can leave the page.



The interface features a light gray background. At the top center is the title "Disease Prediction" in green. Below it is a "Back To Home" button. A white form box contains a dropdown menu with "Malaria" selected, a "Choose File" button, and a file name "C39P4thinF_origina...105554_cell_10.png". A "Submit" button is at the bottom of the form. Below the form, a white box displays the message "Our system 97.96 % sure that patient has the selected disease" in orange text.

The interface features a light gray background. At the top center is the title "Disease Prediction" in green. Below it is a "Back To Home" button. A white form box contains a dropdown menu with "Malaria" selected, a "Choose File" button, and a file name "C3thin_original_IM...162922_cell_145.png". A "Submit" button is at the bottom of the form. Below the form, a white box displays the message "Our system 79.82 % sure that patient is safe" in green text.

Figure 16: Patient portal.

Figure 16 shows the patient portal. The patient can predict disease by uploading lesion images. If the image file has been affected by any disease, it will show a positive result. The patient can make an appointment request and will see an approved option when a doctor will accept the request. The patient can make a video chat through the chat option. For making video chat, the doctor and patient will have to present at the same time.

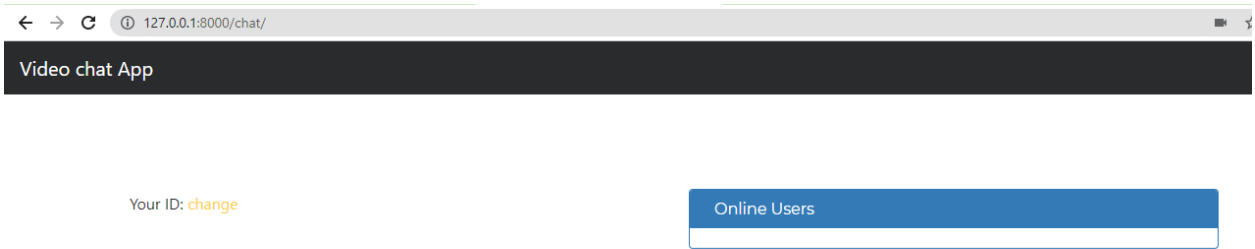


Figure 17: Online video chat.

Figure 17 represents the online video chat of the healthcare system. When a user clicks on the chat option, it will redirect to this site.

3.7 Summary

From this chapter, we get an idea of how to use the system. This section presents the system in detail.

CHAPTER 4

Result and analysis

4.1 Introduction

This section of the report comprises the outcomes that we obtained when using this system.

4.2 Result

Our main focus was that how to diagnose diseases sitting at home. We have worked with a maximum of 9 diseases- malaria, skin cancer, breast cancer, lung cancer, diabetes, pneumonia, covid-19, and heart disease. Figure 16 carries the disease detection image. When we try to predict any disease, we test the system. We give the positive case and negative case. We get the desired output. The system predicts correctly these diseases. When we want to make an appointment, we can do it successfully. We can chat with the doctors through video chat. From the result, we can expect, it can be reliable and useful for healthcare. In the previous part of training and testing, we get an encouraging result. Here are the results of multiple diseases results and plotting of training and validation where we tried multiple deep learning and machine learning models for knowing which will better perform on same dataset images and CSV data into particular disease. Here is showing that particular model graph in which we got the best accuracy for the disease.

Deep Learning Model:

Brain Tumor:

For this comparison, we employed Autoencoder and CNN, both of which provide greater accuracy in the same dataset for brain tumor classification. We trained the Autoencoder model on the dataset images for 20 epochs, with 40 steps for each epoch. Model fit is generated here. Model training takes 38 seconds at first, then 4 seconds every epoch after that. Both graphs below can be used to demonstrate this. The graph of the loss is shown in Figure 9:

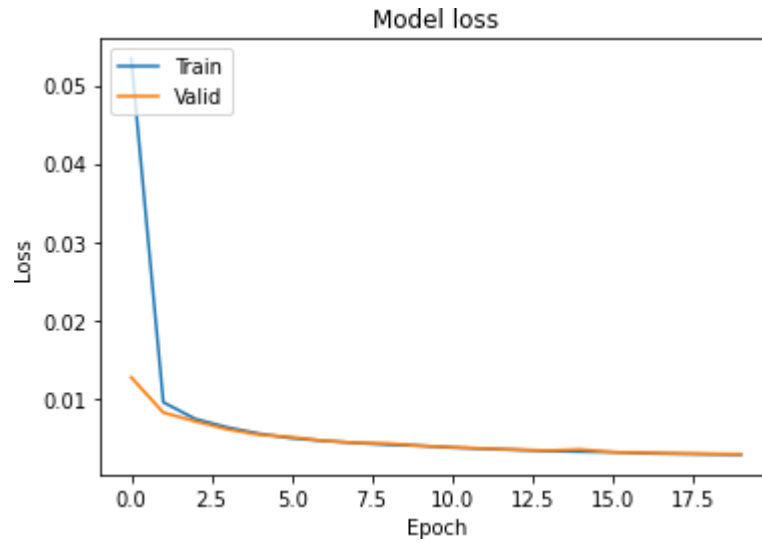
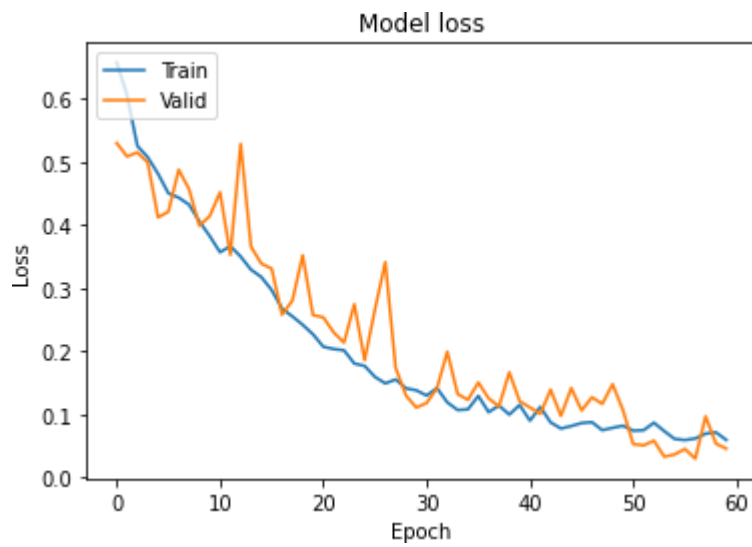
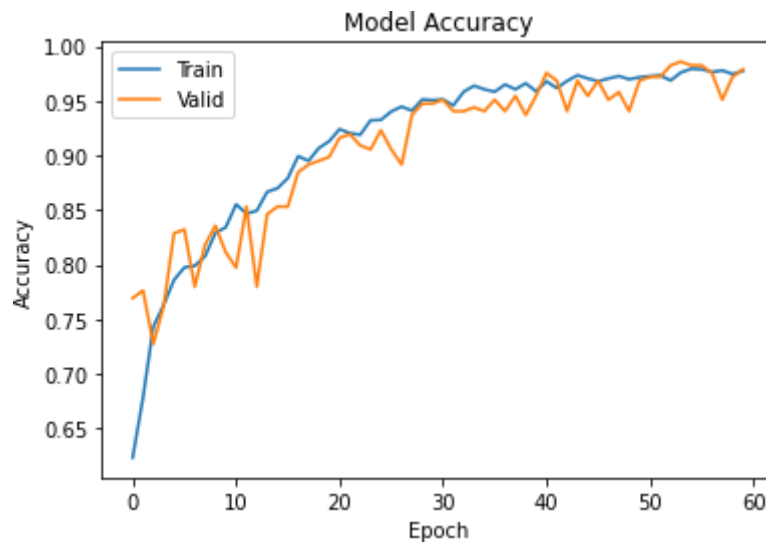


Figure 18: Training and Validation loss for Autoencoder

We trained the CNN model on the dataset images for 60 epochs, with 85 steps each epoch. Model fit is generated here. Model training takes between 9 and 11 seconds per epoch. Both graphs below can be used to demonstrate this. The graph of loss and accuracy is shown in Figure 18:



Training and Validation loss for CNN (a)



Training and Validation accuracy for CNN(b)

Figure 19: Training and validation.

Pneumonia: Here we used CNN. We have used 5 epochs in the malaria hypermeter and the steps-per-epoch is 627. Here model fit generates and the model saves as a .h5 files name. The time taken for model training is 144s/epoch. Also got better accuracy for the model. Fig 20 and 21 show training, validation accuracy, and loss.

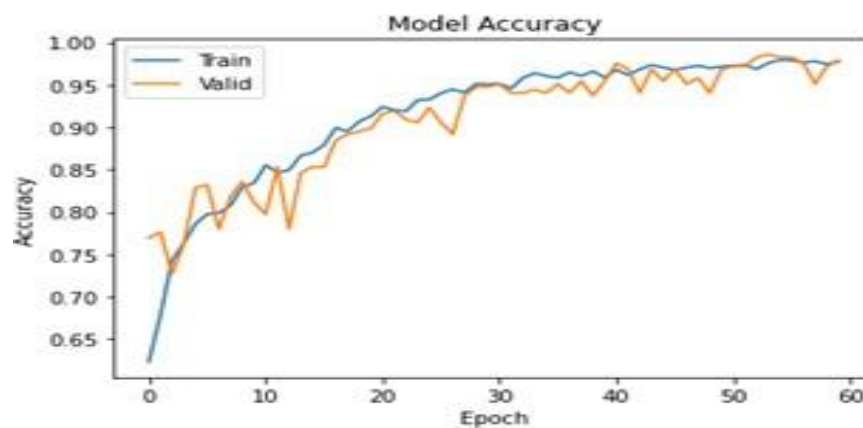


Figure 20: Pneumonia Accuracy

The accuracy we got after training is 96% and validation accuracy is 91%

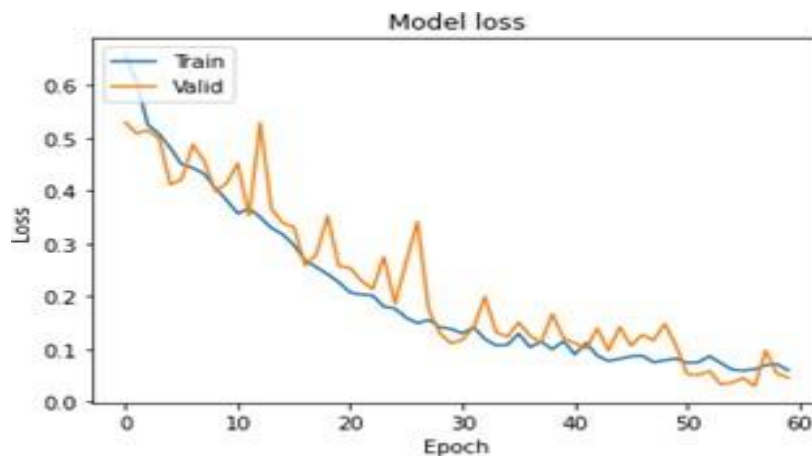


Figure 21: Pneumonia loss.

We got 0.1294 training loss and 0.1672 validation loss. Here we used CNN. We have used 5 epochs in the malaria hypermeter and the steps-per-epoch is 627. Here model fit generates and the model saves as .h5 files name. The time taken for model training is 144s/epoch. Also got better accuracy for the model. Fig 22 and 14 show training, validation accuracy, and loss.

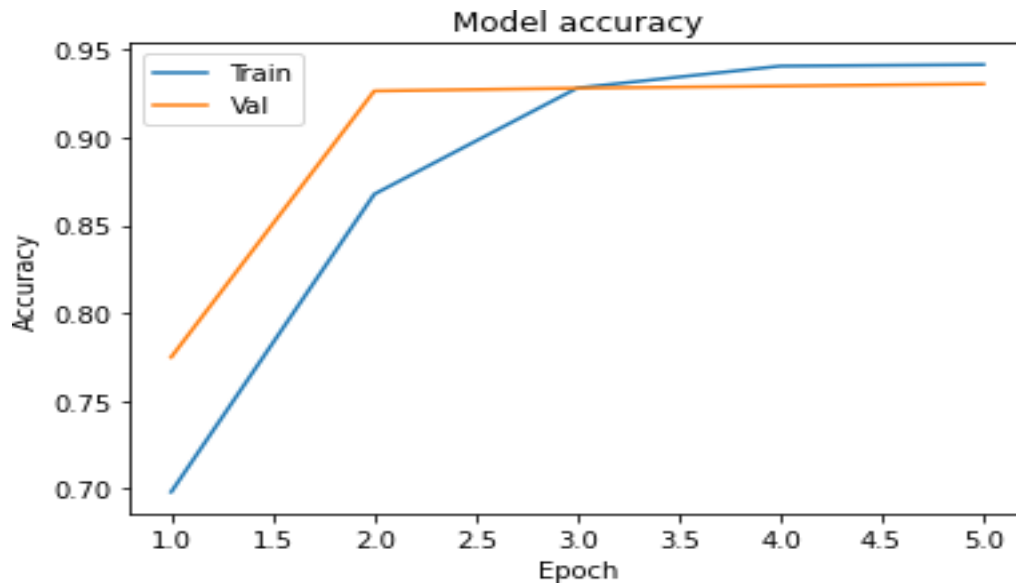


Figure 22: Malaria Accuracy

The accuracy we got after training is 95% and validation accuracy is 94%.

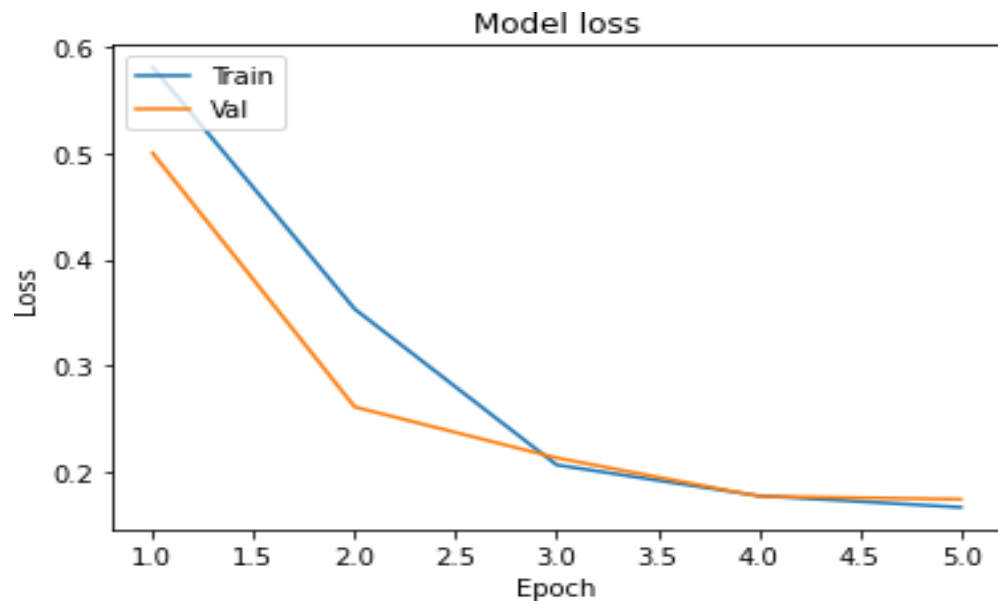


Figure 23: Malaria loss graph.

The loss we got after training is 0.1294 and validation loss is 0.1672.

Covid-19: For this confusion metrics the True Positive/Negative name refers to the anticipated result of a test, whereas the True/False refers to the real result. In Fig 24 it can be seen the confusion matrix table of performance evaluation.

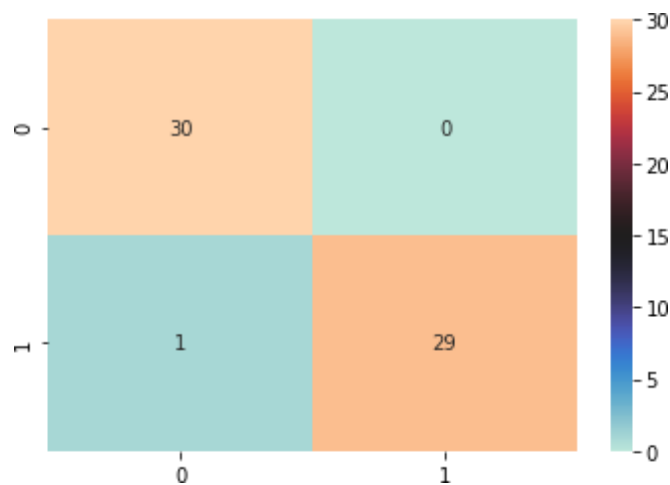


Figure 24: Confusion matrix Table

Here, TP is 30, TN is 29, FP is 1 and FN is 0. So, train generator class indices Covid is 0 and Normal is 1.

All the models are trained for 10 epochs and steps-per-epoch is 7. Here model fit generates and the model saves as .h5 files name. The time taken for model training is 9s/epoch. It can be shown with the help of both graphs below. Fig 16 and 17 shows the graph of the accuracy and loss:

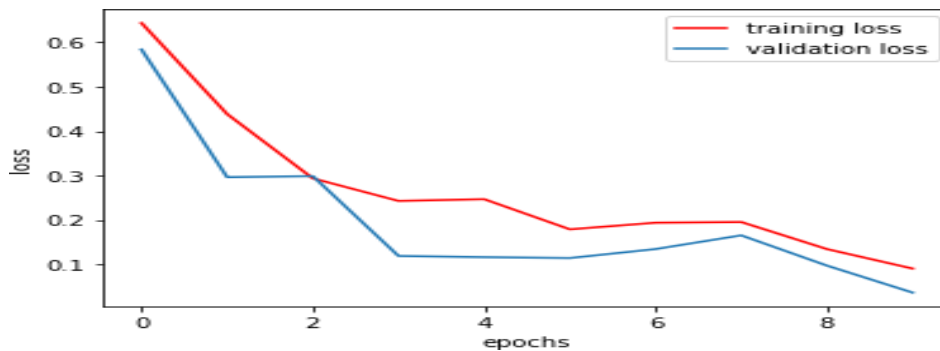


Figure 25: Training and Validation loss

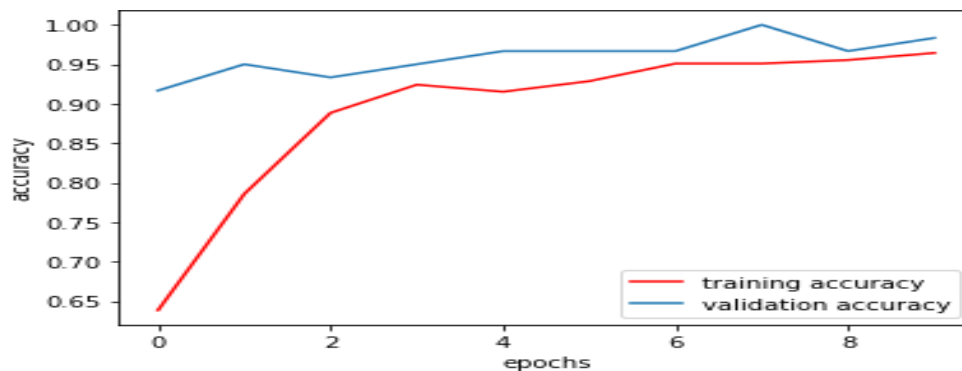


Figure 26: Training and Validation accuracy

Training loss is 0.09 and validation loss is 0.03.

Melanoma: Here we used InceptionV3. We have used 10 epochs in the malaria hypermeter and the steps-per- epoch is 63. Also got better accuracy for the model. Fig 18 and 19 shows training, validation accuracy and loss.

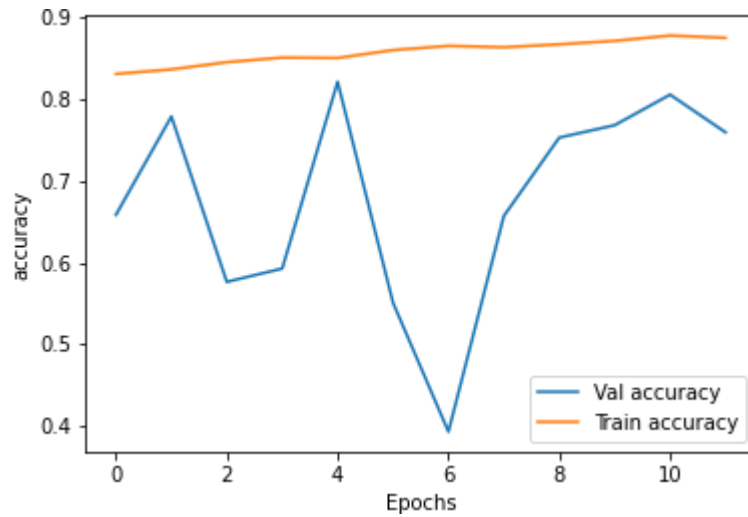


Figure 27: Melanoma Accuracy

The accuracy we got after training is 92% and validation accuracy is 75%.

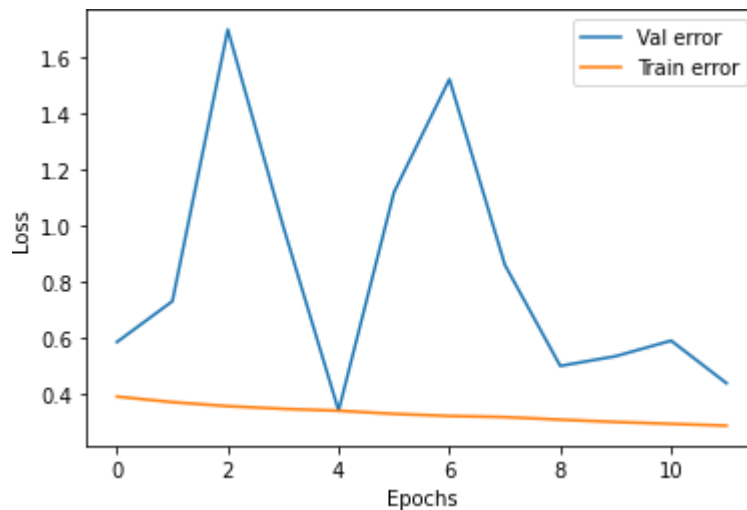


Figure 28 Melanoma loss

The loss we got after training is .2838 and validation loss is .45.

Lung Cancer:

Here we used Resnet50. We have used 10 epochs in malaria hypermeter and steps-per-epoch is 350. Here model fit generates and the model saves as .h5 files name. The time taken for model training is 70s/epoch. Also got better accuracy for the model. Where Train accuracy Score is 99.905 %, Val accuracy Score is 96.667 %, Test accuracy Score is 96.222 %, F1 Score is 96.202 %, Cohen Kappa Score is 94.330 %, ROC AUC Score is 95.430 %. Fig 12 \ shows testing, validation confusion matrix. Where 0 for Lung ACA, 1 for Lung N, 2 for Lung SCC. Fig 20 shows the confusion matrix for lung cancer.

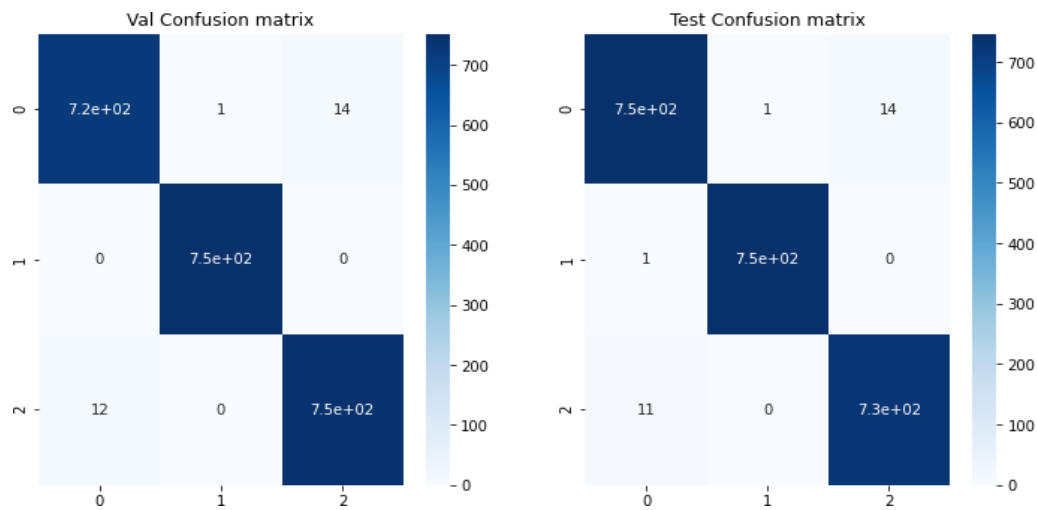


Figure 29: Confusion matrix Table

Breast Cancer:

Here we used Custom CNN. We have used 25 epochs in malaria hypermeter and steps-per-epoch is 169. Here model fit generates and the model saves as .h5 files name. The time taken for model training is 32s/epoch. From any kind off online source, we did not get the better dataset for the breast cancer image processing. Also got better accuracy for the CNN model. Fig 21 and 22 training, validation accuracy and loss.

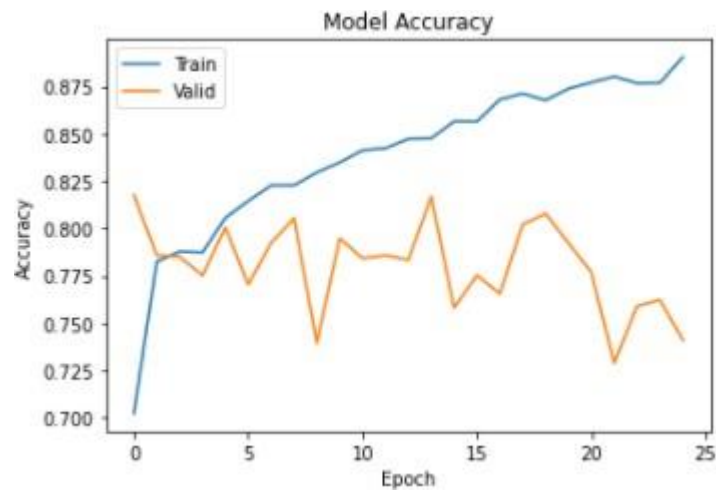


Figure 30: Breast Cancer Accuracy

The accuracy we got after training is 89% and validation accuracy is 74%.

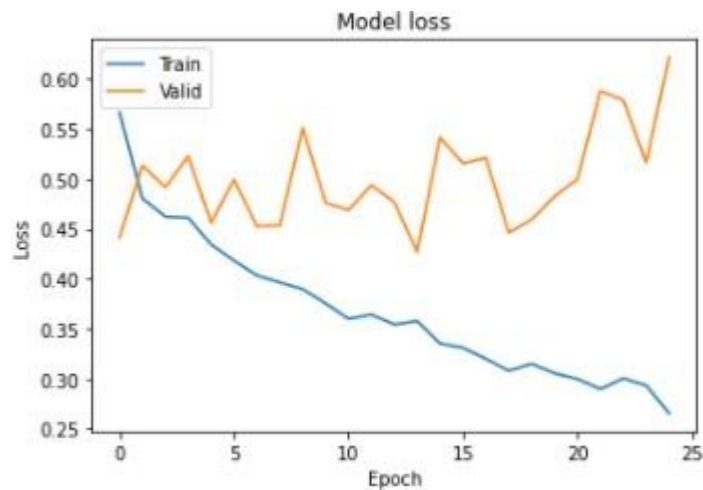


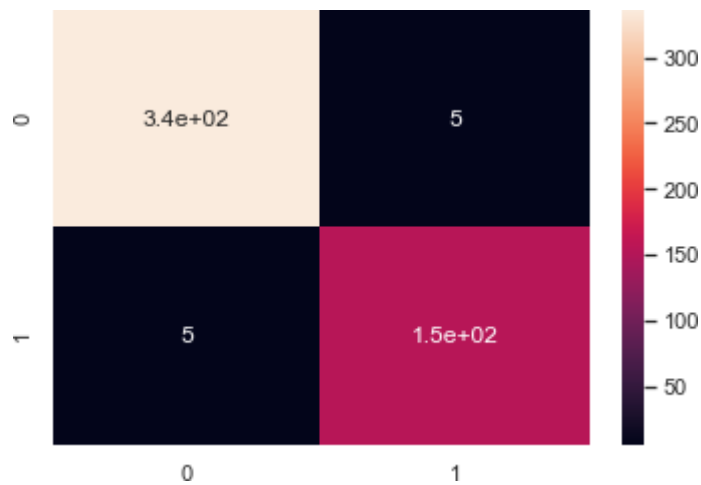
Figure 31: Breast Cancer Accuracy

The loss we got after training is 0.265 and validation loss is 0.621.

Machine Learning Model:

Diabetes:

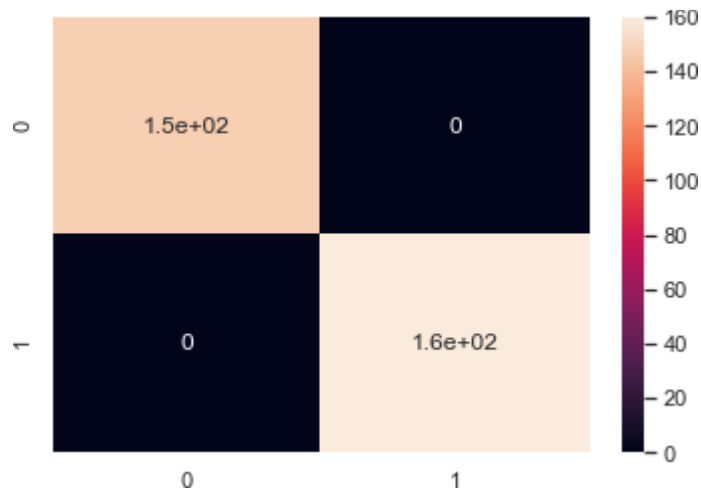
Here we used different types of models. Random Forest, Decision Tree, SVM etc. The best accuracy we got for Decision Tree Classifier. For this confusion metrics the True Positive/Negative name refers to the anticipated result of a test, whereas the True/False refersto the real result. In Fig 17 it can be seen the confusion matrix table of performance evaluation.



Here, TP is 3.4e+02, TN is 1.5e+02, FP is 5 and FN is 5. So, train generator class indices Diabetes is 0 and Normal is 1.

Heart Disease:

Here we used different types of models. Random Forest, Decision Tree, SVM etc. The best accuracy we got for Random Forest Classifier. For this confusion metrics the True Positive/Negative name refers to the anticipated result of a test, whereas the True/False refersto the real result. In Fig 18 it can be seen the confusion matrix table of performance evaluation.



Here, TP is 1.5e+02, TN is 1.6e+02, FP is 0 and FN is 0. So, train generator class indices heartdisease is 0 and Normal is 1 Here are all the disease models result and analysis where we used different types of deep learning and machine learning model. The best accuracy we got, is shown here through analysis.

4.3 Limitations

There is some weakness in our system. These weaknesses are probably due to some of our mistakes. Our chat options are not working properly. Moreover, in predicting skin cancer sometimes we get the wrong result. Some of our mistakes may have been due to incompetence. Everything else is working just fine.

4.4 Discussion

One of the most frequent types of artificial intelligence in healthcare is machine learning. There are various variations of this broad technique, which is at the heart of several approaches to AI and healthcare technology. Precision medicine is the most widely used use of traditional machine learning in the field of artificial intelligence in healthcare. For many healthcare organizations, being able to forecast which treatment techniques are likely to be successful with patients based on their makeup and treatment framework is a big step forward. The bulk of AI in healthcare applications that use machine learning and precision medicine require data for training with a known end result. Our healthcare will give us a better service if we make some changes and improve the performance.

4.5 Summary

This section gives us a important idea about the result,limitations of our “HealthMate “ System

CHAPTER 5

WORKING SHEETS

5.1 Introduction

This section will contain our working routine of the whole system.

5.2 Working plan

We have divided our whole works into two parts. Here the structure is given below.

	A	B	C	D
1	Task	Start Date	End Date	Duration
2	Project Idea	17/02/2021	24/02/2021	7
3	Project Proposal	25/02/2021	3/3/2021	5
4	Training the dataset	4/3/2021	10/3/2021	6
5	Testing the dataset	11/3/2021	17/03/2021	6
6	evaluating the data	18/03/2021	24/03/2021	6
7	Train with different model	25/03/2021	31/03/2021	6
8	Reducing overfitting	1/4/2021	7/4/2021	6
9	Training,teting, evaluation of 9 diseases	8/4/2021	14/4/2021	6
10	Presentation slide making	15/04/2021	21/04/2021	6
11	Powerpoint presentation making,Report writing start	22/04/2021	28/04/2021	6
12	Report writing end,Presentation practice	29/04/2021	5/5/2021	5
13				

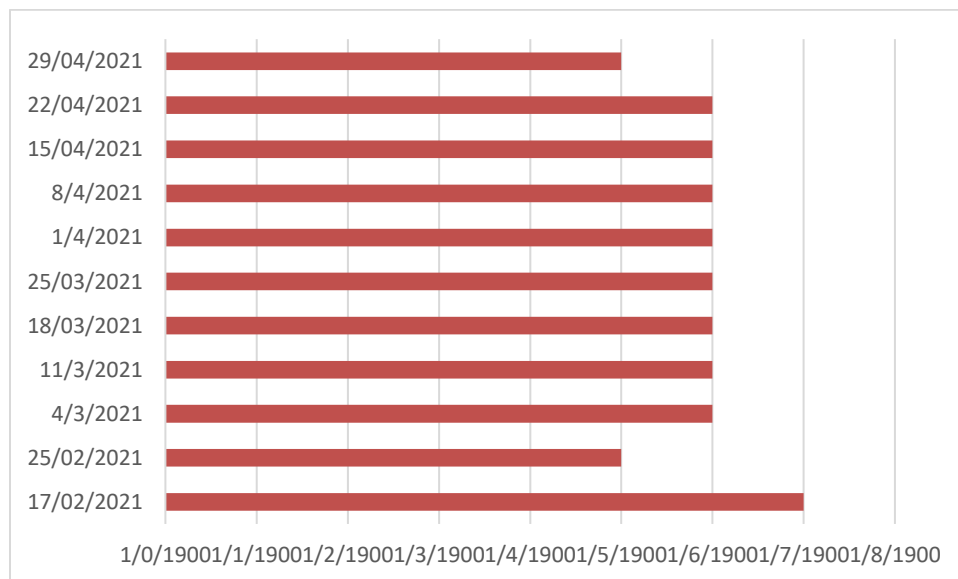


Figure 32: The working plan of our project part-1.

	A	B	C	D
	Task	Start Date	End Date	Duration
	Home page creating	16/06/2021	23/06/2021	7
	Database Creation	24/06/2021	30/6/2021	6
	Login form creating	1/7/2021	7/7/2021	6
	Sign Up form creating	8/7/2021	14/7/2021	6
	Admin portal creating	15/7/2021	21/07/2021	6
	Doctor portal creating	22/07/2021	28/07/2021	6
	Query Form Creating	29/07/2021	4/8/2021	7
	Video app creating	5/8/2021	11/8/2021	6
0	Connecting the individual part	12/8/2021	18/08/2021	6
1	Powerpoint presentation making	19/08/2021	25/08/2021	6
2	Presentation practice	26/08/2021	1/9/2021	5
3				

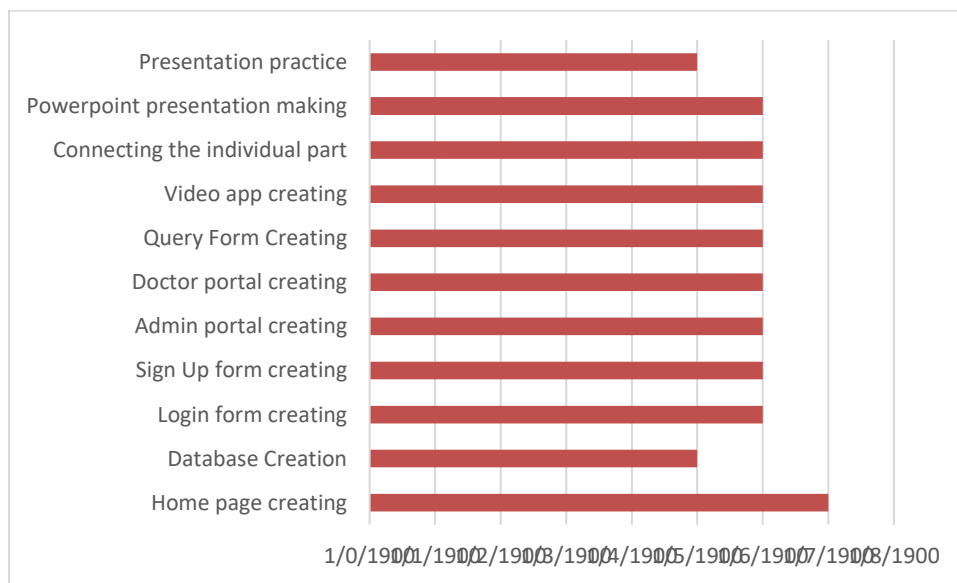


Figure 33: The working plan of our project part-2.

CHAPTER 6

DESIGN IMPACT

6.1 Introduction

In this chapter, we discuss the various impacts that our system has been able to generate.

6.2 Environmental Impact

Although it may not be stated explicitly, a software project has some environmental benefits. For starters, it reduces movement, which saves time and money. Other environmental considerations could include the fact that since images are now digital, paper is being conserved as well.

6.3 Economic Impact

Money is being conserved for both movement and other sources (such as physical diagnostics). Some of these costs will be spared from society, which will have a favorable impact on society's microeconomics, and, as a result, the economy will grow. Aside from that, clinics that adopt the system would receive more money, and doctors will be able to provide more extensive methods to diagnose. Human power is diminished here. On the economic side, people are more aware of the outcomes of many diseases at a lower cost, and they are less likely to visit a hospital for treatment, as traveling outside of the home (hospital) is not safe in this pandemic condition. Also, we notice for COVID19 how people have lost their jobs and are having financial difficulties, but to use our system, they only need a basic understanding of technology and the cost of MRI, X-ray, and cell images to upload to our system and receive the results of their disease prognosis

6.4 Social Impact

There is a beneficial effect on social design. People are more advanced in terms of technology, have learned how to use it properly, and are taking specific disease treatments at an early stage. This is a project that attempts to diagnose a population that is underprivileged in many ways, and because it is useful for those who have difficulty moving due to physical constraints, time constraints, or health constraints, one can claim that it has a very positive social impact. The project may also result in the. Even more helpful studies and algorithms are being developed, which will lead to even more progress. It will lower the death rate.

6.5 Sustainability

The system performance will rise faster if we improve the accuracy of the system, as well as the sensitivity and specificity of the system. It will be suitable for medical applications. And, without any complication, people can make an early diagnosis. Because we are still in the early stages of development, our work has

some restrictions. It will be reliable for everybody because it is not expensive. In the future, it will be a real-time system. In our country, everyone is uneducated, and some people cannot afford expensive therapy, but our system is less expensive, and they can easily obtain results by uploading photographs or data related to their specific ailment. Our work system is really simple, which is why everyone can use it. On the other hand, for individuals who are unfamiliar with the system, we can familiarize them with it and, based on their feedback, we can update our system.

6.6 Income Generation

Yes, individuals can earn money from this project since we see health problems and diseases developing every day, and people are becoming more aware of the issue. People are also becoming more interested in using updated systems, which is why people can easily earn money from this project.

6.7 Benefit

People will profit from this study by learning the outcome of their specific sickness and receiving treatment. Going outside of the house (to a hospital) for treatment is not a simple or safe undertaking in this situation. We also know that acquiring any type of disease test result takes time, but during that time, the patient's health state deteriorates and their health condition becomes serious, but with our method, people get their results promptly and may begin treatment right away. The rate of significant diseases such as Brain Tumor, Covid, Pneumonia, Heart Disease, Melanoma, Diabetes, Breast Cancer, Malaria, Lung Cancer, and others were reduced.

6.8 Summary

This section presents the different kinds of impact on our system.

CHAPTER 7

Conclusions

In this project, our aim is to detect multiple disease with high accuracy. So, we use deep learning and machine learning algorithm on our training dataset. Since, we also have a goal to build a platform between doctor and patient in online. We try to make this system in such a way so that patient can get prediction of their disease sitting at home. Each year, citizens suffer from the acute impacts of food contaminated by microbiological infections, chemical compounds, and poisons, which is a huge hazard to public health. Bangladesh remains one of the top ten countries with the highest TB burden in the world. Pneumonia and other illnesses are the leading causes of death in children under the age of five. As Bangladesh's population gets more urbanized, the prevalence of noncommunicable diseases such as cancer, diabetes, cardiovascular disease, and chronic respiratory disease is rising. The combined stresses of global climate change and urbanization are wreaking havoc on Bangladesh's most vulnerable people. The severity of the condition Bangladesh is further aggravated by filthy living circumstances, which highlight the terrible economic situations of both urban and rural residents. There are various concerns that the Bangladesh health care system has yet to address; governance, accessibility, and affordability are three major issues that are impeding the implementation of remedies to Bangladesh's public health problems. Although it may not be stated explicitly, a software project has some environmental benefits. To begin with, it decreases movement, saving time and money. Other environmental factors could include the fact that, because photos are now digital, less paper is used. Over-centralized health systems, weak governance and regulatory structures, weak management and institutional capacity in the Ministry of Health and Family Welfare, fragmented public service delivery, inefficient allocation of public resources, lack of private sector regulation, lack of HRH, absence of health workers, and poor maintenance are among the major strengths in the health sector.

Bibliography

- [1] Covid: Bangladesh records 235 more deaths, 15,776 new cases. [Online] Available: <https://www.dhakatribune.com/bangladesh/2021/08/03/covid-bangladesh-records-235-more-deaths-15-776-new-cases>
- [2] He, J., Baxter, S.L., Xu, J. *et al.* The practical implementation of artificial intelligence technologies in medicine. *Nat Med* 25, pp. 30–36, 2019. <https://doi.org/10.1038/s41591-018-0307-0>
- [3] Jiang, F.; Jiang, Y.; Zhi, H.; Dong, Y.; Li, H.; Ma, S.; Wang, Y.; Dong, Q.; Shen, H.; Wang, Y.; et al. Artificial intelligence in healthcare: Past, present and future. *Stroke Vasc. Neurol.* 2017, 2.
- [4] Islam, M.M.; Yang, H.-C.; Poly, T.N.; Jian, W.-S.; Li, Y.-C.J. Deep learning algorithms for detection of diabetic retinopathy in retinal fundus photographs: A systematic review and meta-analysis. *Comput. Methods Programs Biomed.* 2020, 191, 105320.
- [5] 32 Examples of AI in Healthcare That Will Make You Feel Better About the Future.[Online]. Available: <https://builtin.com/artificial-intelligence/artificial-intelligence-healthcare>
- [6] Kirk MD, Pires SM, Black RE, Caipo M, Crump JA, Devleesschauwer B, et al. Correction: World Health Organization Estimates of the Global and Regional Disease Burden of 22 Foodborne Bacterial, Protozoal, and Viral Diseases, 2010: A Data Synthesis. *PLoS Med* 2015 Dec;12(12):e1001940
[[FREE Full text](#)] [[CrossRef](#)] [[Medline](#)]
- [7] Davenport TH, Glaser J. Just-in-time delivery comes to knowledge management. *Harv Bus Rev* 2002 Jul;80(7):107-11, 126. [[Medline](#)]
- [8] Davenport T, Kalakota R. The potential for artificial intelligence in healthcare. *Future Healthc J* 2019 Jun;6(2):94-98 [[FREE Full text](#)] [[CrossRef](#)] [[Medline](#)]
- [9] Fink E, Kokku P, Nikiforou S, Hall L, Goldgof D, Krischer J. Selection of patients for clinical trials: an interactive web-based system. *Artificial Intelligence in Medicine* 2004;31(3):241-254 [[FREE Full text](#)] [[CrossRef](#)]
- [10] Beck JT, Rammage M, Jackson GP, Preininger AM, Dankwa-Mullan I, Roebuck MC, et al. Artificial Intelligence Tool for Optimizing Eligibility Screening for Clinical Trials in a Large Community Cancer Center. *JCO Clin Cancer Inform* 2020 Jan;4:50-59 [[FREE Full text](#)] [[CrossRef](#)] [[Medline](#)]
- [11] Forghani R, Savadjiev P, Chatterjee A, Muthukrishnan N, Reinhold C, Forghani B. Radiomics and Artificial Intelligence for Biomarker and Prediction Model Development in Oncology. *Comput Struct Biotechnol J* 2019;17:995-1008 [[FREE Full text](#)] [[CrossRef](#)] [[Medline](#)]
- [12] Zhao M, Jiang Y. Great expectations and challenges of artificial intelligence in the screening of diabetic retinopathy. *Eye* 2019 Dec 11;34(3):418-419. [[CrossRef](#)]
- [13] Peete R, Majowski K, Lauer L, Jay A. Artificial Intelligence in Healthcare. *Artificial Intelligence and Machine Learning for Business for Non-Engineers* 2019:96. [[CrossRef](#)]
- [14] Durkin K. Artificial Intelligence-driven Smart Healthcare Services, Wearable Medical Devices, and Body Sensor Networks. *Am. J. Med. Res* 2019;6(2):37. [[CrossRef](#)]
- [15] White House American artificial intelligence initiative year one annual report. In: policy Oosat, editor. USA executive office of the president of the United States; 2020:E-29.

- [16] M. M. Kamruzzaman, "Architecture of Smart Health Care System Using Artificial Intelligence," 2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), pp. 1-6, 2020. doi: 10.1109/ICMEW46912.2020.9106026.

Code

home.html

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Video Chat</title>
    <!-- Bootstrap Core CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/css/bootstrap.min.css">
    <!-- Custom Fonts -->
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet" type="text/css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700"
rel="stylesheet" type="text/css">
    <link href='https://fonts.googleapis.com/css?family=Kaushan+Script'
rel='stylesheet' type='text/css'>
    <link
```

```

href='https://fonts.googleapis.com/css?family=Droid+Serif:400,700,400italic,7
00italic' rel='stylesheet' type='text/css'>
    <link
href='https://fonts.googleapis.com/css?family=Roboto+Slab:400,100,300,700'
rel='stylesheet' type='text/css'>
    <!-- Theme CSS -->
    <link href="{% static 'css/agency.css' %}" rel="stylesheet">
</head>

<body>
    <nav class="navbar navbar-toggleable-sm navbar-inverse fixed-top bg-
inverse">
        <a class="navbar-brand" href="/">Video chat App</a>
    </nav>

    <section id="chat-app">
        <div class="container">
            <div class="row">
                <div class="col-lg-6 col-md-6 mb-4">
                    Your ID: <h4 id="peer-id" data-toggle="tooltip" data-
placement="top" title="Click to copy peer ID"></h3>
                    <a href="#getUserNameModal" data-
toggle="modal">change</a>
                </div>
                <div class="col-lg-6 col-md-6 mb-5 hide">
                    <div class="form-inline">
                        <div class="form-group mr-sm-3">
                            <label for="inputPeerUserId" class="sr-
only">Password</label>
                            <input type="text" class="form-control"
id="inputPeerUserId" placeholder="Enter your friends ID">
                        </div>
                        <button type="button" class="btn btn-outline-primary"
id='connect-btn'>Connect</button>
                    </div>
                </div>
                <div class="col-lg-6 col-md-6 mb-4">
                    <div class="panel panel-primary">
                        <div class="panel-heading">
                            <h3 class="panel-title">Online Users</h3>
                        </div>
                        <div class="panel-body">
                            <ul class="onlinepeers"></ul>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>

    <div class="container-fluid chat-container">
        <div class="row">
            </div>
        </div>

    <!-- Portfolio Modals -->
    <!-- Use the modals below to showcase details about your portfolio

```

```

projects! -->
  <!-- Portfolio Modal 1 -->
  <div class="portfolio-modal modal" id="videoCallPanel" tabindex="-1"
role="dialog" data-keyboard="false" aria-hidden="true">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="close-modal end-call hide" data-dismiss="modal">
          <div class="lr">
            <div class="rl">
              </div>
            </div>
          </div>
        </div>
        <div class="container">
          <div class="row">
            <div class="col-lg-12">
              <div class="modal-body">
                <!-- Project Details Go Here -->
                <h2 class="title">Video Call</h2>
                <div class="pure-u-2-3" id="video-container">
                  <video id="their-video"
autoplay=""></video>
                  <video id="my-video" muted="true"
autoplay=""></video>
                </div>
                <div class="text-center mt-3">
                  <button type="button" class="btn btn-
secondary mute-audio ml-3 mt-2"><i class="fa fa-microphone-slash"></i>Mute
Audio</button>
                  <button type="button" class="btn btn-
secondary mute-video ml-3 mt-2"><i class="fa fa-video-camera"></i>Mute
Video</button>
                  <button type="button" class="btn btn-
danger end-call ml-3 mt-2" data-dismiss="modal"><i class="fa fa-
times"></i>End Call</button>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>

  <div class="modal" id="getUserNameModal" tabindex="-1" role="dialog"
data-keyboard="false" aria-labelledby="exampleModalLabel" aria-hidden="true"
data-backdrop='static'>
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Enter your
Name</h5>
        </div>
        <div class="modal-body">
          <div class="form-group">
            <input type="text" class="form-control" id="user-
name">

```

```

        </div>
    </div>
    <div class="modal-footer">
        <button type="button" data-dismiss="modal" class="btn
btn-primary username-done">Done</button>
    </div>
</div>
</div>
</div>
</div>

<!-- Modal -->
<div class="modal" id="callConfirmationModal" tabindex="-1" role="dialog"
data-keyboard="false" aria-labelledby="exampleModalLabel" aria-hidden="true"
data-backdrop='static'>
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title peer-name"></h5>
            </div>
            <div class="modal-body">
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-danger reject-call"
data-dismiss="modal">Reject</button>
                <button type="button" class="btn btn-primary accept-call"
data-dismiss="modal">Accept</button>
            </div>
        </div>
    </div>
</div>
</div>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></scri
pt>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js"></
script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/js/bootstrap.min.js"></script>

<script src="{% static 'js/peer.js' %}"></script>
<script src="{% static 'js/app.js' %}"></script>
<script src="{% static 'js/peer-client.js' %}"></script>
</body>

</html>

```

Admin.py

```

from django.contrib import admin

# Register your models here.

```

Apps.py

```
from django.apps import AppConfig
```

```
class MyappConfig(AppConfig):  
    name = 'chatApp'
```

models.py

```
from django.db import models
```

```
# Create your models here.
```

Temp.txt

```
<!DOCTYPE html>  
<html lang="en">  
{% load static %}  
<head>  
    <meta charset="utf-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <meta name="description" content="">  
    <meta name="author" content="">  
    <title>Video Chat</title>  
    <!-- Bootstrap Core CSS -->  
    <link rel="stylesheet"  
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-  
alpha.6/css/bootstrap.min.css">  
    <!-- Custom Fonts -->  
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-  
awesome.min.css" rel="stylesheet" type="text/css">  
    <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700"  
rel="stylesheet" type="text/css">  
    <link href='https://fonts.googleapis.com/css?family=Kaushan+Script'  
rel='stylesheet' type='text/css'>  
    <link  
href='https://fonts.googleapis.com/css?family=Droid+Serif:400,700,400italic,7  
00italic' rel='stylesheet' type='text/css'>  
    <link  
href='https://fonts.googleapis.com/css?family=Roboto+Slab:400,100,300,700'  
rel='stylesheet' type='text/css'>  
    <!-- Theme CSS -->  
    <link href="{% static 'css/agency.css' %}" rel="stylesheet">  
</head>  
  
<body>  
  
    <section id="chat-app">  
        <div class="container">  
            <div class="row">  
                <div class="col-lg-6 col-md-6 mb-4">
```

```

        Your ID: <h4 id="peer-id" data-toggle="tooltip" data-
placement="top" title="Click to copy peer ID"></h4>
        <a href="#getUserUsernameModal" data-
toggle="modal">change</a>
    </div>
    <div class="col-lg-6 col-md-6 mb-5 hide">
        <div class="form-inline">
            <div class="form-group mr-sm-3">
                <label for="inputPeerUserId" class="sr-
only">Password</label>
                <input type="text" class="form-control"
id="inputPeerUserId" placeholder="Enter your friends ID">
            </div>
            <button type="button" class="btn btn-outline-primary"
id='connect-btn'>Connect</button>
        </div>
    </div>
    <div class="col-lg-6 col-md-6 mb-4">
        <div class="panel panel-primary">
            <div class="panel-heading">
                <h3 class="panel-title">Online Users</h3>
            </div>
            <div class="panel-body">
                <ul class="onlinepeers"></ul>
            </div>
        </div>
    </div>
</div>
</div>
</section>

<div class="container-fluid chat-container">
    <div class="row">
    </div>
</div>

<!-- Portfolio Modals -->
<!-- Use the modals below to showcase details about your portfolio
projects! -->
<!-- Portfolio Modal 1 -->
<div class="portfolio-modal modal" id="videoCallPanel" tabindex="-1"
role="dialog" data-keyboard="false" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="close-modal end-call hide" data-dismiss="modal">
                <div class="lr">
                    <div class="rl">
                    </div>
                </div>
            </div>
        </div>
        <div class="container">
            <div class="row">
                <div class="col-lg-12">
                    <div class="modal-body">
                        <!-- Project Details Go Here -->
                        <h2 class="title">Video Call</h2>
                        <div class="pure-u-2-3" id="video-container">

```

```

        <video id="their-video"
autoplay=""></video>
        <video id="my-video" muted="true"
autoplay=""></video>
    </div>

    <div class="text-center mt-3">
        <button type="button" class="btn btn-
secondary mute-audio ml-3 mt-2"><i class="fa fa-microphone-slash"></i>Mute
Audio</button>
        <button type="button" class="btn btn-
secondary mute-video ml-3 mt-2"><i class="fa fa-video-camera"></i>Mute
Video</button>
        <button type="button" class="btn btn-
danger end-call ml-3 mt-2" data-dismiss="modal"><i class="fa fa-
times"></i>End Call</button>
    </div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>

    <div class="modal" id="getUserNameModal" tabindex="-1" role="dialog"
data-keyboard="false" aria-labelledby="exampleModalLabel" aria-hidden="true"
data-backdrop='static'>
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">Enter your
Name</h5>
                </div>
                <div class="modal-body">
                    <div class="form-group">
                        <input type="text" class="form-control" id="user-
name">
                    </div>
                </div>
                <div class="modal-footer">
                    <button type="button" data-dismiss="modal" class="btn
btn-primary username-done">Done</button>
                </div>
            </div>
        </div>
    </div>

    <!-- Modal -->
    <div class="modal" id="callConfirmationModal" tabindex="-1" role="dialog"
data-keyboard="false" aria-labelledby="exampleModalLabel" aria-hidden="true"
data-backdrop='static'>
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title peer-name"></h5>
                </div>
            </div>
        </div>
    </div>

```



```

        <div class="modal-body">
        </div>
        <div class="modal-footer">
            <button type="button" class="btn btn-danger reject-call"
data-dismiss="modal">Reject</button>
            <button type="button" class="btn btn-primary accept-call"
data-dismiss="modal">Accept</button>
        </div>
    </div>
</div>
</div>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></scri
pt>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js"></
script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/js/bootstrap.min.js"></script>

<script src="{% static 'js/peer.js' %}"></script>
<script src="{% static 'js/app.js' %}"></script>
<script src="{% static 'js/peer-client.js' %}"></script>
</body>

</html>

```

Urls.py

```

from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='chat')
]

```

views.py

```

from django.shortcuts import render
import os

def home(request):
    return render(request, 'home.html')

```

hospital

admin.py

```

from django.contrib import admin
from . import models

```

```

admin.site.site_header = 'AI Hospital Administration'
admin.site.register(models.Doctor)
admin.site.register(models.Patient)
admin.site.register(models.Appointment)
admin.site.register(models.Disease)

```

apps.py

```

from django.apps import AppConfig

class HospitalConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'hospital'

```

forms.py

```

from django import forms
from .models import Doctor, Patient

class DoctorForm(forms.ModelForm):
    class Meta:
        model = Doctor
        fields = "__all__"
        labels = {
            "first_name": "First Name",
            "last_name": "Last Name",
            "hospital": "Hospital",
            "email": "Your Email",
            "password": "Your Password"
        }
        error_messages = {
            "first_name": {
                "required": "Your first name must not be empty!"
            },
            "last_name": {
                "required": "Your last name must not be empty!"
            },
            "hospital": {
                "required": "Your hospital name must not be empty!"
            },
            "email": {
                "required": "Email is required"
            },
            "password": {
                "required": "You must need to provide a password"
            }
        }
        widgets = {
            "password": forms.PasswordInput()
        }

```

```

    }

class PatientForm(forms.ModelForm):
    class Meta:
        model = Patient
        fields = "__all__"
        labels = {
            "first_name": "First Name",
            "last_name": "Last Name",
            "email": "Your Email",
            "password": "Your Password"
        }
        error_messages = {
            "first_name": {
                "required": "Your first name must not be empty!"
            },
            "last_name": {
                "required": "Your last name must not be empty!"
            },
            "email": {
                "required": "Email is required"
            },
            "password": {
                "required": "You must need to provide a password"
            }
        }
        widgets = {
            "password": forms.PasswordInput()
        }

```

models.py

```

from __future__ import unicode_literals
from django.db import models
from django.utils.timezone import now

# Create your models here.
class Doctor(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    hospital = models.CharField(max_length=100)
    email = models.EmailField(max_length=100, null=False, default="", unique=True)
    password = models.CharField(max_length=30, null=False, default="")

    def save(self, *args, **kwargs):
        self.email = self.email.lower()
        return super(Doctor, self).save(*args, **kwargs)

    def __str__(self):
        return self.first_name

class Patient(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email =

```

```

models.EmailField(max_length=100,null=False,default="",unique=True)
password = models.CharField(max_length=40,null=False,default="")

def save(self, *args, **kwargs):
    self.email = self.email.lower()
    return super(Patient, self).save(*args, **kwargs)

def __str__(self):
    return self.first_name

class Appointment(models.Model):
    patient = models.IntegerField(null=False,default=-1)
    doctor = models.IntegerField(null=False,default=-1)
    date = models.CharField(max_length=40,null=False,default="01/01/1970")
    approved = models.BooleanField(default=False,null=False)

    def __str__(self):
        return self.date

class Disease(models.Model):
    diseaseName = models.CharField(max_length=40,null=False,default="No
Disease",unique=True)

    def __str__(self):
        return self.diseaseName

```

tests.py

```

from django.test import TestCase

# Create your tests here.

```

Urls.py

```

from django.urls import path
from django.conf.urls.static import static
from django.conf import settings
from . import views

urlpatterns = [
    path("", views.home_page, name = "homepage"),
    path("aboutus/", views.aboutus, name = "aboutus"),
    path("patientlogin/", views.PatientLogin.as_view(), name = "patientlogin"),
    path("doctorlogin/", views.doctorlogin, name = "doctorlogin"),
    path("doctorSignin/", views.doctorSignin, name="doctorSignin"),
    path("patientSignin/", views.patientSignin, name="patientSignin"),
    path('contactus/', views.contactUs, name = "contactus"),
    path("doctorDash/", views.DoctorDash.as_view(), name = "doctorDash"),
    path("patientDash/", views.PatientDash.as_view(), name = "patientDash"),
    path("diseasePrediction/", views.DiseasePrediction.as_view(), name =
"diseasePrediction"),
    path("appointment/", views.MakeAppointments.as_view(), name =
"appointment"),
    path("success/", views.success, name="success")

```

```
]
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

views.py

```
from django.shortcuts import render
from django.urls import reverse
from django.http import HttpResponseRedirect, HttpResponse
from django.views.generic import DetailView, View
from django.core.files.storage import FileSystemStorage
from keras.models import load_model
from keras.preprocessing import image
import numpy as np
import h5py
from .models import Doctor, Patient, Appointment, Disease
from .forms import DoctorForm, PatientForm

# Create your views here.

def home_page(request):
    return render(request, 'hospital/index.html')

def aboutus(request):
    return render(request, 'hospital/aboutus.html')

#####
#                                     #
#   Doctors Views                     #
#                                     #
#####

def doctorlogin(request):
    try:
        if request.session.get("doctorID"):
            return HttpResponseRedirect(reverse("doctorDash"))
        if request.method == "POST":
            email = request.POST["userEmail"]
            password = request.POST["userPassword"]
            doctor = Doctor.objects.get(email=email.lower())
            if password == doctor.password:
                request.session["doctor"] = f"{doctor.first_name} {doctor.last_name}"
                request.session["doctorID"] = doctor.id
                return HttpResponseRedirect(reverse("doctorDash"))
            return render(request, 'hospital/login.html', context={
                "title": "Doctor Login",
                "usertype": "Doctor Email",
                "image": "images/doctor.png"
            })
    except:
        return HttpResponse(f"Email: {request.POST['userEmail']} does not
```

```
exist in database")
```

```
class DoctorDash(View):
    def get(self, request):
        if request.session.get("doctorID"):
            patients = []
            doctor = Doctor.objects.get(pk = request.session["doctorID"])
            appointments = Appointment.objects.filter(doctor=int(doctor.id))
            for item in appointments:
                name = Patient.objects.get(pk=int(item.patient)).first_name
                patients.append(name)
            hasAppointment = appointments.count()>0
            return render(request, "hospital/doctor_dash.html", context={
                "doctor" : doctor,
                "appointments": zip(appointments,patients),
                "hasAppointment": hasAppointment,
                "appointments2": zip(appointments,patients),
            })
        else:
            return HttpResponseRedirect(reverse("doctorlogin"))
    def post(self, request):
        if request.POST.get("logout", False):
            del request.session["doctor"]
            del request.session["doctorID"]
            return HttpResponseRedirect(reverse("homepage"))
        elif request.POST.get("accept", False):
            id = request.POST.get("ID")
            appointment = Appointment.objects.get(pk=int(id))
            appointment.approved = True
            appointment.save()
            return HttpResponseRedirect(reverse("doctorDash"))
        elif request.POST.get("reject", False) or
request.POST.get("complete", False):
            id = request.POST.get("ID")
            Appointment.objects.get(pk=int(id)).delete()
            return HttpResponseRedirect(reverse("doctorDash"))
        elif request.POST.get("chat", False):
            return HttpResponseRedirect(reverse("chat"))

    def doctorSignin(request):
        form = DoctorForm(request.POST or None)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect(reverse("success"))
        else:
            form = DoctorForm(request.POST or None)
            return render(request, "hospital/signin.html", context={
                "title": "Doctor Sign In",
                "form": form
            })
```

```
#####
#
#     End of Doctor Views
#
```

```
#####
```

```
#####  
#  
# Patient Views #  
#  
#####
```

```
class PatientLogin(View):  
    def get(self, request):  
        if request.session.get("patientID"):  
            return HttpResponseRedirect(reverse("patientDash"))  
        return render(request, 'hospital/login.html', context={  
            "title": "Patient Login",  
            "usertype": "Patient Email",  
            "image": "images/patient.png"  
        })  
    def post(self, request):  
        try:  
            email = request.POST["userEmail"]  
            password = request.POST["userPassword"]  
            patient = Patient.objects.get(email=email.lower())  
            if password == patient.password:  
                request.session["patient"] = f"{patient.first_name}  
{patient.last_name}"  
                request.session["patientID"] = patient.id  
                return HttpResponseRedirect(reverse('patientDash'))  
        except:  
            return HttpResponse(f"Email: {request.POST['userEmail']} doesn't  
exist in the database")
```

```
class PatientDash(View):  
    def get(self, request):  
        if request.session.get("patientID"):  
            doctors = []  
            patient = Patient.objects.get(pk =  
request.session.get("patientID"))  
            appointment = Appointment.objects.filter(patient=patient.id)  
            for doc in appointment:  
                name = Doctor.objects.get(pk = int(doc.doctor)).first_name  
                doctors.append(name)  
            #print(appointment.patient.all())  
            isEmpty = len(doctors)==0  
            return render(request, "hospital/patient_dash.html", context={  
                "patient" : patient,  
                "appointments": zip(appointment, doctors),  
                "appointmentCount": isEmpty  
            })  
        else:  
            return HttpResponseRedirect(reverse("patientlogin"))
```

```

def post(self, request):
    if request.POST.get("logout", False):
        del request.session["patient"]
        del request.session["patientID"]
        return HttpResponseRedirect(reverse("homepage"))
    elif request.POST.get("chat", False):
        return HttpResponseRedirect(reverse("chat"))

class MakeAppointments(View):
    def get(self, request):
        if request.session["patientID"]:
            doctors = Doctor.objects.all()
            patient =
Patient.objects.get(pk=int(request.session["patientID"]))
            return render(request, "hospital/appointments.html", context={
                "doctors":doctors,
                "patient":patient
            })
        else:
            return HttpResponseRedirect(reverse("patientlogin"))
    def post(self, request):
        userDate = request.POST["selected_date"]
        doctorID = request.POST["doctors"]
        patientID = request.POST["patient"]
        appointment = Appointment(doctor=doctorID, patient=patientID, date =
userDate)
        appointment.save()
        return HttpResponseRedirect(reverse("patientDash"))

def patientSignin(request):
    form = PatientForm(request.POST or None)
    if form.is_valid():
        form.save()
        return HttpResponseRedirect(reverse("success"))
    else:
        form = PatientForm(request.POST or None)
        return render(request, "hospital/signin.html", context={
            "title": "Patient Sign In",
            "form": form
        })

#####
#                                     #
#      End of Patient Views          #
#                                     #
#####

def contactUs(request):
    return render(request, 'hospital/contactus.html')

```



```

class DiseasePrediction(View):
    def get(self, request):
        if request.session.get("doctorID") or
request.session.get("patientID"):
            diseases = Disease.objects.all().order_by('diseaseName')
            return render(request, "hospital/diseasePrediction.html", context={
                "diseases":diseases
            })
        else:
            return HttpResponseRedirect(reverse('homepage'))
    def post(self, request):
        id = request.POST.get("disease")
        img = request.FILES.get("imagePath")
        fss = FileSystemStorage()
        fss.save(img.name, img)
        path = fss.url(img.name)
        ml_path = "."+path
        path = "../"+ path[1:]
        print(path)
        if int(id) == 3 or int(id) == 4:
            percent = prediction(id, ml_path, 224, 224)

        else:
            percent = prediction(id, ml_path, 64, 64)
        if percent > .5:
            result = True
        else:
            result = False

        fss.delete(path)
        diseases = Disease.objects.all().order_by('diseaseName')
        return render(request, "hospital/diseasePrediction.html", context={
            "diseases":diseases,
            "result": result,
            "percent": "{:.2f}".format(percent*100),
            "safe": "{:.2f}".format((1-percent)*100)
        })
    def success(request):
        return render(request, "hospital/success.html")

    def prediction(model_num, img_path, img_height, img_width):
        model = load_model("./h5models/"+str(model_num)+".h5")
        img = image.load_img(img_path, target_size=(img_height, img_width))
        x = image.img_to_array(img)
        x/=255
        x = np.expand_dims(x, axis=0)
        result = model.predict(x)
        return result[0][0]

```

smartHospital

asgi.py

```

"""
ASGI config for smartHospital project.

It exposes the ASGI callable as a module-level variable named
`application`.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'smartHospital.settings')

application = get_asgi_application()

```

Settings.py

```

"""
Django settings for smartHospital project.

Generated by 'django-admin startproject' using Django 3.2.4.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-53d)t9m=d5bok9y0y74@z5$rg+ck_k=6&^o_m2l%czv!$1(dp='

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'hospital',

```

```

    'chatApp',
    'widget_tweaks',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'smartHospital.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'smartHospital.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [

```

```

        {
            'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
        },
        {
            'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
        },
        {
            'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
        },
        {
            'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
        },
    ]

```

```

# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

```

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'Asia/Dhaka'
```

```
USE_I18N = True
```

```
USE_L10N = True
```

```
USE_TZ = True
```

```

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

```

```

STATIC_URL = '/static/'
STATICFILES_DIRS=[
    BASE_DIR / 'static'
]

```

```

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

```

MEDIA_URL = 'media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

```

Urls.py

```
"""smartHospital URL Configuration
```

```

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:

```

Function views

1. Add an import: `from my_app import views`
2. Add a URL to `urlpatterns`: `path('', views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to `urlpatterns`: `path('', Home.as_view(), name='home')`

Including another URLconf

1. Import the `include()` function: `from django.urls import include, path`
2. Add a URL to `urlpatterns`: `path('blog/', include('blog.urls'))`

"""

```
from django.contrib import admin
from django.urls import path, include
```

```
urlpatterns = [
    path('', include('hospital.urls')),
    path("chat/", include('chatApp.urls')),
    path('admin/', admin.site.urls),
]
```

Wsgi.py

"""

WSGI config for smartHospital project.

*It exposes the WSGI callable as a module-level variable named
`application`.*

For more information on this file, see

<https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi/>

"""

```
import os
```

```
from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'smartHospital.settings')
```

```
application = get_wsgi_application()
```

Chat app

Templates

Home.html

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```

    <meta name="description" content="">
    <meta name="author" content="">
    <title>Video Chat</title>
    <!-- Bootstrap Core CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/css/bootstrap.min.css">
    <!-- Custom Fonts -->
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet" type="text/css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700"
rel="stylesheet" type="text/css">
    <link href='https://fonts.googleapis.com/css?family=Kaushan+Script'
rel='stylesheet' type='text/css'>
    <link
href='https://fonts.googleapis.com/css?family=Droid+Serif:400,700,400italic,7
00italic' rel='stylesheet' type='text/css'>
    <link
href='https://fonts.googleapis.com/css?family=Roboto+Slab:400,100,300,700'
rel='stylesheet' type='text/css'>
    <!-- Theme CSS -->
    <link href="{% static 'css/agency.css' %}" rel="stylesheet">
</head>

<body>
    <nav class="navbar navbar-toggleable-sm navbar-inverse fixed-top bg-
inverse">
        <a class="navbar-brand" href="/">Video chat App</a>
    </nav>

    <section id="chat-app">
        <div class="container">
            <div class="row">
                <div class="col-lg-6 col-md-6 mb-4">
                    Your ID: <h4 id="peer-id" data-toggle="tooltip" data-
placement="top" title="Click to copy peer ID"></h3>
                    <a href="#getUserUsernameModal" data-
toggle="modal">change</a>
                </div>
                <div class="col-lg-6 col-md-6 mb-5 hide">
                    <div class="form-inline">
                        <div class="form-group mr-sm-3">
                            <label for="inputPeerUserId" class="sr-
only">Password</label>
                            <input type="text" class="form-control"
id="inputPeerUserId" placeholder="Enter your friends ID">
                        </div>
                        <button type="button" class="btn btn-outline-primary"
id='connect-btn'>Connect</button>
                    </div>
                </div>
                <div class="col-lg-6 col-md-6 mb-4">
                    <div class="panel panel-primary">
                        <div class="panel-heading">
                            <h3 class="panel-title">Online Users</h3>
                        </div>
                        <div class="panel-body">

```

```

        <ul class="onlinepeers"></ul>
      </div>
    </div>
  </div>
</div>
</section>

<div class="container-fluid chat-container">
  <div class="row">
    </div>
  </div>

  <!-- Portfolio Modals -->
  <!-- Use the modals below to showcase details about your portfolio
  projects! -->
  <!-- Portfolio Modal 1 -->
  <div class="portfolio-modal modal" id="videoCallPanel" tabindex="-1"
  role="dialog" data-keyboard="false" aria-hidden="true">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="close-modal end-call hide" data-dismiss="modal">
          <div class="lr">
            <div class="rl">
              </div>
            </div>
          </div>
          <div class="container">
            <div class="row">
              <div class="col-lg-12">
                <div class="modal-body">
                  <!-- Project Details Go Here -->
                  <h2 class="title">Video Call</h2>
                  <div class="pure-u-2-3" id="video-container">
                    <video id="their-video"
autoplay=""></video>
                    <video id="my-video" muted="true"
autoplay=""></video>
                  </div>

                  <div class="text-center mt-3">
                    <button type="button" class="btn btn-
secondary mute-audio ml-3 mt-2"><i class="fa fa-microphone-slash"></i>Mute
Audio</button>

                    <button type="button" class="btn btn-
secondary mute-video ml-3 mt-2"><i class="fa fa-video-camera"></i>Mute
Video</button>

                    <button type="button" class="btn btn-
danger end-call ml-3 mt-2" data-dismiss="modal"><i class="fa fa-
times"></i>End Call</button>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>

```

```

</div>

<div class="modal" id="getUserNameModal" tabindex="-1" role="dialog"
data-keyboard="false" aria-labelledby="exampleModalLabel" aria-hidden="true"
data-backdrop='static'>
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Enter your
Name</h5>
      </div>
      <div class="modal-body">
        <div class="form-group">
          <input type="text" class="form-control" id="user-
name">
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" data-dismiss="modal" class="btn
btn-primary username-done">Done</button>
      </div>
    </div>
  </div>
</div>

<!-- Modal -->
<div class="modal" id="callConfirmationModal" tabindex="-1" role="dialog"
data-keyboard="false" aria-labelledby="exampleModalLabel" aria-hidden="true"
data-backdrop='static'>
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title peer-name"></h5>
      </div>
      <div class="modal-body">
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-danger reject-call"
data-dismiss="modal">Reject</button>
        <button type="button" class="btn btn-primary accept-call"
data-dismiss="modal">Accept</button>
      </div>
    </div>
  </div>
</div>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></scri
pt>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js"></
script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/js/bootstrap.min.js"></script>

```



```

    <script src="{% static 'js/peer.js' %}"></script>
    <script src="{% static 'js/app.js' %}"></script>
    <script src="{% static 'js/peer-client.js' %}"></script>
</body>

</html>

```

Scripts.js

```

// Generate random room name if needed
if (!location.hash) {
  location.hash = Math.floor(Math.random() * 0xFFFFFFFF).toString(16);
}
const roomHash = location.hash.substring(1);

// TODO: Replace with your own channel ID
const drone = new ScaleDrone('GOQfsahTkQfcEyrb');
// Room name needs to be prefixed with 'observable-'
const roomName = 'observable-' + roomHash;
const configuration = {
  iceServers: [{
    urls: 'stun:stun.services.mozilla.com'
  }]
};
let room;
let pc;

function onSuccess() {};
function onError(error) {
  console.error(error);
};

drone.on('open', error => {
  if (error) {
    return console.error(error);
  }
  room = drone.subscribe(roomName);
  room.on('open', error => {
    if (error) {
      onError(error);
    }
  });
  // We're connected to the room and received an array of 'members'
  // connected to the room (including us). Signaling server is ready.
  room.on('members', members => {
    console.log('MEMBERS', members);
    // If we are the second user to connect to the room we will be creating
    the offer
    const isOfferer = members.length === 2;
    startWebRTC(isOfferer);
  });
});

// Send signaling data via Scaledrone
function sendMessage(message) {

```

```

drone.publish({
  room: roomName,
  message
});
}

function startWebRTC(isOfferer) {
  pc = new RTCPeerConnection(configuration);

  // 'onicecandidate' notifies us whenever an ICE agent needs to deliver a
  // message to the other peer through the signaling server
  pc.onicecandidate = event => {
    if (event.candidate) {
      sendMessage({'candidate': event.candidate});
    }
  };

  // If user is offerer let the 'negotiationneeded' event create the offer
  if (isOfferer) {
    pc.onnegotiationneeded = () => {
      pc.createOffer().then(localDescCreated).catch(onError);
    }
  }

  // When a remote stream arrives display it in the #remoteVideo element
  pc.ontrack = event => {
    const stream = event.streams[0];
    if (!remoteVideo.srcObject || remoteVideo.srcObject.id !== stream.id) {
      remoteVideo.srcObject = stream;
    }
  };

  navigator.mediaDevices.getUserMedia({
    audio: true,
    video: true,
  }).then(stream => {
    // Display your local video in #localVideo element
    localVideo.srcObject = stream;
    // Add your stream to be sent to the connecting peer
    stream.getTracks().forEach(track => pc.addTrack(track, stream));
  }, onError);

  // Listen to signaling data from Scaledrone
  room.on('data', (message, client) => {
    // Message was sent by us
    if (client.id === drone.clientId) {
      return;
    }

    if (message.sdp) {
      // This is called after receiving an offer or answer from another peer
      pc.setRemoteDescription(new RTCSessionDescription(message.sdp), () => {
        // When receiving an offer lets answer it
        if (pc.remoteDescription.type === 'offer') {
          pc.createAnswer().then(localDescCreated).catch(onError);
        }
      }, onError);
    }
  });
}

```

```

    } else if (message.candidate) {
        // Add the new ICE candidate to our connections remote description
        pc.addIceCandidate(
            new RTCIceCandidate(message.candidate), onSuccess, onError
        );
    }
    });
}

function localDescCreated(desc) {
    pc.setLocalDescription(
        desc,
        () => sendMessage({'sdp': pc.localDescription}),
        onError
    );
}

```

Admin.py

```

from django.contrib import admin

# Register your models here.

```

Apps.py

```

from django.apps import AppConfig

class MyappConfig(AppConfig):
    name = 'chatApp'

```

models.py

```

from django.db import models

# Create your models here.

```

Urls.py

```

from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='chat')
]

```

views.py

```

from django.shortcuts import render
import os

def home(request):
    return render(request, 'home.html')

```

