Project Report

1. INTRODUCTION

1.1 Project Overview

Diabetic Retinopathy (DR) is a common complication of diabetes mellitus, which causes lesions on the retina that affect vision. If it is not detected early, it can lead to blindness. Unfortunately, DR is not a reversible process, and treatment only sustains vision. DR early detection and treatment can significantly reduce the risk of vision loss. The manual diagnosis process of DR retina fundus images by ophthalmologists is time, effort and cost-consuming and prone to misdiagnosis unlike computer-aided diagnosis systems.

Transfer learning has become one of the most common techniques that has achieved better performance in many areas, especially in medical image analysis and classification. We used Transfer Learning techniques like Inception V3, Resnet50, Xception V3 that are more widely used as a transfer learning method in medical image analysis and they are highly effective.

1.2 Purpose

The manual diagnosis process of DR retina fundus images by ophthalmologists is time, effort and cost-consuming and prone to misdiagnosis unlike computer-aided diagnosis systems. DR early detection and treatment can significantly reduce the risk of vision loss. AI can save the manual effort and cost and also potentially have more accuracy than human experts, thus improving value of service.

2. LITERATURE SURVEY

2.1 Existing problem

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

2.2 References

1. Mushtaq, G., & Siddiqui, F. (2021, February). Detection of diabetic retinopathy using deep learning methodology. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1070, No. 1, p. 012049). IOP Publishing.

2. M. Z. Atwany, A. H. Sahyoun and M. Yaqub, "Deep Learning Techniques for Diabetic Retinopathy Classification: A Survey," in ***IEEE Access***, vol. 10, pp. 28642-28655, 2022, doi: 10.1109/ACCESS.2022.3157632.

3. Tsiknakis, N., Theodoropoulos, D., Manikis, G., Ktistakis, E., Boutsora, O., Berto, A., ... &

Marias, K. (2021). Deep learning for diabetic retinopathy detection and classification based on fundus images: A review. Computers in Biology and Medicine, 135, 104599.

4. Johari, M. H., Hassan, H. A., Yassin, A. I. M., Tahir, N. M., Zabidi, A., & Rizman, Z. I. & Wahab, NA (2018). Early detection of diabetic retinopathy by using deep learning neural network. *International Journal of Engineering and Technology (UAE)*, *7*(4), 198-201.

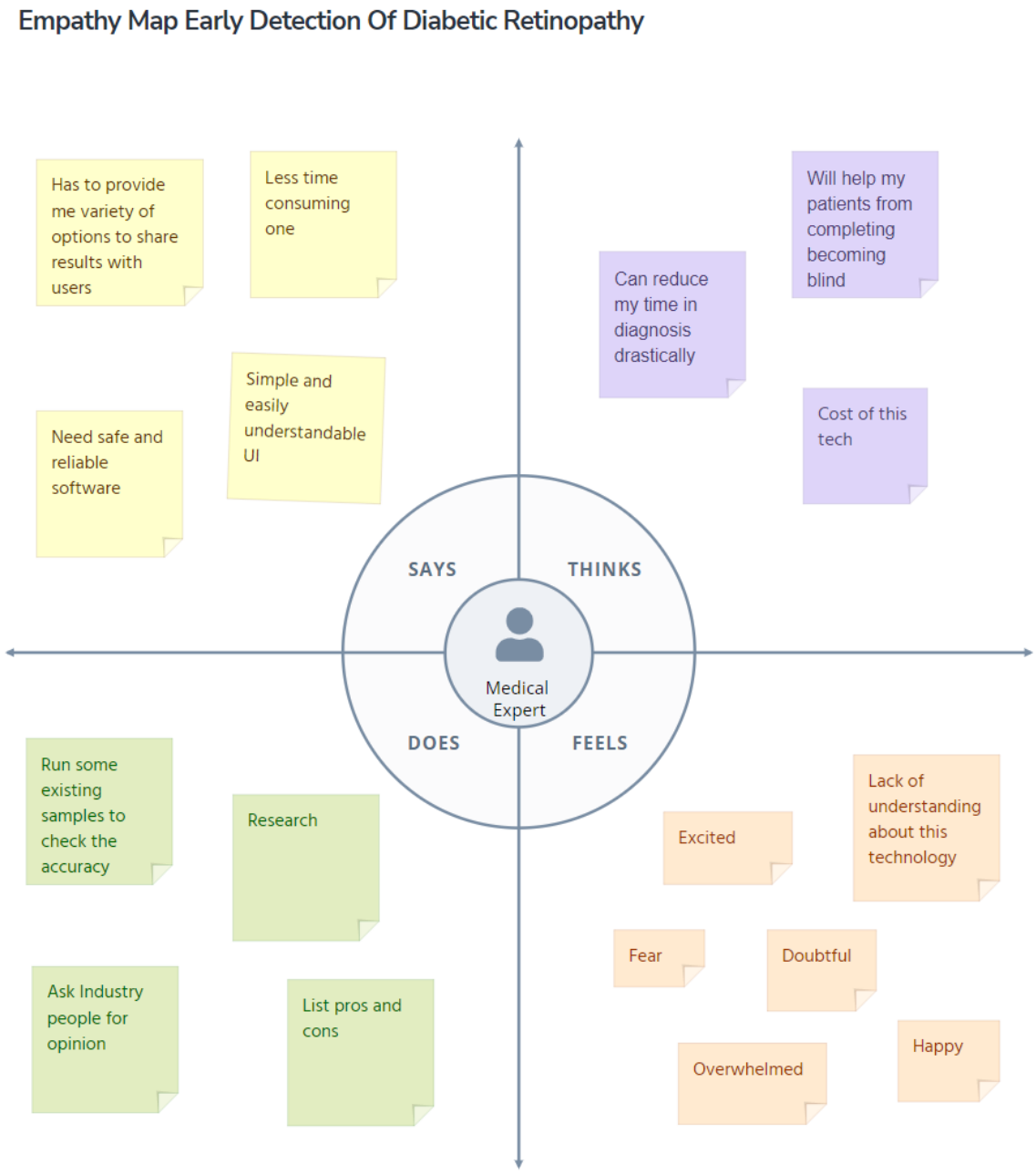
5. Sahlsten, J., Jaskari, J., Kivinen, J., Turunen, L., Jaanio, E., Hietala, K., & Kaski, K. (2019). Deep learning fundus image analysis for diabetic retinopathy and macular edema grading. *Scientific reports*, *9*(1), 1-11.

2.3 Problem Statement Definition

Diabetic Retinopathy (DR) is a degenerative disease that impacts the eyes and is a consequence of Diabetes mellitus, where high blood glucose levels induce lesions on the eye retina. the images of retina are today used for manual disease screening and diagnosis. This labour-intensive task could greatly benefit from automatic detection using deep learning technique. Transfer learning has become one of the most common techniques that has achieved better performance in many areas, especially in medical image analysis and classification. We used Transfer Learning techniques like Inception V3,Resnet50,Xception V3 that are more widely used as a transfer learning method in medical image analysis and they are highly effective.

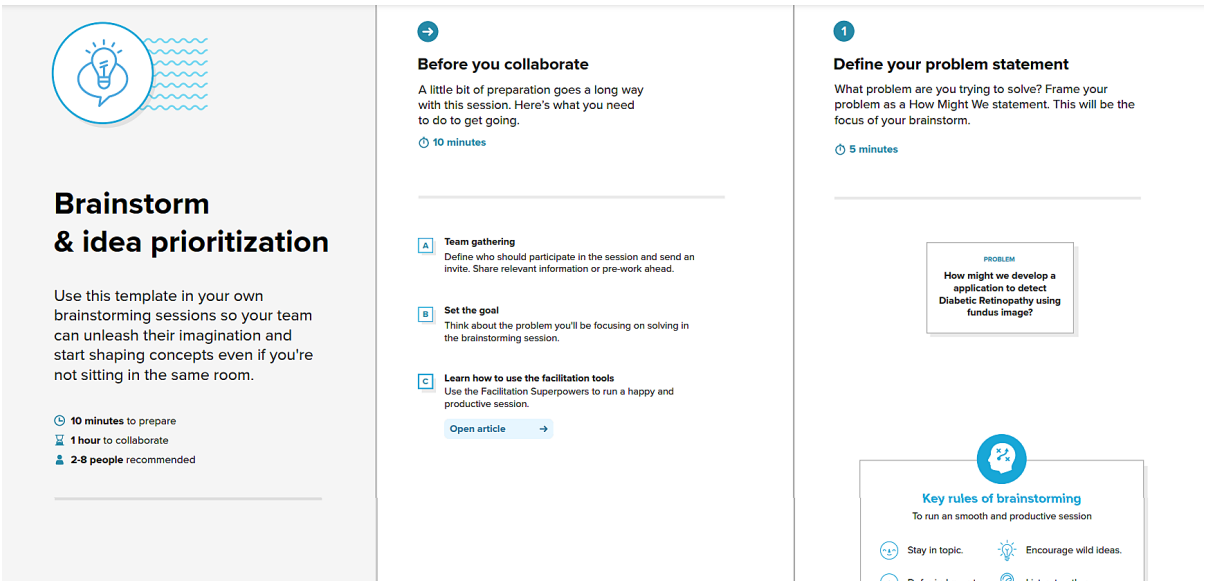
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

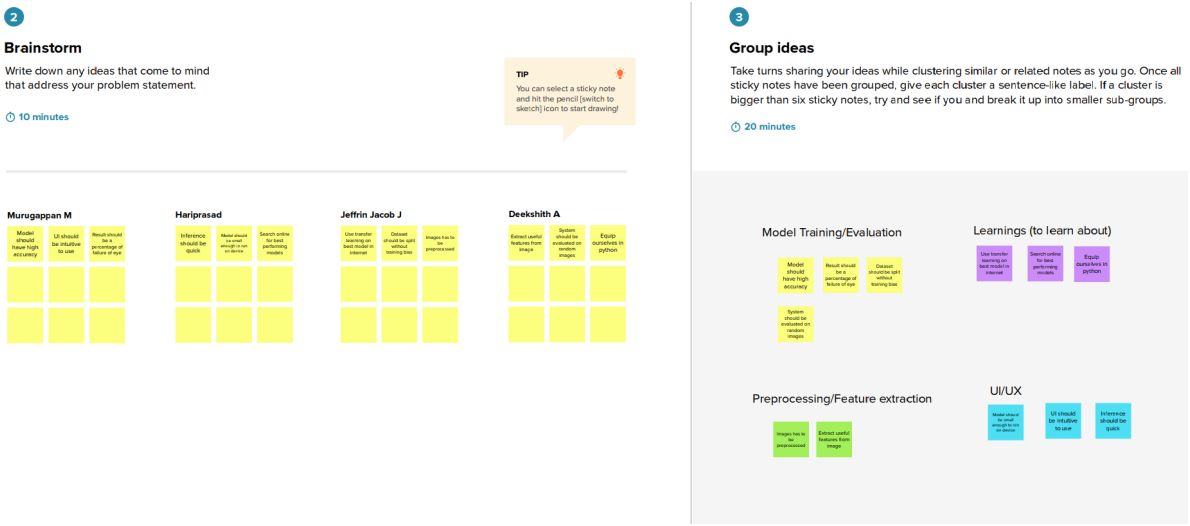


3.2 Ideation & Brainstorming

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



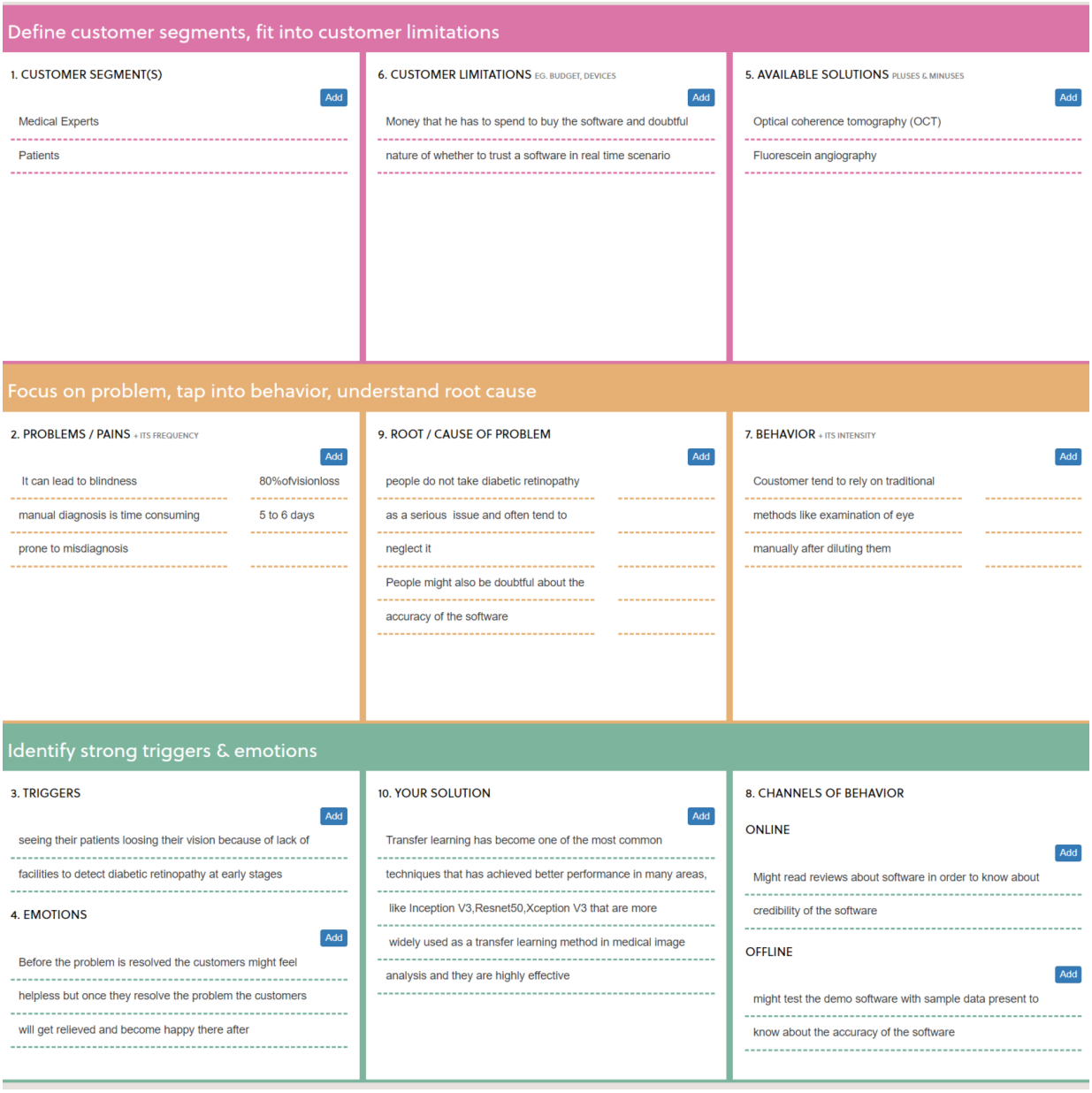
Step-3: Idea Prioritization

3.3 Proposed Solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	**develop a application to detect Diabetic Retinopathy using fundus image**

2.	Idea / Solution description	Use transfer learning (inception v3) on a standard cnn architecture
3.	Novelty / Uniqueness	Transfer learning helps increase accuracy of our system as prior training knowledge is included.
4.	Social Impact / Customer Satisfaction	People with Diabetic Retinopathy are diagnosed at a much earlier stage and can take treatment before complete blindness.
5.	Business Model (Revenue Model)	Sell application for per api call price, this enables us to reach any doctor with a fundus image.
6.	Scalability of the Solution	Requires only deployment servers, highly scalable on cloud platforms.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

S. No	Features	Functionality	Description	Acceptance criteria
1.	Upload	Upload image of eye	upload image of eyes to portal	I can upload or take image
2.	Prediction	See predicted label of my image	Image is clearly mentioned above and label is mentioned below making it clear for which image the prediction is.	I can receive the diagnosis
3.	Authentication	Login page	Enter username and password and login to dashboard	I can receive the severity of the

				retinopathy
		Logout button	Logout from dashboard and redirected to login page, even on clicking back button.	I can receive the suggested remedy

4.2 Non-Functional requirements

Server: python based with gpu (preferrable). Preferable to deploy on cloud providers like IBM cloud with cloud db.
Client: Any device with 4gb of ram and a camera or storage (if saved images are to be uploaded).

Hardware/Software Requirements
Server: any virtual instance with at least 4gb of ram in gpu and python/pytorch installed with flask.
Database: any managed db service with mongodb installed

Performance Requirements

Server: 99% uptime and below 100ms latency for 95% requests. Model inference should take less than 5 seconds.
Client: less than 5% of crashes and good handling of error states.

Security Requirements

Application uses https for all communications with backend and db calls are encrypted.

Assumptions / Constraints

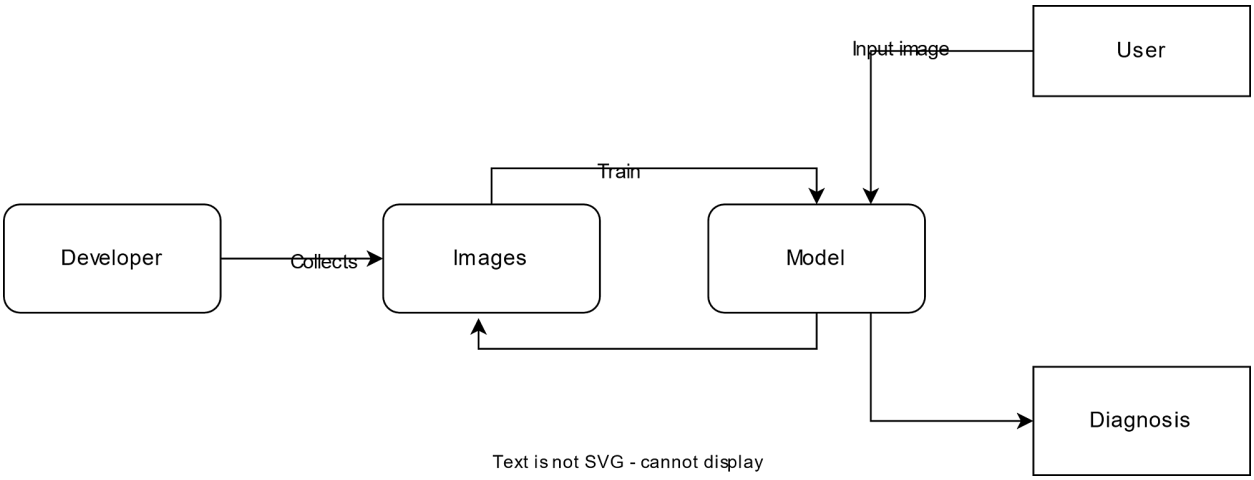
The user's device uses a fairly modern browser chrome 76+, firefox 20+, safari 15+. IE not supported.

Compliance Requirements

Application does not store user images and does in memory inference for model prediction.

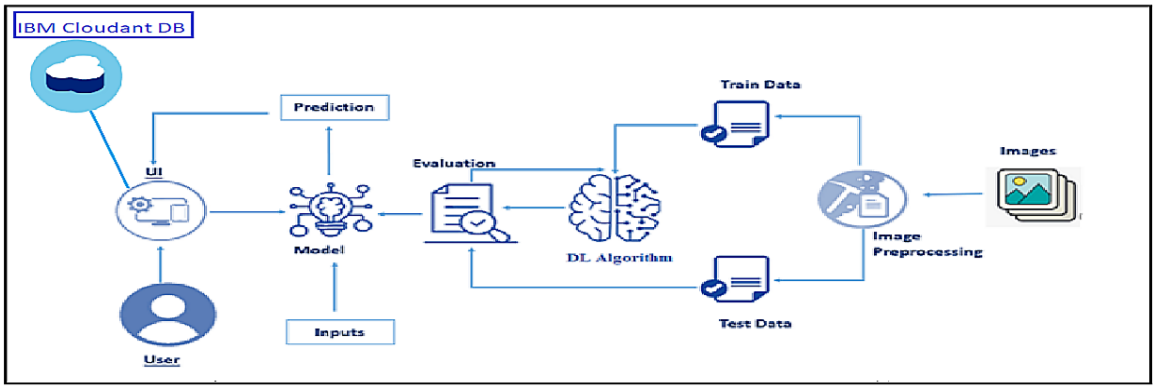
5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture

Solution Architecture:



The UI mentioned here can range from a simple scanner application to a large application in computer with scanners for capturing the images of eyes

The Model used here is based on the principle of transfer learning and the model chosen by our team is inception V3 to train and test data

TECHNOLOGY ARCHITECTURE

S.No	Component	Description	Technology
1.	User Interface	The user interacts with the components of the application's webUI	HTML, CSS, JS, Flask
2.	Application Logic	Logic for processing the data received	Python
3.	Database	Store user data	Postgresql

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story /Task	Acceptance criteria	Priority	Release
Common User	Dashboard	USN-1	As a user, I can I must be able to upload image of my eyes	I can upload or take image	High	Sprint-1
		USN-2	As a user, I will receive the diagnosis as to whether I have retinopathy or not	I can receive the diagnosis	High	Sprint-1
		USN-3	As a user, I receive the severity of the retinopathy	I can receive the severity of the retinopathy	Medium	Sprint-2
		USN-4	As a user, I can receive the suggested remedy	I can receive the suggested remedy	Medium	Sprint-2

6. PROJECT PLANNING & SCHEDULING

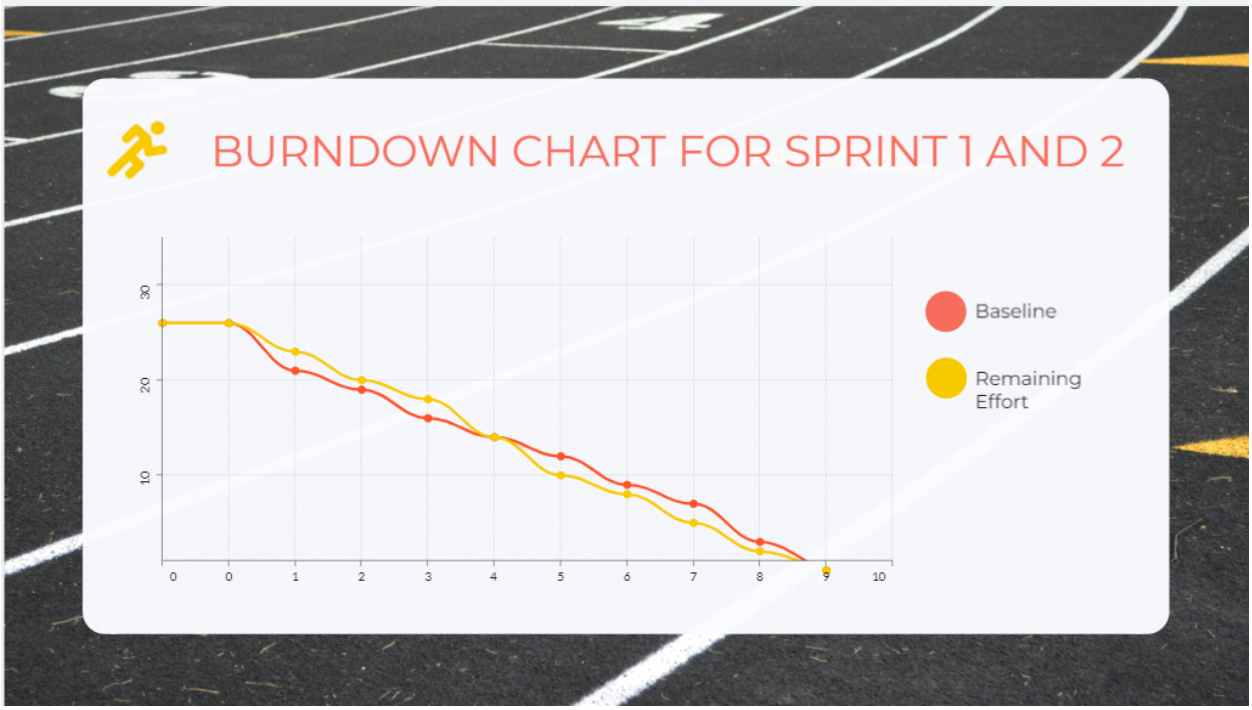
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task	Story Points	Priority	Team Members
Sprint-1	Home/Index page	USN-1	As a user, I should be able to view the landing pege provided with login, register options into the portal	5	Medium	
Sprint-1	Register	USN-2	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	
		USN-3	As a user, I will receive confirmation email once I have registered for the application	3	Low	
Sprint-2	Login	USN-4	As a user, I can log into the application by entering email & password	5	Medium	
Sprint-2	Dashboard	USN-5	As a user, I should be able to upload images and provided with an option to predict the uploaded image	5	High	
		USN-6	As a user, I should be able to logout from the dash board	3	Low	

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	13	6 Days	28 Oct 2022	03 Nov 2022	13	03 Nov 2022
Sprint-2	13	6 Days	04 Nov 2022	10 Nov 2022		

6.3 Reports from JIRA



7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

Prediction of diabetic retinopathy from image

```
@app.route('/predictImage', methods=['POST'])
def predictImage():
    # print(request.files)
    f = request.files['image']
    basepath = os.path.dirname(__file__)
    filepath = os.path.join(basepath, 'uploads', f.filename)  # type: ignore
    print(filepath)
    f.save(filepath)

    img = image.load_img(filepath, target_size=(299, 299))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    img_data = preprocess_input(x)
    _prediction = np.argmax(model.predict(img_data), axis=1)
```

```
index = ['No Diabetic Retinopathy', 'Mild DR',
        'Moderate DR', 'Severe DR', 'Proliferative DR']

result = str(index[_prediction[0]])
print(result)
return render_template('prediction.html', prediction=result)
```

1. The image from request object is stored locally.
2. Image loaded and resized to 299x299 pixels size.
3. Image changed to 1D array for efficient processing using numpy.
4. model prediction is done
5. corresponding value is returned as html file for displaying to user.

7.2 Feature 2

Transfer Learning

```
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)
```

The pretrained Xception model is loaded and set as not trainable so that model does not change weight for our training set. Then, a dense layer That outputs 5 values is created with Xception output as input and model is trained.

7.3 Database Schema (if Applicable)

User Table

Field name	**Type**
username	string
name	string
password	string

8. TESTING

8.1 Test Cases

Test Id	Test case	Test data	Expected result	Actual result	Pass
1	Login	Username: murugu Password: password	Redirects to Dashboard Page	Redirects to Dashboard Page	Yes
2	Upload	Image of retina	Received in backend by model and inference returned	Received in backend and inference returned	Yes
3	Label	Dashboard after upload	Label clearly displayed below image	Label clearly displayed below image	Yes
4	Logout	Click on logout button	User redirected to login page	User redirected to login page	Yes

8.2 User Acceptance Testing

USN-1:

Story: As a user, I must be able to upload image of my eyes

Acceptance criteria: I can upload or take image to the portal

USN-2:

Story: As a user, I will receive the diagnosis as to whether I have retinopathy or not.

Acceptance criteria: I can receive the diagnosis

USN-3:

Story: As a user, I receive the severity of the retinopathy

Acceptance criteria: I can receive the severity of the retinopathy

USN-4:

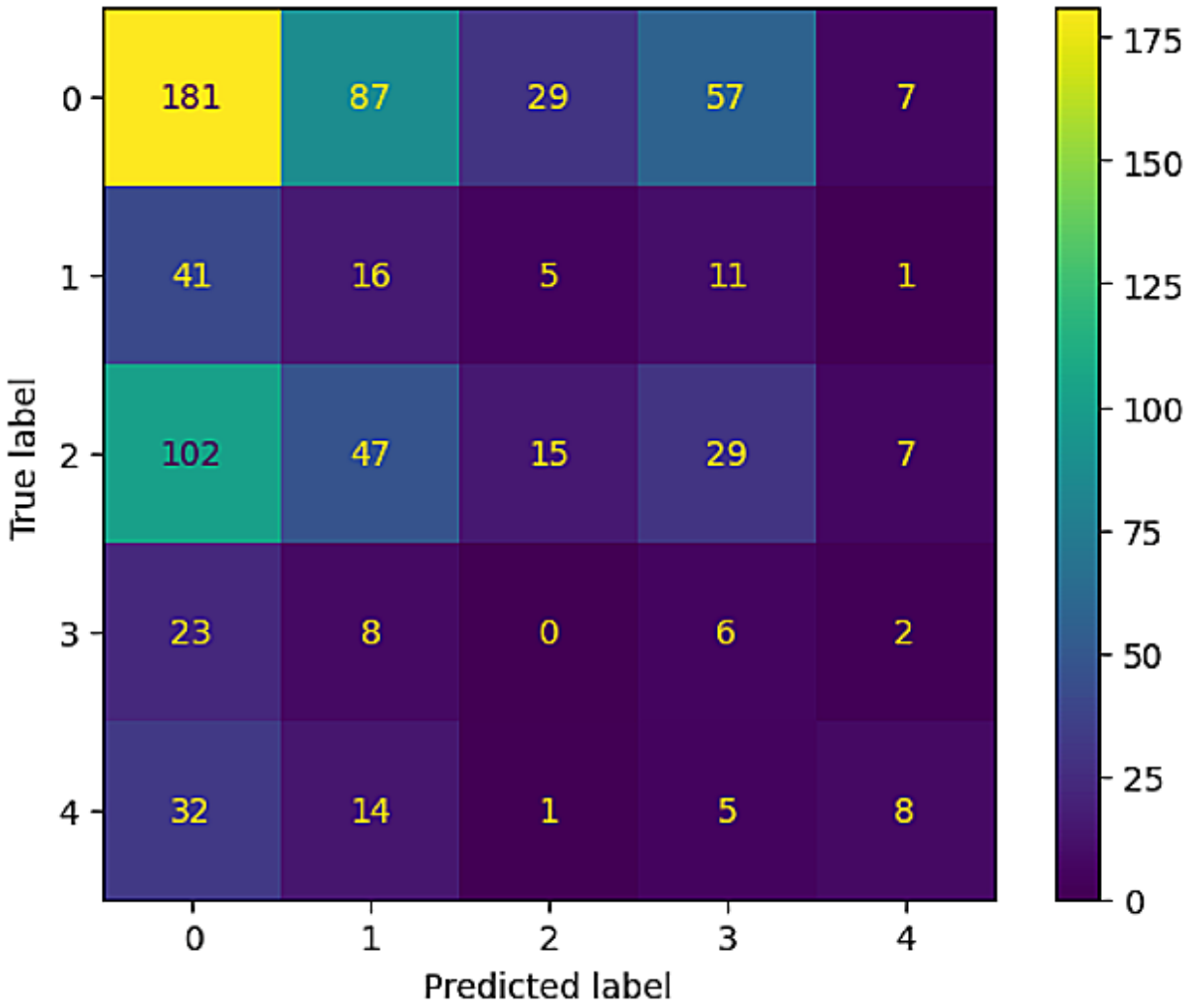
Story: As a user, I can receive the suggested remedy

Acceptance criteria: I can receive the suggested remedy

9. RESULTS

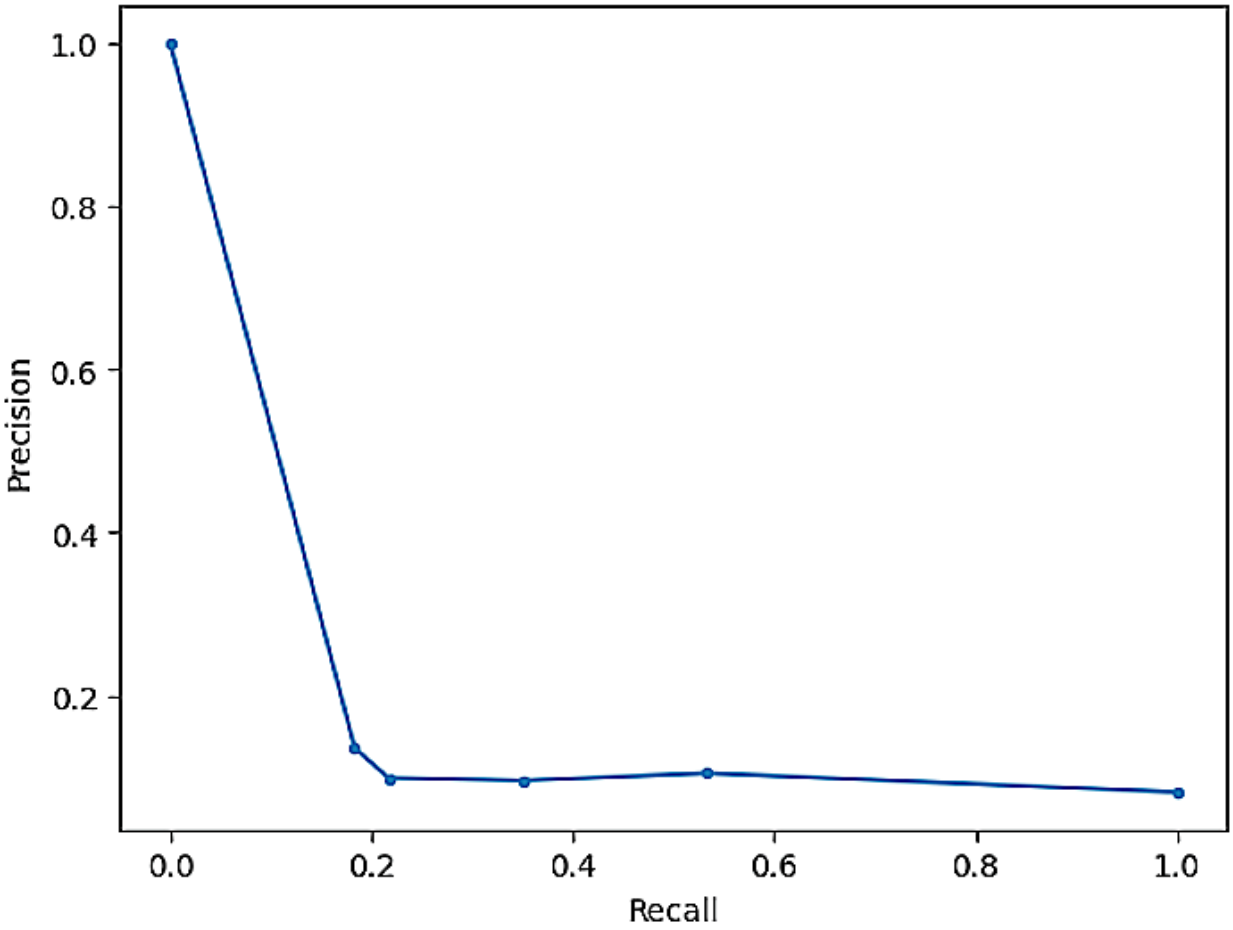
9.1 Performance Metrics

- Confusion matrix:

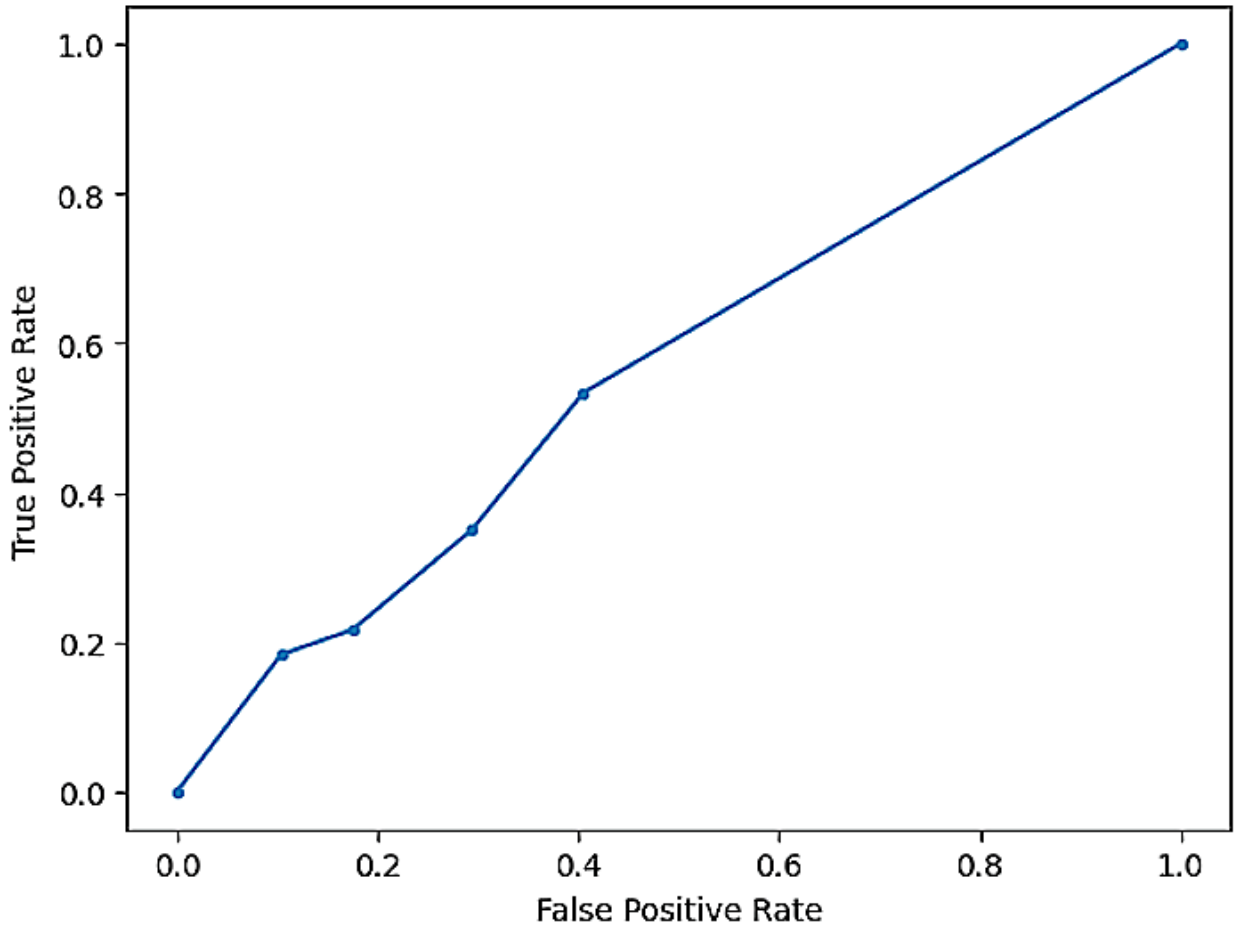


- Accuracy - 30.79%
- Precision - 0.3551145524908687
- Recall - 0.3079019073569482
- F1 score - 0.30613229519605784

- PR curve



- ROC curve



10. ADVANTAGES & DISADVANTAGES

Adevantages:

1. Our model uses pretrained weights from resnet dataset. So, has good context using transfer learning
2. Our application is user friendly and easy to use

Disadvantages:

1. Our model has low accuracy
2. Running inference takes time (Around 2s) and money (faster with gpu, which costs more money)

11. CONCLUSION

We studied the problem statement carefully and looked at existing solutions and decided to proceed with transfer learning technique. We used Xception V3 as the base model and trained a deep layer on top of it for predictions. Our model had good precision and recall values compared to other available solutions by 5% and 7% respectively. We deployed our model in a simple flask application with

authentication for people to be able to use our inference.

12. **FUTURE SCOPE**

Some variations of the model can be tried like adding a extra deep learning layer and fine tuning on top of wrong prediction data again. We can also add more data to the dataset by label from ophthalmologists and other labels. The user interface of the application can be improved by adding more design changes and based on user feedback.

13. **APPENDIX**

Source Code

Model Training

In [1]:

```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
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
```

In [2]:

```
imageSize = [299, 299]
```

In [3]:

```
#importing data set
train_datagen = ImageDataGenerator(rescale=1/255, shear_range=0.2,
zoom_range=0.2, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [4]:

```
training_set = train_datagen.flow_from_directory(
    "dataset\\preprocessed dataset\\preprocessed dataset\\training",
target_size = (299, 299), batch_size=32, class_mode = 'categorical')

test_set = train_datagen.flow_from_directory(
    "dataset\\preprocessed dataset\\preprocessed dataset\\testing",
```

```
target_size=(299, 299), batch_size=32, class_mode='categorical')
Found 3662 images belonging to 5 classes.
Found 734 images belonging to 5 classes.
```

In [5]:

```
xception = Xception(input_shape=imageSize + [3], weights='imagenet',
include_top=False)
```

In [6]:

```
for layer in xception.layers:
    layer.trainable = False
```

In [7]:

```
x = Flatten()(xception.output)
```

In [8]:

```
prediction = Dense(5, activation="softmax")(x)

model = Model(inputs = xception.input, outputs=prediction)
```

In [9]:

```
model.summary()
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
Connected to
==================================================================
==================
 input_1 (InputLayer)        [(None, 299, 299, 3       0
[]
                             )]

 block1_conv1 (Conv2D)       (None, 149, 149, 32       864
['input_1[0][0]']
                             )

 block1_conv1_bn (BatchNormaliz  (None, 149, 149, 32       128
['block1_conv1[0][0]']
 ation)                      )

 block1_conv1_act (Activation)  (None, 149, 149, 32       0
```

```
['block1_conv1_bn[0][0]']
                                )

 block1_conv2 (Conv2D)          (None, 147, 147, 64  18432
['block1_conv1_act[0][0]']
                                )

 block1_conv2_bn (BatchNormaliz  (None, 147, 147, 64  256
['block1_conv2[0][0]']
 ation)                         )

 block1_conv2_act (Activation)  (None, 147, 147, 64  0
['block1_conv2_bn[0][0]']
                                )

 block2_sepconv1 (SeparableConv  (None, 147, 147, 12  8768
['block1_conv2_act[0][0]']
 2D)                            8)

 block2_sepconv1_bn (BatchNorma  (None, 147, 147, 12  512
['block2_sepconv1[0][0]']
 lization)                      8)

 block2_sepconv2_act (Activatio  (None, 147, 147, 12  0
['block2_sepconv1_bn[0][0]']
 n)                             8)

 block2_sepconv2 (SeparableConv  (None, 147, 147, 12  17536
['block2_sepconv2_act[0][0]']
 2D)                            8)

 block2_sepconv2_bn (BatchNorma  (None, 147, 147, 12  512
['block2_sepconv2[0][0]']
 lization)                      8)

 conv2d (Conv2D)                (None, 74, 74, 128)  8192
['block1_conv2_act[0][0]']

 block2_pool (MaxPooling2D)     (None, 74, 74, 128)  0
['block2_sepconv2_bn[0][0]']

 batch_normalization (BatchNorm  (None, 74, 74, 128)  512
['conv2d[0][0]']
```

```
 alization)

 add (Add)                      (None, 74, 74, 128)  0
['block2_pool[0][0]',

'batch_normalization[0][0]']

 block3_sepconv1_act (Activatio  (None, 74, 74, 128)  0          ['add[0][0]']
 n)

 block3_sepconv1 (SeparableConv  (None, 74, 74, 256)  33920
['block3_sepconv1_act[0][0]']
 2D)

 block3_sepconv1_bn (BatchNorma  (None, 74, 74, 256)  1024
['block3_sepconv1[0][0]']
 lization)

 block3_sepconv2_act (Activatio  (None, 74, 74, 256)  0
['block3_sepconv1_bn[0][0]']
 n)

 block3_sepconv2 (SeparableConv  (None, 74, 74, 256)  67840
['block3_sepconv2_act[0][0]']
 2D)

 block3_sepconv2_bn (BatchNorma  (None, 74, 74, 256)  1024
['block3_sepconv2[0][0]']
 lization)

 conv2d_1 (Conv2D)              (None, 37, 37, 256)  32768       ['add[0][0]']

 block3_pool (MaxPooling2D)     (None, 37, 37, 256)  0
['block3_sepconv2_bn[0][0]']

 batch_normalization_1 (BatchNo  (None, 37, 37, 256)  1024
['conv2d_1[0][0]']
 rmalization)

 add_1 (Add)                    (None, 37, 37, 256)  0
['block3_pool[0][0]',

'batch_normalization_1[0][0]']
```

```
 block4_sepconv1_act (Activatio  (None, 37, 37, 256)  0
['add_1[0][0]']
 n)

 block4_sepconv1 (SeparableConv  (None, 37, 37, 728)  188672
['block4_sepconv1_act[0][0]']
 2D)

 block4_sepconv1_bn (BatchNorma  (None, 37, 37, 728)  2912
['block4_sepconv1[0][0]']
 lization)

 block4_sepconv2_act (Activatio  (None, 37, 37, 728)  0
['block4_sepconv1_bn[0][0]']
 n)

 block4_sepconv2 (SeparableConv  (None, 37, 37, 728)  536536
['block4_sepconv2_act[0][0]']
 2D)

 block4_sepconv2_bn (BatchNorma  (None, 37, 37, 728)  2912
['block4_sepconv2[0][0]']
 lization)

 conv2d_2 (Conv2D)              (None, 19, 19, 728)  186368
['add_1[0][0]']

 block4_pool (MaxPooling2D)     (None, 19, 19, 728)  0
['block4_sepconv2_bn[0][0]']

 batch_normalization_2 (BatchNo  (None, 19, 19, 728)  2912
['conv2d_2[0][0]']
 rmalization)

 add_2 (Add)                    (None, 19, 19, 728)  0
['block4_pool[0][0]',

'batch_normalization_2[0][0]']

 block5_sepconv1_act (Activatio  (None, 19, 19, 728)  0
['add_2[0][0]']
 n)
```

```
 block5_sepconv1 (SeparableConv  (None, 19, 19, 728)  536536
['block5_sepconv1_act[0][0]']
 2D)

 block5_sepconv1_bn (BatchNorma  (None, 19, 19, 728)  2912
['block5_sepconv1[0][0]']
 lization)

 block5_sepconv2_act (Activatio  (None, 19, 19, 728)  0
['block5_sepconv1_bn[0][0]']
 n)

 block5_sepconv2 (SeparableConv  (None, 19, 19, 728)  536536
['block5_sepconv2_act[0][0]']
 2D)

 block5_sepconv2_bn (BatchNorma  (None, 19, 19, 728)  2912
['block5_sepconv2[0][0]']
 lization)

 block5_sepconv3_act (Activatio  (None, 19, 19, 728)  0
['block5_sepconv2_bn[0][0]']
 n)

 block5_sepconv3 (SeparableConv  (None, 19, 19, 728)  536536
['block5_sepconv3_act[0][0]']
 2D)

 block5_sepconv3_bn (BatchNorma  (None, 19, 19, 728)  2912
['block5_sepconv3[0][0]']
 lization)

 add_3 (Add)                    (None, 19, 19, 728)  0
['block5_sepconv3_bn[0][0]',

'add_2[0][0]']

 block6_sepconv1_act (Activatio  (None, 19, 19, 728)  0
['add_3[0][0]']
 n)

 block6_sepconv1 (SeparableConv  (None, 19, 19, 728)  536536
```

```
['block6_sepconv1_act[0][0]']
 2D)

 block6_sepconv1_bn (BatchNorma  (None, 19, 19, 728)  2912
['block6_sepconv1[0][0]']
 lization)

 block6_sepconv2_act (Activatio  (None, 19, 19, 728)  0
['block6_sepconv1_bn[0][0]']
 n)

 block6_sepconv2 (SeparableConv  (None, 19, 19, 728)  536536
['block6_sepconv2_act[0][0]']
 2D)

 block6_sepconv2_bn (BatchNorma  (None, 19, 19, 728)  2912
['block6_sepconv2[0][0]']
 lization)

 block6_sepconv3_act (Activatio  (None, 19, 19, 728)  0
['block6_sepconv2_bn[0][0]']
 n)

 block6_sepconv3 (SeparableConv  (None, 19, 19, 728)  536536
['block6_sepconv3_act[0][0]']
 2D)

 block6_sepconv3_bn (BatchNorma  (None, 19, 19, 728)  2912
['block6_sepconv3[0][0]']
 lization)

 add_4 (Add)                    (None, 19, 19, 728)  0
['block6_sepconv3_bn[0][0]',

'add_3[0][0]']

 block7_sepconv1_act (Activatio  (None, 19, 19, 728)  0
['add_4[0][0]']
 n)

 block7_sepconv1 (SeparableConv  (None, 19, 19, 728)  536536
['block7_sepconv1_act[0][0]']
 2D)
```

```
 block7_sepconv1_bn (BatchNorma  (None, 19, 19, 728)  2912
['block7_sepconv1[0][0]']
 lization)

 block7_sepconv2_act (Activatio  (None, 19, 19, 728)  0
['block7_sepconv1_bn[0][0]']
 n)

 block7_sepconv2 (SeparableConv  (None, 19, 19, 728)  536536
['block7_sepconv2_act[0][0]']
 2D)

 block7_sepconv2_bn (BatchNorma  (None, 19, 19, 728)  2912
['block7_sepconv2[0][0]']
 lization)

 block7_sepconv3_act (Activatio  (None, 19, 19, 728)  0
['block7_sepconv2_bn[0][0]']
 n)

 block7_sepconv3 (SeparableConv  (None, 19, 19, 728)  536536
['block7_sepconv3_act[0][0]']
 2D)

 block7_sepconv3_bn (BatchNorma  (None, 19, 19, 728)  2912
['block7_sepconv3[0][0]']
 lization)

 add_5 (Add)                    (None, 19, 19, 728)  0
['block7_sepconv3_bn[0][0]',

'add_4[0][0]']

 block8_sepconv1_act (Activatio  (None, 19, 19, 728)  0
['add_5[0][0]']
 n)

 block8_sepconv1 (SeparableConv  (None, 19, 19, 728)  536536
['block8_sepconv1_act[0][0]']
 2D)

 block8_sepconv1_bn (BatchNorma  (None, 19, 19, 728)  2912
```

```
['block8_sepconv1[0][0]']
 lization)

 block8_sepconv2_act (Activatio  (None, 19, 19, 728)  0
['block8_sepconv1_bn[0][0]']
 n)

 block8_sepconv2 (SeparableConv  (None, 19, 19, 728)  536536
['block8_sepconv2_act[0][0]']
 2D)

 block8_sepconv2_bn (BatchNorma  (None, 19, 19, 728)  2912
['block8_sepconv2[0][0]']
 lization)

 block8_sepconv3_act (Activatio  (None, 19, 19, 728)  0
['block8_sepconv2_bn[0][0]']
 n)

 block8_sepconv3 (SeparableConv  (None, 19, 19, 728)  536536
['block8_sepconv3_act[0][0]']
 2D)

 block8_sepconv3_bn (BatchNorma  (None, 19, 19, 728)  2912
['block8_sepconv3[0][0]']
 lization)

 add_6 (Add)                    (None, 19, 19, 728)  0
['block8_sepconv3_bn[0][0]',

'add_5[0][0]']

 block9_sepconv1_act (Activatio  (None, 19, 19, 728)  0
['add_6[0][0]']
 n)

 block9_sepconv1 (SeparableConv  (None, 19, 19, 728)  536536
['block9_sepconv1_act[0][0]']
 2D)

 block9_sepconv1_bn (BatchNorma  (None, 19, 19, 728)  2912
['block9_sepconv1[0][0]']
 lization)
```

```
 block9_sepconv2_act (Activatio  (None, 19, 19, 728)  0
['block9_sepconv1_bn[0][0]']
 n)

 block9_sepconv2 (SeparableConv  (None, 19, 19, 728)  536536
['block9_sepconv2_act[0][0]']
 2D)

 block9_sepconv2_bn (BatchNorma  (None, 19, 19, 728)  2912
['block9_sepconv2[0][0]']
 lization)

 block9_sepconv3_act (Activatio  (None, 19, 19, 728)  0
['block9_sepconv2_bn[0][0]']
 n)

 block9_sepconv3 (SeparableConv  (None, 19, 19, 728)  536536
['block9_sepconv3_act[0][0]']
 2D)

 block9_sepconv3_bn (BatchNorma  (None, 19, 19, 728)  2912
['block9_sepconv3[0][0]']
 lization)

 add_7 (Add)                    (None, 19, 19, 728)  0
['block9_sepconv3_bn[0][0]',

'add_6[0][0]']

 block10_sepconv1_act (Activati  (None, 19, 19, 728)  0
['add_7[0][0]']
 on)

 block10_sepconv1 (SeparableCon  (None, 19, 19, 728)  536536
['block10_sepconv1_act[0][0]']
 v2D)

 block10_sepconv1_bn (BatchNorm  (None, 19, 19, 728)  2912
['block10_sepconv1[0][0]']
 alization)

 block10_sepconv2_act (Activati  (None, 19, 19, 728)  0
```

```
['block10_sepconv1_bn[0][0]']
 on)

 block10_sepconv2 (SeparableCon  (None, 19, 19, 728)  536536
['block10_sepconv2_act[0][0]']
 v2D)

 block10_sepconv2_bn (BatchNorm  (None, 19, 19, 728)  2912
['block10_sepconv2[0][0]']
 alization)

 block10_sepconv3_act (Activati  (None, 19, 19, 728)  0
['block10_sepconv2_bn[0][0]']
 on)

 block10_sepconv3 (SeparableCon  (None, 19, 19, 728)  536536
['block10_sepconv3_act[0][0]']
 v2D)

 block10_sepconv3_bn (BatchNorm  (None, 19, 19, 728)  2912
['block10_sepconv3[0][0]']
 alization)

 add_8 (Add)                    (None, 19, 19, 728)  0
['block10_sepconv3_bn[0][0]',

'add_7[0][0]']

 block11_sepconv1_act (Activati  (None, 19, 19, 728)  0
['add_8[0][0]']
 on)

 block11_sepconv1 (SeparableCon  (None, 19, 19, 728)  536536
['block11_sepconv1_act[0][0]']
 v2D)

 block11_sepconv1_bn (BatchNorm  (None, 19, 19, 728)  2912
['block11_sepconv1[0][0]']
 alization)

 block11_sepconv2_act (Activati  (None, 19, 19, 728)  0
['block11_sepconv1_bn[0][0]']
 on)
```

```
 block11_sepconv2 (SeparableCon  (None, 19, 19, 728)  536536
['block11_sepconv2_act[0][0]']
 v2D)

 block11_sepconv2_bn (BatchNorm  (None, 19, 19, 728)  2912
['block11_sepconv2[0][0]']
 alization)

 block11_sepconv3_act (Activati  (None, 19, 19, 728)  0
['block11_sepconv2_bn[0][0]']
 on)

 block11_sepconv3 (SeparableCon  (None, 19, 19, 728)  536536
['block11_sepconv3_act[0][0]']
 v2D)

 block11_sepconv3_bn (BatchNorm  (None, 19, 19, 728)  2912
['block11_sepconv3[0][0]']
 alization)

 add_9 (Add)                    (None, 19, 19, 728)  0
['block11_sepconv3_bn[0][0]',

'add_8[0][0]']

 block12_sepconv1_act (Activati  (None, 19, 19, 728)  0
['add_9[0][0]']
 on)

 block12_sepconv1 (SeparableCon  (None, 19, 19, 728)  536536
['block12_sepconv1_act[0][0]']
 v2D)

 block12_sepconv1_bn (BatchNorm  (None, 19, 19, 728)  2912
['block12_sepconv1[0][0]']
 alization)

 block12_sepconv2_act (Activati  (None, 19, 19, 728)  0
['block12_sepconv1_bn[0][0]']
 on)

 block12_sepconv2 (SeparableCon  (None, 19, 19, 728)  536536
```

```
['block12_sepconv2_act[0][0]']
 v2D)

 block12_sepconv2_bn (BatchNorm  (None, 19, 19, 728)  2912
['block12_sepconv2[0][0]']
 alization)

 block12_sepconv3_act (Activati  (None, 19, 19, 728)  0
['block12_sepconv2_bn[0][0]']
 on)

 block12_sepconv3 (SeparableCon  (None, 19, 19, 728)  536536
['block12_sepconv3_act[0][0]']
 v2D)

 block12_sepconv3_bn (BatchNorm  (None, 19, 19, 728)  2912
['block12_sepconv3[0][0]']
 alization)

 add_10 (Add)                   (None, 19, 19, 728)  0
['block12_sepconv3_bn[0][0]',

'add_9[0][0]']

 block13_sepconv1_act (Activati  (None, 19, 19, 728)  0
['add_10[0][0]']
 on)

 block13_sepconv1 (SeparableCon  (None, 19, 19, 728)  536536
['block13_sepconv1_act[0][0]']
 v2D)

 block13_sepconv1_bn (BatchNorm  (None, 19, 19, 728)  2912
['block13_sepconv1[0][0]']
 alization)

 block13_sepconv2_act (Activati  (None, 19, 19, 728)  0
['block13_sepconv1_bn[0][0]']
 on)

 block13_sepconv2 (SeparableCon  (None, 19, 19, 1024  752024
['block13_sepconv2_act[0][0]']
 v2D)                           )
```

```
 block13_sepconv2_bn (BatchNorm  (None, 19, 19, 1024  4096
['block13_sepconv2[0][0]']
 alization)                     )

 conv2d_3 (Conv2D)              (None, 10, 10, 1024  745472
['add_10[0][0]']
                                )

 block13_pool (MaxPooling2D)    (None, 10, 10, 1024  0
['block13_sepconv2_bn[0][0]']
                                )

 batch_normalization_3 (BatchNo  (None, 10, 10, 1024  4096
['conv2d_3[0][0]']
 rmalization)                   )

 add_11 (Add)                   (None, 10, 10, 1024  0
['block13_pool[0][0]',
                                )
'batch_normalization_3[0][0]']

 block14_sepconv1 (SeparableCon  (None, 10, 10, 1536  1582080
['add_11[0][0]']
 v2D)                           )

 block14_sepconv1_bn (BatchNorm  (None, 10, 10, 1536  6144
['block14_sepconv1[0][0]']
 alization)                     )

 block14_sepconv1_act (Activati  (None, 10, 10, 1536  0
['block14_sepconv1_bn[0][0]']
 on)                            )

 block14_sepconv2 (SeparableCon  (None, 10, 10, 2048  3159552
['block14_sepconv1_act[0][0]']
 v2D)                           )

 block14_sepconv2_bn (BatchNorm  (None, 10, 10, 2048  8192
['block14_sepconv2[0][0]']
 alization)                     )

 block14_sepconv2_act (Activati  (None, 10, 10, 2048  0
```

```
['block14_sepconv2_bn[0][0]']
 on)                            )

 flatten (Flatten)              (None, 204800)       0
['block14_sepconv2_act[0][0]']

 dense (Dense)                  (None, 5)            1024005
['flatten[0][0]']

==================================================================================
====================
Total params: 21,885,485
Trainable params: 1,024,005
Non-trainable params: 20,861,480
__________________________________________________________________________________
____________________
```

In [10]:

```
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

In [11]:

```
r = model.fit(training_set, validation_data=test_set, epochs=30,
steps_per_epoch=len(training_set)//32, validation_steps=len(test_set)//32)
Epoch 1/30
3/3 [==============================] - 52s 5s/step - loss: 10.3830 - accuracy:
0.2917
Epoch 2/30
3/3 [==============================] - 26s 8s/step - loss: 20.3412 - accuracy:
0.5521
Epoch 3/30
3/3 [==============================] - 24s 8s/step - loss: 15.4075 - accuracy:
0.6042
Epoch 4/30
3/3 [==============================] - 19s 6s/step - loss: 8.4522 - accuracy:
0.4271
Epoch 5/30
3/3 [==============================] - 20s 6s/step - loss: 7.6308 - accuracy:
0.4688
Epoch 6/30
3/3 [==============================] - 19s 5s/step - loss: 4.7706 - accuracy:
0.6146
```

```
Epoch 7/30
3/3 [==============================] - 16s 5s/step - loss: 5.9880 - accuracy:
0.6667
Epoch 8/30
3/3 [==============================] - 15s 5s/step - loss: 6.4603 - accuracy:
0.6667
Epoch 9/30
3/3 [==============================] - 12s 4s/step - loss: 4.2353 - accuracy:
0.7396
Epoch 10/30
3/3 [==============================] - 15s 4s/step - loss: 4.3710 - accuracy:
0.6979
Epoch 11/30
3/3 [==============================] - 13s 4s/step - loss: 4.4179 - accuracy:
0.6562
Epoch 12/30
3/3 [==============================] - 13s 4s/step - loss: 5.3877 - accuracy:
0.5625
Epoch 13/30
3/3 [==============================] - 14s 4s/step - loss: 6.7465 - accuracy:
0.7083
Epoch 14/30
3/3 [==============================] - 13s 4s/step - loss: 5.9678 - accuracy:
0.7188
Epoch 15/30
3/3 [==============================] - 14s 4s/step - loss: 3.2693 - accuracy:
0.7083
Epoch 16/30
3/3 [==============================] - 14s 5s/step - loss: 4.0086 - accuracy:
0.6562
Epoch 17/30
3/3 [==============================] - 12s 4s/step - loss: 6.3655 - accuracy:
0.6458
Epoch 18/30
3/3 [==============================] - 14s 4s/step - loss: 4.0626 - accuracy:
0.6771
Epoch 19/30
3/3 [==============================] - 14s 4s/step - loss: 4.4661 - accuracy:
0.6146
Epoch 20/30
3/3 [==============================] - 12s 4s/step - loss: 2.9937 - accuracy:
0.7292
Epoch 21/30
```

```
3/3 [==============================] - 12s 4s/step - loss: 2.4128 - accuracy:
0.7604
Epoch 22/30
3/3 [==============================] - 13s 4s/step - loss: 2.1370 - accuracy:
0.7917
Epoch 23/30
3/3 [==============================] - 12s 4s/step - loss: 4.1273 - accuracy:
0.6667
Epoch 24/30
3/3 [==============================] - 12s 4s/step - loss: 2.4140 - accuracy:
0.7708
Epoch 25/30
3/3 [==============================] - 14s 4s/step - loss: 3.1519 - accuracy:
0.7604
Epoch 26/30
3/3 [==============================] - 12s 4s/step - loss: 2.4650 - accuracy:
0.7604
Epoch 27/30
3/3 [==============================] - 14s 4s/step - loss: 4.5149 - accuracy:
0.6042
Epoch 28/30
3/3 [==============================] - 13s 4s/step - loss: 3.3439 - accuracy:
0.6562
Epoch 29/30
3/3 [==============================] - 11s 4s/step - loss: 1.6271 - accuracy:
0.8229
Epoch 30/30
3/3 [==============================] - 13s 4s/step - loss: 3.4004 - accuracy:
0.7500
```

In [12]:

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

Application code

```
from flask import Flask, render_template, request, redirect, url_for
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 cloudant.client import Cloudant

# basepath = os.path.dirname(__file__)
# print(basepath)
print(os.getcwd())
model = load_model(r'updated_Xception.h5')
client = Cloudant.iam("ca414387-f653-4fa8-8a90-a828be636391-bluemix",
                      "OSs0X0E0p-9BoyBZcMpe2sgf8h8gAJMUmYyNKGst9LIy",
connect=True)
myDB = client.create_database('retinopathy')
app = Flask(__name__)

# pages


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


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


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


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

```
@app.route('/registerUser', methods=['post'])
def registerUser():
    x = [x for x in request.form.values()]
    print(x)
    data = {
        '_id': x[1],
        'name': x[0],
        'pass': x[2]
    }
    print(data)
    query = {'_id': {'$eq': data['_id']}}

    docs = myDB.get_query_result(query)
    print(docs)

    if (len(docs.all()) == 0):
        url = myDB.create_document(data)
        return render_template('register.html', pred='Registration successful, please login with your details')
    else:
        return render_template('register.html', pred="you are already registered. please login with your credential")


@app.route('/loginUser', methods=['POST'])  # type: ignore
def loginUser():
    print(request.form)
    user = request.form['email']
    passw = request.form['password']
    print(user, passw)
```

```
    query = {'_id': {'$eq': user}}
    docs = myDB.get_query_result(query)
    print(docs)

    if (len(docs.all()) == 0):
        return render_template('login.html', pred="username not found")
    else:
        if (user == docs[0][0]['_id'] and passw == docs[0][0]['pass']):
            return redirect(url_for('prediction'))
        else:
            print('Invalid user')


@app.route('/predictImage', methods=['POST'])
def predictImage():
    # print(request.files)
    f = request.files['image']
    basepath = os.path.dirname(__file__)
    filepath = os.path.join(basepath, 'uploads', f.filename)  # type: ignore
    print(filepath)
    f.save(filepath)

    img = image.load_img(filepath, target_size=(299, 299))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    img_data = preprocess_input(x)
    _prediction = np.argmax(model.predict(img_data), axis=1)

    index = ['No Diabetic Retinopathy', 'Mild DR',
             'Moderate DR', 'Severe DR', 'Proliferative DR']

    result = str(index[_prediction[0]])
```

```
    print(result)
    return render_template('prediction.html', prediction=result)


# main driver function
if __name__ == '__main__':
    app.run(debug=True)
```

GitHub: https://github.com/IBM-EPBL/IBM-Project-10558-1659186640

Project Demo Link: