HX8001 PROFESSIONAL READINESS FOR INNOVATION,EMPLOYABILITY AND ENTREPRENEURSHIP

PROJECT REPORT

PROJECT TITLE :Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy

TEAM ID :PNT2022TMID24494

TEAM MEMBERS	:SENTHIL KUMAR T	13319106069
	BHARATH RAJ M	13319106009
	RAMESWARAN N	13319106064
	KARTHICK V	13319106036

BACHELOR OF ENGINEERING
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

VELAMMAL INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY : CHENNAI 600025

Table of Contents

INTRODUCTION

PROJECT OVERVIEW

Diabetic retinopathy is the most common microvascular complication in diabetes for the screening of which the retinal imaging is the most widely used method due to its high sensitivity in detecting retinopathy. The evaluation of the severity and degree of retinopathy associated with a person having diabetes, is currently performed by medical experts based on the fundus or retinal images of the patient's eyes. As the number of patients with diabetes is rapidly increasing, the number of retinal images produced by the screening programmes will also increase, which in turn introduces a large labor-intensive burden on the medical experts as well as cost to the healthcare services. This could be alleviated with an automated system either as support for medical experts work or as full diagnosis tool. There are two recent studies that have investigated the use of deep learning systems in automated detection of diabetic retinopathy. Both show that an automated system, based on the deep learning artificial neural network approach, can achieve high sensitivity with high specificity in detecting the referable diabetic retinopathy, defined as moderate or worse diabetic retinopathy. There are also other referable eye complications that have recently been investigated with this approach, such as diabetic macular edema and possible glaucoma and age-related macular degeneration.

For an automated system to be clinically viable, it should be able to classify retinal images based on clinically used severity scales, such as the proposed international clinical diabetic retinopathy which are also used in Finland. Another substantial barrier to broader and more effective use of deep learning systems is thought to be the large quantity of annotated images needed for the model to learn.

In this project, our aim is to identify retinopathy using five different diabetic retinopathy and macular edema classification systems. In addition to the earlier studies of the referable diabetic retinopathy classification system we also present state-of-the-art results for the clinically used five grade classification and for the first time four grade macular edema classification. Moreover, we present what preprocessing and regularization steps to the images need to be done for the good functionality of the deep learning system and investigate systematically how the size with a much smaller number of images used in training affects its performance.

PURPOSE

Diabetic retinopathy is a complication of diabetes, caused by high blood sugar levels damaging the back of the eye (retina). It can cause blindness if left undiagnosed and untreated. However, it usually takes several years for diabetic retinopathy to reach a stage where it could threaten your sight. Diabetic eye screening is important as it helps to prevent sight loss. As someone with diabetes, your eyes are at risk of damage from diabetic retinopathy. Screening can detect the condition early before you notice any changes to your vision

LITERATURE SURVEY

EXISTING PROBLEM

Diabetes is a globally prevalent disease that can cause visible microvascular complications such as diabetic retinopathy in the human eye retina, the images of which are today used for manual disease screening and diagnosis. This labor-intensive task could greatly benefit from automatic detection using deep learning techniques.

REFERENCE

M. Z. Atwany, A. H. Sahyoun and M. Yaqub.Deep Learning Techniques for Diabetic Retinopathy Classification: This paper reviews and analyzes state-of-the-art deep learning methods in supervised, self-supervised, and Vision Transformer setups, proposing retinal fundus image classification and detection.the paper discusses the available retinal fundus datasets for Diabetic Retinopathy that are used for tasks such as detection, classification, and segmentation. The paper also assesses research gaps in the area of DR detection/classification and addresses various challenges that need further study and investigation

S. Gupta, A. Panwar, A. Kapruwan, N. Chaube and M. ChauhanReal Time Analysis of Diabetic Retinopathy Lesions by Employing Deep Learning and Machine Learning Algorithms using Color Fundus DataThe color fundus dataset scans after processing are passed to multiple Deep Learning (DL) models employed to learn characteristics. These models trained on millions of different images from thousands of classes. Finally, several machine learning classifiers were used to classify lesions using the collected characteristics. The

extracted result shows a very eye-catching performance. This enables experts to create architecture that fully addresses the problem of classifying unidentified scans into the right class or category.

T. A. Soomro *et al*.Deep Learning Models for Retinal Blood Vessels Segmentation.This paper presents a comprehensive review of the principle and application of deep learning in retinal image analysis.This paper characterizes each deep learning based segmentation method as described. Analyzing along with the limitations and advantages of each method. In the end, we offer some recommendations for future improvement for retinal image analysis.

B. Bulut, V. Kalın, B. B. Güneş and R. Khazhin.Deep Learning Approach For Detection Of Retinal Abnormalities Based On Color Fundus Images.This research uses the Xception model with transfer learning method to classify images obtained from Akdeniz University Hospital Eye Diseases Department. During the analysis, the Xception model containing 50 different parameter combinations was trained by scanning the appropriate hyper -parameter space for the model. Comparisons were made for the top 9 models with the highest performance.in order to test the performance of the model with an independent data set, open access fundus images were used for test analysis and binary classification AUC (Area Under Curve) values were calculated for 21 different diseases

N.Memari, S. Abdollahi, M. M. Ganzagh and M. Moghbel.Computer-assisted diagnosis (CAD) system for Diabetic Retinopathy screening using color fundus images using Deep learning.The proposed computer-assisted diagnosis system starts with the segmentation of the blood vessels. Then, microaneurysms and exudates are segmentation from the image. Statistical and regional features are then extracted utilizing first, second, and higher-order image features. A Deep Learning framework will be utilized for extracting additional statistical image descriptors as Deep Learning has superior contextual analysis capabilities compared to other machine learning techniques.

B. Goutam, M. F. Hashmi, Z. W. Geem and N. D. Bokde.A Comprehensive Review of Deep Learning Strategies in Retinal Disease Diagnosis Using Fundus Images. This article presents a comprehensive study of different deep learning strategies employed in recent times for the diagnosis of five major eye diseases, i.e., Diabetic retinopathy, Glaucoma, age-related macular degeneration, Cataract, and Retinopathy of prematurity.

H. Yeh, C. -J. Lin, C. -C. Hsu and C. -Y. Lee.Deep-learning based automated segmentation of Diabetic Retinopathy symptoms.Deep learning is used in many types of preprocessing for segmentation. We preprocessed fundus images and

inputted them into the model for training. Finally, LDF image was used to obtain the best preprocessing method for optic disc segmentation in fundus images.

H. Kaushik, D. Singh, M. Kaur, H. Alshazly, A. Zaguia and H.Hamam.Diabetic Retinopathy Diagnosis From Fundus Images Using Stacked Generalization of Deep ModelsIn this research, a methodology to eliminate these unnecessary reflectance properties of the images using a novel image processing schema and a stacked deep learning technique for the diagnosis. For the luminosity normalization of the image, the gray world color constancy algorithm is implemented which does image desaturation and improves the overall image quality.

A.Bali and V. Mansotra.Deep Learning-based Techniques for the Automatic Classification of Fundus Images: A Comparative Study In this paper different deep learning (DL) techniques for automatic classification of fundus images have been discussed and results are compared on the basis of accuracy, f1-score and AUC.

PROBLEM STATE DEFINITION

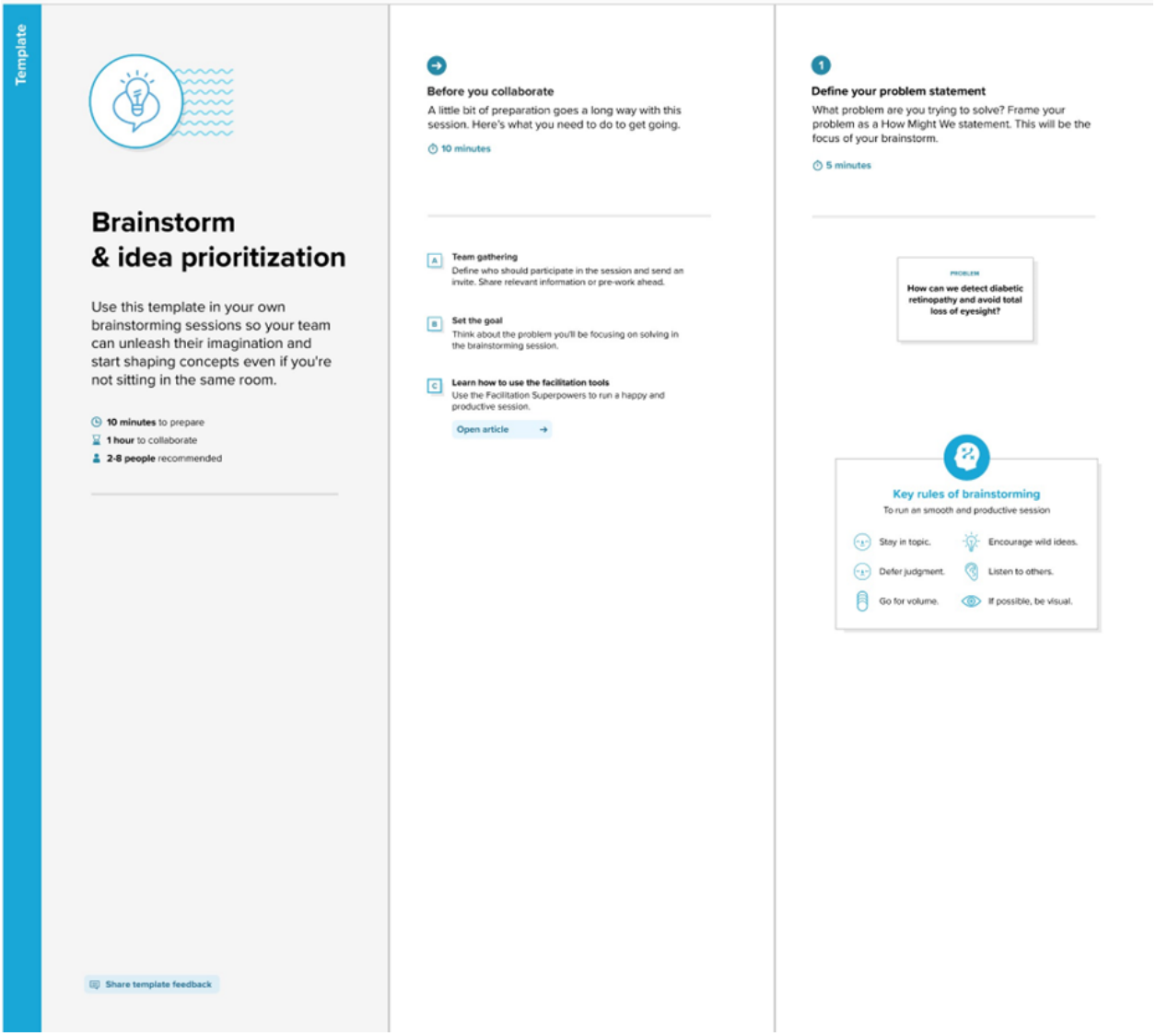
I am	Describe customer with 3-4 key characteristics - *who are they?*	Describe the customer and their attributes here
I'm trying to	*List their outcome or "job" the care about - what are they trying to achieve?*	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way – *what bothers them most?*	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists – *what needs to be solved?*	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view – *how does it impact them emotionally?*	Describe the emotions the result from experiencing the problems or barriers

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A patient who recently got diagnosed with diabetes	find if i lose my eyesight for forever	I cannot find anyone who can identify the disease	A lot of doctors are busy and sometimes can be expensive	afraid and insecure to discuss it with other
PS-2	a patient	find if i am affected by diabetic retinopathy	I can not find the results	the current method is mostly manual and are prone to many errors	scared of losing eyesight

IDEATION AND PROPOSED SOLUTION

EMPATHY MAP CANVAS

IDEATION AND BRAINSTORMING



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

Senthil Kumar

- Early stage symptoms are not easily identified
- Use of Deep learning
- Do not mentally stress patient
- Diagnosis must be non invasive
- Balanced food diet
- Use of IoT

Rameswaran

- Usage of python for its utility
- A handheld device
- Exercise to maintain body fitness
- Image processing using MATLAB
- Consuming diet foods
- Maintain proper levels of sugar

Bharath Raj

- Medication should be taken at earliest
- Dataset from hospitals
- Cheaper cost of manufacturing
- Use of computerized approach
- Regular eye check ups
- Not to push them away from society

Karthick

- Suggestion to avoid high sugar foods
- Use of automated program
- Include option to diagnose other eye disease
- Consult a doctor
- Do not bombard them with advice
- Use of mobile devices

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

Product features

- A handheld device
- Use of IoT
- Use of mobile devices
- Cheaper cost of manufacturing

To be followed by patients

- Maintain proper levels of sugar
- Balanced food diet

Advice to patients

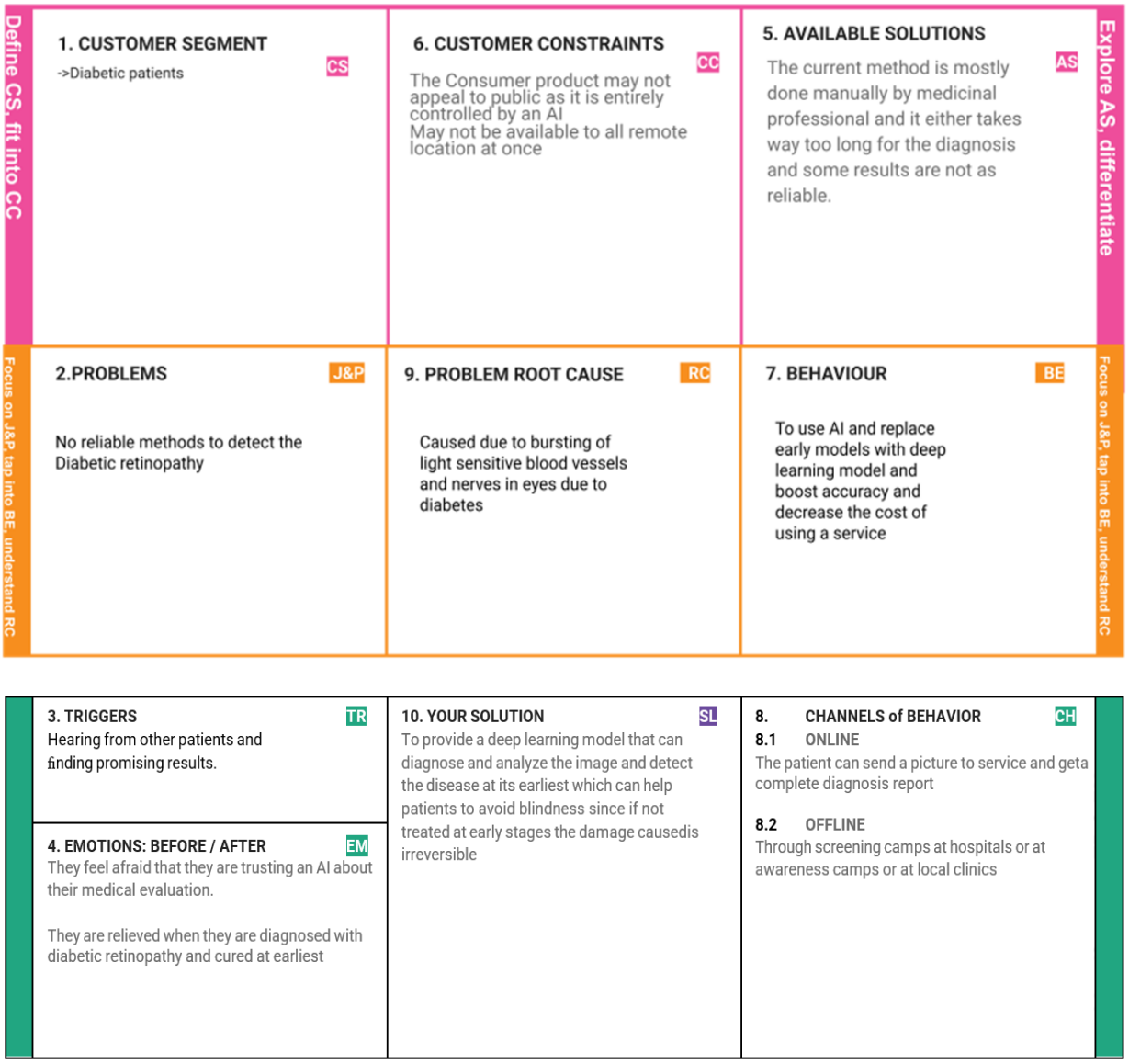
- Suggestion to avoid high sugar foods

PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Diabetes is a globally prevalent disease that can cause visible microvascular complications such as diabetic retinopathy in the human eye retina, the images of which are today used for manual disease screening and diagnosis. This Labor-intensive task could greatly benefit from automatic detection using deep learning techniques.
2.	Idea / Solution description	Here we present a deep learning system that identifies referable diabetic retinopathy comparably or better than presented in the previous studies, although we use only a small fraction of images (less than 1/4th) in training but are aided with higher image resolutions.
3.	Novelty / Uniqueness	One of the most important decisions had to be made is which programming language can be used for satisfying our goal for extracting knowledge from our data. The suitable programming language is Python. Because it has a lot of tools and

		framework to create a strong ANN.
4.	Social Impact / Customer Satisfaction	This may help the Diabetic patient to detect DR in early stages by health camps and in regular interval of checkup with their retinal images.
5.	Business Model (Revenue Model)	Can be collaborated with the Diabetics Diagnosis center for regular check up. Government camps and NGO healthcare camps can be conducted for awareness.
6.	Scalability of the Solution	Can also detect if the patient is affected by Glaucoma or not, since both the tests make use of the same fundus image.

PROBLEM SOLUTION FIT



REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENT AND NONFUNCTIONAL REQUIREMENT

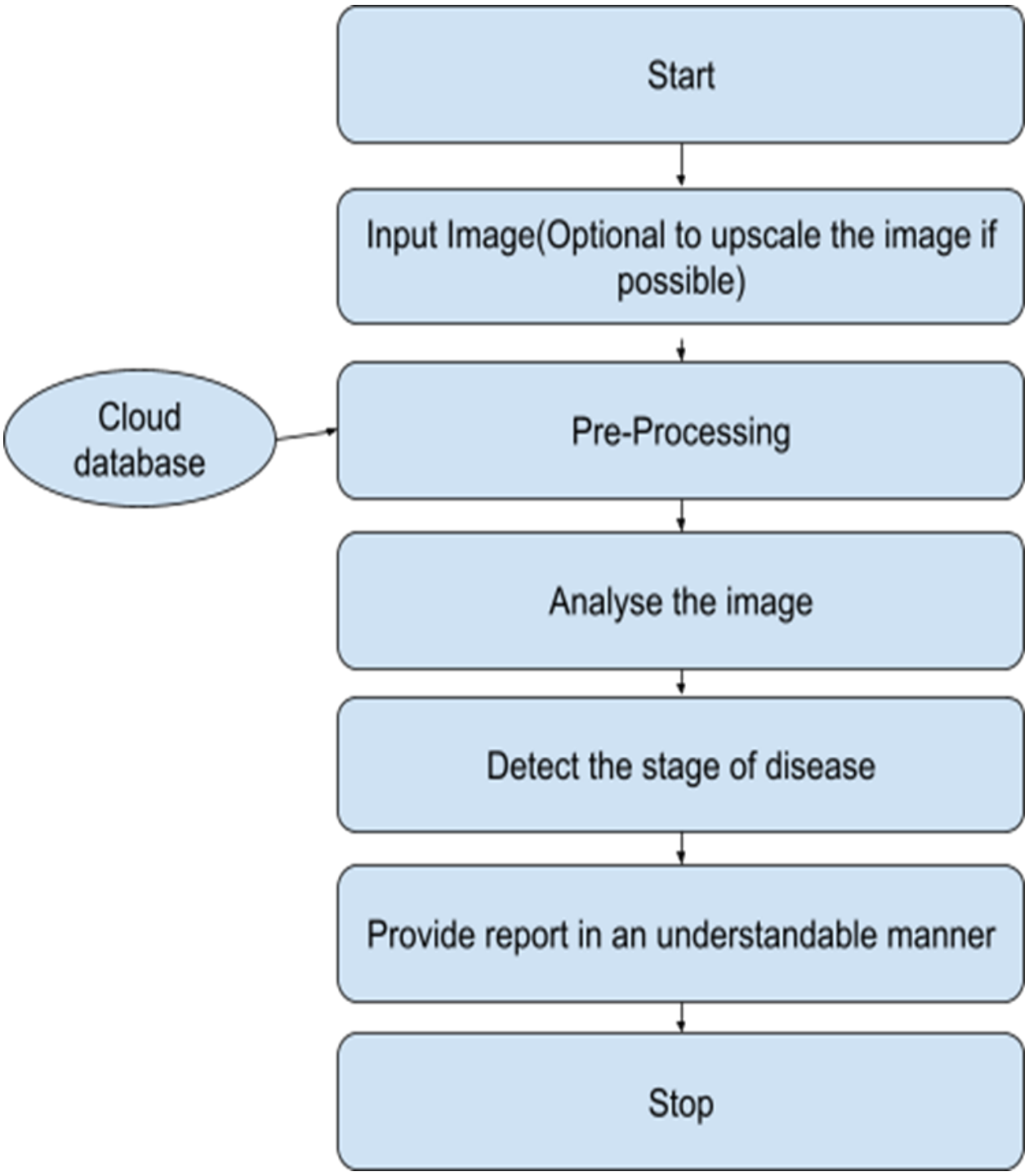
Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	User registration via form using email ID and Password
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	Using the registered email ID and Password as login credentials
FR-4	Profile Dashboard	Viewing the profile, Changing Password and Previous Records
FR-5	Checking For Diabetic Retinopathy	Uploading the FUNDUS Image of the eye to predict the disease
FR-6	User Tracking	Maintaining the Record of how far the eye has been affected
FR-7	Feedback & Support	Collection of feedback whether the results were accurate enough

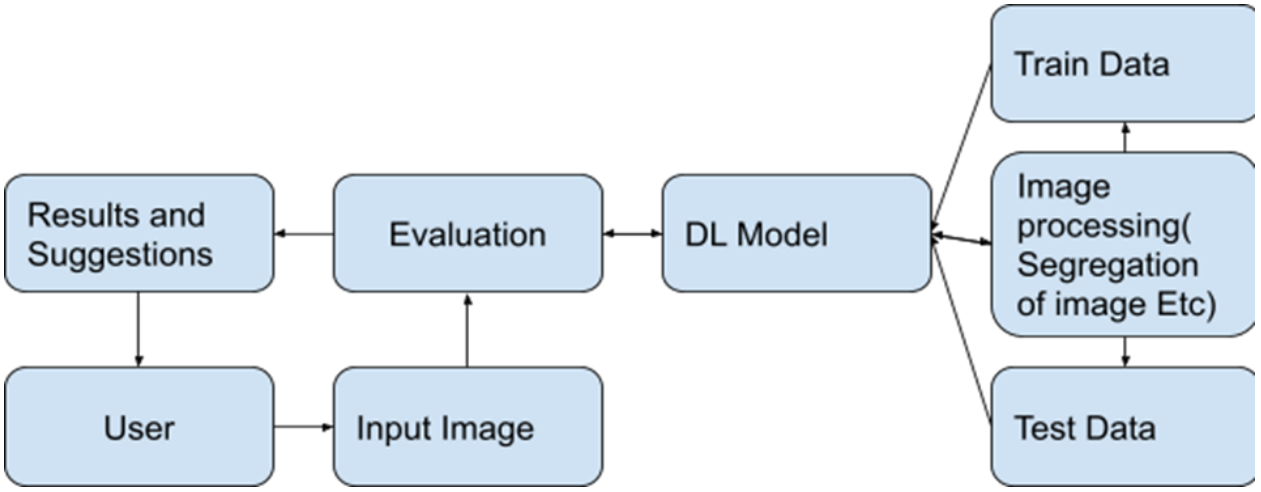
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• The system should have a quality attribute that must be able to assess the ease of the usage of the UI. • The system should not expect any technical prerequisites from the user's side.
NFR-2	**Security**	• User details and login credentials should be safe and secure. • The confirmation of a valid user is required for authentication.
NFR-3	**Reliability**	• Portable and cross-platform independent. • Easy to use and flexible.
NFR-4	**Performance**	• The system should handle the traffic efficiently and service requests while consuming less bandwidth. • The accuracy of the result of a measurement, calculation, or specification should be dependent on the datasets. • The page should not take a lot of time to load the contents and display them.
NFR-5	**Availability**	• The version of the application should be available even at the time of maintenance and updating. • The system should run 24 hours a day, 7 days a week [24/7 available].
NFR-6	**Scalability**	• The application should be in the way of adding new functionalities or modules without affecting the existing functionalities.

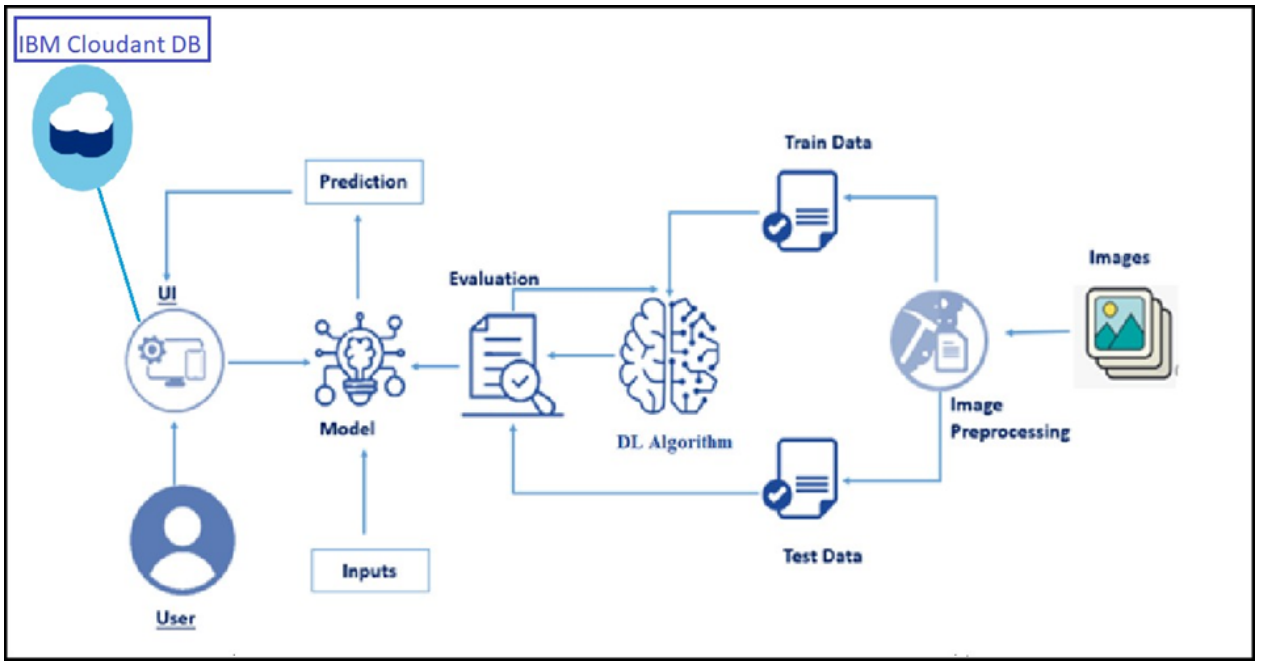
PROJECT DESIGN

DATA FLOW DIAGRAM



SOLUTION ARCHITECTURE





USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Diabetic Patient	Image Upload	USN-1	As a user, I should be able to upload images of eye	Must be able to upload easily with little verification	High	Sprint-1
		USN-2	As a user, I will receive the diagnosis report within an hour	Diagnosis must not take more than an hour	High	Sprint-2
		USN-3	As a user, I can receive information about my diagnosis in an easy to understand manner	Information presented must be simple and yet have all info for the patient	Low	Sprint-3
		USN-4	As a user, I find the severity of the disease (if affected)	Severity of disease must be notified properly	Medium	Sprint-4

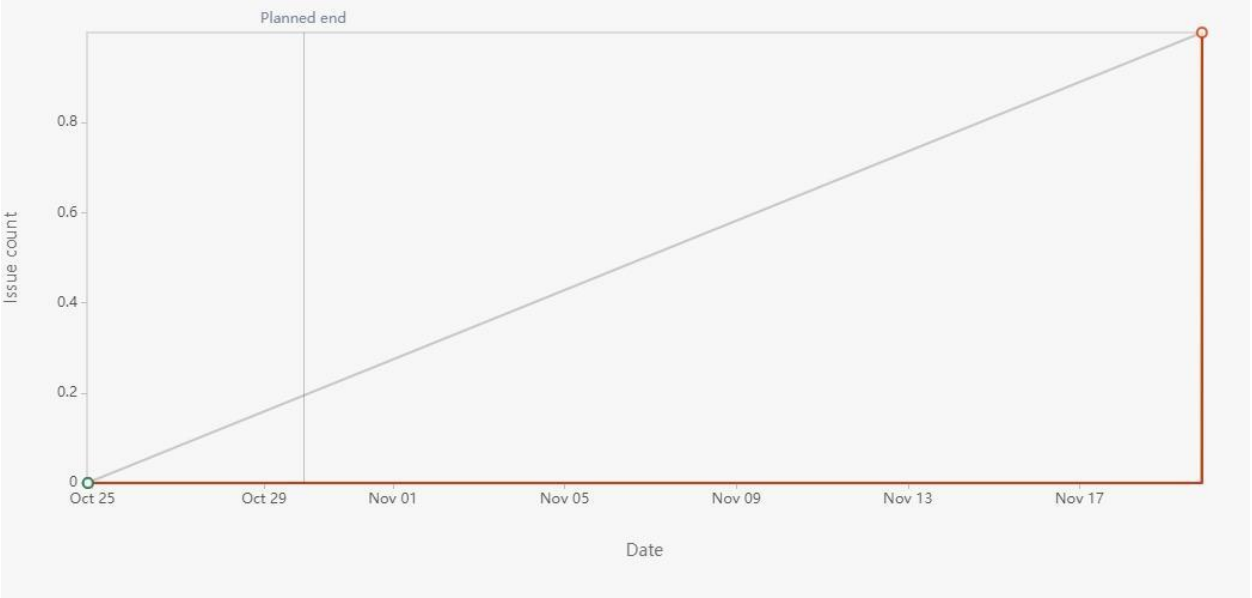
PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation

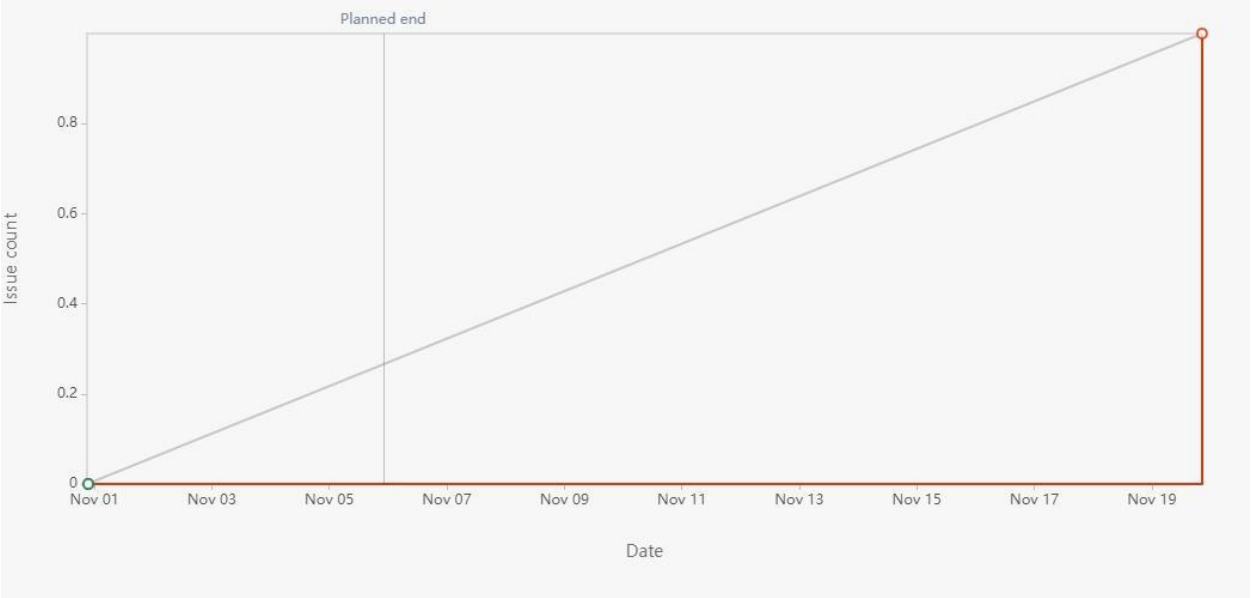
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email or phone number and password, and confirming my password	10	Medium	Bharath Raj Senthil Kumar
Sprint-1	Login	USN-2	As a user, I can log into the application by entering Login credentials	10	Medium	Bharath Raj Senthil Kumar
Sprint-2	Upload Images	USN-3	As a user, I should be able to upload the image of eye Retina.	15	Medium	Rameswaran Karthick
Sprint-2	Dashboard	USN-4	As a user, based on my requirement I can navigate through the dashboard.	15	Medium	Rameswaran Karthick
Sprint-3	Train the model	Task-1	As a developer, the dataset will be uploaded and trained by a developed algorithm.	10	Medium	Bharath Raj Senthil Kumar
Sprint-4	Testing & Evaluation	Task-2	As a developer, we tested the trained model using the provided dataset and model will be evaluated for accurate results.	10	Medium	Bharath Raj Senthil Kumar
Sprint-4	Display predicted result	USN-5	As a user, I can view the predicted result in the dashboard.	10	Medium	Rameswaran Karthick

REPORT FROM JIRA

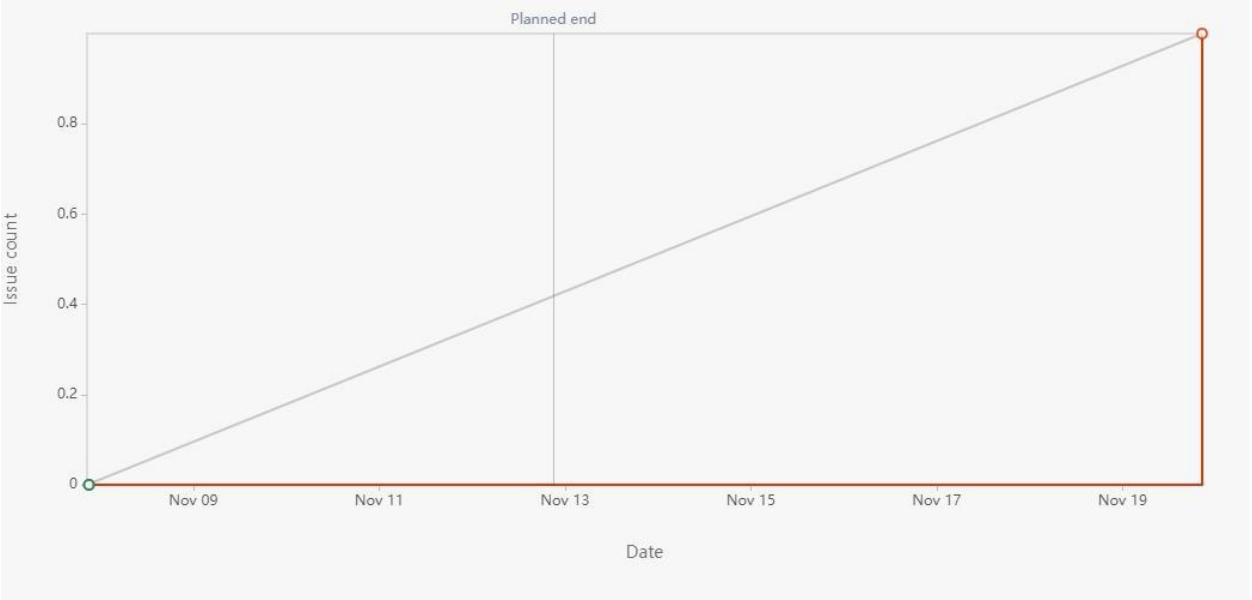
Sprint-1



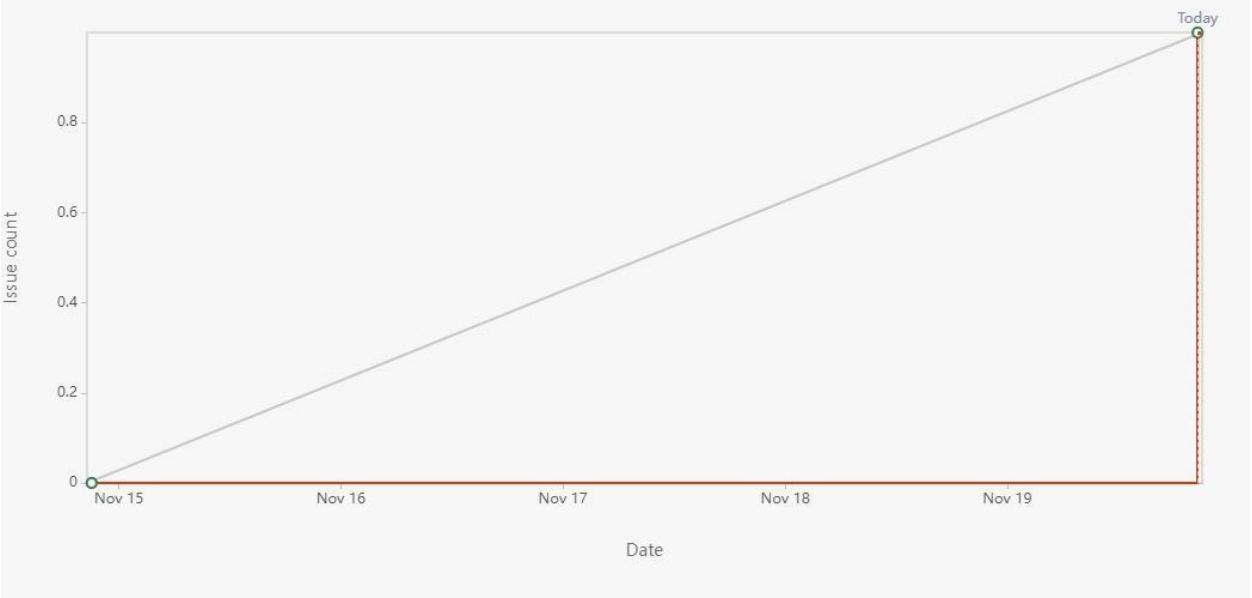
Sprint-2



Sprint-3



Sprint-4



CODING AND SOLUTIONING

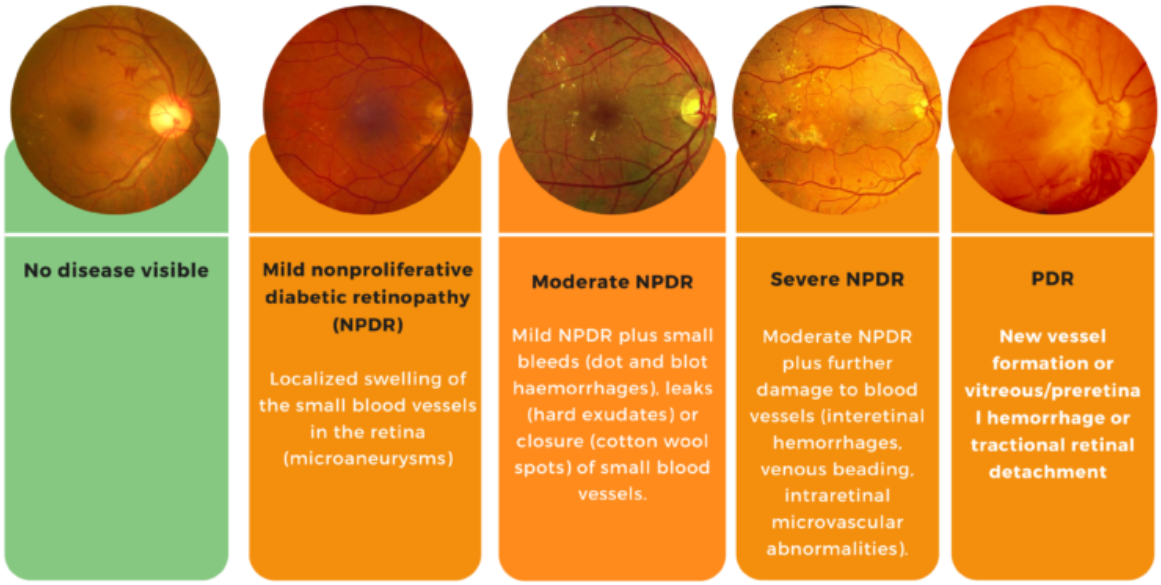
We have developed a website which authenticates users and helps them upload and check the seriousness of the diabetics. We have developed a multilayer deep convolutional neural network that classifies the user image of an eye to which extent the disease diabetics have been affected. The model will classify the images into 5 categories of diabetics and report them on asking for prediction.

TESTING

TEST CASES

```
imageSize = [299,299]
trainPath= r"/content/preprocessed dataset/preprocessed dataset/training"
testPath= r"/content/preprocessed dataset/preprocessed dataset/testing"
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
,load_img
from tensorflow.keras.applications.xception import Xception,preprocess_input
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_rang
e=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
training_set=train_datagen.flow_from_directory('/content/drive/MyDrive/IBM_
DataSet/preprocessed dataset/preprocessed
dataset/training',target_size=(299,299),batch_size=32,class_mode="categorical"
)
test_set=test_datagen.flow_from_directory('/content/drive/MyDrive/IBM_Data
Set/preprocessed dataset/preprocessed dataset/testing',target_size=(299,299
xception = Xception(input_shape=imageSize
+[3],weights='imagenet',include_top=False)
for layer in xception.layers:
  layer.trainable = False
x=Flatten()(xception.output)
```

```
prediction = Dense(5, activation='softmax')(x)
model = Model(inputs=xception.input, outputs=prediction)
model.summary()
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accu
racy'])
r=model.fit(training_set,validation_data=test_set,epochs=30,steps_per_epoch=l
en(training_set)//32,validation_steps=len(test_set)//32)
model.save("updated.xception.diabetic.retinopathy.hs")
```



USER ACCEPTANCE TESTING

```
import os
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import
preprocess_input
from flask import Flask, request,flash, render_template, redirect,url_for
from cloudant.client import Cloudant
from twilio.rest import Client
model = load_model(r"Updated-xception-diabetic-retinopathy.h5")
app = Flask(__name__)
app.secret_key="abc"
app.config['UPLOAD_FOLDER'] = "User_Images"
# Authenticate using an IAM API key
client =
Cloudant.iam('f9f5e849-68df-4795-bf18-687937264b9a-bluemix','TbxD
```

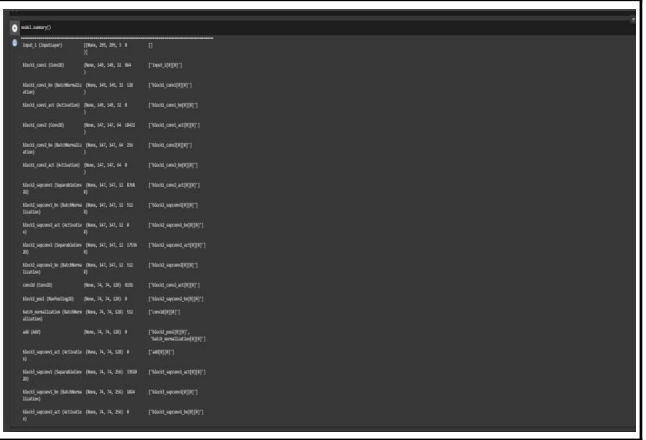
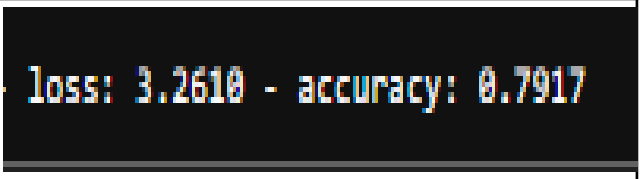
```
hChV8Cdt_vRuRODWLO8c3KYGM6Zx6cCkAKvyoF4g',connect=Tru
e)
# Create a database using an initialized client
my_database = client.create_database('my_database')
if my_database.exists():
    print("Database '{0}' successfully created.".format('my_db'))
# default home page or route


user = ""


@app.route('/')
def index():
    return render_template('index.html', pred="Login", vis ="visible")


@ app.route('/index')
def home():
    return render_template("index.html", pred="Login", vis ="visible")




# registration page
@ app.route('/register',methods=["GET","POST"])
def register():
    if request.method == "POST":
        name =  request.form.get("name")
        mail = request.form.get("emailid")
        mobile = request.form.get("num")
        pswd = request.form.get("pass")
        data = {
            'name': name,
            'mail': mail,
            'mobile': mobile,
            'psw': pswd
        }
        print(data)
        query = {'mail': {'$eq': data['mail']}}
        docs = my_database.get_query_result(query)
        print(docs)
        print(len(docs.all()))
```

```
        if (len(docs.all()) == 0):
            url = my_database.create_document(data)
            return render_template("register.html", pred=" Registration
Successful , please login using your details ")
        else:
            return render_template('register.html', pred=" You are already a
member , please login using your details ")
    else:
        return render_template('register.html')




@ app.route('/login', methods=['GET','POST'])
def login():
    if request.method == "GET":
        user = request.args.get('mail')
        passw = request.args.get('pass')
        print(user, passw)
        query = {'mail': {'$eq': user}}
        docs = my_database.get_query_result(query)
        print(docs)
        print(len(docs.all()))
        if (len(docs.all()) == 0):
            return render_template('login.html', pred="")
        else:
            if ((user == docs[0][0]['mail'] and passw == docs[0][0]['psw'])):
                flash("Logged in as " + str(user))
                return render_template('index.html', pred="Logged in as
"+str(user), vis ="hidden", vis2="visible")
            else:
                return render_template('login.html', pred="The password is
wrong.")
    else:
        return render_template('login.html')




@ app.route('/logout')
def logout():
    return render_template('logout.html')
```

```
@app.route("/predict",methods=["GET", "POST"])
def predict():
    if request.method == "POST":
        f = request.files['file']
        # getting the current path 1.e where app.py is present
        basepath = os.path.dirname(__file__)
        #print ( " current path " , basepath )
        # from anywhere in the system we can give image but we want that
        filepath = os.path.join(str(basepath), 'User_Images', str(f.filename))
        #print ( " upload folder is " , filepath )
        f.save(filepath)
        img = image.load_img(filepath, target_size=(299, 299))
        x = image.img_to_array(img)  # ing to array
        x = np.expand_dims(x, axis=0)  # used for adding one more
dimension
        #print ( x )
        img_data = preprocess_input(x)
        prediction = np.argmax(model.predict(img_data), axis=1)
        index = [' No Diabetic Retinopathy ', ' Mild NPDR ',
                 ' Moderate NPDR ', ' Severe NPDR ', ' Proliferative DR ']
        result = str(index[prediction[0]])
        print(result)
        return render_template('prediction.html', prediction=result, fname =
filepath)
    else:
        return render_template("prediction.html")


if __name__ == "__main__":
    app.debug = False
    app.run()
```

RESULTS

PERFORMANCE METRICS

S.NO	PARAMETER	VALUES	SCREENSHOT
1	Model Summary	Total params:21,885,485 Trainable params:1,024,005 Non-trainable params:20,861,480	
	Accuracy	Training Accuracy– 0.7917 Validation Accuracy– loss3.2610	

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Deep learning is well-suited for image analysis tasks. This is because deep learning algorithms can automatically learn features from images, which is essential for accurate image analysis.
- Deep learning is efficient at handling large amounts of data. This is important for medical image analysis, as medical images are often very large.
- Deep learning is scalable. This means that it can be used to train models on very large datasets, which is important for medical image analysis tasks where data is often limited.
- Deep learning is able to learn from data with little supervision. This is important for medical image analysis, as often there is limited labeled data available.
- Deep learning is robust. This means that it is less likely to over fit to the data, which is important for medical image analysis where data is often limited.

DISADVANTAGES

- There are several disadvantages of deep learning for early detection of diabetic retinopathy.
- One disadvantage is that deep learning requires a large amount of data to train the models.
- This can be a challenge for researchers who do not have access to a large dataset.
- Another challenge is that deep learning models can be very complex, which can make them difficult to interpret.
- Finally, deep learning models can be computationally intensive, which can make them difficult to deploy in resource-limited settings.

CONCLUSIONS

Diabetic retinopathy (DR) is a leading cause of blindness in the United States. Early detection and treatment of DR is critical to preventing vision loss.

However, DR is often asymptomatic in its early stages, making it difficult to detect. Deep learning (DL) is a type of artificial intelligence that can be used to automatically detect patterns in data.

DL has been shown to be effective for detecting DR in images of the retina. In this study, a DL algorithm was used to automatically detect DR in fundus images.

The algorithm was able to accurately detect DR in early stages, before it is symptomatic. This could potentially lead to earlier diagnosis and treatment of DR, which could help to prevent vision loss.

FUTURE SCOPE

There is a great potential for deep learning in fundus image analysis for early detection of diabetic retinopathy. However, there are a few challenges that need to be addressed. First, the current data sets are small and lack diversity. Second, the images are often low quality and need to be pre-processed before they can be used for deep learning. Third, the ground truth labels for the images are often not available. Finally, the current deep learning models are not able to generalize well to real-world data.

APPENDIX

Demo Link: https://youtu.be/mvjNhaaInHQ

GITHUB LINK : IBM-EPBL/IBM-Project-53588-1661419282

SOURCE CODE

Training Data Model Code

```
imageSize = [299,299]
trainPath= r"/content/preprocessed dataset/preprocessed dataset/training"
testPath= r"/content/preprocessed dataset/preprocessed dataset/testing"
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
,load_img
from tensorflow.keras.applications.xception import Xception,preprocess_input
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_rang
e=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
training_set=train_datagen.flow_from_directory('/content/drive/MyDrive/IBM_
DataSet/preprocessed dataset/preprocessed
dataset/training',target_size=(299,299),batch_size=32,class_mode="categorical"
)
test_set=test_datagen.flow_from_directory('/content/drive/MyDrive/IBM_Data
Set/preprocessed dataset/preprocessed dataset/testing',target_size=(299,299
xception = Xception(input_shape=imageSize +
[3],weights='imagenet',include_top=False)
for layer in xception.layers:
  layer.trainable = False
```

```
x=Flatten()(xception.output)
prediction = Dense(5, activation='softmax')(x)
model = Model(inputs=xception.input, outputs=prediction)
model.summary()
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accu
racy'])
r=model.fit(training_set,validation_data=test_set,epochs=30,steps_per_epoch=l
en(training_set)//32,validation_steps=len(test_set)//32)
model.save("updated.xception.diabetic.retinopathy.hs")
```

Application Code

```
import numpy as np
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import
preprocess_input
from flask import Flask, request,flash, render_template,
redirect,url_for
from cloudant.client import Cloudant
from twilio.rest import Client
model = load_model(r"Updated-xception-diabetic-retinopathy.h5")
app = Flask(__name__)
app.secret_key="abc"
app.config['UPLOAD_FOLDER'] = "Testing Images"
# Authenticate using an IAM API key
client =
Cloudant.iam('f9f5e849-68df-4795-bf18-687937264b9a-bluemix','TbxDhCh
V8Cdt_vRuRODWLO8c3KYGM6Zx6cCkAKvyoF4g',connect=True)
# Create a database using an initialized client
my_database = client.create_database('my_database')
if my_database.exists():
    print("Database '{0}' successfully created.".format('my_db'))
# default home page or route

user = ""

@app.route('/')
def index():
          return render_template('index.html', pred="Login", vis
="visible")
```

```
@ app.route('/index')
def home():
          return   render_template("index.html",   pred="Login",   vis
="visible")


# registration page
@ app.route('/register',methods=["GET","POST"])
def register():
    if request.method == "POST":
        name =  request.form.get("name")
        mail = request.form.get("emailid")
        mobile = request.form.get("num")
        pswd = request.form.get("pass")
        data = {
            'name': name,
            'mail': mail,
            'mobile': mobile,
            'psw': pswd
        }
        print(data)
        query = {'mail': {'$eq': data['mail']}}
        docs = my_database.get_query_result(query)
        print(docs)
        print(len(docs.all()))
        if (len(docs.all()) == 0):
            url = my_database.create_document(data)
                    return  render_template("register.html",  pred="
Registration Successful , please login using your details ")
        else:
              return render_template('register.html', pred=" You are
already a member , please login using your details ")
    else:
        return render_template('register.html')


@ app.route('/login', methods=['GET','POST'])
def login():
    if request.method == "GET":
        user = request.args.get('mail')
        passw = request.args.get('pass')
```

```
        print(user, passw)
        query = {'mail': {'$eq': user}}
        docs = my_database.get_query_result(query)
        print(docs)
        print(len(docs.all()))
        if (len(docs.all()) == 0):
            return render_template('login.html', pred="")
        else:
                    if ((user == docs[0][0]['mail'] and passw ==
docs[0][0]['psw'])):
                flash("Logged in as " + str(user))
                 return render_template('index.html', pred="Logged in
as "+str(user), vis ="hidden", vis2="visible")
            else:
                     return render_template('login.html', pred="The
password is wrong.")
    else:
        return render_template('login.html')


@ app.route('/logout')
def logout():
    return render_template('logout.html')

@app.route("/predict",methods=["GET", "POST"])
def predict():
    if request.method == "POST":
        f = request.files['file']
        # getting the current path 1.e where app.py is present
        basepath = os.path.dirname(__file__)
        #print ( " current path " , basepath )
         # from anywhere in the system we can give image but we want
that
           filepath = os.path.join(str(basepath), 'Testing Images',
str(f.filename))
        #print ( " upload folder is " , filepath )
        f.save(filepath)
        img = image.load_img(filepath, target_size=(299, 299))
        x = image.img_to_array(img)  # ing to array
          x = np.expand_dims(x, axis=0)  # used for adding one more
dimension
        #print ( x )
```

```
        img_data = preprocess_input(x)
        prediction = np.argmax(model.predict(img_data), axis=1)
        index = [' No Diabetic Retinopathy ', ' Mild NPDR ',
                 ' Moderate NPDR ', ' Severe NPDR ', ' Proliferative
DR ']
        result = str(index[prediction[0]])
        print(result)
         return render_template('prediction.html', prediction=result,
fname = filepath)
    else:
        return render_template("prediction.html")

if __name__ == "__main__":
    app.debug = False
    app.run()
```