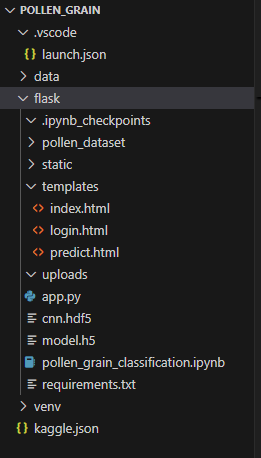
Program flow

Structure of the program:



1. Index.html :

<!DOCTYPE html>

<html>

<head>

  <title>Pollen Grain Classification</title>

  <style>

    body {

      margin: 0;

      font-family: Arial, sans-serif;

    }

    header {

      background-color: #a6ce39;

      color: white;

      padding: 20px;

      font-size: 24px;

    }

    nav {

      position: absolute;

      top: 20px;

      right: 30px;

    }

    nav a {

      color: #f06d06;

      margin-left: 20px;

      text-decoration: none;

    }

    .content {

      text-align: center;

      background-image: url('<https://images.unsplash.com/photo-1504196606672-aef5c9cefc92'>);

      background-size: cover;

      height: 90vh;

      color: white;

      padding-top: 100px;

    }

    input[type="file"] {

      margin: 10px;

    }

    button {

      padding: 10px 20px;

    }

  </style>

</head>

<body>

<header>

  A Pollen Grain Classification

  <nav>

    <a href="login.html">About</a>

    <a href="#">Contact</a>

  </nav>

</header>

<div class="content">

  <div>

    <img src="<https://icons.iconarchive.com/icons/icons8/ios7/512/Files-Upload-icon.png>" width="100">

  </div>

  <input type="file">

  <button>Submit</button>

  <h2>The pollen grain type is arecaceae</h2>

</div>

</body>

</html>

1. Login.html :

<!DOCTYPE html>

<html>

<head>

  <title>About - Pollen Grain Classification</title>

  <style>

    body {

      margin: 0;

      font-family: Arial, sans-serif;

    }

    header {

      background-color: #a6ce39;

      color: white;

      padding: 20px;

      font-size: 24px;

    }

    nav {

      position: absolute;

      top: 20px;

      right: 30px;

    }

    nav a {

      color: #f06d06;

      margin-left: 20px;

      text-decoration: none;

    }

    .hero {

      background-image: url('<https://images.unsplash.com/photo-1470115636492-6d2b56f9146f'>);

      background-size: cover;

      background-position: center;

      height: 80vh;

      display: flex;

      flex-direction: column;

      justify-content: center;

      align-items: center;

      color: white;

      text-align: center;

    }

    .hero h1 {

      font-size: 48px;

      margin: 0;

    }

    .hero p {

      font-size: 18px;

      max-width: 700px;

    }

    .about-section {

      text-align: center;

      margin: 40px 20px;

    }

  </style>

</head>

<body>

<header>

  Pollen Grain Classification

  <nav>

    <a href="index.html">Home</a>

    <a href="#">Contact</a>

    <a href="#">Predict</a>

  </nav>

</header>

<div class="hero">

  <h1>Pollen Grain Prediction</h1>

  <p>We assist researchers and scientists in the field of botany and plant sciences by providing interface for users to upload images of pollen grains and give predictions on the type of pollen present in the image.</p>

</div>

<div class="about-section">

  <h2>ABOUT US</h2>

</div>

</body>

</html>

1. Predict.html :

<!DOCTYPE html>

<html><head><title>Predict</title></head>

<body>

<h2>Upload pollen image:</h2>

<form method="post" enctype="multipart/form-data">

  <input type="file" name="file">

  <button type="submit">Predict</button>

</form>

{% if label %}

  <h3>Label: {{ label }} (Confidence: {{ '%.2f'|format(prob) }})</h3>

{% endif %}

</body></html>

1. App.py :

from flask import Flask, render\_template, request, redirect, url\_for, session

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

import numpy as np

import os

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret'

UPLOAD\_FOLDER = 'uploads/'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

model = load\_model('model/model.h5')

@app.route('/')

def home():

    return render\_template('index.html')

@app.route('/login', methods=['GET','POST'])

def login():

    if request.method == 'POST':

        user = request.form['username']

        pwd = request.form['password']

        if user == 'admin' and pwd == 'pass':

            session['user'] = user

            return redirect(url\_for('predict'))

        return render\_template('login.html', error='Invalid')

    return render\_template('login.html')

@app.route('/predict', methods=['GET','POST'])

def predict():

    if 'user' not in session:

        return redirect(url\_for('login'))

    if request.method == 'POST':

        img = request.files['file']

        path = os.path.join(app.config['UPLOAD\_FOLDER'], img.filename)

        img.save(path)

        image = load\_img(path, target\_size=(224,224))

        arr = img\_to\_array(image)/255.0

        arr = np.expand\_dims(arr, axis=0)

        pred = model.predict(arr)

        cls = np.argmax(pred, axis=1)[0]

        return render\_template('predict.html', label=str(cls), prob=float(np.max(pred)))

    return render\_template('predict.html', label=None)

1. Pollen\_grain\_classification.ipynb :

We have to install the following pip:

* Pip install python==3.10.2
* Pip install kaggle
* Pip install –upgrade pip
* Pip install PyDrive
* Pip install gdown
* Pip install matplotlib
* Pip install tensorflow==2.16.1 scikit-learn
* Pip install opencv-python
* Pip install keras

1. Read the Data: (Jupiter notebook)

from kaggle.api.kaggle\_api\_extended import KaggleApi

#Authenticate using the API token

api = KaggleApi()

api.authenticate()

# Download and unzip the dataset

api.dataset\_download\_files('andrewmvd/pollen-grain-image-classification', path='pollen\_dataset', unzip=True)

print("Dataset downloaded to pollen\_dataset")

1. path="C:\pollen\_grain\data"
2. from typing import Counter

import os

names= [name.replace(' ','\_').split('\_')[0] for name in os.listdir(path)]

classes = Counter(names)

print(classes)

1. Total no of images:

print("Number of images:",len(names))

1. from matplotlib import pyplot as plt

plt.figure(figsize=(8,3))

plt.title('Classs counts in Dataset')

plt.bar(\*zip(\*classes.items()))

plt.xticks(rotation='vertical')

plt.show()

1. path\_class={key:[] for key in classes.keys()}

for name in os.listdir(path):

key= name.replace(' ','\_').split('\_')[0]

path\_class[key].append(os.path.join(path ,name))

1. from PIL import Image

fig = plt.figure(figsize=(15,15))

for i, key in enumerate(path\_class.keys()):

img1= Image.open(path\_class[key][0])

img2= Image.open(path\_class[key][1])

img3= Image.open(path\_class[key][2])

ax= fig.add\_subplot(8,9, 3\*i+1)

ax.set\_xticks([])

ax.set\_ytick=([])

ax.imshow(img1)

ax.set\_title(key)

ax= fig.add\_subplot(8,9, 3\*i+2)

ax.set\_xticks([])

ax.set\_ytick=([])

ax.imshow(img2)

ax.set\_title(key)

ax= fig.add\_subplot(8,9, 3\*i+3)

ax.set\_xticks([])

ax.set\_ytick=([])

ax.imshow(img3)

ax.set\_title(key)

1. import cv2

import os

size= [cv2.imread(os.path.join(path ,name)).shape for name in os.listdir(path)]

x,y,\_= zip(\*size)

fig =plt.figure(figsize=(5,5))

#scatter plot

plt.scatter(x,y)

plt.title("Image size scatterplot")

#add diagonal red line

plt.plot([0,800],[0,800],'r')

plt.show()

1. Image Pre-Processing:

import cv2

def process\_img (img,size =(128,128)):

    img=cv2.resize(img,size) # resize images

    img=img/255 # devide values to 255

    return img

import os

import cv2

import numpy as np

path= "C:/pollen\_grain/data"

x,y =[],[]

for name in os.listdir(path):

  img\_path = os.path.join(path, name)

  img = cv2.imread(img\_path)

  if img is not None:

    x.append(process\_img(img))

    y.append(name.replace(' ','\_').split('\_')[0])

  else:

    print(f"Warning: Could not read image file: {img\_path}")

x = np.array(x)

import numpy as np

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

le = LabelEncoder()

Y\_le = le.fit\_transform(y)

Y\_le = Y\_le.reshape(-1, 1) if isinstance(Y\_le, np.ndarray) else np.array(Y\_le).reshape(-1, 1)

encoder = OneHotEncoder(sparse\_output=False)

Y\_cat = encoder.fit\_transform(Y\_le)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x, Y\_cat, test\_size=0.285,stratify=Y\_le)

print("Image in each class in Test set:{}".format(np.sum(Y\_test, axis=0)))

import tensorflow as tf

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt

import os

#venv\Scripts\activate

# pip list

#pip install tensorflow

#pip install tensorflow-cpu

#pip install ipykernel -U --force-reinstall

# python3.10 -m pip install tensorflow

# set paths and parameters

img\_height, img\_width = 128, 128

batch\_size = 32

data\_dir = "dataset"  # Folder containing 'train', 'val', 'test' folders

img\_height, img\_width = 128, 128

batch\_size = 32

data\_dir = "dataset"  # Folder containing 'train', 'val', 'test' folders