**AIDOC – *Pioneer in Mobile Healthcare***

**A PROJECT REPORT**

***Submitted by***

**Daksh Patel 171310132025**

**Jainam Patel 171310132029**

**Meet Patel 171310132034**

**Tirth Patel 171310132040**

***In fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

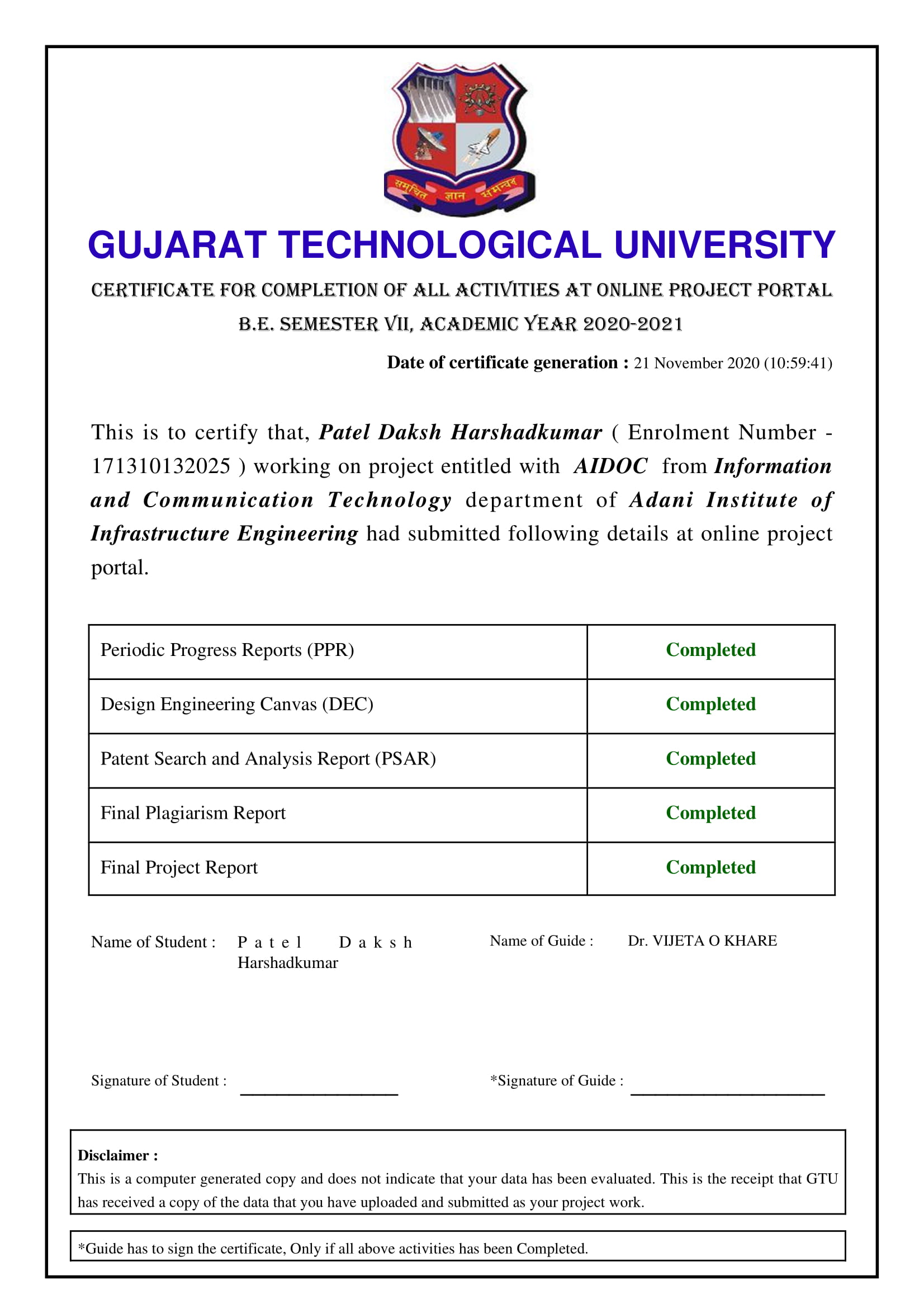
**Information and Communication Technology**

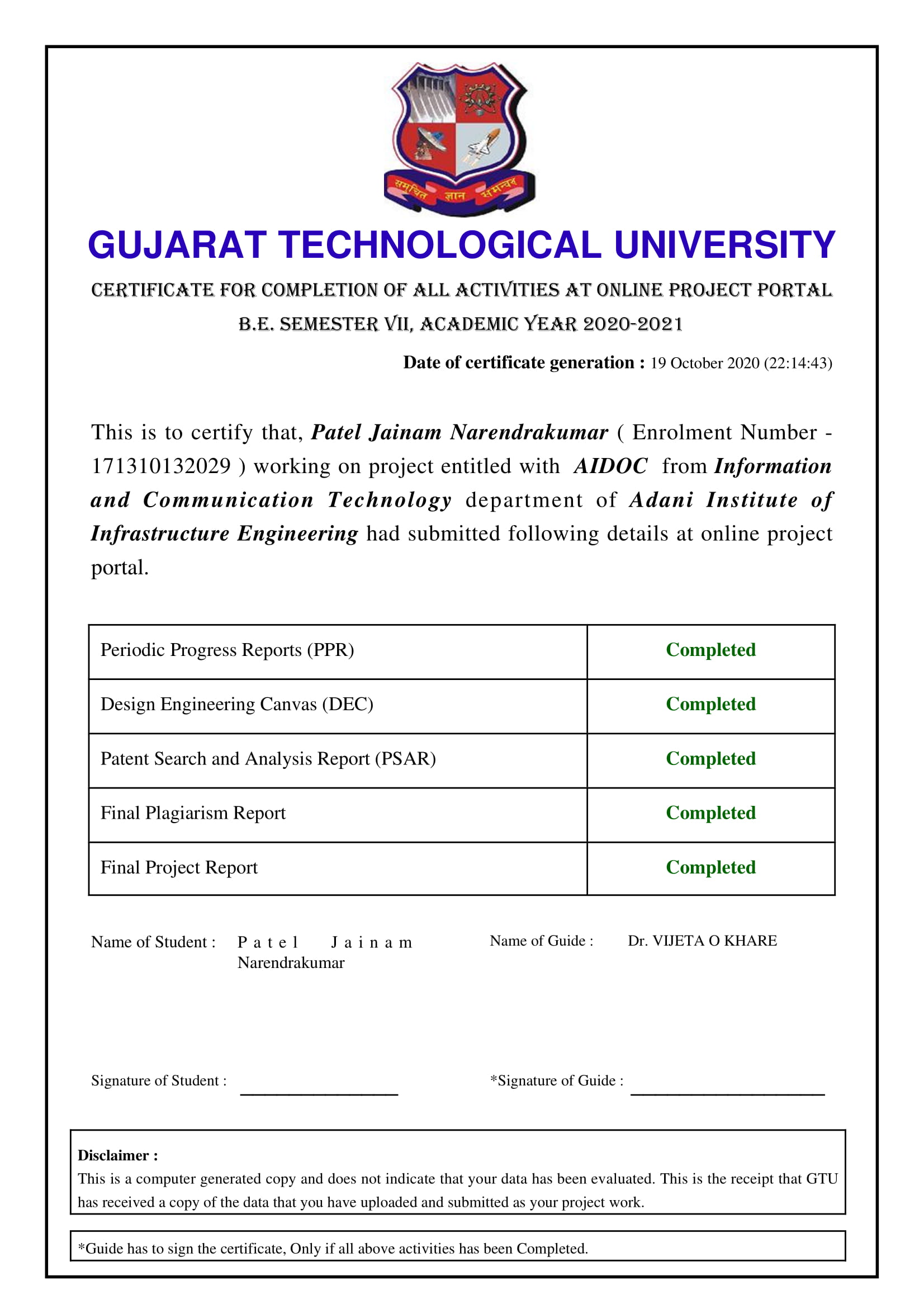


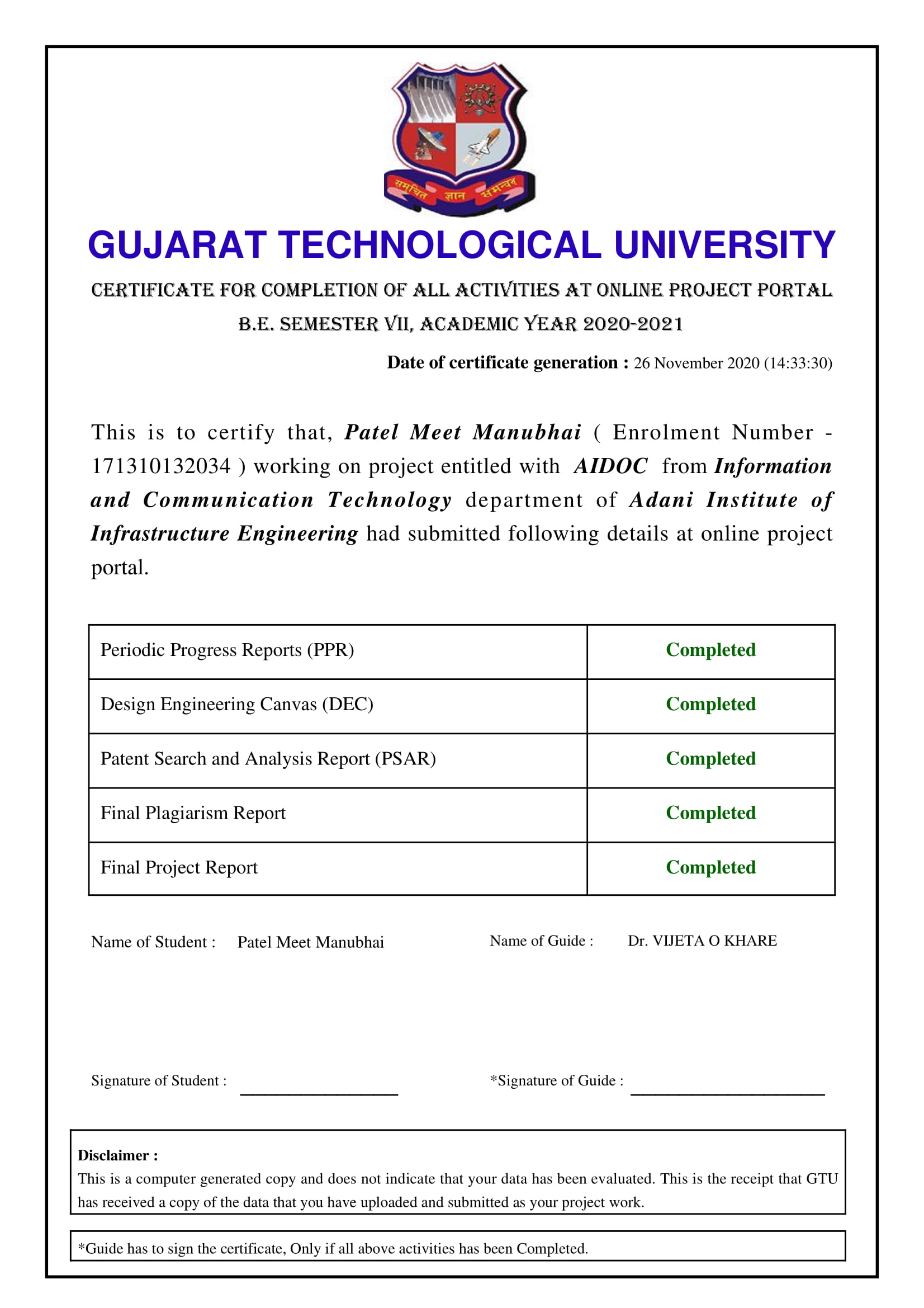
**Adani Institute of Infrastructure Engineering, Ahmedabad**

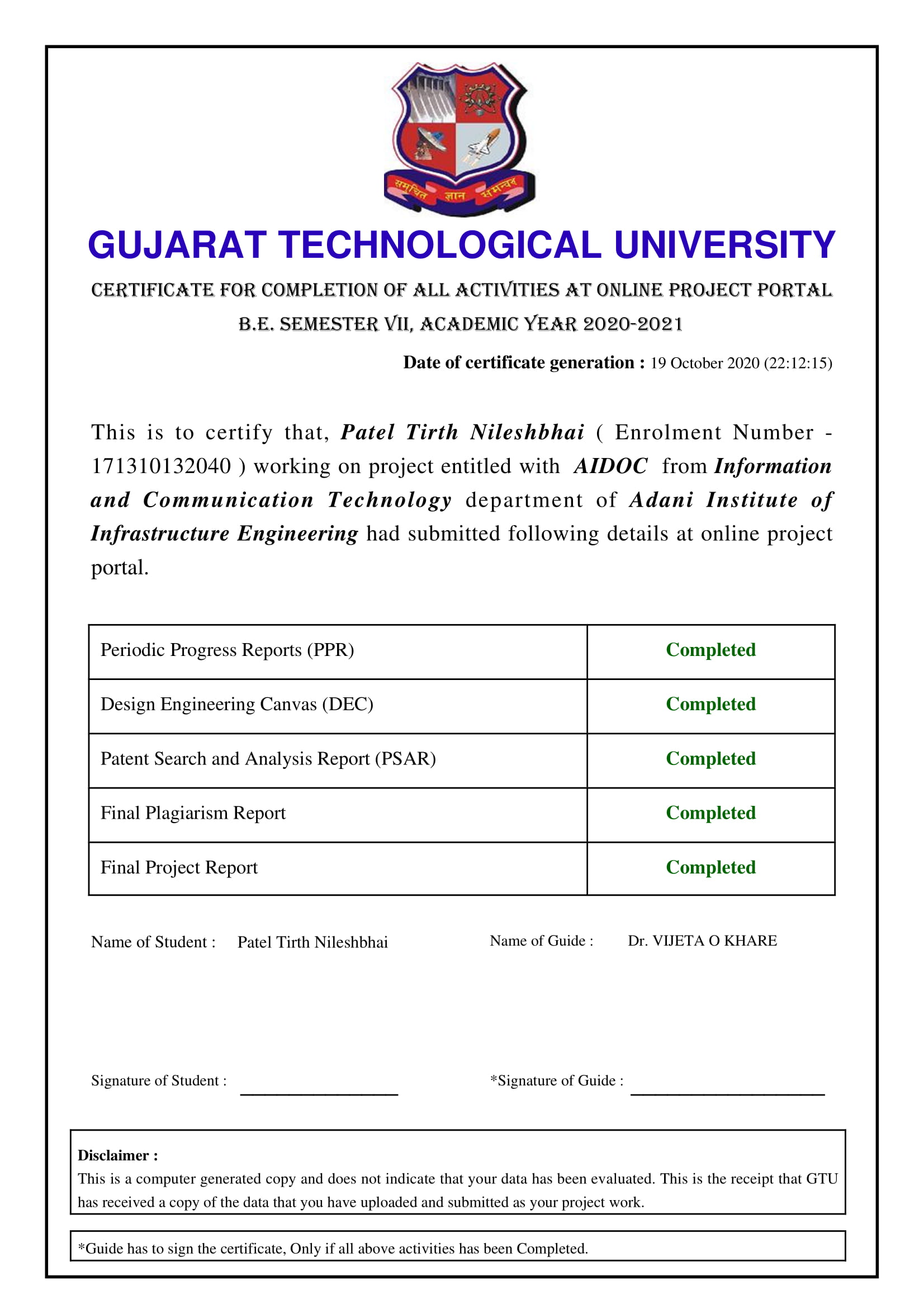
**Gujarat Technological University, Ahmedabad**

**Academic Year (2020-21)**

****

****

****

****

**CERTIFICATE**

**Date:**

This is to certify that the Project Work entitled “**AIDOC**” has been carried out by **DAKSH HARSHADKUMAR PATEL (171310132025)** under my guidance in partial fulfilment of the degree of Bachelor of Engineering in **INFORMATION AND COMMUNICATION TECHNOLOGY(ICT) (7th Semester)** of Gujarat Technological University, Ahmedabad during the academic year 2020-21.



**Dr. Vijeta Khare Dr. Hitesh Chhikaniwala**

**(Internal Guide) Head of Department, ICT**

**AIIE AIIE**

**CERTIFICATE**

**Date:**

This is to certify that the Project Work entitled “**AIDOC**” has been carried out by **JAINAM NARENDRAKUMAR PATEL (171310132029)** under my guidance in partial fulfilment of the degree of Bachelor of Engineering in **INFORMATION AND COMMUNICATION TECHNOLOGY(ICT) (7th Semester)** of Gujarat Technological University, Ahmedabad during the academic year 2020-21.



**Dr. Vijeta Khare Dr. Hitesh Chhikaniwala**

**(Internal Guide) Head of Department, ICT**

**AIIE AIIE**

**CERTIFICATE**

**Date:**

This is to certify that the Project Work entitled “**AIDOC**” has been carried out by **MEET MANUBHAI PATEL (171310132034)** under my guidance in partial fulfilment of the degree of Bachelor of Engineering in **INFORMATION AND COMMUNICATION TECHNOLOGY(ICT) (7th Semester)** of Gujarat Technological University, Ahmedabad during the academic year 2020-21.



**Dr. Vijeta Khare Dr. Hitesh Chhikaniwala**

**(Internal Guide) Head of Department, ICT**

**AIIE AIIE**

**CERTIFICATE**

**Date:**

This is to certify that the Project Work entitled “**AIDOC**” has been carried out by **TIRTH NILESHBHAI PATEL (171310132040)** under my guidance in partial fulfilment of the degree of Bachelor of Engineering in **INFORMATION AND COMMUNICATION TECHNOLOGY(ICT) (7th Semester)** of Gujarat Technological University, Ahmedabad during the academic year 2020-21.



**Dr. Vijeta Khare Dr. Hitesh Chhikaniwala**

**(Internal Guide) Head of Department, ICT**

**AIIE AIIE**

**DECLARATION OF ORIGINALITY**

We hereby certify that we are the sole author of this report and that neither any part of this work nor the whole of the work has been submitted for a degree to any other University or Institution.

We certify that, to the best of our knowledge, our work does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in our report, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that we have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Indian Copyright Act, we certify that we have obtained a written permission from the copyright owner(s) to include such material(s) in our work and have included copies of such copyright clearances to our appendix.

We declare that this is a true copy of our report, including any final revisions, as approved by our supervisor.

Date:

Place: Ahmedabad, Gujarat

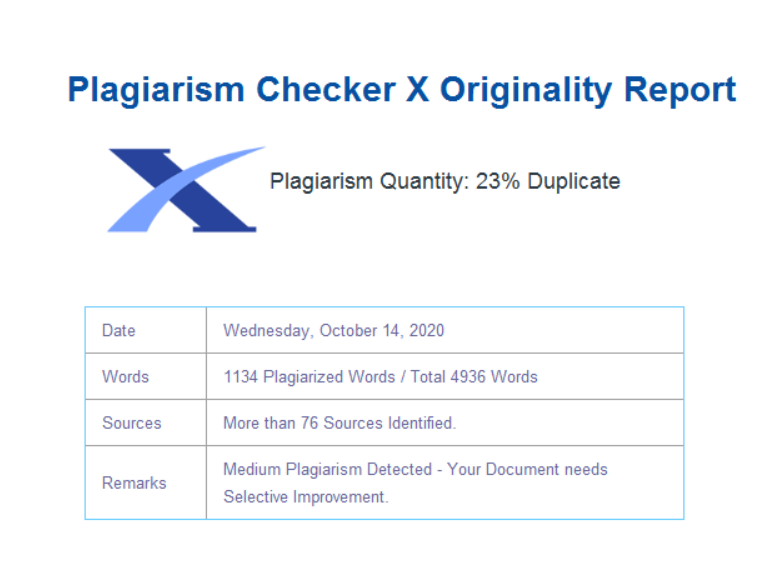
Signature of Student

|  |  |
| --- | --- |
| **DAKSH PATEL** | **171310132025** |
| **JAINAM PATEL** | **171310132029** |
| **MEET PATEL** | **171310132034** |
| **TIRTH PATEL** | **171310132040** |

Signature of Guide

Dr. Vijeta Khare

**PLAGIARISM SEARCH REPORT/CERTIFICATE**

****

**ACKNOWLEDGEMENT**

We consider it privilege to be associated with Adani Institute of Infrastructure Engineering, Ahmedabad in this academic endeavour. We are heartily thankful to all professors and all staff members of our college for the co-operation and keep interests extended by them throughout our BE course. It is the base that they have built, which sustains such good job.

Presently, we would like to acknowledge and thanks to our guid Dr. Vijeta Khare, ICT department, AIIE who guided us throughout our project and helped us to create our project, her ideas and help proved to be extremely valuable during the creation of our project. We are extremely highly indebted to our Department Head Dr. Hitesh Chhikaniwala HOD of ICT department, AIIE for providing us this opportunity to prepare our project and give us stage to develop our mind.

We would also thankful to all staff members of ICT department for their kind support at various stages of the work. It is beyond the reach of our languages to express our sincere gratitude to our parents for their infinite inspiration and sacrifices for the benefit of our academic career.

|  |  |
| --- | --- |
| **DAKSH PATEL** | **171310132025** |
| **JAINAM PATEL** | **171310132029** |
| **MEET PATEL** | **171310132034** |
| **TIRTH PATEL** | **171310132040** |

**INDEX**

## **Chapter 1: Introduction**

* 1. Problem Summary
  2. Aim of Project
  3. Objective of Project
  4. Prior Art Research
  5. Material / Tools required

1. **Chapter 2: Design- Analysis, Design Methodology and Implementation Strategy**
   1. AEIOU Canvas
   2. Empathy Mapping Canvas
   3. Ideation Canvas
   4. Product Development Canvas
2. **Chapter 3: Implementation**
   1. Implementation
3. **Chapter 4: Summary**
   1. Summary of the Results
   2. Future Work
4. **References**

**CH - 1 : Introduction**

**1.1 Problem Summary:-**

In this project, we are going to develop cross platform mobile application to detect diseases related to images such as Pneumonia can be detected from chest X-ray images, Fracture can be detected from Bone X-ray, Diabetic retinopathy can be detected from eye images, COVID-19 can be detected from chest X-ray.

Our Project mainly outlines three Modules; Administrator, Doctor, Stakeholder.

App Highlights following activities in broader way:-

i) Registration (Sign Up) of User

ii) User Sign In

iii) Displaying Medical Articles

iv) Prediction of desired disease

v) Generation of report

vi) Chat-bot

vii) Recommending List of Doctors in given Proximity

**1.2 Aim of Project:-**

Creating a tool which help individuals to prognose diseases proactively with minimal clicks. Suggesting set of world class doctors and medical facilities which could help them if severity of discrepancy rises.

**1.3 Objectives of Project:-**

Developing a Cross platform application which would leverage the AI and revolutionize the way in which medical industry are operating today.

**1.4 Prior Art Search:-**

**1) An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare**

Published Year: March 2019

Authors: Okeke Stephen, Mangal Sain, Uchenna Joseph Maduh, Doun Jeong

Abstract:

This study proposes a convolutional neural network model trained from scratch to classify and detect the presence of pneumonia from a collection of chest X-ray image samples. Unlike other methods that rely solely on transfer learning approaches or traditional handcrafted techniques to achieve a remarkable classification performance, we constructed a convolutional neural network model from scratch to extract features from a given chest X-ray image and classify it to determine if a person is infected with pneumonia. This model could help mitigate the reliability and interpretability challenges often faced when dealing with medical imagery. Unlike other deep learning classification tasks with sufficient image repository, it is difficult to obtain a large amount of pneumonia dataset for this classification task; therefore, we deployed several data augmentation algorithms to improve the validation and classification accuracy of the CNN model and achieved remarkable validation accuracy.

**2) Efficient Pneumonia Detection in Chest Xray Images Using Deep Transfer Learning**

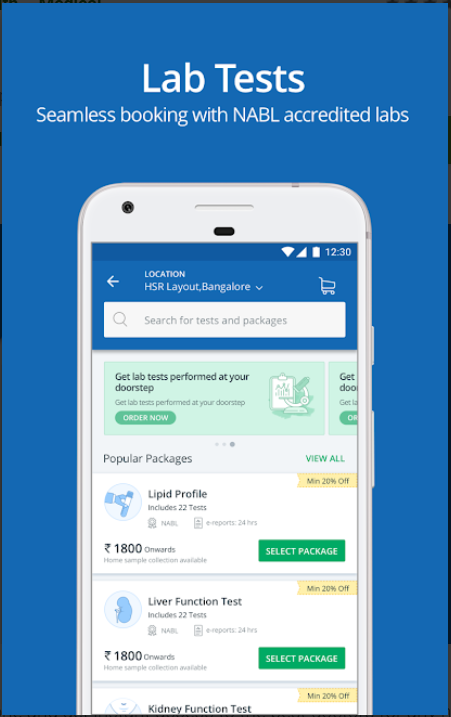
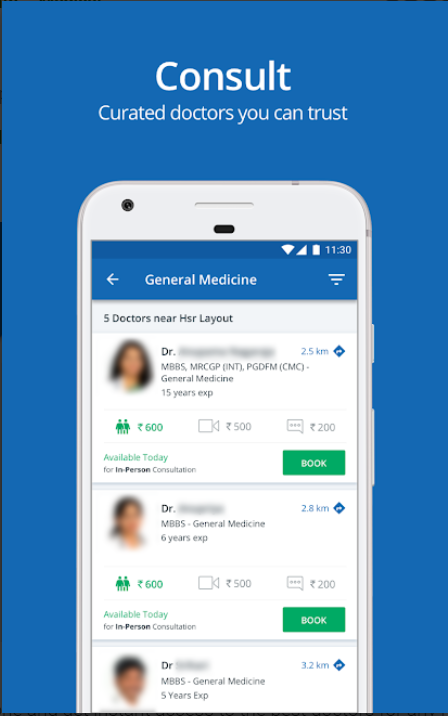
Published Year: June 2020

Authors: Mohammad Farukh Hashmi, Satyarth Katiyar, Avinash G Keskar, Neeraj Dhanraj Bokde, Zong Woo Geem5

Abstract:

Pneumonia causes the death of around 700,000 children every year and affects 7% of the global population. Chest X-rays are primarily used for the diagnosis of this disease. However, even for a trained radiologist, it is a challenging task to examine chest X-rays. There is a need to improve the diagnosis accuracy. In this work, an efficient model for the detection of pneumonia trained on digital chest X-ray images is proposed, which could aid the radiologists in their decision-making process. A novel approach based on a weighted classifier is introduced, which combines the weighted predictions from the state-of-the-art deep learning models such as ResNet18, Xception, InceptionV3, DenseNet121, and MobileNetV3 in an optimal way. This approach is a supervised learning approach in which the network predicts the result based on the quality of the dataset used. Transfer learning is used to fine-tune the deep learning models to obtain higher training and validation accuracy. Partial data augmentation techniques are employed to increase the training dataset in a balanced way. The proposed weighted classifier is able to outperform all the individual models. Finally, the model is evaluated, not only in terms of test accuracy, but also in the AUC score. The final proposed weighted classifier model is able to achieve a test accuracy of 98.43% and an AUC score of 99.76 on the unseen data from the Guangzhou Women and Children’s Medical Centre pneumonia dataset. Hence, the proposed model can be used for a quick diagnosis of pneumonia and can aid the radiologists in the diagnosis process.

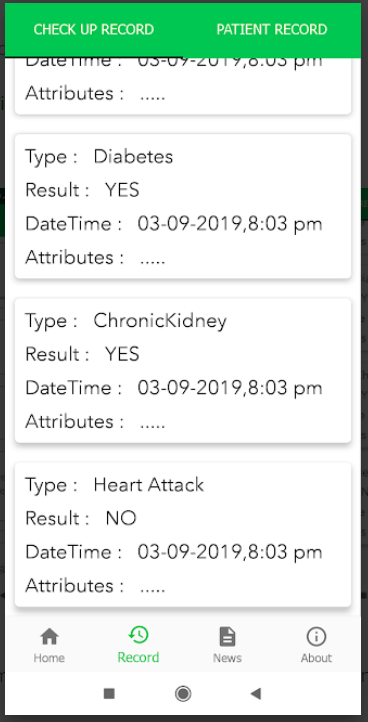
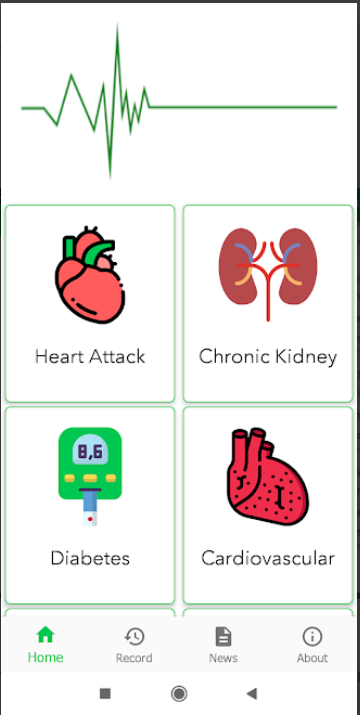
**3) Tata Health App: -**

Tata Health app allows instant access to the best doctors for any health issues, online lab test booking, doctor appointment booking, specialty consultation for dermatological issues, paediatrician advice, gynaecology, and much more

What we are recreating is an application which will utilize the consultancy functionality with AI Base Diagnosis of various diseases.

**4) Disease Detector App:**

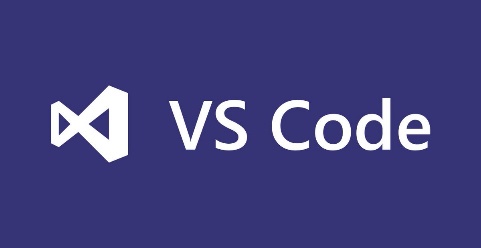
 

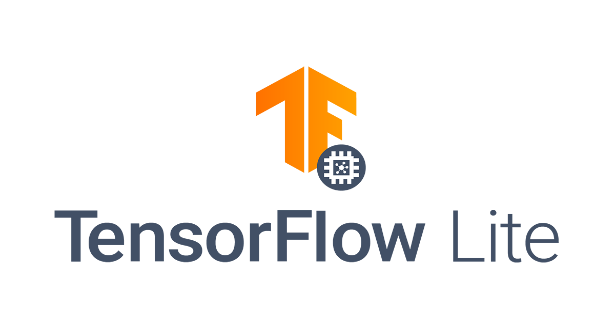
Disease Detector is especially intended to improve healthcare system for hospitals and medical examination by using AI technology.

Above Application Detects the disease based on Q/A and dosen’t detect diseases related to Images which our application can perform with ease.

**1.5 Materials/Tools Used:-**

Tools used by our Developer Team is as follow:-



**CH - 2 : Design: Analysis, Design Methodology and Implementation Strategy.**

**2.1 AEIOU Canvas:-**

AEIOU stands for 5 elements: Activity, Environment, Interaction, Object, and User.

• Activities are goal-directed sets of actions—paths towards things people want to accomplish. What are the modes people work in, and the specific activities and processes they go through?

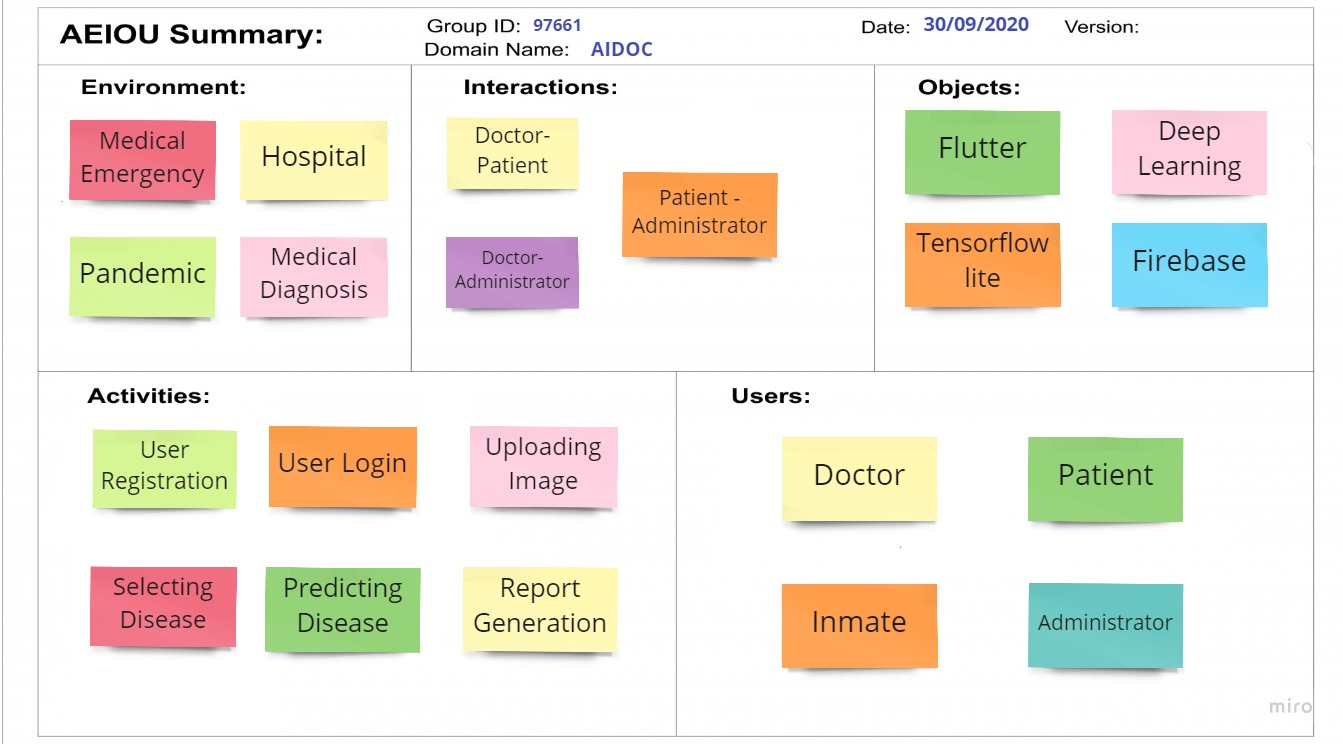
• Environments include the entire arena where activities take place. What is the character and function of the space overall, of each individual's spaces, and of shared spaces?

• Interactions are between a person and someone or something else; they are the building blocks of activities. What is the nature of routine and special interactions between people, between people and objects in their environment, and across distances?

• Objects are building blocks of the environment, key elements sometimes put to complex or unintended uses (thus changing their function, meaning and context). What are the objects and devices people have in their environments and how do they relate to their activities?

• Users are the people whose behaviours, preferences, and needs are being observed. Who is there? What are their roles and relationships? What are their values and prejudices?

In AEIOU canvas, our activities include Registration Activity of User, Login Activity of user, after login of user: Uploading image from gallery which is to be prognosed. Once the Image is selected, App predicts the confidence of desired disease. After prediction, User can write the file into external storage of smartphone. The Environment part of canvas includes Medical Emergency, Medical Diagnosis, etc. The objects used in our project are Flutter, Deep Learning, TensorFlow lite, Firebase. Users directly or indirectly related to our project are Doctors, Patient, Inmate, App Administrator



**2.2 Empathy Mapping Canvas**

An Empathy map is a research tool which allows you to really get inside the head of your customer. It is mapped out with the customer’s thoughts and feelings, their primary senses of seeing and hearing as well as their pains and gains. It can be used to research your users or different segments of your customer base.

Details of our Empathy Mapping Canvas:

User: Doctor, Inmate, App administrator, Patient

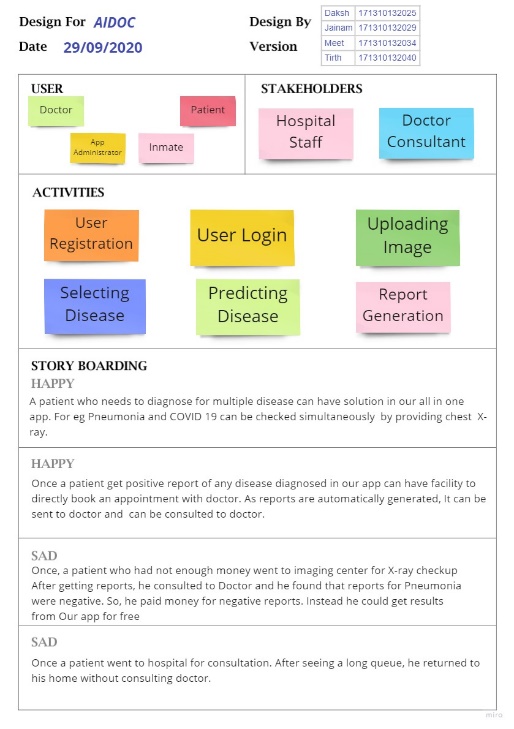
Stakeholders: Hospital Staff, Doctor Consultant

Activities: User registration, User login, Uploading image, Selection of Disease, Prediction of Disease, Report Generation

Story Boarding:

Happy: A patient who needs to diagnose for multiple disease can have solution in our all in one app. For eg Pneumonia and COVID 19 can be checked simultaneously by providing chest X-ray.

Sad: Once, a patient who had not enough money went to imaging center for X-ray checkupAfter getting reports, he consulted to Doctor and he found that reports for Pneumonia were negative. So, he paid money for negative reports. Instead he could get results from Our app for free



**2.3 Ideation Canvas**

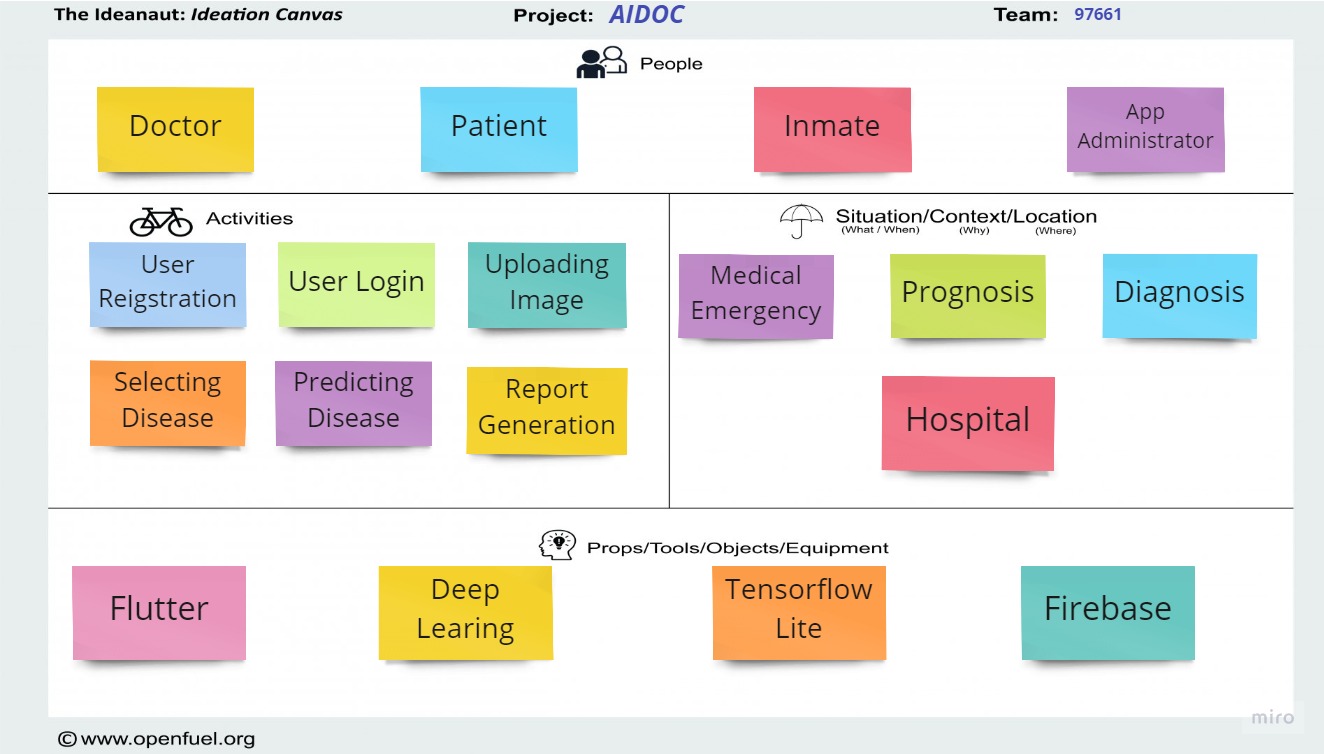
This was the second step of our project. From this canvas, we got the idea that who are the people involved. We have inspected which type of activities are related to our project? What are the suitable locations for our activities? At last we found the possible solutions for concerned problems.

Details of Our Ideation Canvas

People: Doctor, Patient, Inmate, App administrator

Situation /Context/Location: Medical Emergency, Prognosis, Diagnosis,Hospital

Props/Tools/Obejcts/Equipment: Flutter, Deep Learning, Tensorflow Lite, Firebase



**2.4 Product Development Canvas**

Product Canvas will require us to start building a structutre of our product around the emotional need of the user. Building solution around the emotive needs of user is central to design thinking.

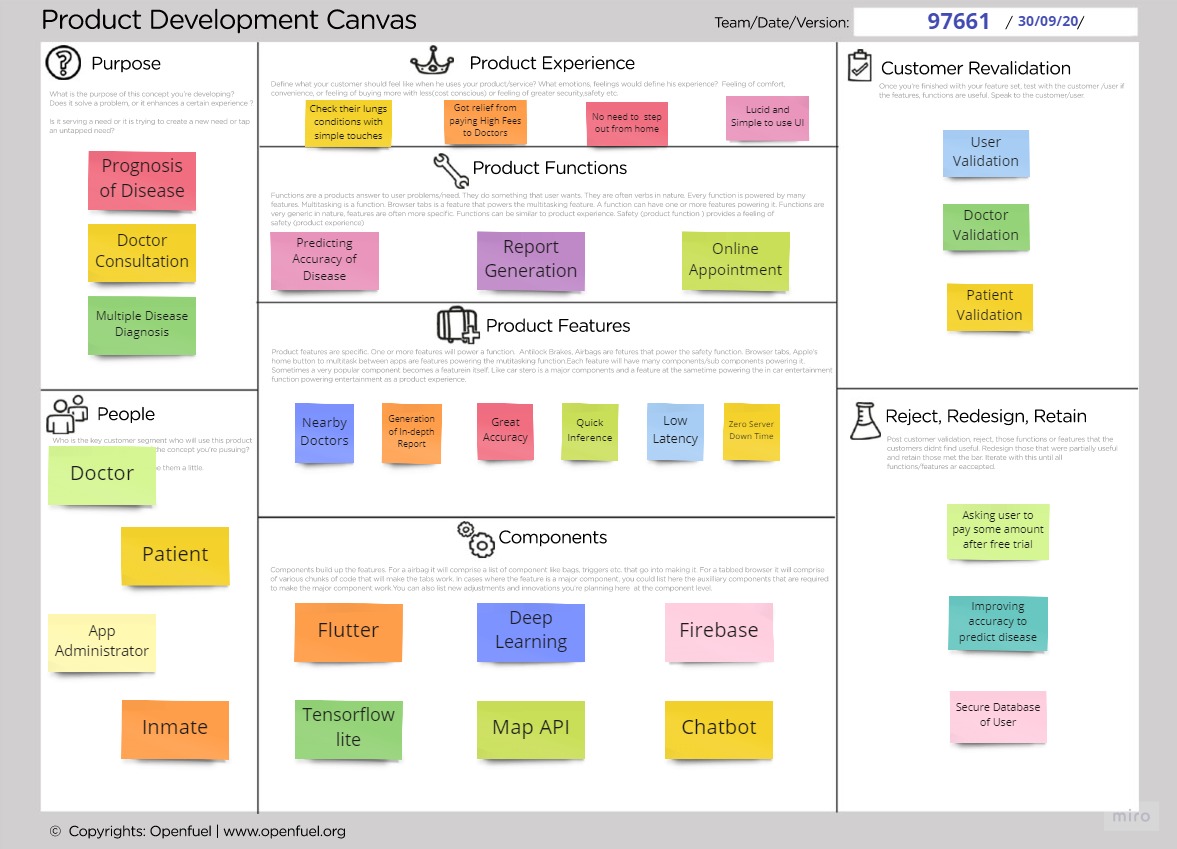
Designing Product Strategy had taught us a powerful and practical technique used by thousands of product people to design and execute a new product idea or even analyze and rejuvenate an existing one. From possible solutions we got the idea about the product which should be developed in order to solve the problems.

Details of Product Development Canvas:

Product Experience: Check user lungs condititon with minimal clicks, Got relief from paying High fees to Doctor,No need to step out from home, Lucid and Simple to use UI

Product Functions: Predicting accuracy of disease, Report Generation, Online Appointment,

Product Features; Nearby doctors, Great Accuracy, Quick Interface, Low Latency, Zero Server Down time



**CH - 3 : Implementation**

**3.1 Implementation**

**1) Integrating CNN model to our App using Machine Teachable by Google**

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone.

Working of Machine Teachable:

1. Gather

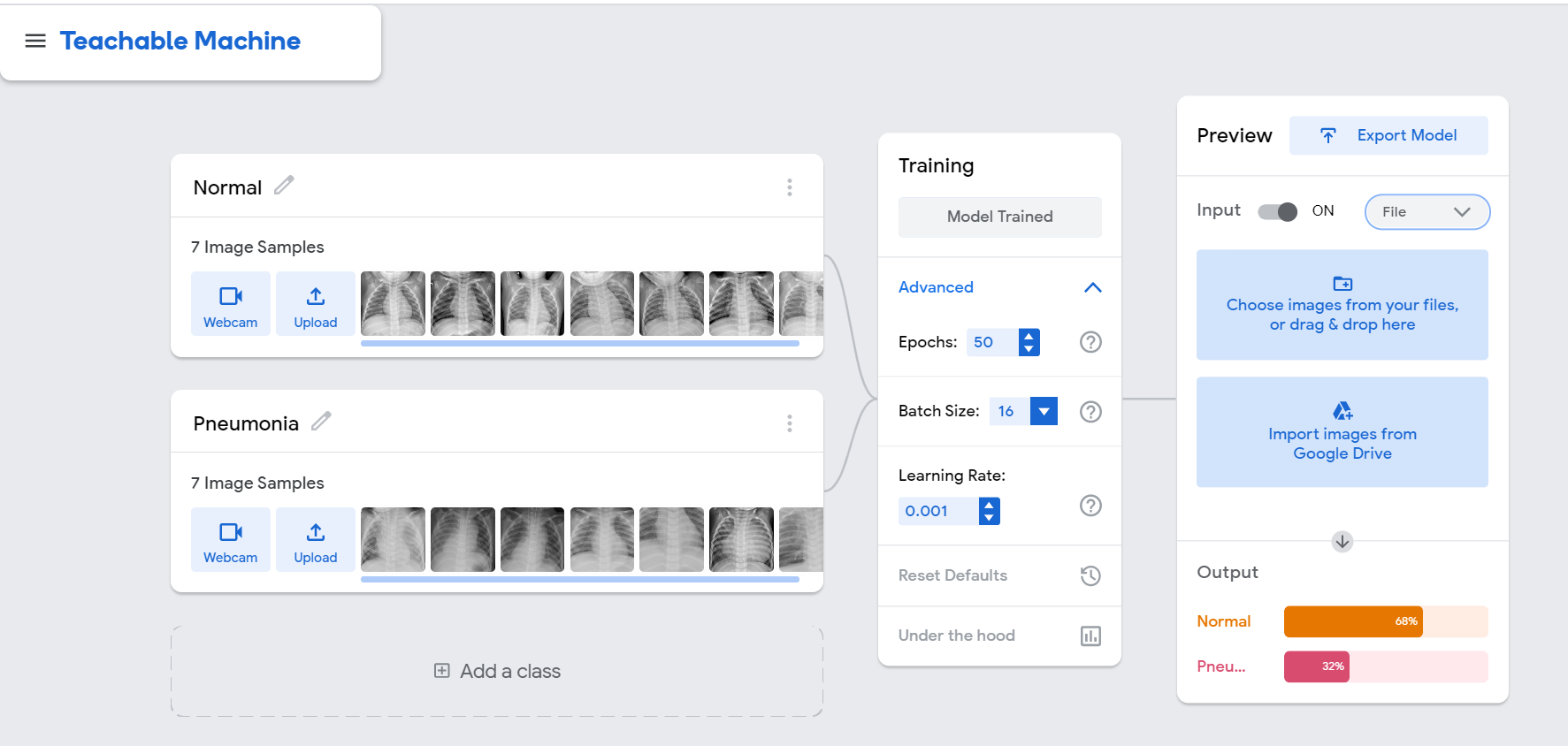
Gather and group your examples into classes, or categories, that you want the computer to learn.

1. Train

Train your model, then instantly test it out to see whether it can correctly classify new examples.

1. Export

Export your model for your projects: sites, apps, and more. You can download your model or host it online for free.



We have used the dataset from Kaggle which contains 5863 image of chest x-ray of two categories i.e. Pneumonia and Normal.

Link for dataset: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

We have exported tflite file which is required for integration of deep learning model with smartphone. TensorFlow Lite is an open source deep learning framework for on-device inference.

**2) Registration and Login of User using Firebase Console**

Setup of Firebase:

1. App registration on Firebase Console

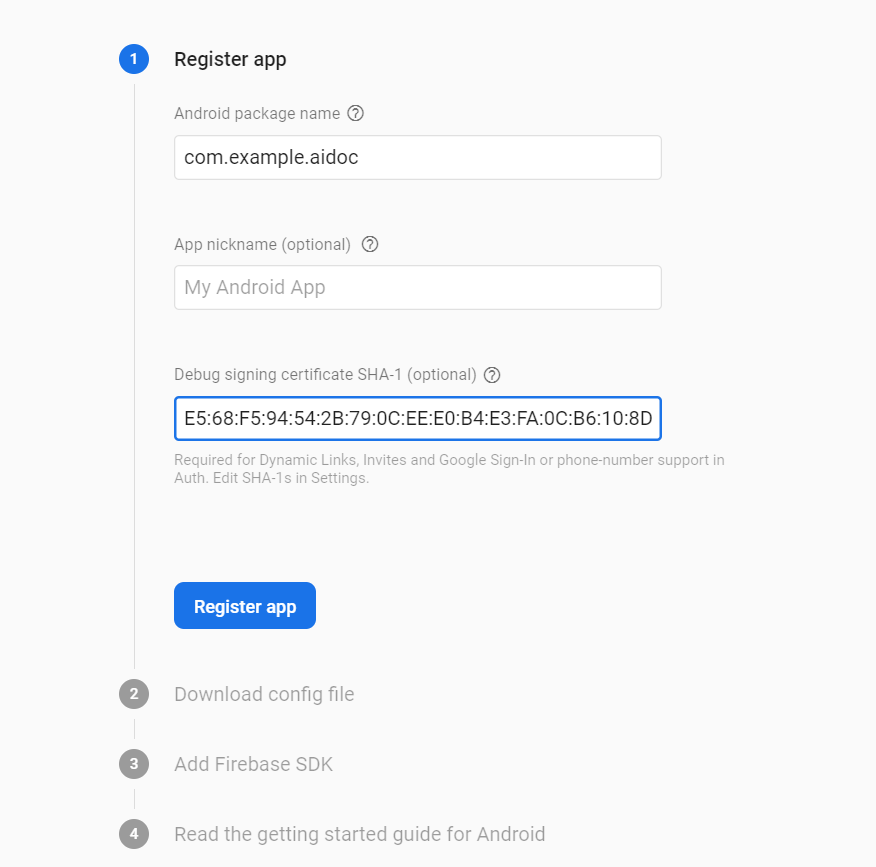
Android Package name: package name can be obtained from project directory in

app\build.gradle

1. Debug signing certificate SHA -1

It can be obtained by code in terminal written below:

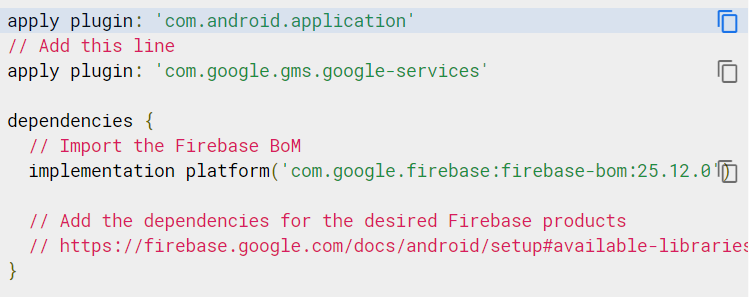
C:\Users\meet0\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android



1. After registration of app, it allow us to download config file named “google-services.json”.
2. Modify your project level build gradle



1. Modify app level build gradle



**Registration of User:**

User Registers himself by providing necessary details, which are highly secured. The fields which are mandatory to provide in order to get il with our service is as follow:-

a). User Name

b). E-mail

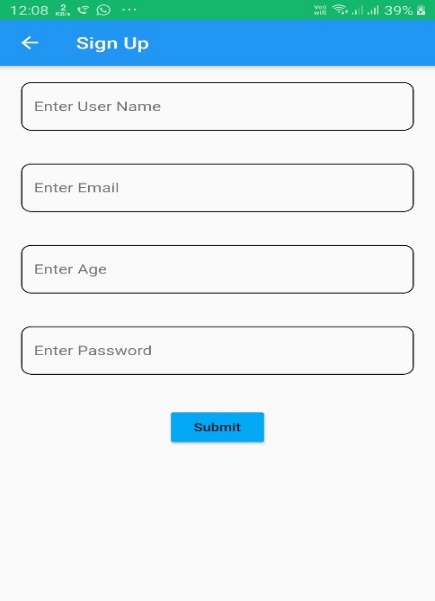
c). Age

d). Password

**User registration code**:

**import 'package:firebase\_auth/firebase\_auth.dart'**;  
**import 'package:firebase\_database/firebase\_database.dart'**;  
**import 'package:flutter/material.dart'**;  
  
**import 'home.dart'**;  
  
**class** EmailSignUp **extends** StatefulWidget {  
 @override  
 \_EmailSignUpState createState() => \_EmailSignUpState();  
}  
  
**class** \_EmailSignUpState **extends** State<EmailSignUp> {  
 bool **isLoading** = **false**;  
 **final \_formKey** = GlobalKey<FormState>();  
 FirebaseAuth **firebaseAuth** = FirebaseAuth.*instance*;  
 DatabaseReference **dbRef** =  
 FirebaseDatabase.*instance*.reference().child(**"Users"**);  
 TextEditingController **emailController** = TextEditingController();  
 TextEditingController **nameController** = TextEditingController();  
 TextEditingController **passwordController** = TextEditingController();  
 TextEditingController **ageController** = TextEditingController();  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 appBar: AppBar(title: Text(**"Sign Up"**)),  
 body: Form(  
 key: **\_formKey**,  
 child: SingleChildScrollView(  
 child: Column(children: <Widget>[  
 Padding(  
 padding: EdgeInsets.all(20.0),  
 child: TextFormField(  
 controller: **nameController**,  
 decoration: InputDecoration(  
 labelText: **"Enter User Name"**,  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 ),  
 *// The validator receives the text that the user has entered.* validator: (value) {  
 **if** (value.**isEmpty**) {  
 **return 'Enter User Name'**;  
 }  
 **return null**;  
 },  
 ),  
 ),  
 Padding(  
 padding: EdgeInsets.all(20.0),  
 child: TextFormField(  
 controller: **emailController**,  
 decoration: InputDecoration(  
 labelText: **"Enter Email"**,  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 ),  
 *// The validator receives the text that the user has entered.* validator: (value) {  
 **if** (value.**isEmpty**) {  
 **return 'Enter an Email Address'**;  
 } **else if** (!value.contains(**'@'**)) {  
 **return 'Please enter a valid email address'**;  
 }  
 **return null**;  
 },  
 ),  
 ),  
 Padding(  
 padding: EdgeInsets.all(20.0),  
 child: TextFormField(  
 controller: **ageController**,  
 decoration: InputDecoration(  
 labelText: **"Enter Age"**,  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 ),  
 *// The validator receives the text that the user has entered.* validator: (value) {  
 **if** (value.**isEmpty**) {  
 **return 'Enter Age'**;  
 }  
 **return null**;  
 },  
 ),  
 ),  
 Padding(  
 padding: EdgeInsets.all(20.0),  
 child: TextFormField(  
 obscureText: **true**,  
 controller: **passwordController**,  
 decoration: InputDecoration(  
 labelText: **"Enter Password"**,  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 ),  
 *// The validator receives the text that the user has entered.* validator: (value) {  
 **if** (value.**isEmpty**) {  
 **return 'Enter Password'**;  
 } **else if** (value.**length** < 6) {  
 **return 'Password must be atleast 6 characters!'**;  
 }  
 **return null**;  
 },  
 ),  
 ),  
 Padding(  
 padding: EdgeInsets.all(20.0),  
 child: **isLoading** ? CircularProgressIndicator()  
 : RaisedButton(  
 color: Colors.*lightBlue*,  
 onPressed: () {  
 **if** (**\_formKey**.**currentState**.validate()) {  
 setState(() {  
 **isLoading** = **true**;  
 });  
 registerToFb();  
 }  
 },  
 child: Text(**'Submit'**),  
 ),  
 )  
 ]))));  
 }  
  
 **void** registerToFb() {  
 **firebaseAuth** .createUserWithEmailAndPassword(  
 email: **emailController**.**text**, password: **passwordController**.**text**)  
 .then((result) {  
 **dbRef**.child(result.**user**.**uid**).set({  
 **"email"**: **emailController**.**text**,  
 **"age"**: **ageController**.**text**,  
 **"name"**: **nameController**.**text** }).then((res) {  
 **isLoading** = **false**;  
 */\*Navigator.pushReplacement(  
 context,  
 MaterialPageRoute(builder: (context) => Home(uid: result.user.uid)),  
 );\*/* });  
 }).catchError((err) {  
 showDialog(  
 context: **context**,  
 builder: (BuildContext context) {  
 **return** AlertDialog(  
 title: Text(**"Error"**),  
 content: Text(err.message),  
 actions: [  
 FlatButton(  
 child: Text(**"Ok"**),  
 onPressed: () {  
 Navigator.*of*(context).pop();  
 },  
 )  
 ],  
 );  
 });  
 });  
 }  
  
 @override  
 **void** dispose() {  
 **super**.dispose();  
 **nameController**.dispose();  
 **emailController**.dispose();  
 **passwordController**.dispose();  
 **ageController**.dispose();  
 }  
}

**Output:-**



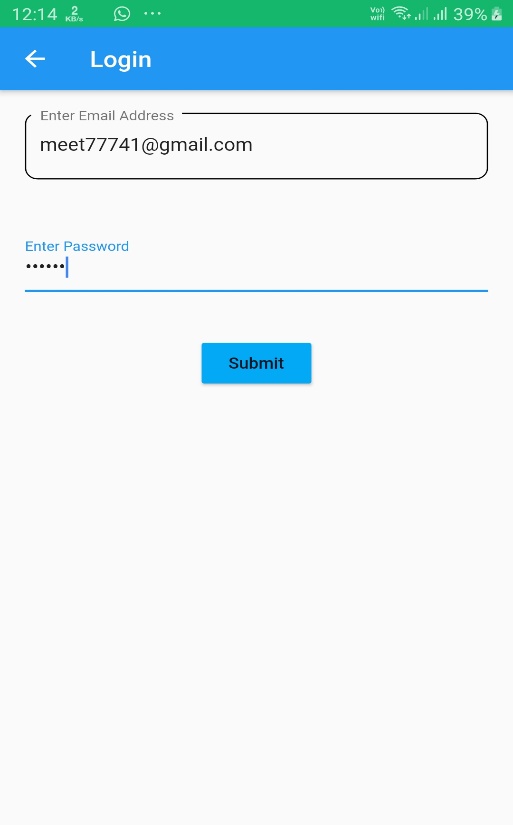
**User Login :**

User credentials are verified with firebase when user enters the details for login. If the credentials are matched, then User can login and page will be redirected to Home screen of app or else an exception will be thrown.

**User login code:**

**import 'package:firebase\_auth/firebase\_auth.dart'**;  
**import 'package:flutter/material.dart'**;  
  
**import 'home.dart'**;  
  
**class** EmailLogIn **extends** StatefulWidget {  
 @override  
 \_EmailLogInState createState() => \_EmailLogInState();  
}  
  
**class** \_EmailLogInState **extends** State<EmailLogIn> {  
 **final \_formKey** = GlobalKey<FormState>();  
 TextEditingController **emailController** = TextEditingController();  
 TextEditingController **passwordController** = TextEditingController();  
  
 bool **isLoading** = **false**;  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 appBar: AppBar(title: Text(**"Login"**)),  
 body: Form(  
 key: **\_formKey**,  
 child: SingleChildScrollView(  
 child: Column(children: <Widget>[  
 Padding(  
 padding: EdgeInsets.all(20.0),  
 child: TextFormField(  
 controller: **emailController**,  
 decoration: InputDecoration(  
 labelText: **"Enter Email Address"**,  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 ),  
 *// The validator receives the text that the user has entered.* validator: (value) {  
 **if** (value.**isEmpty**) {  
 **return 'Enter Email Address'**;  
 }  
 **else if**(!value.contains(**'@'**)){  
 **return 'Please enter a valid email address!'**;  
 }  
 **return null**;  
 },  
 ),  
 ),  
 Padding(  
 padding: EdgeInsets.all(20.0),  
 child: TextFormField(  
 obscureText: **true**,  
 controller: **passwordController**,  
 decoration: InputDecoration(  
 labelText: **"Enter Password"**,  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 ),  
 *// The validator receives the text that the user has entered.* validator: (value) {  
 **if** (value.**isEmpty**) {  
 **return 'Enter Password'**;  
 } **else if** (value.**length** < 6) {  
 **return 'Password must be atleast 6 characters!'**;  
 }  
 **return null**;  
 },  
 ),  
 ),  
 Padding(  
 padding: EdgeInsets.all(20.0),  
 child: **isLoading** ? CircularProgressIndicator()  
 : RaisedButton(  
 color: Colors.*lightBlue*,  
 onPressed: () {  
 **if** (**\_formKey**.**currentState**.validate()) {  
 setState(() {  
 **isLoading** = **true**;  
 });  
 logInToFb();  
 }  
 },  
 child: Text(**'Submit'**),  
 ),  
 )  
 ]))));  
 }  
  
 **void** logInToFb() {  
 FirebaseAuth.*instance* .signInWithEmailAndPassword(  
 email: **emailController**.**text**, password: **passwordController**.**text**)  
 .then((result) {  
 **isLoading** = **false**;  
 Navigator.*pushReplacement*(  
 **context**,  
 MaterialPageRoute(builder: (context) => Home(uid: result.**user**.**uid**)),  
 );  
 }).catchError((err) {  
 print(err.message);  
 showDialog(  
 context: **context**,  
 builder: (BuildContext context) {  
 **return** AlertDialog(  
 title: Text(**"Error"**),  
 content: Text(err.message),  
 actions: [  
 FlatButton(  
 child: Text(**"Ok"**),  
 onPressed: () {  
 Navigator.*of*(context).pop();  
 },  
 )  
 ],  
 );  
 });  
 });  
 }  
}

**Output:-**



**3) Preparing our own Custom deep learning model and converting it to tflite**

For trial purpose, we had used machine teachable to train our model. But in this step, we have created our own custom deep learning model using transfer learning. We have used inception v3 model. After training, model is saved as .h5 file in which weights and bias are stored. But due to combability issue, It directly cannot be integrated with app. So, we need to convert our .h5 model to .tflite file which is compatible with our smartphones.

We have used the dataset from Kaggle which contains 5863 image of chest x-ray of two categories i.e. Pneumonia and Normal.

Link for dataset: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

***I.* CNN model:**

import os

os.environ['KAGGLE\_USERNAME'] = 'meetpatel07'

os.environ['KAGGLE\_KEY'] = '9a6c73b1674993e844a85662b4eb2471'

!kaggle datasets download -d paultimothymooney/chest-xray-pneumonia

!unzip chest-xray-pneumonia.zip

!unzip chest\_xray.zip

"""## Checking out the data/images"""

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

training\_dir = './chest\_xray/train'

validation\_dir = './chest\_xray/test'

test\_dir = './chest\_xray/val'

img = mpimg.imread(training\_dir + '/NORMAL/IM-0115-0001.jpeg')

imgplot = plt.imshow(img)

plt.show()

def plot\_images(path, labeled=False, max\_images=6):

amount = 0

fig = plt.figure(figsize=(12, 6))

for file in os.listdir(path):

if file.endswith('.jpeg'):

if amount == max\_images:

break

img = mpimg.imread(os.path.join(path, file))

plt.subplot(231+amount)

if labeled:

plt.title(file.split('\_')[1])

imgplot = plt.imshow(img)

amount += 1

plot\_images(training\_dir + '/NORMAL')

plot\_images(training\_dir + '/PNEUMONIA', labeled=True)

"""## Train the model"""

import keras

from keras.applications.inception\_v3 import InceptionV3

from keras.preprocessing.image import ImageDataGenerator

rescale = 1./255.0

target\_size = (150, 150)

batch\_size = 32

class\_mode = 'categorical'

train\_datagen = ImageDataGenerator(

width\_shift\_range=0.1,

height\_shift\_range=0.1,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

rescale=rescale

)

train\_generator = train\_datagen.flow\_from\_directory(

training\_dir,

target\_size=target\_size,

class\_mode=class\_mode,

batch\_size=batch\_size

)

validation\_datagen = ImageDataGenerator(rescale=rescale)

validation\_generator = validation\_datagen.flow\_from\_directory(

validation\_dir,

target\_size=target\_size,

class\_mode=class\_mode,

batch\_size=batch\_size

)

test\_datagen = ImageDataGenerator(rescale=rescale)

test\_generator = test\_datagen.flow\_from\_directory(

test\_dir,

target\_size=target\_size,

class\_mode=class\_mode,

batch\_size=1

)

def get\_model():

base\_model = InceptionV3(weights='imagenet', include\_top=False)

x = base\_model.output

x = keras.layers.GlobalAveragePooling2D()(x)

x = keras.layers.Dense(1024, activation='relu')(x)

x = keras.layers.BatchNormalization()(x)

predictions = keras.layers.Dense(2, activation='softmax')(x)

model = keras.models.Model(inputs=base\_model.inputs, outputs=predictions)

for layer in base\_model.layers:

layer.trainable = False

model.summary()

return model

model = get\_model()

model.compile(keras.optimizers.Adam(0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit\_generator(

train\_generator,

steps\_per\_epoch=len(train\_generator),

epochs=5,

verbose=1,

validation\_data=validation\_generator,

validation\_steps=len(validation\_generator)

)

"""## Evaluate the model"""

result = model.evaluate\_generator(test\_generator, steps=len(test\_generator), verbose=1)

print('Loss:', result[0])

print('Accuracy:', result[1])

y\_pred\_batches = []

y\_true\_batches = []

y\_img\_batches = []

for i in range(0, len(test\_generator)):

y\_img\_batch, y\_true\_batch = test\_generator[i]

y\_true\_batch = y\_true\_batch.argmax(axis=-1)

y\_img\_batches.append(y\_img\_batch)

y\_true\_batches.append(y\_true\_batch)

y\_pred\_batch = model.predict(y\_img\_batch)

y\_pred\_batch = y\_pred\_batch.argmax(axis=-1)

y\_pred\_batches.append(y\_pred\_batch)

y\_pred\_batches

def predict\_images(images\_path, label):

amount = 0

fig = plt.figure(figsize=(18, 8))

for file in os.listdir(images\_path):

if file.endswith('.jpeg'):

if amount == 8:

break

img = mpimg.imread(os.path.join(images\_path, file))

plt.subplot(241+amount)

plt.title('{} ---- {}'.format(label, 'NORMAL' if y\_pred\_batches[amount] == 0 else 'PNEUMONIA'))

imgplot = plt.imshow(img)

amount += 1

predict\_images('./chest\_xray/val/NORMAL', 'NORMAL')

predict\_images('./chest\_xray/val/PNEUMONIA', 'PNEUMONIA')

"""## Model saving and loading"""

model.save('awesome\_model.h5')

new\_model = keras.models.load\_model('awesome\_model.h5')

***II.* Conversion of .h5 to tflite file**

import tensorflow as tf

model = tf.keras.models.load\_model('E:\Dataset\awesome\_model.h5')

converter = tf.lite.TFLiteConverter.from\_keras\_model(model)

tflite\_model = converter.convert()

open("model\_unquant.tflite", "wb").write(tflite\_model)

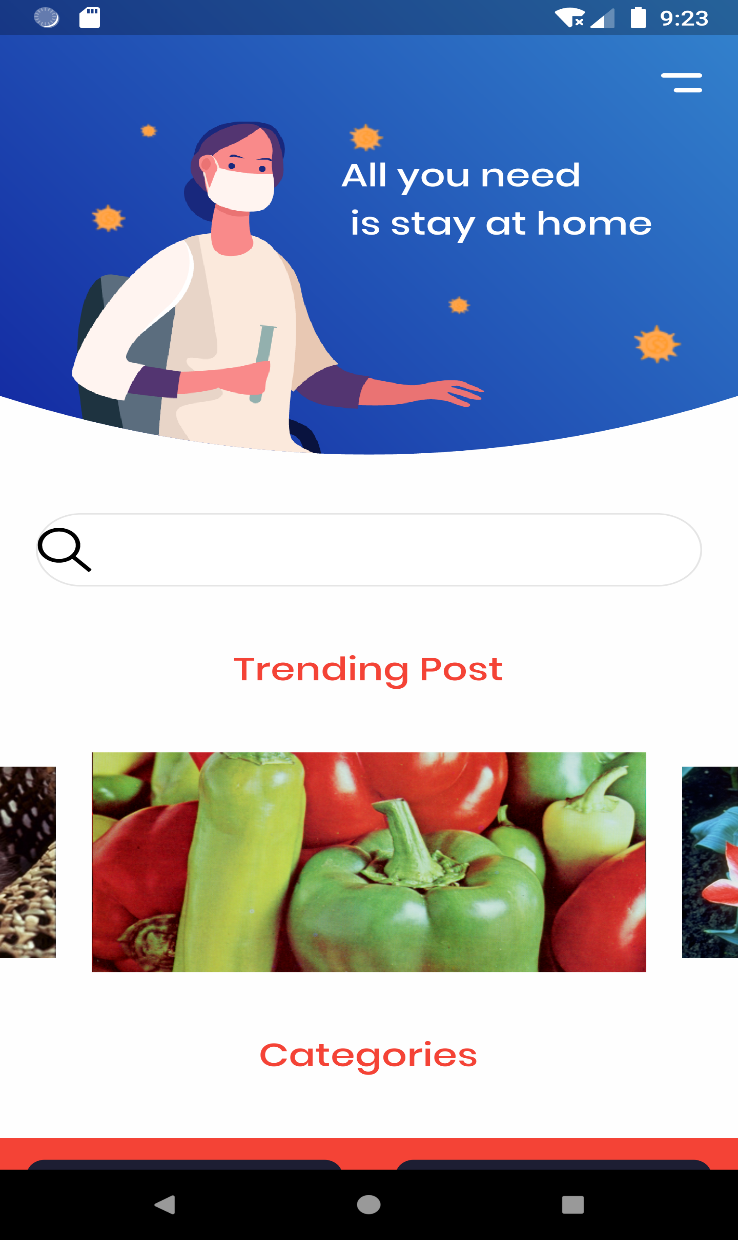
**4) Creating UI for dashboard**

Our dashboard contains an image, navigation bar, Carousel slide show with some images, and grid view for selection of disease. For instance, we have created Carousel slideshow with images but in future it will be replaced by articles related to health.

**Code for Dashboard**:

**import 'package:firebase\_auth/firebase\_auth.dart'**;  
**import 'package:firebase\_database/firebase\_database.dart'**;  
**import 'package:flutter/material.dart'**;  
**import 'package:cloud\_firestore/cloud\_firestore.dart'**;  
**import 'main.dart'**;  
**import 'constant.dart'**;  
**import 'package:carousel\_slider/carousel\_slider.dart'**;  
**import 'package:flutter\_svg/flutter\_svg.dart'**;  
**import 'Pneumonia.dart'**;  
**class** Home **extends** StatelessWidget {  
 Home({**this**.**uid**});  
 **final** String **uid**;  
 **final** String **title** = **"Home"**;  
  
 List<String> **imageLinks** = [  
 **'https://homepages.cae.wisc.edu/~ece533/images/fruits.png'**,  
 **'https://homepages.cae.wisc.edu/~ece533/images/cat.png'**,  
 **'https://homepages.cae.wisc.edu/~ece533/images/peppers.png'**,  
 **'https://homepages.cae.wisc.edu/~ece533/images/tulips.png'** ];  
  
 **void** \_navigateToNextScreen(BuildContext context) {  
 Navigator.*of*(context).push(MaterialPageRoute(builder: (context) => NewScreen()));  
 }  
  
  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 appBar: AppBar(  
 title: Text(**title**),  
 actions: <Widget>[  
 IconButton(  
 icon: Icon(  
 Icons.*exit\_to\_app*,  
 color: Colors.*white*,  
 ),  
 onPressed: () {  
 FirebaseAuth auth = FirebaseAuth.*instance*;  
 auth.signOut().then((res) {  
 Navigator.*pushReplacement*(  
 context,  
 MaterialPageRoute(builder: (context) => MyApp()),  
 );  
 });  
 },  
 )  
 ],  
 ),  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 ClipPath(  
 clipper: MyClipper(),  
 child: Container(  
 padding: EdgeInsets.only(left: 40, top: 50, right: 20),  
 height: 350,  
 width: double.*infinity*,  
 decoration: BoxDecoration(  
 gradient: LinearGradient(  
 begin: Alignment.*topRight*,  
 end: Alignment.*bottomLeft*,  
 colors: [  
 Color(0xFF3383CD),  
 Color(0xFF11249F)  
 ]  
 ),  
 image: DecorationImage(  
 image: AssetImage(**"assets/images/virus.png"**),  
 ),  
 ),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.**start**,  
 children: <Widget>[  
  
 Align(  
 alignment: Alignment.*topRight*,  
 child: SvgPicture.asset(**"assets/icons/menu.svg"**)  
 ),  
 SizedBox(height: 20),  
 Expanded(  
 child: Stack(  
 children: <Widget>[  
 SvgPicture.asset(**"assets/icons/Drcorona.svg"**,  
 width: 230,  
 fit: BoxFit.**fitWidth**,  
 alignment: Alignment.*topCenter*,  
 ),  
 Positioned(  
 top: 20,  
 left: 150,  
 child: Text(  
 **"All you need \n is stay at home"**,  
 style: kHeadingTextStyle.copyWith(  
 color: Colors.*white*,  
 ),  
 ),  
 ),  
 Container(),  
 ],  
 ),  
 )  
 ],  
 ),  
 ),  
 ),  
 Container(  
 margin: EdgeInsets.symmetric(horizontal:20),  
 height: 50,  
 width: double.*infinity*,  
 decoration: BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.circular(25),  
 border: Border.all(  
 color: Color(0xFFE5E5E5)  
 ),  
 ),  
 child: Row(  
 children: <Widget>[  
  
 SvgPicture.asset(**"assets/icons/search.svg"**,width:50,height: 30,)  
 ],  
 )  
 ),  
 SizedBox(height: 40),  
 Container(  
 child: Text(**"Trending Post"**, style: kHeadingTextStyle.copyWith(color: Colors.*red*,),),  
 ),  
 SizedBox(height:40),  
 Container(  
 child: Column(  
 children: <Widget>[  
 CarouselSlider(  
 height: 150.0,  
 items: **imageLinks**.map((imageLink) {  
 **return** Builder(  
 builder: (BuildContext context) {  
 **return** Container(  
 width: MediaQuery.*of*(context).**size**.**width**,  
 margin: EdgeInsets.symmetric(horizontal: 10.0),  
 child: Image.network(  
 imageLink,  
 fit: BoxFit.**cover**,  
 ));  
 },  
 );  
 }).toList(),  
  
 reverse:  
 **false**, *//is false by default (reverses the order of items)* enableInfiniteScroll:  
 **true**, *//is true by default (it scrolls back to item 1 after the last item)* autoPlay: **true**, *//is false by default* initialPage:  
 0, *//allows you to set the first item to be displayed* scrollDirection: Axis.**horizontal**, *//can be set to Axis.vertical* pauseAutoPlayOnTouch: Duration(  
 seconds: 5), *//it pauses the sliding if carousel is touched,* onPageChanged: (int pageNumber) {  
 *//this triggers everytime a slide changes* },  
 viewportFraction: 0.8,  
 enlargeCenterPage: **true**, *//is false by default* aspectRatio:  
 16 / 9, *//if height is not specified, then this value is used* )  
 ],  
 ),  
 ),  
 SizedBox(height: 40,),  
 Container(  
 child: Text(**"Categories"**,style: kHeadingTextStyle.copyWith(color: Colors.*red*,),)  
 ),  
 SizedBox(height: 40),  
 Container(  
 height:650,  
 color: Colors.*red*,  
 child: Column(  
 children: <Widget>[  
 Expanded(child:Row(  
 children: <Widget>[  
 Expanded(child:**new** GestureDetector(  
 onTap: (){\_navigateToNextScreen(context);},  
 child: Container(  
 margin: EdgeInsets.all(15.0),  
 decoration: BoxDecoration(  
 color:Color(0xFF1D1E33),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 child:Column(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 children: <Widget>[  
 Text(**'Pneumonia'**, style: kHeadingTextStyle.copyWith(color: Colors.*white*, fontSize: 25),),  
 ],  
 )  
 ),  
 ),  
 ),  
 Expanded(child:**new** GestureDetector(  
 child: Container(  
 margin: EdgeInsets.all(15.0),  
 decoration: BoxDecoration(  
 color:Color(0xFF1D1E33),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 child:Column(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 children: <Widget>[  
 Text(**'Fracture'**, style: kHeadingTextStyle.copyWith(color: Colors.*white*, fontSize: 25),),  
 ],  
 )  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 Expanded(child:Row(  
 children: <Widget>[  
 Expanded(child:**new** GestureDetector(  
 child: Container(  
 margin: EdgeInsets.all(15.0),  
 decoration: BoxDecoration(  
 color:Color(0xFF1D1E33),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 child:Column(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 children: <Widget>[  
 Text(**'COVID 19'**, style: kHeadingTextStyle.copyWith(color: Colors.*white*, fontSize: 25),),  
 ],  
 )  
 ),  
 ),  
 ),  
 Expanded(child:**new** GestureDetector(child:Container(  
 margin: EdgeInsets.all(15.0),  
 decoration: BoxDecoration(  
 color:Color(0xFF1D1E33),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 child:Column(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 children: <Widget>[  
 Text(**'Pneumonia'**, style: kHeadingTextStyle.copyWith(color: Colors.*white*, fontSize: 25),),  
 ],  
 )  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 Expanded(child:Row(  
 children: <Widget>[  
 Expanded(child:**new** GestureDetector(  
 child: Container(  
 margin: EdgeInsets.all(15.0),  
 decoration: BoxDecoration(  
 color:Color(0xFF1D1E33),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
  
 ),  
 ),  
 ),  
 Expanded(child:Container(  
 margin: EdgeInsets.all(15.0),  
 decoration: BoxDecoration(  
 color:Color(0xFF1D1E33),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
  
 ),  
 ),  
 ],  
 ),  
 ),  
 ],  
 ),  
 ),  
 ],  
 ),  
 ),  
 drawer: NavigateDrawer(uid: **this**.**uid**));  
 }  
}  
  
**class** MyClipper **extends** CustomClipper<Path>{  
 @override  
 Path getClip(Size size){  
 **var** path = Path();  
 path.lineTo(0, size.**height** - 80);  
 path.quadraticBezierTo(size.**width**/2, size.**height**, size.**width**, size.**height** - 80);  
 path.lineTo(size.**width**, 0);  
 path.close();  
 **return** path;  
 }  
 @override  
 bool shouldReclip(CustomClipper<Path> oldClipper){  
 **return false**;  
 }  
}  
  
**class** NavigateDrawer **extends** StatefulWidget {  
 **final** String **uid**;  
 NavigateDrawer({Key key, **this**.**uid**}) : **super**(key: key);  
 @override  
 \_NavigateDrawerState createState() => \_NavigateDrawerState();  
}  
  
**class** \_NavigateDrawerState **extends** State<NavigateDrawer> {  
  
 **void** initState() {  
 **super**.initState();  
 getEmail();  
 }  
   
 Future<**void**> getEmail() **async** {  
 **var** result = (**await** FirebaseDatabase.*instance*.reference().child(**"Users"**).child(**widget**.**uid**).once()).**value**;  
 *//print(result['email']* print(result);  
  
 }  
  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Drawer(  
 child: ListView(  
 padding: EdgeInsets.*zero*,  
 children: <Widget>[  
 UserAccountsDrawerHeader(  
 accountEmail: FutureBuilder(  
 future: FirebaseDatabase.*instance* .reference()  
 .child(**"Users"**)  
 .child(**widget**.**uid**)  
 .once(),  
 builder: (context, AsyncSnapshot<DataSnapshot> snapshot) {  
 **if** (snapshot.**hasData**) {  
 **return** Text(snapshot.**data**.**value**[**'email'**]);  
 } **else** {  
 **return** CircularProgressIndicator();  
 }  
 }),  
 accountName: FutureBuilder(  
 future: FirebaseDatabase.*instance* .reference()  
 .child(**"Users"**)  
 .child(**widget**.**uid**)  
 .once(),  
 builder: (context, AsyncSnapshot<DataSnapshot> snapshot) {  
 **if** (snapshot.**hasData**) {  
 **return** Text(snapshot.**data**.**value**[**'name'**]);  
 } **else** {  
 **return** CircularProgressIndicator();  
 }  
 }),  
  
 decoration: BoxDecoration(  
 color: Colors.*blue*,  
 ),  
 ),  
 ListTile(  
 leading: **new** IconButton(  
 icon: **new** Icon(Icons.*home*, color: Colors.*black*),  
 onPressed: () => **null**,  
 ),  
 title: Text(**'Home'**),  
 onTap: () {  
  
 print(**widget**.**uid**);  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (context) => Home(uid: **widget**.**uid**)),  
 );  
 },  
 ),  
 ListTile(  
 leading: **new** IconButton(  
 icon: **new** Icon(Icons.*settings*, color: Colors.*black*),  
 onPressed: () => **null**,  
 ),  
 title: Text(**'Settings'**),  
 onTap: () {  
 getEmail();  
 print(**widget**.**uid**);  
 },  
 ),  
 ],  
 ),  
 );  
 }  
}

**Output:**



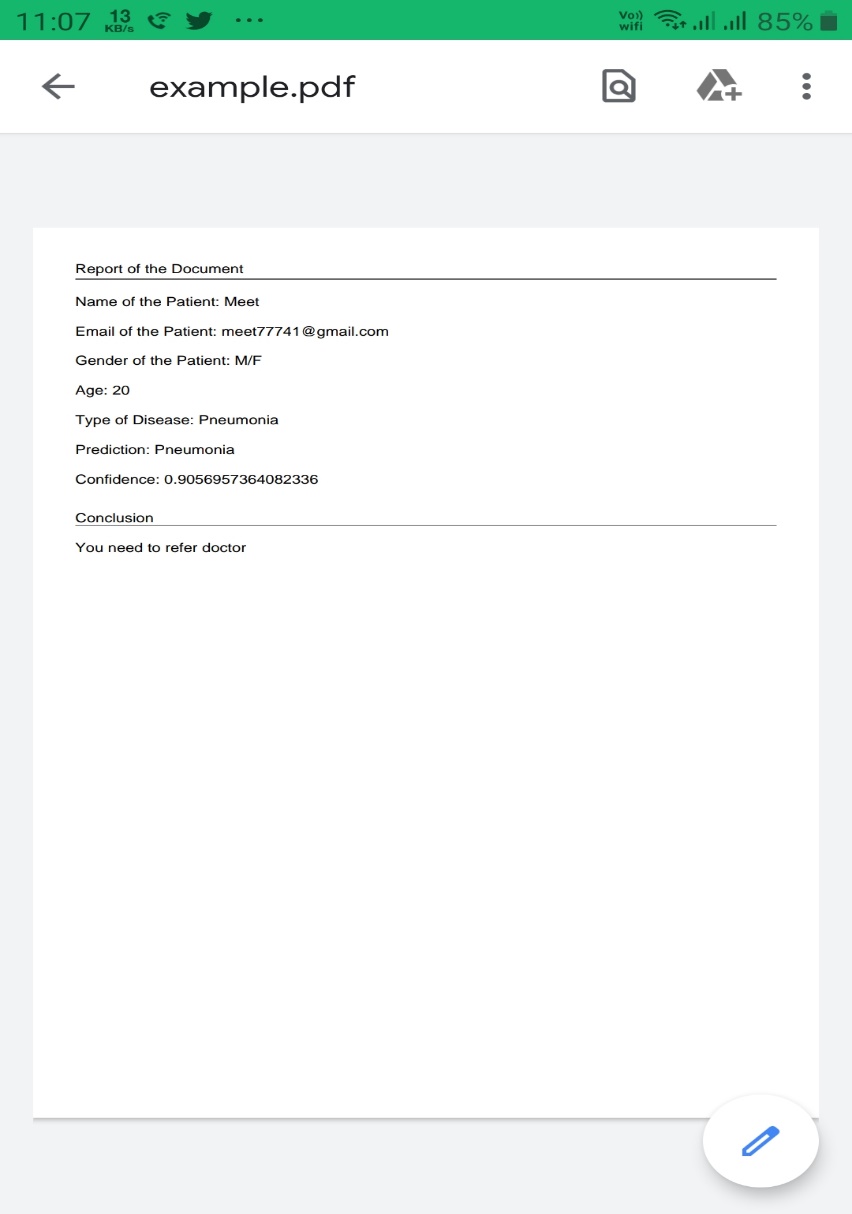
**5) Report Generation**

Once the disease is predicted, the results will be shown to user. A user has also facility to save report to External storage of smartphone. User only needs to click a button which saves report in pdf format. Report contains details like name of user, email, age, prediction of disease i.e. class, confidence.

**Code for saving pdf**:

**final pdf** = pw.Document();  
  
writeOnPdf(){  
 **pdf**.addPage(  
 pw.MultiPage(  
 pageFormat: PdfPageFormat.*a4*,  
 margin: pw.EdgeInsets.all(32),  
  
 build: (pw.Context context){  
 **return** <pw.Widget>[  
 pw.Header(  
 level: 0,  
 child: pw.Text(**"Report of the Document"**)  
 ),  
  
 pw.Paragraph(  
 text: **"Name of the Patient: XYZ"** ),  
 pw.Paragraph(  
 text: **"Gender of the Patient: M/F"** ),  
  
 pw.Paragraph(  
 text:**"Type of Disease: Pneumonia"** ),  
  
 pw.Paragraph(  
 text:**"Prediction:** ${**\_recognition**[0][**"label"**]}**"** ),  
 pw.Paragraph(  
 text:**"Confidence:** ${**\_recognition**[0][**"confidence"**]}**"** ),  
  
  
  
 pw.Header(  
 level: 1,  
 child: pw.Text(**"Conclusion"**)  
 ),  
  
 pw.Paragraph(  
 text: **"You need to refer doctor"** )  
  
 ];  
  
 }  
 )  
 );  
}  
  
Future savePdf() **async**{  
 Directory documentDirectory =**await** getExternalStorageDirectory();  
  
 String documentPath = documentDirectory.**path**;  
  
 File file = File(**"**$documentPath**/example.pdf"**);  
  
 file.writeAsBytesSync(**pdf**.save());  
 print(**'Documentpath :** $documentPath**'**);  
}

**Output:**



**Ch - 4 : Summary and Future Work**

**4.1 Summary:**

In a Nutshell, we have designed and developed mobile application which has a capability of forecasting the wide range of disease from user given input, which is supposed to be in form of image. To get started user needs to follow through registration process which requires some basic users’ profile that may help in future. Our Application manifests available diseases which can be accurately and easily forecasted with zero or minimum user intervention. For an Instance, user has an image of Chest X-ray and wishes to check whether he/she is affected by pneumonia in such case our app comes handy. By following the steps guided through our tutor user can get his/her full fledge in depth report and analysis of disease. Along with this, if the severity of disease is above the par then our application also displays the nearby doctor in specific specialization with location on google map.

**4.2 Future Work:**

* News Article Integration
* Pointing Nearby Doctors’ location on google map
* Deploying our app on AWS Server
* AI chatbot
* Online Doctor Consultation
* Enhancing UI

**References**

**1.** The Complete Flutter Development Bootcamp with Dart - <https://www.udemy.com/course/flutter-bootcamp-with-dart/>

**2.** Android - <https://developer.android.com/docs>

**3.** Flutter - <https://flutter.dev/docs>

**4.** Firebase - <https://firebase.flutter.dev/docs/overview>

**5.** Efficient Pneumonia Detection in Chest X-ray Images Using Deep Transfer Learning - <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7345724/>

**6.** Identifying pneumonia in chest X-rays: A deep learning approach - <https://www.researchgate.net/publication/333623227_Identifying_Pneumonia_in_Chest_X-Rays_A_Deep_Learning_Approach>

**7.** Flutter & Firebase App Build - <https://www.youtube.com/playlist?list=PL4cUxeGkcC9j--TKIdkb3ISfRbJeJYQwC>

**8.** CLASSIFICATION OF IMAGES BASED ON CONVOLUTION NEURAL NETWORKS - <https://www.freepatentsonline.com/y2020/0034669.html>

**9.** Image identification method based on depth learning and transfer learning - <https://patentscope.wipo.int/search/en/detail.jsf?docId=CN202688881&tab=NATIONALBIBLIO>

**10.** DIABETIC RETINOPATHY RECOGNITION SYSTEM BASED ON FUNDUS IMAGES - <https://patentscope.wipo.int/search/en/detail.jsf?docId=CN207746181&tab=NATIONALBIBLIO>

**11.** DEEP LEARNING ALGORITHM-BASED CLASSIFICATION METHOD OF BACTERIAL PNEUMONIA AND VIRAL PNEUMONIA IN CHILDREN - <https://patentscope.wipo.int/search/en/detail.jsf?docId=CN222364420&tab=NATIONALBIBLIO&_cid=P11-KFFJIM-39429-1>

**12.** SYSTEMS AND METHODS FOR IMAGE PROCESSING - <https://patentscope.wipo.int/search/en/detail.jsf?docId=US254721434&tab=NATIONALBIBLIO&_cid=P11-KFFNJ2-99073-1>

**13.** <https://github.com/>

**14.** <https://stackoverflow.com/>

**15.** Chatbot - <https://cloud.google.com/dialogflow/docs>