Eye Disease Detection Using Deep Learning

Introduction

This project focuses on detecting eye diseases using deep learning techniques. By leveraging Convolutional Neural Networks (CNNs), the model can identify and classify various eye diseases such as diabetic retinopathy, cataracts, and glaucoma. The system uses medical image datasets and applies state-of-the-art image processing techniques to enhance detection accuracy. The goal is to provide an automated and efficient screening tool for early diagnosis and treatment.

Use Cases

Scenario 1 - Hospital Eye Screening Automation

Hospitals can incorporate this AI model into their screening procedure to diagnose eye diseases at an early stage. It helps ophthalmologists by targeting high-risk cases, lowering manual workload. This facilitates early diagnosis and treatment, enhancing patient care. The system helps maintain effective resource utilization in crowded medical centers.

Scenario 2 - Remote Healthcare Assistance

Telemedicine platforms can utilize this system for remote detection of eye diseases. Patients from rural or underserved regions may upload eye images for preliminary AI-based screening. This saves patients from repeated hospital visits, offering quicker and convenient healthcare. Physicians can remotely evaluate cases and advise accordingly.

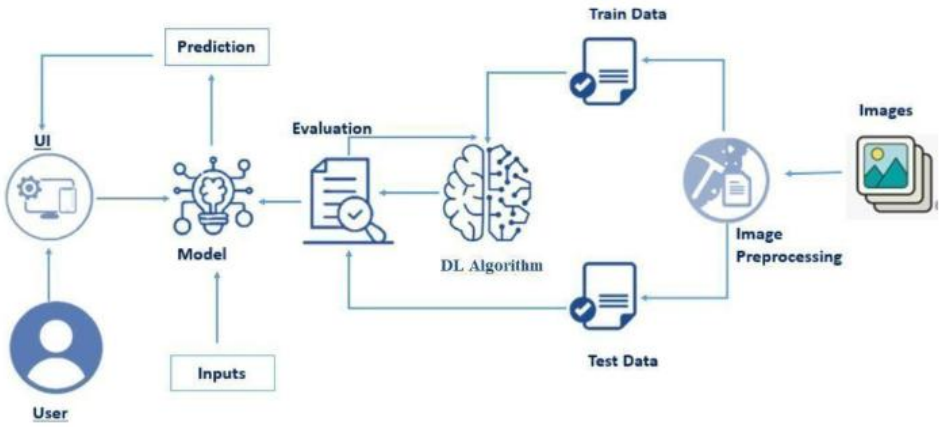
Scenario 3 - Research and Development

Scientists can use this model to analyze disease trends and risk factors. AI-based analysis enables monitoring of prevalence trends among various groups. This facilitates research on environmental and genetic factors affecting eye diseases. The results can be used to develop better treatment options and preventive care.

Prerequisites

We have to install all the necessary requisites to develop the model which detects the eye disease detection. This model will detect whether the eye has any of the diseases like cataract, Diabetic Retinopathy, Glaucoma. Here we are developing the model and as well as we are redirecting the result to the web page. To develop the deep learning application by using flask. We have to install the libraries or the packages into the visual studio like NumPy, Pandas, Matplotlib, Keras, TensorFlow, Flask. By using these packages, we can easily develop the model and analysis the model. We have to install the Anaconda Navigator so that we can use the python libraries very easily and also helpful to split the data into testing and training parts. By using the Jupiter notebook, the splitting of data would be very effective.

Technical Architecture

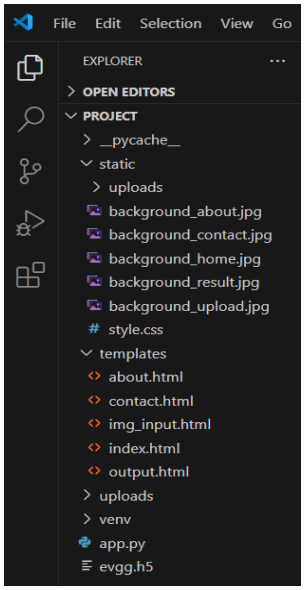


Project Flow

To accomplish this, we have to complete all the activities listed below,

- Data Collection & Extraction from Database
- Data Preparation
- Web Page designing
- Module Development in Google Colab
- Performance Testing
- Project Demonstration & Documentation
    - Record explanation Video for project end to end solution
    - Project Documentation-Step by step project development procedure

Project Structure



Dataset Details

The dataset of the eye disease detection would have 4 categories like normal, Diabetic Retinopathy, Cataract, Glaucoma and by using this we can train the deep learning module and also test the module whether the eye has any of the above diseases or not.

Dataset Source: Kaggle(https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification)
Classes: Normal, Diabetic Retinopathy, Cataract, Glaucoma
No. of Images: ~10,000 annotated images
Image Size: 224x224 pixels (preprocessed)

Deep Learning Model Details

Architecture: CNN
Input Shape: (224, 224, 3)
Framework: TensorFlow/Keras or PyTorch
Training: 50 epochs, batch size 32

Project Demonstration & Documentation

1. Data Collection & Preprocessing
2. Model Development & Training
3. Web Browser Building
4. Flask Application Development
5. Performance Testing & Output
6. Final Documentation &Conclusion

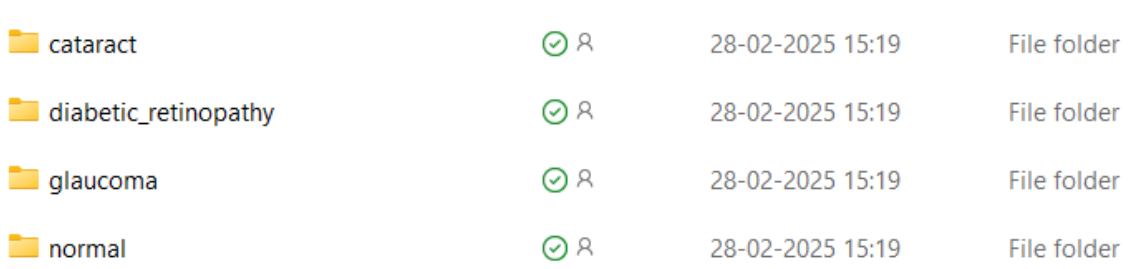
1.Data Collection and preprocessing

Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes and generate insights from the data.

Data Collection

- Collect the data
  We need to collect the database set from the Kaggle platform where there would be numerous datasets used for the modules development.
  Dataset:- https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification

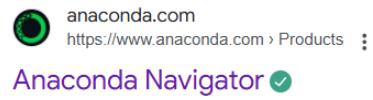
- In this dataset there would be 4 categories like Normal, Glaucoma, Cataract, Diabetic Retinopathy



- We have to save the path where we have downloaded the zip file and then we have to extract all the data to another folder.

Data Preprocessing

- After we have downloaded the file, we have the process the dataset and then we have to split the data into the testing data and training data.
- For the splitting the data we have to install the **Anaconda Navigator** from the browser.



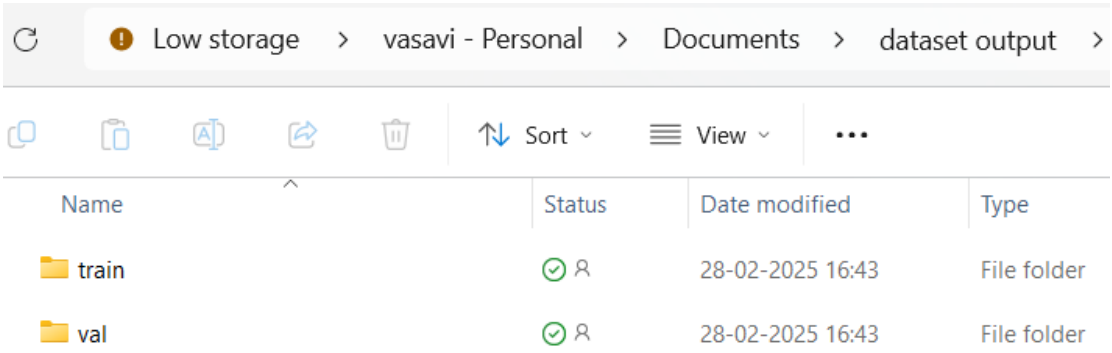
- After that we have to sign in to the anaconda cloud and then launch the **Jupiter notebook** and write the code for the splitting of the data.

```
import split-folders

# Define the input and output directories
input_folder = r'C:\Users\VASAVI\OneDrive\Documents\internship dataset\dataset'  #
output_folder = r'C:\Users\VASAVI\OneDrive\Documents\internship dataset\output'  #

# Split the dataset into 80% training and 20% testing
splitfolders.ratio(input_folder, output=output_folder, seed=1337, ratio=(0.8, 0.2))
```

- For the splitting of the data purpose, we use the **split-folder** library for this we have use the command as **pip install split-folders** in the anaconda command prompt

```
(base) C:\Users\VASAVI>pip install split-folders
Requirement already satisfied: split-folders in c:\users\vasavi\anaconda\lib\site-packages (0.5.1)
```

- After splitting the output is saved to another folder and then have two different folders as **train folder** and **test folder**



2. Model Development & Training

Model Development

For the module development we use the google colab. After the developing of the model, we have to save and download with the extension of the **filename.h5**

- First, we have to mount the dataset from the drive to the google colab.

```
from google.colab import drive
import os

# Mount Google Drive
drive.mount('/content/drive')

# Define dataset path (update this path based on your Google Drive location)
dataset_path = "/content/drive/MyDrive/Eye_Disease_Dataset"

# Check if dataset is correctly mounted
print("Dataset files:", os.listdir(dataset_path))
```

- Here it gives the output as the folders that have in the **Eye_Disease_Dataset**

```
Drive already mounted at /content/drive;
Dataset files: ['train', 'val']
```

- After that we have to load the training data, testing data, validation data and data Augmentation and Loading into the module.
- For the purpose of the splitting of the data in the module.

```
# Data Augmentation and Loading
train_datagen = ImageDataGenerator(
    rescale=1./255,   # Normalize pixel values
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2  # Split some data for validation
)
```

- We have to take the train generator for generating of the training data.

```
# Load training data
train_generator = train_datagen.flow_from_directory(
    os.path.join(dataset_path, 'train'),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'  # Use 80% for training
)
```

- We have to take the load validation generator for generating of the validation data.

```
# Load validation data
validation_generator = train_datagen.flow_from_directory(
    os.path.join(dataset_path, 'train'),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'  # Use 20% for validation
)
```

- We have to take the test generator for generating of the testing data.

```
# Load testing data
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    os.path.join(dataset_path, 'val'),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
```

- By using this we can split the images very accurately.

```
Found 2699 images belonging to 4 classes.
Found 673 images belonging to 4 classes.
Found 845 images belonging to 4 classes.
```

- We have to install all the necessary packages into the module.

```
!pip install tensorflow keras numpy pandas opencv-python matplotlib
```

- After installing we have to import all the libraries into the module

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG19
import os
```

- After that we can split the testing and training data into two classes.

```
train_dir = os.path.join(dataset_path, 'train')
test_dir = os.path.join(dataset_path, 'val')

img_size = (224, 224)
batch_size = 32

train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir, target_size=img_size, batch_size=batch_size, class_mode='categorical')

test_generator = test_datagen.flow_from_directory(
    test_dir, target_size=img_size, batch_size=batch_size, class_mode='categorical')
```

- We get the data and then split the images

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

Module Training

- By using the latest algorithms and the techniques we can train the data

```
base_model = VGG19(input_shape=(224, 224, 3), weights='imagenet', include_top=False)

for layer in base_model.layers:
    layer.trainable = False  # Freeze the pre-trained layers

model = Sequential([
    base_model,
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')  # 4 classes: Normal, Cataract, DR, Glaucoma
])

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

- We have to fit the data into the train and test generators.

```
history = model.fit(train_generator, validation_data=test_generator, epochs=10)
```

- After training the data we have to save the model as **filename.h5**

```
model.save('/content/drive/MyDrive/evgg.h5')
```

- We can check the accuracy of the model after training it.

```
loss, accuracy = model.evaluate(test_generator)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

```
27/27 ━━━━━━━━━━━━━━━━━━━━ 646s 24s/step - accuracy: 0.8293 - loss: 0.4345
Test Accuracy: 84.02%
```

- After that we can download the module into our own local disk

```
from google.colab import files
files.download('/content/evgg.h5')
```

3. Web Browser Building

For the deep learning application, we have to build the flask application so that we can upload the image and then by clicking the predict button we can easily predict the disease in the eye we have given for the testing purpose.

- For this we have to build a **home page** or the **index page** where it opens by default when we run the application.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>LIVEDOC - Eye Disease Detection</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>

<body class="home-page"> <!-- Ensuring the background image applies -->

    <!-- Header Section -->
    <header>
        <nav>
            <a href="/">Home</a>
            <a href="/inp">Predict</a>
            <a href="/about">About Us</a>
            <a href="/contact">Contact</a>
        </nav>
    </header>

    <!-- Main Content -->
    <main class="container">
        <h1>We're Determined for Your Better Life</h1>
        <p>You can get the care you need 24/7 – be it online or in person. You will be treated by caring specialist doctors.</p>

        <a href="/inp" class="btn">Make an Appointment</a>
    </main>

    <!-- Footer -->
    <footer>
        <p>© 2025 LIVEDOC. All rights reserved.</p>
    </footer>

</body>
</html>
```

- After just by clicking on the upload image button it redirects to the **image_input page** where we can choose the file we want to test.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>LIVEDOC - Predict Eye Disease</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>

<body class="home-page"> <!-- Ensuring the background image applies -->
    <header>
        <nav>
            <a href="/">Home</a>
            <a href="/inp">Predict</a>
            <a href="/about">About Us</a>
            <a href="/contact">Contact</a>
        </nav>
    </header>
    <main class="container">
        <h1>Eye Care with Top Professionals and In Budget</h1>
        <p>We've built a healthcare system that puts your needs first. For us, there is nothing more important than the health of you and your

        <!-- Image Upload Form -->
        <form action="/predict" method="post" enctype="multipart/form-data">
            <label for="image">Upload an Eye Image</label>
            <input type="file" id="image" name="image" required>
            <button type="submit" class="btn">Predict Disease</button>
        </form>
    </main>

    <!-- Footer -->
    <footer>
        <p>© 2025 LIVEDOC. All rights reserved.</p>
    </footer>

</body>
</html>
```

- Then by clicking on the predict button it gives the result and the result would be given in the **output page.**

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>LIVEDOC - Prediction Result</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>

<body class="result-page"> <!-- Apply background image specific to this page -->
    <header>
        <nav>
            <a href="/">Home</a>
            <a href="/inp">Predict</a>
            <a href="/about">About Us</a>
            <a href="/contact">Contact</a>
        </nav>
    </header>
    <main class="container">
        <h2>Prediction Result</h2>
        <p>The predicted disease is: <strong>{{ prediction }}</strong></p>

        <a href="/inp" class="btn">Try Again</a>
    </main>
    <footer>
        <p>© 2025 LIVEDOC. All rights reserved.</p>
    </footer>

</body>
</html>
```

- By clicking on the about button we would redirect to the **about page**

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Us - LIVEDOC</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body class="about-page"> <!-- Ensuring the background image applies -->
    <header>
        <nav>
            <a href="/">Home</a>
            <a href="/inp">Predict</a>
            <a href="/about">About Us</a>
            <a href="/contact">Contact</a>
        </nav>
    </header>
    <main class="container">
        <h1>About LIVEDOC</h1>
        <p>LIVEDOC is an AI-based Eye Disease Detection platform designed to help individuals detect common eye diseases like Cataract, Diabeti

        <h2>Our Mission</h2>
        <p>We aim to provide early detection of eye diseases using AI to ensure better treatment and eye care for everyone.</p>

        <h2>How It Works</h2>
        <ul>
            <li>Upload an eye image.</li>
            <li>AI model processes the image.</li>
            <li>Instant diagnosis result is displayed.</li>
        </ul>
        <h2>Meet Our Team</h2>
        <p>We are a team of AI enthusiasts and healthcare professionals committed to making eye care accessible for everyone.</p>
    </main>
    <footer>
        <p>© 2025 LIVEDOC. All rights reserved.</p>
    </footer>
</body>
```

- By clicking on the contact button, we would redirect to the **contact page**

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact - LIVEDOC</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>

<body class="contact-page"> <!-- Ensuring background image applies -->
    <header>
        <nav>
            <a href="/">Home</a>
            <a href="/inp">Predict</a>
            <a href="/about">About Us</a>
            <a href="/contact">Contact</a>
        </nav>
    </header>
    <main class="container">
        <h1>Contact Us</h1>
        <p>Email: support@eyediseaseai.com</p>
        <p>Phone: +1 234 567 890</p>
        <p>Address: 123 AI Health Street, Vision City</p>

        <a href="{{ url_for('home') }}" class="btn">Back to Home</a>
    </main>
    <footer>
        <p>© 2025 LIVEDOC. All rights reserved.</p>
    </footer>

</body>
</html>
```

By using this we can build the basic web page.

4. Flask Application Development

For this we have install the app.py where we have to write the code which is used for the working of the application.

- We have to create a virtual environment

```
PS C:\Users\VASAVI\OneDrive\Desktop\project> python -m venv venv
```

- After creating we have to activate the virtual environment

```
PS C:\Users\VASAVI\OneDrive\Desktop\project> venv\Scripts\activate
```

- After the installation completes, we have to check the version of the python and pip

```
(venv) PS C:\Users\VASAVI\OneDrive\Desktop\project> python --version
Python 3.10.8
(venv) PS C:\Users\VASAVI\OneDrive\Desktop\project> pip --version
pip 22.2.2 from C:\Users\VASAVI\OneDrive\Desktop\project\venv\lib\site-packages\pip (python 3.10)
```

- After the path is redirecting to the virtual environment, we have to install all the prerequisites.

```
>> pip install flask tensorflow keras numpy opencv-python matplotlib
```

- Write the code in the app.py where the model and the flask application would work together.
- In this we would load the trained model into the application

```
# Load the trained model
MODEL_PATH = "evgg.h5"
model = load_model(MODEL_PATH)
```

- After that we would have to write the code necessary code for preprocess and making the prediction

```
# Preprocess and make prediction
img_array = preprocess_image(file_path)
predictions = model.predict(img_array)

classes = ["Cataract", "Diabetic Retinopathy", "Glaucoma", "Normal"]

predicted_class = classes[np.argmax(predictions)]

return render_template('result.html', prediction=predicted_class, img_path=file_path)
```

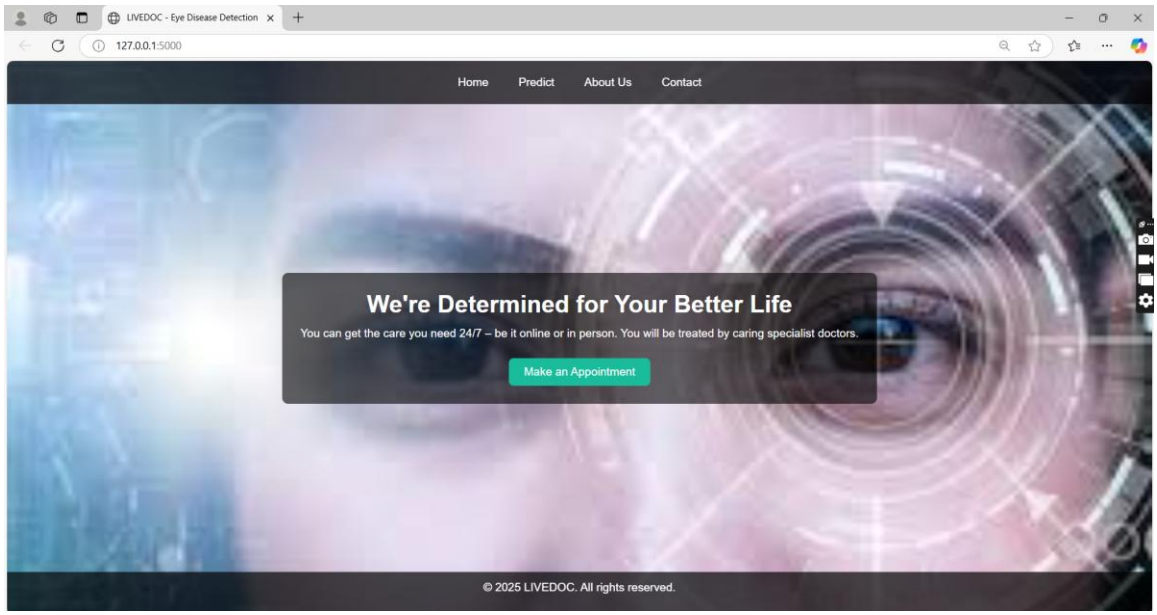
5. Performance Testing & Output

After all the completion of creation of the virtual environment and model loading in the vs code, we have to run the flask application by using the command **flask run** or **python app.py**

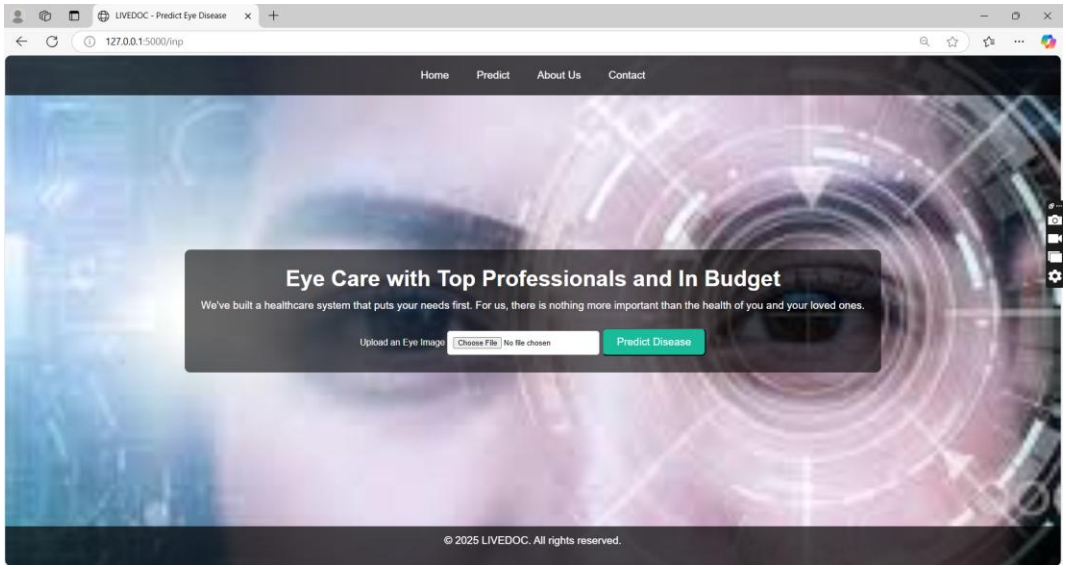
- By this it would give the browser link, so that by using that we can get the web application.

```
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
```

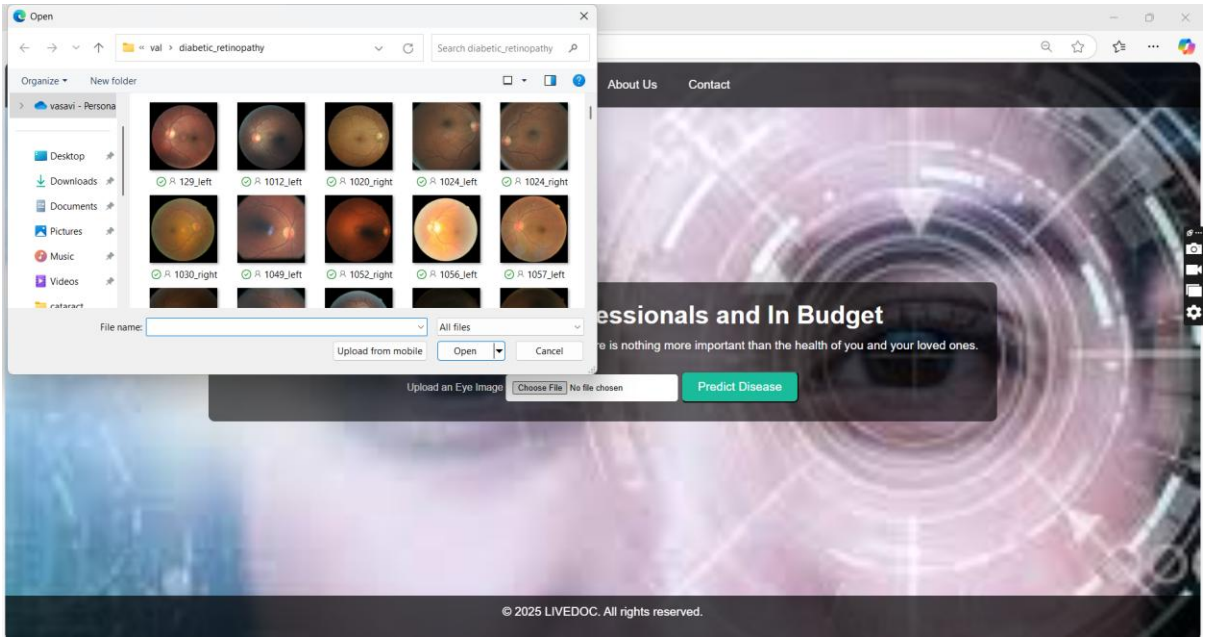
- By clicking on the link, we redirect to the browser with **home page**.



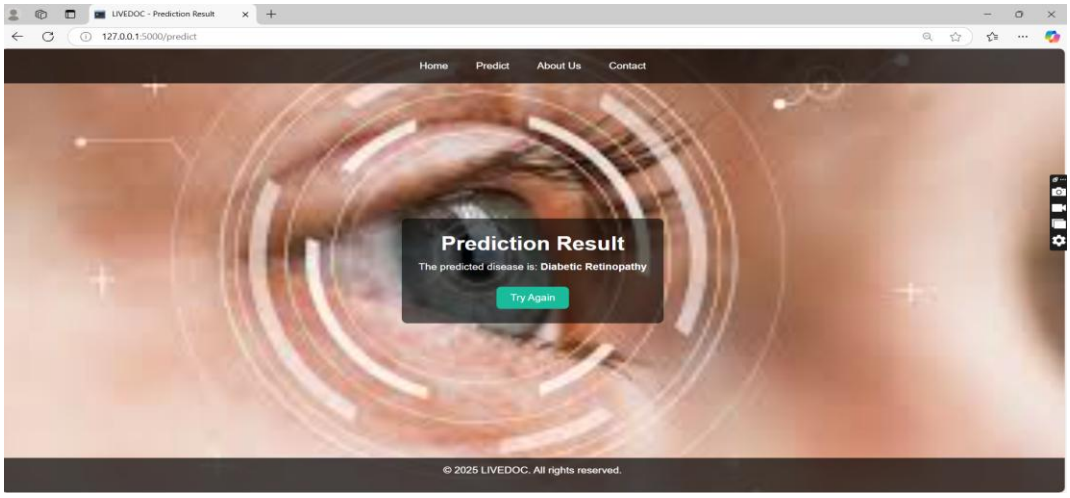
- Clicking on the **Make an Appointment** button we redirect to the **upload page.**



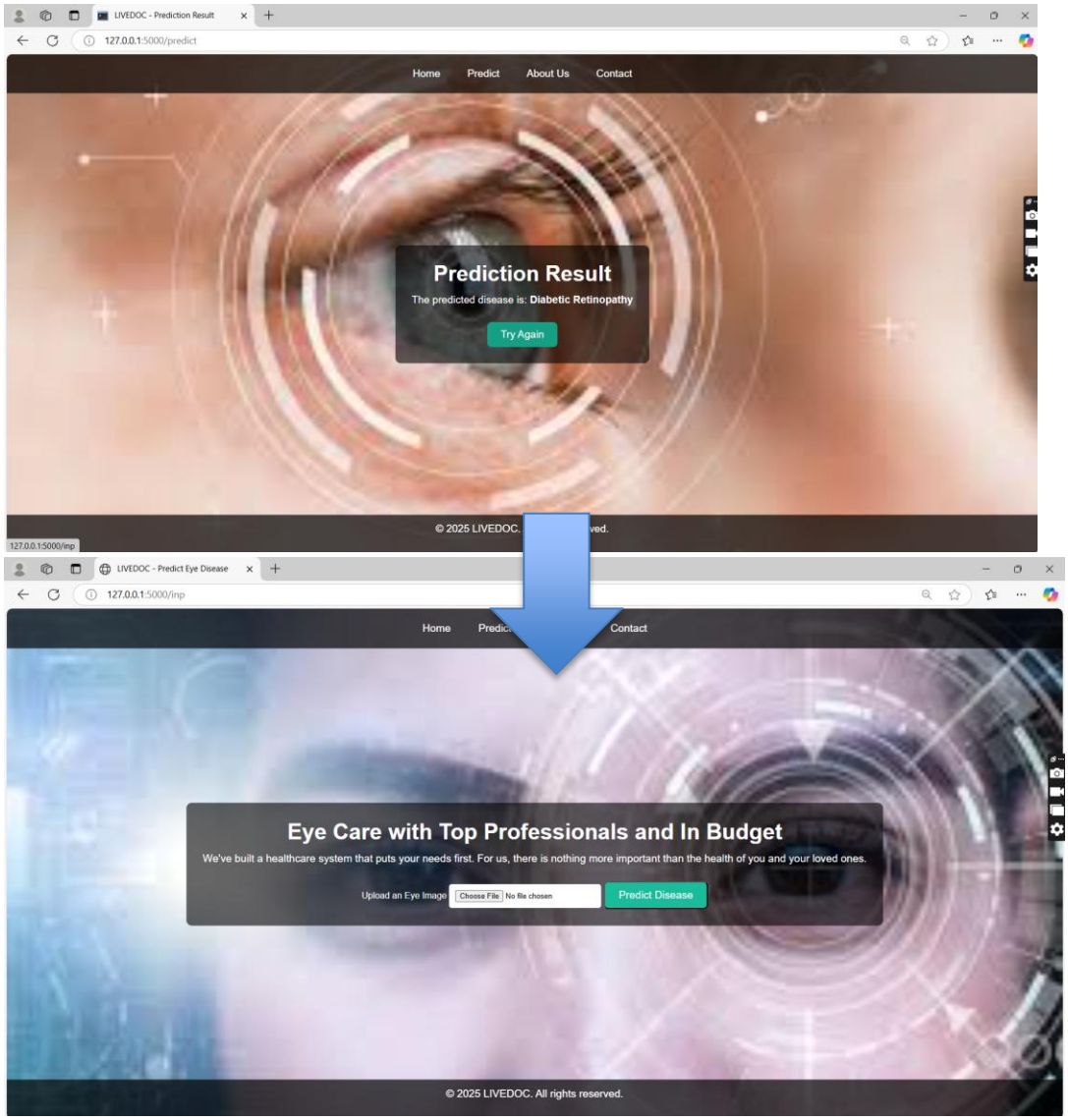
- In that page, we have to choose the file which we want to detect the disease, it would give a prompt as below.



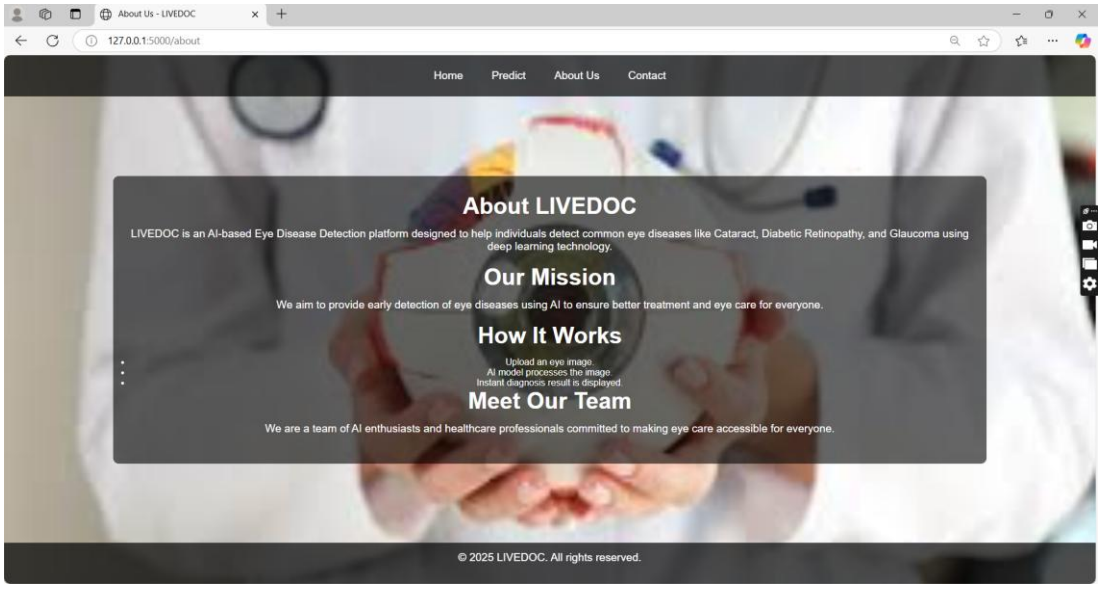
- And then by clicking the predict disease button it would give the predicted disease in the eye.



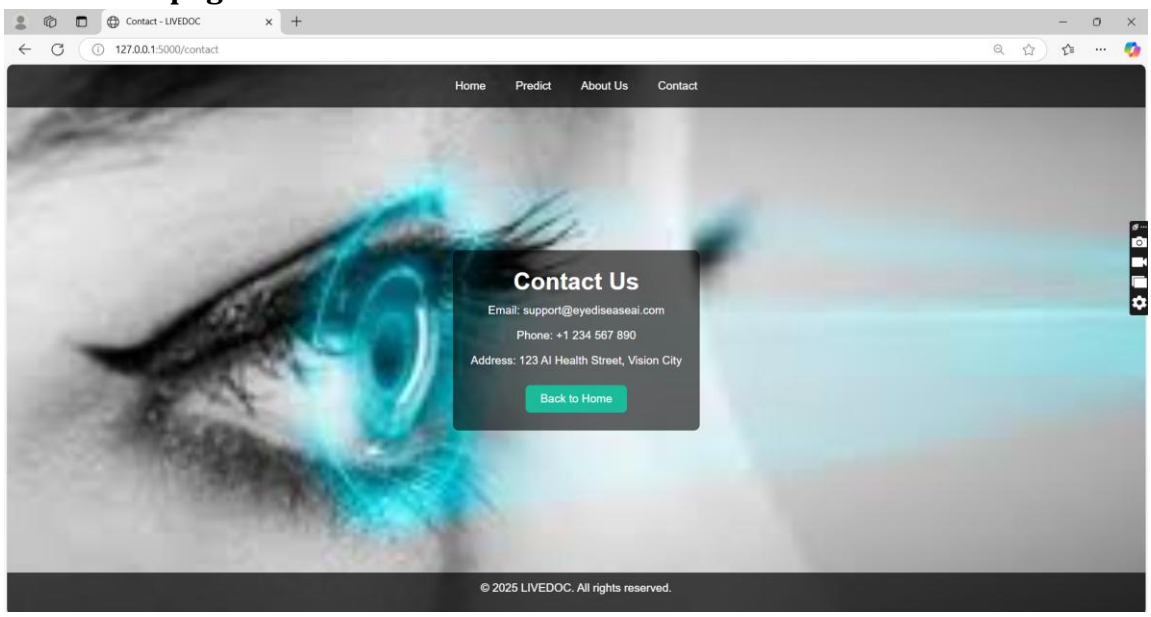
- After that if we want to retry with another image, by just clicking on the try again button we redirect to the predict page.



- If we want to know about the website by clicking on the about button we redirect to the **about page**.

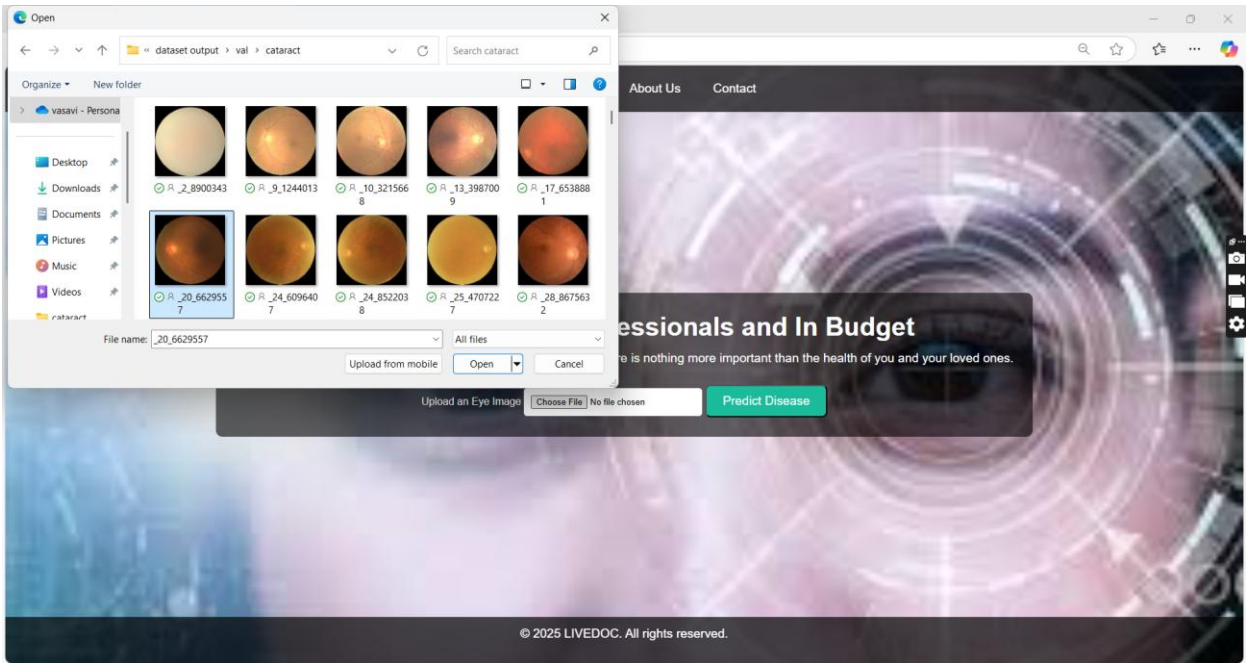


- If we want to know contact the website by clicking on the about button we redirect to the **contact page**.

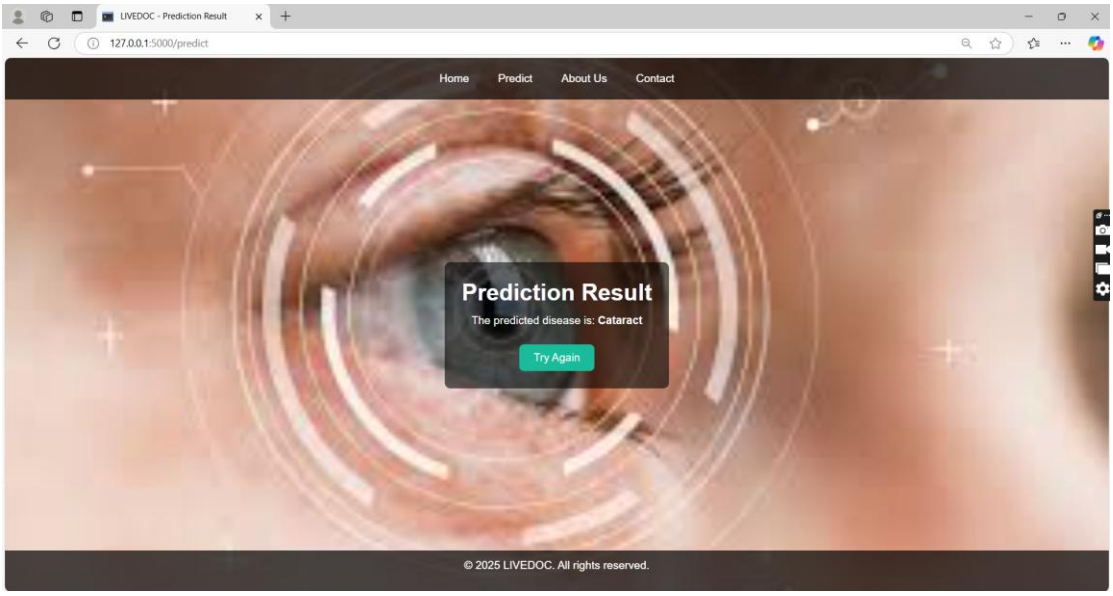


Outputs:

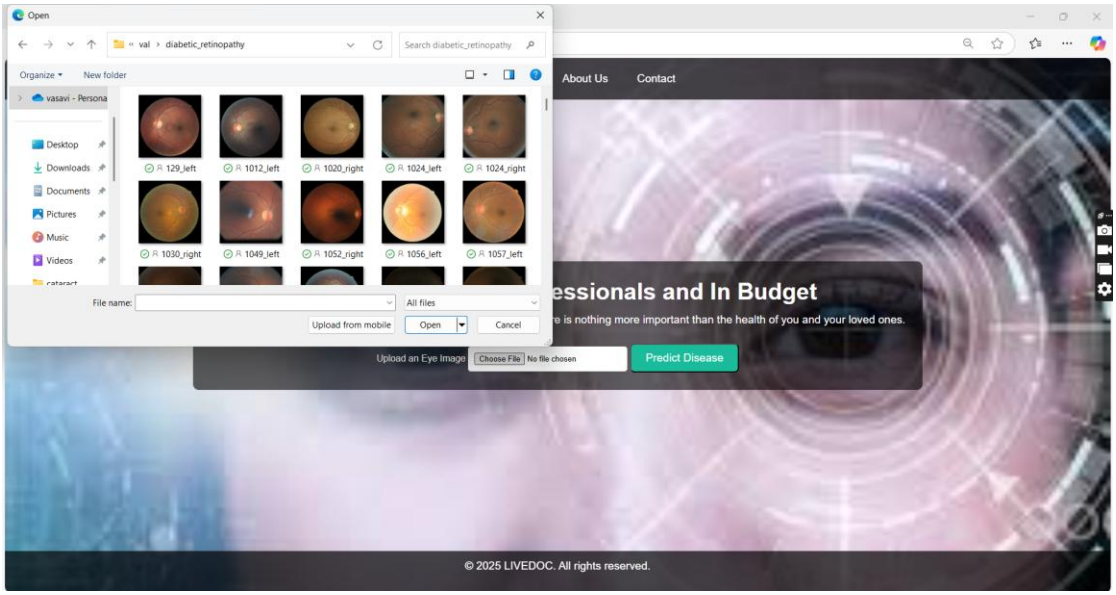
Input-1:(cataract)



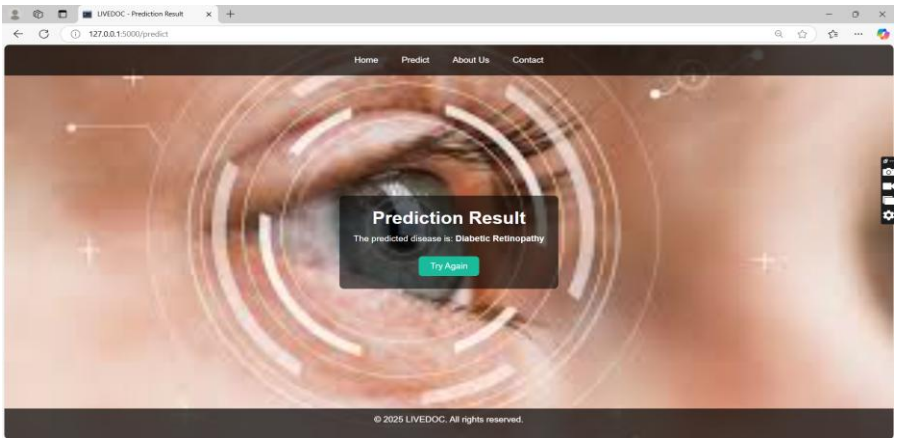
Output-1:



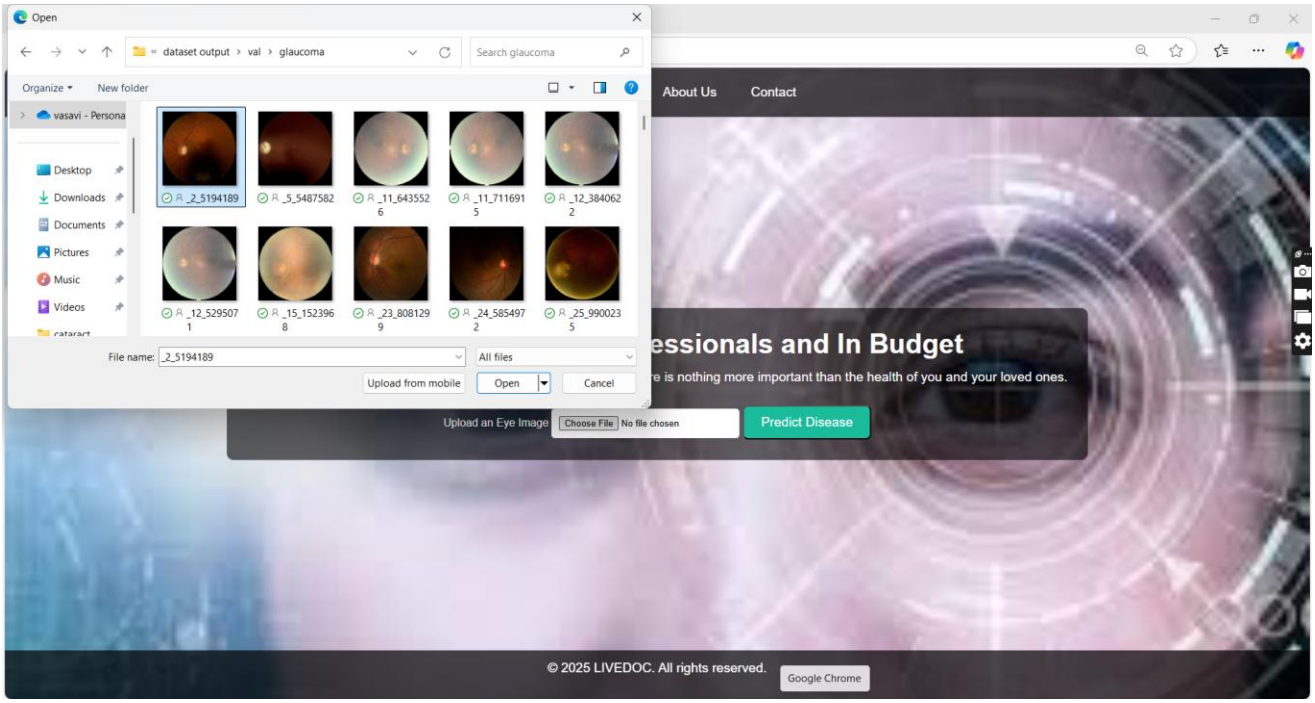
Input-2: (Diabetic-Retinopathy)



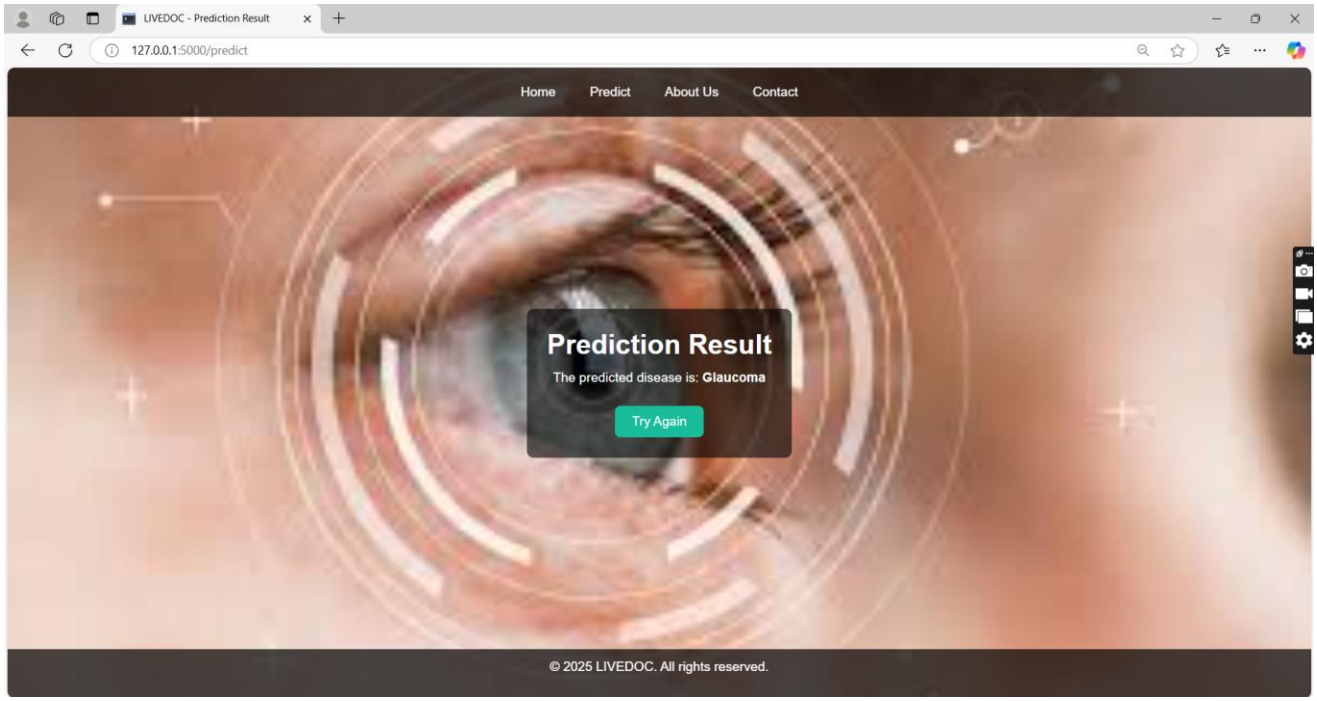
Output-2:



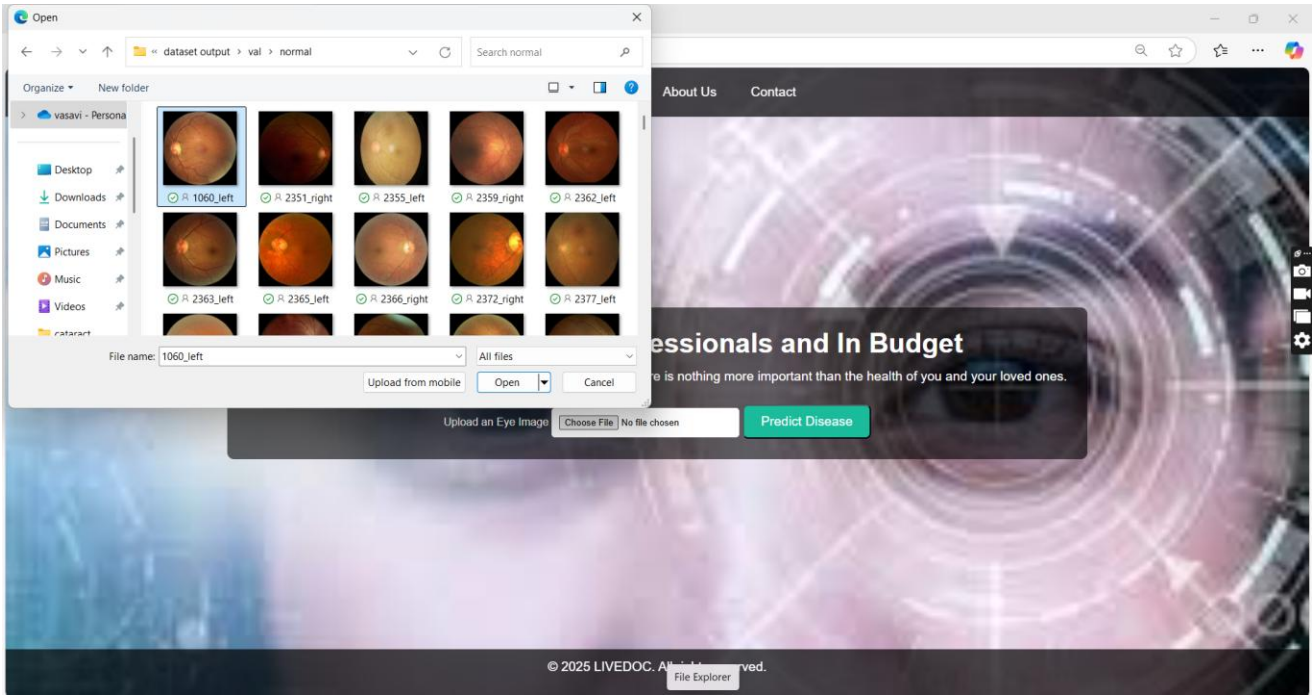
Input-3:(Glaucoma)



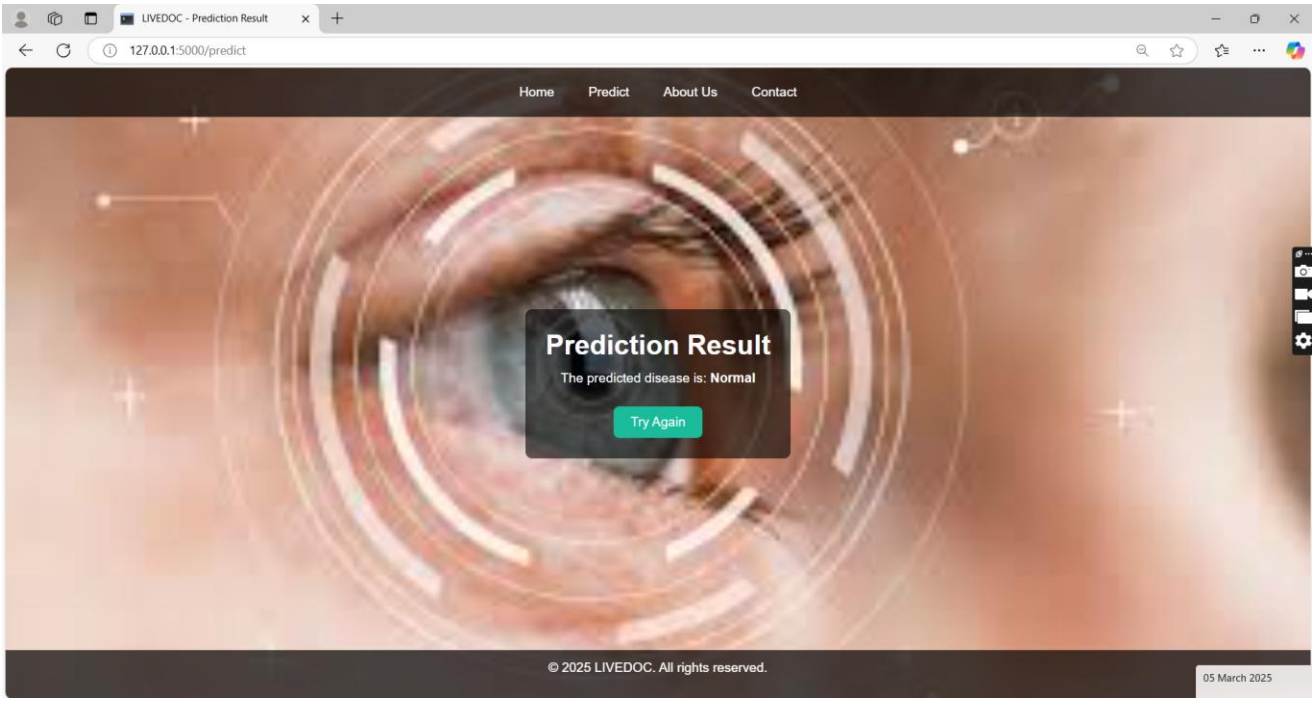
Output-3:



Input-4:(Normal)



Output-4:



This is we can predict the eye disease by using the deep learning model and the flask application.

6. Final Documentation &Conclusion

Final Documentation

This project aims to classify eye diseases such as **Cataracts, Glaucoma, Normal, and Diabetic Retinopathy** using deep learning models like **VGG19, XceptionV3, and InceptionV3.** The model processes a set of eye images using **Tensorflow and Keras** and is deployed through a **Flask web application.** The user uploads an eye image then the system can make a high accuracy prediction of the disease. The goal of the project is to help aid in early diagnosis, as well as access to ophthalmic healthcare.

Conclusion

This project provides an automated system for detecting eye diseases using deep learning. It offers significant benefits for early diagnosis, reducing the workload fort healthcare professionals and improving accessibility to eye care. Further improvements can be made by integrating more advanced architectures and larger datasets to enhance accuracy and reliability.