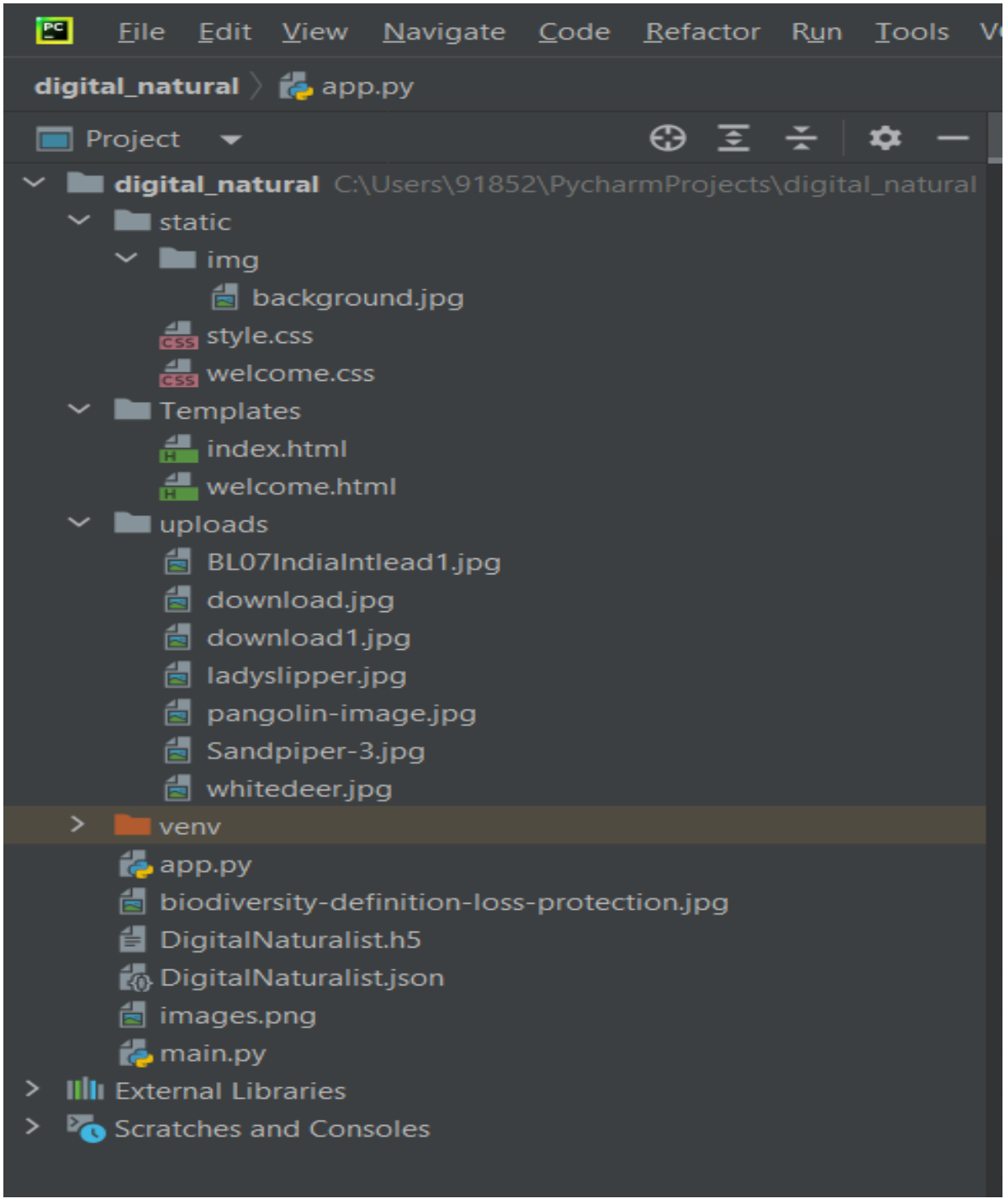
Project Structure

Date	17 November 2022
Team ID	PNT2022TMID18534
Project Name	Project - Digital Naturalist - AI Enabled tool for Biodiversity Researchers
Description	The following document shows the intended project structure within the flask application done in PyCharmIDE



app.py | welcome.html | welcome.css | index.html | style.css

```python
from __future__ import division, print_function

import os

import numpy as np
import tensorflow as tf
from flask import Flask, redirect, render_template, request
from keras.applications.inception_v3 import preprocess_input
from keras.models import model_from_json
from werkzeug.utils import secure_filename

global graph
graph=tf.compat.v1.get_default_graph()
#this list is used to log the predictions in the server console
predictions = ["Corpse Flower",
               "Great Indian Bustard",
               "Lady's slipper orchid",
               "Pangolin",
               "Spoon Billed Sandpiper",
               "Seneca White Deer"
              ]
#this list contains the link to the predicted species
found = [
        "https://en.wikipedia.org/wiki/Amorphophallus_titanum",
        "https://en.wikipedia.org/wiki/Great_Indian_bustard",
        "https://en.wikipedia.org/wiki/Cypripedioideae",
        "https://en.wikipedia.org/wiki/Pangolin",
        "https://en.wikipedia.org/wiki/Spoon-billed_sandpiper",
        "https://en.wikipedia.org/wiki/Seneca_white_deer",
        ]
app = Flask(__name__,template_folder="Templates")
```

app.py | welcome.html | welcome.css | index.html | style.css

```
@app.route('/index')
def pop():
    return render_template('index.html')

@app.route('/',methods=['GET','POST'])
def index():
    # Home Page
    return render_template("welcome.html")
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'GET':
        return ("<h6 style=\"font-face:\"Courier New\";\">No GET request herd.....</h6 >")
    if request.method == 'POST':
        # Fetching the uploaded image from the post request using the id 'uploadedimg'
        f = request.files['uploadedimg']
        basepath = os.path.dirname(__file__)
        #Securing the file by creating a path in local storage
        file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
        #Saving the uploaded image locally
        f.save(file_path)
        #loading the locally saved image
        img = tf.keras.utils.load_img(file_path, target_size=(224, 224))
        #converting the loaded image to image array
        x = tf.keras.utils.img_to_array(img)
        x = preprocess_input(x)
        # Converting the preprecessed image to numpy array
        inp = np.array([x])
        with graph.as_default():
            #loading the saved model from training
            json_file = open('DigitalNaturalist.json', 'r')
            loaded_model_json = json_file.read()
```

```
66                  #adding weights to the trained model
67                  loaded_model.load_weights("DigitalNaturalist.h5")
68                  #predecting the image
69                  preds = np.argmax(loaded_model.predict(inp),axis=1)
70                  #logs are printed to the console
71                  print("Predicted the Species " + str(predictions[preds[0]]))
72              text = found[preds[0]]
73              return redirect(text)
74
75
76
77  if __name__ == '__main__':
78      #Threads enabled so multiple users can request simultaneously
79      #debug is turned off, turn on during development to debug the errors
80      #application is binded to port 8000
81      app.run(threaded = True,debug=True,port="8000")
82
```