Project Documentation

Team ID -593124

Project Title : Deep Learning Model for Eye Disease Prediction

1. INTRODUCTION

1.1 Project Overview

This project aims to advance the field of ophthalmology by leveraging the power of deep learning (DL) in artificial intelligence (AI) for the classification of various eye diseases. Eye conditions, often resulting from age-related changes or complications like diabetes, are broadly categorized into four major types: Normal, Cataract, Diabetic Retinopathy, and Glaucoma. The application of AI, particularly DL methods, has emerged as a transformative approach in healthcare, offering high-performance classification capabilities. In this context, our project utilizes these cutting-edge AI technologies to enhance the accuracy and efficiency of eye disease detection using image analysis.

A significant aspect of our approach is the adoption of transfer learning techniques, which have shown remarkable success in the realm of image analysis and classification. We have integrated established models such as Inception V3, VGG-16, and Xception, renowned for their efficacy in transfer learning within image analysis domains. These models are chosen for their proven track record in delivering enhanced performance, making them highly suitable for our objective of precise and reliable eye disease classification.

1.2 Purpose

The primary purpose of this project is to address the growing need for accurate, efficient, and accessible diagnostic methods in the detection of eye diseases. By employing sophisticated AI models, the project seeks to revolutionize the way eye diseases are identified, moving away from traditional, often slower diagnostic methods to more rapid and reliable AI-driven techniques. The expected outcome is a significant improvement in the early detection and classification of eye diseases, ultimately contributing to better patient outcomes, reduced healthcare costs, and enhanced accessibility to quality eye care, especially in underserved communities.

The integration of AI in eye disease diagnosis through this project is not just a technological advancement but also a step forward in making healthcare more inclusive and effective. By harnessing the capabilities of advanced DL models, we aim to set a new standard in eye care, paving the way for broader applications of AI in various aspects of healthcare diagnostics.

2. LITERATURE SURVEY

2.1 Existing Problem

Current methods for eye disease prediction and diagnosis, such as fundus photography and optical coherence tomography (OCT), are challenged by issues of accuracy, accessibility, and cost-effectiveness. These traditional methods often require significant time and resources, can be subjective, and are not always readily available, especially in resource-limited settings. Furthermore, the accuracy of these methods can vary, depending on the expertise of the medical professionals interpreting the results, leading to inconsistencies in diagnosis.

2.2 References

In the context of AI and eye disease diagnosis, several studies and papers have contributed to the understanding and advancement of this field. For instance, research on the application of deep learning models like Convolutional Neural Networks (CNNs) for retinal image analysis has shown promising results in improving diagnostic accuracy. Papers such as "Deep learning for detecting retinal detachment and discerning macular status using ultra-widefield fundus images" (Annals of Translational Medicine, 2020) and "Automated detection of diabetic retinopathy using deep learning" (American Journal of Ophthalmology, 2018) have been instrumental in demonstrating the potential of AI in this domain.

2.3 Problem Statement Definition

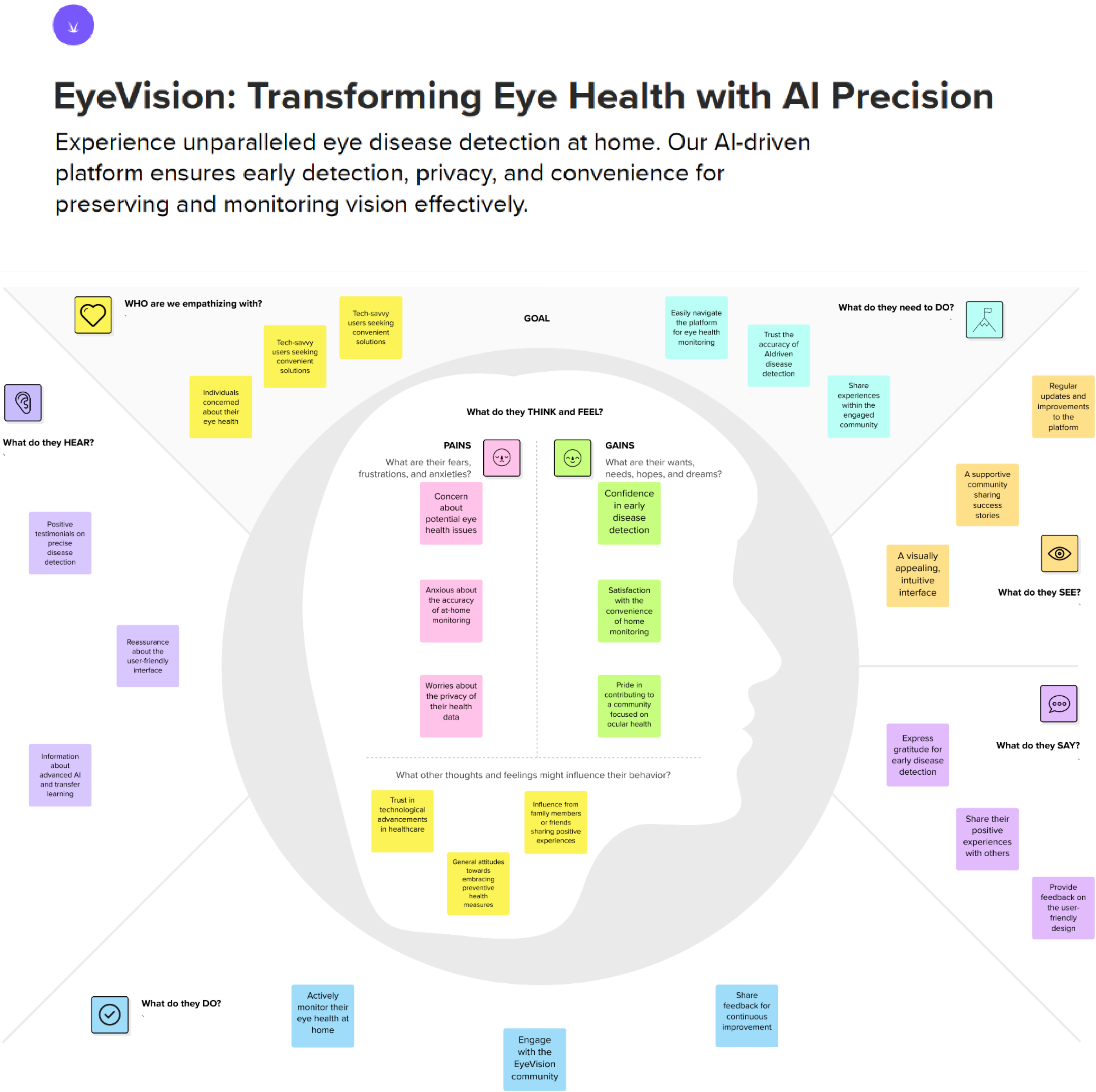
The core problem addressed by this project is the need for an improved method of diagnosing eye diseases, one that transcends the limitations of traditional diagnostic techniques. The current landscape of eye disease diagnosis, primarily reliant on methods like fundus photography and optical coherence tomography (OCT), faces significant challenges:

1. **Accuracy and Subjectivity**: Traditional methods often depend heavily on the expertise and subjective interpretation of medical professionals. This reliance can lead to variability in diagnoses, potentially affecting the accuracy and reliability of the outcomes.
2. **Accessibility and Availability**: These conventional diagnostic tools are not uniformly accessible, particularly in under-resourced or remote areas. The lack of accessibility not only limits the reach of these diagnostic methods but also exacerbates health disparities.
3. **Cost and Time Efficiency**: Traditional diagnostic processes can be time-consuming and expensive, both in terms of equipment and the need for specialized personnel. This aspect makes regular screening and early detection of eye diseases challenging, especially for populations with limited healthcare budgets.
4. **Scalability and Adaptability**: The growing prevalence of eye diseases globally demands diagnostic methods that can scale effectively and adapt to diverse patient populations. Current methods may not always meet these demands due to their inherent limitations.

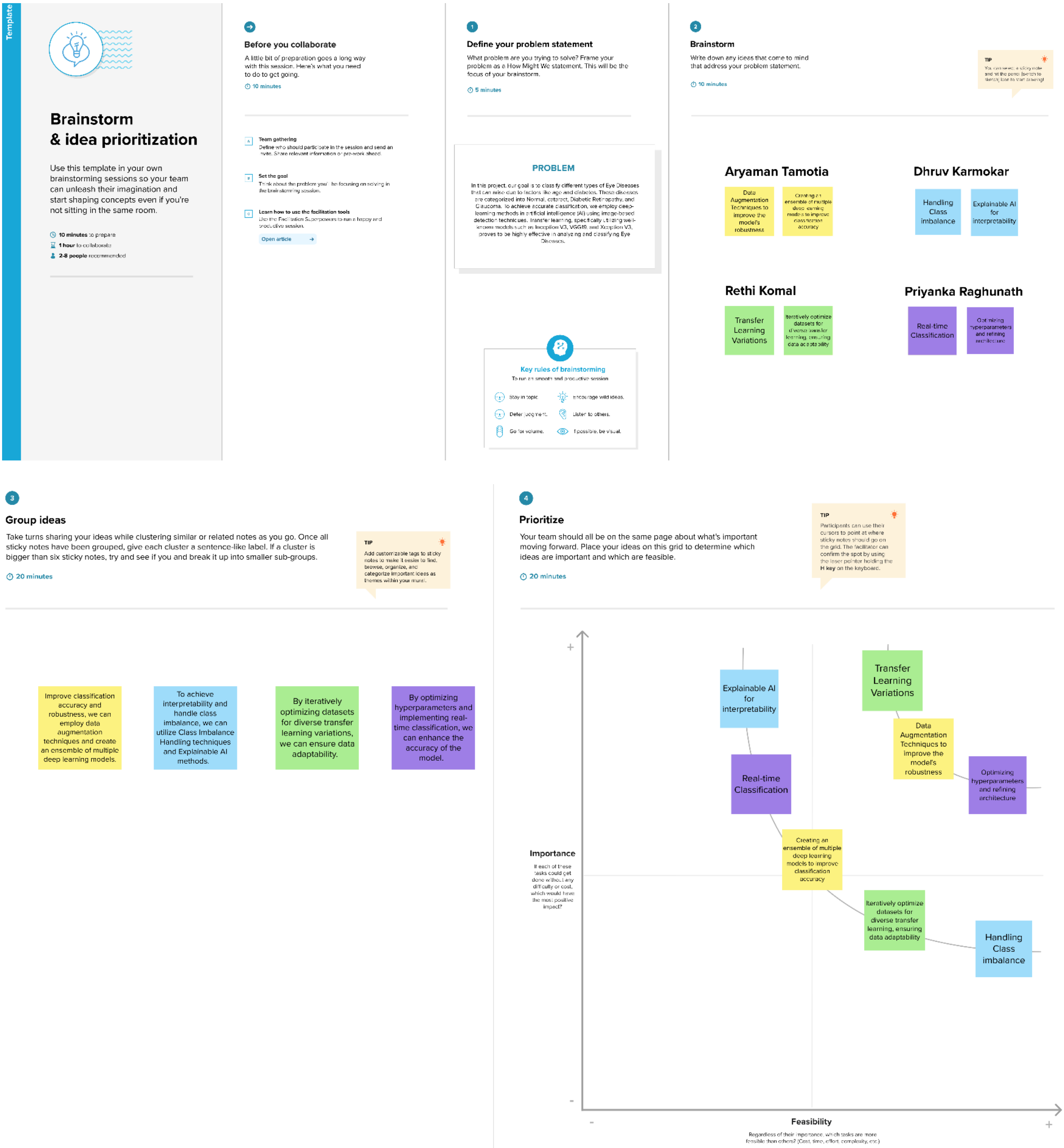
By leveraging artificial intelligence, specifically deep learning models such as Inception V3, VGG-16, and Xception, this project aims to develop a solution that addresses these challenges. The goal is to create a diagnostic tool that is not only more accurate and less subjective but also more accessible, cost-effective, and scalable.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



4. REQUIREMENT ANALYSIS

Table-1 : Components & Technologies:

Component	Description	Technology
User Interface	Web interface for image input and prediction.	HTML, CSS, JavaScript
Application Logic	The logic that integrates the prediction results from the models and formats them for presentation on the UI.	Flask (Python Web Framework)
Cloud Database	Database service on cloud	Flask
File Storage	Storage for Image Uploads	Local Filesystem
Machine Learning Model	Deep learning algorithms, namely Inception V3, VGG16, and Xception V3, used for analyzing retina images and predicting eye diseases.	Inception V3, VGG16, Xception V3 (Deep Learning Algorithms)
Infrastructure (Server / Cloud)	The application is deployed on Heroku, which provides managed services, including server orchestration, deployment, and scaling.	Heroku (Cloud Application Platform)

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Utilized Flask, a lightweight and flexible Python web framework, to develop the web application interface for the eye disease prediction system. This framework enabled the creation of a user-friendly web interface for uploading retina images and displaying diagnostic results.	Python (Python Web Framework)
2.	Security Implementations	While specific advanced security measures have not been implemented, basic security practices inherent to Flask and web application development have been adhered to. This includes secure handling of user data and basic protection against common web vulnerabilities.	Basic Web Security Practices (Inherent in Flask)
3.	Scalable Architecture	Deployed on Heroku, a cloud platform as a service, to ensure scalability and ease of deployment. Heroku allows the application to handle varying loads with its dynamic scaling capabilities and simplifies the deployment process.	Heroku (Cloud Platform as a Service)
4.	Availability	No specific technology or methodology has been implemented solely for ensuring high availability. However, the choice of Heroku as a deployment platform indirectly contributes to availability through its reliable infrastructure and managed services.	Indirect Availability via Heroku Platform
5.	Performance	Implemented three cutting-edge deep learning algorithms (Inception V3, VGG16, Xception V3) to boost the accuracy and efficiency of eye disease prediction. These algorithms, renowned for high performance in image analysis, enhance diagnostic precision and speed.	Inception V3, VGG16, Xception V3 (Deep Learning Algorithms)

Proposed Solution Template:

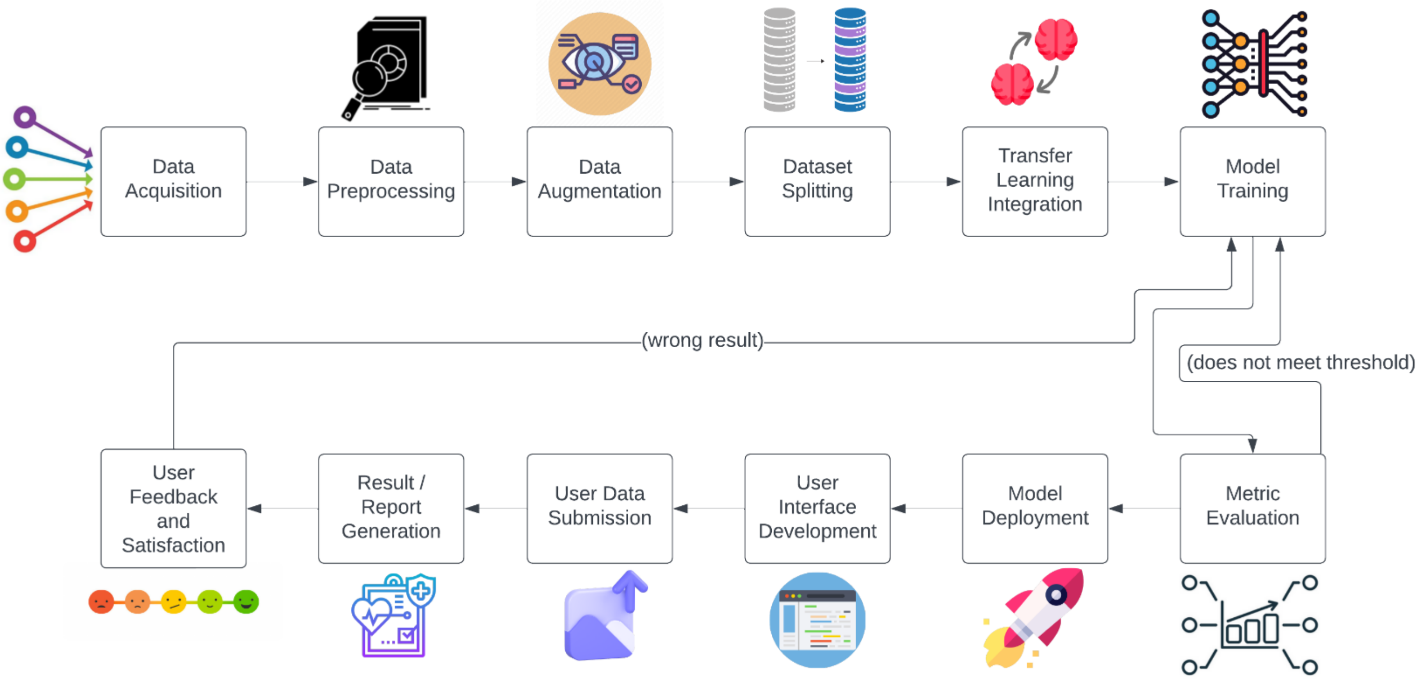
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In this project, we are focused on classifying various types of eye diseases, which can be caused by factors such as age and diabetes. These diseases are primarily categorized into four groups: Normal, cataract, Diabetic Retinopathy, and Glaucoma. Deep learning methods within artificial intelligence have emerged as powerful tools for accurately identifying these eye diseases using image data. Transfer learning has become a prevalent technique, proving its effectiveness in various domains, particularly in image analysis and classification tasks. To enhance our classification performance, we have employed popular transfer learning models such as Inception V3, VGG16, and Xception V3, known for their robust performance in image analysis.
2.	Idea / Solution description	The project's core idea is to employ deep learning and transfer learning techniques to create an AI-based system for the classification of eye diseases, specifically Normal, cataract, Diabetic Retinopathy, and Glaucoma, using image data. This solution involves data collection, preprocessing, and the utilization of pre-trained models like Inception V3, VGG16, and Xception V3 for accurate disease classification. It will be deployed with a user-friendly interface that ensures ethical data handling and privacy while also offering educational resources for users to better understand eye diseases and their potential treatments.

3.	Novelty / Uniqueness	The project's uniqueness lies in its innovative application of deep learning to classify a wide spectrum of eye diseases, including but not limited to Normal, cataract, Diabetic Retinopathy, and Glaucoma. This multifaceted classification approach enhances the precision of disease identification, thus allowing for more targeted and effective medical interventions. By leveraging well-established pre-trained models such as Inception V3, VGG16, and Xception V3 for transfer learning, the project underscores its dedication to harnessing cutting-edge technology for improved healthcare outcomes. In addition, the project places a strong emphasis on ethical considerations and privacy measures, ensuring responsible data management and safeguarding patient confidentiality. Furthermore, the provision of user-friendly interfaces and educational resources sets this project apart by not only aiding in disease diagnosis but also by fostering healthcare education, culminating in a comprehensive and distinctive solution within the realm of medical image analysis.
4.	Social Impact / Customer Satisfaction	The AI-based system designed for classifying eye diseases holds significant promise for creating a positive societal impact. It achieves this by enabling early disease detection, which can result in timely medical interventions and better patient outcomes. The system's user-friendly interface promotes greater access to healthcare services, particularly in underserved regions, expanding its reach. Furthermore, it provides valuable support to healthcare professionals in diagnosing and planning treatment for eye diseases, potentially reducing their workloads and improving healthcare delivery. Moreover, the inclusion of educational resources within the system not only fosters awareness and health literacy but also empowers users to make informed decisions about their eye health. This comprehensive approach enhances the overall patient experience and reinforces the system's value in healthcare. Ethical data handling and privacy considerations remain pivotal in cultivating trust among users and healthcare providers, ultimately leading to a high level of customer satisfaction and positive healthcare outcomes.

5.	Business Model (Revenue Model)	The business model for the AI-powered system designed to classify eye diseases offers several avenues for revenue generation. Firstly, it can introduce a subscription-based model, targeting medical professionals, clinics, and healthcare institutions, with pricing structures that vary based on usage and included features, ensuring a steady income stream. Alternatively, a pay-per-use model may be implemented, allowing individual users to pay for each instance of disease classification as required, providing adaptability and cost-effectiveness. Another option is to license the technology to healthcare providers, allowing them to integrate the system into their existing healthcare solutions, like electronic health records, for a licensing fee. Data monetization represents a potential approach through partnerships with research institutions and healthcare analytics companies, granting them access to de-identified patient data for research purposes. Lastly, collaboration with telemedicine platforms can involve providing the classification service as an additional feature, potentially resulting in revenue through revenue-sharing agreements or flat fees, ensuring a diversified income portfolio for the business.
6.	Scalability of the Solution	The importance of scalability in the proposed eye disease classification solution, which relies on AI and deep learning, cannot be overstated. It entails efficiently managing expanding volumes of eye image data through adaptable data storage and processing methods. The deep learning model's ability to adjust to larger and more intricate datasets is paramount, with the use of scalable model architectures and advanced hardware for sustained performance. Maintaining cost-effectiveness, ensuring worldwide accessibility, upholding stringent security and privacy measures, streamlining user management, and establishing feedback mechanisms are all essential elements in the plan to ensure the solution can grow and meet the escalating demand for enhanced eye disease diagnosis and treatment.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

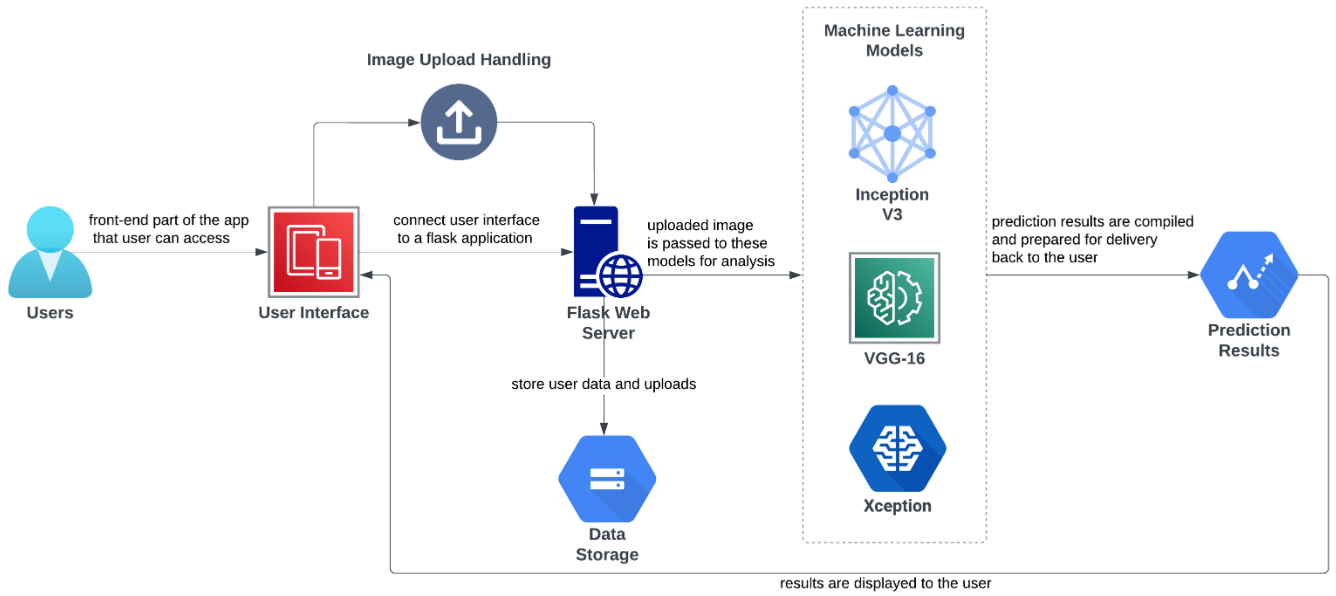


User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Patient	Dashboard	USN-1	As a patient, I want to access a dashboard where I can view my eye health information and options to upload eye data.	The dashboard effectively presents options for uploading data and viewing past reports.	High	Sprint-1
	Uploading Eye Data	USN-2	As a patient, I need to upload my eye data so that I can get a diagnosis for my condition.	The system confirms successful uploads and ensures data privacy.	High	Sprint-1
	Patient Report	USN-3	As a patient, I want to view my diagnostic report to understand the health of my eyes.	The report details diagnostic results in an understandable format.	Medium	Sprint-2
	Recommendation	USN-4	As a patient, I want to receive recommendations based on my diagnosis to manage my eye health.	The system provides actionable health recommendations post-diagnosis.	Medium	Sprint-3
	Queries & Feedback	USN-5	As a patient, I want to be able to submit queries and feedback about the diagnosis and app usability.	The system allows easy submission of feedback and provides acknowledgment of receipt.	Low	Sprint-4
Doctor	Dashboard	USN-6	As a doctor, I want to access a dashboard to review patient eye data and provide my expert diagnosis.	The dashboard consolidates patient data and allows for efficient review and annotation.	High	Sprint-1
	Reviewing Data	USN-7	As a doctor, I need to review patient uploads to determine the health of their eyes and potential diseases.	The system allows me to easily navigate through patient data and images.	High	Sprint-2

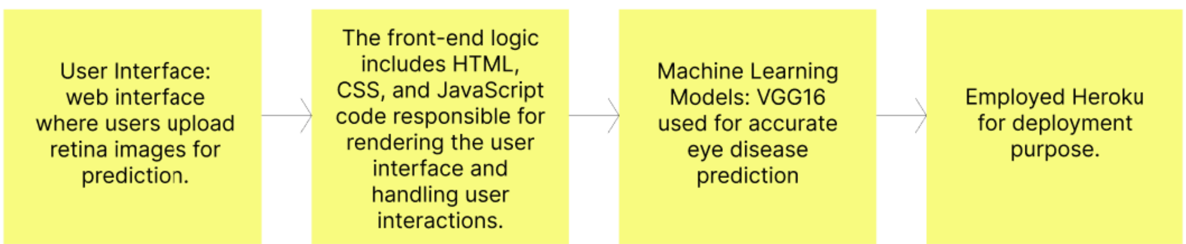
	Generating Reports	USN-8	As a doctor, I want to generate reports that give a clear diagnosis and recommendations for my patients.	The system enables the creation of comprehensive reports with diagnostic results and health recommendations.	High	Sprint-1
	Patient Interaction	USN-9	As a doctor, I need to interact with patients based on their diagnostic reports to provide further medical advice.	The system facilitates secure communication channels between me and the patients.	Medium	Sprint-2
Developer	System Updates	USN-10	As a developer, I want to implement system updates to enhance application performance and security.	The system supports automated updates without downtime and notifies developers of successful integration.	High	Sprint-1
	Monitoring & Maintenance	USN-11	As a developer, I need to monitor system performance and conduct regular maintenance for optimal operation.	The system includes tools for real-time performance tracking and alerts for maintenance requirements.	High	Sprint-2

5.2 Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

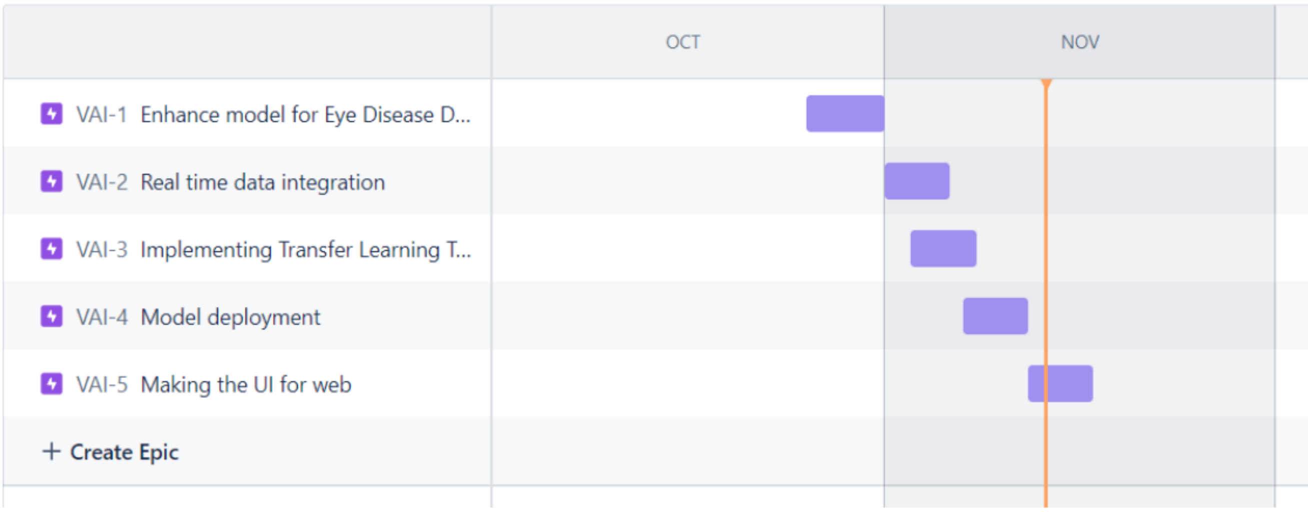
Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	VGG16	USN-1	Enhance model for Eye Disease Detection	7	High	2
Sprint-1	VGG16	USN-2	Real time data integration	4	Medium	1
Sprint-2	Metric Evaluation	USN-3	Implementing Transfer Learning Techniques	9	High	3
Sprint-3	Using different transfer learning techniques for report generation	USN-4	Model deployment	10	High	3
Sprint-4	HTML, CSS, javascript	USN-5	Making the UI for web	6	Medium	2

6.3 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)
Sprint 1	11	7 Days	26 Oct 2023	01 Nov 2023
Sprint 2	9	5 Days	02 Nov 2023	06 Nov 2023
Sprint 3	10	6 Days	07 Nov 2023	12 Nov 2023
Sprint 4	6	3 Days	13 Nov 2023	15 Nov 2023



7. CODING & SOLUTIONING

- **Phase one – Setting up the environment**
  In this phase we created a separate anaconda environment for the project, in-order to avoid any clashes in requirements, then we installed the TensorFlow, Flask, Pillow and other libraries using 'pip install' command.

- **Phase two – Data collection**
  Once the environment was ready, our first task was to collect the historical data

- **Phase three – Data pre-processing and data visualization**
  In this phase we visualized the time series data and cleaned the data and made it ready for the algorithm

- **Phase four – Model building**
  Once the data was ready it was time to feed the data to our VGG16 model to train it, this is what was done in this phase

- **Phase five – Deployment**
  After the model was ready, we finally deployed the model using HTML, CSS and Flask

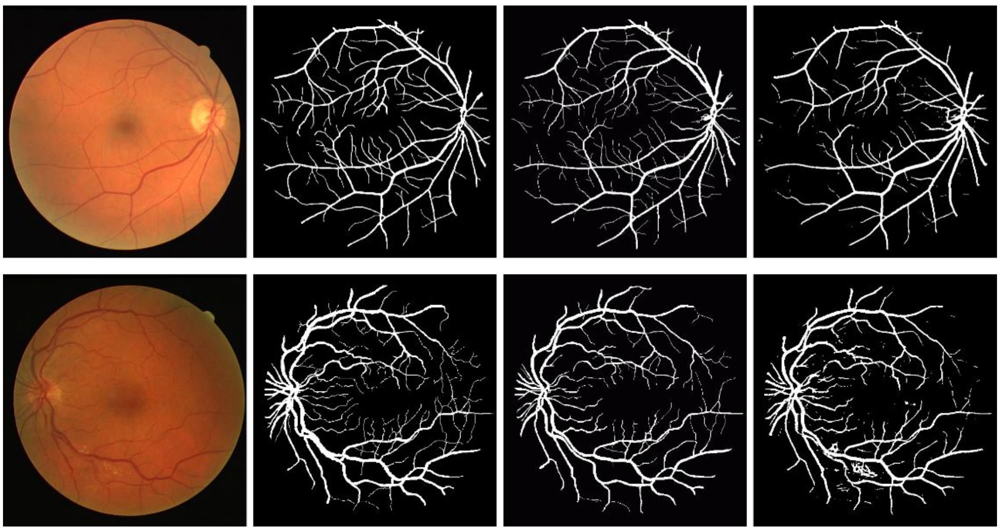
<u>Let's now look at these phases one by one in detail</u>

Setting up the environment:

- This phase was simple and straightforward, we started by downloading anaconda navigator, then installed the Jupyter notebook.
- Once this was done we began installing required libraries- TensorFlow, Flask, Pillow and other libraries using 'pip install' command.

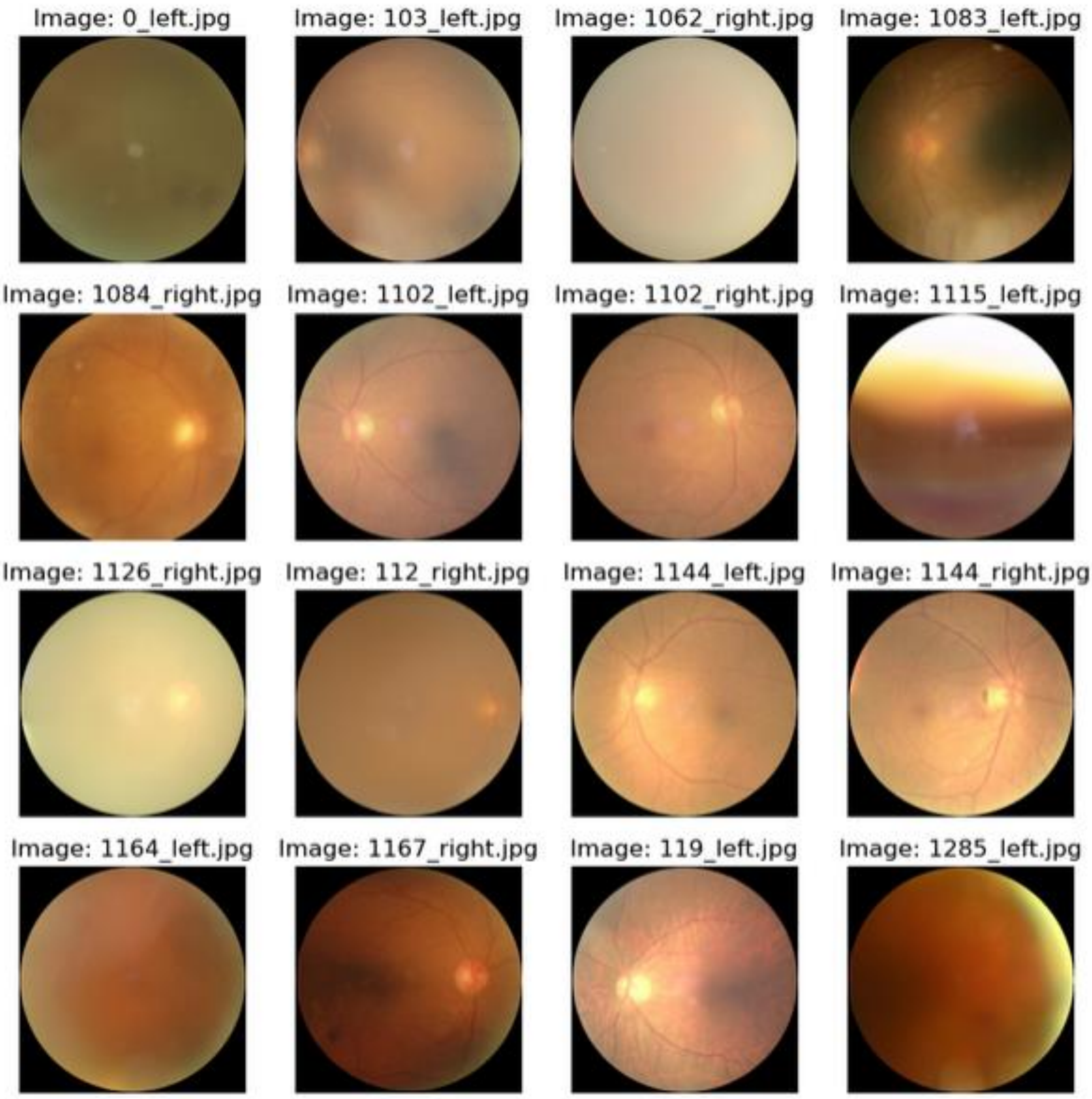
Data collection:

- The data collected in this project comes directly from Kaggle link provided to us https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification
- The dataset consists of Normal, Diabetic Retinopathy, Cataract and Glaucoma retinal images where each class have approximately 1000 images. These images are collected from various sources like IDRiD, Oculur recognition, HRF etc.



Dataset

```
In [12]: folder_path = "dataset/cataract"
image_filenames = os.listdir(folder_path)
image_filenames = image_filenames[:16]
fig, axes = plt.subplots(4, 4, figsize=(8, 8))

for i, filename in enumerate(image_filenames):
    image_path = os.path.join(folder_path, filename)
    img = Image.open(image_path)
    axes[i // 4, i % 4].imshow(img)
    axes[i // 4, i % 4].set_title(f"Image: {filename}")
    axes[i // 4, i % 4].axis("off")

plt.tight_layout()
plt.show()
```



Data Augmentation:

Now the pre-processed data is augmented to enrich the dataset and improve model robustness.

Data augumentation

In [34]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split=0.2
)
```

In [35]:
```
train_set = train_datagen.flow_from_directory(
    data_path,
    target_size=IMG_SIZE[:2],
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training',
    shuffle=True,
)
```

```
Found 3376 images belonging to 4 classes.
```

In [36]:
```
test_set = train_datagen.flow_from_directory(
    data_path,
    target_size=IMG_SIZE[:2],
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation',
    shuffle=False,
)
```

```
Found 841 images belonging to 4 classes.
```

Using Transfer Learning:

Now we use pre-trained models such as Inception V3, VGG-16, and Xception for feature extraction and model enhancement.

Transfer Learning in VGG16

In [37]:
```
vgg16 = VGG16(input_shape=IMG_SIZE, weights='imagenet', include_top=False)
```

In [38]:
```
for layer in vgg16.layers:
    layer.trainable = False
```

In [39]:
```
folders = glob('dataset/*')
```

In [40]:
```
# our layers
x = Flatten()(vgg16.output)
```

In [41]:
```
prediction = Dense(len(folders), activation='softmax')(x)

# create a model object
model = Model(inputs=vgg16.input, outputs=prediction)
```

In [42]:
```
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten_1 (Flatten)         (None, 25088)             0

 dense_1 (Dense)             (None, 4)                 100356

=================================================================
Total params: 14815044 (56.51 MB)
Trainable params: 100356 (392.02 KB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

Implementing Callbacks:

Now we are integrating callbacks to handle real-time events within the application, such as triggering the analysis of retina images once uploaded and subsequently displaying diagnostic results to the user interface automatically.

Callbacks

In [43]:
```
BUR_callback = BackupAndRestore(backup_dir="./temp1/backup", save_freq='epoch',
                                delete_checkpoint=True,)
```

In [44]:
```
early_callback = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True,
                               start_from_epoch=2)
```

In [45]:
```
csvlog_callback = CSVLogger(
    'logger.csv', separator=',', append=False
)
```

In [46]:
```
import tensorflow as tf

decay_steps = 3  # Number of epochs before reducing the learning rate
decay_rate = tf.math.exp(-0.1)  # The factor by which the learning rate will be reduced

def lr_scheduler(epoch, lr):
    if epoch % decay_steps == 0 and epoch > 0:
        return lr * decay_rate
    return lr

lr_callback = LearningRateScheduler(lr_scheduler, verbose=1)
```

In [47]:
```
checkpoint_filepath = '/temp1/checkpoint'
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

Model Compilation and Fitting:

Now we are compiling and fitting the model, which involves setting up the loss function, optimizer, and metrics for training, and then training the model on the pre-processed data to learn patterns relevant for eye disease prediction.

Compilation and Fitting

In [50]:
```
model.compile(
  loss='categorical_crossentropy',
  optimizer=Adam(learning_rate=0.0005),
  metrics=['accuracy'],
)
```

In [51]:
```
# fit the model

r = model.fit(
  train_set,
  validation_data=test_set,
  epochs=30,
  steps_per_epoch=len(train_set),
  validation_steps=len(test_set),
  callbacks=[BUR_callback, csvlog_callback, lr_callback]
)
```

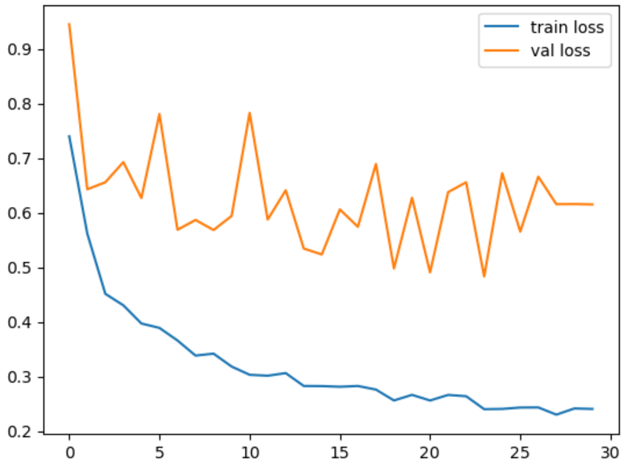
```
Epoch 1: LearningRateScheduler setting learning rate to 0.0005000000237487257.
Epoch 1/30
106/106 [==============================] - 278s 3s/step - loss: 0.7398 - accuracy: 0.6943 - val_loss: 0.9448 - val_accurac
y: 0.6409 - lr: 5.0000e-04

Epoch 2: LearningRateScheduler setting learning rate to 0.0005000000237487257.
Epoch 2/30
106/106 [==============================] - 262s 2s/step - loss: 0.5617 - accuracy: 0.7793 - val_loss: 0.6428 - val_accurac
y: 0.7503 - lr: 5.0000e-04

Epoch 3: LearningRateScheduler setting learning rate to 0.0005000000237487257.
Epoch 3/30
106/106 [==============================] - 259s 2s/step - loss: 0.4514 - accuracy: 0.8303 - val_loss: 0.6556 - val_accurac
y: 0.7372 - lr: 5.0000e-04

Epoch 4: LearningRateScheduler setting learning rate to 0.0004524187243077904.
Epoch 4/30
106/106 [==============================] - 267s 3s/step - loss: 0.4303 - accuracy: 0.8377 - val_loss: 0.6925 - val_accurac
y: 0.7265 - lr: 4.5242e-04
```

Data Visualization:

Now we are plotting the loss and accuracy for training and validation metrics, visualizing the model's performance over epochs to assess and fine-tune its predictive accuracy and generalization ability.

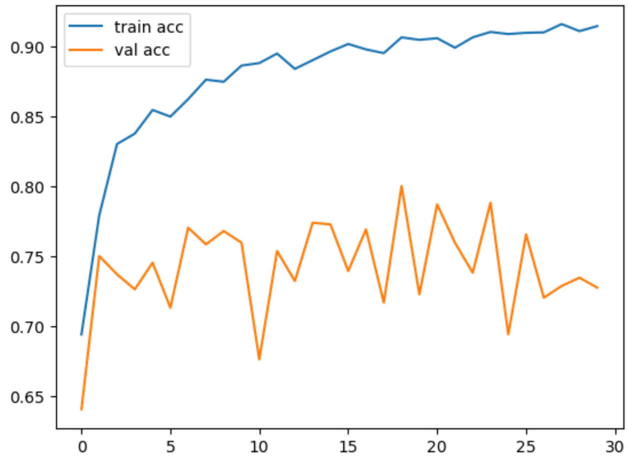
Data Visualization

In [52]:
```
# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
```



<Figure size 640x480 with 0 Axes>

In [53]:
```
# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```



<Figure size 640x480 with 0 Axes>

Creating h5 File:

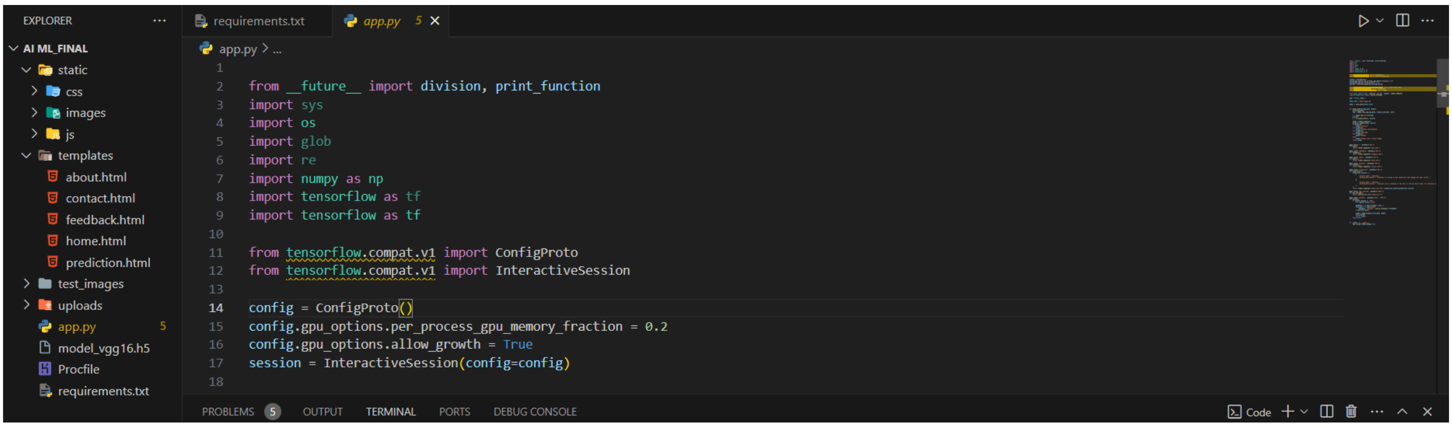
Now we are creating an .h5 file, which serves to save the trained model, encapsulating its architecture, weights, and training configuration, ensuring it can be easily reloaded and deployed for future predictions.

Creating h5 file

In [54]:
```
from tensorflow.keras.models import load_model
model.save('model_vgg16.h5')
```

Model Deployment:

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.

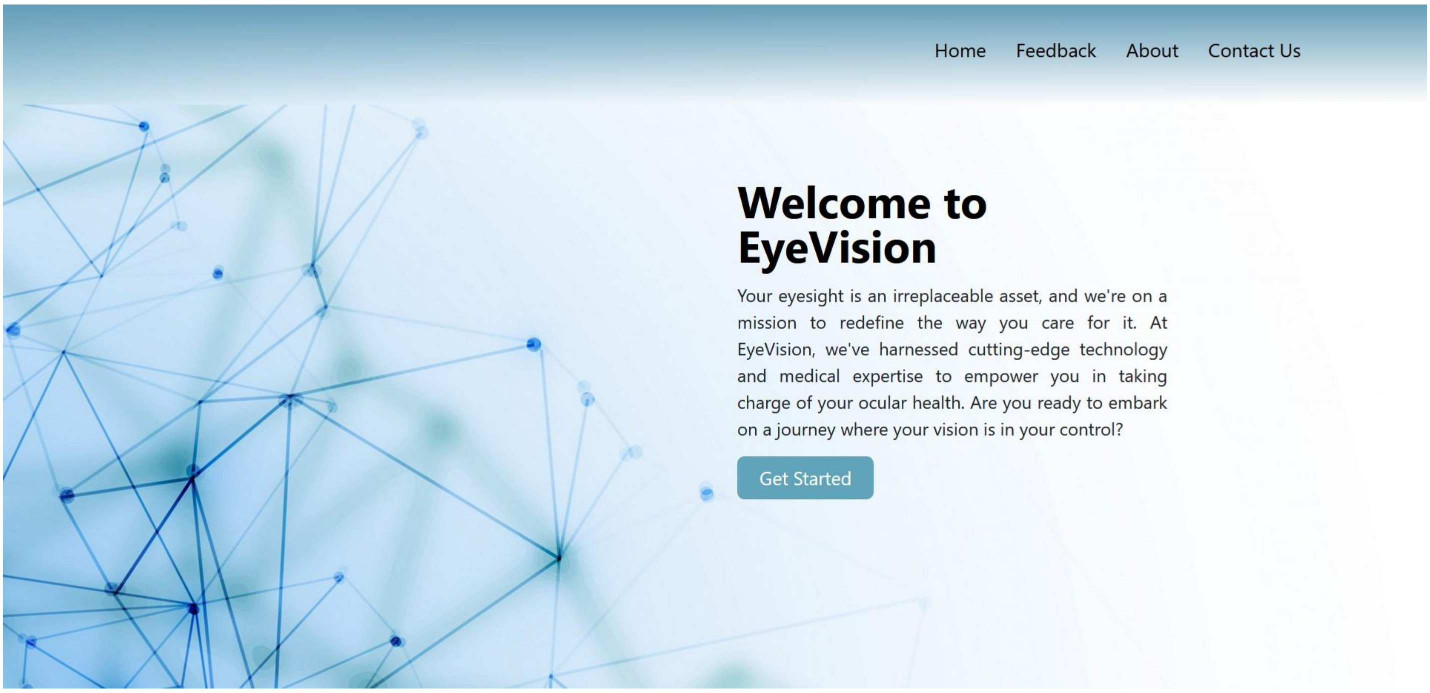


In the **flask application**, the input parameters are taken from the HTML page. These factors are then given to the model to predict the type of eye disease and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the:
"Get Started → Upload Image" button, the next page is opened where the user chooses the image and predicts the output.

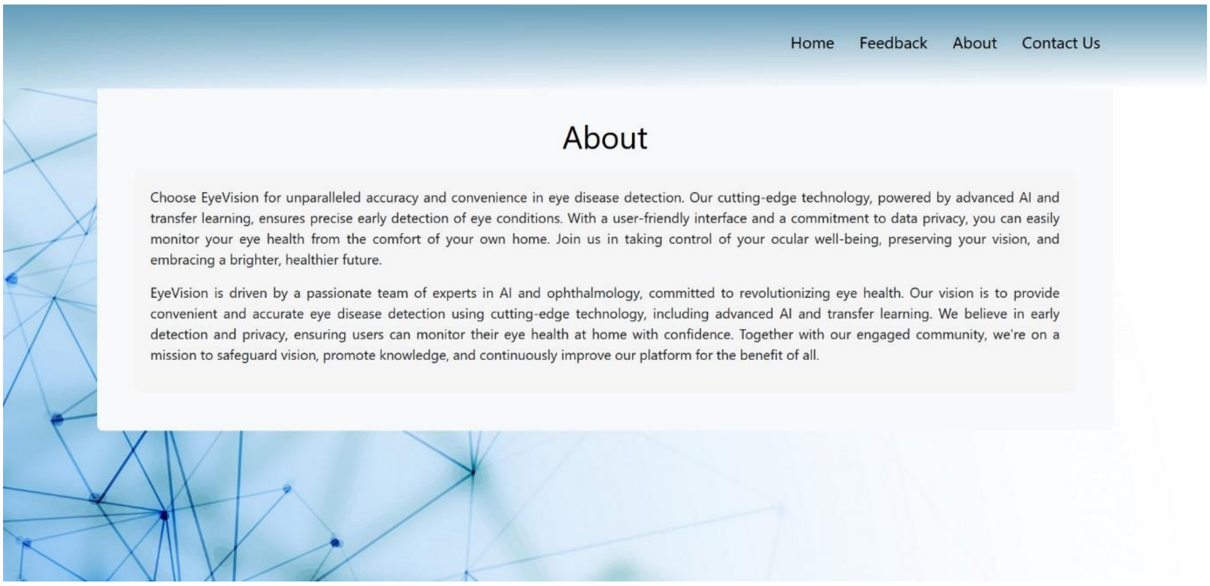
I] Create HTML Pages

- We use HTML to create the front-end part of the web page.
- Here, we have created 4 HTML pages - home.html, about.html, contact.html, and prediction.html
- home.html displays the home page
- about.html displays an introduction about the project
- prediction.html gives the user the interface to check for eye diseases asdasfor the images uploaded by the user
- contact.html gives the user a platform to ask us their queries and share their valuable feedback.
- We also use JavaScript and CSS to enhance our functionality and view of HTML page

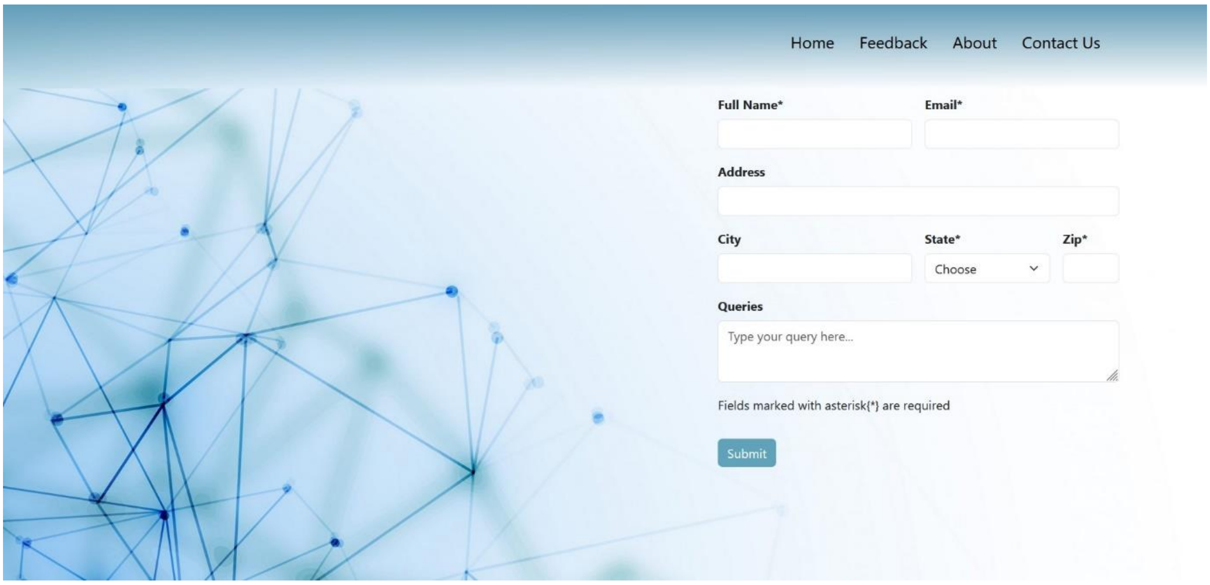
A} Home: -



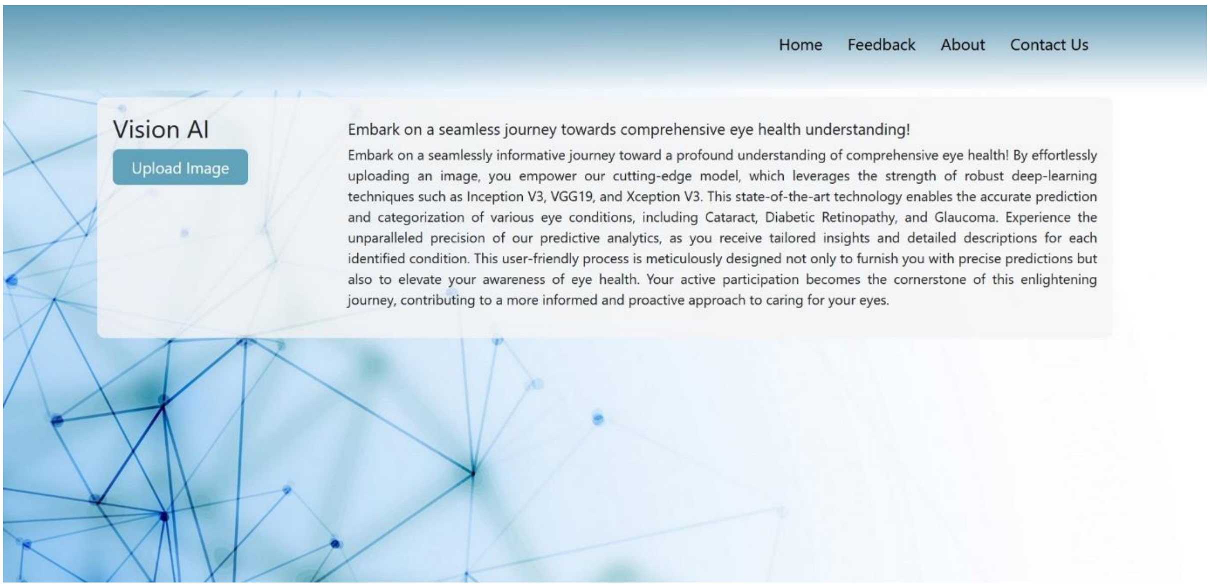
B} About Us: -



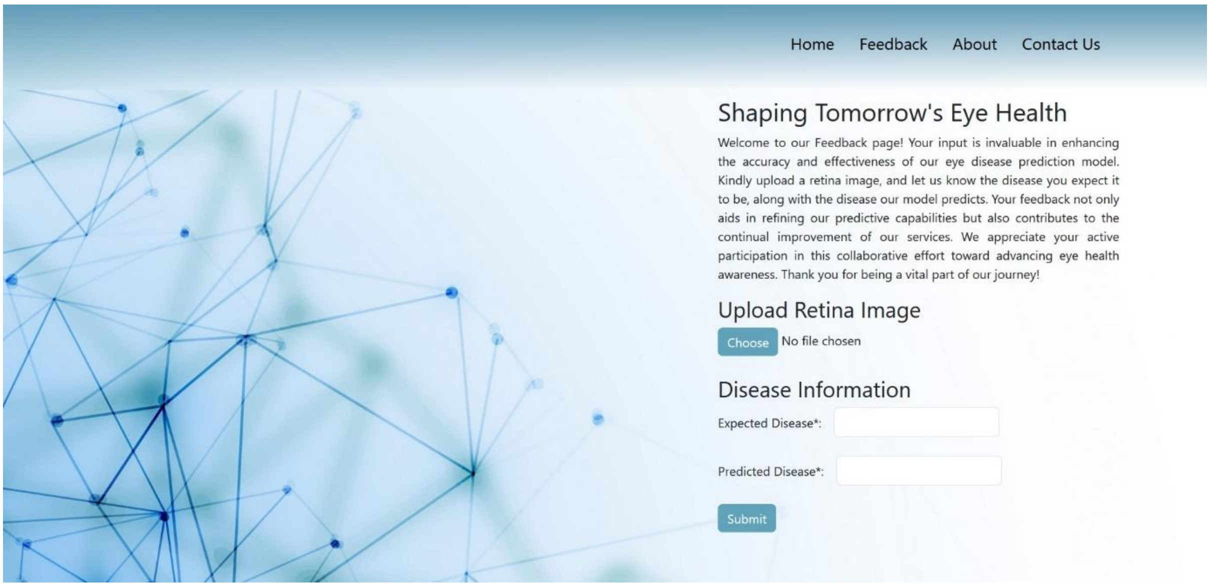
C} Contact Us: -



D} Prediction: -



E} Feedback: -



II] Build python code

A} Importing Libraries: -

The first step involves importing the necessary libraries for the program.

```
from __future__ import division, print_function
import sys
import os
import glob
import re
import numpy as np
import tensorflow as tf
import tensorflow as tf

from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
```

B} Creating our Flask Application and Loading our Model: -

Initializing the Flask app and loading the pre-trained model using the load_model method.

```
app = Flask(__name__)

MODEL_PATH ='model_vgg16.h5'

model = load_model(MODEL_PATH)
```

C} Routing to the HTML Page: -
Setting up routes to different HTML pages using Flask.

```
@app.route('/', methods=['GET'])
def home():
    return render_template('home.html')

@app.route('/feedback', methods=['GET'])
def feedback():
    return render_template('feedback.html')

@app.route('/about', methods=['GET'])
def about():
    return render_template('about.html')

@app.route('/contact', methods=['GET'])
def contact():
    return render_template('contact.html')

@app.route('/prediction', methods=['GET'])
def prediction():
    prediction_results = [
        {
            'disease_name': 'Glaucoma',
            'disease_description': 'Glaucoma is a group of eye conditions that damage the optic nerve...'
        },
        {
            'disease_name': 'Cataract',
            'disease_description': 'Cataracts are a clouding of the lens in the eye which leads to a decrease in vision...'
        },
    ]
    return render_template('prediction.html',prediction_results=prediction_results)

@app.route('/get_started', methods=['POST'])
def get_started():
    return redirect(url_for('prediction'))
```

D} Showcasing Prediction on UI: -
Processing the uploaded image and displaying the prediction result.

```
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files['file']

        basepath = os.path.dirname(__file__)
```

```
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        preds = model_predict(file_path, model)
        result=preds
        return result
    return None
```

E} Predicting the Results: -
The image file uploaded via the UI is processed and analysed using the deep learning model.

F} Finally, Run the Application: -
Running the Flask application on a local server.

```
if __name__ == '__main__':

    app.run(port=5001,debug=True)
```

III] Run the application

- Navigate to the folder where your app.py resides.
- Now type “python app.py” command.
- It will show the local host where your app is running on http://127.0.0.1.5001/
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.

```
PS C:\Users\dhruv\Downloads\AI ML_Final> python -u "c:\Users\dhruv\Downloads\AI ML_Final\app.py"
2023-11-21 23:56:09.224730: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in perf
ormance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
 * Restarting with stat
2023-11-21 23:56:16.697206: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in perf
ormance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Debugger is active!
 * Debugger PIN: 630-662-987
```

Navigate to the localhost (**http://127.0.0.1:5001/**)where you can view your web page.

8. PERFORMANCE TESTING

8.1 Performance Metrics

Model Summary:

Total params: 14815044 (56.51 MB)

Trainable params: 100356 (392.02 KB)

Non-trainable params: 14714688 (56 MB)

```
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten_1 (Flatten)         (None, 25088)             0

 dense_1 (Dense)             (None, 4)                 100356

=================================================================
Total params: 14815044 (56.51 MB)
Trainable params: 100356 (392.02 KB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

Accuracy:

Training Accuracy – 91.59%
Validation Accuracy – 80.02%

In [51]:

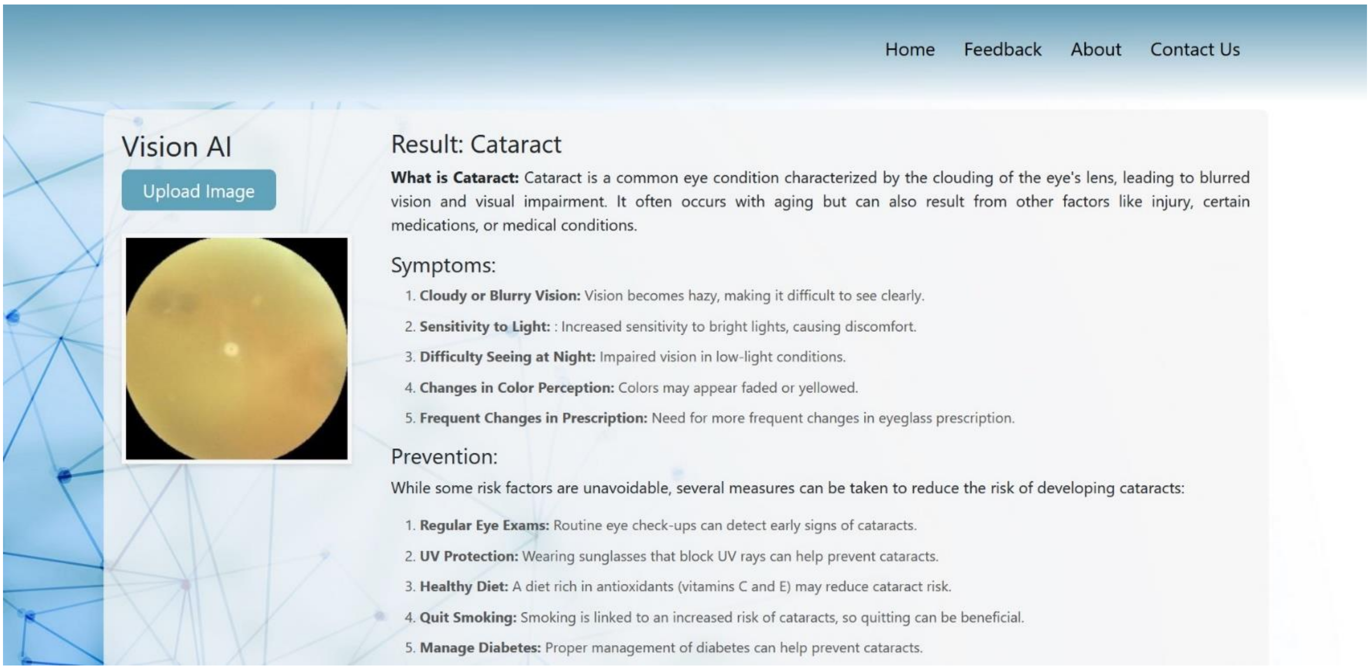
```
# fit the model

r = model.fit(
  train_set,
  validation_data=test_set,
  epochs=30,
  steps_per_epoch=len(train_set),
  validation_steps=len(test_set),
  callbacks=[BUR_callback, csvlog_callback, lr_callback]
)
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.0005000000237487257.
Epoch 1/30
106/106 [==============================] - 278s 3s/step - loss: 0.7398 - accuracy: 0.6943 - val_loss: 0.9448 - val_accurac
y: 0.6409 - lr: 5.0000e-04

Epoch 2: LearningRateScheduler setting learning rate to 0.0005000000237487257.
Epoch 2/30
106/106 [==============================] - 262s 2s/step - loss: 0.5617 - accuracy: 0.7793 - val_loss: 0.6428 - val_accurac
y: 0.7503 - lr: 5.0000e-04

Epoch 3: LearningRateScheduler setting learning rate to 0.0005000000237487257.
Epoch 3/30
106/106 [==============================] - 259s 2s/step - loss: 0.4514 - accuracy: 0.8303 - val_loss: 0.6556 - val_accurac
y: 0.7372 - lr: 5.0000e-04

Epoch 4: LearningRateScheduler setting learning rate to 0.0004524187243077904.
Epoch 4/30
106/106 [==============================] - 267s 3s/step - loss: 0.4303 - accuracy: 0.8377 - val_loss: 0.6925 - val_accurac
y: 0.7265 - lr: 4.5242e-04
```

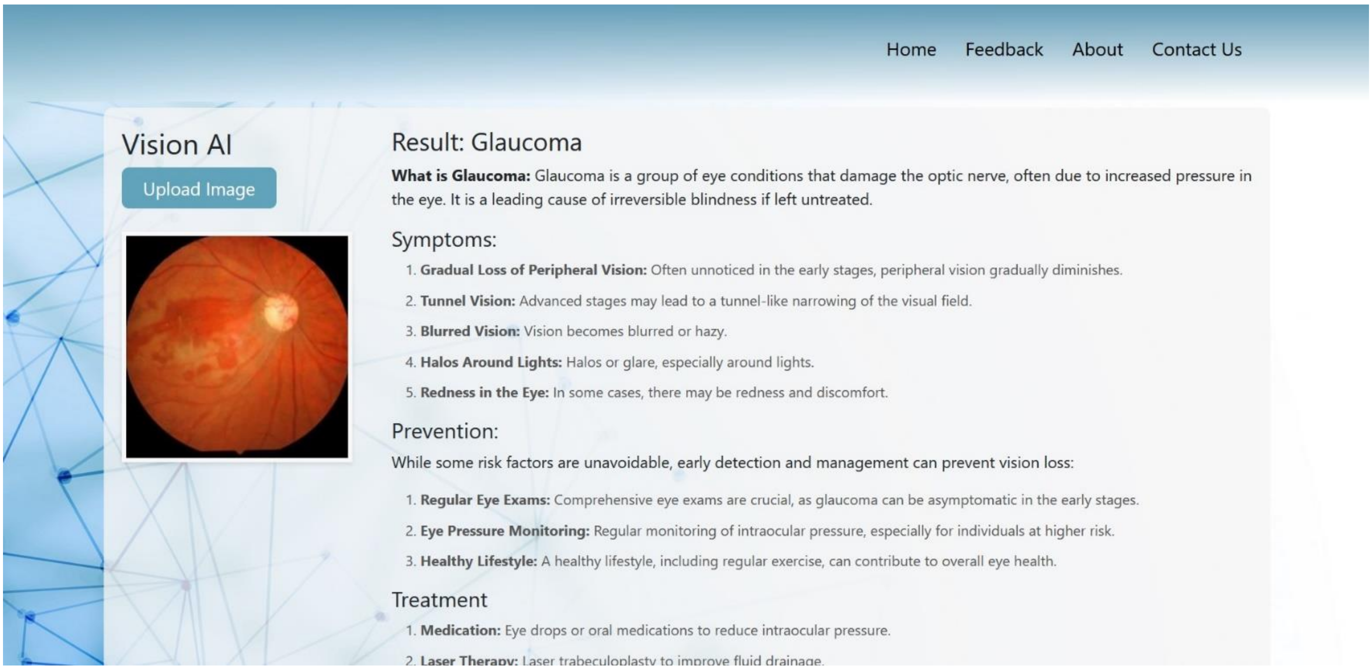
9. RESULTS

9.1 Output Screenshots

I] Test Case 1



II] Test Case 2



10. ADVANTAGES & DISADVANTAGES

Advantages

1. **High Accuracy**: The AI-ML model boasts an 80% accuracy rate in diagnosing eye diseases, marking a significant improvement in diagnostic reliability.
2. **Accessibility and Open Access**: Deployed on Heroku, the model is available widely, ensuring free and easy access to advanced diagnostic tools, especially beneficial for under-resourced areas.
3. **Open-Source Collaboration**: Being open-source, the project encourages community engagement, allowing for ongoing improvements and updates, fostering a collaborative approach to healthcare innovation.
4. **Scalability**: The model's digital nature allows for easy scalability, potentially serving many patients without the need for extensive additional resources.
5. **Reduced Diagnosis Time**: Compared to traditional methods, the AI model can potentially reduce the time taken for diagnosis, facilitating quicker decision-making in clinical settings.

Disadvantages

1. **Occasional Inaccuracies**: The model, while generally accurate, might produce incorrect outputs in some cases, highlighting the need for human oversight in diagnoses.
2. **Data Quality and Diversity**: The effectiveness of the model is highly dependent on the quality and diversity of the training data, with limitations in these areas potentially leading to reduced accuracy.
3. **Computational Resources**: The need for significant computational resources for deployment and operation could be a limiting factor, particularly in less developed healthcare infrastructures.
4. **Ethical and Privacy Concerns**: As with any AI application in healthcare, there are inherent risks related to data privacy and ethical use, necessitating strict adherence to ethical guidelines and data protection regulations.
5. **Maintenance and Updating**: Continuous updating and maintenance are required to keep the model effective, which could be resource intensive.
6. **Potential for Over-reliance**: There's a risk of over-reliance on the AI model for diagnosis, which could lead to underutilization of human expertise and judgment in complex cases.

11. CONCLUSION

The "Deep Learning Model for Eye Disease Prediction" project represents a significant stride forward in the application of artificial intelligence in healthcare, particularly in the domain of ophthalmology. By harnessing the capabilities of deep learning and transfer learning techniques, this project has successfully developed a model that not only achieves an 80% accuracy rate in diagnosing eye diseases but also addresses several critical challenges in the field.

The use of advanced AI models like Inception V3, VGG-16, and Xception for eye disease diagnosis is a testament to the potential of AI to revolutionize healthcare diagnostics. The deployment of the model on Heroku, ensuring wide accessibility and free access, underscores a commitment to democratizing healthcare technology. This approach significantly enhances the potential reach of the project, making advanced diagnostic tools available to a broader population, including those in remote or under-resourced areas.

As an open-source project, it stands as a beacon of collaborative innovation, inviting contributions from the global community to further refine and enhance its capabilities. This aspect of the project not only accelerates its development but also ensures that it remains at the forefront of technological advancement.

However, the journey of this project also highlights the inherent challenges in integrating AI into healthcare. The occasional inaccuracies, the dependence on high-quality data, and the need for substantial computational resources are reminders of the complexities involved in deploying AI solutions in real-world settings. Moreover, the project brings to the fore ethical considerations such as data privacy and the potential biases in AI decision-making, underscoring the need for ongoing vigilance and adherence to ethical standards.

Looking forward, the project opens up numerous possibilities for further research and development. The scalability of the model presents opportunities for expansion into other areas of medical diagnostics. Continuous improvements in the model, driven by advances in AI and machine learning, promise even greater accuracy and efficiency in the future.

In conclusion, the "Deep Learning Model for Eye Disease Prediction" stands as a pioneering effort, blending technological innovation with practical application in healthcare. It exemplifies the transformative power of AI in improving diagnostic accuracy, accessibility, and efficiency. As the project evolves, it holds the promise of making a significant impact on global health, particularly in enhancing eye care and potentially saving the sight of millions worldwide.

12. FUTURE SCOPE

The future scope of this project holds promising opportunities for advancement and broader applications in AI-driven healthcare diagnostics. Key areas for exploration include:

- **Enhanced Model Precision**: Refining the current deep learning architecture to achieve higher diagnostic accuracy.
- **Diverse Training Data**: Expanding the dataset's diversity for improved model generalization across different demographics.
- **Integration with Telemedicine**: Incorporating the model into telemedicine platforms to extend its reach and offer remote diagnostic services.
- **Real-Time Monitoring**: Exploring real-time monitoring capabilities for early detection and intervention in eye diseases.
- **Collaboration with Healthcare Partners**: Collaborative efforts with healthcare institutions, research organizations, and industry partners for seamless integration into existing healthcare ecosystems.
- **Continuous Validation and Improvement**: Establishing mechanisms for continuous model validation and improvement based on feedback and evolving medical knowledge.
- **Multimodal Diagnostics**: Exploring the integration of multiple diagnostic modalities for a comprehensive understanding of eye health.

The future scope of this project extends beyond eye disease prediction, offering potential contributions to broader healthcare practices. Embracing these opportunities can position the project as a catalyst for advancements in accessible and accurate healthcare solutions.\

13. APPENDIX

Dataset Details:

The dataset used in this project is sourced from Kaggle and is available at Eye Diseases Classification Dataset. It provides a diverse collection of images for training and testing the deep learning models.

User Guide:

- On the Home Page, click on “Get Started”.
- Click "Upload Image" to input the eye image.
- Press "Predict" to obtain the model's diagnosis.

GitHub Link: https://github.com/smartinternz02/SI-GuidedProject-612884-1698842175.git

Project Link on Heroku: https://eye-vision-30e4dec7701a.herokuapp.com/

Demonstration Video Link:
https://www.dropbox.com/scl/fi/2rafwq06ax2b2bltfwgsu/EyeVision-Demo.mp4?rlkey=v7qc6b5l7p9w4lr80t7wkjse9&dl=0