Digital Naturalists AI Enabled tool for Biodiversities and Researchers

Team ID	PNT2022TMID23849
Project Name	Digital Naturalists AI Enabled tool for Biodiversities and Researchers

1.INTRODUCTION:

➤This research aims to develop and evaluate a design framework for creating digital devices that support the exploration of animal behaviors in the wild. This paper quickly shares the main concepts and theories from the fields forming Digital Naturalism's foundation while presenting the key challenges emerging from these critical intersections between field biology and computational media.

➤It then reviews the development of this research's hybrid methodology designed specifically for its multi-year series of "Qualitative Action Research" fieldwork carried out at a rainforest field station.

➤This paper analyzes the resulting on-site ethnographies, workshops, design projects, and interactive performances, whose take-aways are synthesized into design guidelines for digital-natural media.

➤This framework, itself, is then evaluated via an extra iteration of fieldwork and the results discussed. Finally, the paper identifies targets for continued research development. Further areas of interest are presented which will promote Digital Naturalism's progression.

2.PROJECT DESCRIPTION:

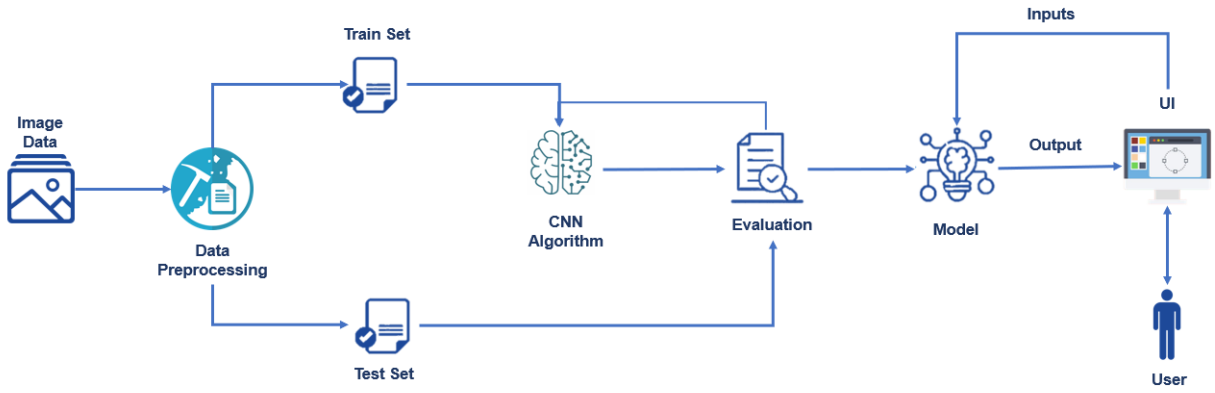
- ➤ A naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature. Being able to identify the flora and fauna around us often leads to an interest in protecting wild spaces, and collecting and sharing information about the species we see on our travels is very useful for conservation groups like NCC.

- ➤ When venturing into the woods, field naturalists usually rely on common approaches like always carrying a guidebook around everywhere or seeking help from experienced ornithologists. There should be a handy tool for them to capture, identify and share the beauty to the outside world.

- ➤ Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals and other species they see on their hikes, canoe trips and other excursions.

- In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals (2 subclasses in each for a quick understanding)and get the prediction of the bird when an image is been given.

Technical Architecture :



3.OBJECTIVES:

BY THE END OF PROJECT WE WILL BE ABLE TO:

- Using the flask frame work we will be able to build an application.
- Understanding of time series data.
- Techniques and concepts of time series forecasting.
- Splits the data and analysis.

4. PROJECT FLOW:

TO COMPLETE CERTAIN ACTIVITIES AND TO ACCOMPLISH THE TASKS LISTED BELOW:

- Collection of data
- Dataset creation
- Preprocessing of data
- Libraries imports
- Dataset imports
- Data analyze
- Fit the model on train data and check for accuracies using test data as well.
- Featuring the model and dependencies
- Build web application using flask

5.DATA COLLECTION:

We have downloaded the dataset from :
https://drive.google.com/file/d/1hjPKoJi-3t0yZJnPoJF7gNAMn0rgIXcr/view?usp=sharing

6. DATA PREPROCESSING:

Data pre processing includes the following tasks:

- Importing the libraries: The required libraries to import the python scripts are,

6.1 NUMPY:

Its the open source numerical python script.It contains the multi dimensional array and matrix data structure also perform the mathematical operation on array such as trigonometric,statistical and algebra.

6.2 PANDAS:

One of the top python programming languages and it is fast , flexible and easy to use the open source data analysis

7.AI HELP IN BIODIVERSITY:

Animal conservation is becoming one of the key issues in saving biodiversity on the Earth. AI can play a vital role in detecting, recognizing, and keeping track of wild animals wandering in their natural environment or conserved within the wildlife sanctuaries.Most importantly, AI can help in preventing the extinction of endangered plants and animals. If such animals are kept under observation or tracked by the forest rangers, they can be protected from natural disasters such as fires in the forests, floods, and all illegal activities like poaching.To conserve wild animals, AI-enabled devices, applications, & analysis/monitoring system is used to keep their track records and understand the behavior of animals for right predictions of such models. Let's find out how AI-enabled applications can be used for conserving animals.

7.1 HANDLING MISSING DATA:

➤ After loading the dataset check the rows and columns for their null values with complete information.

➤ If there is any null values ,following can be done:

➤ Using data imputation the data is imputing in sklearn.

➤ Filling the NaN values with help of median, mode and mean using fillna() method.

➤ Delete the records.

➤ Now we can see the null values in the closing value column and also check how many numbers of null values in the column using sum() function.

➤ Drop the null from the column.

➤ Axis=0 drop the row.

➤ Data frame has to change permanent indicates, ' in place=True'.

➤ Reset_index consider the closing value column to reset the index of data frame list of integer from 0 to length of data.

7.2 FEATURE SCALING:

Feature scaling to normalize the independent variables to scale the crude oil prices between (0 ,1) to avoid the computation . Some of the common methods are standardization and Normalization.

7.3 STANDARDIZATION:

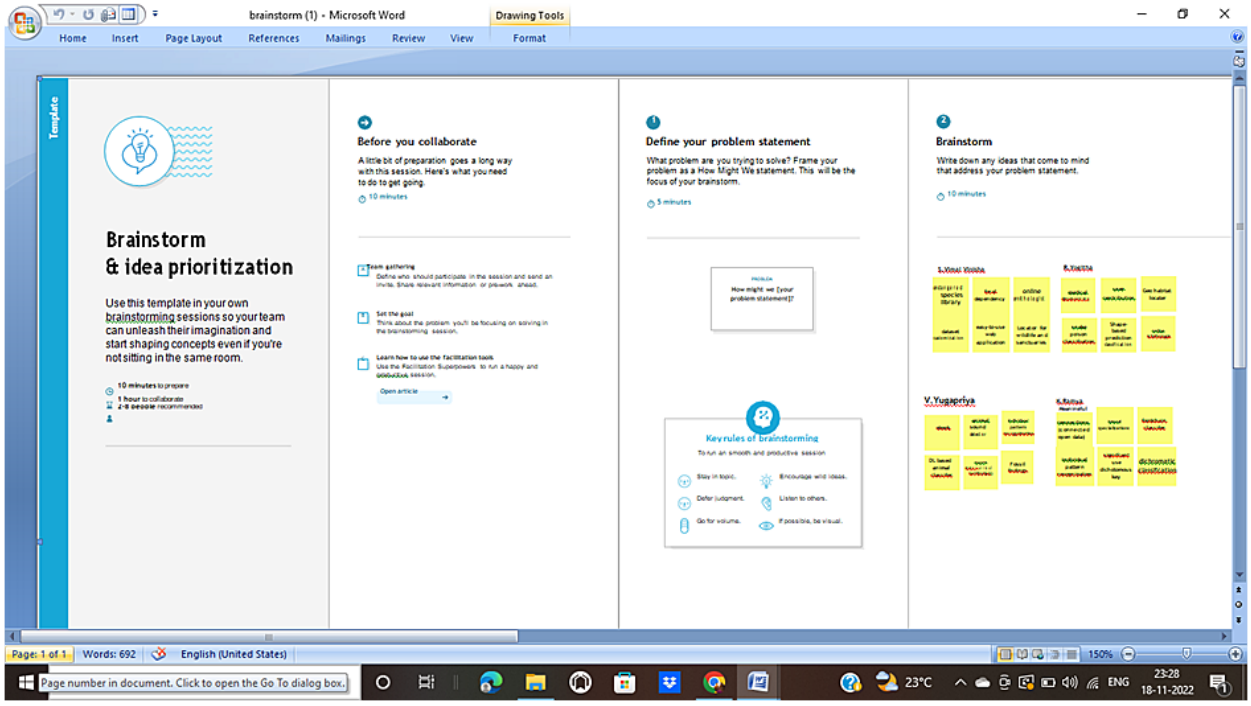
The process of developing and implementing the technical products and stability of the products.

7.3 NORMALIZATION:

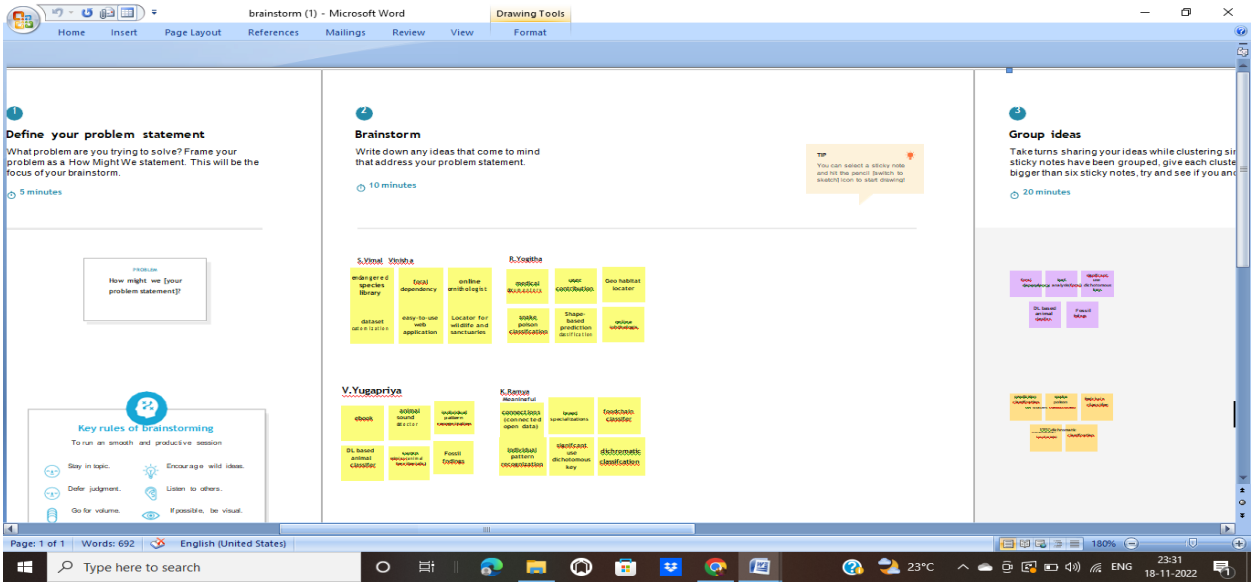
The process of organizing the data base or data frame to reduce the redundancy and improve the integrity of the data , improve and simplifies the data design.

Ideation & Brainstorming:

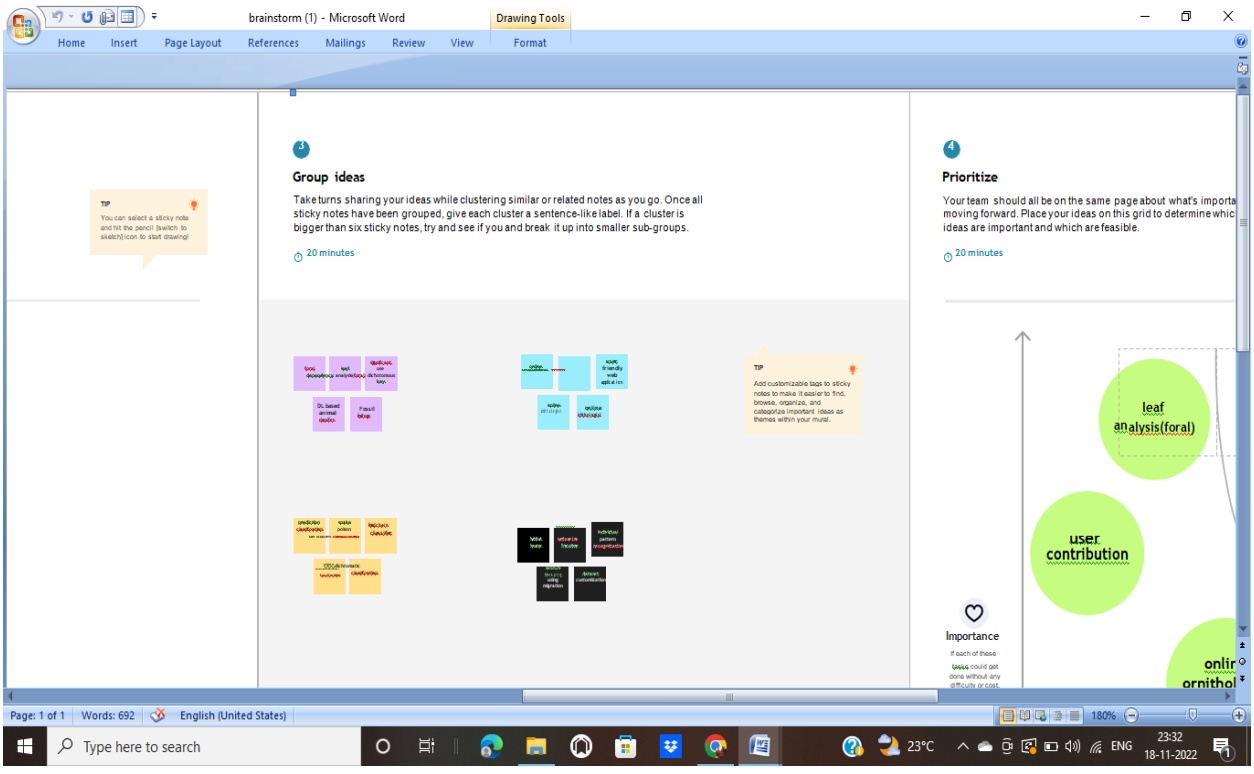
Step-1: Team Gathering, Collaboration and Select the Problem Statement.



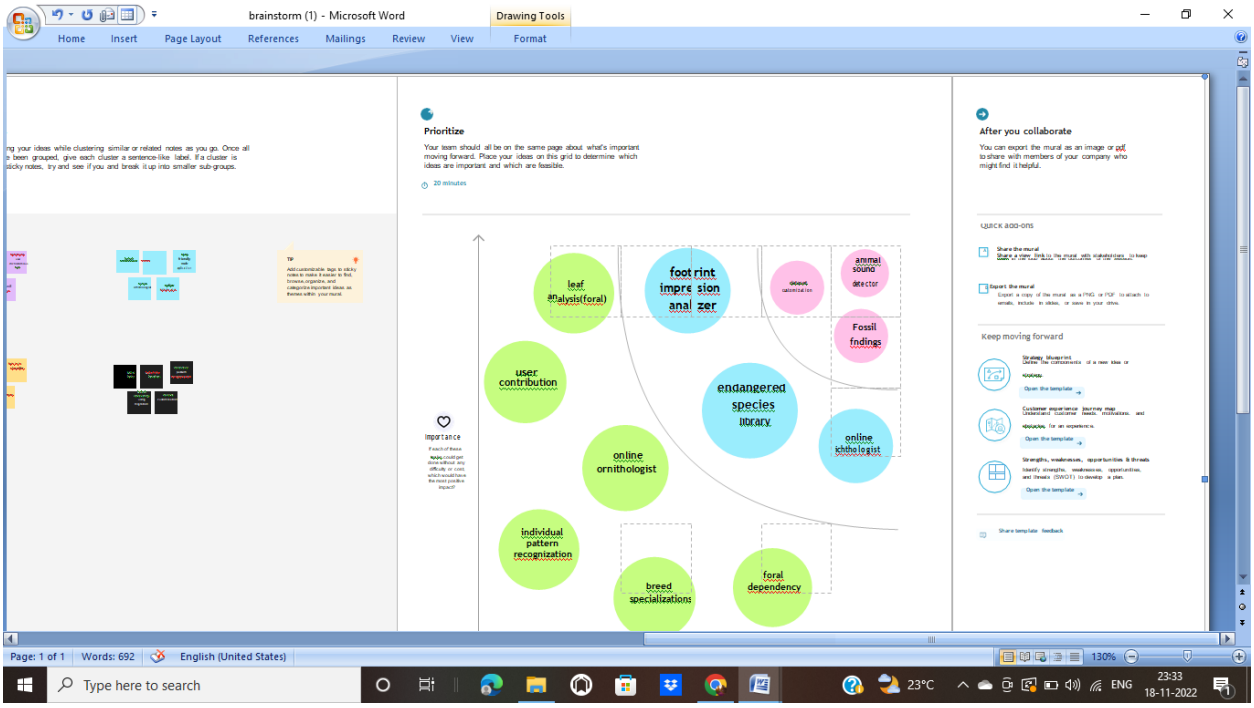
Step-2: Brainstorm, Idea Listing and Grouping



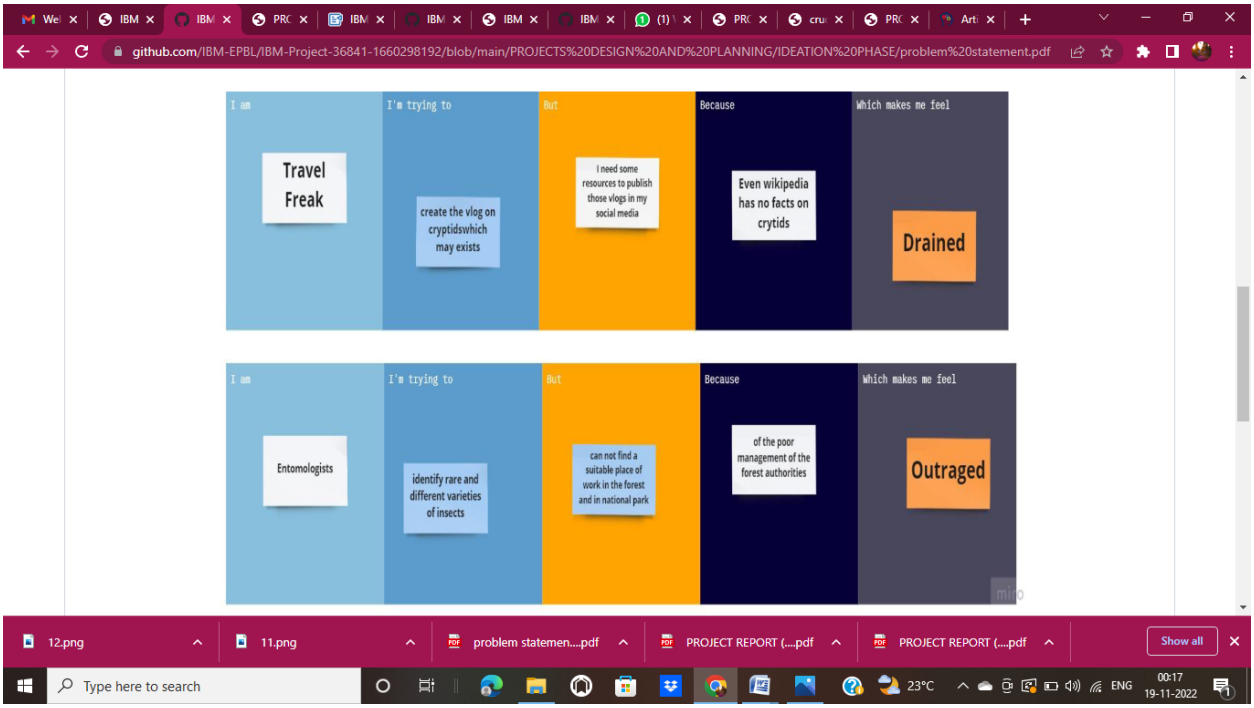
Step-3: Idea Prioritization



Prioritize:

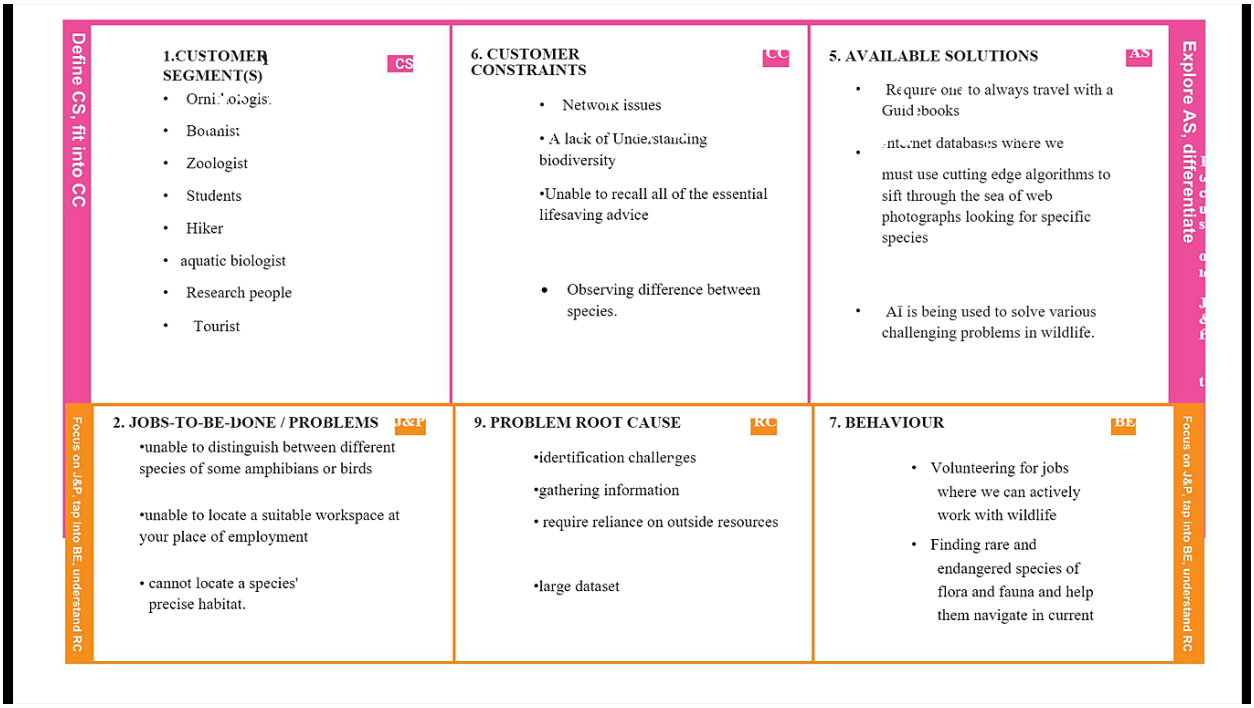


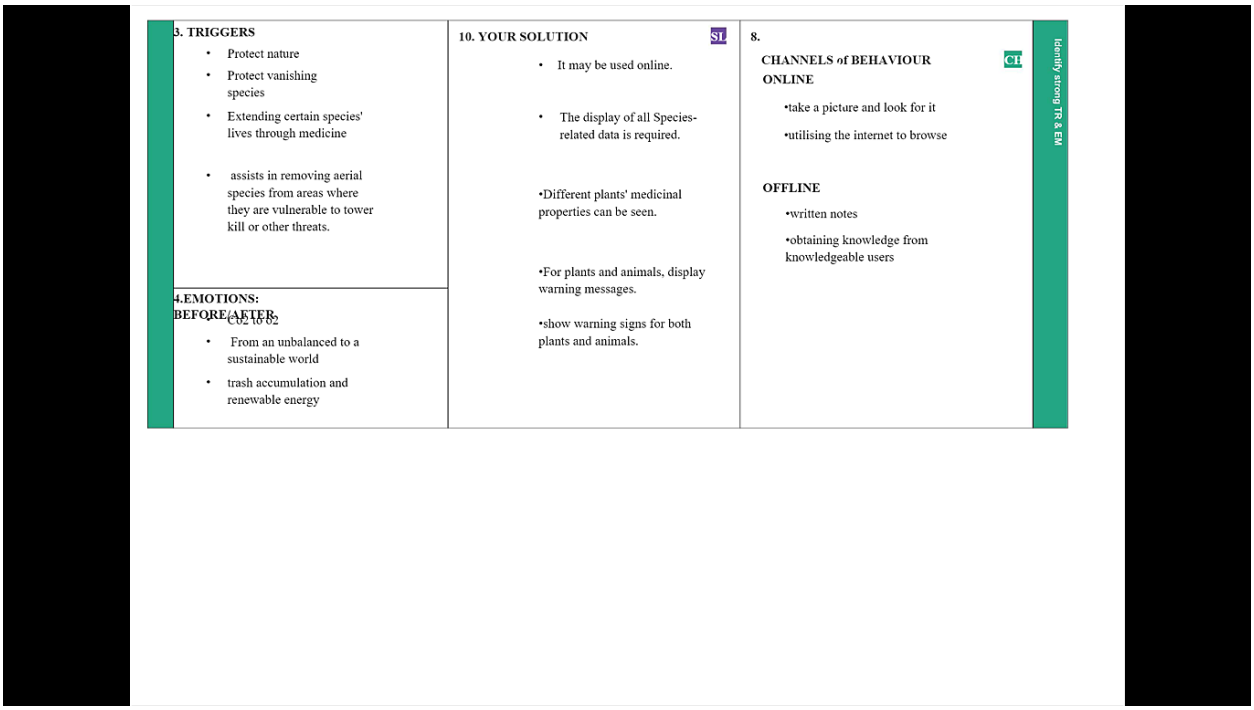
PROBLEM STATEMENTS:



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Travel Freak	create the vlog on cryptidswhich may exists	I need some resources to publish those vlogs in my social media	Even wikipedia has no facts on crytids	Drained
PS-2	Entomologists	identify rare and different varieties of insects	can not find a suitable place of work in the forest and in national park	of the poor management of the forest authorities	Outraged

PROBLEM SOLUTION FIT:





INDEX CODING:

SPRINT 1:

#1. Downloaded Datasets
#2. Augmenting Data

```
from keras.preprocessing.image import ImageDataGenerator
import cv2
from os import listdir
import time

# Nicely formatted time string to make a note of how much time it takes for augmentation
def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return f"{h}:{m}:{round(s,1)}"

def augment_data(file_dir, n_generated_samples, save_to_dir):
    """
    Arguments:
    file_dir: A string representing the directory where images that we want to augment are found.
```

```
    n_generated_samples: A string representing the number of generated samples using the
given image.
    save_to_dir: A string representing the directory in which the generated images will be saved.
    """
    #from keras.preprocessing.image import ImageDataGenerator
    #from os import listdir
    data_gen = ImageDataGenerator(rotation_range=30,
                                  width_shift_range=0.1,
                                  height_shift_range=0.15,
                                  shear_range=0.25,
                                  zoom_range = 0.2,
                                  horizontal_flip=True,
                                  vertical_flip=False,
                                  fill_mode='nearest',
                                  brightness_range=(0.5,1.2)
                                 )

    for filename in listdir(file_dir):
        # load the image
        image = cv2.imread(file_dir + '/' + filename)
        # reshape the image
        image = image.reshape((1,)+image.shape)
        # prefix of the names for the generated sampels.
        save_prefix = 'aug_' + filename[:-4]
        # generate 'n_generated_samples' sample images
        i=0
        for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_dir,
                                   save_prefix=save_prefix, save_format='jpg'):
            i += 1
            if i > n_generated_samples:
                break
start_time = time.time()

#3. Augmentation Structure Creation
augmented_data_path = r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\augmented data'

#For Birds
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist
Dataset\Bird\Great Indian Bustard Bird', n_generated_samples=8,
save_to_dir=augmented_data_path+'Bird/GIB_AUG')
```

```
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist Dataset\Bird\Spoon Billed Sandpiper Bird', n_generated_samples=8, save_to_dir=augmented_data_path+'Bird/SPS_AUG')

#For MAMMALS
# augment data for the examples with label equal to GIB in Flower
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist Dataset\Flower\Corpse Flower', n_generated_samples=8, save_to_dir=augmented_data_path+'Flower/Corpse_AUG')
# augment data for the examples with label equal to GIB in Flower
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist Dataset\Flower\Lady Slipper Orchid Flower', n_generated_samples=8, save_to_dir=augmented_data_path+'Flower/LS_Orchid_AUG')

#For Flowers
# augment data for the examples with label equal to GIB in Mammals
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist Dataset\Mammal\Pangolin Mammal', n_generated_samples=8, save_to_dir=augmented_data_path+'Mammal/Pangolin_AUG')
# augment data for the examples with label equal to GIB in Mammals
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist Dataset\Mammal\Senenca White Deer Mammal', n_generated_samples=8, save_to_dir=augmented_data_path+'Mammal/SW_Deer_AUG')


end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

SPRINT-2:

```
#1. Downloaded Datasets
#2. Augmenting Data

from keras.preprocessing.image import ImageDataGenerator
import cv2
from os import listdir
import time

# Nicely formatted time string to make a note of how much time it takes for augmentation
```

```
def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return f"{h}:{m}:{round(s,1)}"

def augment_data(file_dir, n_generated_samples, save_to_dir):
    """
    Arguments:
    file_dir: A string representing the directory where images that we want to augment are found.
    n_generated_samples: A string representing the number of generated samples using the
given image.
    save_to_dir: A string representing the directory in which the generated images will be saved.
    """
    #from keras.preprocessing.image import ImageDataGenerator
    #from os import listdir
    data_gen = ImageDataGenerator(rotation_range=30,
                                  width_shift_range=0.1,
                                  height_shift_range=0.15,
                                  shear_range=0.25,
                                  zoom_range = 0.2,
                                  horizontal_flip=True,
                                  vertical_flip=False,
                                  fill_mode='nearest',
                                  brightness_range=(0.5,1.2)
                                 )

    for filename in listdir(file_dir):
        # load the image
        image = cv2.imread(file_dir + '/' + filename)
        # reshape the image
        image = image.reshape((1,)+image.shape)
        # prefix of the names for the generated sampels.
        save_prefix = 'aug_' + filename[:-4]
        # generate 'n_generated_samples' sample images
        i=0
        for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_dir,
                                   save_prefix=save_prefix, save_format='jpg'):
            i += 1
            if i > n_generated_samples:
                break
```

```
start_time = time.time()

#3. Augmentation Structure Creation
augmented_data_path = r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\augmented data'

#For Birds
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist
Dataset\Bird\Great Indian Bustard Bird', n_generated_samples=8,
save_to_dir=augmented_data_path+'Bird/GIB_AUG')
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist
Dataset\Bird\Spoon Billed Sandpiper Bird', n_generated_samples=8,
save_to_dir=augmented_data_path+'Bird/SPS_AUG')

#For MAMMALS
# augment data for the examples with label equal to GIB in Flower
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist
Dataset\Flower\Corpse Flower', n_generated_samples=8,
save_to_dir=augmented_data_path+'Flower/Corpse_AUG')
# augment data for the examples with label equal to GIB in Flower
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist
Dataset\Flower\Lady Slipper Orchid Flower', n_generated_samples=8,
save_to_dir=augmented_data_path+'Flower/LS_Orchid_AUG')

#For Flowers
# augment data for the examples with label equal to GIB in Mammals
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist
Dataset\Mammal\Pangolin Mammal', n_generated_samples=8,
save_to_dir=augmented_data_path+'Mammal/Pangolin_AUG')
# augment data for the examples with label equal to GIB in Mammals
augment_data(file_dir=r'C:\Users\vijay\OneDrive\Desktop\Digital Naturalist\Digital Naturalist
Dataset\Mammal\Senenca White Deer Mammal', n_generated_samples=8,
save_to_dir=augmented_data_path+'Mammal/SW_Deer_AUG')



end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}"
```

SPRINT-3:


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "a988a44c",
   "metadata": {},
   "outputs": [],
   "source": [
    "from future import division, print_function\n",
    "\n",
    "import os\n",
    "\n",
    "import numpy as np\n",
    "import tensorflow as tf\n",
    "from flask import Flask, redirect, render_template, request\n",
    "from keras.applications.inception_v3 import preprocess_input\n",
    "from keras.models import model_from_json\n",
    "from werkzeug.utils import secure_filename\n",
    "\n",
    "global graph\n",
    "graph=tf.compat.v1.get_default_graph()\n",
    "#this list is used to log the predictions in the server console\n",
    "predictions = [\"Corpse Flower\", \n",
    ""              \"Great Indian Bustard\", \n",
    ""              \"Lady's slipper orchid\", \n",
    ""              \"Pangolin\", \n",
    ""              \"Spoon Billed Sandpiper\", \n",
    ""              \"Seneca White Deer\"\n",
    ""             ]\n",
    "#this list contains the link to the predicted species              \n",
    "found = [\n",
    ""       \"https://en.wikipedia.org/wiki/Amorphophallus_titanum\",\n",
    ""       \"https://en.wikipedia.org/wiki/Great_Indian_bustard\",\n",
    ""       \"https://en.wikipedia.org/wiki/Cypripedioideae\",\n",
    ""       \"https://en.wikipedia.org/wiki/Pangolin\",\n",
    ""       \"https://en.wikipedia.org/wiki/Spoon-billed_sandpiper\",\n",
    ""       \"https://en.wikipedia.org/wiki/Seneca_white_deer\",\n",
```

```
"       ]\n",
"app = Flask(name)\n",
"\n",
"@app.route('/', methods=['GET'])\n",
"def index():\n",
"    # Home Page\n",
"    return render_tem
```

SPRINT 4:

```
# -- coding: utf-8 --


from _future_ import print_function


from _future_ import division

import os

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

global graph
graph=tf.compat.v1.get_default_graph()
#this list is used to log the predictions in the server console
predictions = np.array(["Seneca White Deer",
            "Pangolin",
            "Lady's slipper orchid",
            "Corpse Flower",
            "Spoon Billed Sandpiper",
            "Great Indian Bustard"
            ])
```

```
#this list contains the link to the predicted species
found = np.array([
   "Seneca White Deer",
            "Pangolin",
            "Lady's slipper orchid",
            "Corpse Flower",
            "Spoon Billed Sandpiper",
            "Great Indian Bustard"
      ])
app = Flask(_name_)
model = load_model("model.h5")

@app.route('/', methods=['GET'])
def index():
    # Home Page
    return render_template("index.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':
    # fecting 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 preprocessed image to numpy array
    inp = np.array([x])
    with graph.as_default():
      #loading the saved model from training
      json_file = open('DigitalNaturalist.json')
      loaded_model_json = json_file.read()
      json_file.close()
```

```
    loaded_model = model_from_json(loaded_model_json)
    #adding weights to the trained model
    loaded_model.load_weights("model.h5")
    #predecting the image
    preds = np.argmax(loaded_model.predict(inp),axis=1)
    #logs are printed to the console
    print("The predicted species is " , predictions[preds[0]])
  text = "The predicted species is " + found[preds[0]]
  return render_template("index.html", RESULT = text)


if _name_ == '_main_':
  #Threads enabled so multiple users can request simutaneously
  #debud is turned off, turn on during development to debug the errors
  #applications is binded to port 8000
  app.run(threaded = True,debug=True,port="8000")
```

BUILD PYTHON CODE:

- We have a built a flask file ‘app.py’ which is a web framework written in python for server-side scripting.
- Let’s see step by step procedure for building the backend application.
- The app starts running when the “__name__” constructor is called in main. render_template is used to return HTML files.

```
from keras.preprocessing.image import ImageDataGenerator
import cv2
from os import listdir
import time

# Nicely formatted time string to make a note of how much time it takes for augmentation
def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return f"{h}:{m}:{round(s,1)}"

def augment_data(file_dir, n_generated_samples, save_to_dir):
    """
    Arguments:
    file_dir: A string representing the directory where images that we want to augment are found.
    n_generated_samples: A string representing the number of generated samples using the given image.
    save_to_dir: A string representing the directory in which the generated images will be saved.
    """

    #from keras.preprocessing.image import ImageDataGenerator
    #from os import listdir

    data_gen = ImageDataGenerator(rotation_range=30,
                                  width_shift_range=0.1,
                                  height_shift_range=0.15,
                                  shear_range=0.25,
                                  zoom_range = 0.2,
                                  horizontal_flip=True,
                                  vertical_flip=False,
                                  fill_mode='nearest',
                                  brightness_range=(0.5,1.2)
                                 )
```

```
    for filename in listdir(file_dir):
        # load the image
        image = cv2.imread(file_dir + '/' + filename)
        # reshape the image
        image = image.reshape((1,)+image.shape)
        # prefix of the names for the generated sampels.
        save_prefix = 'aug_' + filename[:-4]
        # generate 'n_generated_samples' sample images
        i=0
        for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_dir,
                                   save_prefix=save_prefix, save_format='jpg'):
            i += 1
            if i > n_generated_samples:
                break
```

```
start_time = time.time()

augmented_data_path = 'D:/TSB Projects/Digital Naturalist/augmented data/'

#For Birds
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir='D:/TSB Projects/Digital Naturalist/Digital Naturalist Dataset/Bird/Great Indian Bustard Bird', n_generated_samples=8, save_to_dir=augmented_data_path+'Bird/GIB_AUG')
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir='D:/TSB Projects/Digital Naturalist/Digital Naturalist Dataset/Bird/Spoon Billed Sandpiper Bird', n_generated_samples=8, save_to_dir=augmented_data_path+'Bird/SPS_AUG')

#For MAMMALS
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir='D:/TSB Projects/Digital Naturalist/Digital Naturalist Dataset/Flower/Corpse Flower', n_generated_samples=8, save_to_dir=augmented_data_path+'Flower/Corpse_AUG')
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir='D:/TSB Projects/Digital Naturalist/Digital Naturalist Dataset/Flower/Lady Slipper Orchid Flower', n_generated_samples=8, save_to_dir=augmented_data_path+'Flower/LS_Orchid_AUG')

#For Flowers
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir='D:/TSB Projects/Digital Naturalist/Digital Naturalist Dataset/Mammal/Pangolin Mammal', n_generated_samples=8, save_to_dir=augmented_data_path+'Mammal/Pangolin_AUG')
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir='D:/TSB Projects/Digital Naturalist/Digital Naturalist Dataset/Mammal/Senenca White Deer Mammal', n_generated_samples=8, save_to_dir=augmented_data_path+'Mammal/SW_Deer_AUG')


end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
#For matrix calculations and data Managememnt
import numpy as np

#Importing libraries required for the model
import tensorflow as tf
import keras
import keras.backend as K

from keras.optimizers import SGD, Adam, Adagrad, RMSprop
from keras.applications import *
from keras.preprocessing import *
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Activation, BatchNormalization,Dropout
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

#For plotting charts used for data visualizations
import matplotlib.pyplot as plt

#Libraries for Locating and loading data
import glob
from PIL import Image
import os
from os import listdir
```

```
#Setting path to our dataset folder
dirName = 'D:/TSB Projects/Digital Naturalist/augmented data'
folders = listdir(dirName)

#Getting the names for all the folders containing data
def getListOfFiles(dirName):
# create a list of sub directories and files(if any)
# names in the given directory
    listOfFile = os.listdir(dirName)
    allFiles = list()
    for fol_name in listOfFile:
        fullPath = os.path.join(dirName, fol_name)
        allFiles.append(fullPath)

    return allFiles

Folders = getListOfFiles(dirName)
len(Folders)
subfolders = []
for num in range(len(Folders)):
    sub_fols = getListOfFiles(Folders[num])
    subfolders+=sub_fols
#Now, the subfolders contains the address to all our data folders for each class
subfolders
```

```
#\\\\\\\\/////////\\\\\\\\\/////////\\\\\\\\\\\\\\\\\
#\\\\\\\\/////////\\\\\\\\\/////////\\\\\\\\\\\\\\\\\
#Loading the data and pre processing it to make it in trainable format
#\\\\\\\\/////////\\\\\\\\\/////////\\\\\\\\\\\\\\\\\
#\\\\\\\\/////////\\\\\\\\\/////////\\\\\\\\\\\\\\\\\

#X data will includes the data generated for each image
#Y data will include a id no, unique for every different species, so are having 6 classes
#there for we will get 6 ids = [0,1,2,3,4,5]
#That will be tha label we're classifying.
X_data = []
Y_data = []
id_no=0

#to make a list of tuples, where we'll store the info about the image, category and species
found = []

#itering in all folders under Augmented data folder
for paths in subfolders:
    #setting folder path for each unique class and category
    files = glob.glob (paths + "/*.jpg")
    #adding tuples to the list that contain folder name and subfolder name
    found.append((paths.split('\\')[-2],paths.split('\\')[-1]))

    #itering all files under the folder one by one
    for myFile in files:
        img = Image.open(myFile)
        #img.thumbnail((width, height), Image.ANTIALIAS) # resizes image in-place keeps ratio
        img = img.resize((224,224), Image.ANTIALIAS) # resizes image without ratio
        #convert the images to numpy arrays
        img = np.array(img)
        if img.shape == ( 224, 224, 3):
            # Add the numpy image to matrix with all data
            X_data.append (img)
            Y_data.append (id_no)
    id_no+=1
```

```
#to see our data
print(X_data)
print(Y_data)


#converting lists to np arrays again
X = np.array(X_data)
Y = np.array(Y_data)

# Print shapes to see if they are correct
print("x-shape",X.shape,"y shape", Y.shape)

X = X.astype('float32')/255.0
#The Keras library offers a function called to_categorical() that you can use to one hot encode
#integer data. The sequence has an example of all known values
#so we can use the to_categorical() function directly
y_cat = to_categorical(Y_data, len(subfolders))

print("X shape",X,"y_cat shape", y_cat)
print("X shape",X.shape,"y_cat shape", y_cat.shape)

#\\\\\\\\/////////\\\\\\\\/////////\\\\\\\\\/////////
#\\\\\\\\/////////\\\\\\\\/////////\\\\\\\\\/////////
#Splitting the data to Test and Train
#\\\\\\\\/////////\\\\\\\\/////////\\\\\\\\\/////////
#\\\\\\\\/////////\\\\\\\\/////////\\\\\\\\\/////////
X_train, X_test, y_train, y_test = train_test_split(X, y_cat, test_size=0.2)
print("The model has " + str(len(X_train)) + " inputs")
```

```
from __future__ import division, print_function
import os
import numpy as np
from keras.preprocessing import image
from keras.models import load_model
import tensorflow as tf
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename
from keras.models import model_from_json
```

```
global graph
graph=tf.get_default_graph()
# Define a flask app
app = Flask(__name__)


# Load your trained model
json_file = open('final_model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
loaded_model.load_weights("final_model.h5")

print('Model loaded. Check http://127.0.0.1:5000/')
```

BUILD THE HTML PAGE CODING:

```
(base) F:\Projects\Digital Naturalist\Flask>python app.py
Using TensorFlow backend.
```

```
Model loaded. Check http://127.0.0.1:5000/
 * Debugger is active!
 * Debugger PIN: 257-358-499
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

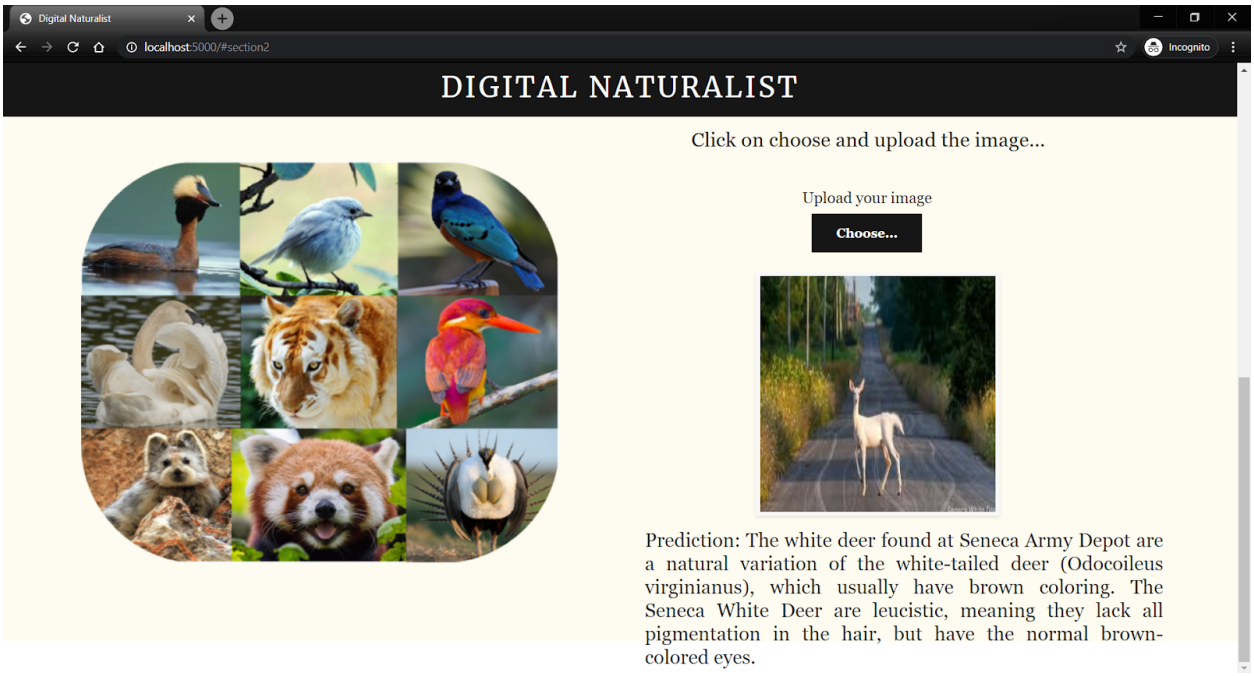
```
#printing the keys we have for the stores values
print(h.history.keys())
#appendind the data for each epoch in a arr, and for each batch size
histories_acc.append(h.history['acc'])
histories_val_acc.append(h.history['val_acc'])
histories_loss.append(h.history['loss'])
histories_val_loss.append(h.history['val_loss'])

#converting into numpy arrays
histories_acc = np.array(histories_acc)
histories_val_acc = np.array(histories_val_acc)
histories_loss = np.array(histories_loss)
histories_val_loss = np.array(histories_val_loss)

#here we have 3 columns and 6 rows each,ever row represetns differnt bath size,
#every column represent different epoch scores.
print('histories_acc',histories_acc,
      'histories_loss', histories_loss,
      'histories_val_acc', histories_val_acc,
      'histories_val_loss', histories_val_loss)
```

Open the browser and navigate to localhost:

The home page looks like this. When you click on the button “Drop the scan”, you’ll be redirected to the predict section.



CONCLUSION:

Technology has been criticized because of its instrumentalizing tendency, and theimplications of that tendency, for example, the increasing separation of humans fromnon-human nature that results in a lack of care that puts the environment under threat.It also includes the impact on humans’ sense of self and well-being. By respondingto the criticism directed at technologies

through design criteria that referenced thecriticism directly, the projects aimed to refocus and reshape priorities and designprocesses. The simple act of setting the criteria was important for foregrounding thequestions, “How are we using digital technologies?” and “What impact is this havingon humans and non-humans?”One of the significant impacts of this lens was the cascade created by prioritizingindividual places and rejecting the idea that “one size fits all.” Attending to placemeant the design process had to respect local ways of working including the rhythmsof the place, for example, seasonal and daily working patterns. This in turn influenceddigital technology choices and other material decisions which ultimately influencedphysical designs and sensory interactions. By attending to place within the designprocess, the designer–researcher (lead author) became more connected to the placesin which she designed. This led to the insight that making in a place and in responseto a place could foster a connection and a realization which drove the second phaseof research.The initial Digital Nature Hybrid designs went some way to revealing places andshowing non-human nature in a new light. It demonstrated how digital technologiescould be rooted in context and culture and how they could amplify particular sensorystimuli to resonate with the sense of place. This in itself had value, for designers,organization and visitors.However, the engagements with the Digital Nature Hybrids were brief and toosuperficial to prompt significant change or connection, unless the people concernedwere already primed for change or connection. Analyzing the limitations of theDigital Nature Hybrids with respect to the design criteria showed that effort, focus,skill and social connection that made focal things and practices meaningful were lesspresent in the Rhubaphone and Audio Apples and so the question of how to build inthese attributes became central to the second phase.The subsequent projects created deeper, richer encounters with natural environ-ments and showed the power of combining artistic activities with technological activ-ities. The activities required participants to use their technologically amplified sensesand creative skills to attend to and reflect on the world.Using technologies to kindle traditional skills and practices shows one way thattechnologies might contribute to re-energizing the culture of amateur naturalists andnurture care for non-human nature.