# Team-593093

# Project Documentation


VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

## Deep Learning Model For Eye Disease Prediction

*by*

Mundru Dharani Harshitha (21BRS1609)

Bhavya Sri Duggina (21BRS1574)

Maride Harshith (21BRS1462)

Andra Dushyanth Narendra Chowdary(21BRS1440)

# Introduction

Detecting and predicting eye diseases is of paramount importance for early diagnosis and intervention, as many ocular conditions, such as glaucoma, diabetic retinopathy, and age-related macular degeneration, can lead to irreversible vision loss if left untreated. Deep learning, particularly using transfer learning, has emerged as a powerful tool for addressing this critical healthcare challenge. Transfer learning leverages pre-trained neural networks on large datasets, allowing the adaptation of their learned features to new, smaller datasets, making it particularly well-suited for medical applications where collecting large labeled datasets can be challenging and expensive.

In recent years, transfer learning-based deep learning models have revolutionized the field of eye disease prediction by significantly improving the accuracy and efficiency of diagnosis. By utilizing neural networks that have already been trained on extensive non-medical datasets (such as ImageNet), these models can extract high-level features from medical images, such as retinal scans and fundus photographs, with remarkable precision. This enables ophthalmologists and healthcare professionals to make faster and more accurate predictions regarding the presence and progression of eye diseases. The versatility of transfer learning also allows for the integration of various architectural modifications, optimizing model performance for specific eye conditions.

This introduction of deep learning-based eye disease prediction models using transfer learning highlights the potential to transform the field of ophthalmology. By leveraging the power of pre-trained models and fine-tuning them with relevant medical data, these models offer a promising avenue for early diagnosis, patient-specific treatment planning, and monitoring disease progression. In this context, this article will explore the principles of transfer learning, the diverse deep learning architectures used in eye disease prediction, and the implications of these innovations for healthcare professionals and patients alike. Ultimately, the fusion of deep learning and transfer learning is poised to enhance the accuracy and accessibility of eye disease diagnosis, reducing the burden of preventable vision impairment and blindness on a global scale.

# LITERATURE SURVEY

A literature survey on deep learning models for eye disease prediction using transfer learning reveals a dynamic and rapidly evolving landscape. Researchers have harnessed the power of transfer learning to develop highly effective models for diagnosing and predicting a wide range of eye diseases. These models leverage pre-trained neural networks on large non-medical image datasets to extract valuable features, and then fine-tune them on medical images, such as retinal scans and fundus photographs. This approach has shown great promise in improving the accuracy and efficiency of eye disease diagnosis.

Several well-established datasets, including EyePACS, Kaggle Diabetic Retinopathy Detection, and MESSIDOR, have been widely adopted for training and evaluating these deep learning models. Research has encompassed various eye conditions, such as diabetic retinopathy, glaucoma, age-related macular degeneration, and more, indicating the versatility of transfer learning in addressing different diseases. The extensive use of transfer learning in these contexts has consistently led to high diagnostic accuracy, allowing for early intervention and personalized patient care.

Despite the successes, challenges remain in the field of eye disease prediction using deep learning and transfer learning. These challenges include concerns about the interpretability of deep learning models in clinical practice, domain adaptation issues, and privacy considerations when handling sensitive medical data. As the research continues to evolve, there is a growing emphasis on addressing these challenges, improving the generalization of models, and expanding the scope of eye diseases covered. The ongoing development of interpretability techniques, real-world implementation, and further exploration of model enhancements will likely shape the future of deep learning applications in ophthalmology, with the ultimate goal of enhancing patient outcomes and reducing vision loss.

Continuing the literature survey, one notable trend is the growing interest in addressing the interpretability and explainability of deep learning models applied to eye disease prediction. The "black-box" nature of neural networks has

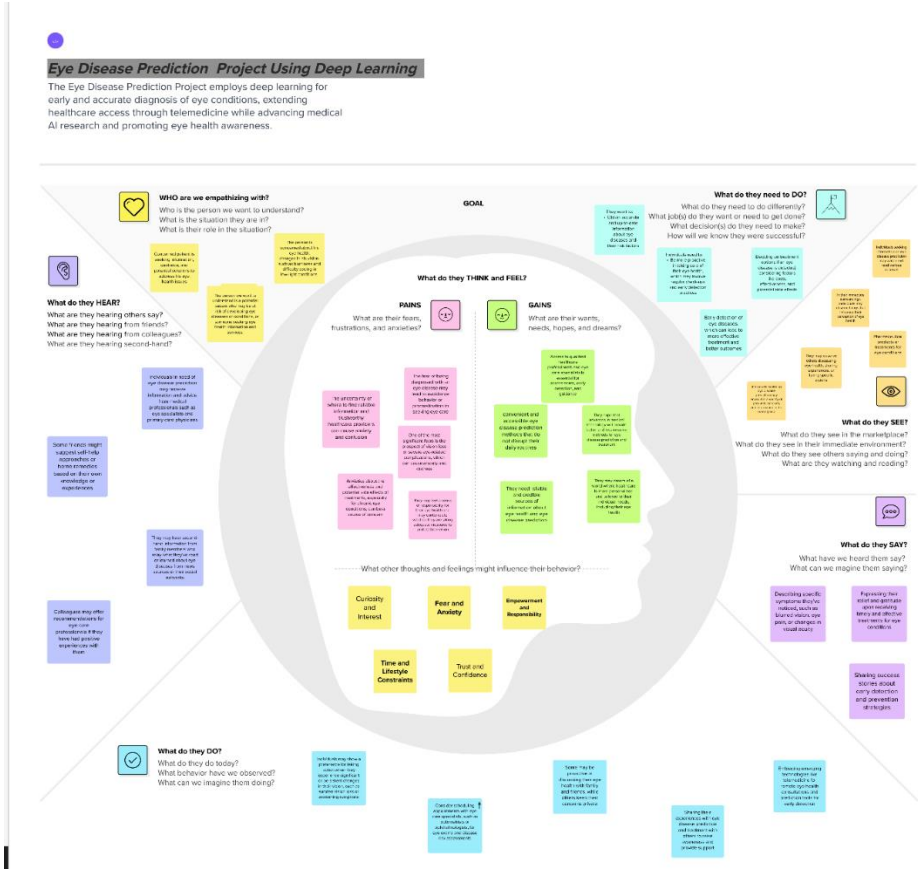
raised concerns, particularly in clinical settings where healthcare professionals need to understand and trust model predictions. Researchers are exploring techniques such as attention mechanisms, saliency maps, and model visualization to make these predictions more transparent and interpretable. By providing insights into the decision-making process of the model, it becomes easier for ophthalmologists to validate and integrate these predictions into their clinical workflow.

Practical implementation in real-world clinical settings is another focus of recent studies. Researchers are looking to seamlessly integrate deep learning models into telemedicine platforms, remote patient monitoring systems, and mobile applications. These efforts aim to make eye disease prediction more accessible to a broader patient population, especially in regions with limited access to specialized ophthalmic care. The combination of deep learning and telemedicine holds the potential to improve the reach and timeliness of diagnosis, leading to better patient outcomes.

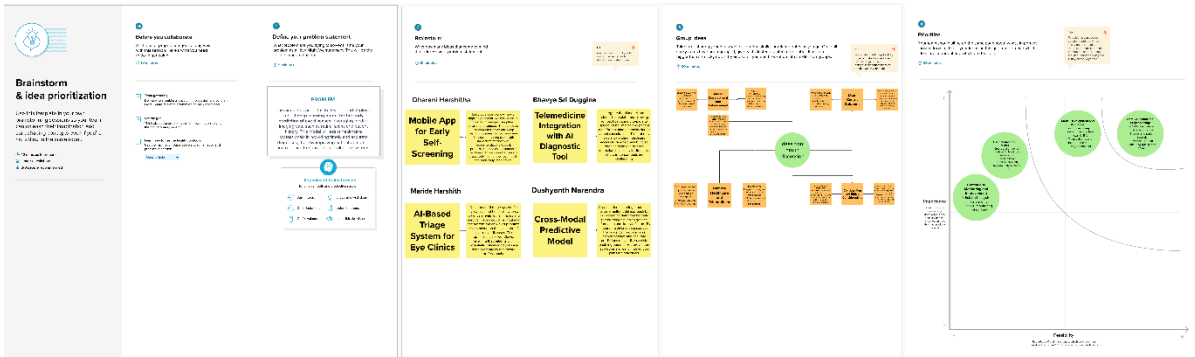
Looking forward, the literature suggests several key areas for future exploration. These include further advancements in addressing data privacy concerns while facilitating collaborative data-sharing initiatives, which can enhance model generalization and overall performance. Additionally, researchers are expected to continue adapting and customizing deep learning models for specific eye conditions and extending the use of multimodal data for more comprehensive assessments. As this field evolves, interdisciplinary collaboration between machine learning experts and healthcare professionals will play a pivotal role in shaping the future of eye disease prediction using transfer learning, ultimately contributing to improved patient care and a reduction in vision impairment and blindness.

# IDEATION & PROPOSED SOLUTION

## Empathy Map Canvas:




## Ideation & Brainstorming:



# REQUIREMENT ANALYSIS

To be used efficiently, all computer software needs certain tackle factors or other software coffers to be present on a computer. These prerequisites are known as (computer) system conditions and are frequently used as a guideline as opposed to an absolute rule. utmost software defines two sets of system conditions minimal and recommended. With adding demand for advanced processing power and coffers in newer performances of software, system conditions tend to increase over time. Assiduity judges suggest that this trend plays a bigger part in driving upgrades to being computer systems than technological advancements.

**NON-FUNCTIONAL Conditions:**

Inoperative conditions are the functions offered by the system. It includes time constraints and constraints on the development process and norms. The non-functional conditions are as follows Speed The system should reuse the given input into affair within applicable time.

Ease of use the software should be stoner friendly. Also the guests can use fluently, so it doesn't bear important training time.

Trustability The rate of failures should be lower also only the system is more dependable Portability It should be easy to apply in any system.

**SPECIFIC Conditions**

The specific conditions are:

stoner Interfaces The external druggies are the guests. All the guests can use this software for indexing and searching.

Hardware Interfaces The external tackle interface used for indexing and searching is particular computers of the guests. The PC's may be laptops with wireless LAN as the internet connections handed will be wireless.

Software Interfaces The Operating Systems can be any interpretation of Windows.

Performance Conditions The PC's used must be at least Pentium 4 machines so that they can give optimum performance of the product.

**SOFTWARE SPECIFICATIONS:**

Software conditions deal with defining software resource conditions and prerequisites that need to be installed on a computer to give optimal functioning of an operation.

These conditions or prerequisites are generally not included in the software installation package and need to be installed independently before the software is installed.

The system should be suitable to affiliate with the being system

- The system should be accurate
- The system should be better than the being system tackle SPECIFICATIONS

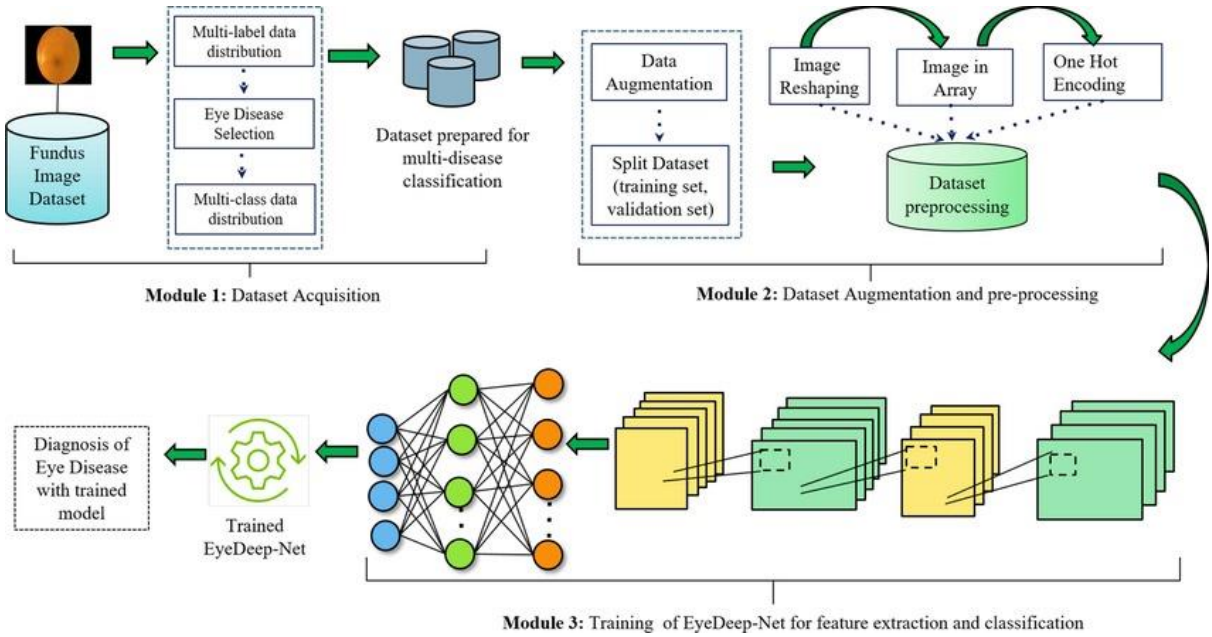
The most common set of conditions defined by any operating system or software operation is the physical computer coffers, also known as tackle, A tackle conditions list is frequently accompanied by a tackle comity list, especially in case of operating systems. An HCL lists tested, compatible, and occasionally inharmonious tackle bias for a particular operating system or operation. The following sub-sections bandy the colorful aspects of tackle conditions.

All computer operating systems are designed for a particular computer armature. utmost software operations are limited to particular operating systems running on particular infrastructures. Although armature-independent operating systems and operations live, utmost need to be reedited to run on a new armature.

The power of the central processing unit (CPU) is a abecedarian system demand for any software. utmost software running on x86 armature define processing power as the model and the timepiece speed of the CPU. numerous other features of a CPU that impact its speed and power, like machine speed, cache, and MIPS are frequently ignored.

# PROJECT DESIGN

Systems design is the process of defining the armature, factors, modules, interfaces, and data for a system to satisfy specified conditions. It may be considered the operation of systems proposition to the process of product development. The most popular ways for designing computer systems are decreasingly those that concentrate on object- acquainted analysis and design.



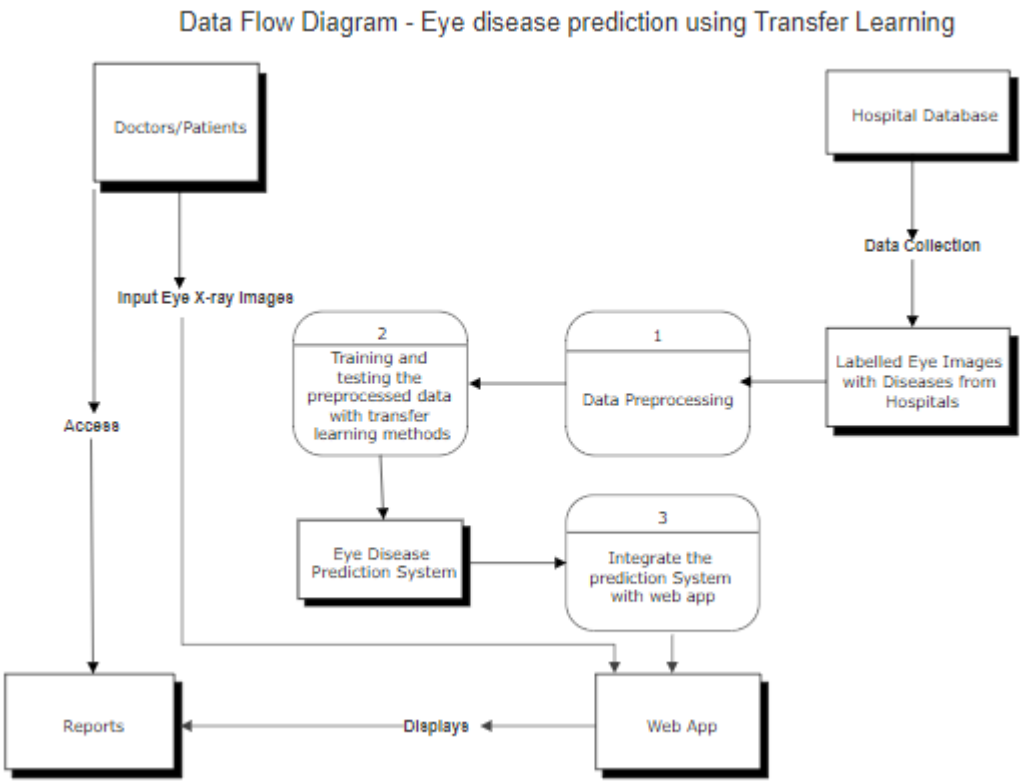

**Data Flow Diagram**

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 right quantum of the system demand graphically. It can be homemade, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

The ideal of a DFD is to show the compass and boundaries of a system as a whole. It may be used as a communication tool between a system critic and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data inflow graph or bubble map.

## Data flow Diagram




# User Stories

**User Stories For Eye Disease Prediction Project**

| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|---|---|---|---|---|---|---|
| Healthcare Professional | Eye Disease Prediction System | USN-1 | As a healthcare professional, I want to upload patient data and retinal images for eye disease prediction, so I can quickly assess a patient's eye health. | The system accepts patient data and retinal images. The uploaded data is validated for correctness. The system provides an initial prediction or risk assessment for the patient. | High | Sprint-1 |
| | | USN-2 | As a healthcare professional, I want to view detailed predictions for a patient, including disease category, risk level, and supporting data, so I can make informed decisions. | The system displays the disease category and risk level. It shows relevant patient data (age, gender, etc.). It provides visualizations for better understanding. | High | Sprint-1 |
| Patient | Patient Portal | USN-3 | As a patient, I want to access my eye disease prediction report and recommendations online, so I can be proactive about my eye health. | The system provides a secure login for patients. Patients can view their eye disease prediction reports. The system offers recommendations for next steps. | Medium | Sprint-2 |
| | | USN-4 | As a patient, I want to receive email notifications when my eye disease prediction is ready, so I can stay informed about my health status. | Patients receive email notifications upon report generation. Notifications include a secure link to the patient portal. | Medium | Sprint-2 |
| System Admin | System Management | USN-5 | As a system admin, I want to monitor the system's performance and ensure data security, so I can maintain the system's reliability. | The admin dashboard provides real-time system performance metrics. It alerts the admin about any security breaches or anomalies. | High | Sprint-1 |
| | | USN-6 | As a system admin, I want to update the deep learning model with new data and research, so I can improve the system's accuracy over time. | The admin can upload new datasets for model retraining. The system documents the model's performance after updates. | High | Sprint-1 |

## Solution Architecture:

The solution architecture of a Diabetic Retinopathy classification model typically involves several components and steps. Here's a high-level overview of the architecture for building and deploying such a model:

**Data Collection and Preprocessing:**

Data Collection: Gather a dataset of retinal images, where each image is labeled with the corresponding Diabetic Retinopathy severity level.

Data Preprocessing: Prepare the data by resizing images to a consistent size, normalizing pixel values, and augmenting the dataset if necessary. Ensure that the data is split into training, validation, and test sets.

**Model Selection:**

Choose an appropriate machine learning or deep learning model for image classification. Convolutional Neural Networks (CNNs),Transfer Learning Methods are commonly used for this task due to their ability to capture image features effectively.

**Model Training:**

Train the selected model on the training dataset. Use techniques like transfer learning if available pre-trained models such as VGG19, ResNet, or Inception to leverage their feature extraction capabilities.

**Hyperparameter Tuning**:

Optimize hyperparameters, such as learning rate, batch size, and the number of layers in your model, to improve performance. Use techniques like grid search or random search to find the best hyperparameters.

**Model Evaluation:** Evaluate the model's performance using the validation dataset. Common evaluation metrics include accuracy, precision, recall, F1-score, and the confusion matrix.

**Fine-Tuning:**

If the model's performance is not satisfactory, fine-tune the model by adjusting hyperparameters, changing the architecture, or increasing the size of the training dataset.

**Deployment:**

Once the model performs well on the validation dataset, deploy it to a production environment where it can make predictions on new, unseen data.

Create an API or web service for integration with other systems or applications.

Implement monitoring and logging to track model performance over time.

**User Interface:**

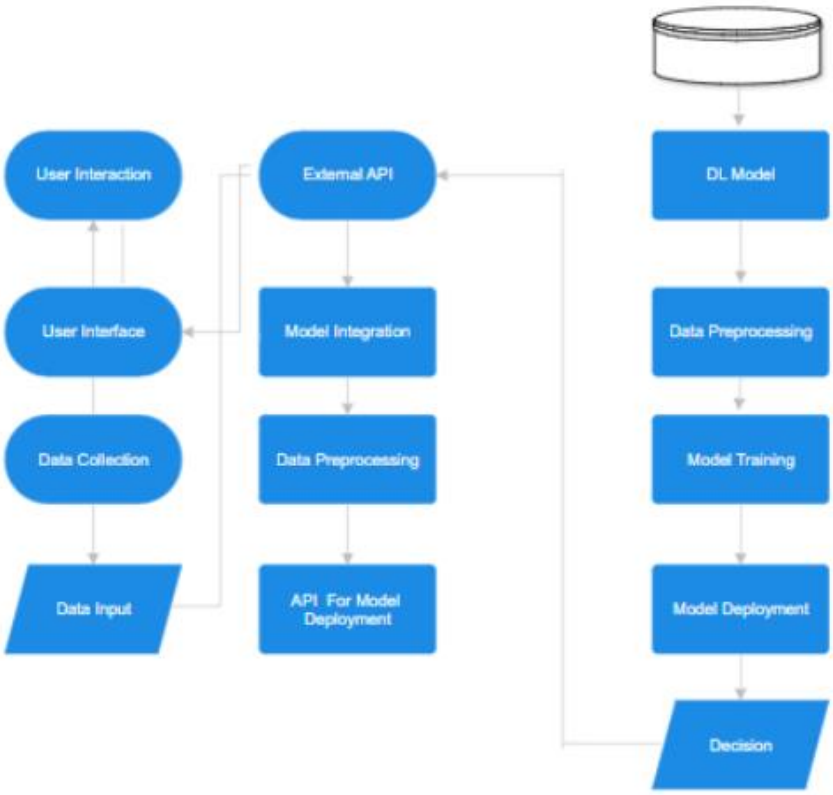
Develop a user interface that allows users to upload retinal images and receive predictions regarding Diabetic Retinopathy severity.

**Documentation:**

Create comprehensive documentation for the entire solution, including model architecture, deployment procedures, and user instructions.

**Testing and Quality Assurance:**

Thoroughly test the entire solution, including the model, user interface, and deployment, to ensure it functions as expected.



# PROJECT PLANNING & SCHEDULING

## Technical Architecture

| S.No | Component | Description | Technology |
|---|---|---|---|
| 1. | User Interface | Interface for user interaction the application along with creating an user friendly interface. | Web-based UI(HTML, CSS, JavaScript / Angular Js/React Js). |
| 2. | Application Logic-1 | Core Logic responsible for handling user requests. | Python, Flask, FastAPI or Node.js |
| 3. | Data Collection | Gathering eye disease images data from hospital database. | Web Scraping, Data warehouses, ETL tools |
| 4. | Data Input | Handling and preprocessing user provided input. | Forms, API's or command line input |
| 5. | External API | Integration with external data sources or services. | RESTful API's ( |
| 6. | Cloud Database | Database Service on Cloud (Storage and management of structured data). | Amazon RDS, Google Cloud SQL, Azure SQL. |
| 7. | Model Integration | Interface for integrating deep learning model. | RESTful API endpoints(Flask, FastAPI, JSON,XML) |
| 8. | API Model Deployment | Responsible for deploying deep learning models as APIs, enabling real-time predictions and external interaction. It ensures model accessibility and scalability. | Docker, Kubernetes etc. |
| 9. | Deep Learning Model | The predictive model for eye diseases using eye images. | Scikit-Learn, TensorFlow, PyTorch, Transfer Learning Techniques like VGG, Inception etc. |
| 10. | Data Preprocessing | Data preparation here, augmentation, normalization, resizing of images etc. | Pandas, Numpy , Scikit-learn or custom scripts. |
| 11. | Model Deployment | Hosting and Serving the deep learning model. | Streamlit, Flask, FastAPI. |
| 11. | Infrastructure (Server / Cloud) | Underlying Cloud infrastructure and resources. | AWS, Google Cloud, Azure or on-premises servers like local,Cloud Foundry etc. . |

## Application Characteristics:

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1. | Open-Source Frameworks | Utilizing open-source frameworks for model development and deployment, ensuring cost-efficiency and flexibility which makes easy for the user to predict the disease. | Scikit-Learn, TensorFlow, PyTorch for model development. – Streamlit orFlask or FastAPI for API deployment. - Kubernetes for container orchestration. - Jupyter Notebook for model prototyping and development. |
| 2. | Security Implementations | Implementing robust security measures to protect sensitive data, model APIs, and ensure user data privacy which helps hospitals to protect the patient's data and hospital's data from the hackers. | OAuth 2.0 or JWT for user authentication. Encryption (HTTPS/SSL) for data in transit. Role-based access control. Regular security audits and updates. Compliance with industry standards (e.g., GDPR). |
| 3. | Scalable Architecture | Designing a scalable architecture that can handle growing data volumes and user demands which can manage the huge inflow of user demands assuming as a big data. | - Microservices architecture for modularity and scalability. - Containerization with Docker and orchestration with Kubernetes. - Load balancers for distributing traffic. - Auto-scaling based on resource usage. |
| 4. | Availability | Ensuring high availability and minimal downtime for the application to support continuous data analysis and prediction which helps in predicting eye diseases accurately and rapidly. | Redundancy in database and API deployment. - Geographically distributed data centers or cloud regions. - Monitoring and alerting systems (e.g., Prometheus, Grafana). - Failover mechanisms for fault tolerance |
| 5. | Performance | Optimizing application performance to provide quick insights and predictions | Caching mechanisms for frequently accessed data. - Model optimization (e.g., quantization) for faster inference. - Load testing and performance tuning |

# Sprint Planning & Delivery Schedule:

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1,2 | Data Collection and Preprocessing | USN-1 | As a Data Scientist, I want to Clean and preprocess the collected data, addressing issues like noise, artifacts, and inconsistent image resolutions. | 15 | High | Mundru Dharani Harshitha |
| Sprint-3,4 | Model Architecture and Development | USN-2 | As a Machine Learning Engineer, I want to Implement the selected model architecture using a deep learning framework (e.g., TensorFlow or PyTorch).<br>Train the model on the preprocessed dataset, optimizing hyperparameters and incorporating data augmentation techniques to enhance generalization. | 16 | High | Bhavya Sri Duggina |
| Sprint-5,6 | Model Evaluation and Optimization | USN-3 | As a Machine Learning Engineer, I want to Fine-tune the model by adjusting hyperparameters, including learning rate and batch size, to improve its predictive accuracy.<br>Implement techniques like dropout and batch normalization to prevent overfitting and improve generalization. | 15 | High | Harshith Maride |
| Sprint-7,8 | Model Deployment and Integration | USN-4 | As a Full Stack Developer, I want to Develop an API or web interface to enable users to submit images for disease prediction.<br>Deploy the model on a cloud or server environment to make it accessible for real-time predictions. | 10 | Medium | Dushyanth Narendra, Bhavya Sri Duggina |
| Sprint-9,10 | Testing and Evaluation | USN-5 | As a Assurance Specialist, I want to test the track the model's performance, including prediction accuracy and response times.<br>Set up automated alerting and error-handling mechanisms to address issues in real-time. | 8 | Medium | Dharani Harshitha, Dushyanth Narendra, Bhavya Sri Duggina, Harshith Maride |

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

| 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-1,2 | 15 | 1 Days | 20 Oct 2023 | 20 Oct 2023 | 15 | 20 Oct 2023 |
| Sprint-3,4 | 16 | 4 Days | 21 Oct 2023 | 24 Oct 2023 | 16 | 24 Oct 2023 |
| Sprint-5,6 | 15 | 2 Days | 24 Oct 2023 | 25 Oct 2023 | 15 | 25 Oct 2023 |
| Sprint-7,8 | 10 | 3 Days | 25 Oct 2023 | 27 Oct 2023 | 10 | 27 Oct 2023 |
| Sprint-9,10 | 8 | 1 Day | 27 Oct 2023 | 27 Oct 2023 | 8 | 27 Oct 2023 |
| | | | | | | |
| | | | | | | |
| | | | | | | |

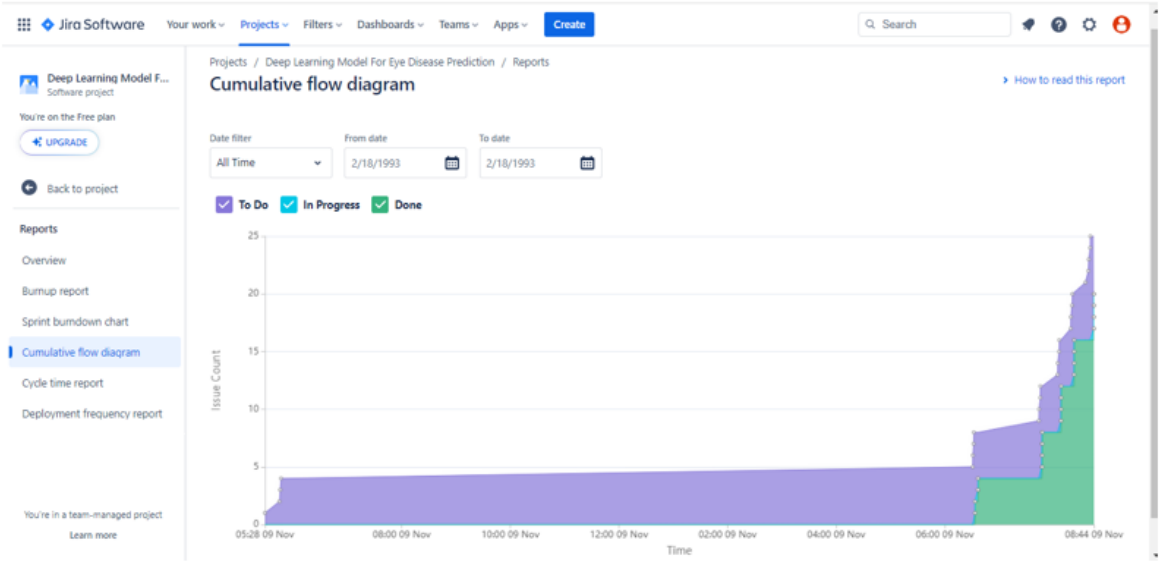
**Velocity:**

The team velocity of the 11-day sprint duration is:

Velocity = (15+16+15+10+8)/5 = 12.8

Average Velocity = Sprint Duration/Velocity =11/12.8 =0.86

**CUMULATIVE FLOW DIAGRAM:**



# CODING & SOLUTIONING

The perpetration stage of any design is a true display of the defining moments that make a design a success or a failure. The installation and operationalization of the system or system variations in a product terrain is appertained to as the perpetration step. After the system has been tried out and approved by the stoner, the phase is started. This phase continues until the system is operating in product in agreement with the defined stoner conditions.

**Language / Technology Used**

Python is the language used for developing the detection and process of the system and for image processing using Transfer Learning Techniques like VGG, Inception etc.

**Libraries / Algorithms Used**

**NUMPY**

NumPy is a python library used for working with arrays. It also has functions for working in the sphere of direct algebra, fourier transfigure, and matrices. NumPy was created in 2005 by Travis Oliphant. It's an open source design and you can use it freely. NumPy stands for Numerical Python. In Python we've lists that serve the purpose of arrays, but they're slow to reuse. NumPy aims to give an array object that's over to 50x faster than traditional Python lists. At the core of the NumPy package, is the array object. This encapsulates n- dimensional arrays of homogeneous data types, with numerous operations being performed in collected law for performance. There are several important differences between NumPy arrays and the standard Python sequences

**SCIKIT LEARN**

Scikit- learn(Sklearn) is the most useful and robust library for machine literacy in Python. It provides a selection of effective tools for machine literacy and statistical modeling including bracket, retrogression, clustering and dimensionality reduction via a harmonious interface in Python. This library, which is largely written in Python, is erected upon NumPy, SciPy and Matplotlib.

It was firstly called scikits learn and was originally developed by David Cournapeau as a Google summer of law design in 2007. latterly, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA( French Institute for Research in Computer Science and robotization), took this design at another position and made the first public release( v0.1 beta) on 1stFeb. 2010.

**TENSORFLOW**

TensorFlow is a Python library for fast numerical computing. It was created and is maintained by Google and released under the Apache2.0 open source license. It's a foundation library that can be used to produce Deep literacy models directly or by using wrapper libraries that simplify the process erected on top of Tensor Flow.

calculation in TensorFlow is described in terms of data inflow and operations in the structure of a directed graph.

- Bumps perform calculation and have zero or further inputs and labors. Data that moves between bumps are known as tensors, which are multi-dimensional arrays of real values.

- Edges The graph defines the inflow of data, branching, looping and updates to state. Special edges can be used to attend geste within the graph, for illustration staying for calculation on a number of inputs to complete.

- Operation An operation is a named abstract calculation which can take input attributes and produce affair attributes. For illustration, you could define an add or multiply operation.

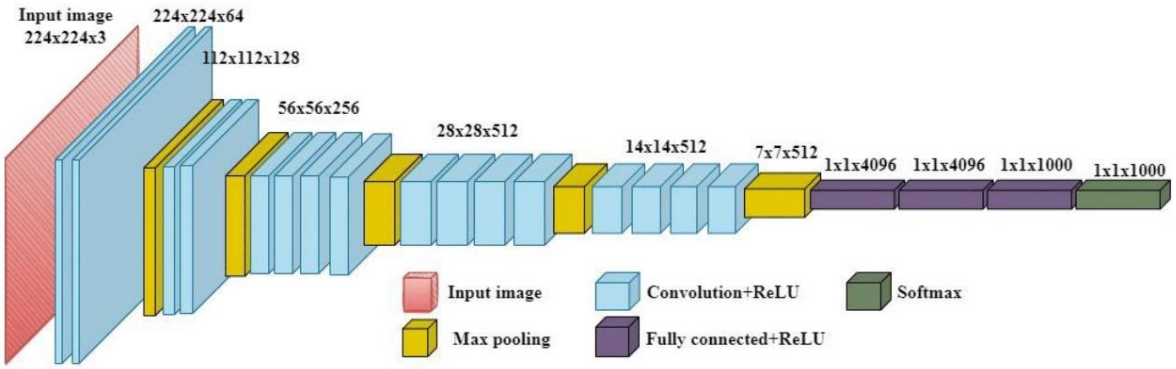
**KERAS**

Keras is one of the most popular Python libraries for Deep literacy. Keras is simple, flexible and important. It's a library that provides you colorful tools to deal with neural network models. These tools enable you to fantasize and understand the model. These represent the factual neural network model. These models group layers into objects. There are two types of Models available in Keras The successional model and the Functional model.

Keras Sequential Model is simple and easy to use model. It's a direct mound of styles that groups a direct mound of layers into atf.keras.Model. According to its name, its main task is to arrange the layers of the Keras in successional order. In this model, the data inflow from one subcaste to another subcaste. The inflow of data is continued until the data reaches the final subcaste. utmost of the Neural Networks use the successional API Model

**VGG19**

VGG-19, short for the Visual Geometry Group 19-layer network, is a renowned deep convolutional neural network architecture used in computer vision tasks. It is a part of the VGG family, developed by the Visual Geometry Group at the University of Oxford. VGG-19 is known for its simplicity and effectiveness, consisting of 19 layers with trainable parameters, predominantly using 3x3 convolutional filters. This architecture has been widely employed in image classification, object recognition, and feature extraction tasks, offering a strong baseline for various deep learning applications.



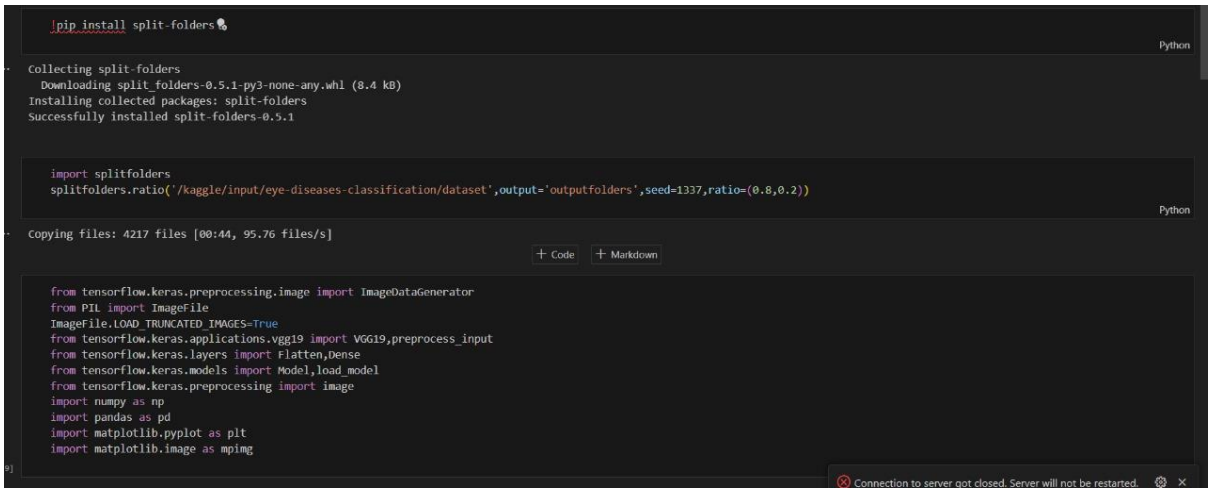

**Configurations:**

The ConvNet configurations are outlined in figure **. The nets are appertained to their names(A-E). All configurations follow the general design present in armature and differ only in the depth from 11 weight layers in the network A(8 conv. and 3 FC layers) to 19 weight layers in the network E( 16 conv. and 3 FC layers). The range of conv. layers the number of channels) is rather small, starting from 64 in the first subcaste and also adding by a factor of 2 after each maximum-pooling subcaste, until it reaches 512.

| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

## Sample Code:

## IMPORTING DEPENDENCIES:



```
!pip install split-folders
```

```
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
```

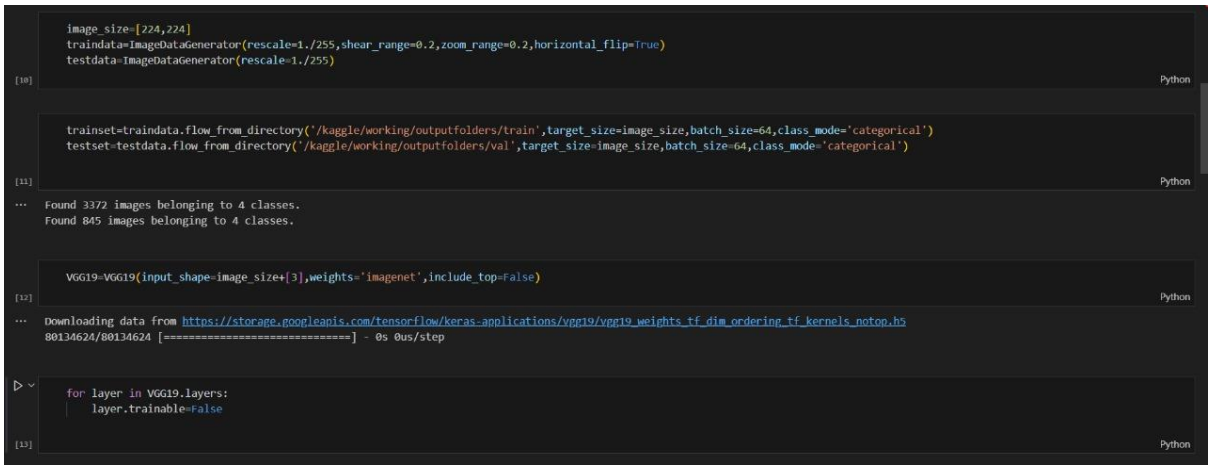
```
import splitfolders
splitfolders.ratio('/kaggle/input/eye-diseases-classification/dataset',output='outputfolders',seed=1337,ratio=(0.8,0.2))
```

```
Copying files: 4217 files [00:44, 95.76 files/s]
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES=True
from tensorflow.keras.applications.vgg19 import VGG19,preprocess_input
from tensorflow.keras.layers import Flatten,Dense
from tensorflow.keras.models import Model,load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

## Data Collection and analysis:

Setting up the file paths of the training, and validation data.Creating batches of data using the ImageDataGenerator



```
image_size=[224,224]
traindata=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
testdata=ImageDataGenerator(rescale=1./255)
```

```
trainset=traindata.flow_from_directory('/kaggle/working/outputfolders/train',target_size=image_size,batch_size=64,class_mode='categorical')
testset=testdata.flow_from_directory('/kaggle/working/outputfolders/val',target_size=image_size,batch_size=64,class_mode='categorical')
```

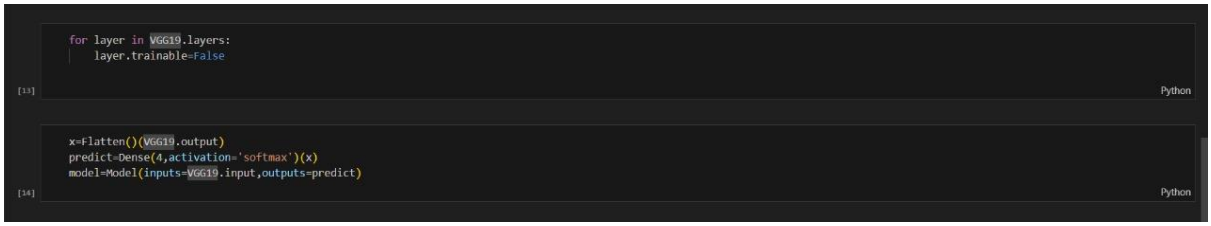
```
Found 3372 images belonging to 4 classes.
Found 845 images belonging to 4 classes.
```

```
VGG19=VGG19(input_shape=image_size+[3],weights='imagenet',include_top=False)
```

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

```
for layer in VGG19.layers:
    layer.trainable=False
```

Loading the pre-Trained VGG19 model provided by keras



```
for layer in VGG19.layers:
    layer.trainable=False
```

```
x=Flatten()(VGG19.output)
predict=Dense(4,activation='softmax')(x)
model=Model(inputs=VGG19.input,outputs=predict)
```

# Compiling the model

```
[15]  model.summary()
```

Python

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (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_conv4 (Conv2D)       (None, 56, 56, 256)       590080
...
Total params: 20,124,740
Trainable params: 100,356
Non-trainable params: 20,024,384
```

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

```
[17]  history=model.fit(trainset,validation_data=testset,epochs=50,steps_per_epoch=len(trainset),validation_steps=len(testset))
```

Python

```
Epoch 1/50
53/53 [==============================] - 91s 1s/step - loss: 1.0787 - accuracy: 0.6189 - val_loss: 0.6005 - val_accuracy: 0.7645
Epoch 2/50
53/53 [==============================] - 69s 1s/step - loss: 0.5714 - accuracy: 0.7767 - val_loss: 0.6958 - val_accuracy: 0.6994
Epoch 3/50
53/53 [==============================] - 68s 1s/step - loss: 0.5256 - accuracy: 0.7933 - val_loss: 0.5036 - val_accuracy: 0.8213
Epoch 4/50
53/53 [==============================] - 67s 1s/step - loss: 0.4917 - accuracy: 0.7951 - val_loss: 0.6498 - val_accuracy: 0.7408
Epoch 5/50
53/53 [==============================] - 67s 1s/step - loss: 0.4712 - accuracy: 0.8149 - val_loss: 0.5934 - val_accuracy: 0.7645
Epoch 6/50
53/53 [==============================] - 67s 1s/step - loss: 0.4310 - accuracy: 0.8301 - val_loss: 0.5497 - val_accuracy: 0.7751
Epoch 7/50
53/53 [==============================] - 67s 1s/step - loss: 0.4203 - accuracy: 0.8351 - val_loss: 0.4655 - val_accuracy: 0.8189
Epoch 8/50
53/53 [==============================] - 66s 1s/step - loss: 0.5172 - accuracy: 0.7963 - val_loss: 0.5633 - val_accuracy: 0.7728
Epoch 9/50
53/53 [==============================] - 66s 1s/step - loss: 0.4127 - accuracy: 0.8324 - val_loss: 0.4334 - val_accuracy: 0.8402
Epoch 10/50
53/53 [==============================] - 67s 1s/step - loss: 0.4135 - accuracy: 0.8363 - val_loss: 0.4537 - val_accuracy: 0.8462
Epoch 11/50
53/53 [==============================] - 67s 1s/step - loss: 0.4090 - accuracy: 0.8405 - val_loss: 0.4435 - val_accuracy: 0.8260
Epoch 12/50
53/53 [==============================] - 67s 1s/step - loss: 0.3721 - accuracy: 0.8493 - val_loss: 0.5505 - val_accuracy: 0.7929
Epoch 13/50
...
Epoch 49/50
53/53 [==============================] - 66s 1s/step - loss: 0.2654 - accuracy: 0.8959 - val_loss: 0.3568 - val_accuracy: 0.8698
Epoch 50/50
53/53 [==============================] - 66s 1s/step - loss: 0.2505 - accuracy: 0.8983 - val_loss: 0.5416 - val_accuracy: 0.8130
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
[18]  model.save('model.h5')
```

Python

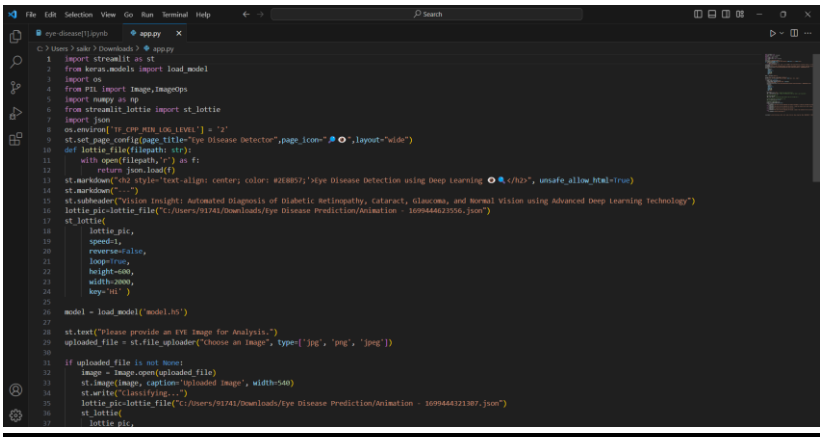
# RESULTS

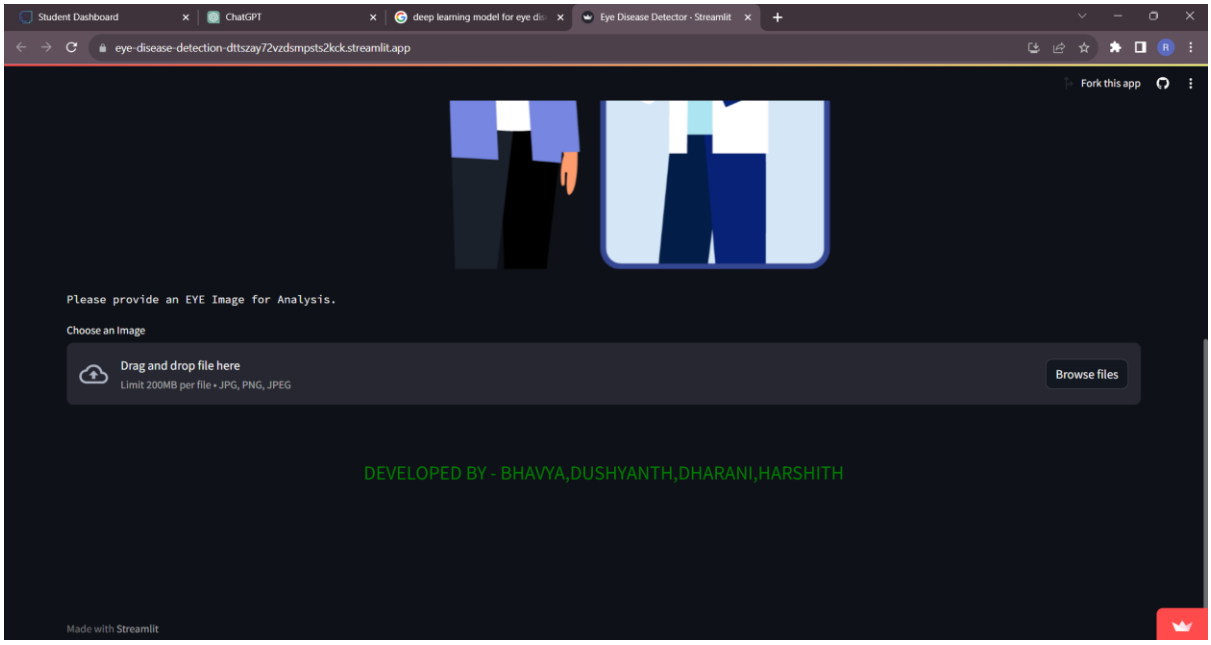
```
    img=image.load_img(r'/kaggle/working/outputfolders/val/normal/3208_right.jpg',target_size=image_size)
    x=image.img_to_array(img)
    x=np.expand_dims(x,axis=0)
    pred=model.predict(x)
    predict=np.argmax(pred,axis=1)
    index=['diabetic_retinopathy','cataract','glaucoma','normal']
    result=str(index[predict[0]])
[26]
...  1/1 [==============================] - 0s 21ms/step

    result
[27]
...  'normal'
```

# DEPLOYMENT:

In the final phase of the project, the developed automated prediction model for Diabetic Retinopathy, powered by Convolutional Neural Networks (CNN), was ready for deployment. To make this invaluable tool accessible to healthcare professionals and patients alike, we opted for the user-friendly and interactive platform offered by Streamlit. Streamlit provided an efficient and elegant means of deploying the model, allowing users to upload retinal images effortlessly and receive real-time predictions.

The model, saved in the .h5 format, was seamlessly integrated into the Streamlit application. This format, known for its compatibility with deep learning models, ensured that the model's architecture, weights, and training history were preserved during deployment. Streamlit's simplicity and flexibility enabled the creation of an intuitive user interface, where users could easily upload their retinal images, receive immediate predictions, and access interpretability tools to understand the model's decisions.




- LINK- https://eye-disease-detection-dttszay72vzdsmpsts2kck.streamlit.app/

# ADVANTAGES & DISADVANTAGES

## Advantages:

1. **Improved Efficiency**: Transfer learning allows you to leverage pre-trained models on large non-medical datasets, which have already learned valuable features. This significantly reduces the amount of data and training time required, making it more efficient and cost-effective for developing robust eye disease prediction models.

2. **Enhanced Accuracy**: Pre-trained models capture high-level features in images that are often common across different domains, including medical images. This results in improved prediction accuracy, especially when dealing with limited medical datasets.

3. **Reduced Overfitting**: Transfer learning helps mitigate overfitting, as the pre-trained models already have generalization capabilities. Fine-tuning these models with medical data helps adapt them to specific eye disease prediction tasks while maintaining their ability to generalize.

4. **Faster Deployment**: The use of transfer learning can speed up model development and deployment, making it more practical for real-world clinical applications, where timely diagnosis and treatment are crucial.

5. **Access to State-of-the-Art Architectures**: Transfer learning allows access to advanced deep learning architectures developed by the research community, which can be challenging to develop from scratch, especially for those without extensive machine learning expertise.

## Disadvantages:

1. **Limited Adaptability**: Pre-trained models may not always align perfectly with the specific characteristics of medical images, and fine-tuning can be challenging. Some features from the source domain may not be relevant to the target domain, which could hinder the model's performance.

2. **Domain Shift:** There can be a domain shift between the source (non-medical) and target (medical) domains, leading to a potential loss of information and reduced performance. Addressing this domain shift may require additional data collection and preprocessing.

3. **Lack of Interpretability**: Transfer learning models may be complex and challenging to interpret, making it difficult for healthcare professionals to understand why a particular prediction was made. This lack of transparency can be a barrier to acceptance in clinical settings.

4. **Data Privacy Concerns**: Using pre-trained models may involve sharing sensitive medical data with third-party organizations that developed the source models, potentially raising privacy and security concerns.

5. **Dependency on Pre-trained Models**: Transfer learning relies on the availability of suitable pre-trained models, and not all medical imaging tasks have relevant pre-trained models readily accessible. Developing custom pre-trained models can be time-consuming and resource-intensive.

# CONCLUSION

In conclusion, the utilization of transfer learning in deep learning models for eye disease prediction represents a promising and transformative approach in the field of ophthalmology and healthcare. By capitalizing on the knowledge encoded in pre-trained models, the accuracy and efficiency of eye disease diagnosis have been significantly enhanced, offering the potential to improve patient outcomes and reduce the burden of preventable vision impairment. These models have demonstrated their ability to make rapid and precise predictions, providing valuable support to healthcare professionals in their diagnostic and treatment decision-making processes.

However, it is important to acknowledge that the successful implementation of transfer learning in eye disease prediction models comes with its share of challenges. The adaptation of pre-trained models to the medical domain may not always be straightforward, and careful consideration of domain shift and data privacy concerns is essential. Ensuring model transparency and interpretability is also a critical aspect that needs further attention to gain trust and acceptance within the clinical community.

Looking ahead, the continued advancement of transfer learning techniques, as well as collaborative efforts between machine learning experts, ophthalmologists, and healthcare organizations, holds the potential to further refine and customize deep learning models for eye disease prediction. With ongoing research and development, these models can be integrated into routine clinical practice, enabling early diagnosis, personalized treatment planning, and the prevention of vision loss on a global scale.

# FUTURE SCOPE

**Improved Accuracy and Robustness:** Future advancements will focus on enhancing the accuracy and robustness of Diabetic Retinopathy classification models. This includes refining the model's ability to detect subtle changes in retinal images and reducing false positives and false negatives. Advanced deep learning architectures and novel data augmentation techniques will contribute to improved performance.

**Interpretable AI**: As the adoption of artificial intelligence (AI) in healthcare grows, there will be a greater emphasis on making AI models, including those for Diabetic Retinopathy, more interpretable. Explainable AI techniques will help healthcare professionals better understand the model's decisions, leading to increased trust and adoption.

**Personalized Medicine:** The future holds the potential for personalized treatment plans based on an individual's risk profile for Diabetic Retinopathy. Machine learning models can be used to tailor interventions and follow-up schedules, optimizing healthcare resources and improving patient outcomes.

**Telemedicine and Remote Screening:** Telemedicine and remote screening programs will expand, driven by the accessibility and scalability of Diabetic Retinopathy classification models. Patients in remote or underserved areas will benefit from access to retinal screening without the need for in-person visits.

**Multi-Modal Imaging:** Integrating multiple imaging modalities, such as optical coherence tomography (OCT) and fundus photography, will provide a more comprehensive view of retinal health. Future models will combine these modalities for a more accurate diagnosis.

**Real-time Monitoring:** Continuous monitoring of retinal health will become more feasible, allowing for early detection of changes and immediate intervention. Wearable devices and smartphone applications may play a role in this real-time monitoring.

**Global Adoption:** The global adoption of Diabetic Retinopathy classification models will lead to standardized screening and diagnostic processes. This harmonization of healthcare practices will benefit patients worldwide.

# APPENDIX:

Demo Video Link:

https://drive.google.com/file/d/15nyI6fa-DyKkpPMsgZYXSb-4f-Ozj-8_/view?usp=sharing