**Epicurean Harnessing CNNs for**

**Hotdog Detection**

**Project Description:**

This project aims to develop an advanced image classification system that can accurately distinguish between images of “hotdogs” and & “not-hotdog”. To achieve this, we have curated a new high-quality dataset of images sourced from Google and other publicly available repositories. Our system leverages the power of Convolutional Neural Networks (CNNs), a deep learning architecture well-suited for image recognition tasks, to analyse and classify images.

Our system is designed to provide a robust and accurate hotdog detection capability, with the

potential to pave the way for more sophisticated food recognition systems in the future. The final system will be deployed in a user-friendly interface, allowing users to upload images and receive instant feedback on whether the image is a hotdog or not.

**Scenario 1:** **Quality Control in a Food Manufacturing Company**

A food manufacturing company aims to ensure the consistency and quality of its hotdog products. They plan to use a CNN model to automatically classify images of products on the production line as either hotdogs or non-hotdogs, identifying mislabelling or packaging errors. This automation can reduce manual inspection costs, minimize customer dissatisfaction due to packaging errors, and maintain high product quality.

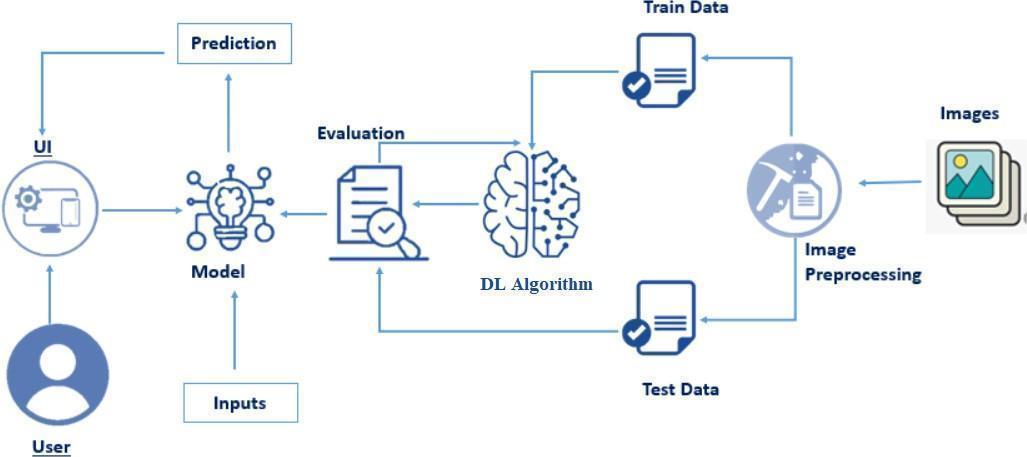
**Scenario 2:** **Enhanced Customer Experience in a Food Delivery Service**

A food delivery service, offering a variety of dishes including hotdogs, faces challenges with incorrect deliveries, leading to negative reviews. The company intends to integrate a CNN model into their app, allowing customers to scan their delivered items to verify they match the order, specifically distinguishing between hotdogs and non-hotdogs. This feature aims to improve customer satisfaction by ensuring order accuracy and addressing issues in the delivery process.

**Scenario 3:** **Social Media Content Moderation for a Food Brand**

A popular food brand, known for its hotdog products, encourages customers to share photos on social media using a branded hashtag but faces challenges with irrelevant or inappropriate content. The brand plans to deploy a CNN model to automatically classify and filter user-generated content, distinguishing images containing hotdogs from irrelevant submissions. This moderation helps maintain a curated social media presence, reinforcing the brand's identity and engaging appropriately with customers, thereby strengthening brand loyalty.

**Technical Architecture:**

****

**Prerequisites:**

**To complete this project, you must require the following software, concepts, and packages**

* **Jupyter Notebook**
* Refer to the link below to download Jupyter Notebook
* Link (Jupyter Notebook) : <https://jupyter.org/>
* Link (Spyder): https://www.youtube.com/watch?v=8YGPfGDYAgI
* **Python packages:**
* Type the below installation statements in the Jupyter notebook cells and install them.
* Type “pip install numpy” and click enter.
* Type “pip install pandas” and click enter..
* Type “pip install tensorflow==2.3.2” and click enter.
* Type “pip install keras==2.3.1” and click enter.
* Type “pip install Flask” and click enter.

**Prior Knowledge:**

You must have prior knowledge of the following topics to complete this project.

* **Deep Learning Concepts**
* **Neural Networks:** <https://www.youtube.com/watch?v=bfmFfD2RIcg>
* **CNN:** <https://www.youtube.com/watch?v=zfiSAzpy9NM>
* **Architecture :** <https://www.v7labs.com/blog/convolutional-neural-networks-guide>
* **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

Link: <https://www.youtube.com/watch?v=4L_xAWDRs7w&t=1s>

**Project Objectives:**

By the end of this project you’ll understand the following:

●       Preprocessing the images.

●       Applying Convolution Neural Networks on the dataset.

●       How deep neural networks detect the Hotdogs

●       You will be able to know how to find the accuracy of the model.

●       You will be able to Build web applications using the Flask framework.

**Project Flow:**

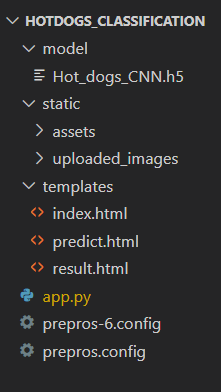
* The user interacts with the UI (User Interface) to choose the image.
* The chosen image is analysed by the model which is integrated with flask application.
* The Model analyses the image, then the prediction is showcased on the Flask UI.

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

* Data Collection.
* Create a Train and Test path.
* Data Pre-processing.
* Import the required library
* Configure ImageDataGenerator class
* Apply ImageDataGenerator functionality to Trainset and Test set
* Model Building
* Convolution and Max Pooling
* Flattening
* Neural Network Connection
* Training the Model
* Save the Model
* Test the Model
* Application Building
* Create an HTML file
* Build Python Code

**Project Structure:**

Create a Project folder that contains files as shown below



* The Data folder contains the training, testing and Validation Images for training our model.
* We are building a Flask Application that needs HTML pages stored in the **templates folder.**
* Python script **app.py** for server-side scripting
* we need the model that is saved and the saved model in this content is a **Hot\_dogs\_CNN.h5**
* templates folder contains index.html, predict.html & result.html pages.

**Milestone 1: Data Collection**

  There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

**Activity 1: Download the dataset**

It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used hotdog-silicon-valley data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

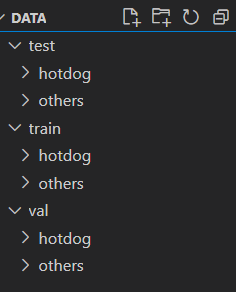
Link: <https://www.kaggle.com/datasets/antareepdey/hotdog-silicon-valley>

As the dataset is downloaded. Let us understand the data properly.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques

   We are going to build our training model on Jupyter Notebook.

Download the dataset view the Folder structure and folders present in the Data.



This is the Structure of the Data folder where we have test, train and validation data.

**Activity 2: Create training and testing dataset**

To build a DL model we have six classes in our dataset. But In the project dataset folder training and testing data are needed. So, in this case, we just have to assign a variable and pass the folder path to it.

Three different transfer learning models are used in our project and the best model is selected.

The image input size of the model is 64,64.

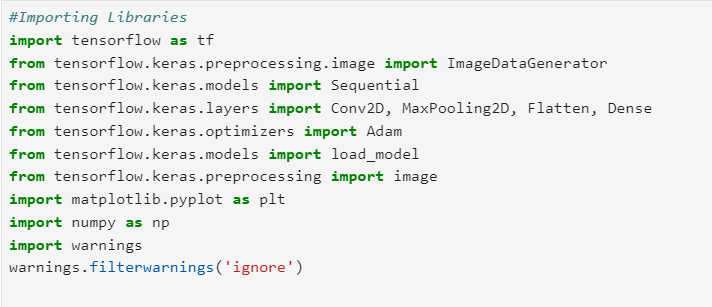
**Milestone 2: Image Preprocessing**

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

**Activity 1: Importing the libraries**

Import the necessary libraries as shown in the image

.



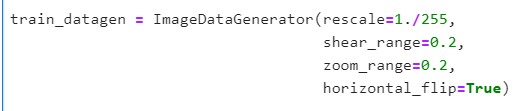
**Activity 2: Configure ImageDataGenerator class**

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

* Image shifts via the width\_shift\_range and height\_shift\_range arguments.
* The image flips via the horizontal\_flip and vertical\_flip arguments.
* Image rotations via the rotation\_range argument
* Image brightness via the brightness\_range argument.
* Image zoom via the zoom\_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.



**Activity 3: Apply ImageDataGenerator functionality to Training\_data and Validation\_data**

Let us apply ImageDataGenerator functionality to the Training\_data and Validation\_data by using the following code. For Training set keras. preprocessing.image\_dataset\_from\_directory function.

This function will return batches of images from the subdirectories

Arguments:

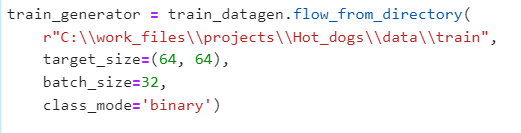
* directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
* batch\_size: Size of the batches of data which is  16.
* target\_size: Size to resize images after they are read from disk.
* class\_mode:

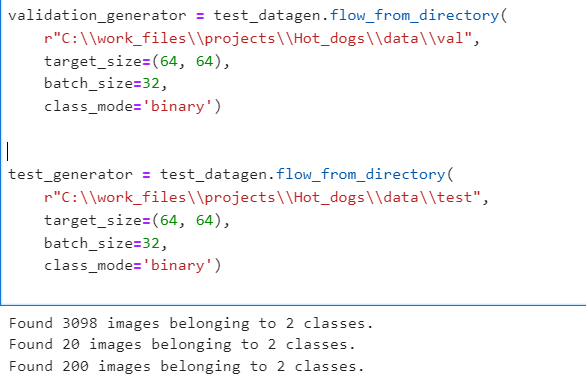
-  ‘int': means that the labels are encoded as integers (e.g. for sparse\_categorical\_crossentropy loss).

- 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical\_crossentropy loss).

- 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary\_crossentropy).

 - None (no labels).





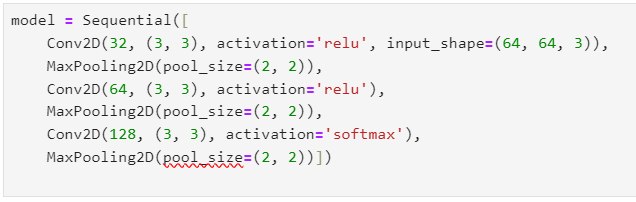
**Milestone 3: Model Building**

Now it's time to build our model. Deep understanding of the model – Link is referred to in the prior knowledge section. Kindly refer to it before starting the model-building part.

**Activity 1: Convolution and Max Pooling**

For one of the models we CNN model using the Sequential API in Keras, designed for input images of size 64x64 pixels with 3 color channels (RGB). It includes three convolutional layers with 32, 64, and 128 filters, respectively, and each uses a 3x3 kernel size. The first two convolutional layers use the ReLU activation function, while the third uses Soft Max.Here, we have considered images of dimension (64,64,3).

Flatten layer flattens the input. Does not affect the batch size.



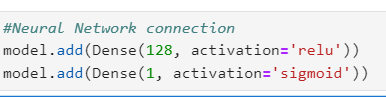
**Activity 2: Flattening**

Flattening converts the matrix in to single dimension array and can be used for building the Neural Network

****

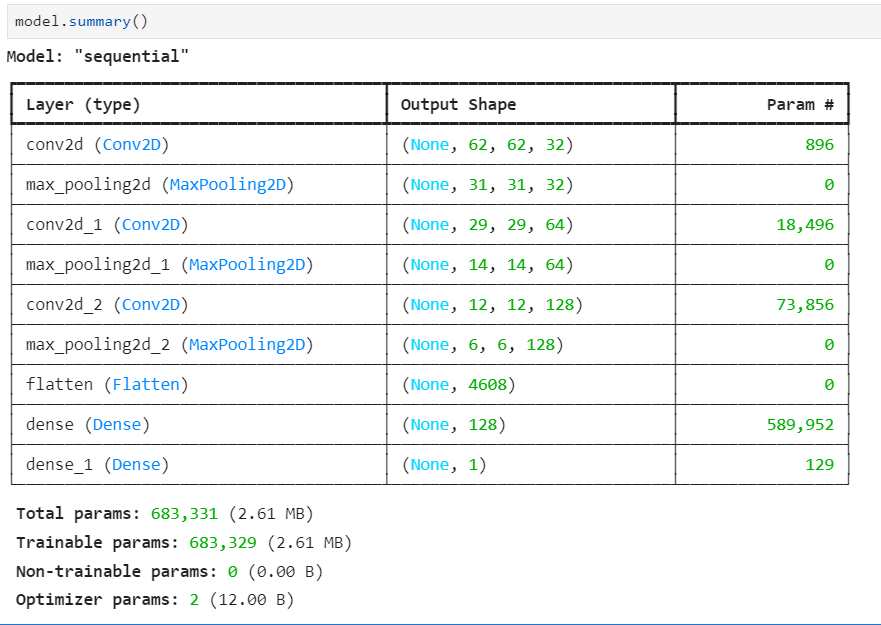
**Activity 3: Neural Network Connection**

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.



There are 128 neurons present in the filter layers with the activation function as reLU and from the last neuron we will get the value between 0 to 1 so to diffentiate both the categories we use sigmoid activation function.

Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, and summary to get the full information about the model and its layers.

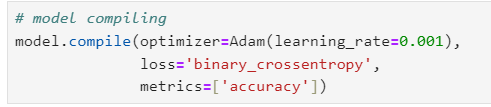


**Activity 3: Configure the Learning Process**

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.

Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using Adam Optimizer

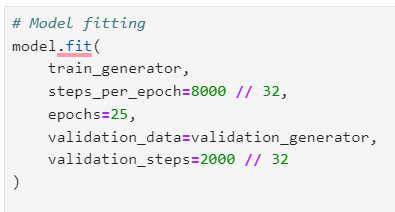
Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process



**Activity 4: Train the model**

Now, let us train our model with our image dataset. The model is trained for 25 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 20 epochs and probably there is further scope to improve the model.

**Fit** function is used to train a deep-learning neural network



**Arguments:**

* steps\_per\_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of     steps\_per\_epoch as the total number of samples in your dataset divided by the batch size.
* Epochs: an integer and number of epochs we want to train our model for.
* validation\_data can be either:

                      - an inputs and targets list

                      - a generator

                      - an inputs, targets, and sample\_weights list which can be used to evaluate

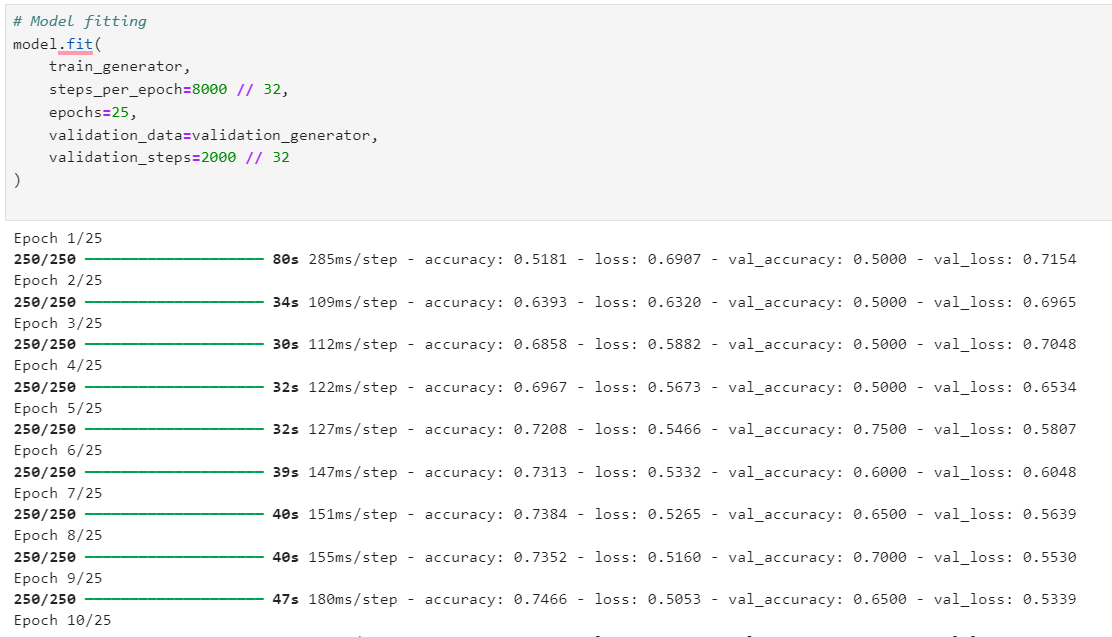
                        the loss and metrics for any model after any epoch has ended.

* validation\_steps: only if the validation\_data is a generator then only this argument

can be used. It specifies the total number of steps taken from the generator before it is

stopped at every epoch and its value is calculated as the total number of validation data points

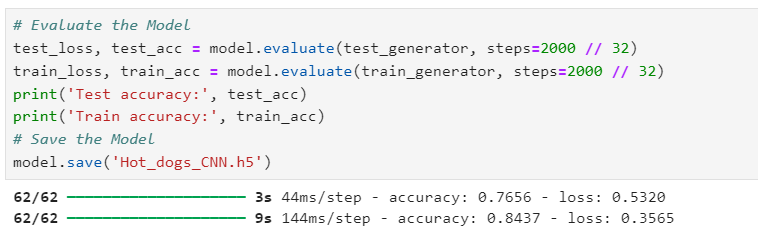
in your dataset divided by the validation batch size.



From the above run time, we can easily observe that at 10 epochs the model is giving better accuracy.

**Activity 5: Save the Model**

Saving the model after calculating the train and test accuracy.

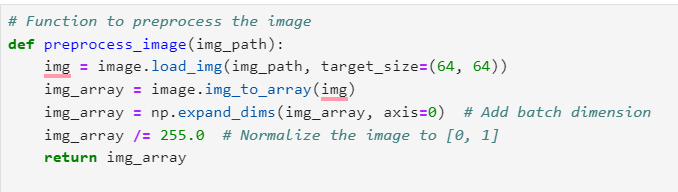


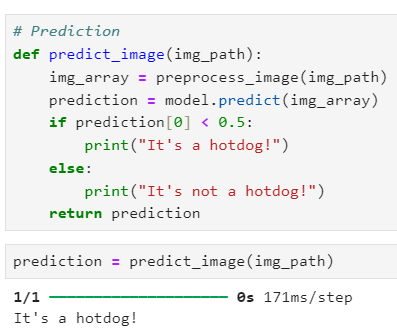


**Activity 8: Testing the Model**

Model testing is the process of evaluating the performance of a deep learning model on a dataset that it has not seen before. It is a crucial step in the development of any machine learning model, as it helps to determine how well the model can generalize to new data.







* In the above code, we have tested the model with an image of a hotdog

**Milestone 4: Save the Model**

The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.



**Milestone 5: Application Building**

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

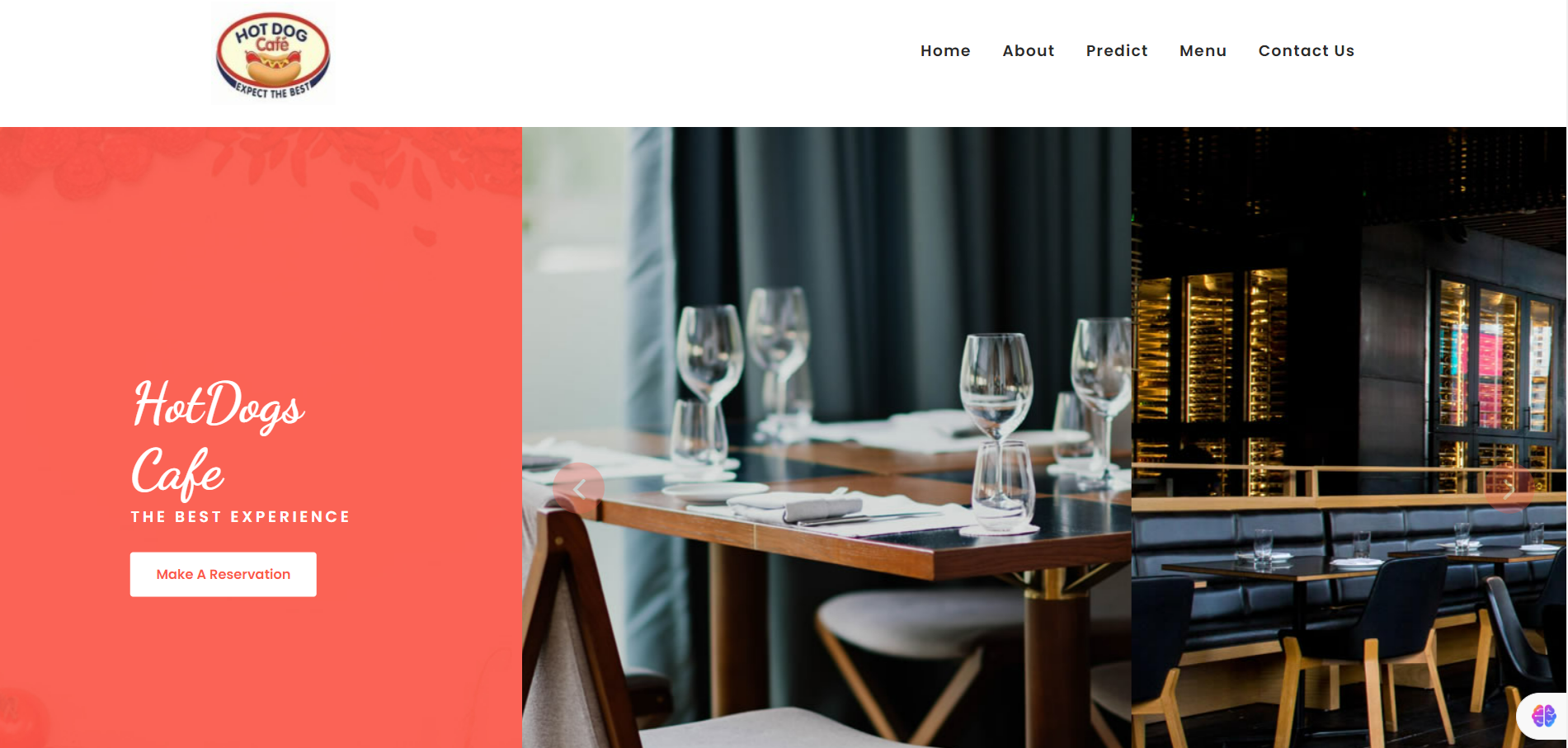
* Building HTML Pages
* Building server-side script

**Activity1: Building Html Pages:**

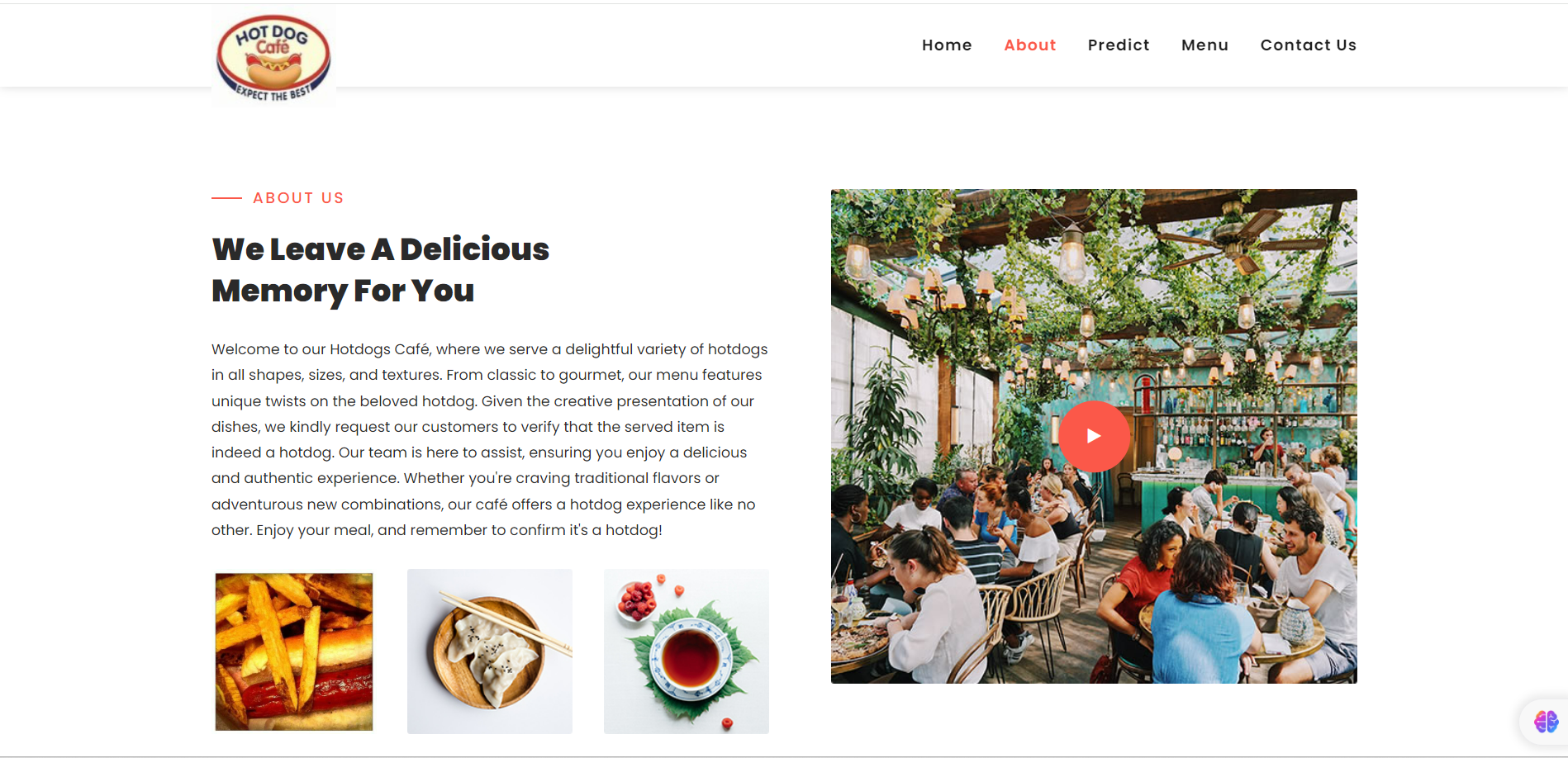
For this project create one HTML file namely

* Index.html

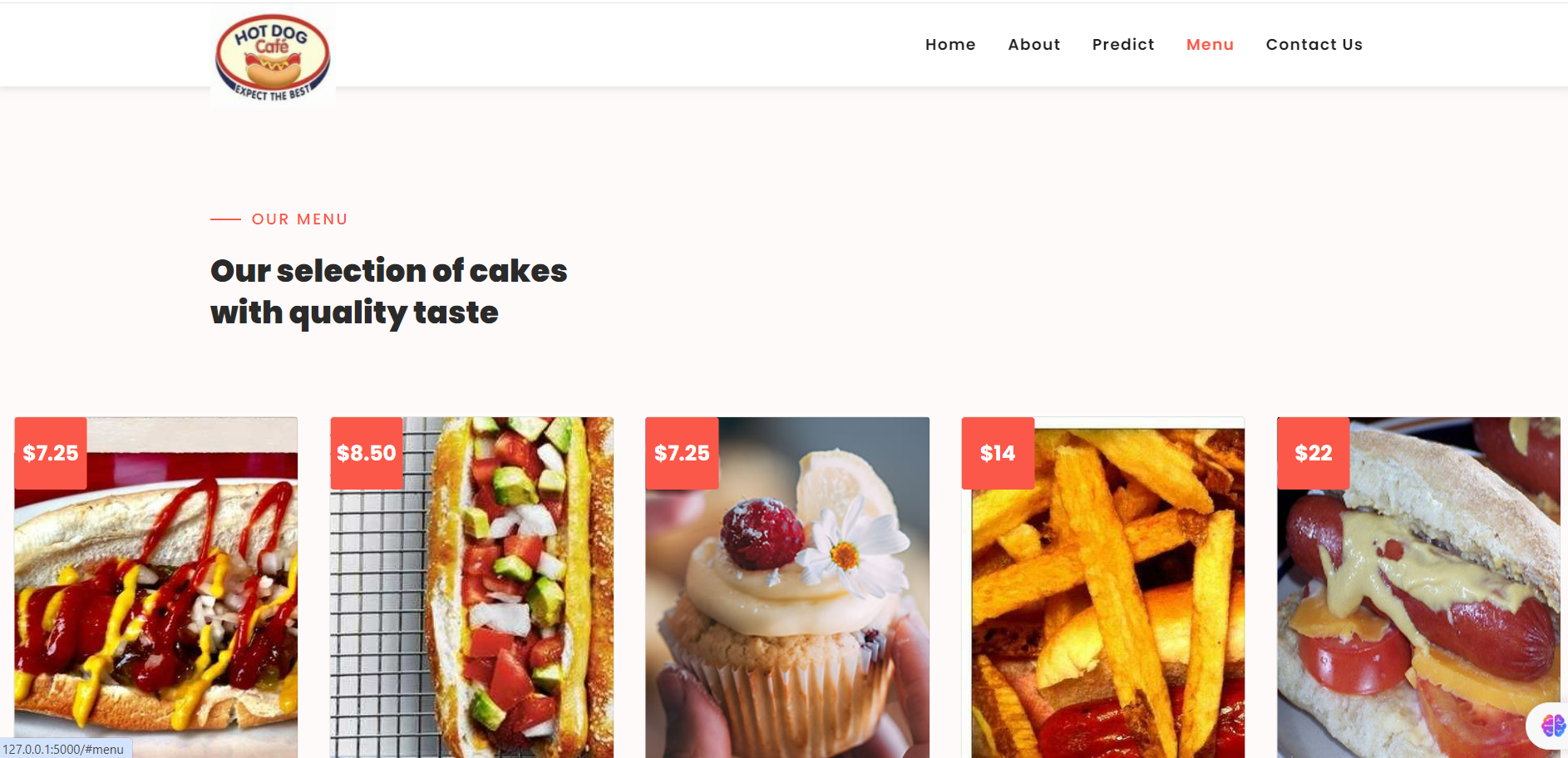
Let’s see how our index.html page looks like:



When you click on about button on the top , you will be redirecting to the following page



When you click on menu button on the Nav bar , you will be redirecting to the following page



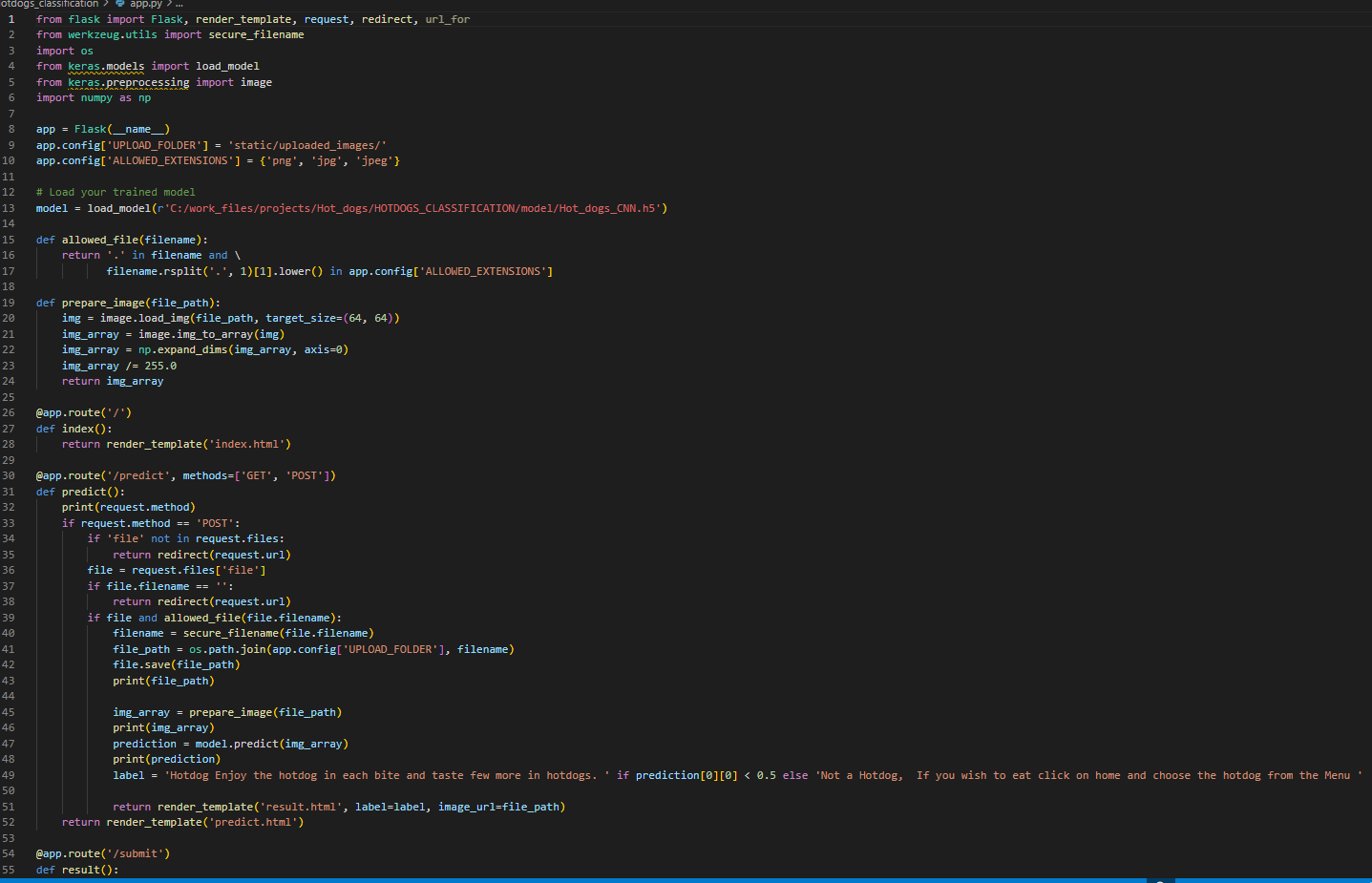
**Activity 2: Build Python code:**

* Import the libraries
* Loading the saved model and initializing the Flask app
* Render HTML pages:
* Once we upload the file into the app, then verifying the file uploaded properly or not. Here we will be using the declared constructor to route to the HTML page that we have created earlier.
* In the above example, ‘/’ URL is bound with home.html function. Hence, when the home page of the web server is opened in the browser, the HTML page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

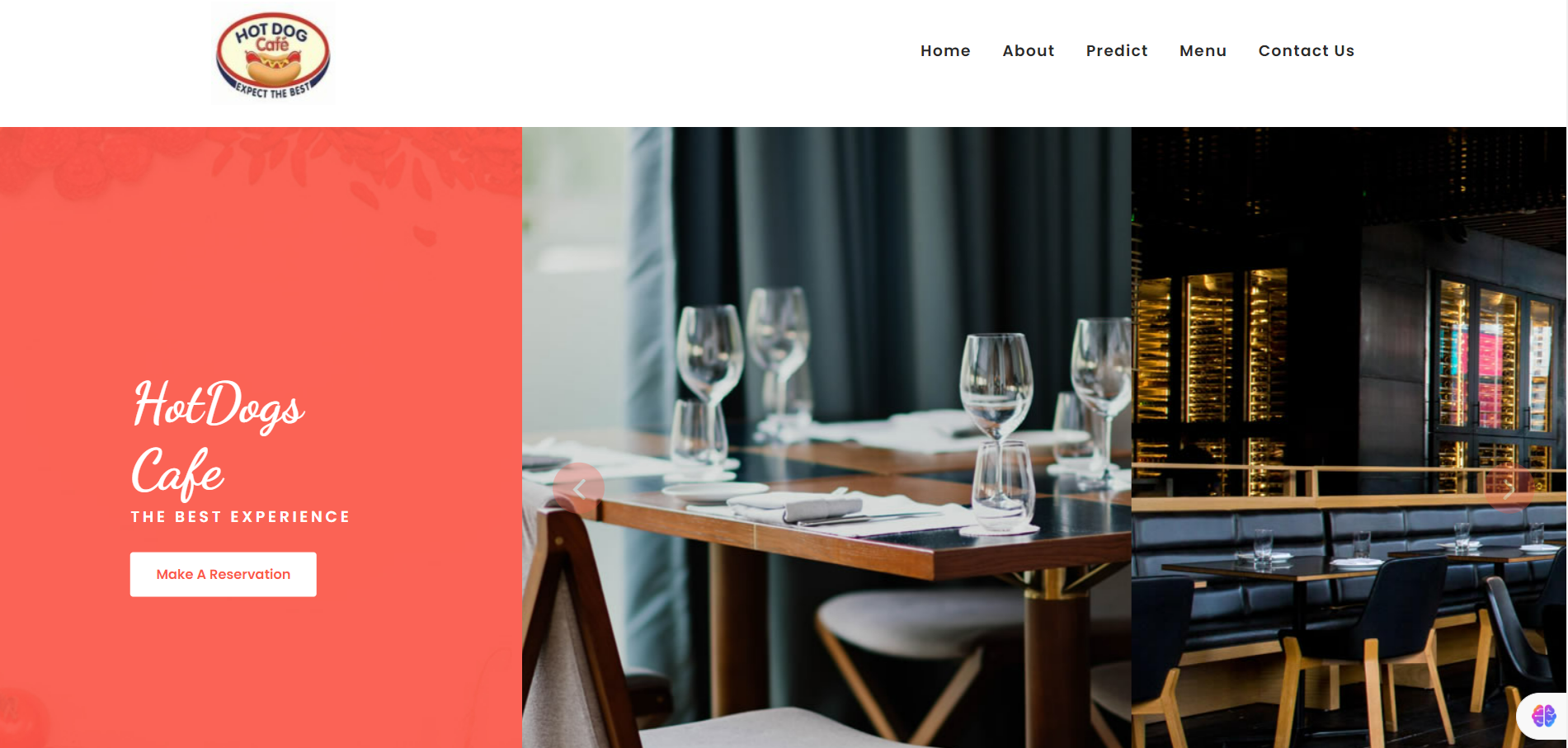
Here we are routing our app to res function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. This prediction value will rendered to the text that we have mentioned in the index.html page earlier.

**Activity 3: Run the application**

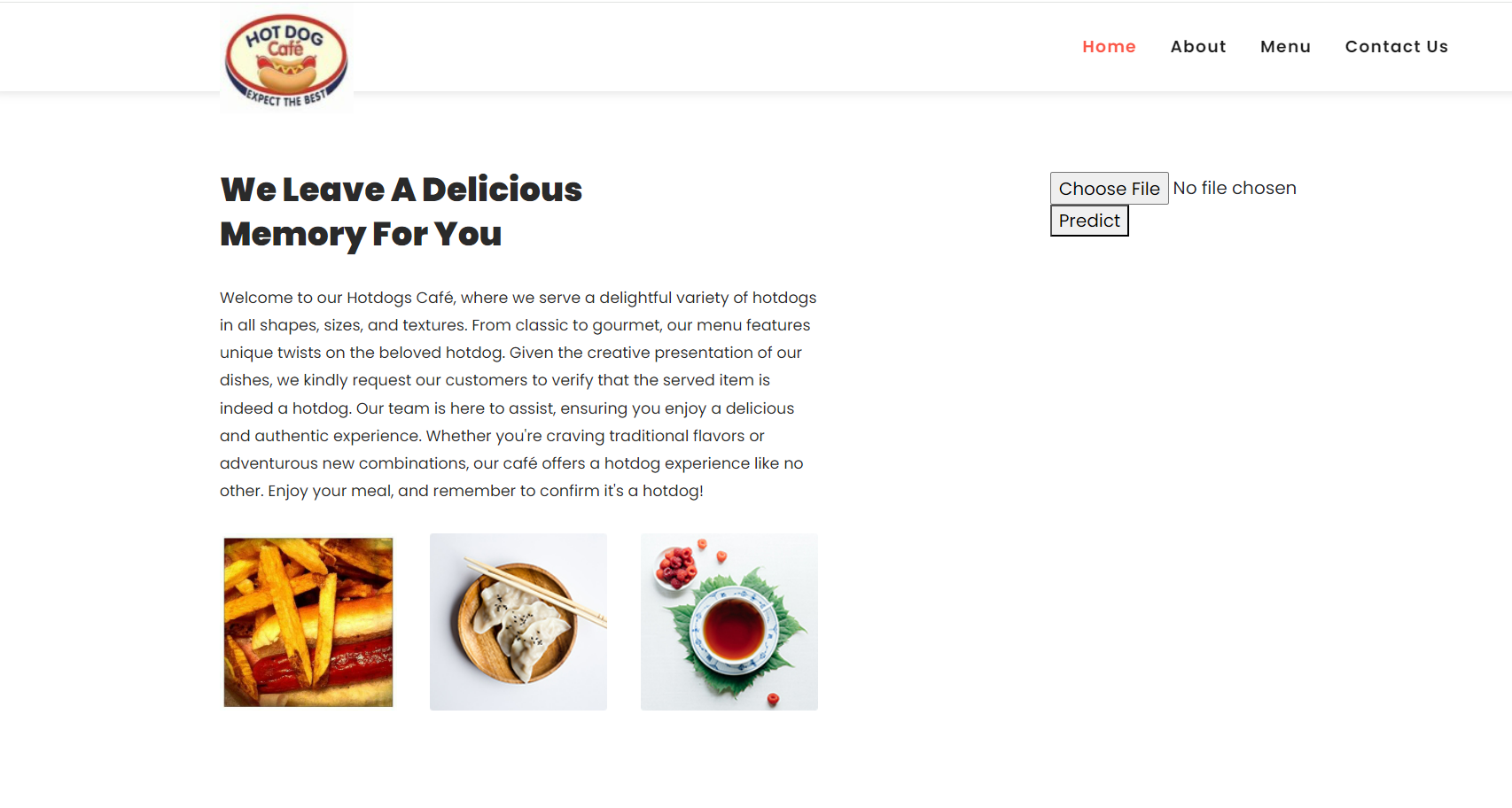
* Open the Anaconda prompt from the start menu.
* Navigate to the folder where your Python script is.
* Now type the “python app.py” command.
* Navigate to the localhost where you can view your web page.
* Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



click on Predict button



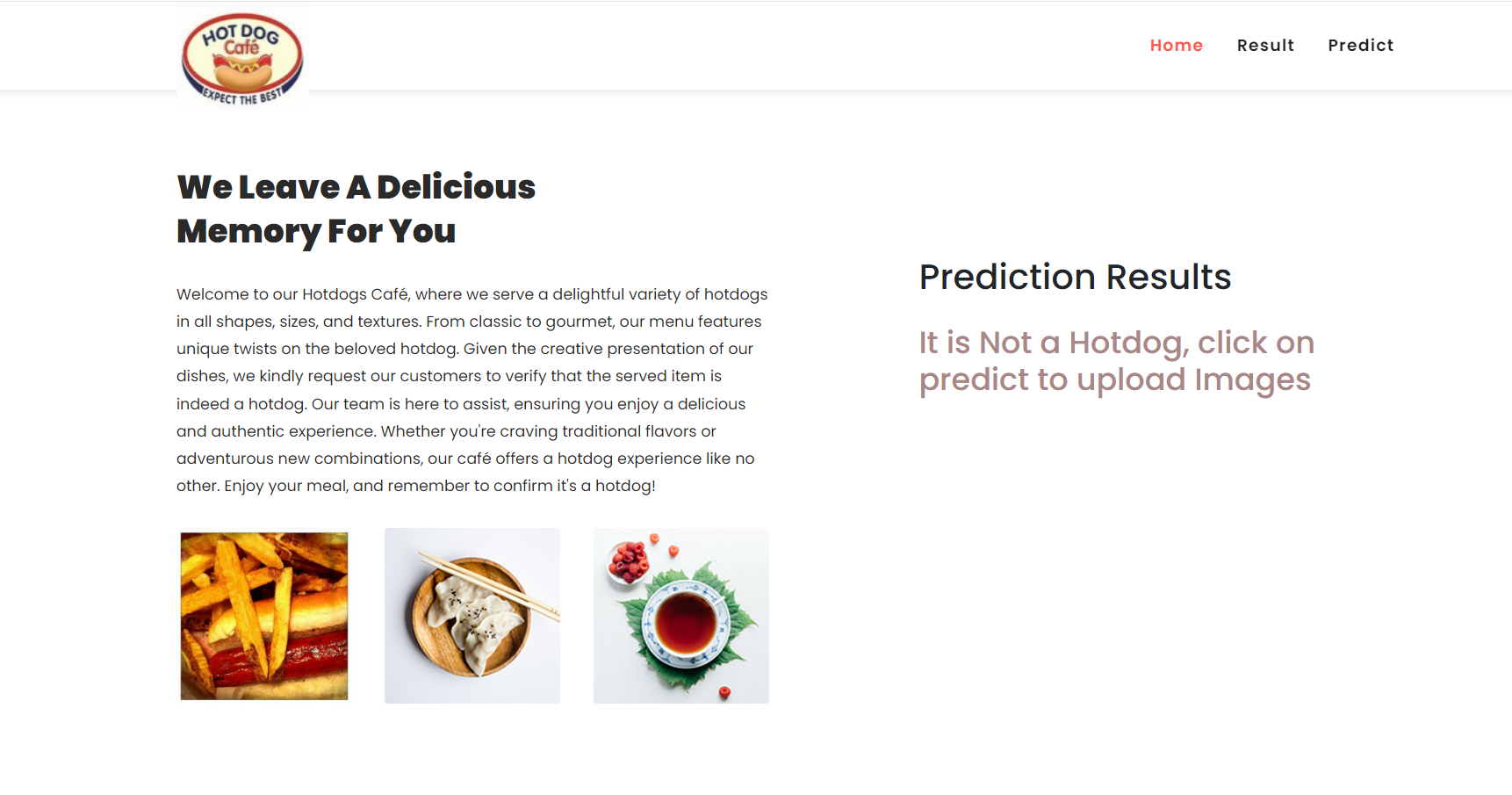
Once you upload the image and click on the submit button, the output will be displayed on the below page



Input 1:



Output:1



Input:2



Output:2

