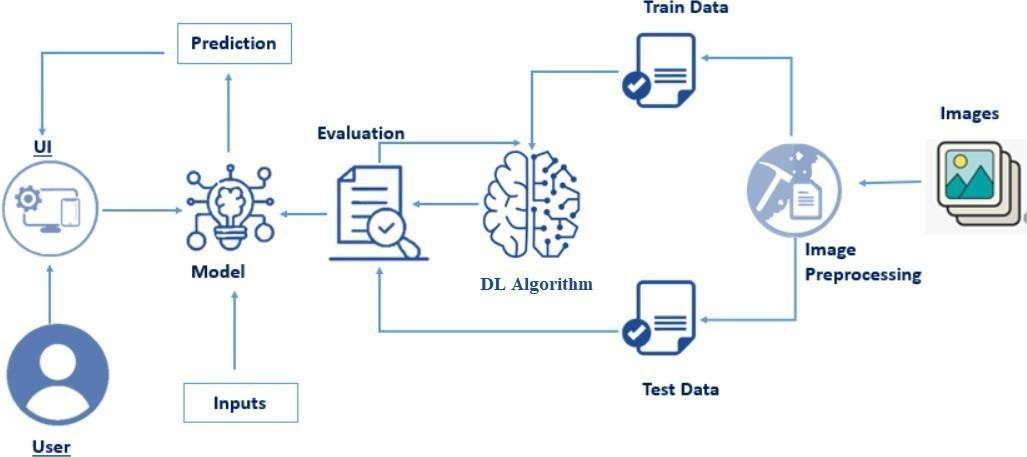
Dog Breed Identification using Transfer Learning

# Introduction:

The field of dog breed identification involves training a machine learning model to classify dog images into specific breed categories. This task is made complex by the wide variety of dog breeds, each having distinct physical characteristics, such as size, shape, coat color, and facial features. Deep learning algorithms can learn to recognize these unique features and patterns in images to make accurate predictions about the breed of a given dog. Here we use Transfer Learning Approach i.e., VGG19 Architecture for dog breed identification, a large dataset of labeled dog images is required.

This dataset should include images from various dog breeds, with each image properly labeled with its corresponding breed. During the training phase, the Model learns to associate specific visual patterns and features with each breed, adjusting its internal weights to improve classification accuracy. Dog breed identification using machine learning has numerous practical applications. It can assist veterinarians in diagnosing and treating specific breeds, help dog owners better understand their pets, and aid in dog-related services such as adoption, breeding, and training. Furthermore, it can contribute to research efforts in studying the genetic and phenotypic characteristics of different dog breeds.

# Technical Architecture:



**Prerequisites:**

# To complete this project, you must require the following software’s, concepts, and packages

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and VS code.

To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

Link: [Click here to](https://www.youtube.com/watch?v=5mDYijMfSzs&feature=emb_logo) watch the video

# 1. To build Machine learning models you must require the following packages

* **Numpy**:

o It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations

# Scikit-learn:

o It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy

# Flask:

Web framework used for building Web applications

# Python packages:

* + open anaconda prompt as administrator
  + Type “pip install numpy” and click enter.
  + Type “pip install pandas” and click enter.
  + Type “pip install scikit-learn” and click enter.
  + Type “pip install tensorflow==2.12.0” and click enter.
  + Type “pip install keras==2.12.0” and click enter.
  + Type “pip install Flask” and click enter.

# Deep Learning Concepts

* + **CNN:** a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery.

[CNN Basic](https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add)

* + **VGG19:** VGG19 is a deep convolutional neural network architecture for image classification, consisting of 19 layers with small convolution filters.

**[VGG19](https://keras.io/api/applications/vgg/)**

* + **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

**[Flask Basics](https://www.youtube.com/watch?v=lj4I_CvBnt0)**

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

# Project Objectives:

By the end of this project you will:

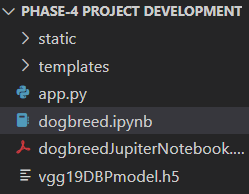
* Know fundamental concepts and techniques of Convolutional Neural Network.
* Gain a broad understanding of image data.
* Know how to pre-process/clean the data using different data preprocessing techniques.
* know how to build a web application using the Flask framework.

# Project Flow:

* The user interacts with the UI (User Interface) to choose the image.
* The chosen image is analyzed by the model which is integrated with flask application.
* CNN Models analyze the image, then prediction is showcased on the Flask UI. To accomplish this, we must complete all the activities and tasks listed below.
* Data Collection.
  + Create Train and Test Folders.
* Data Preprocessing.
  + Import the ImageDataGenerator library
  + Configure ImageDataGenerator class
  + ApplyImageDataGenerator functionality to Train dataset and Test dataset
* Model Building
  + Import the model building Libraries.
  + Importing the VGG19.
  + Initializing the model
  + Adding Fully connected Layer
  + Configure the Learning Process
  + Training and Testing the model.
  + Save the Model
* Application Building
  + Create an HTML file
  + Build Python Code

# Project Structure:

Create a Project folder which contains files as shown below.



* We are building a Flask Application that needs HTML pages stored in the **templates** folder and a python script **app.py** for server-side scripting
* We need the model which is saved and the saved model in this content is a **vgg19DBPmodel.h5**
* **Templates** folder contains index.html & other html pages.
* The **Static** folder contains css & js files.

# Milestone 1: Define Problem / Problem Understanding Activity 1: Specify the business problem.

Refer Project Description

# Activity 2: Business requirements

Here are some potential business requirements for Dog Breed Identification.

# Accurate Prediction:

The predictor must be able to accurately classify the images of different Dog Breeds.

So that there will be no misclassification which decreases the accuracy of the model.

# Real-time data acquisition:

The predictor must be able to acquire real-time data from the various sources. The data acquisition must be seamless and efficient to ensure that the predictor is always up-to-date with the latest information.

# User-friendly interface:

The predictor must have a user-friendly interface that is easy to navigate and understand.

The interface should present the results of the predictor in a clear and concise manner.

# Report generation:

Generate a report outlining the predicted Dog Breeds. By analyzing data and applying advanced algorithms, the project generates detailed reports that include key findings, Dog Breed classifications, and relevant insights. The report should be presented in a clear and concise manner, with appropriate insights to help patients confirm their results.

# Activity 3: Literature Survey (Student Will Write)

A literature survey would involve researching and reviewing existing studies, articles, and other publications on the topic of the project. The survey would aim to gather information on current systems, their strengths and weaknesses, and any gaps in knowledge that the project could address. The literature survey would also look at the methods and techniques used in previous projects, and any relevant data or findings that could inform the design and implementation of the current project.

# Activity 4: Social or Business Impact.

The Dog Breed Identification project can have both social and business impacts.

# Social Impact:

Accurate dog breed identification contributes to improved dog welfare. By understanding the breed's unique needs, such as exercise requirements, grooming needs, and potential health risks, owners can provide appropriate care and ensure the well-being of their dogs. Dog breed identification promotes responsible pet ownership. Understanding the specific breed characteristics, temperament, and exercise requirements helps potential dog owners make informed decisions about which breed is best suited to their lifestyle, living situation, and family dynamics.

# Business Impact:

Pet stores and breeders can leverage dog breed identification to provide accurate information about the breeds they offer. This helps potential customers make informed decisions when choosing a dog based on their lifestyle, preferences, and compatibility. Dog breed identification can influence the development of pet services and products. Understanding specific breed characteristics allows pet service providers, such as trainers, groomers, and veterinarians, to tailor their services to meet the unique needs of different breeds. It can also inform the design and marketing of products, such as breed-specific food, toys, and accessories.

# Milestone 2: Data Collection & Image Preprocessing:

In this milestone First, we will collect images of Dog Breeds then organized into subdirectories based on their respective names as shown in the project structure. Create folders of types of Dog Breeds that need to be recognized. In this project, we have collected images of 120 types of Images like affenpinscher, beagle, appenzeller, basset, bluetick, boxer, cairn, doberman, german\_shepherd, golden\_retriever, kelpie, komondor, leonberg,

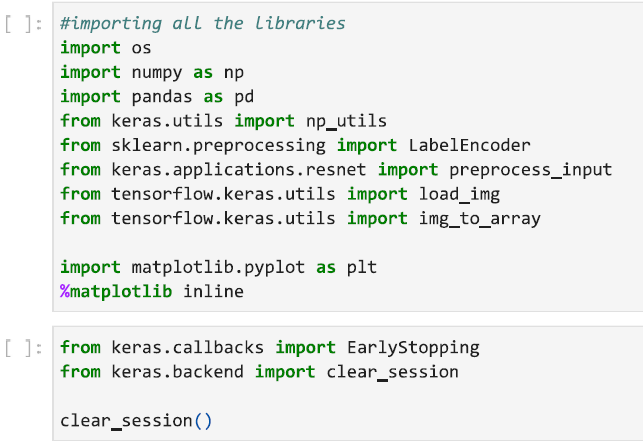
mexican\_hairless, pug, redbone, shih-tzu, toy\_poodle, vizsla, whippet they are saved in the respective sub directories with their respective names.

# Download the Dataset -

**<https://www.kaggle.com/competitions/dog-breed-identification/data?select=train>**

In Image Processing, we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc.

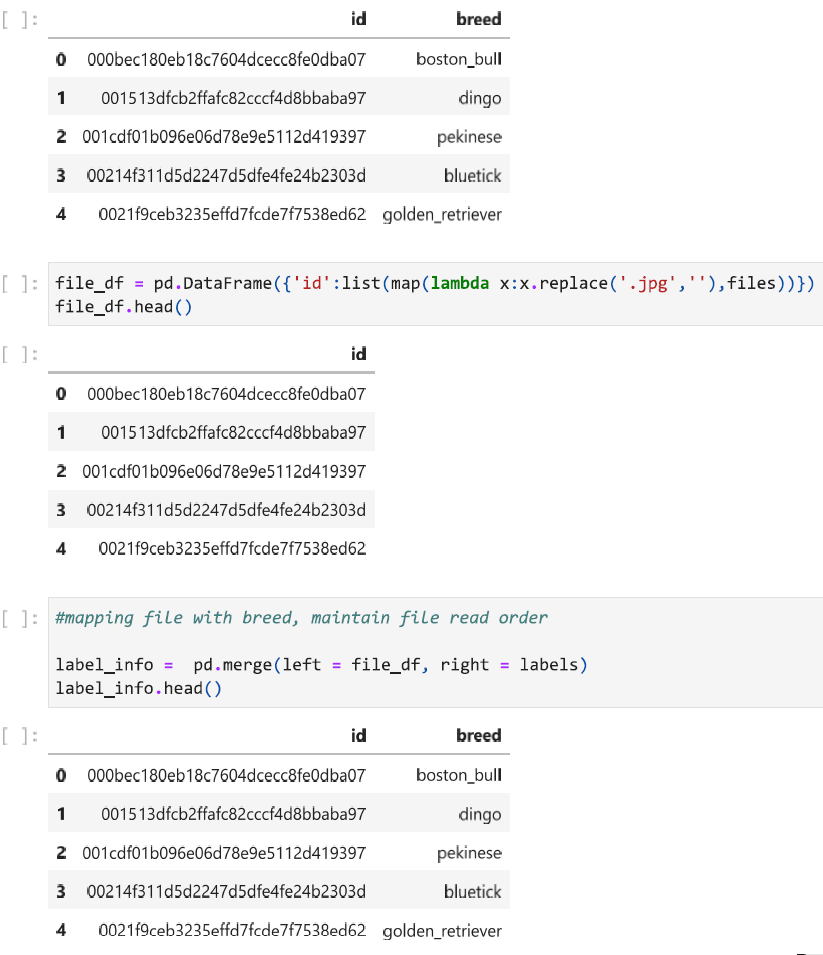
# Importing Libraries.



This section imports necessary libraries for data manipulation, image preprocessing, and model building. It includes tools for working with images, dataframes, and deep learning with Keras.

# Setting up the Data.





This part organizes the training image data by reading file names, loading target information from 'labels.csv,' and creating a DataFrame ('label\_info') that associates file names with corresponding labels

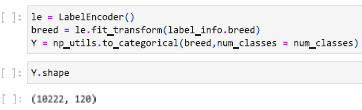
# Preprocessing and Visualization.



Here, an image is loaded and displayed for visualization purposes

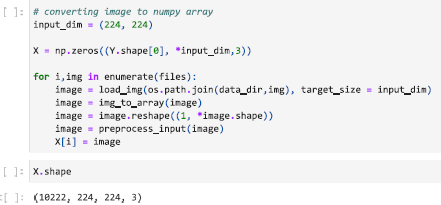
# Label Encoding

# 



This section encodes the dog breed labels into numerical format and converts them to one-hot vectors.

# Image Loading and Preprocessing



Images are loaded, resized to (224, 224), and preprocessed using the VGG19-specific preprocessing function.

# Building the VGG19 Model

# 

This section initializes the VGG19 model, removes the top layer, adds a global average pooling layer, dropout layer, and a dense output layer for 120 dog breeds.

# Freezing Layers

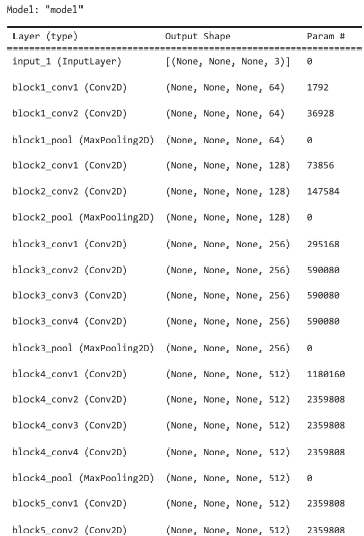
# 

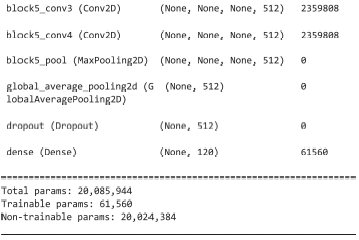
This code freezes all layers in the VGG19 model except for the last layer.

# Model Compilation

# 

This part configures the model for training using the Adam optimizer and categorical crossentropy loss.

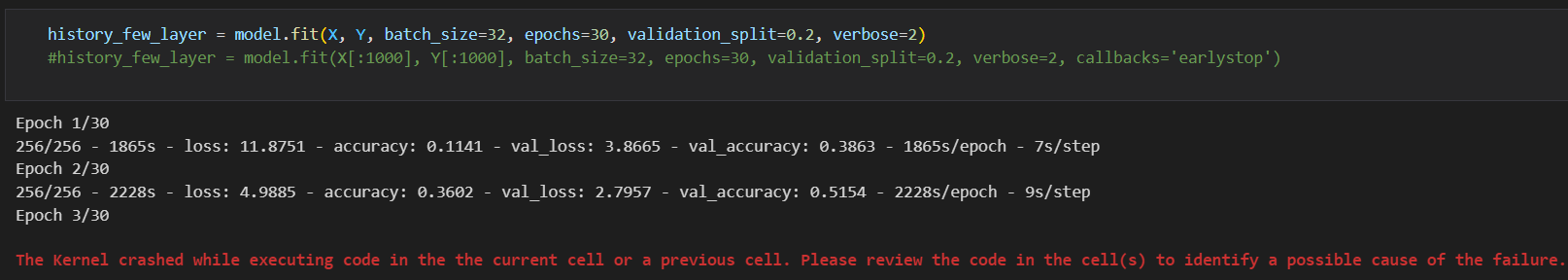




# Model Training

# 

The model is trained on the first 1000 samples for 30 epochs with a batch size of 32 and a validation split of 20%.



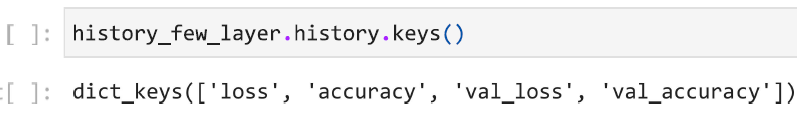
The model is trained on all samples for 30 epochs with a batch size of 32 and a validation split of 20%

# Model Saving

# 

The trained model is saved in an H5 file.

# Model Evaluation



This line retrieves keys from the training history, which can be used for further analysis or visualization.

# Milestone 4: Application Building

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.

In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the type of Colon Diseases and showcased on the

HTML page to notify the user. Whenever the user interacts with the UI and selects the “Inspect” button, the next page is opened where the user chooses the image and predicts the output.

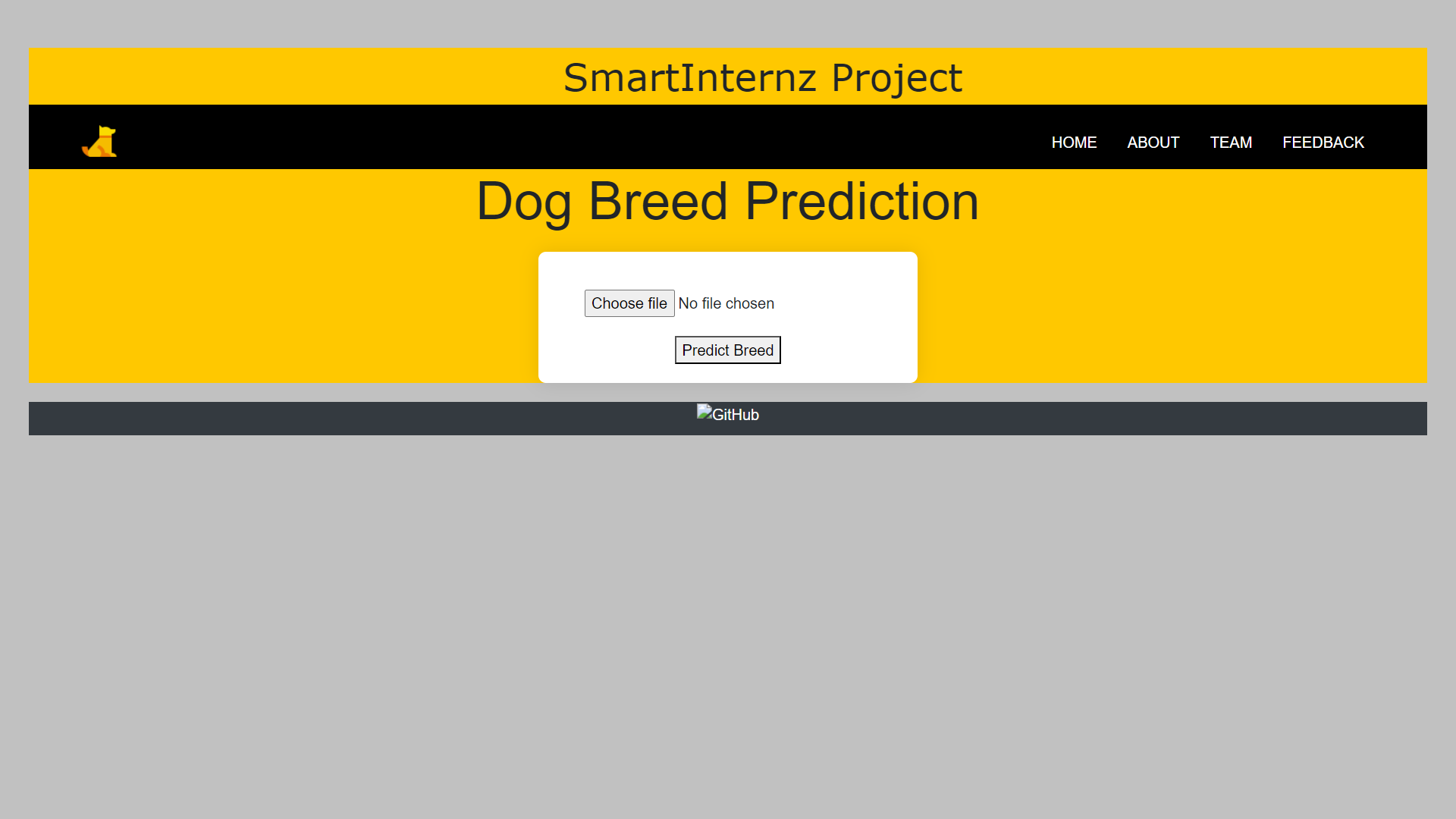
# Activity 1 : Create HTML Pages

* We use HTML to create the front end part of the web page.
* Here, we have created 3 HTML pages- index.html, about.html, feedback.html and team.html
* index.html displays the home page.
* about.html displays an introduction about the project.
* feedback.html helps us for development.
* team.html displays the team members.
* For more information regarding HTML

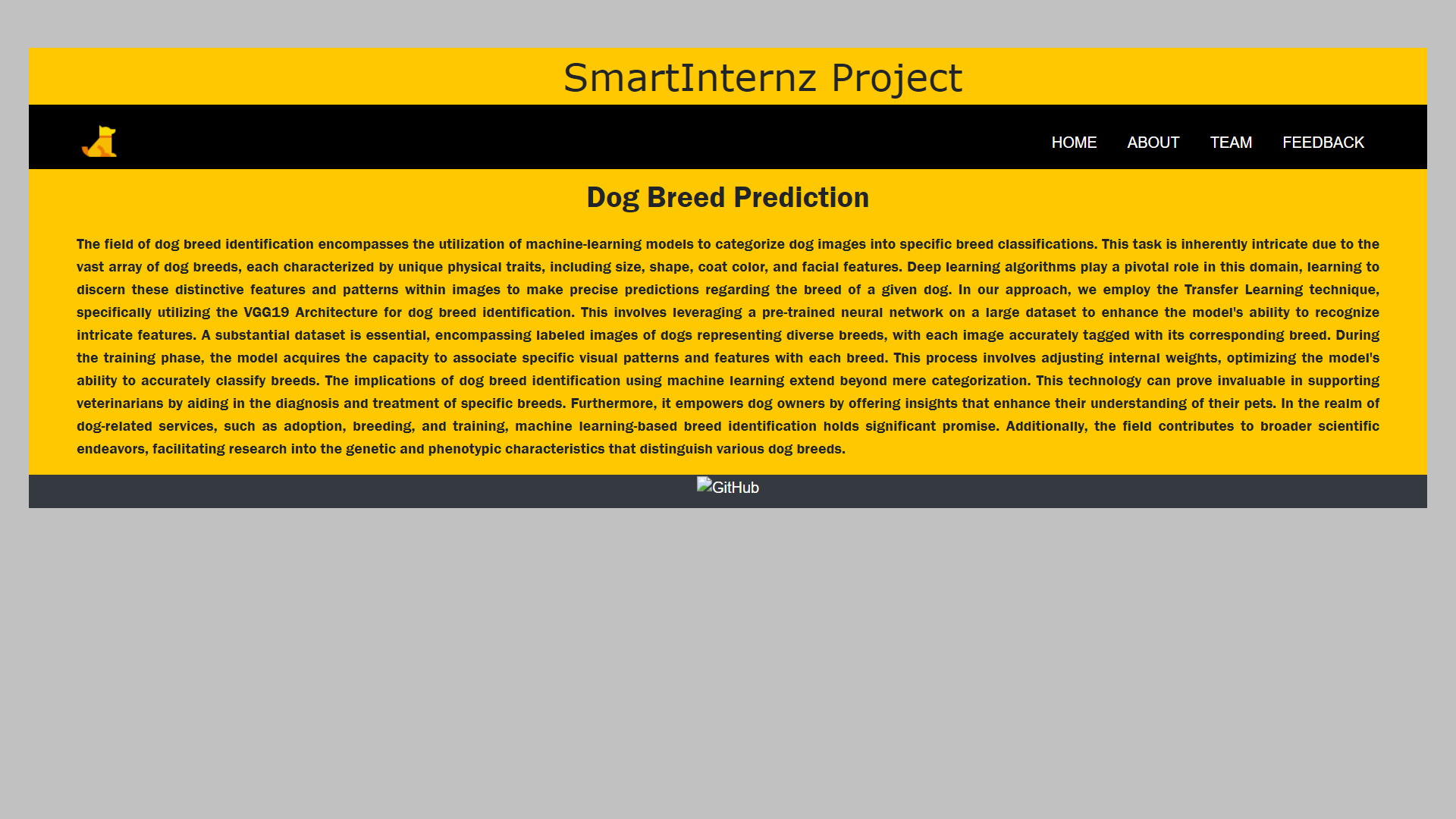
<https://www.w3schools.com/html/>

* We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.
* **Link :**[CSS](https://www.w3schools.com/css/) , [JS](https://www.w3schools.com/js/DEFAULT.asp)

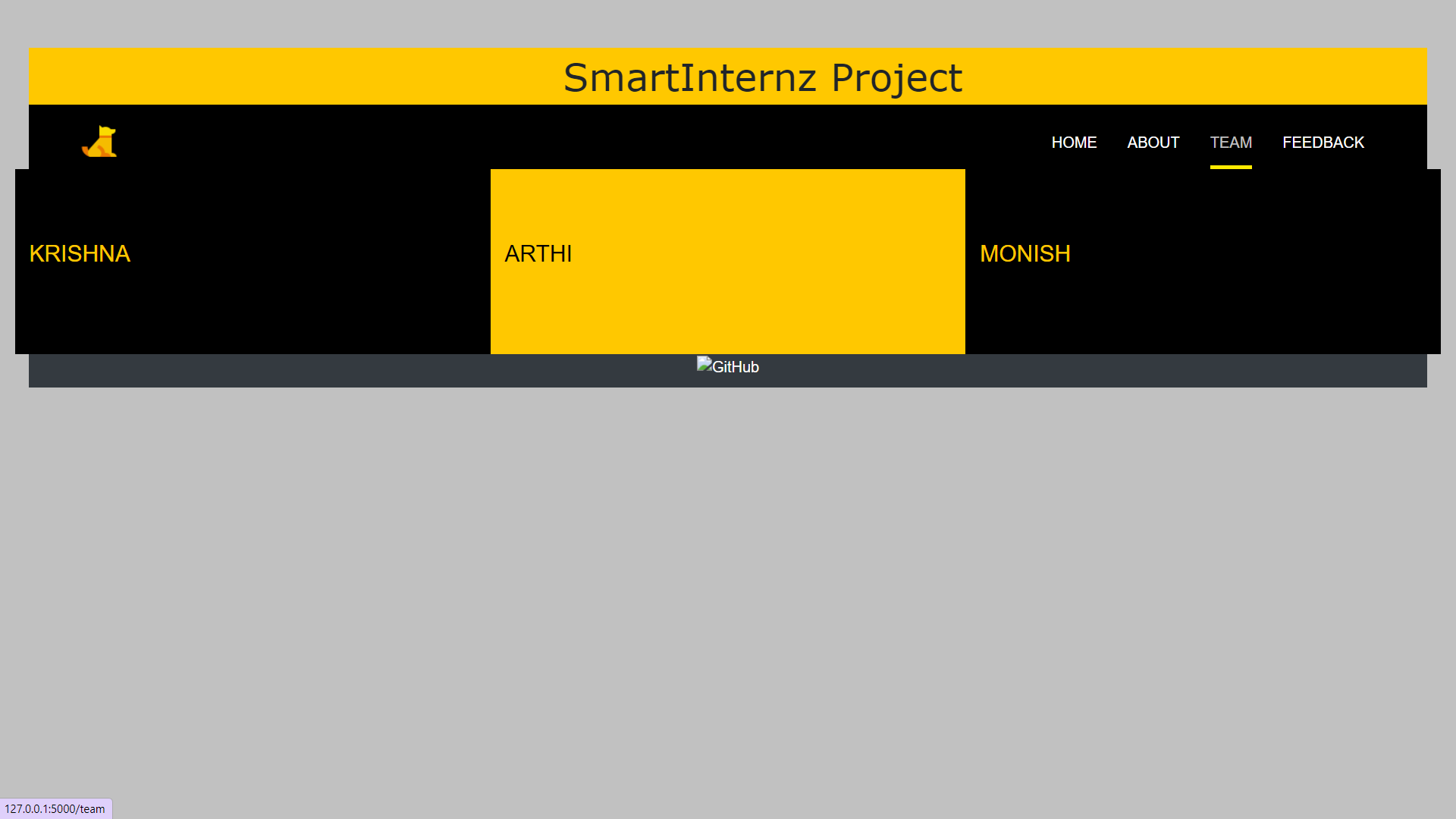
# index.html looks like this



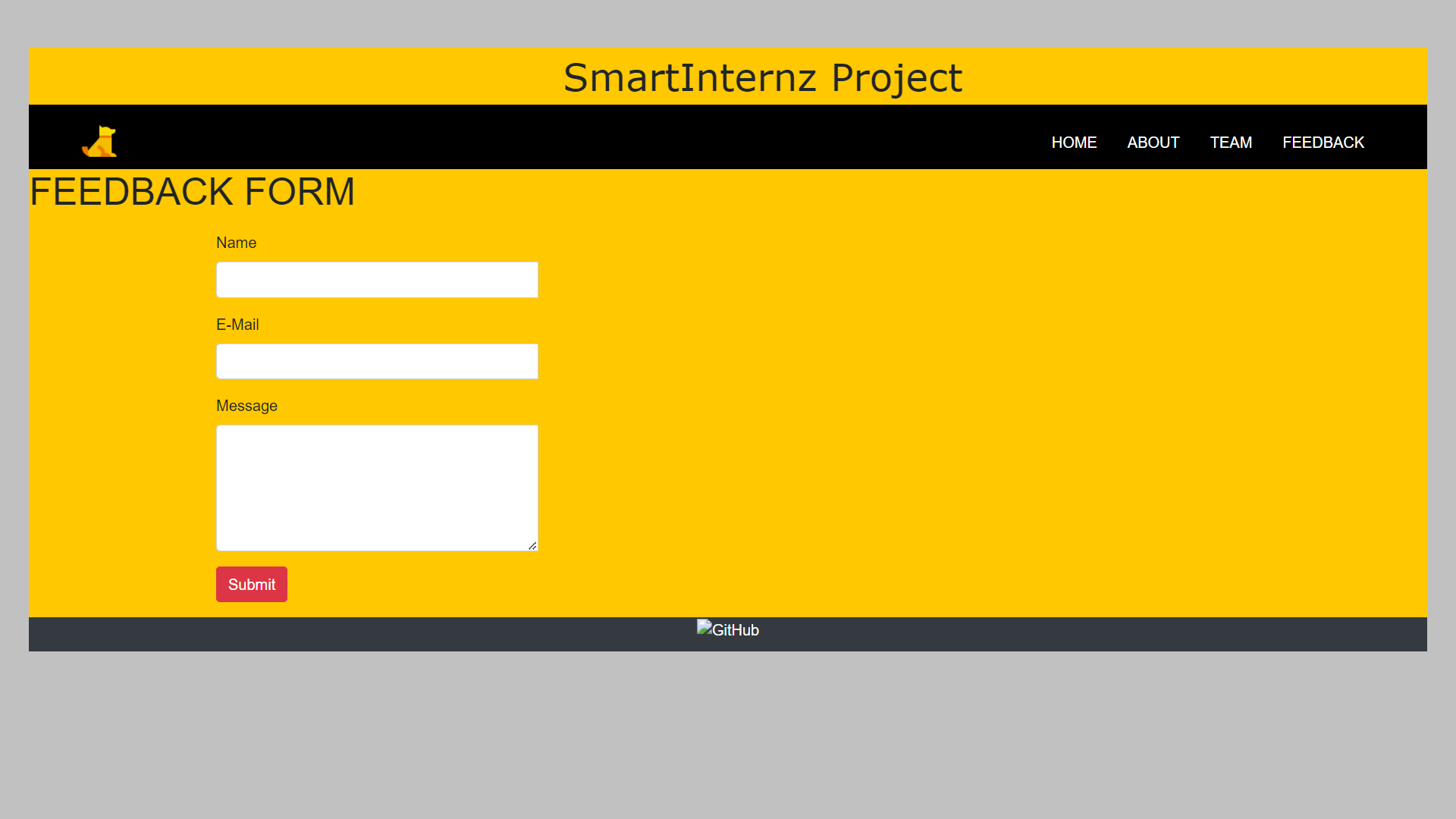
**about.html:-**



# Team.html:-



**feedback.html:-**

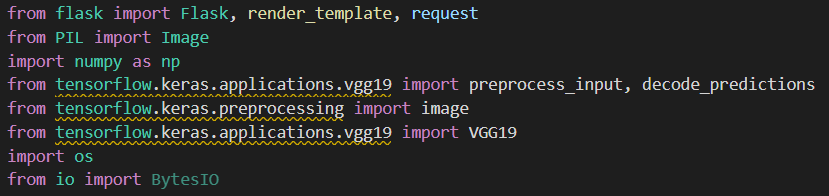


**Activity 2: Build python code**

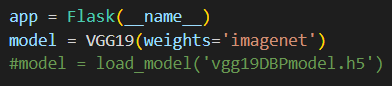
# Task 1: Importing Libraries

The first step is usually importing the libraries that will be needed in the program.

Importing the flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module ( name ) as argument Pickle library to load the model file.



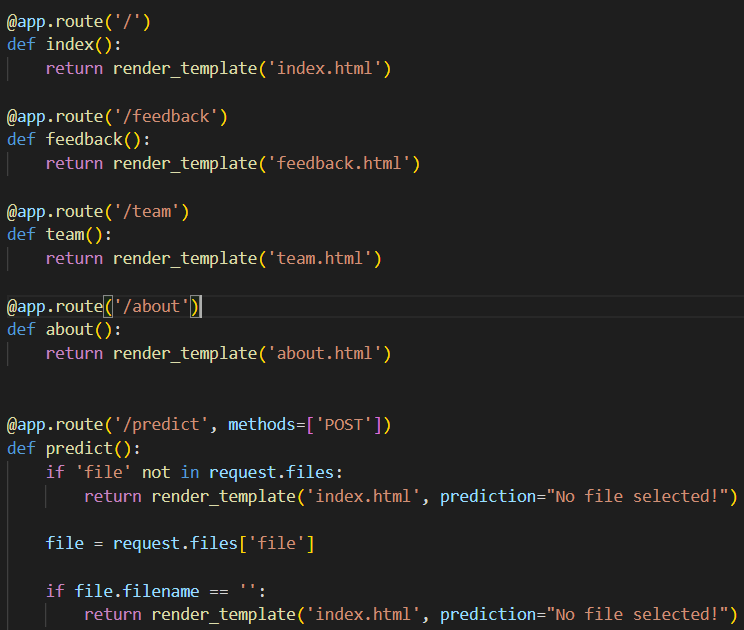
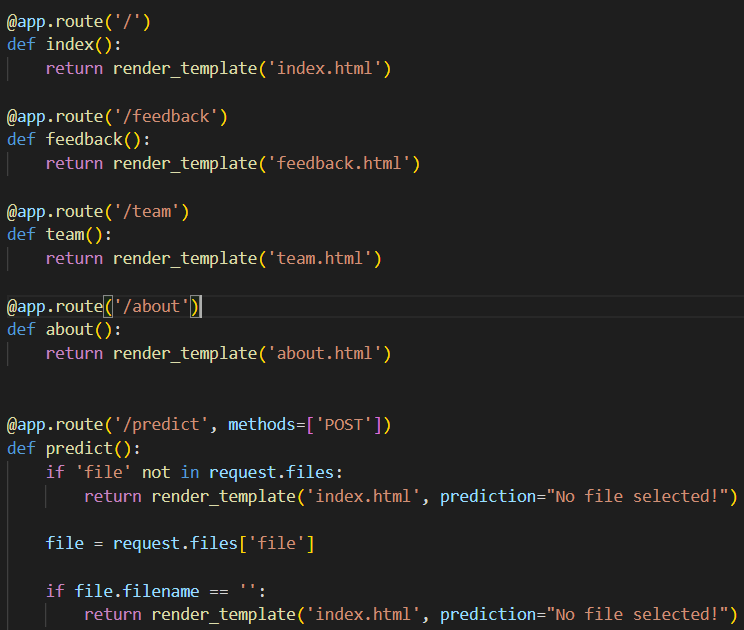
# Task 2: Creating our flask application and loading our model by using load\_model method



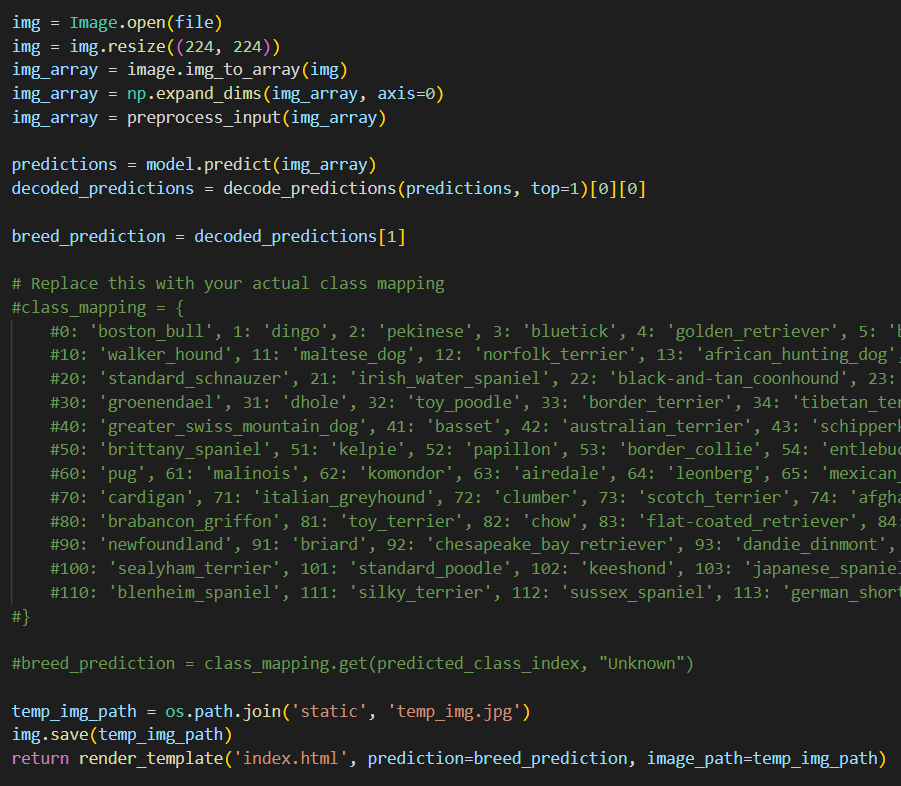
**Task 3: Routing to the html Page**

Here, the declared constructor is used to route to the HTML page created earlier.

In the above example, ‘/’ URL is bound with index.html function. Hence, when the home page of a web server is opened in the browser, the html page will be rendered. Whenever you browse an image from the html page this photo can be accessed through POST or GET Method.



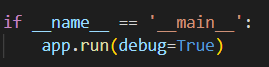
# Showcasing prediction on UI:



Here we are defining a function which requests the browsed file from the html page using the post method. The requested picture file is then saved to the uploads folder in this same directory using OS library. Using the load image class from Keras library we are retrieving the saved picture from the path declared. We are applying some image processing techniques and then sending that preprocessed image to the model for predicting the class. This returns the numerical value of a class (like 0 to 19.) which lies in the 0th index of the variable preds. This numerical value is passed to the index variable declared. This returns the name of the class. This name is rendered to the prediction variable used in the html page.

# Finally, Run the application

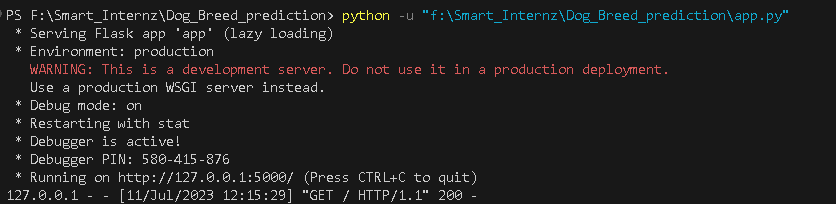
This is used to run the application in a local host.



# Activity 3:Run the application

* Open the anaconda prompt from the start menu.
* Navigate to the folder where your app.py resides.
* Now type “python app.py” command.
* It will show the local host where your app is running on **<http://127.0.0.1.5000/>**
* Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
* Enter the values, click on the predict button and see the result/prediction on the web page.

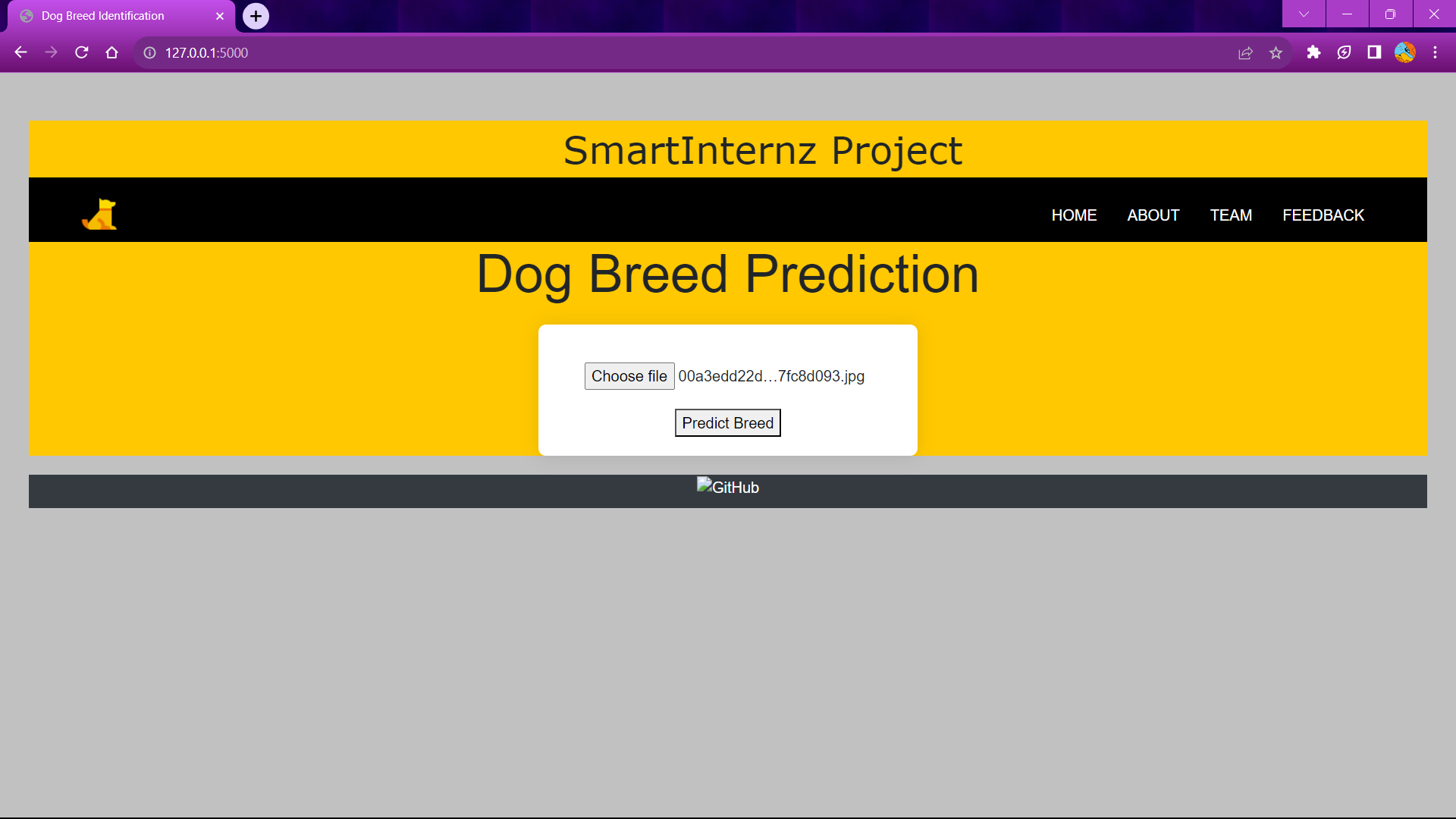
Then it will run on localhost: 5000

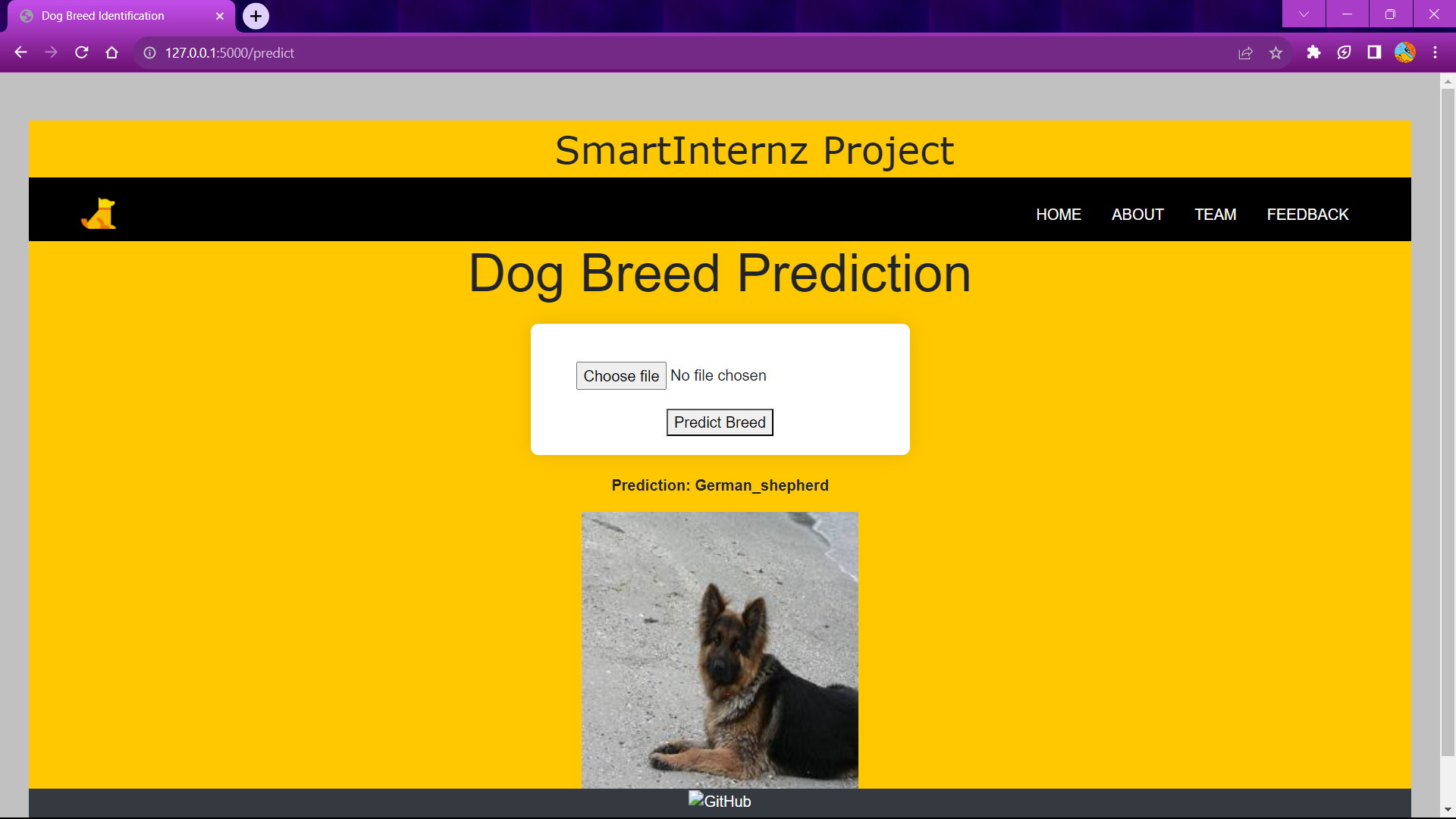


Navigate to the localhost (http://127.0.0.1:5000/)where you can view your web page.

# FINAL OUTPUT:

**Output 1:**





**Output 2:**

