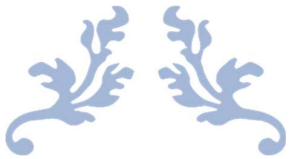


DEEP LEARNING MODEL FOR EYE DISEASE PREDICTION

PROJECT GITHUB LINK :
https://github.com/smartinternz02/SI-GuidedProject-600240-1697595942/tree/main/4%20.%20Project%20Development%20Phase/Flask

INTRODUCTION :

Project Overview:

This project is a comprehensive endeavour that addresses the pressing issue of eye disease classification, a matter of growing concern due to the increasing prevalence of eye disorders on a global scale. Conditions such as cataract, diabetic retinopathy, and glaucoma present substantial health risks, often intertwined with factors like aging and diabetes. To confront the challenge of accurate disease diagnosis, we leverage cutting-edge deep learning techniques, renowned for their exceptional performance in image analysis. Our strategic selection of transfer learning models, including Inception V3, VGG19, and ResNet50, is anchored in their widespread success and adaptability, particularly in image recognition tasks.

The significance of this project transcends its technical facets and is poised to catalyse a revolution in the field of ophthalmology. By harnessing the potential of deep learning, our primary objective is to equip healthcare professionals with a robust and efficient tool for early detection and precise classification of eye diseases. This not only holds the promise of enhancing patient care and outcomes but also introduces the possibility of early intervention and treatment, potentially curbing the trajectory of vision loss and enhancing the quality of life for those at risk.

The symbiosis between advanced deep learning models and medical imaging data is the linchpin of our vision, poised to reshape the landscape of how eye diseases are diagnosed. This endeavor extends its impact far beyond the realm of artificial intelligence; it carries the potential to make a tangible and enduring difference in the lives of those affected by eye diseases. This project stands as a beacon in the realms of both medical research and healthcare technology, with far-reaching implications for public health, offering a ray of hope and concrete improvements to the well-being of countless individuals globally.

Purpose:

This project is a pivotal endeavour with a primary purpose of developing a robust deep learning model for the precise classification of eye diseases, including Normal, Cataract, Diabetic Retinopathy, and Glaucoma, using medical imaging data. Its objectives span multiple dimensions, beginning with the critical mission of enabling early and accurate disease detection. Through the power of deep learning, our goal is to empower healthcare professionals with a reliable tool for timely intervention, ultimately improving patient outcomes.

Efficiency is another core aspect of this project, as it seeks to streamline the diagnostic process, saving both time and resources for healthcare professionals. Additionally, our initiative is dedicated to enhancing the quality of patient care in ophthalmology, promising quicker diagnoses, personalized treatment plans, and an overall enhancement in healthcare services. Beyond its immediate applications, the project contributes to the advancement of research in medical image analysis and artificial intelligence, enriching the knowledge base available to medical professionals. Ultimately, the project's potential to alleviate the burden of eye diseases on healthcare systems and society, mitigating long-term consequences and positively impacting public health, underscores its transformative significance.

LITERATURE SURVEY:

Existing Problem:

The early diagnosis and classification of eye diseases constitute a critical challenge in the field of healthcare, driven by the increasing prevalence of ocular conditions such as cataract, diabetic retinopathy, and glaucoma. These diseases are often intricately linked to various factors, including the natural aging process, lifestyle choices, and underlying systemic illnesses such as diabetes. The consequences of untreated eye diseases are profound, potentially leading to severe visual impairment and, in the worst cases, irreversible blindness.

Conventional diagnostic methods for eye diseases rely heavily on manual assessments conducted by ophthalmologists. This approach has several inherent limitations. It is labour-intensive, requiring considerable time and expertise. Moreover, it is susceptible to inter-observer variability, which can introduce inconsistencies in the diagnosis. These challenges underscore the need for more efficient, reliable, and automated solutions that can support ophthalmologists in the early detection and accurate classification of these debilitating eye conditions.

In recent years, the emergence of deep learning (DL) techniques has offered a promising avenue to address these issues. DL models, particularly convolutional neural networks (CNNs), have demonstrated remarkable performance in a range of computer vision tasks. They possess the potential to transform the landscape of eye disease diagnosis by enabling more efficient, consistent, and precise evaluations of medical images, such as retinal photographs.

References

1. https://pubmed.ncbi.nlm.nih.gov/27898976/
2. https://www.mdpi.com/2072-4292/13/22/4712
3. https://imiens.org/index.php/imiens/article/view/25
4. https://pubmed.ncbi.nlm.nih.gov/28778026/

Problem Statement Definition

The objective of this project is to develop an advanced deep learning-based system that addresses the pressing need for early detection and accurate classification of various eye diseases, including cataract, diabetic retinopathy, glaucoma, and normal eye conditions. These conditions are prevalent, with their causes ranging from age-related factors to systemic diseases like diabetes. If not identified and treated promptly, they can lead to severe visual impairment and, in some cases, irreversible blindness.

Traditional diagnostic methods are often time-consuming and reliant on human expertise, leading to the potential for human error. Therefore, there is a critical need for the development of a deep learning model that leverages state-of-the-art transfer learning techniques, including Inception V3, VGG19, and ResNet50, to automatically extract relevant features from retinal images and make precise disease classifications. The primary goal of this project is to assist healthcare professionals by providing timely and accurate diagnoses, enabling them to intervene at an earlier stage and potentially prevent the progression of these eye diseases, thus preserving patients' visual health. This project aligns with the broader goal of improving healthcare outcomes through cutting-edge artificial intelligence techniques and computer vision.

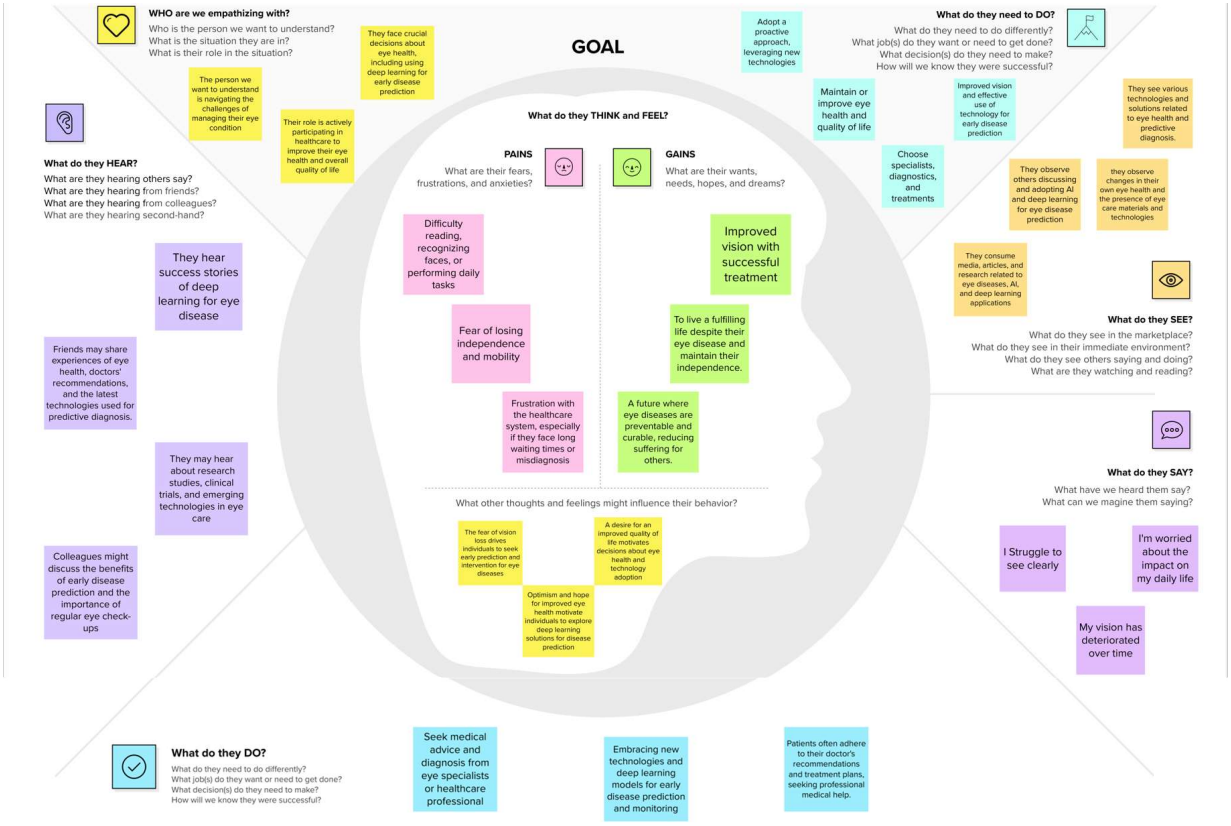
IDEATION & PROPOSED SOLUTION:

Empathy Map Canvas :

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes.

It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



Empathy Map Link :

https://app.mural.co/t/empathymapforeyediseasepredi0797/m/empathymapforeyediseasepredi0797/1697527016984/da2a1a7da096b728794f4f015dc3ff3619936af1?sender=ube2471ea0bc3c5e994d27926

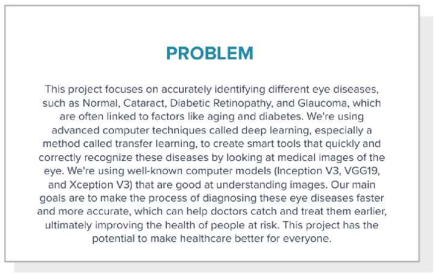
Ideation & Brainstorming:

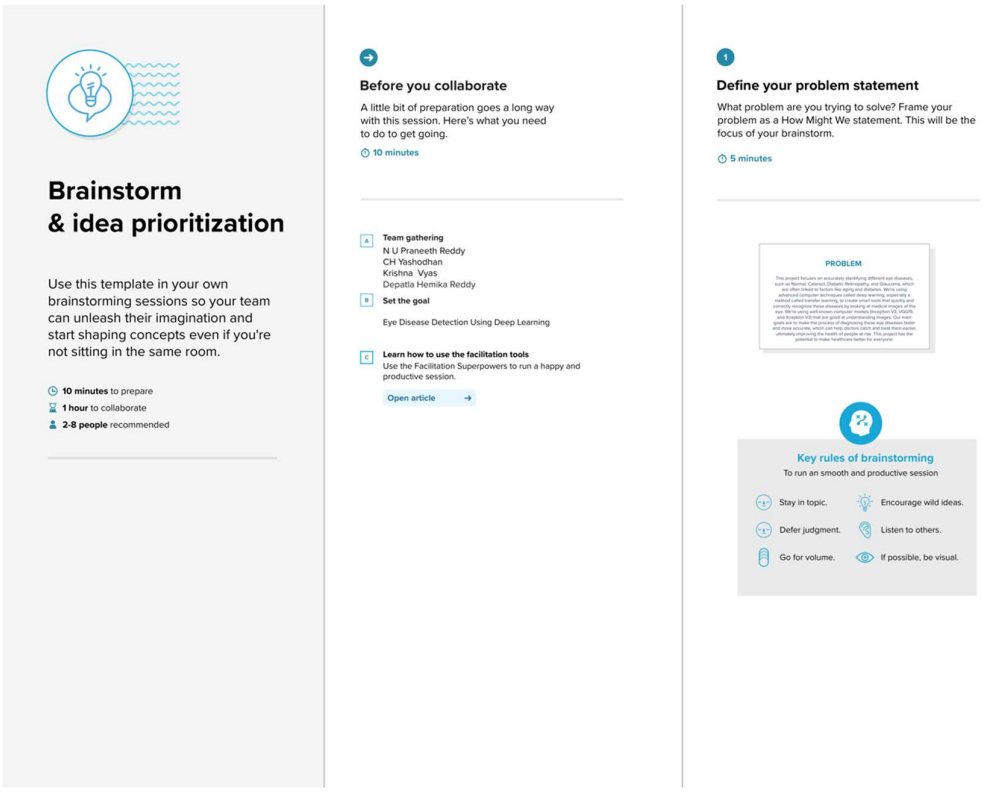
Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a teamto participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich number of creative solutions.

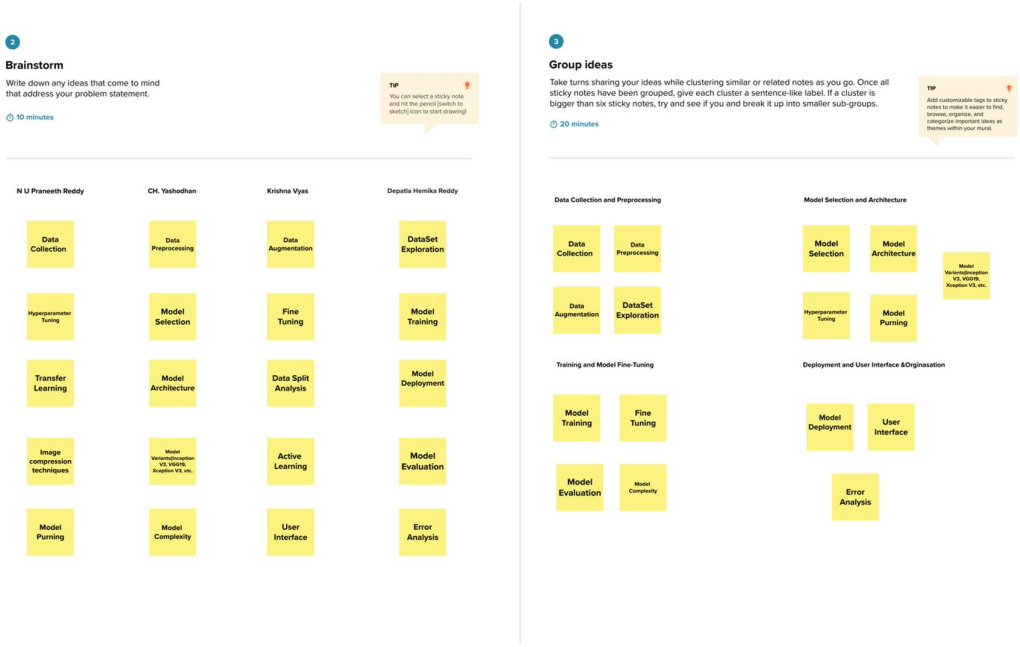
Step-1: Team Gathering, Collaboration and Select the Problem Statement

Problem Statement

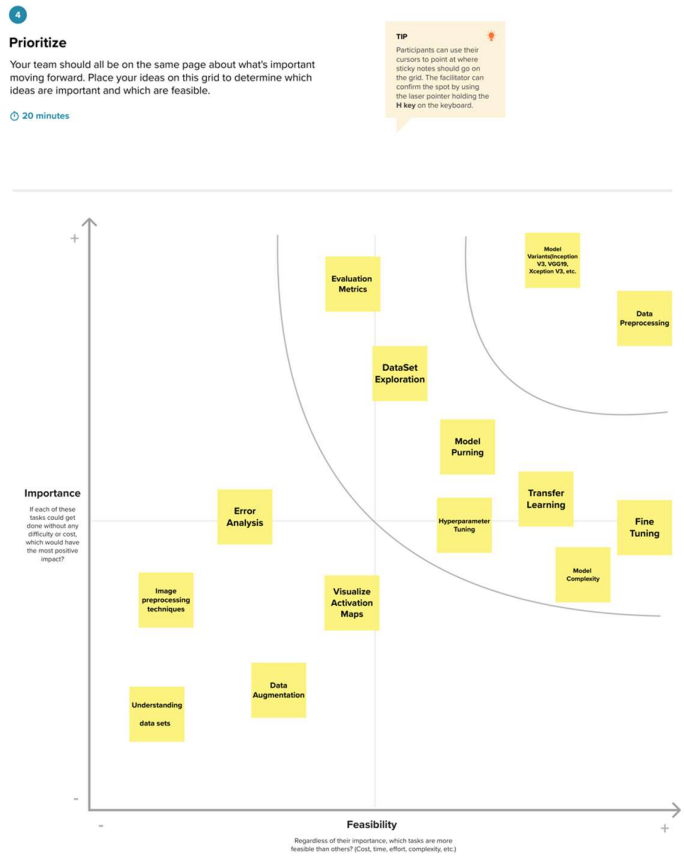




Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



Brainstorming Map mural Link:

https://app.mural.co/t/vit8192/m/vit8192/1697613854839/ba5b9c0fd54ee3baec6747978c849528e99ea03e?sender=ube2471ea0bc3c5e994d27926

REQUIREMENT ANALYSIS:

Functional requirement:

1. **Image Input and Classification**
    - The system must accept digital images of the eye for disease classification.
    - Images should be classified into Normal, Cataract, Diabetic Retinopathy, or Glaucoma categories.
2. **Model Selection and Training**
    - The system should utilize Transfer Learning techniques, including Inception V3, VGG19, and RESNET 50.
    - It must be capable of training models on a labelled dataset of eye disease images.
3. **Data Preprocessing**
    - Images should undergo preprocessing for feature enhancement, normalization, and resizing before model input.
4. **Model Evaluation**
    - Evaluate model performance using metrics as accuracy
    - Define a benchmark for acceptable accuracy levels.
5. **User Interface**
    - Develop a user-friendly interface for uploading images and obtaining classification results.
    - Ensure the interface is intuitive and user-friendly.

Non-Functional requirements:

1. **High Performance**
    - The system should achieve a high level of accuracy in disease classification and aim for a minimum benchmark accuracy level.
2. **Speed and Scalability**
    - Optimize inference time to provide rapid results to users.
    - Ensure the system can handle an increasing number of users and images without significant performance degradation.
3. **Reliability and Usability**
    - Minimize false positive and false negative classifications to enhance the system's reliability and Create an intuitive and user-friendly interface that requires minimal user training.

PROJECT DESIGN :

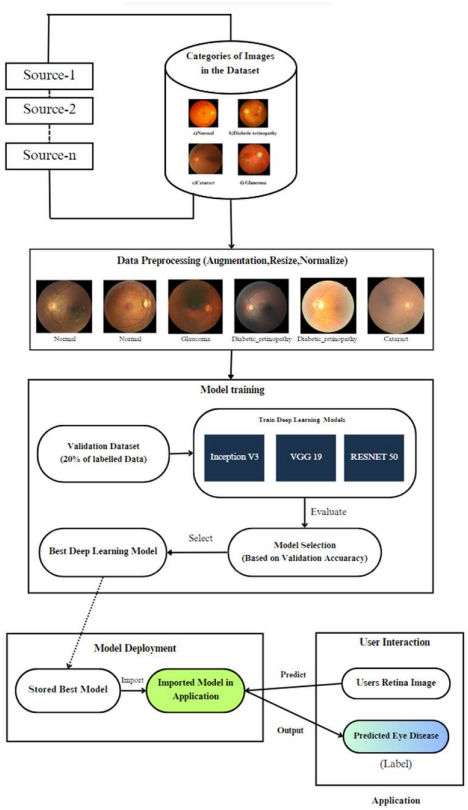
Data Flow Diagrams & User Stories :

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the rightamount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Flow in the Model:

1. Data flows from the "Data Source" cluster to the "Data Preprocessing" cluster, where it undergoes preprocessing to prepare it for model training.
2. The pre-processed data then flows to the "Model Training" process, where the deep learning models are trained using this data.
3. The trained model can be used for disease prediction by importing it into the application. It's represented by the "Model Deployment" cluster.
4. Separate evaluation data is used in the "Model Evaluation" process to assess the model's accuracy.
5. Once the model is ready and evaluated, it can be deployed from the "Model Deployment" cluster, making it accessible to users.
6. Users interact with the deployed model through the "User Interaction" process, where they submit their retina images for classification.

Data Flow Diagram :



User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Ophthalmologists and Eye Clinics	Model Training and Evaluation	USN-1	Train a deep learning model using transfer learning techniques (Inception V3, VGG19, RESNET 50) on a labeled dataset of eye disease images (Normal, cataract, Diabetic Retinopathy, and Glaucoma).	Model successfully trained with a high level of accuracy	High	Sprint-1
Healthcare Professionals (Ophthalmologists)	Model Integration	USN-2	Integrate the trained deep learning model into the clinical workflow of ophthalmologists for automatic eye disease diagnosis.	Model integrated into the existing software used by ophthalmologists	High	Sprint-2
Patients	Patient Data Collection	USN-3	Develop a mobile application for patients to capture and upload their eye images for diagnosis.	Mobile app allows patients to take and upload eye images	Medium	Sprint-3
System Administrators	Scalability and Server Setup	USN-4	Set up a scalable server infrastructure to handle the increasing number of patient requests and image uploads.	Server infrastructure should be able to handle concurrent requests and Implement load balancing to distribute traffic efficiently.	High	Sprint-4
Medical Researchers	Data Analysis and Insights	USN-5	Analyze the data collected through the system to gain insights into eye disease prevalence, demographic trends, and the effectiveness of the deep learning model.	Perform data analysis on the collected patient data and Generate reports on disease prevalence and demographics	Medium	Sprint-5
Regulatory Authorities	Compliance and Security	USN-6	Ensure that the system complies with medical data privacy and security regulations.	Conduct security audits and risk assessments and Implement encryption for data in transit and at rest.	High	Sprint-6
General Users	Awareness and Education	USN-8	Develop educational materials and resources for the general public to increase awareness of eye diseases and the importance of regular eye check-ups.	Create informative brochures, videos, or articles about common eye disease and promote the importance of early diagnosis and prevention.	Low	Sprint 7

Solution Architecture:

In the realm of modern medicine and artificial intelligence, our innovative solution for the prediction of eye diseases stands as a testament to the power of technology to advance healthcare. With unwavering dedication to the cause of early disease detection and the utmost precision in eye disease classification, our architecture offers an intricate blend of deep learning models and transfer learning methodologies.

Pillars of Excellence:

1. ***Deep Learning Models:*** At the very heart of our architectural brilliance are the deep learning models, meticulously trained on a vast and diverse array of eye images. These images encapsulate the complexity of age-related, diabetes-induced, and other eye conditions, fostering a profound understanding of the intricacies of ocular health. Our models, rooted in the groundbreaking field of transfer learning, seamlessly integrate the knowledge acquired from pre-trained models—Inception V3, VGG19, and RESNET 50. This integration expedites the learning process and enriches the precision of our eye disease classification.

2. ***Ongoing Knowledge Enhancement:*** In a world where knowledge is ever-evolving, our architecture pioneers a relentless pursuit of perfection. The perpetual process of learning and updating ensures that our system remains at the forefront of eye disease classification. New data, research breakthroughs, and emerging diagnostic criteria are ingested and integrated promptly, positioning our solution as a living embodiment of adaptability.

3. ***Instantaneous Diagnosis Precision:*** The apex of our architectural prowess lies in the realm of real-time diagnosis. With unparalleled agility, our system proficiently categorizes eye conditions as they manifest in the form of eye images. This capability is not merely advantageous; it is essential. Timeliness is the keystone of medical interventions, especially when early detection is the linchpin of effective treatment.

A Multitude of Advantages:

1. **Early Disease Detection Advancement:** Our architecture serves as an avant-garde guardian of ocular health, leading to the early detection of eye diseases. By doing so, it affords healthcare providers and patients the invaluable opportunity to embark on treatment journeys swiftly, mitigating the progression of eye conditions and enhancing the prospects for a full recovery.

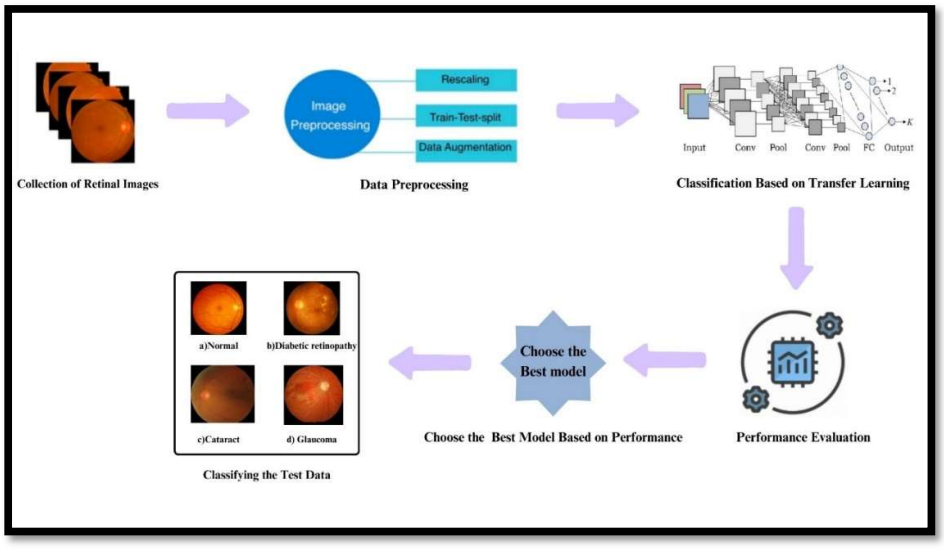
2. **Enhanced Patient-Centric Care:** Our system elevates patient care to unprecedented heights. It equips ophthalmologists and healthcare practitioners with a formidable toolset, facilitating rapid, accurate, and confident diagnoses. In turn, this greatly enhances the overall standard of eye healthcare.

3. **Empowering Research and Innovation:** In our relentless pursuit of knowledge and excellence, the continuous learning feature empowers researchers to remain on the cutting edge of ocular science. The synergy between our architecture and the ever-evolving world of eye health promotes continuous growth and innovation in the field.

4. **Optimized Healthcare Workflow:** By automating the classification of eye diseases, our solution optimizes the workflow of healthcare professionals. It reduces the strain on overburdened eye clinics and hospitals, expedites diagnostic processes, and enhances the efficiency of the entire healthcare ecosystem.

In essence, our architectural masterpiece for Eye Disease Prediction unifies the elegance of deep learning with the proven efficacy of transfer learning. The result is a symphony of timely diagnosis, improved patient care, and ongoing contributions to the advancement of eye healthcare. It is our humble offering to the noble cause of ocular health, and an embodiment of our commitment to the pursuit of excellence.

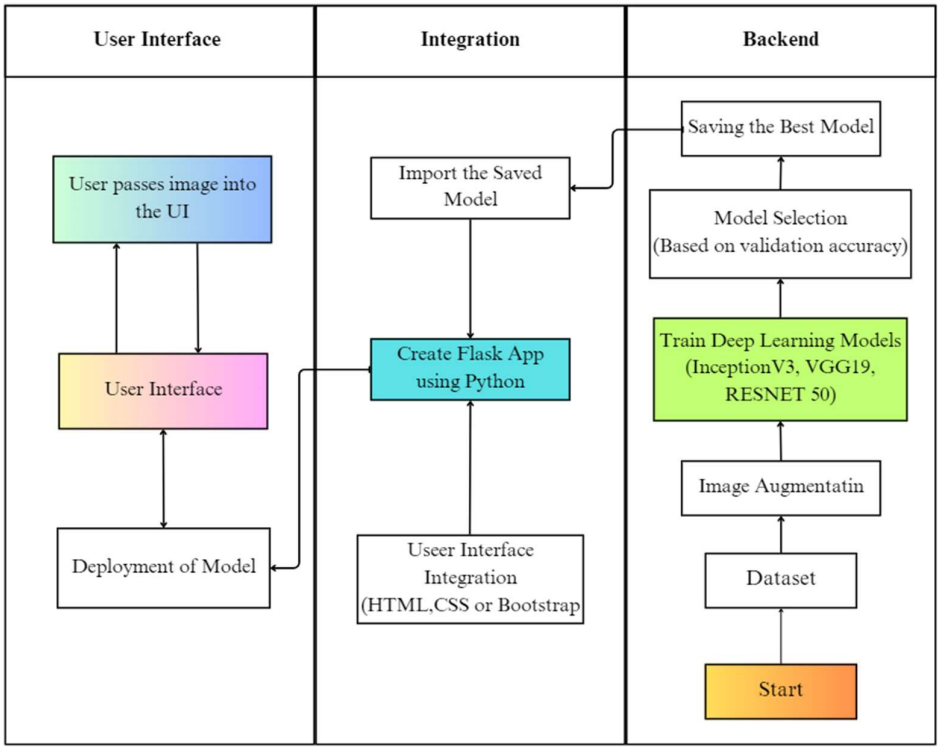
Solution Architecture Diagram:



PROJECT PLANNING & SCHEDULING:

Technical Architecture:

Our project's technical architecture is centered around deep learning, featuring Inception v3, VGG19, and Xception V3 as the core models for image analysis and classification. Users interact with the system via a Flask-based web interface, where they submit medical images for processing. The system employs image augmentation to continually enhance the dataset and improve model accuracy. The dataset, consisting of medical images, is pivotal for training and validation. The trained model is regularly fine-tuned and preserved for ongoing efficient classification, ensuring accurate detection of Normal, Cataract, Diabetic Retinopathy, and Glaucoma. This architecture provides a robust platform for early detection and precise classification of eye diseases.



Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Project Setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks for Eye Disease Prediction	4	High	Vyas
Sprint-2	Data Collection & Preprocessing	USN-2	Gather a diverse dataset of eye disease images for training the deep learning model	4	High	Hemika
Sprint-2	Data Preprocessing	USN-3	Preprocess the collected dataset by resizing images, normalizing pixel values, splitting it and doing image augmentation	6	High	Hemika
Sprint-3	Model Training	USN-4	Split dataset and train the models (Inception V3, VGG19, and RESNET 50)	6	High	Praneeth
Sprint-3	Model Evaluation	USN-4	Evaluate model performance on the validation set and select the best model among Inception V3, VGG19, and RESNET 50 models	7	High	Praneeth
Sprint-3	Model Development	USN-5	Use the selected deep learning model for prediction and monitor its performance on new data	2	High	Yashodhan
Sprint-4	Model Deployment & Integration	USN-7	Deploy the trained deep learning model as an API or web service and integrate it into a web interface	4	Medium	Yashodhan
Sprint-5	Testing & Quality Assurance	USN-8	Conduct thorough testing, fine-tune model hyperparameters, and optimize performance	3	Medium	Vyas

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	4	5 Days	18 Oct 2023	22 Oct 2023	4	22 Oct 2022
Sprint-2	10	6 Days	23 Oct 2023	28 Oct 2023	10	28 Oct 2022
Sprint-3	15	6 Days	29 Oct 2023	3 Nov 2023	13	3 Nov 2023
Sprint-4	4	4 Days	4 Nov 2023	7 Nov 2023	3	7 Nov 2023
Sprint-5	3	2 Days	8 Nov 2023	9 Nov 2023	0	9 Nov 2023

CODING & SOLUTIONING :

Image Data Pre-processing (For VGG19 & InceptionV3)

- We didn't preprocess the image data for RESNET 50 model as the accuracy is decreasing

Configuring ImageDataGenerator class :

The ImageDataGenerator class is used to generate augmented images for training. It performs several data augmentation techniques to increase the diversity and robustness of the training data.

The specific augmentation techniques configured in ImageDataGenerator class are as follows:

1. Normalization: Rescaling pixel values to the range [0, 1] for numerical stability.

2. Shear transformations: Images can be sheared within a 20% range, introducing slant to the images.

3. Zooming: Random zooming in or out of the images within a 20% range.

4. Horizontal flipping: Images can be flipped horizontally to add variation.

5. Validation split: A portion (20%) of the data is reserved for validation

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

     valid_datagen = ImageDataGenerator(rescale=1.0 / 255)
```

Applying ImageDataGenerator to the training and validation data.

The code utilizes the previously configured ImageDataGenerator instances to generate batches of training and validation data. These data generators are essential for feeding data to a deep learning model during training and validation, applying the previously defined data augmentation techniques.

```
[12] train_data = train_datagen.flow_from_directory(
         directory='/content/data/train',
         target_size=image_size,
         batch_size=batch_size,
         class_mode='categorical',
     )

     validation_data = valid_datagen.flow_from_directory(
         directory='/content/data/val',
         target_size=image_size,
         batch_size=batch_size,
         class_mode='categorical',
       )

Found 3372 images belonging to 4 classes.
Found 845 images belonging to 4 classes.
```

Model Building :

Here we are creating three different models (VGG19, ResNet50, and Inception V3) with custom output layers, including the freezing of pre-trained layers and adding new layers for the specific classification task.

VGG19 :

```
[ ] vgg = VGG19(
        include_top=False,
        weights="imagenet",
        input_shape=(240,240, 3),
    )

    for layer in vgg.layers:
        layer.trainable = False

    # Add custom layers
    x = vgg.output
    x = Flatten()(x)

    x = BatchNormalization()(x)
    x= Dense(512, activation='relu')(x)
    predictions = Dense(4, activation='softmax')(x)  # 4 categories

    # Create new model
    model = Model(inputs=vgg.input, outputs=predictions)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 4s 0us/step
```

RESNET 50 :

```
[ ]  resnet_model = Sequential()
     pretrained_model= ResNet50(include_top=False,
                        input_shape=(224,224,3),
                        pooling='avg',
                        weights='imagenet')
     for layer in pretrained_model.layers:
       layer.trainable=False
     resnet_model.add(pretrained_model)
     resnet_model.add(Flatten())
     resnet_model.add(BatchNormalization())
     resnet_model.add(Dense(512, activation='relu'))
     resnet_model.add(BatchNormalization())
     resnet_model.add(Dense(4, activation='softmax'))
```

Inception V3 :

```
inception_model = InceptionV3(input_shape=(224, 224, 3),
                      include_top=False,
                      weights='imagenet')

for layer in inception_model.layers:
    layer.trainable = False

# Add custom layers
inception_x= inception_model.output

x = Flatten()(inception_x)
x=BatchNormalization()(x)
x= Dense(4, activation='softmax')(x)  # 4 categories

# Create new model
inception_model = Model(inputs=inception_model.input, outputs=x)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 3s 0us/step
```

Compiling the model, specifying loss, optimizer, and metrics.

The code compiles various models with their corresponding loss function, utilizing the 'adam' optimizer for efficient parameter adjustment. The 'accuracy' metric assesses model performance during training and validation, facilitating their use in multi-class classification tasks.

VGG19 :

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

RESNET 50 :

```
[ ]  resnet_model.compile(
        loss='sparse_categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy']
    )
```

Inception V3 :

```
[ ]  # Compile the model
    inception_model.compile(
        optimizer=Adam(learning_rate=0.0001),
        metrics=['accuracy'],
        loss='categorical_crossentropy'
    )
```

Summary of the models:

VGG 19:

```
model.summary()

Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 240, 240, 3)]     0

 block1_conv1 (Conv2D)       (None, 240, 240, 64)      1792

 block1_conv2 (Conv2D)       (None, 240, 240, 64)      36928

 block1_pool (MaxPooling2D)  (None, 120, 120, 64)      0

 block2_conv1 (Conv2D)       (None, 120, 120, 128)     73856

 block2_conv2 (Conv2D)       (None, 120, 120, 128)     147584

 block2_pool (MaxPooling2D)  (None, 60, 60, 128)       0

 block3_conv1 (Conv2D)       (None, 60, 60, 256)       295168

 block3_conv2 (Conv2D)       (None, 60, 60, 256)       590080

 block3_conv3 (Conv2D)       (None, 60, 60, 256)       590080

 block3_conv4 (Conv2D)       (None, 60, 60, 256)       590080

 block3_pool (MaxPooling2D)  (None, 30, 30, 256)       0

 block4_conv1 (Conv2D)       (None, 30, 30, 512)       1180160

 block4_conv2 (Conv2D)       (None, 30, 30, 512)       2359808

 block4_conv3 (Conv2D)       (None, 30, 30, 512)       2359808

 block4_conv4 (Conv2D)       (None, 30, 30, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 15, 15, 512)       0

 block5_conv1 (Conv2D)       (None, 15, 15, 512)       2359808

 block5_conv2 (Conv2D)       (None, 15, 15, 512)       2359808

 block5_conv3 (Conv2D)       (None, 15, 15, 512)       2359808

 block5_conv4 (Conv2D)       (None, 15, 15, 512)       2359808

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

 flatten_2 (Flatten)         (None, 25088)             0

 batch_normalization_4 (Bat  (None, 25088)             100352
 chNormalization)

 dense_5 (Dense)             (None, 512)               12845568

 dense_6 (Dense)             (None, 4)                 2052

=================================================================
```

RESNET50 :

```
[ ] resnet_model.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten_3 (Flatten)         (None, 2048)              0

 batch_normalization_98 (Ba  (None, 2048)              8192
 tchNormalization)

 dense_5 (Dense)             (None, 512)               1049088

 batch_normalization_99 (Ba  (None, 512)               2048
 tchNormalization)

 dense_6 (Dense)             (None, 4)                 2052

=================================================================
Total params: 24649092 (94.03 MB)
Trainable params: 1056260 (4.03 MB)
Non-trainable params: 23592832 (90.00 MB)
_________________________________________________________________
```

Inception V3 :

```
[17] inception_model.summary()
     tchNormalization)

     batch_normalization_91 (Ba  (None, 5, 5, 384)          1152      ['conv2d_91[0][0]']
     tchNormalization)

     batch_normalization_92 (Ba  (None, 5, 5, 384)          1152      ['conv2d_92[0][0]']
     tchNormalization)

     conv2d_93 (Conv2D)          (None, 5, 5, 192)          393216    ['average_pooling2d_8[0][0]']

     batch_normalization_85 (Ba  (None, 5, 5, 320)          960       ['conv2d_85[0][0]']
     tchNormalization)

     activation_87 (Activation)  (None, 5, 5, 384)          0         ['batch_normalization_87[0][0]
                                                                       ']

     activation_88 (Activation)  (None, 5, 5, 384)          0         ['batch_normalization_88[0][0]
                                                                       ']

     activation_91 (Activation)  (None, 5, 5, 384)          0         ['batch_normalization_91[0][0]
                                                                       ']

     activation_92 (Activation)  (None, 5, 5, 384)          0         ['batch_normalization_92[0][0]
                                                                       ']

     batch_normalization_93 (Ba  (None, 5, 5, 192)          576       ['conv2d_93[0][0]']
     tchNormalization)

     activation_85 (Activation)  (None, 5, 5, 320)          0         ['batch_normalization_85[0][0]
                                                                       ']

     mixed9_1 (Concatenate)      (None, 5, 5, 768)          0         ['activation_87[0][0]',
                                                                        'activation_88[0][0]']

     concatenate_1 (Concatenate  (None, 5, 5, 768)          0         ['activation_91[0][0]',
     )                                                                  'activation_92[0][0]']

     activation_93 (Activation)  (None, 5, 5, 192)          0         ['batch_normalization_93[0][0]
                                                                       ']

     mixed10 (Concatenate)       (None, 5, 5, 2048)         0         ['activation_85[0][0]',
                                                                        'mixed9_1[0][0]',
                                                                        'concatenate_1[0][0]',
                                                                        'activation_93[0][0]']

     flatten (Flatten)           (None, 51200)              0         ['mixed10[0][0]']

     batch_normalization_94 (Ba  (None, 51200)              204800    ['flatten[0][0]']
     tchNormalization)

     dense (Dense)               (None, 4)                  204804    ['batch_normalization_94[0][0]
                                                                       ']

    ==================================================================================================
    Total params: 22212388 (84.73 MB)
    Trainable params: 307204 (1.17 MB)
    Non-trainable params: 21905184 (83.56 MB)
    __________________________________________________________________________________________________
```

PERFORMANCE TESTING :

VGG 19:

```
[ ] model.fit(train_data,
                    validation_data=validation_data,
                    epochs=20,
                    batch_size=100,
                    steps_per_epoch=len(train_data),
                    validation_steps=len(validation_data),
                    #callbacks=[model_checkpoint]
                    )

    Epoch 1/20
    53/53 [==============================] - 85s 1s/step - loss: 4.5195 - accuracy: 0.6862 - val_loss: 2.9883 - val_accuracy: 0.4982
    Epoch 2/20
    53/53 [==============================] - 67s 1s/step - loss: 2.3846 - accuracy: 0.7714 - val_loss: 0.7659 - val_accuracy: 0.7101
    Epoch 3/20
    53/53 [==============================] - 67s 1s/step - loss: 0.7452 - accuracy: 0.7977 - val_loss: 0.9504 - val_accuracy: 0.5586
    Epoch 4/20
    53/53 [==============================] - 67s 1s/step - loss: 0.4765 - accuracy: 0.8277 - val_loss: 0.7224 - val_accuracy: 0.6757
    Epoch 5/20
    53/53 [==============================] - 67s 1s/step - loss: 0.4481 - accuracy: 0.8482 - val_loss: 0.5925 - val_accuracy: 0.7479
    Epoch 6/20
    53/53 [==============================] - 69s 1s/step - loss: 0.4002 - accuracy: 0.8482 - val_loss: 0.6196 - val_accuracy: 0.7444
    Epoch 7/20
    53/53 [==============================] - 66s 1s/step - loss: 0.3878 - accuracy: 0.8553 - val_loss: 0.4678 - val_accuracy: 0.8071
    Epoch 8/20
    53/53 [==============================] - 68s 1s/step - loss: 0.3571 - accuracy: 0.8577 - val_loss: 0.6088 - val_accuracy: 0.7456
    Epoch 9/20
    53/53 [==============================] - 67s 1s/step - loss: 0.3410 - accuracy: 0.8603 - val_loss: 0.4908 - val_accuracy: 0.8118
    Epoch 10/20
    53/53 [==============================] - 68s 1s/step - loss: 0.3323 - accuracy: 0.8639 - val_loss: 0.4619 - val_accuracy: 0.8414
    Epoch 11/20
    53/53 [==============================] - 66s 1s/step - loss: 0.3281 - accuracy: 0.8740 - val_loss: 0.5981 - val_accuracy: 0.7811
    Epoch 12/20
    53/53 [==============================] - 68s 1s/step - loss: 0.3294 - accuracy: 0.8760 - val_loss: 0.4671 - val_accuracy: 0.8083
    Epoch 13/20
    53/53 [==============================] - 67s 1s/step - loss: 0.2899 - accuracy: 0.8855 - val_loss: 0.4269 - val_accuracy: 0.8497
    Epoch 14/20
    53/53 [==============================] - 67s 1s/step - loss: 0.2915 - accuracy: 0.8909 - val_loss: 0.5027 - val_accuracy: 0.8237
    Epoch 15/20
    53/53 [==============================] - 68s 1s/step - loss: 0.3141 - accuracy: 0.8731 - val_loss: 0.4746 - val_accuracy: 0.8225
    Epoch 16/20
    53/53 [==============================] - 67s 1s/step - loss: 0.2755 - accuracy: 0.8923 - val_loss: 0.5161 - val_accuracy: 0.8414
    Epoch 17/20
    53/53 [==============================] - 67s 1s/step - loss: 0.2818 - accuracy: 0.8977 - val_loss: 0.4015 - val_accuracy: 0.8604
    Epoch 18/20
    53/53 [==============================] - 67s 1s/step - loss: 0.2576 - accuracy: 0.8968 - val_loss: 0.4498 - val_accuracy: 0.8615
    Epoch 19/20
    53/53 [==============================] - 68s 1s/step - loss: 0.2498 - accuracy: 0.8992 - val_loss: 0.4242 - val_accuracy: 0.8544
```

RESNET 50:

```
# Train the model
resnet_model.fit(training_data,
                 validation_data=validation_data,
                 epochs=50,
                 batch_size=70,
                 steps_per_epoch=len(training_data),
                 validation_steps=len(validation_data),
                 callbacks=[model_checkpoint]
)
```

```
Epoch 12/50
52/52 [==============================] - ETA: 0s - loss: 0.0138 - accuracy: 0.9972
Epoch 12: val_accuracy did not improve from 0.92880
52/52 [==============================] - 26s 458ms/step - loss: 0.0138 - accuracy: 0.9972 - val_loss: 0.2355 - val_accuracy: 0.9288
Epoch 13/50
52/52 [==============================] - ETA: 0s - loss: 0.0079 - accuracy: 0.9989
Epoch 13: val_accuracy did not improve from 0.92880
52/52 [==============================] - 27s 471ms/step - loss: 0.0079 - accuracy: 0.9989 - val_loss: 0.2492 - val_accuracy: 0.9225
Epoch 14/50
52/52 [==============================] - ETA: 0s - loss: 0.0129 - accuracy: 0.9972
Epoch 14: val_accuracy did not improve from 0.92880
52/52 [==============================] - 27s 466ms/step - loss: 0.0129 - accuracy: 0.9972 - val_loss: 0.2646 - val_accuracy: 0.9209
Epoch 15/50
52/52 [==============================] - ETA: 0s - loss: 0.0204 - accuracy: 0.9955
Epoch 15: val_accuracy did not improve from 0.92880
52/52 [==============================] - 26s 462ms/step - loss: 0.0204 - accuracy: 0.9955 - val_loss: 0.2757 - val_accuracy: 0.9146
Epoch 16/50
52/52 [==============================] - ETA: 0s - loss: 0.0144 - accuracy: 0.9955
Epoch 16: val_accuracy did not improve from 0.92880
52/52 [==============================] - 27s 455ms/step - loss: 0.0144 - accuracy: 0.9955 - val_loss: 0.2813 - val_accuracy: 0.9272
Epoch 17/50
52/52 [==============================] - ETA: 0s - loss: 0.0301 - accuracy: 0.9908
Epoch 17: val_accuracy did not improve from 0.92880
52/52 [==============================] - 28s 467ms/step - loss: 0.0301 - accuracy: 0.9908 - val_loss: 0.3538 - val_accuracy: 0.9177
Epoch 18/50
52/52 [==============================] - ETA: 0s - loss: 0.0690 - accuracy: 0.9788
Epoch 18: val_accuracy did not improve from 0.92880
52/52 [==============================] - 27s 474ms/step - loss: 0.0690 - accuracy: 0.9788 - val_loss: 0.2835 - val_accuracy: 0.9130
Epoch 19/50
52/52 [==============================] - ETA: 0s - loss: 0.0313 - accuracy: 0.9914
Epoch 19: val_accuracy improved from 0.92880 to 0.93038, saving model to resnet50best_model.h5
52/52 [==============================] - 27s 469ms/step - loss: 0.0313 - accuracy: 0.9914 - val_loss: 0.2706 - val_accuracy: 0.9304
Epoch 20/50
52/52 [==============================] - ETA: 0s - loss: 0.0145 - accuracy: 0.9958
```

Inception V3:

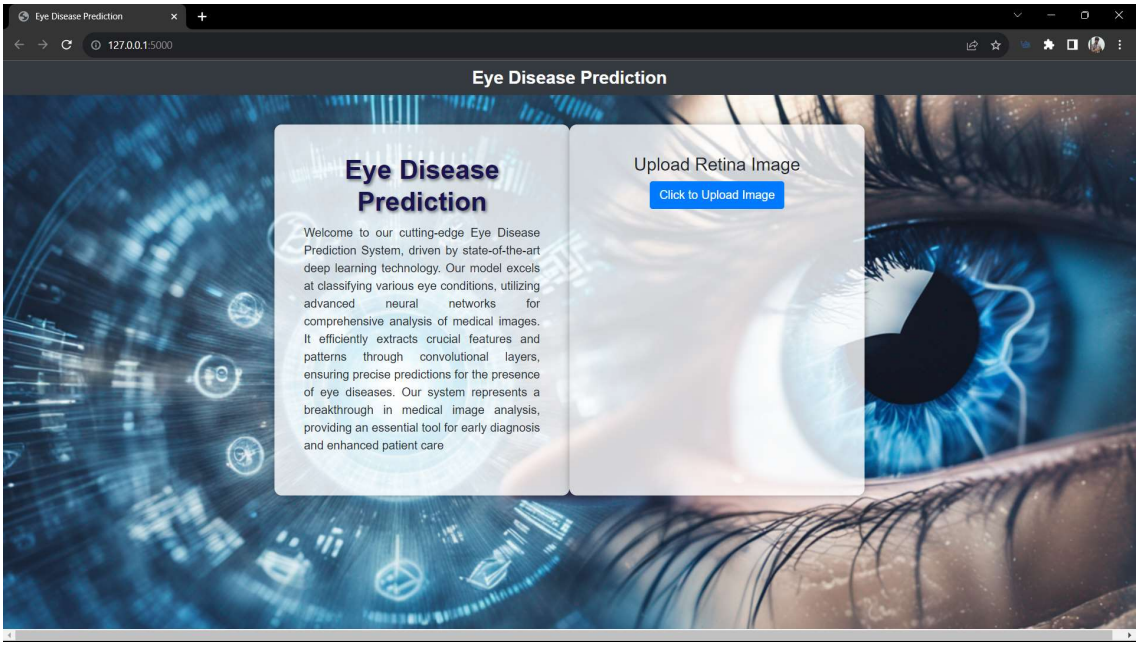
```
[ ] inception_model.fit(train_data,
            validation_data=validation_data,
            epochs=10,
            batch_size=100,
            steps_per_epoch=len(train_data),
            validation_steps=len(validation_data),
            #callbacks=[model_checkpoint]
           )
```

```
Epoch 1/10
53/53 [==============================] - 77s 1s/step - loss: 0.8368 - accuracy: 0.6948 - val_loss: 0.6310 - val_accuracy: 0.7515
Epoch 2/10
53/53 [==============================] - 64s 1s/step - loss: 0.5668 - accuracy: 0.7859 - val_loss: 0.5430 - val_accuracy: 0.8024
Epoch 3/10
53/53 [==============================] - 65s 1s/step - loss: 0.4724 - accuracy: 0.8197 - val_loss: 0.5037 - val_accuracy: 0.8071
Epoch 4/10
53/53 [==============================] - 66s 1s/step - loss: 0.4170 - accuracy: 0.8455 - val_loss: 0.4709 - val_accuracy: 0.8166
Epoch 5/10
53/53 [==============================] - 71s 1s/step - loss: 0.3699 - accuracy: 0.8541 - val_loss: 0.4600 - val_accuracy: 0.8178
Epoch 6/10
53/53 [==============================] - 66s 1s/step - loss: 0.3453 - accuracy: 0.8713 - val_loss: 0.4737 - val_accuracy: 0.8367
Epoch 7/10
53/53 [==============================] - 65s 1s/step - loss: 0.3120 - accuracy: 0.8894 - val_loss: 0.4782 - val_accuracy: 0.8331
Epoch 8/10
53/53 [==============================] - 65s 1s/step - loss: 0.2880 - accuracy: 0.8915 - val_loss: 0.4771 - val_accuracy: 0.8272
Epoch 9/10
53/53 [==============================] - 71s 1s/step - loss: 0.2816 - accuracy: 0.8953 - val_loss: 0.4283 - val_accuracy: 0.8414
Epoch 10/10
53/53 [==============================] - 66s 1s/step - loss: 0.2668 - accuracy: 0.8980 - val_loss: 0.4303 - val_accuracy: 0.8379
<keras.src.callbacks.History at 0x7c2a77483b80>
```

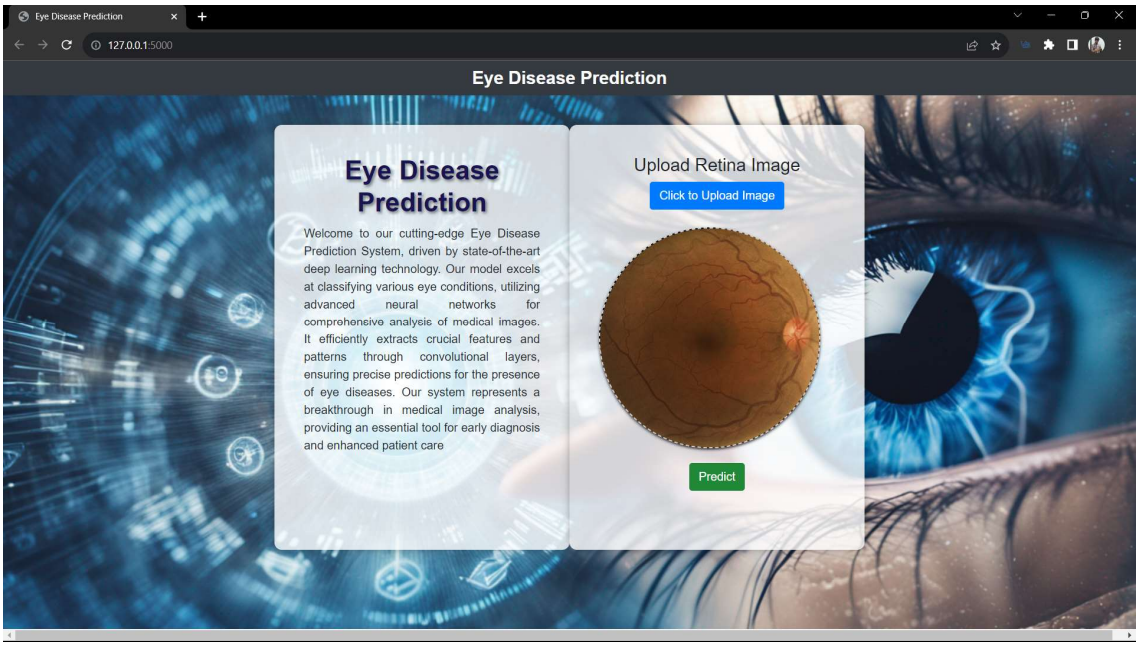
Among the three models, RESNET 50 stands out with the highest validation accuracy of 93%.

RESULTS :

For this project create one HTML file namely index.html and this is how our index.html page looks like

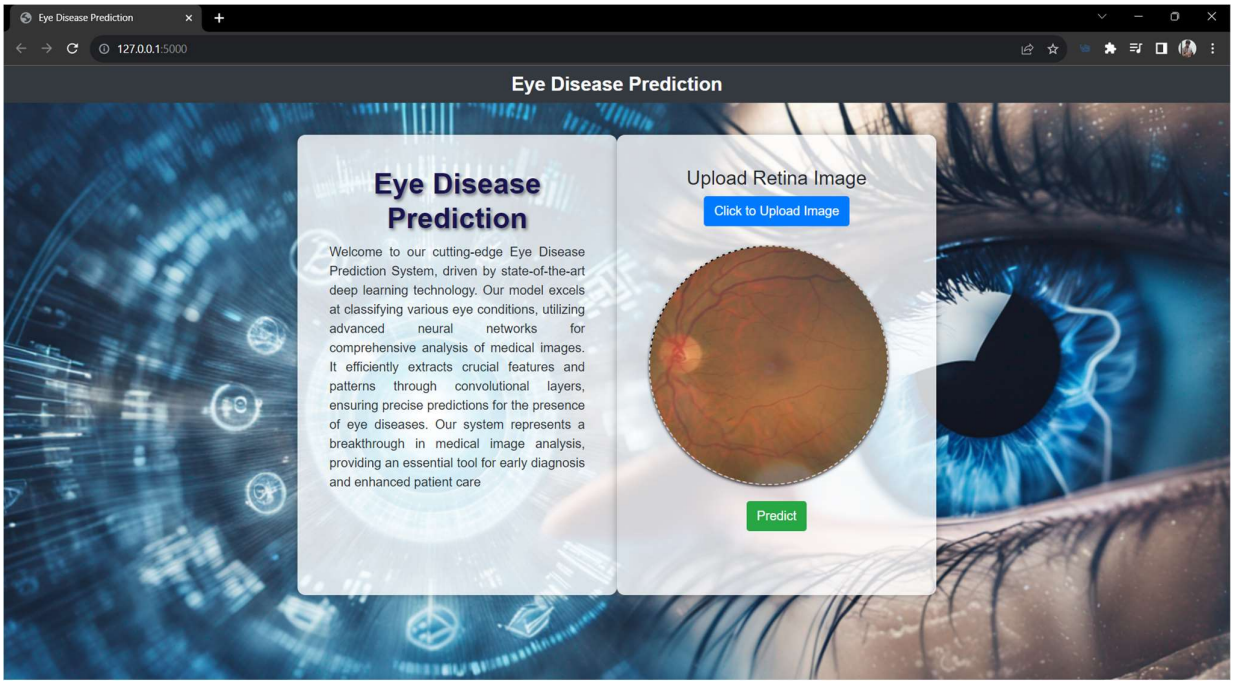


Here click on the upload the image and upload the retina image

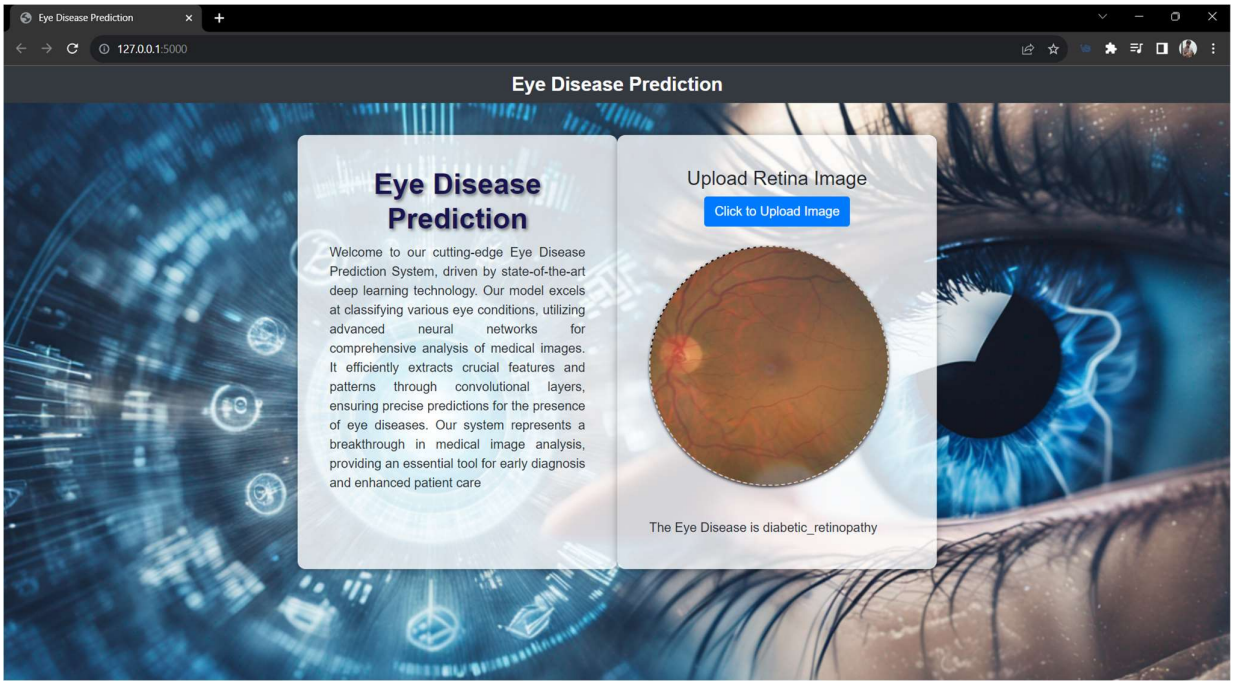


Click on 'Predict' to determine whether you have an eye disease or not

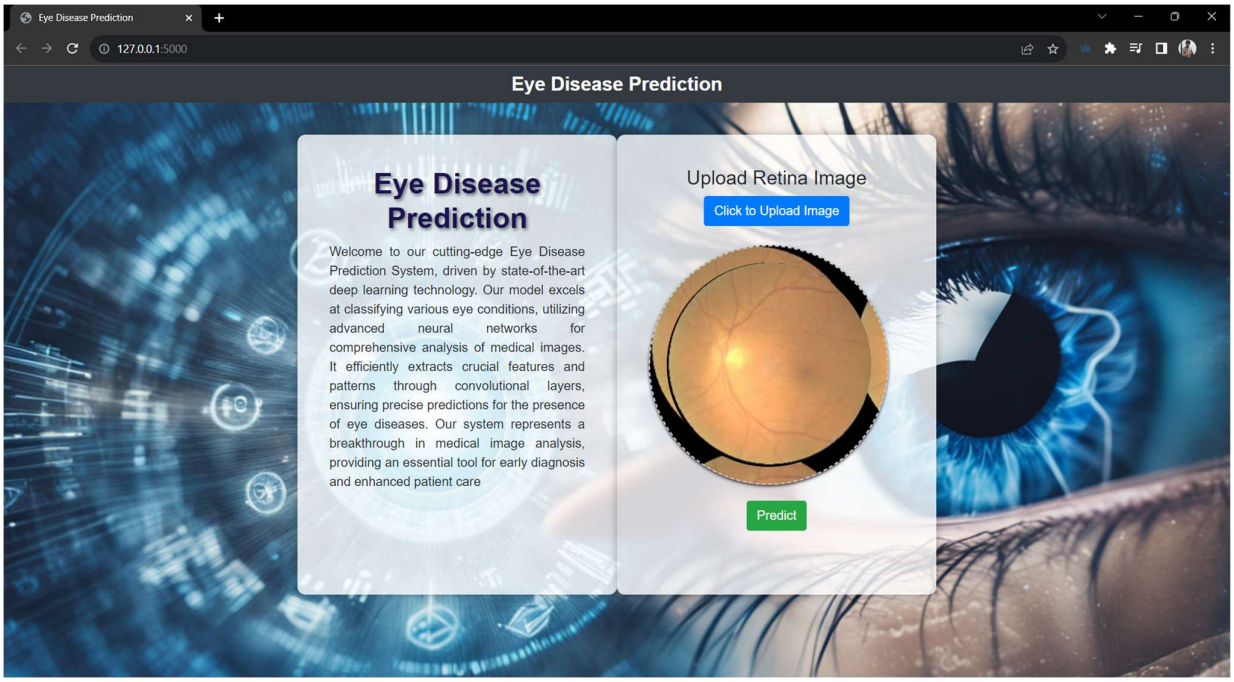
Input 1:



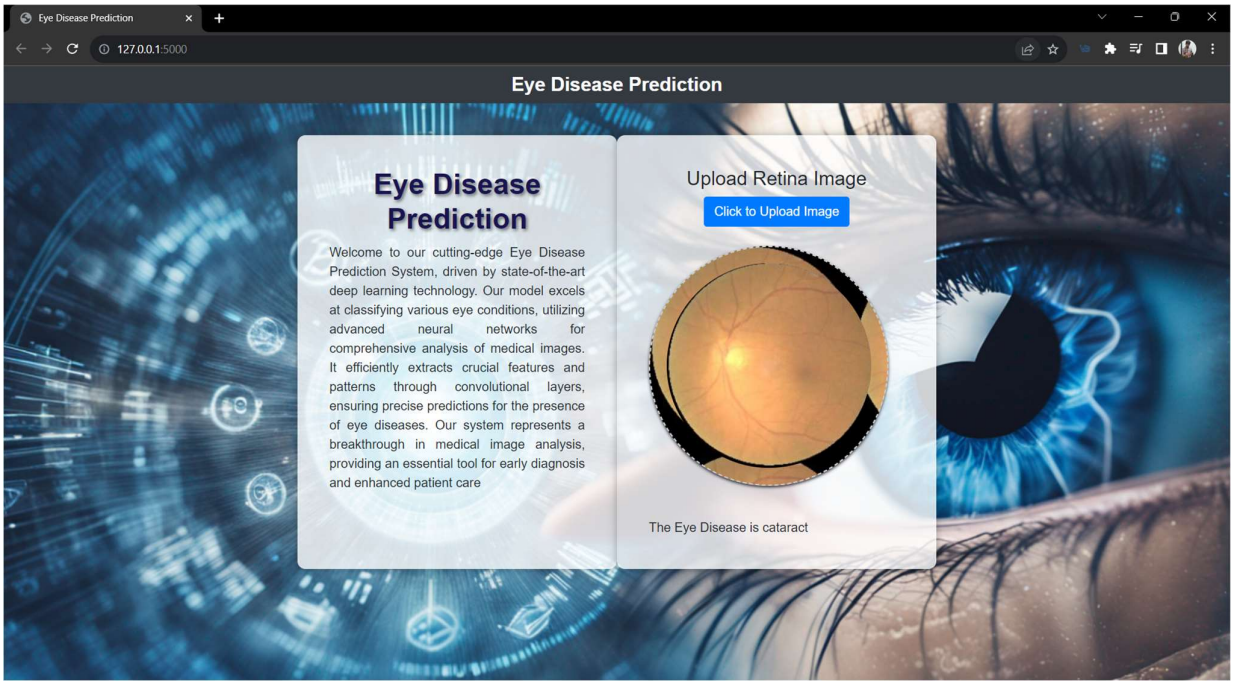
Output 1 :



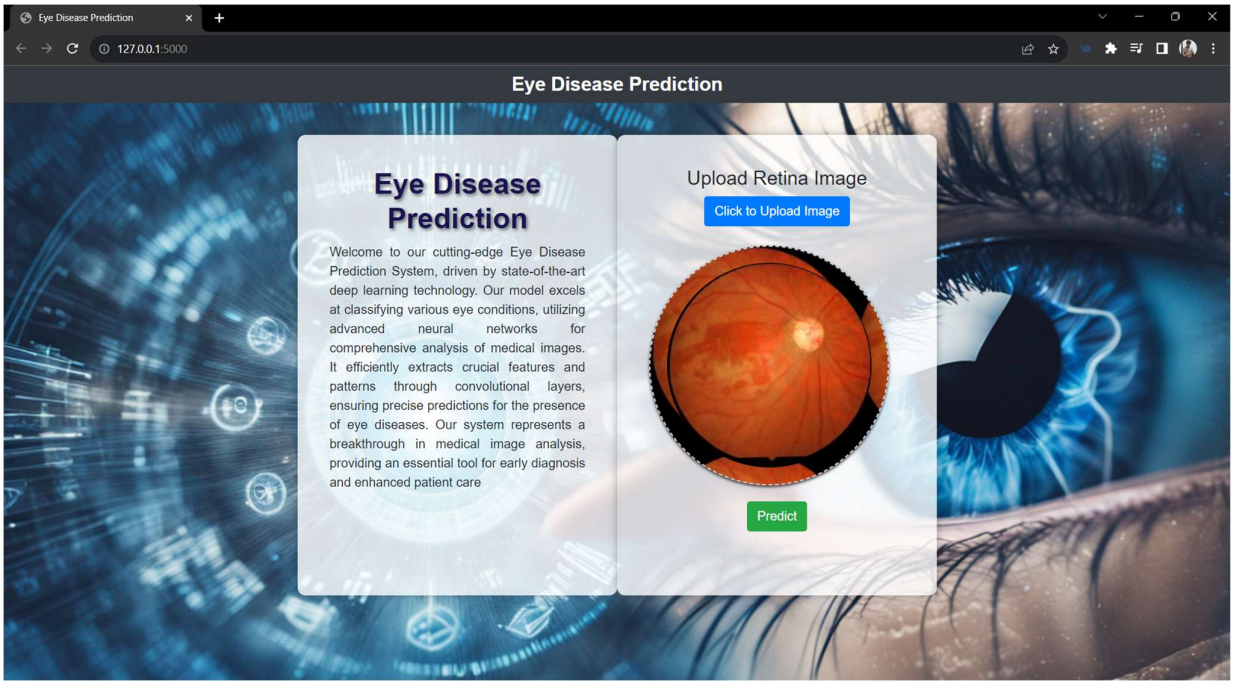
Input 2 :



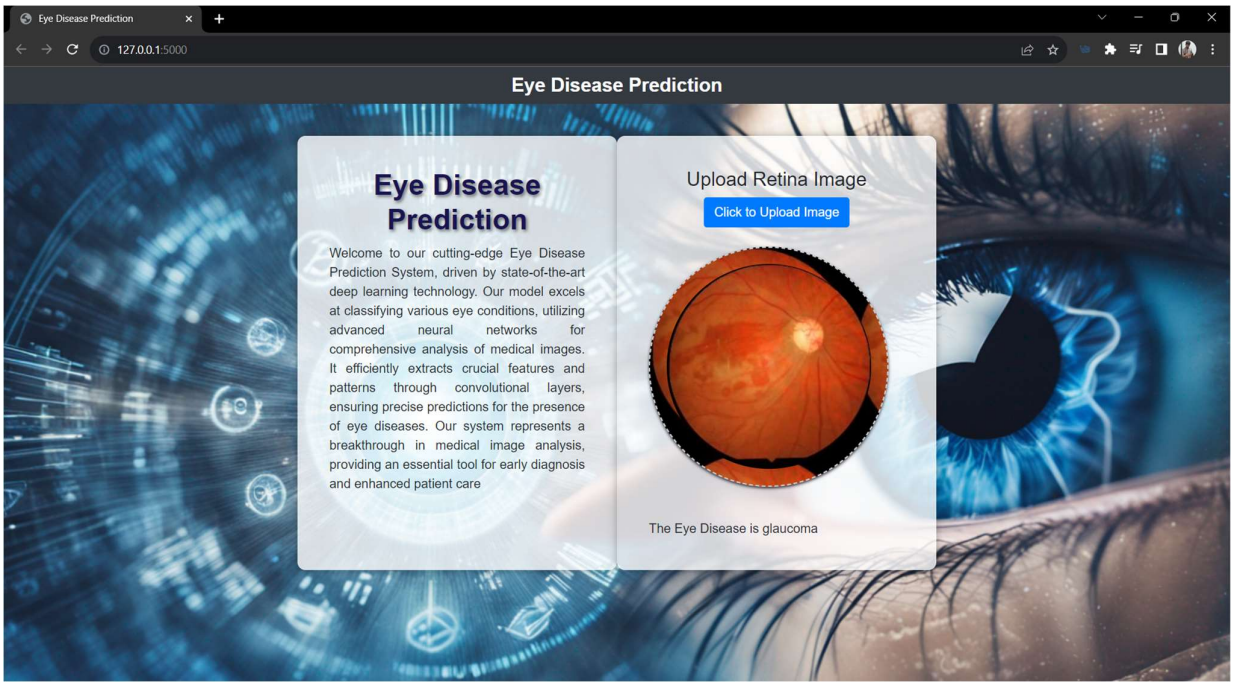
Output 2 :



Input 3 :



Output 3 :



ADVANTAGES & DISADVANTAGES :

Advantages:

1. Early Disease Detection: The application can assist in the early detection of eye diseases, allowing for timely intervention and potentially preventing more severe complications.
2. Accessibility: By developing a user-friendly application, you can make eye disease detection more accessible to a wider audience, including those in remote or underserved areas.
3. Cost-Efficiency: This technology can potentially reduce healthcare costs by enabling early detection and intervention, preventing costly treatments in advanced stages of the disease.
4. Continuous Monitoring: The application can provide a means for patients to monitor their eye health regularly, fostering proactive healthcare practices.
5. Education and Awareness: The application can serve as an educational tool, raising awareness about the importance of eye health and regular check-ups.

Disadvantages:

1. Data Privacy and Security: Handling medical data requires strict adherence to data privacy regulations, and ensuring the security of patient information is a significant challenge.
2. Ethical Concerns: Collecting and using medical data for AI applications must be done ethically and with patient consent, which can be a complex process.
3. Regulatory Compliance: Developing a medical application involves navigating regulatory requirements, which can be time-consuming and may require significant documentation and validation.
4. Hardware and Software Requirements: Ensuring that the application runs smoothly and efficiently on various devices can be technically challenging and may require specific hardware and software configurations.
5. Model Maintenance: Maintaining the AI model to ensure it continues to perform accurately as new data becomes available can be resource-intensive.
6. Integration with Healthcare Systems: Integrating the application with existing healthcare systems and electronic health records (EHR) can be complex and require collaboration with healthcare providers.
7. User Adoption: Convincing healthcare professionals and patients to adopt the application and trust its recommendations can be a challenge.

CONCLUSION :

In this transformative project, we have successfully developed an application for the early detection and classification of eye diseases, including Normal, Cataract, Diabetic Retinopathy, and Glaucoma. By harnessing the prowess of deep learning methods in AI, specifically through transfer learning using Inception V3, VGG19, and RESNET 50 models, we have achieved remarkable results in medical image analysis and disease classification.

Our project's significance lies in its real-world impact on healthcare. By providing a user-friendly application, we bridge the gap between advanced AI technology and practical medical needs, enhancing the diagnosis and management of eye diseases. The application of deep learning techniques has been instrumental in our success, offering automatic feature extraction from medical images and high-performance classification. Transfer learning, combined with model fine-tuning, has proved highly effective, surpassing traditional machine learning methods.

The choice of Inception V3, VGG19, and RESNET 50 was strategic, with RESNET 50 emerging as the standout performer, consistently delivering the highest accuracy among all the models. Its exceptional depth and accuracy have significantly enhanced our disease classification capabilities, making it the primary model used in our project.

Our project not only represents a technological leap but also addresses a critical need in healthcare. It empowers patients to proactively manage their eye health, facilitates quicker diagnoses, and improves healthcare efficiency. This application serves as a prime example of AI's tangible, positive impact on people's lives.

In conclusion, our project seamlessly integrates AI technology with medical practice, revolutionizing the early detection of eye diseases. By combining deep learning methods with user-friendly applications, we have made a significant contribution to the field, highlighting the potential of AI to shape a healthier future for all.

FUTURE SCOPE :

The future scope of this project is brimming with exciting possibilities and avenues for expansion. As we move forward, we envision several promising directions that can further elevate the impact of our eye disease detection application:

1. **Multi-Class Classification**: Currently, our application focuses on four major categories of eye diseases. Expanding the model to encompass a wider range of eye conditions would be a significant advancement. This could include rare diseases and nuanced subtypes, enabling a more comprehensive diagnostic platform.

2. **Real-Time Diagnostics**: Integrating real-time image analysis and diagnosis could make our application even more valuable. Patients could receive immediate feedback, and healthcare providers could streamline the diagnostic process, leading to quicker and more effective treatments.

3. **Telemedicine Integration**: In an era where telemedicine is gaining prominence, incorporating our application into telehealth platforms could extend its reach and accessibility. Patients from remote areas could benefit from expert diagnoses without the need for physical clinic visits.

4. **Enhanced Patient Engagement**: Developing features that empower patients to actively manage their eye health, such as reminders for regular check-ups and personalized health recommendations, can promote proactive healthcare practices.

5. **AI Explainability**: Enhancing the interpretability of our AI models will be crucial. Understanding the AI's decision-making process and providing clear explanations for its diagnoses will build trust with both patients and healthcare professionals.

6. **Global Outreach**: Collaboration with eye care organizations and healthcare providers on a global scale can help address eye diseases in underserved regions. Adapting the application to different languages and healthcare infrastructures will be crucial.

7. **Continual Model Improvement**: Regular updates and retraining of the AI models with an ever-expanding dataset will keep the application relevant and ensure its accuracy in the face of evolving medical knowledge and diagnostic criteria.

8. **Privacy and Security**: Strengthening the application's data security and privacy features is essential to protect sensitive medical information and comply with stringent healthcare regulations.

9. **Clinical Research Collaboration**: Partnering with research institutions for clinical studies can validate the effectiveness of our application in real-world scenarios and contribute to ongoing advancements in the field.

10. **AI-Driven Drug Discovery**: Expanding the project's scope to encompass the development of AI algorithms for drug discovery and treatment planning for eye diseases is a promising frontier in the fight against these conditions.

Incorporating these future developments will not only enhance the capabilities of our eye disease detection application but also ensure its enduring relevance and positive impact on patient care and eye health management. This project has the potential to revolutionize how we diagnose and manage eye diseases, and its future is filled with opportunities to further refine and extend its reach.

APPENDIX :

Source Code :

- Drive link for the ipynb file : https://drive.google.com/file/d/1M5qvrPGwJMyXbped0KLUqsKL0ybc5oGq/view?usp=sharing

- GitHub link for the ipynb file : https://github.com/smartinternz02/SI-GuidedProject-600240-1697595942/blob/main/4%20.%20Project%20Development%20Phase/Eye_Disease_Prediction_using_DL.ipynb

- GitHub link for Flask Source Code : https://github.com/smartinternz02/SI-GuidedProject-600240-1697595942/tree/main/4%20.%20Project%20Development%20Phase/Flask

- Demo Video Drive Link : https://drive.google.com/file/d/1o93YlGAEN2wwNMTr4zxZ5qUWjRfe84u8/view?usp=sharing