DETECTING BUILDING DEFECTS USING VGG16

A UG PROJECT PHASE -1 REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

VADICHERLA NARESH	**18UK1A05H7**
LAKKARSU SHIVA	**18UK1A05K6**
SAI REVANTH METTELA	**18UK1A05M0**
BOMMA KAVYA	**18UK1A05J4**

Under the guidance of

Mr. V. RANADHEER REDDY

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE
WARANGAL

CERTIFICATE OF COMPLETION UG PROJECT PHASE -1

This is to certify that the UG Project Phase – 1 project report entitled **“DETECTING BUILDING DEFECTS USING VGG16”** is being submitted by **V. NARESH (18UK1A05H7), L. SHIVA (18UK1A05K6), M. SAI REVANTH (18UK1A05M0), B. KAVYA (18UK1A05J4),** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2021-2022.

Project Guide **HOD**

Mr. V. RANADHEER REDDY **Dr. R. NAVEEN KUMAR**

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. Prasad Rao**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG project phase -1 in the institute.

We extend our heartfelt thanks **to Dr. R. Naveen Kumar**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG project phase -1.

We express heartfelt thanks to **Miss. Sri Tulasi**, AI Developer, Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the major project and for their support in completing the UG project phase -1.

We express heartfelt thanks to the Major Project Guide**, Mr. V. Ranadheer Reddy**, Assistant Professor, Department of CSE for his constant support and giving necessary guidance for completion of the UG project phase -1.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

ABSTRACT

Detection of defects including cracks and flakes on the wall surfaces in high-rise buildings is a crucial task of buildings maintenance. If left undetected and untreated, these defects can significantly affect the structural integrity and the aesthetic aspect of buildings, Time and cost-effective methods of building condition survey are of practicing need for the building owners and maintenance agencies to replace the time- and labor-consuming approach of the manual survey.

Clients are increasingly looking for fast and effective means to quickly and frequently survey and communicate the condition of their buildings so that essential repairs and maintenance work can be done in a proactive and timely manner before it becomes too dangerous and expensive. Traditional methods for this type of work commonly comprise of engaging building surveyors to undertake a condition assessment which involves a lengthy site inspection to produce a systematic recording of the physical condition of the building elements, including cost estimates of immediate and projected long-term costs of renewal, repair, and maintenance of the building.

In this project detecting building defects such as cracks, flakes, and roof defects, we are using CNN pre-trained model VGG16 to analyse the type of building defect on the given parameters. The objective of the project is to build an application to detect the type of building defect. The model uses an integrated webcam to capture the video frame and the video frame is compared with the pre-trained model and the type of building defect is identified and showcased on the OpenCV window and emergency pull is initiated.

Keywords: Deep Learning, VGG16 model, CNN layers, Image Preprocessing, Building Defects like Cracks, Flakes, Roofs.

CONTENTS

LIST OF CHAPTERS **Page No**

LIST OF FIGURES

1. INTRODUCTION

Clients with multiple assets are increasingly requiring intimate knowledge of the condition of each of their operational assets to enable them to effectively manage their portfolio and improve business performance. This is being driven by the increasing adverse effects of climate change, demanding legal and regulatory requirements for sustainability, safety and well-being, and increasing competitiveness. Clients are looking for fast and effective means to quickly and frequently survey and communicate the condition of their buildings so that essential maintenance and repairs can be done in a proactive and timely manner before it becomes too dangerous and expensive. Traditional methods for this type of work commonly comprise of engaging building surveyors to undertake a condition assessment which involves a lengthy site inspection resulting in a systematic recording of the physical conditions of the building elements with the use of photographs, note taking, drawings and information provided by the client. The data collected are then analysed to produce a report that includes a summary of the condition of the building and its elements. This isalso used to produce estimates of immediate and projected long-term costs of renewal, repair and maintenance of the building. Image analysis techniques for detecting defects have been proposed as an alternative to the manual on-site inspection methods. While the latter is time-consuming and not suitable for quantitative analysis, image analysis-based detection techniques, on the other hand, can be quite challenging and fully dependent on the quality of images taken under different real-world situations (e.g., light, shadow, noise, etc.)

A brief discussion of a selection of the most common defects that arise from the presence of moisture and dampness in buildings is presented. This is followed by a brief overview of CNN. These are a class of deep learning techniques primarily used for solving fundamental problems in computer vision. This provides the theoretical basis of the work undertaken. We propose a deep learning-based detection and localisation model using transfer learning utilising the VGG-16 model for feature extraction and classification. Next, we briefly discuss the localisation problem and the class activation mapping (CAM) technique which we incorporated with our model for defect localisation.

Based on visibility, defects in any building can be of two major types:

- Defects discernible to the naked eye (External)
- Defects not discernible to the naked eye (Internal)

In this project, our focus has been on the externally discernible defects on a building.

Based on the dependence on structure, building defects can be classified as:

- Structural Defects (Affects structural elements)
- Non-structural Defects (Affects non-structural elements)

While structural defects affect the overall stability of the building (roofing, external walls etc.), non-structural defects relate to the aesthetic appearance of buildings (paint, plastering etc.).

Some of the defects that are discernible to the human eye, according to are as follows:

- **Cracks** - Often an effect of ageing of buildings, can be very dangerous if not dealt with at an early stage.

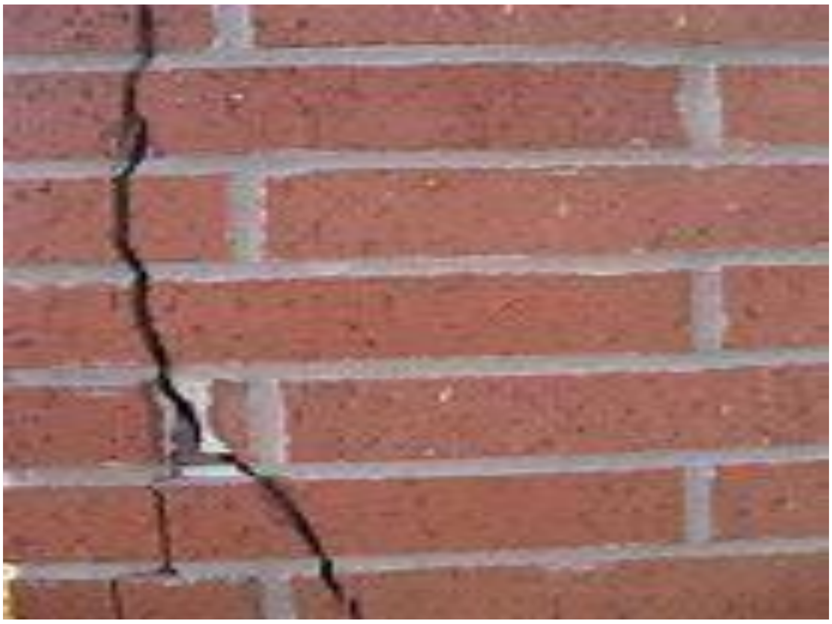


Figure 1. Cracks

- **Flakes** – A small, flat, very thin piece of something, typically been peeled off from a larger piece.

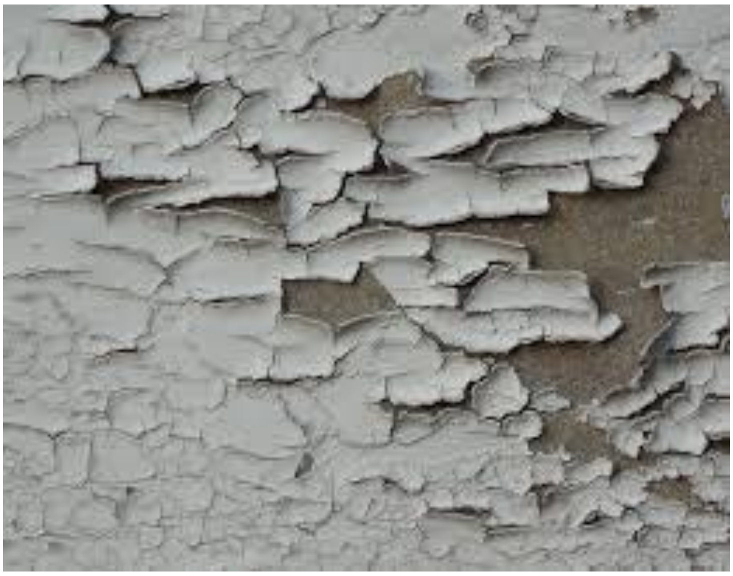


Figure 2. Flakes

- **Roof Sagging** - Sagging roofs are dangerous and need to be identified early to prevent roof collapse.

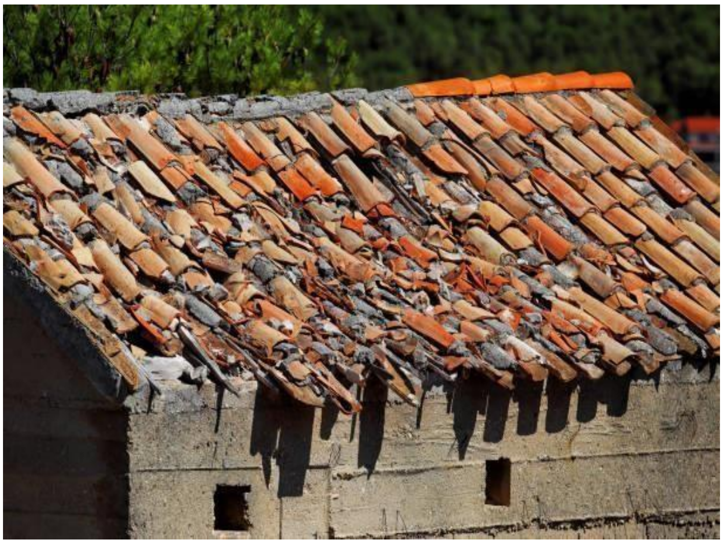


Figure 3. Roofs

1.1. MOTIVATION

Defects affect several stakeholders including tenants, construction companies and government agencies. Consumers or tenants will be the most affected by a faulty building as it is their lives that are under jeopardy if their property is faulty. It is to also be considered that tenants make a large share of emotional investment in the houses that they buy and damage to such property will lead to a loss of both their monetary as well as emotional investment. Construction companies, on the other hand, need to evaluate their projects and rid them off defects (if found) in order to maintain their credibility in the industry. The role of government agencies as stakeholders can’t be overlooked as the final responsibility of the lives of its people lie in the hands of those with administrative power. Therefore, these agencies too are in need of sophisticated techniques to evaluate structures and issue certificates of authorization (given the safety standards are met).

Over the years, there have been several attempts such as the one into assess building defects and evaluate structural conditions. Such analysis gives a great outlook on the factors that need to be considered and also allows for timely repairs, thus avoiding catastrophic occurrences in the future. But, the problem with most methods in use at present is that they involve a lot of manual labour and manual surveying. Therefore, there is a need for an imperative to design a system that would be autonomous in its operation of being able to detect defects in buildings and raise early alarms. It is also important that such a system should be robust, fast and convenient to use.

Tenants will need a Safe Home
Construction Companies will want to uphold Quality
Government agencies will need to Evaluate

Figure 4. Motivation for the Project

1.2. OVERVIEW

Building defects is one of the major components of building problems that significantly needed attention. When a building fails to function as it should, we must immediately seek for detection. for this we have built CNN (Convolution Neural Network) model, where it is provided with dataset. Here data pre-processing and model building applications such as importing necessary libraries, Pre-trained CNN model, Pre-trained CNN model, Training and testing the model. Save the Model are done.

Then we installed flask and requests. We then included three html pages named home, intro, upload. Where home is the home page, intro page is the introduction, and at last upload page is used to display the output. and we have also created app.py which is python scripting file.

1.3. PURPOSE

The main aim or idea behind this project is to detect the defects before it causes the damage. These defects can significantly affect the structural integrity and the aesthetic aspect of buildings.to prevent this We are using CNN pre-trained model VGG16 to analyse the type of building defect on the given parameters.

The objective of the project is to build an application to detect the type of building defect.

1.4. PROBLEM STATEMENT

Construction deficiencies such as poor workmanship and low quality of materials, design deficiencies like not according to the specification and faulty design, limited time and cost, external environment and etc. lead to various types of defects in buildings.

To build a model that will detect external damage to a building, assess the intensity of the defect and inform stakeholders, thus reducing the need for manual inspection.

1.5. SCOPE OF THE PROJECT

The study was limited to the selected building, because of the limited time in covering several other buildings with same defective symptoms (Building defects)

With the problem statement defined, the next step was to identify the approach to be used. The scope of this project deals with the use of Computer Vision to detect damages on external surfaces of buildings.

1.6. THE LIMITATIONS OF DATA

Since there was not too much work being performed in this domain, it was difficult for us to collect large amounts of data for our use. There was a dearth of both the quantity and quality of data for us to work with.

Therefore, we had to rely on external data acquisition for our project. We have collected the data for the project from:

- A pre-defined crack, flake, roof images dataset
- Data Labelling Sites (Dataturks)
- Google Images, Handheld Devices.

2. LITERATURE SURVEY

2.1. INTRODUCTION

The use of computer vision in building damage detection has been mostly concerned around the detection and identification of cracks on walls. This is due to the ability to identify cracks on a surface with relative ease than the other defects on walls. Also, public data sources for cracks are relatively accessible when compared to the extreme lack of public datasets for other wall defects.

It is also to be noted that there has been more work performed in detecting damage in buildings post-calamity, when opposed to the idea of detecting issues with structures before they lead to great damages.

2.2. EXISTING PROBLEM

As the time passes many buildings due to poor maintaince develops defects like cracks, flakes etc., if left undetected it can cause hazardous damages. Traditional methods for this type of work commonly comprise of engaging building surveyors to undertake a condition assessment which involves a lengthy site inspection to produce a systematic recording of the physical condition of the building elements, including cost estimates of immediate and projected long-term costs of renewal, repair and maintenance of the building.

2.3. PROPOSED SOLUTION

In this project detecting building defects such as cracks, flakes and roof defects, we are using CNN pre-trained model VGG16 to analyse the type of building defect on the given parameters. The objective of the project is to build an application to detect the type of building defect.

The model uses an integrated webcam to capture the video frame and the video frame is compared with the pre-trained model and the type of building defect is identified and showcased on the OpenCV window and emergency pull is initiated.

3. THEORITICAL ANALYSIS

3.1. INTRODUCTION

Buildings are structures built with tremendous investment of time, money and emotion. Therefore, every stakeholder involved in the process starting from construction companies to the tenants, wants to make sure that a structure is built well and that it can serve its purpose without any safety hazards. The safety of buildings is dependent on several factors such as foundational stability, structural integrity, construction material used, climatic conditions etc. While a lot of these factors can only be evaluated by a civil engineering expert, there are factors like detecting visible structural damage that might cause a severe investment of time via manual inspection. Therefore, in this project we have made an attempt to detect and identify visually discernible defects in buildings, thus reducing the need for manual inspection. Such a method also introduces objectivity in evaluation of defects.

In this project detecting building defects such as cracks, flakes, and roof defects,we are using CNN pre-trained model VGG16 to analyse the type of building defect on the given parameters. The objective of the project is to build an application to detect the type of building defect. The model uses an integrated webcam to capture the video frame and the video frame is compared with the pre-trained model and the type of building defect is identified and showcased on the OpenCV window and emergency pull is initiated.

3.2. CNN LAYERS

Although, CNN have different architectures, almost all follow the same general design principles of successively applying convolutional layers and pooling layers to an input image. In such arrangement, the ConvNet continuously reduces the spatial dimensions of the input from previous layer while increasing the number of features extracted from the input image.

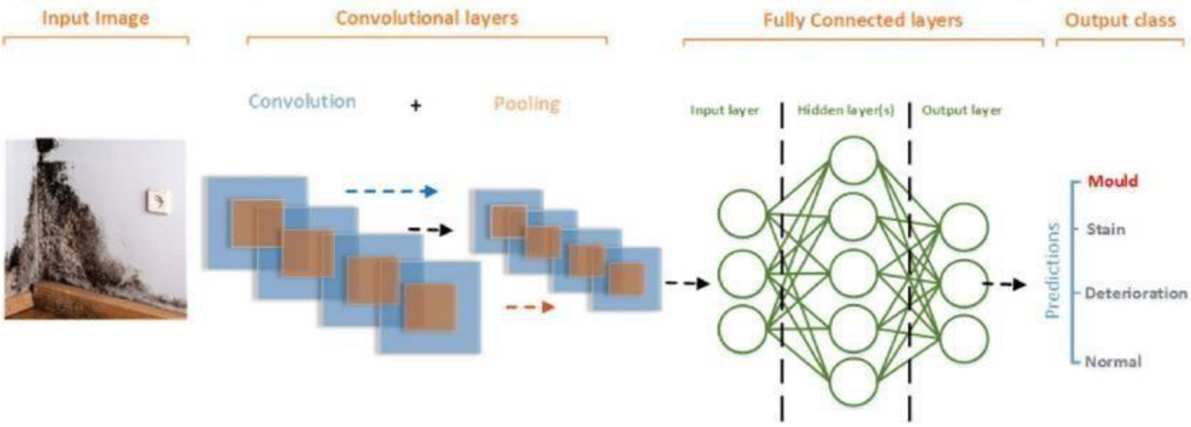


Figure 5. Basic ConvNet Architecture

Input images in neural networks are expressed as multi-dimensional arrays where each colour pixel is represented by a number between 0 and 255. Grey scale images are represented by a 1-D array, while RGB images are represented by a 3-D array, where the colour channels (Red, Green and Blue) represent the depth of the array. In the convolutional layers, different filters with smaller dimensions arrays but same depth as the input image (dimensions can be 1x1xm, 3x3xm, or 5x5xm, where m is the depth of the input image), are used to detect the presence of specific features or patterns present in the original image. The filter slides (convolved) over the whole image starting at the top left corner while computing the dot product of the pixel value in the original image with the values in the filter to generate a feature map. ConvNets use pooling layers to reduce the spatial size of the network by breaking down the output of the convolution layers into smaller regions were the maximum value of every smaller region is taken out and the rest is dropped (max-pooling) or the average of all values is computed (average-pooling). As a result, the number of parameters and computation required in the neural network is reduced significantly. The next series of layer in ConvNets are the fully connected (FC) layers. As the name suggests, it is a fully connected network of neurons (perceptrons). Every neuron in one sub-layer within the FC network, has a connection with all neurons in the successive sub-layer. At the output layer, a classification which is based on the features extracted by the previous layers is performed. Typically, for a multi-classifier neural network, a Softmax activation function is considered, which outputs a probability (a number ranging from 0-1) for each of the classification labels which the network is trying to predict.

3.3. VGG16(Pre-trained CNN model):

A pre-trained model like the VGG-16 is an already pre-trained model on a huge dataset (ImageNet) with a lot of diverse image categories. Considering this fact, the model should have learned a robust hierarchy of features, which are spatial, rotation, and translation invariant with regard to features learned by CNN models. Hence, the model, having learned a good representation of features for over a million images belonging to 1,000 different categories, can act as a good feature extractor for new images suitable for computer vision problems. These new images might never exist in the ImageNet dataset or might be of totally different categories, but the model should still be able to extract relevant features from these images.

Understanding the VGG-16 model:

- Input to the architecture are color images of size 224*224.
- The image is passed through a stack of convolutional layers.
- Every convolution filter has very small receptive field: 3*3, Stride 1.
- Uses row and column padding to maintain spatial resolution after convolution.
- There are 13 Convolution Layers.
- There are 5 max-pool layers.
- Max pooling window size 2*2, stride 2.
- Not every convolution layer is followed by max-pool layer.
- 3 Fully connected layers.
- First two FC layers have 4096 channels each.
- Last FC layer has 1000 channels.
- Last layer is softmax layer with 1000 channels, one for each category of images in imageNet database.
- Hidden layers have ReLU as activation Function.
- Convolutions are responsible for extracting features from the image. The initial layers of a CNN extract low-level features such as edges and as the number of layers increase, more complex features are extracted.

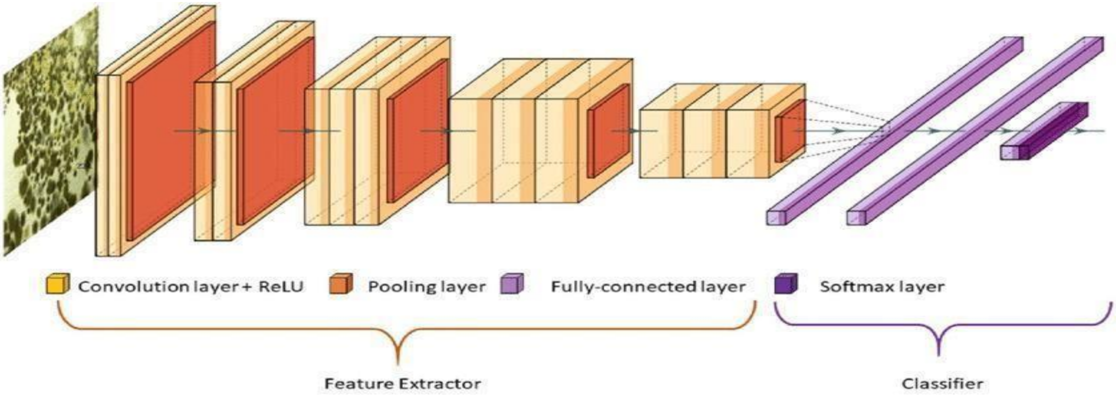


Figure 6. VGG-16 Model

3.4. BLOCK DIAGRAM

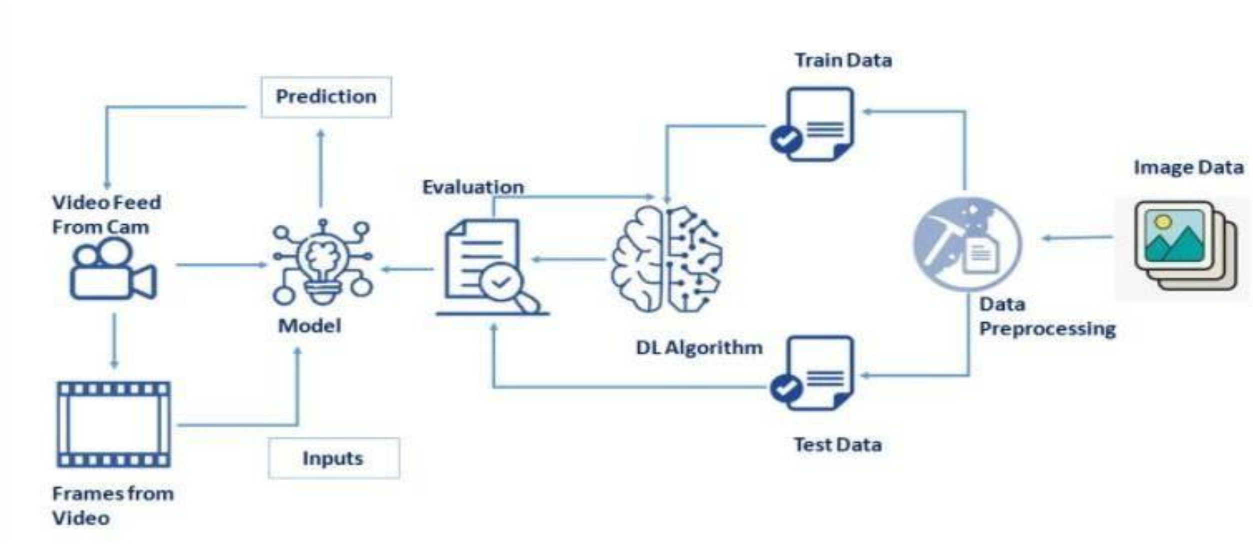


Figure 7. Block Diagram

3.5. SOFTWARE REQUIREMENTS

The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communication with the project team and user throughout the software development process.

3.5.1. SOFTWARE REQUIREMENTS:

- **Anaconda prompt** - Anaconda is a distribution of Python. It is free and open-source and makes package management and deployment simpler.it has tools to easily collect data from sources using ML and AI, it has good community support and it is the industry standard for developing, testing and training on a single machine.

- **Jupyter notebook** -Jupyter is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document.

- **Spyder** - Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging and introspection features.

- **Kaggle** - was used as a platform that provided us with the GPU computational power for our CNN based architecture.
- Chrome - Used to execute localhost.
- Python (TensorFlow, Keras, NumPy)
- HTML
- Flask

3.5.2. HARDWARE REQUIREMENTS:

- System Type - 64-bit Operating System
- RAM-8GB
- Processor-11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz 2.42 GH.

4. EXPERIMENTAL INVESTIGATIONS

While working on the solution we investigated what actually flask is and as our project needed three html web pages, we saw few videos on how to create web pages using html and CNN on YouTube. Before this we first noted down the aim of project and blue print of the project in order to gain knowledge about them through internet, books etc. In html, we learnt about adding buttons, adding images, adding text, adding navbar and etc so as to include them in our pages. We then learnt how to write code for flask python scripting file and testing file with the VGG16by seeing some videos and also by referring some books. As our project is a flask application, in order to gain knowledge about flask we watched videos that are provided in our guided project itself.

5. DESIGN

5.1. THE DEEP LEARNING-BASED APPROACH

In this approach we have used a CNN architecture to classify images.

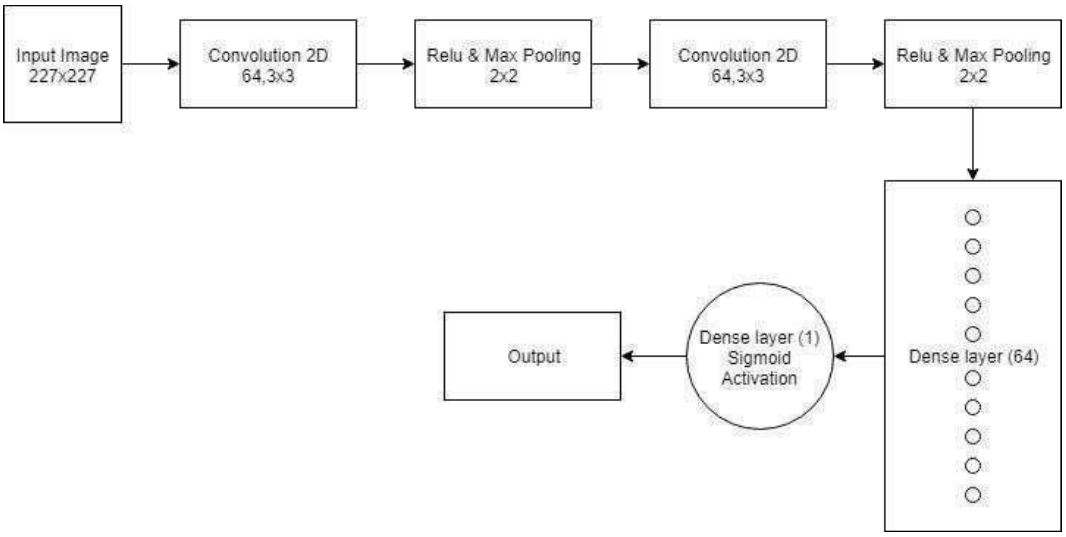


Figure 8. CNN Architecture

5.1.2. TRAINING DATA SPECIFICS

Origin: CNN Dataset (3.1.1)
Number of Images: 10,000
Dimension of Images: 227x 227
Number of Epochs: 10
Validation Split: 10%

5.2.IMAGE PREPROCESSING

Image Pre-processing includes the following main tasks

- Import ImageDataGenerator Library.
- Configure ImageDataGenerator Class.
- Applying ImageDataGenerator functionality to the trainset and test set

Note: The ImageDataGenerator accepts the original data, randomly transforms it, and returns only the new, transformed data.

5.3.FLOW CHART

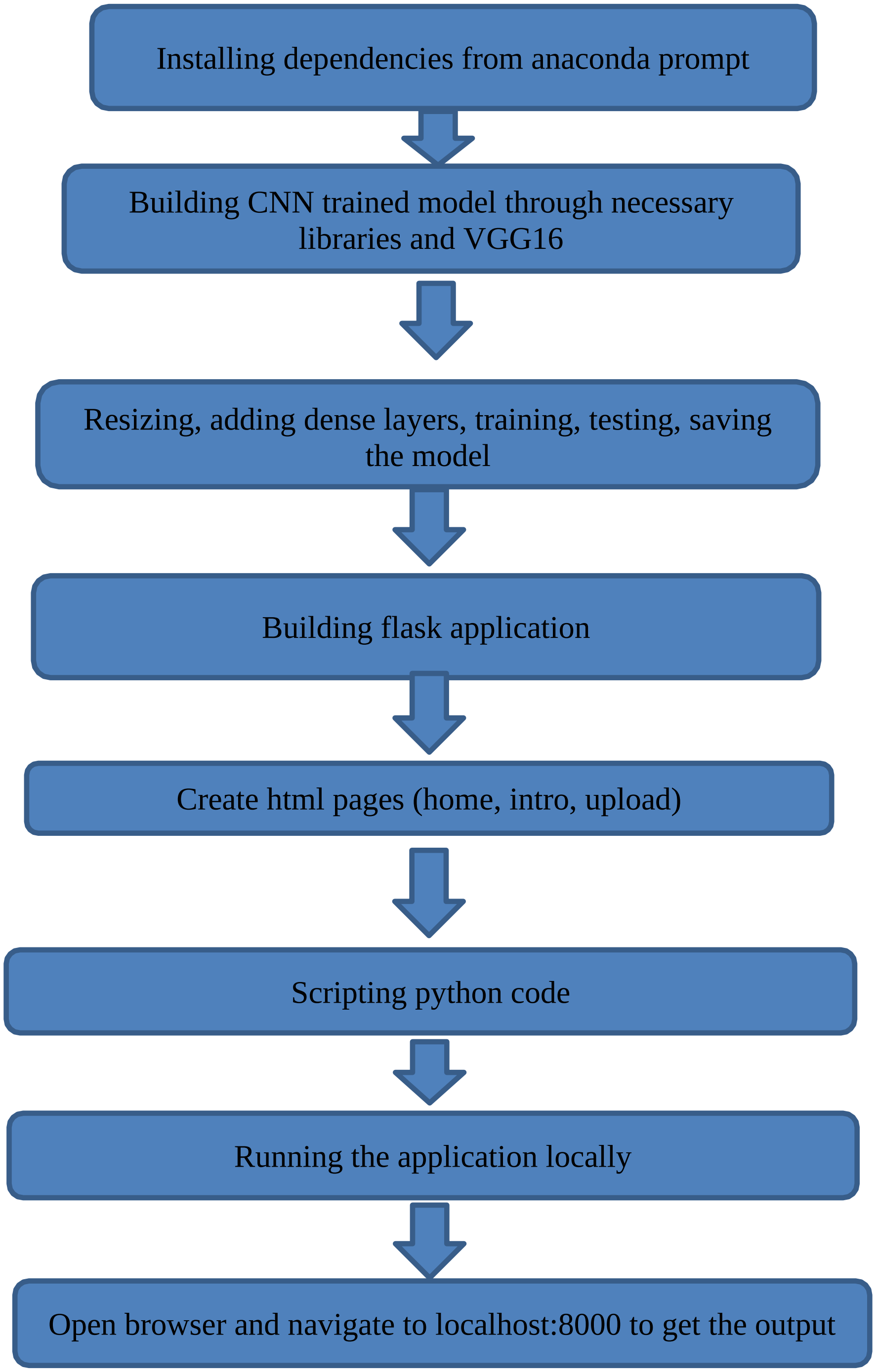

Installing dependencies from anaconda prompt
Building CNN trained model through necessary libraries and VGG16
Resizing, adding dense layers, training, testing, saving the model
Building flask application
Create html pages (home, intro, upload)
Scripting python code
Running the application locally
Open browser and navigate to localhost:8000 to get the output

Figure 9. Flow Chart

5.3.1 PROJECT FLOW

- The user interacts with the UI (User Interface) to open the integrated webcam.
- The video frames are captured and analyzed by the model which is integrated with the flask application.
- Once the model analyses the video frames, the prediction is showcased on the UI and OpenCV window

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

Data Collection:

- Collect the dataset or create the dataset

Data Preprocessing:

- Import the ImageDataGenerator library
- Configure ImageDataGenerator class
- ApplyImageDataGenerator functionality to Trainset and Testset

Model Building:

- Import the model building Libraries
- Resizing the images
- Pre-trained CNN model as a Feature Extractor
- Adding Output Layer
- Configure the Learning Process
- Training and testing the model
- Save the Model
- Test The model

Application Building:

- Create an HTML file
- Build Python Code

6. CONCLUSION

In UG Project Phase-1, we have worked on problem statement, literature survey and also done the experimental analyses which are required for the project to move forward. We also discussed about the flowcharts, convolutional neural network,VGG16 model which are used in the project. Based on the experimental analysis we have designed the model for the project. Entire designing part is involved in UG Project Phase-1.

The general accepted idea is that the dataset, as well as the chosen model lead to great performances, thus both need to receive attention from the developer. The training hardware doesn‘t need to be expensive, if the application does not mandate it. Smaller, more specific applications can yield great results and thus, improve the workings of a small lab or business just by using general purpose laptops and generic detection models which will be tweaked for the defects we look into. While general applications that have a target as to detect lots of defects, need very large and balanced datasets, a hardware setup with lots of computational power and a specific detection model that is not just tweaked for defect detection, but built from the ground up for the specific action that we want it to perform.

There is no rule of thumb when choosing which object detection model shall generalize the best on the particular dataset of interest. First of all, the developer needs to be in constant contact with the system technician or quality inspector, in order to assure that the dataset is of great quality and is reliable.

7. FUTURE SCOPE

UG Project Phase-2 is the extension of UG Project Phase-1. UG Project Phase-2 involves all the coding and implementation of the design which we have retrieved from UG Project Phase-1. All the implementation is done and conclusions will be retrieved in the phase. We will also work on the applications, advantages, and disadvantages of the project in this phase. Future scope of the project will be also discussed in the UG Project Phase-2.

DETECTING BUILDING DEFECTS USING VGG16

A UG PROJECT PHASE -2 REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

VADICHERLA NARESH	**18UK1A05H7**
LAKKARSU SHIVA	**18UK1A05K6**
SAI REVANTH METTELA	**18UK1A05M0**
BOMMA KAVYA	**18UK1A05J4**

Under the guidance of

Mr. V. RANADHEER REDDY

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

WARANGAL

CERTIFICATE OF COMPLETION UG PROJECT PHASE – 2

This is to certify that the UG Project Phase – 2 project report entitled **"DETECTING BUILDING DEFECTS USING VGG16"** is being submitted by **V. NARESH (18UK1A05H7), L. SHIVA (18UK1A05K6), M. SAI REVANTH (18UK1A05M0), B. KAVYA (18UK1A05J4),** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2021-2022.

Project Guide **HOD**

Mr. V. RANADHEER REDDY **Dr. R. NAVEEN KUMAR**

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. Prasad Rao**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG project phase - 2 in the institute.

We extend our heartfelt thanks **to Dr. R. Naveen Kumar**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG project phase -2.

We express heartfelt thanks to **Miss. Sri Tulasi**, AI Developer, Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the major project and for their support in completing the UG project phase -2.

We express heartfelt thanks to the Major Project Guide**, Mr. V. Ranadheer Reddy**, Assistant Professor, Department of CSE for his constant support and giving necessary guidance for completion of the UG project phase - 2.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

TABLE OF CONTENTS:

LIST OF CHAPTERS Page No

LIST OF FIGURES PAGE NO:

1. INTRODUCTION

Defect detection using computer vision models started to pick up popularity in the 21st century, as the object detection models became more and more popular. The exponential growth in popularity of deep learning methods for defect detection and other computer vision related applications in recent years is fueled by lots of researchers plunging into this sector, as well as by hardware and data breakthroughs. Lots of areas of defect detection solutions were reviewed in this paper and as demonstrated deep learning methods achieve state-of-the-art performance in defect detection, while also having great generalization properties.

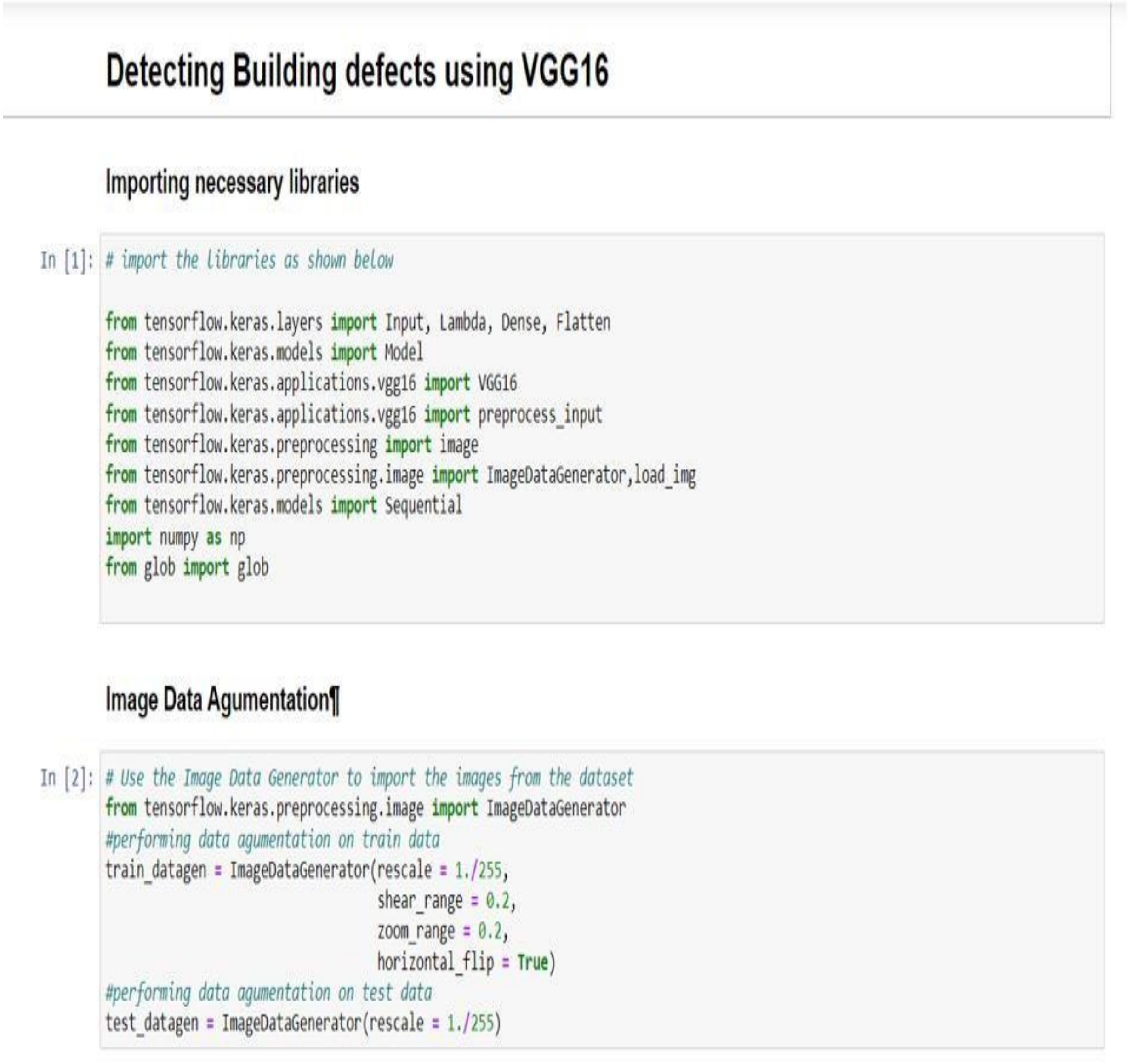
A Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image by using a CNN

Detection of defects including cracks and flakes on the wall surfaces in high-rise buildings is a crucial task of buildings maintenance. If left undetected and untreated, these defects can significantly affect the structural integrity and the aesthetic aspect of buildings, timely and cost-effective methods of building condition surveys are of practicing need for the building owners and maintenance agencies to replace the time and labor-consuming approach of the manual survey. In this project detecting building defects such as cracks, flakes, and roof defects, we are using CNN pre-trained model VGG16 to analyse the type of building defect on the given parameters. The objective of the project is to build an application to detect the type of building defect.

UG Project Phase-2 involves all the coding and implementation of the design which we have retrieved from UG Project Phase-1. All the implementation is done and conclusions are retrieved in this phase. We will also work on the applications, advantages, and disadvantages of the project in this phase. Future scope of the project will be also discussed in the UG Project Phase-2.

2. CODE SNIPPETS

2.1. MODEL CODE



Detecting Building defects using VGG16

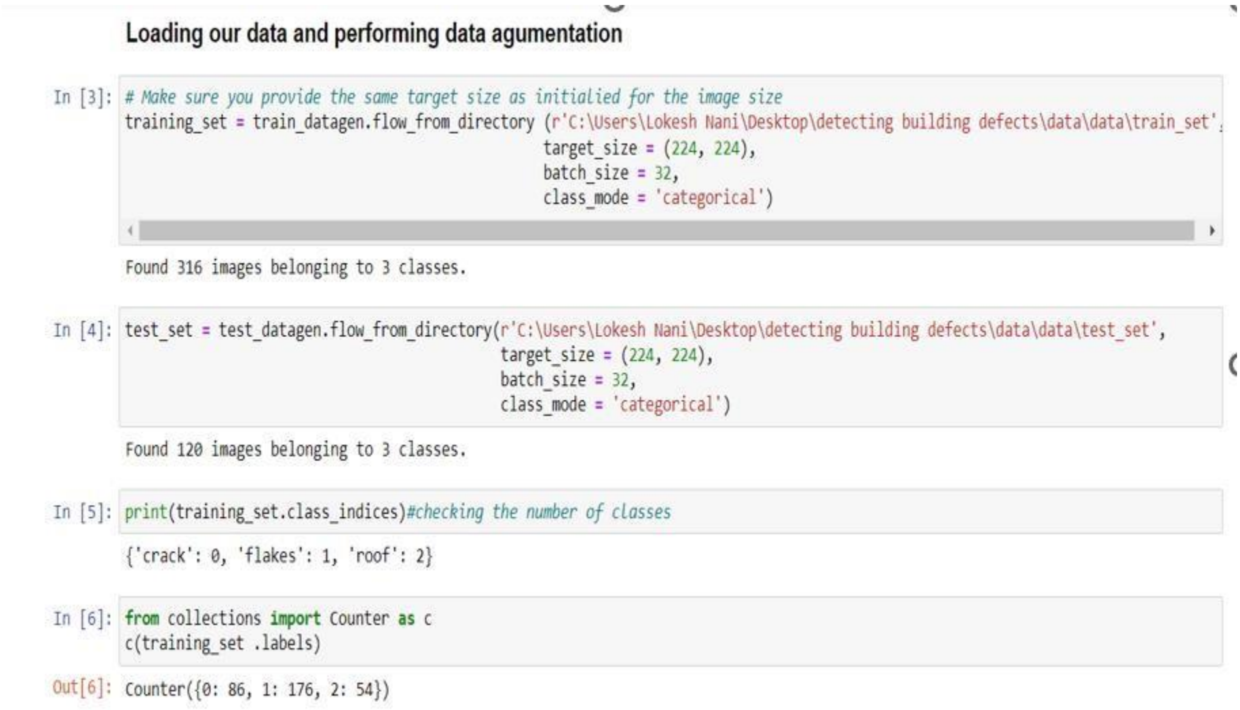
Importing necessary libraries

```
In [1]: # import the libraries as shown below

        from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
        from tensorflow.keras.models import Model
        from tensorflow.keras.applications.vgg16 import VGG16
        from tensorflow.keras.applications.vgg16 import preprocess_input
        from tensorflow.keras.preprocessing import image
        from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
        from tensorflow.keras.models import Sequential
        import numpy as np
        from glob import glob
```

Image Data Agumentation¶

```
In [2]: # Use the Image Data Generator to import the images from the dataset
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        #performing data agumentation on train data
        train_datagen = ImageDataGenerator(rescale = 1./255,
                                           shear_range = 0.2,
                                           zoom_range = 0.2,
                                           horizontal_flip = True)
        #performing data agumentation on test data
        test_datagen = ImageDataGenerator(rescale = 1./255)
```

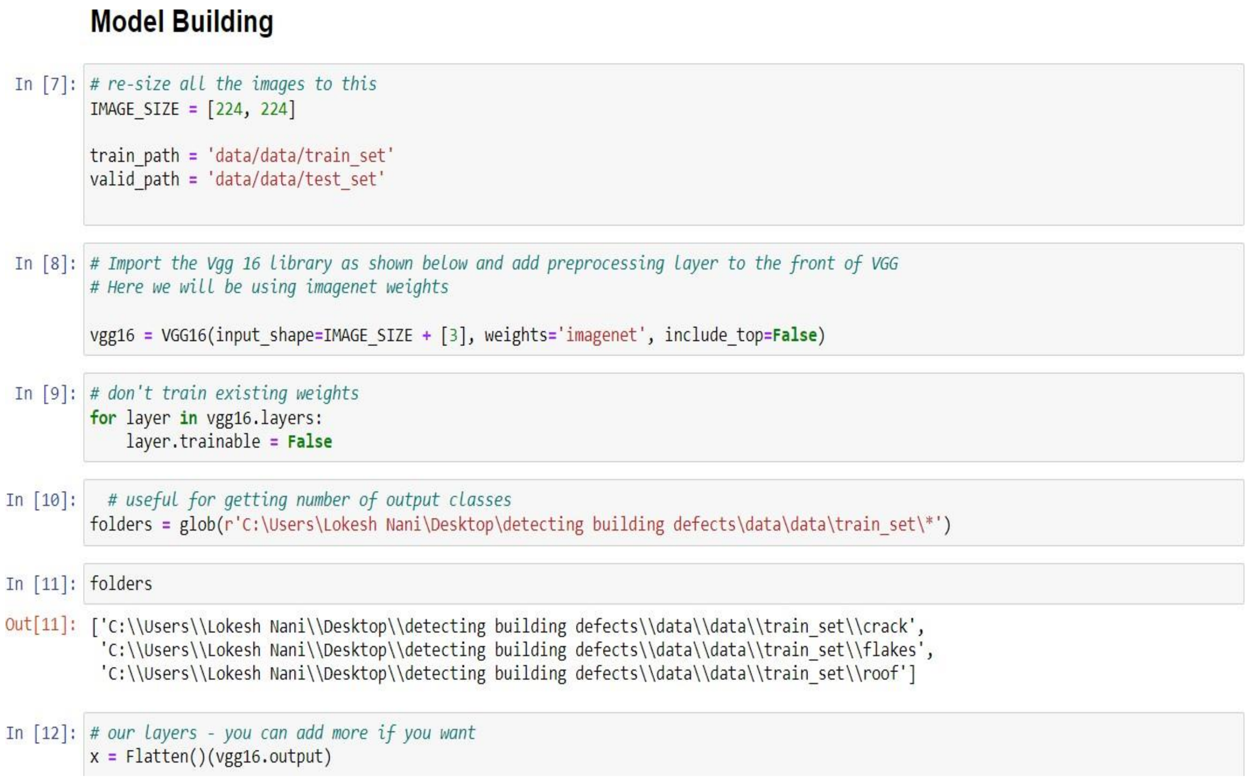
Figure 1. ipynb code describing importing necessary libraries and Image Data Agumentation



Loading our data and performing data agumentation

```
In [3]: # Make sure you provide the same target size as initialied for the image size
        training_set = train_datagen.flow_from_directory (r'C:\Users\Lokesh Nani\Desktop\detecting building defects\data\data\train_set',
                                                         target_size = (224, 224),
                                                         batch_size = 32,
                                                         class_mode = 'categorical')

        Found 316 images belonging to 3 classes.

In [4]: test_set = test_datagen.flow_from_directory(r'C:\Users\Lokesh Nani\Desktop\detecting building defects\data\data\test_set',
                                                    target_size = (224, 224),
                                                    batch_size = 32,
                                                    class_mode = 'categorical')

        Found 120 images belonging to 3 classes.

In [5]: print(training_set.class_indices)#checking the number of classes

        {'crack': 0, 'flakes': 1, 'roof': 2}

In [6]: from collections import Counter as c
        c(training_set .labels)

Out[6]: Counter({0: 86, 1: 176, 2: 54})
```

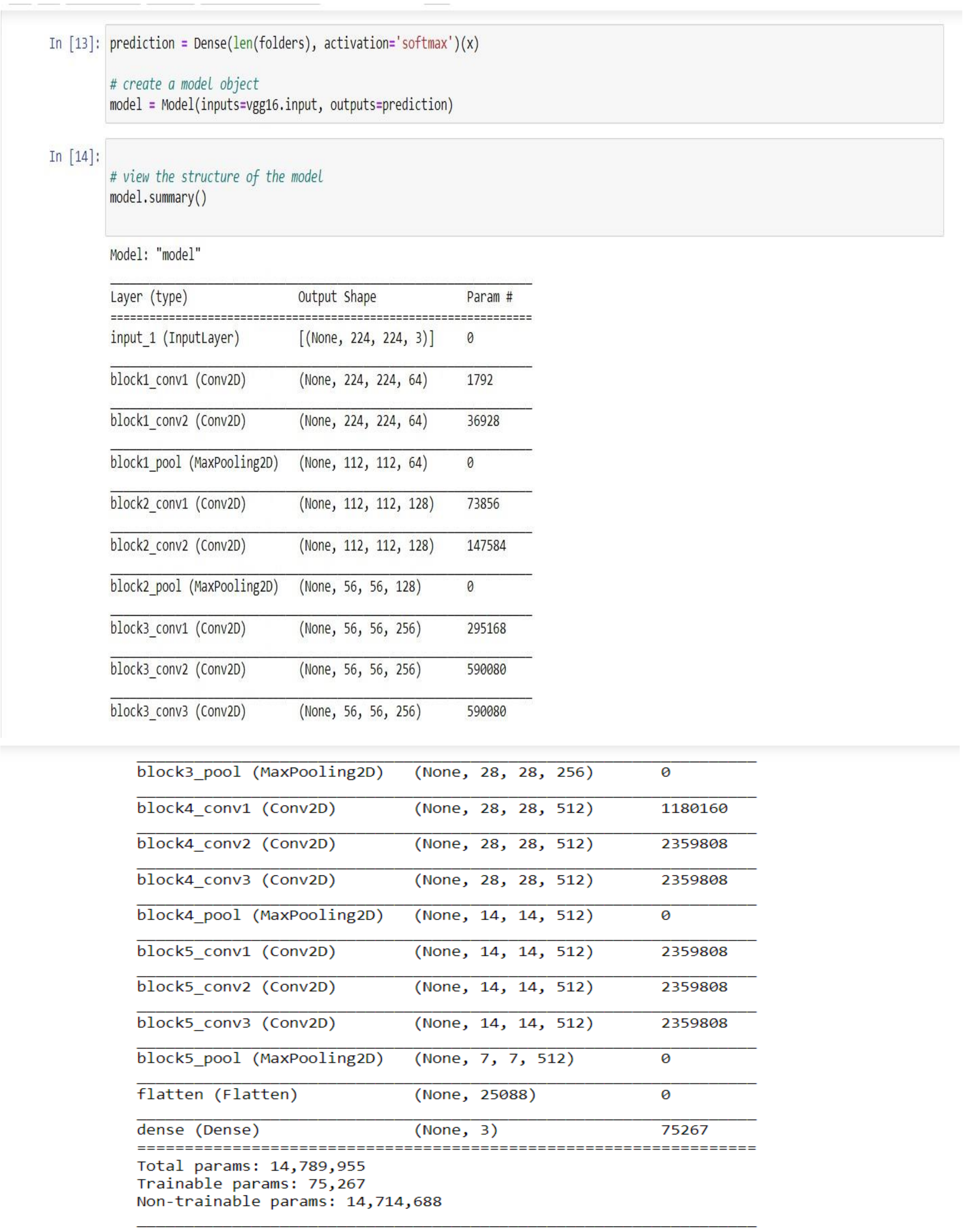
Figure 2. loading our data and performing data agumentation



Model Building

```
In [7]: # re-size all the images to this
        IMAGE_SIZE = [224, 224]

        train_path = 'data/data/train_set'
        valid_path = 'data/data/test_set'

In [8]: # Import the Vgg 16 library as shown below and add preprocessing layer to the front of VGG
        # Here we will be using imagenet weights

        vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

In [9]: # don't train existing weights
        for layer in vgg16.layers:
            layer.trainable = False

In [10]:  # useful for getting number of output classes
        folders = glob(r'C:\Users\Lokesh Nani\Desktop\detecting building defects\data\data\train_set\*')

In [11]: folders

Out[11]: ['C:\\Users\\Lokesh Nani\\Desktop\\detecting building defects\\data\\data\\train_set\\crack',
          'C:\\Users\\Lokesh Nani\\Desktop\\detecting building defects\\data\\data\\train_set\\flakes',
          'C:\\Users\\Lokesh Nani\\Desktop\\detecting building defects\\data\\data\\train_set\\roof']

In [12]: # our layers - you can add more if you want
        x = Flatten()(vgg16.output)
```

Figure 3. Pre-Trained CNN Model as a Feature Extractor

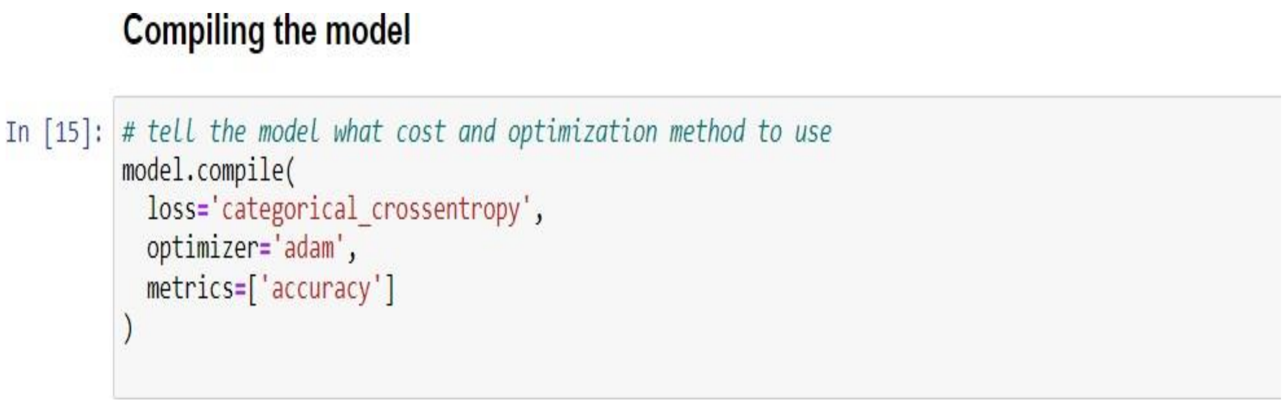


```
In [13]: prediction = Dense(len(folders), activation='softmax')(x)

         # create a model object
         model = Model(inputs=vgg16.input, outputs=prediction)
```

```
In [14]:
         # view the structure of the model
         model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 3)                 75267
=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
_________________________________________________________________
```

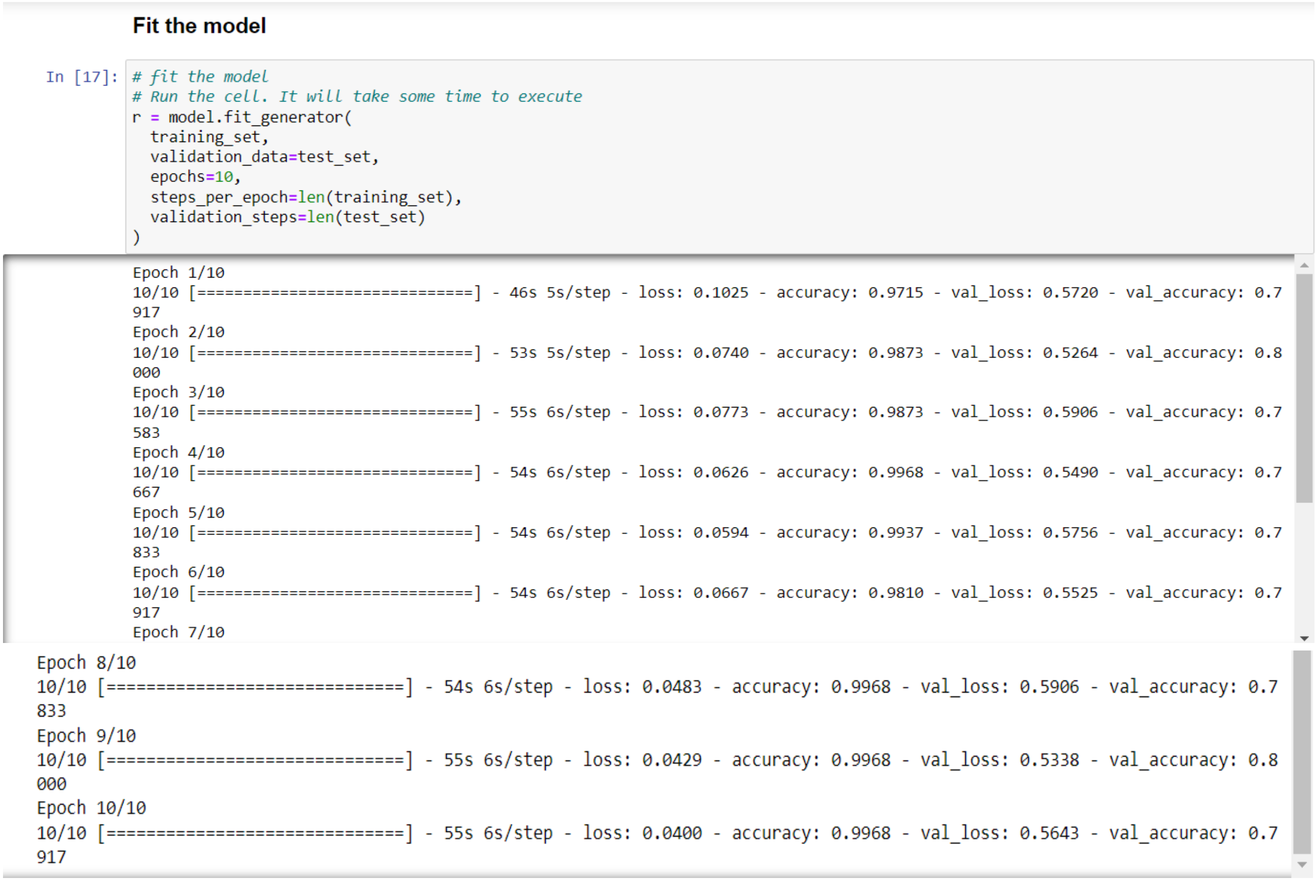
Figure 4. ipynb code describing Adding Dense Layer and Get the full information about the model and its layers



Compiling the model

```
In [15]: # tell the model what cost and optimization method to use
         model.compile(
           loss='categorical_crossentropy',
           optimizer='adam',
           metrics=['accuracy']
         )
```

Figure 5. Compiling The Model



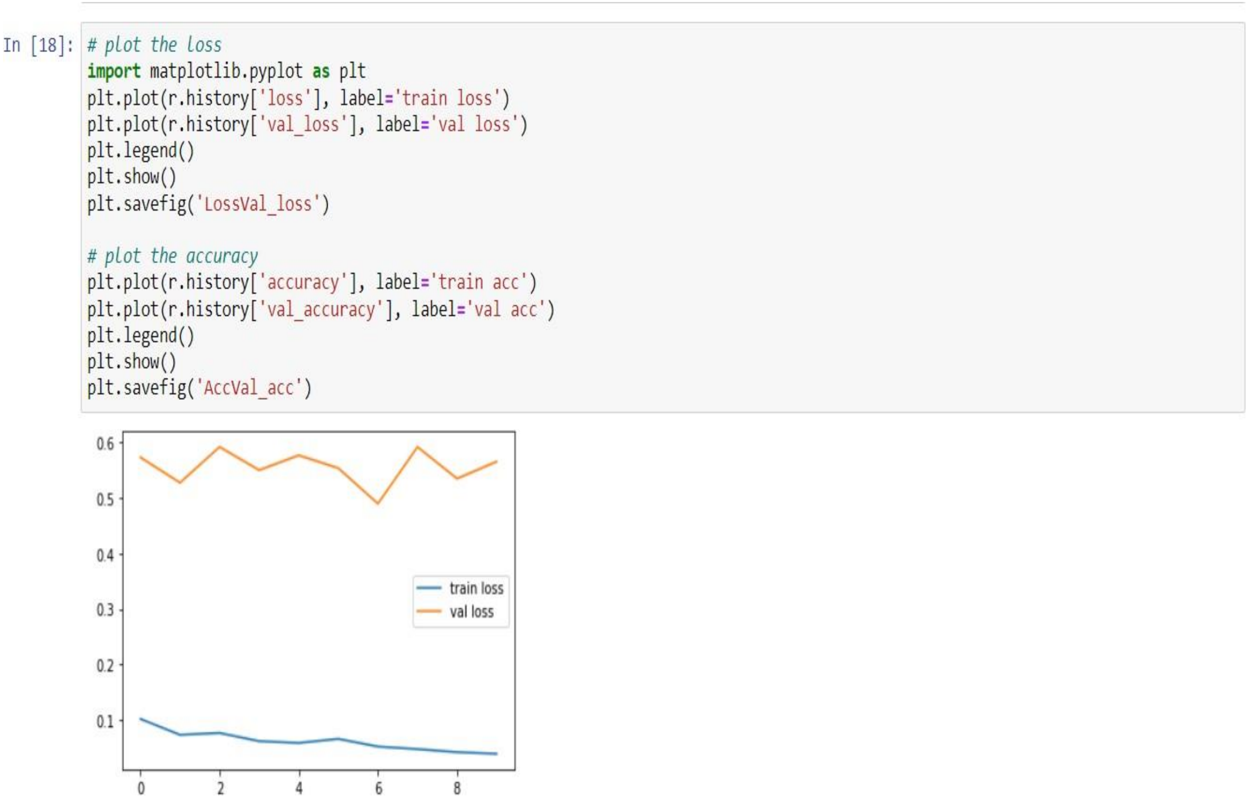
Fit the model

```
In [17]: # fit the model
         # Run the cell. It will take some time to execute
         r = model.fit_generator(
           training_set,
           validation_data=test_set,
           epochs=10,
           steps_per_epoch=len(training_set),
           validation_steps=len(test_set)
         )
```

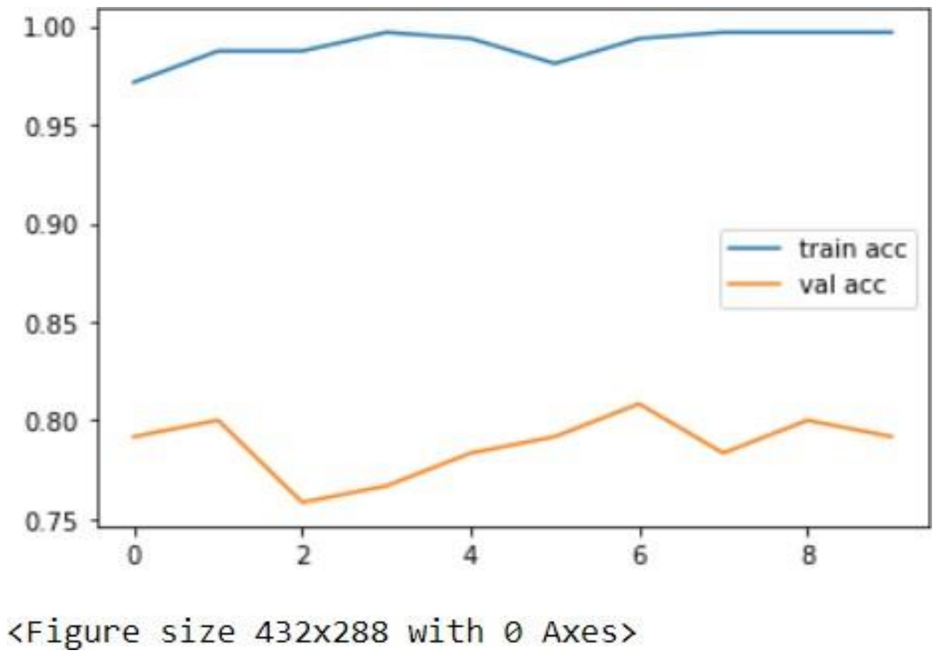
```
Epoch 1/10
10/10 [==============================] - 46s 5s/step - loss: 0.1025 - accuracy: 0.9715 - val_loss: 0.5720 - val_accuracy: 0.7
917
Epoch 2/10
10/10 [==============================] - 53s 5s/step - loss: 0.0740 - accuracy: 0.9873 - val_loss: 0.5264 - val_accuracy: 0.8
000
Epoch 3/10
10/10 [==============================] - 55s 6s/step - loss: 0.0773 - accuracy: 0.9873 - val_loss: 0.5906 - val_accuracy: 0.7
583
Epoch 4/10
10/10 [==============================] - 54s 6s/step - loss: 0.0626 - accuracy: 0.9968 - val_loss: 0.5490 - val_accuracy: 0.7
667
Epoch 5/10
10/10 [==============================] - 54s 6s/step - loss: 0.0594 - accuracy: 0.9937 - val_loss: 0.5756 - val_accuracy: 0.7
833
Epoch 6/10
10/10 [==============================] - 54s 6s/step - loss: 0.0667 - accuracy: 0.9810 - val_loss: 0.5525 - val_accuracy: 0.7
917
Epoch 7/10
```

```
Epoch 8/10
10/10 [==============================] - 54s 6s/step - loss: 0.0483 - accuracy: 0.9968 - val_loss: 0.5906 - val_accuracy: 0.7
833
Epoch 9/10
10/10 [==============================] - 55s 6s/step - loss: 0.0429 - accuracy: 0.9968 - val_loss: 0.5338 - val_accuracy: 0.8
000
Epoch 10/10
10/10 [==============================] - 55s 6s/step - loss: 0.0400 - accuracy: 0.9968 - val_loss: 0.5643 - val_accuracy: 0.7
917
```

Figure 6. Fit the Model and Number of epochs run with its loss and accuracy

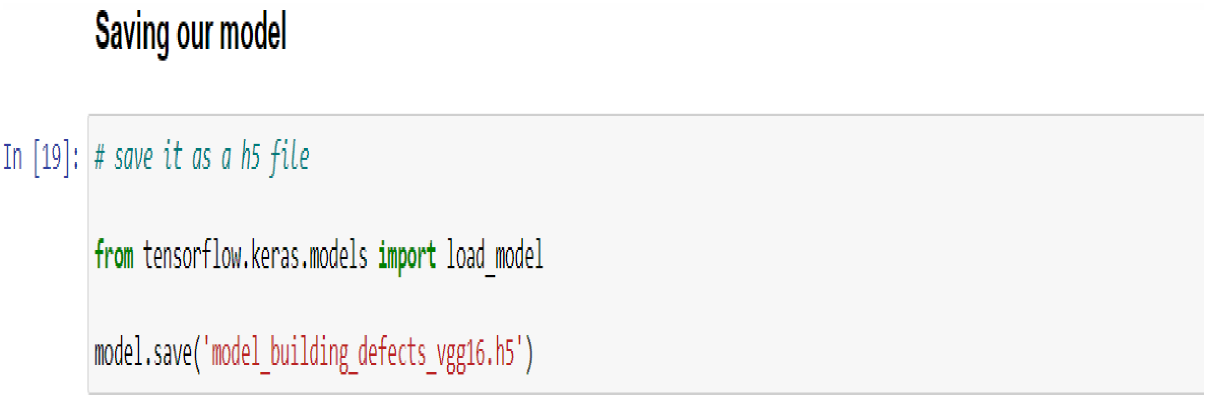


```
In [18]: # plot the loss
import matplotlib.pyplot as plt
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```



```
<Figure size 432x288 with 0 Axes>
```

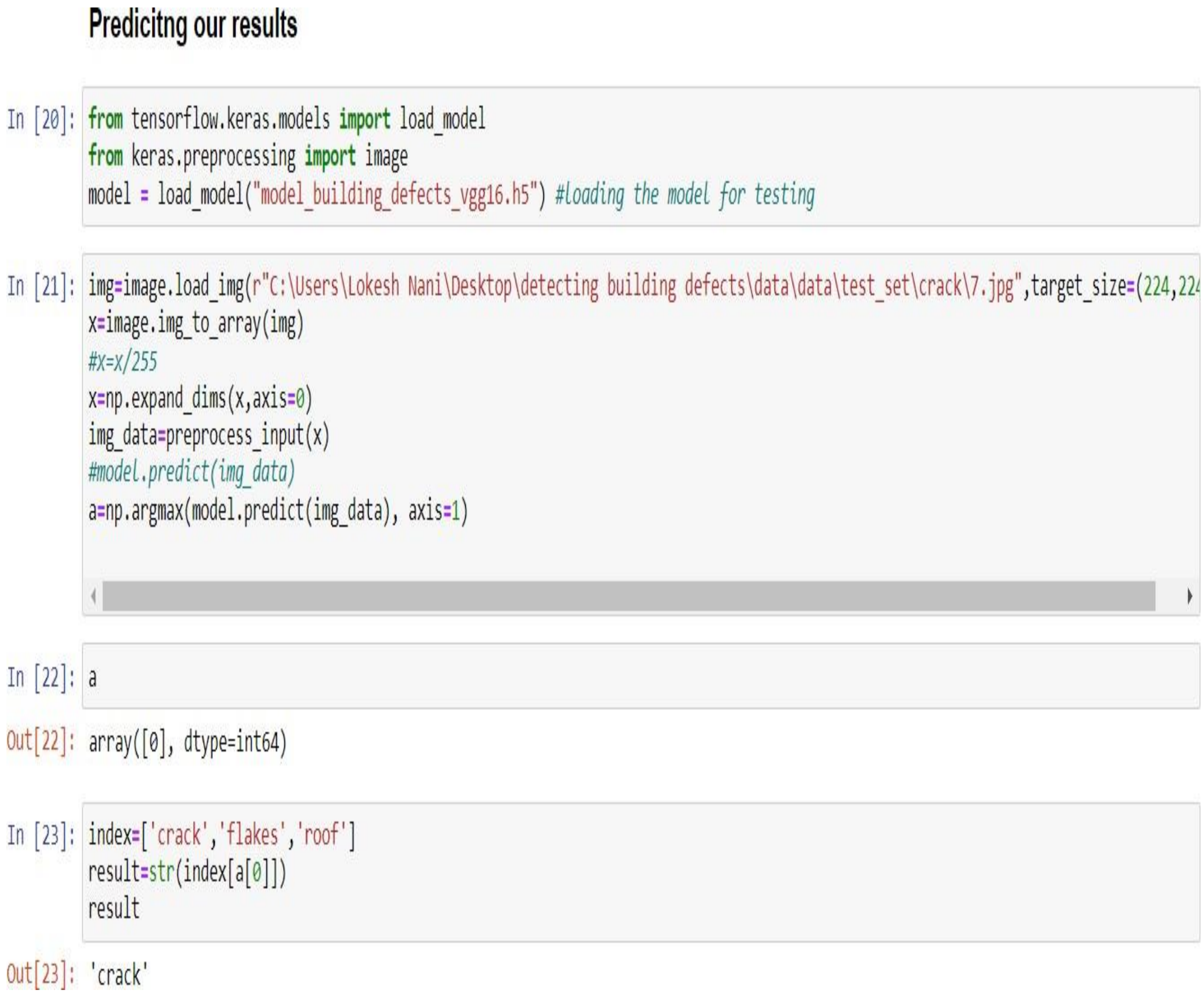
Figure 7. Model Accuracy and Loss



Saving our model

```
In [19]: # save it as a h5 file

         from tensorflow.keras.models import load_model

         model.save('model_building_defects_vgg16.h5')
```

Figure 8. Saving Our Model



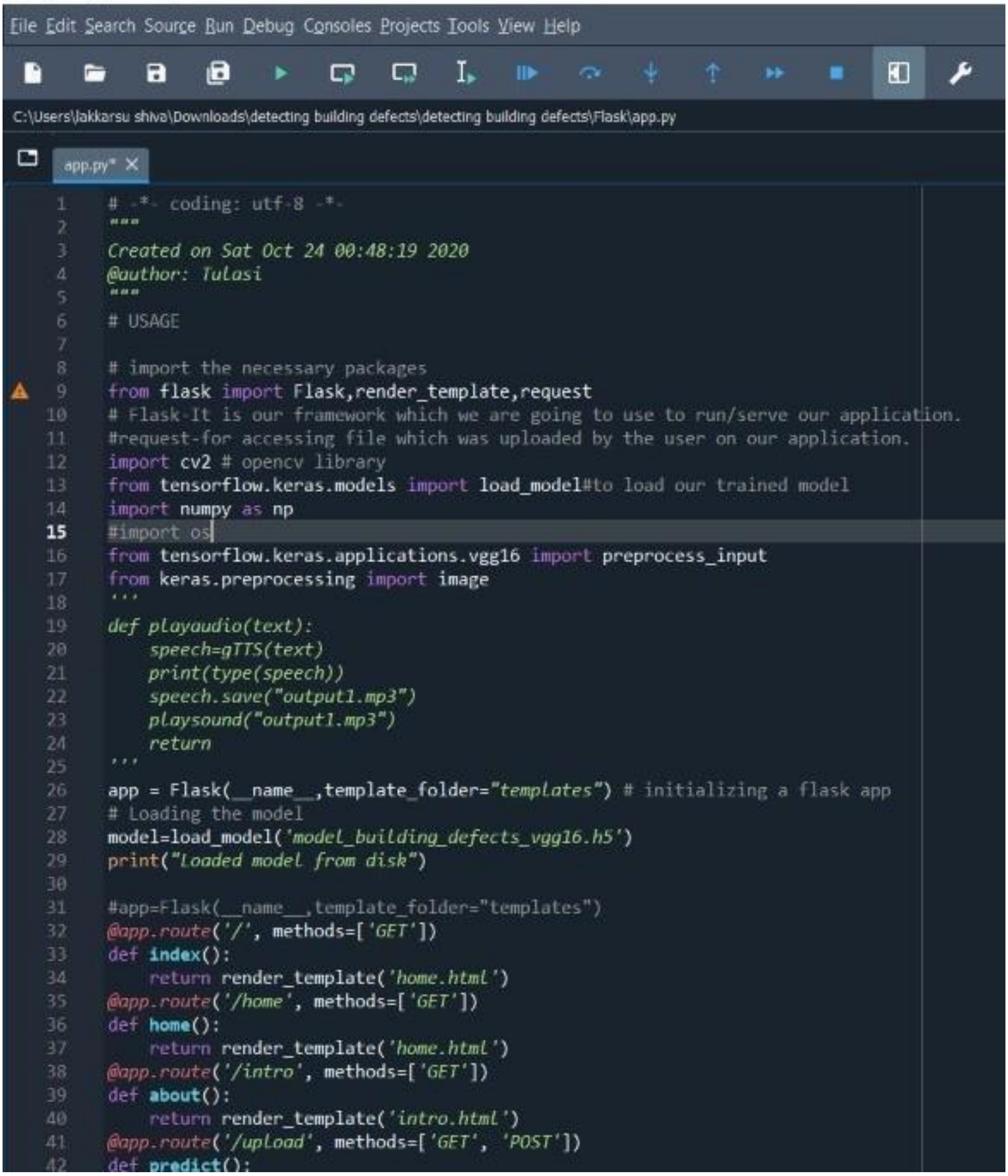
Predicitng our results

```
In [20]: from tensorflow.keras.models import load_model
         from keras.preprocessing import image
         model = load_model("model_building_defects_vgg16.h5") #loading the model for testing

In [21]: img=image.load_img(r"C:\Users\Lokesh Nani\Desktop\detecting building defects\data\data\test_set\crack\7.jpg",target_size=(224,224
         x=image.img_to_array(img)
         #x=x/255
         x=np.expand_dims(x,axis=0)
         img_data=preprocess_input(x)
         #model.predict(img_data)
         a=np.argmax(model.predict(img_data), axis=1)

In [22]: a

Out[22]: array([0], dtype=int64)

In [23]: index=['crack','flakes','roof']
         result=str(index[a[0]])
         result

Out[23]: 'crack'
```

Figure 9. Test the Model then Predicting Our Results

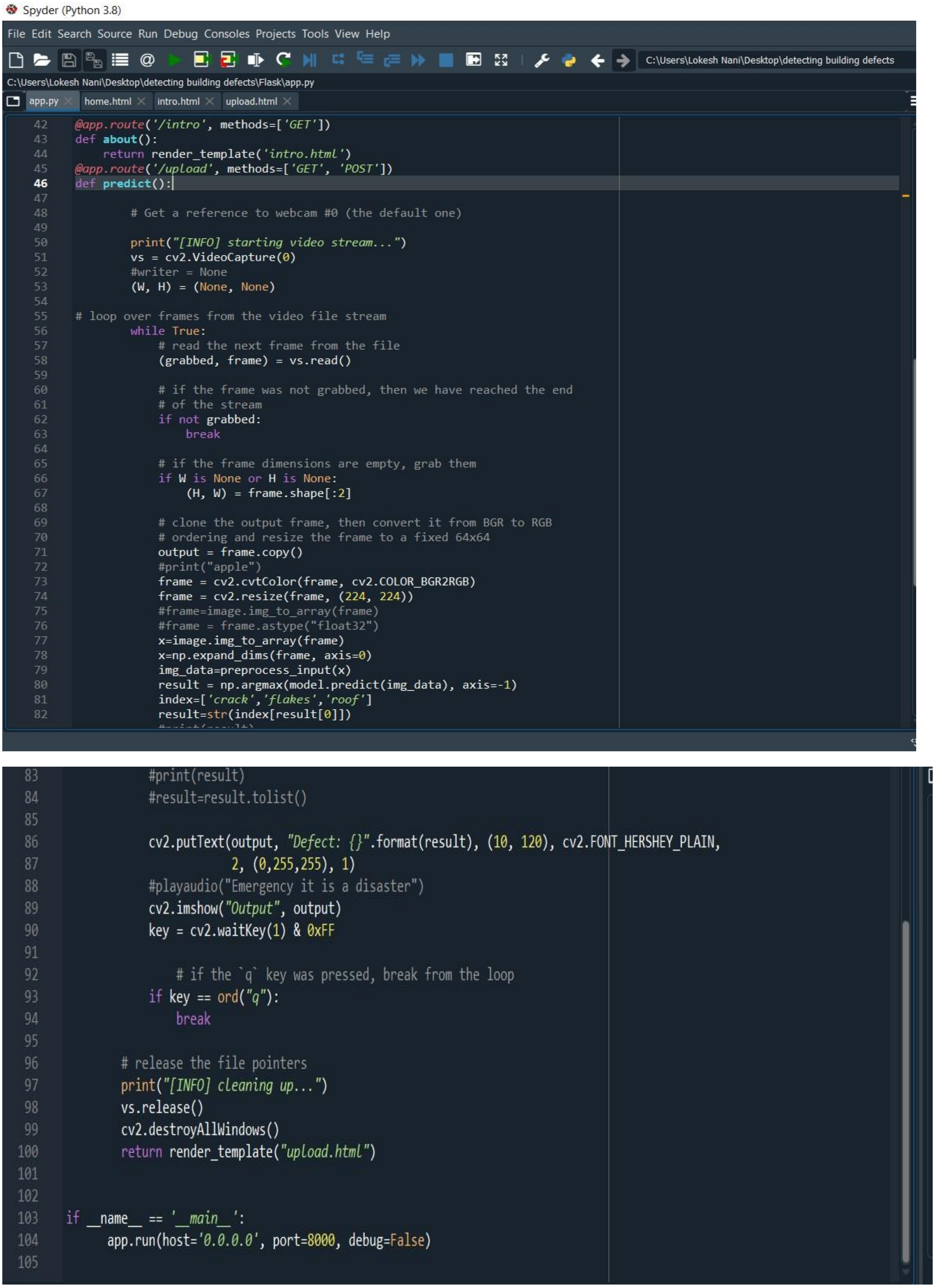
2.2. PYTHON CODE

- **APP.PY CODE:**



File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\lakkarsu shiva\Downloads\detecting building defects\detecting building defects\Flask\app.py
app.py*

```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 24 00:48:19 2020
@author: Tulasi
"""
# USAGE

# import the necessary packages
from flask import Flask,render_template,request
# Flask-It is our framework which we are going to use to run/serve our application.
#request-for accessing file which was uploaded by the user on our application.
import cv2 # opencv library
from tensorflow.keras.models import load_model#to load our trained model
import numpy as np
#import os
from tensorflow.keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
'''
def playaudio(text):
    speech=gTTS(text)
    print(type(speech))
    speech.save("output1.mp3")
    playsound("output1.mp3")
    return
'''
app = Flask(__name__,template_folder="templates") # initializing a flask app
# Loading the model
model=load_model('model_building_defects_vgg16.h5')
print("Loaded model from disk")

#app=Flask(__name__,template_folder="templates")
@app.route('/', methods=['GET'])
def index():
    return render_template('home.html')
@app.route('/home', methods=['GET'])
def home():
    return render_template('home.html')
@app.route('/intro', methods=['GET'])
def about():
    return render_template('intro.html')
@app.route('/upload', methods=['GET', 'POST'])
def predict():
```

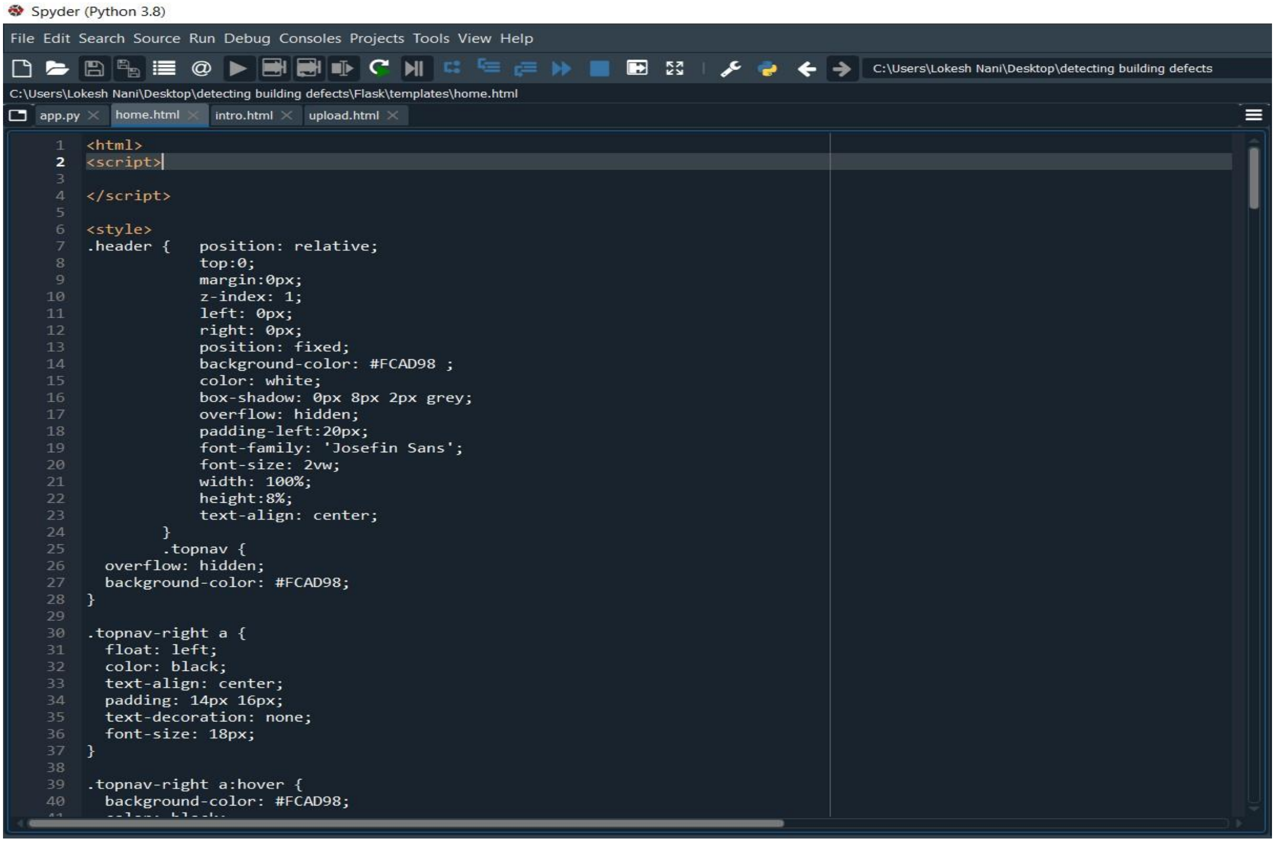

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\app.py
app.py
home.html
intro.html
upload.html

```python
@app.route('/intro', methods=['GET'])
def about():
    return render_template('intro.html')
@app.route('/upload', methods=['GET', 'POST'])
def predict():

        # Get a reference to webcam #0 (the default one)

        print("[INFO] starting video stream...")
        vs = cv2.VideoCapture(0)
        #writer = None
        (W, H) = (None, None)

# loop over frames from the video file stream
        while True:
            # read the next frame from the file
            (grabbed, frame) = vs.read()

            # if the frame was not grabbed, then we have reached the end
            # of the stream
            if not grabbed:
                break

            # if the frame dimensions are empty, grab them
            if W is None or H is None:
                (H, W) = frame.shape[:2]

            # clone the output frame, then convert it from BGR to RGB
            # ordering and resize the frame to a fixed 64x64
            output = frame.copy()
            #print("apple")
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame = cv2.resize(frame, (224, 224))
            #frame=image.img_to_array(frame)
            #frame = frame.astype("float32")
            x=image.img_to_array(frame)
            x=np.expand_dims(frame, axis=0)
            img_data=preprocess_input(x)
            result = np.argmax(model.predict(img_data), axis=-1)
            index=['crack','flakes','roof']
            result=str(index[result[0]])
            #print(result)
            #result=result.tolist()

            cv2.putText(output, "Defect: {}".format(result), (10, 120), cv2.FONT_HERSHEY_PLAIN,
                        2, (0,255,255), 1)
            #playaudio("Emergency it is a disaster")
            cv2.imshow("Output", output)
            key = cv2.waitKey(1) & 0xFF

                # if the `q` key was pressed, break from the loop
            if key == ord("q"):
                break

        # release the file pointers
        print("[INFO] cleaning up...")
        vs.release()
        cv2.destroyAllWindows()
        return render_template("upload.html")


if __name__ == '__main__':
      app.run(host='0.0.0.0', port=8000, debug=False)
```

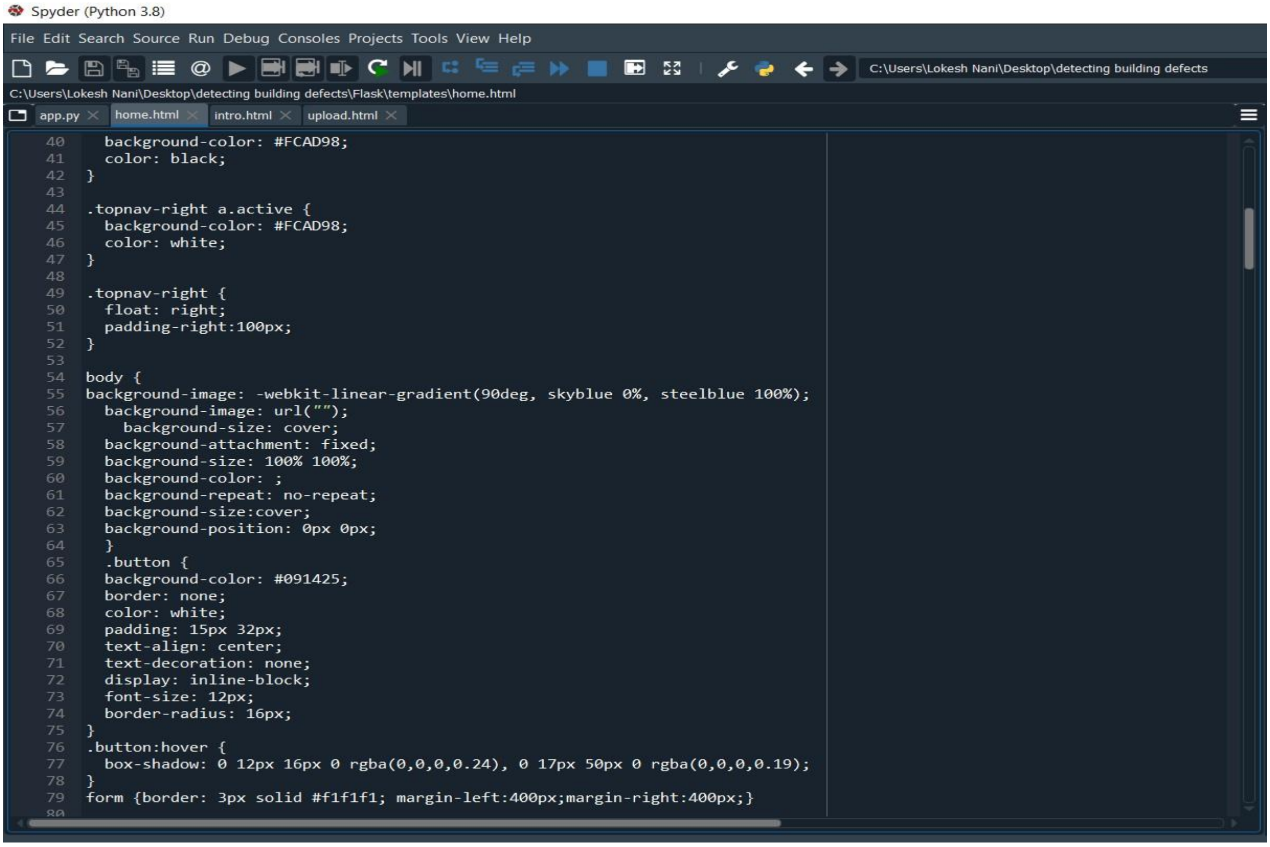
Figure 10. Python code used for rendering all the HTML page

2.3.HTML CODE

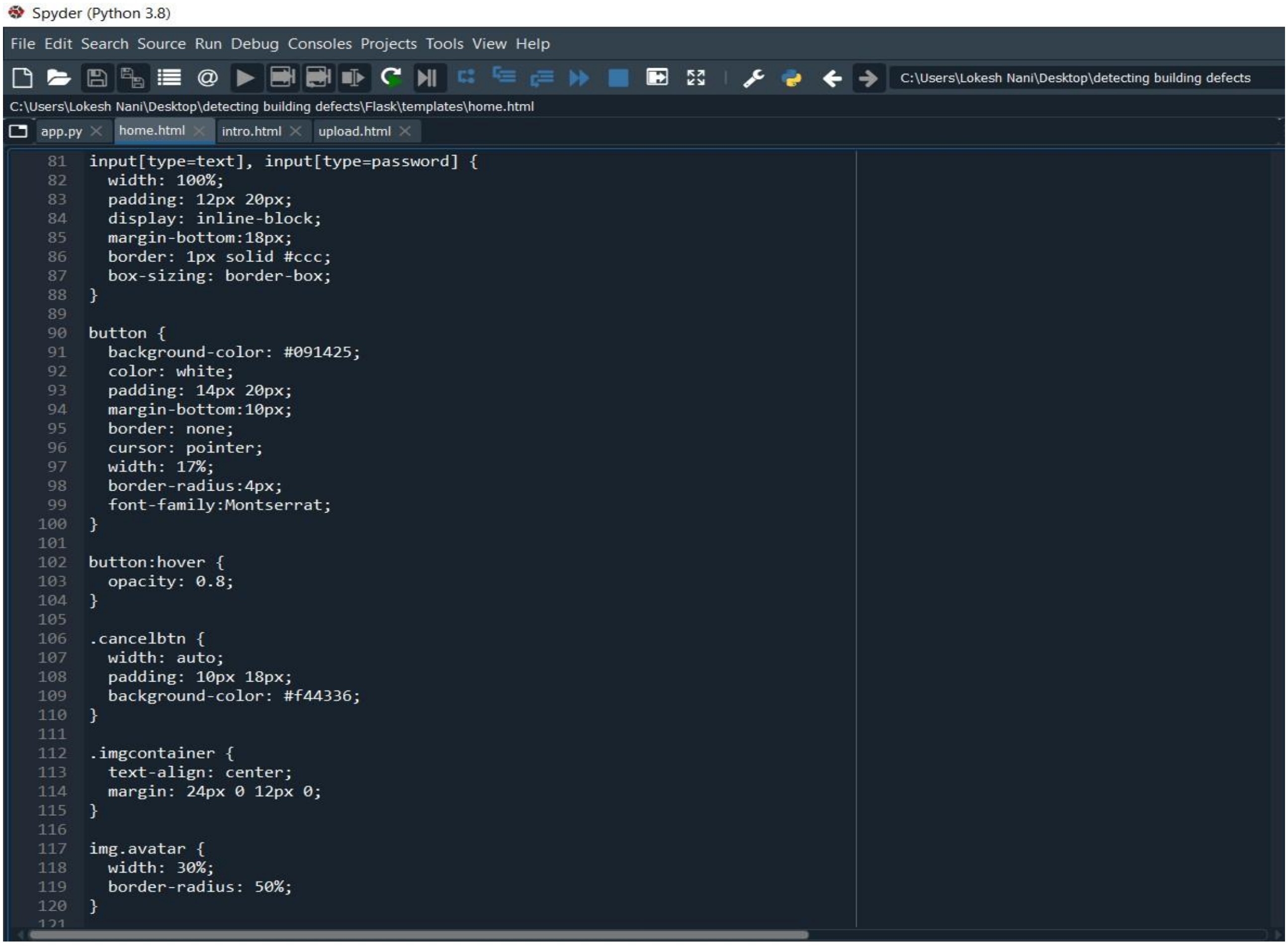
- **HOME.HTML:**


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\home.html
app.py
home.html
intro.html
upload.html

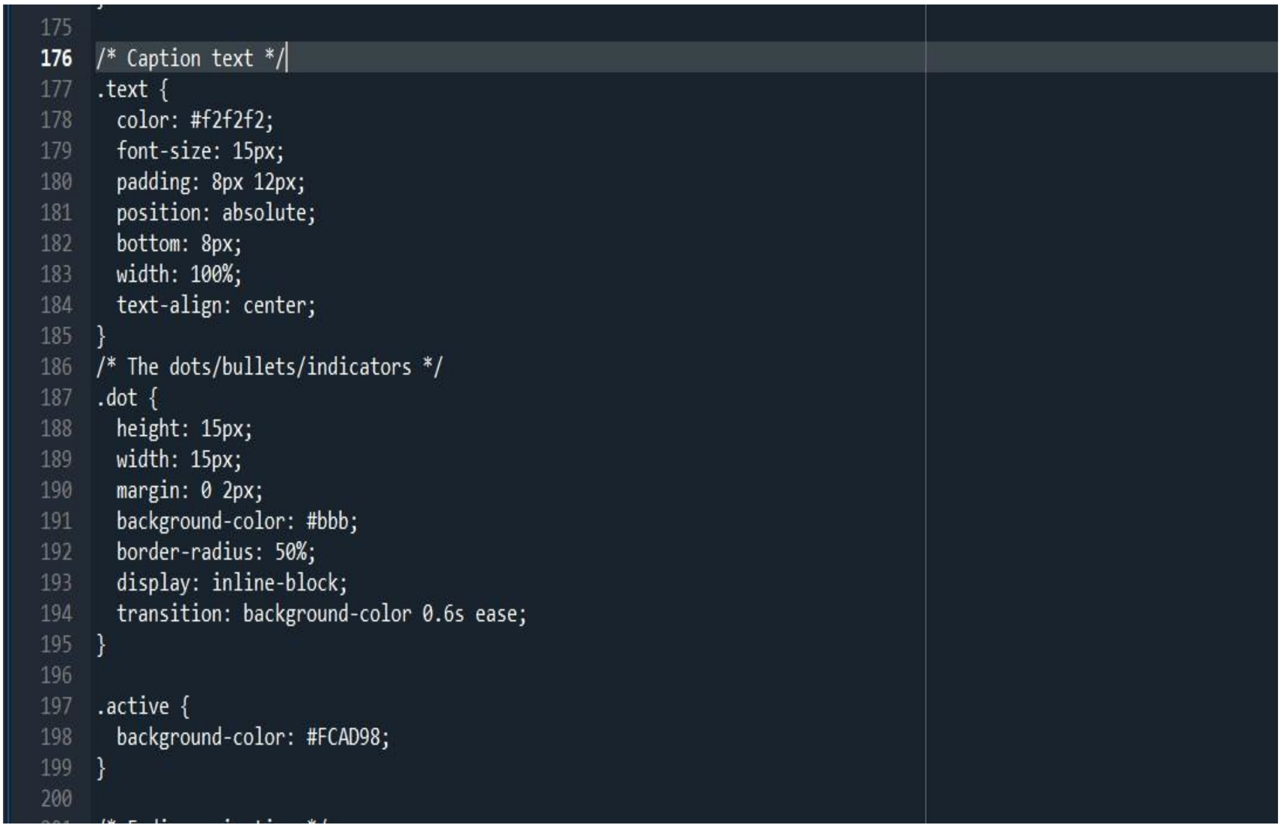
```
<html>
<script>

</script>

<style>
.header {   position: relative;
            top:0;
            margin:0px;
            z-index: 1;
            left: 0px;
            right: 0px;
            position: fixed;
            background-color: #FCAD98 ;
            color: white;
            box-shadow: 0px 8px 2px grey;
            overflow: hidden;
            padding-left:20px;
            font-family: 'Josefin Sans';
            font-size: 2vw;
            width: 100%;
            height:8%;
            text-align: center;
        }
        .topnav {
  overflow: hidden;
  background-color: #FCAD98;
}

.topnav-right a {
  float: left;
  color: black;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
}

.topnav-right a:hover {
  background-color: #FCAD98;
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\home.html
app.py
home.html
intro.html
upload.html

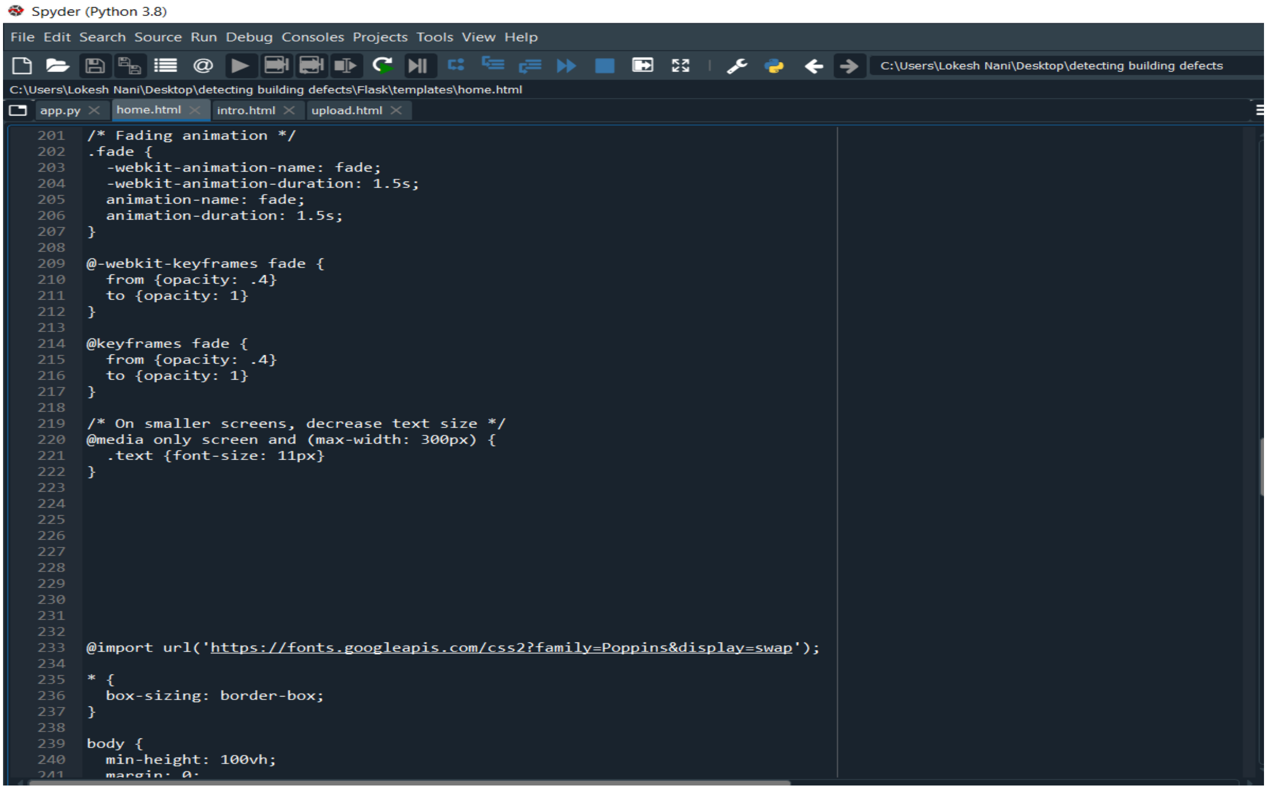
```
  background-color: #FCAD98;
  color: black;
}

.topnav-right a.active {
  background-color: #FCAD98;
  color: white;
}

.topnav-right {
  float: right;
  padding-right:100px;
}

body {
background-image: -webkit-linear-gradient(90deg, skyblue 0%, steelblue 100%);
  background-image: url("");
    background-size: cover;
  background-attachment: fixed;
  background-size: 100% 100%;
  background-color: ;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  }
  .button {
  background-color: #091425;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 12px;
  border-radius: 16px;
}
.button:hover {
  box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\home.html
app.py
home.html
intro.html
upload.html

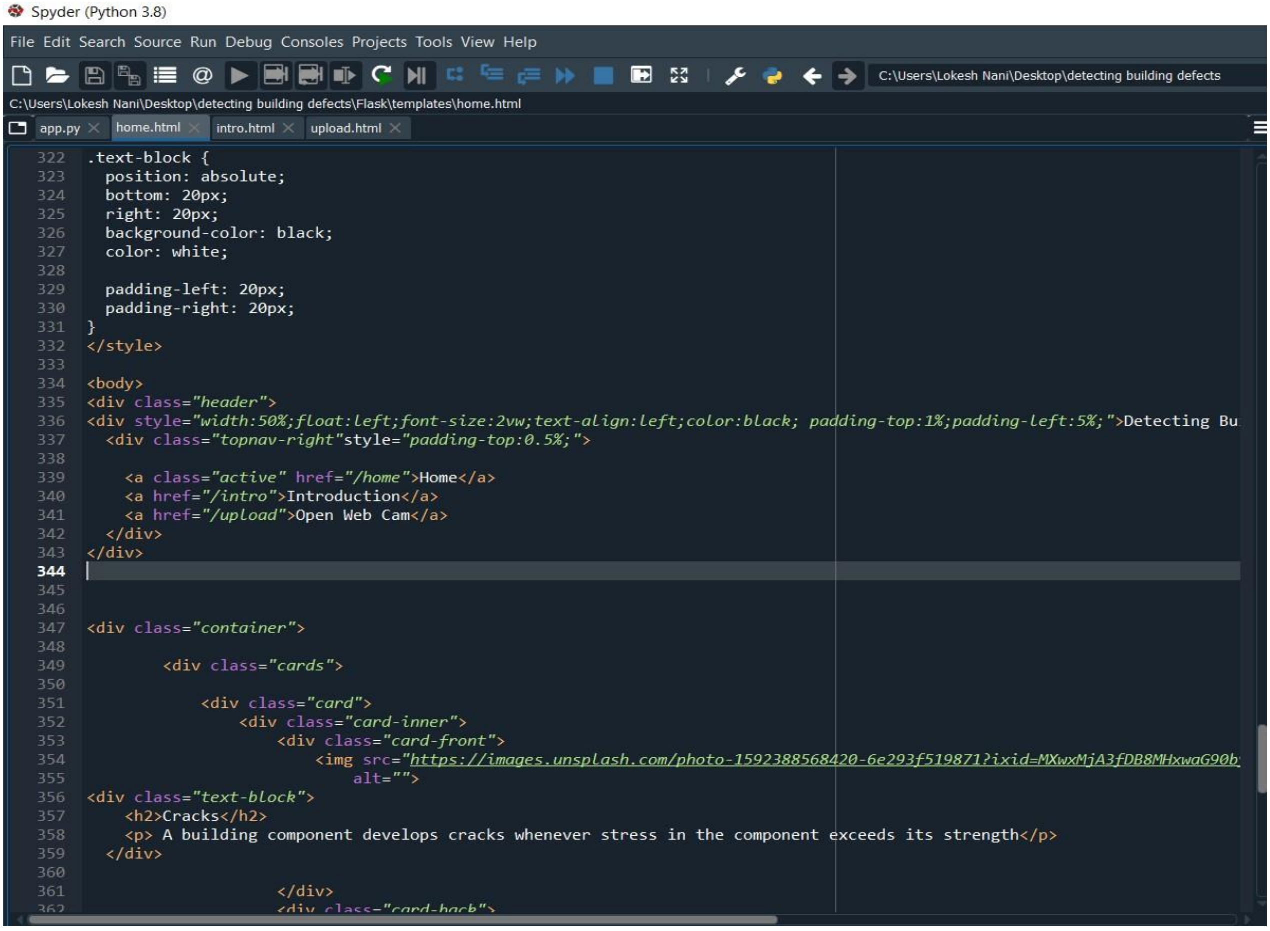
```
input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

button {
  background-color: #091425;
  color: white;
  padding: 14px 20px;
  margin-bottom:10px;
  border: none;
  cursor: pointer;
  width: 17%;
  border-radius:4px;
  font-family:Montserrat;
}

button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
}

img.avatar {
  width: 30%;
  border-radius: 50%;
}
```



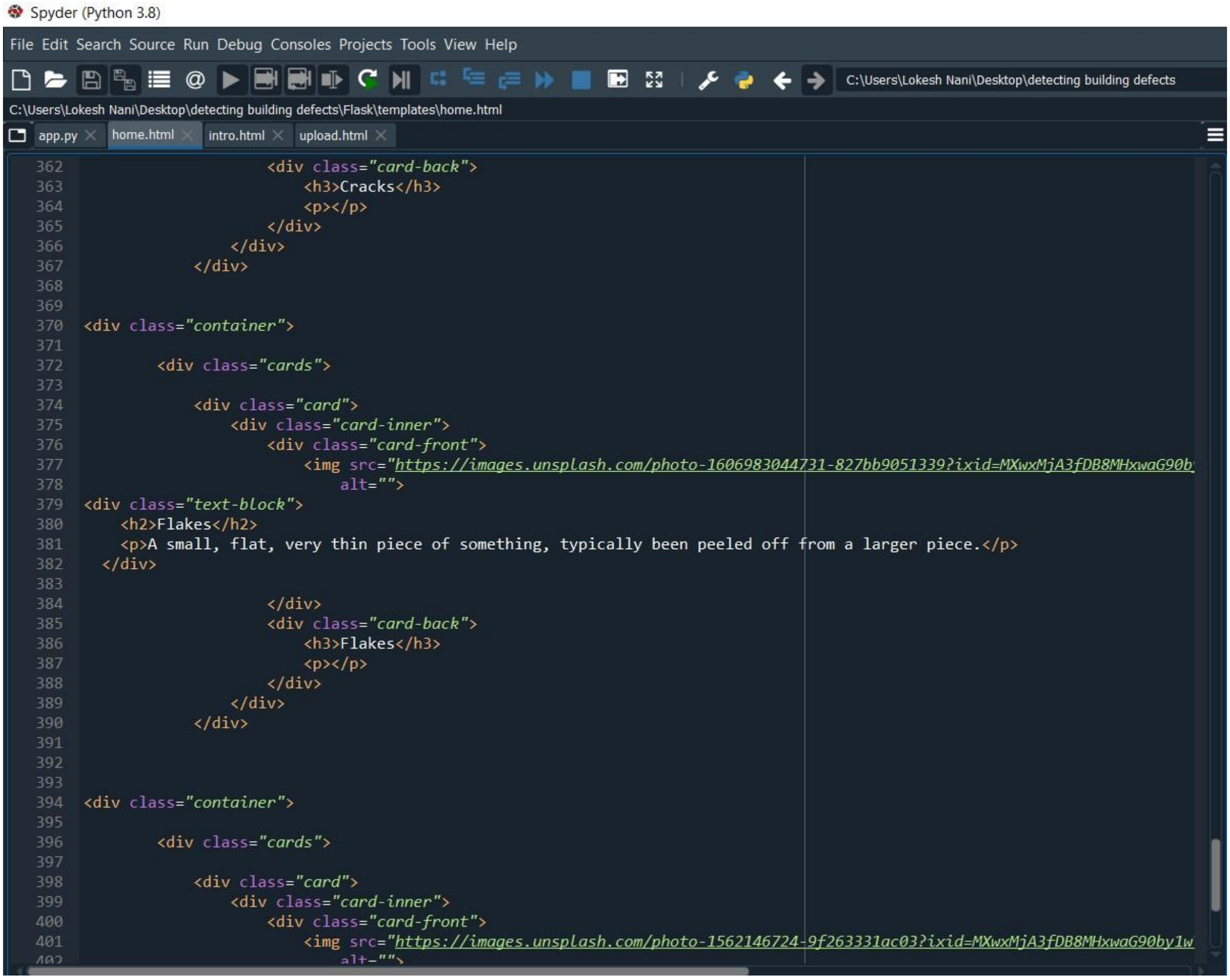
```

/* Caption text */
.text {
  color: #f2f2f2;
  font-size: 15px;
  padding: 8px 12px;
  position: absolute;
  bottom: 8px;
  width: 100%;
  text-align: center;
}
/* The dots/bullets/indicators */
.dot {
  height: 15px;
  width: 15px;
  margin: 0 2px;
  background-color: #bbb;
  border-radius: 50%;
  display: inline-block;
  transition: background-color 0.6s ease;
}

.active {
  background-color: #FCAD98;
}

```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\home.html
app.py
home.html
intro.html
upload.html

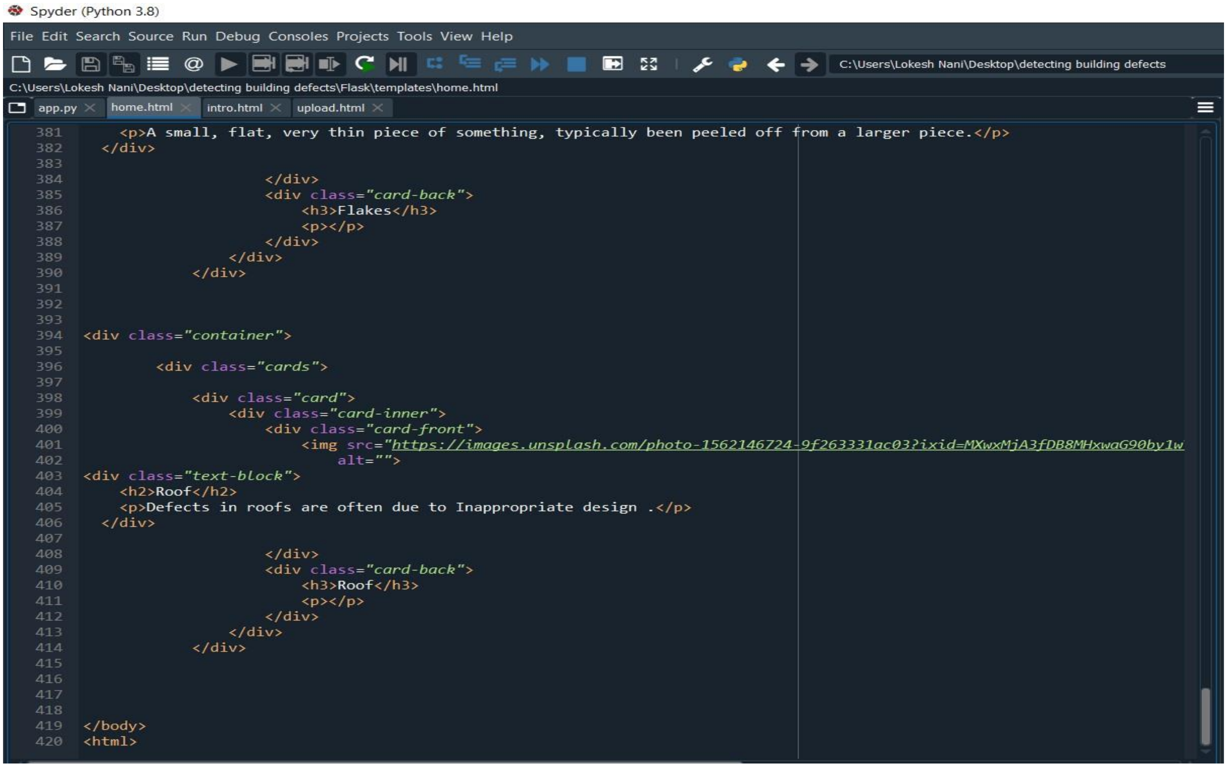
```
/* Fading animation */
.fade {
  -webkit-animation-name: fade;
  -webkit-animation-duration: 1.5s;
  animation-name: fade;
  animation-duration: 1.5s;
}

@-webkit-keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

@keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
  .text {font-size: 11px}
}











@import url('https://fonts.googleapis.com/css2?family=Poppins&display=swap');

* {
  box-sizing: border-box;
}

body {
  min-height: 100vh;
  margin: 0;
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\home.html
app.py
home.html
intro.html
upload.html

```
.text-block {
  position: absolute;
  bottom: 20px;
  right: 20px;
  background-color: black;
  color: white;

  padding-left: 20px;
  padding-right: 20px;
}
</style>

<body>
<div class="header">
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:black; padding-top:1%;padding-left:5%;">Detecting Bu
  <div class="topnav-right"style="padding-top:0.5%;">

    <a class="active" href="/home">Home</a>
    <a href="/intro">Introduction</a>
    <a href="/upload">Open Web Cam</a>
  </div>
</div>



<div class="container">

        <div class="cards">

            <div class="card">
                <div class="card-inner">
                    <div class="card-front">
                        <img src="https://images.unsplash.com/photo-1592388568420-6e293f519871?ixid=MXwxMjA3fDB8MHxwaG90b
                            alt="">
<div class="text-block">
    <h2>Cracks</h2>
    <p> A building component develops cracks whenever stress in the component exceeds its strength</p>
  </div>

                    </div>
                    <div class="card-back">
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\home.html
app.py
home.html
intro.html
upload.html

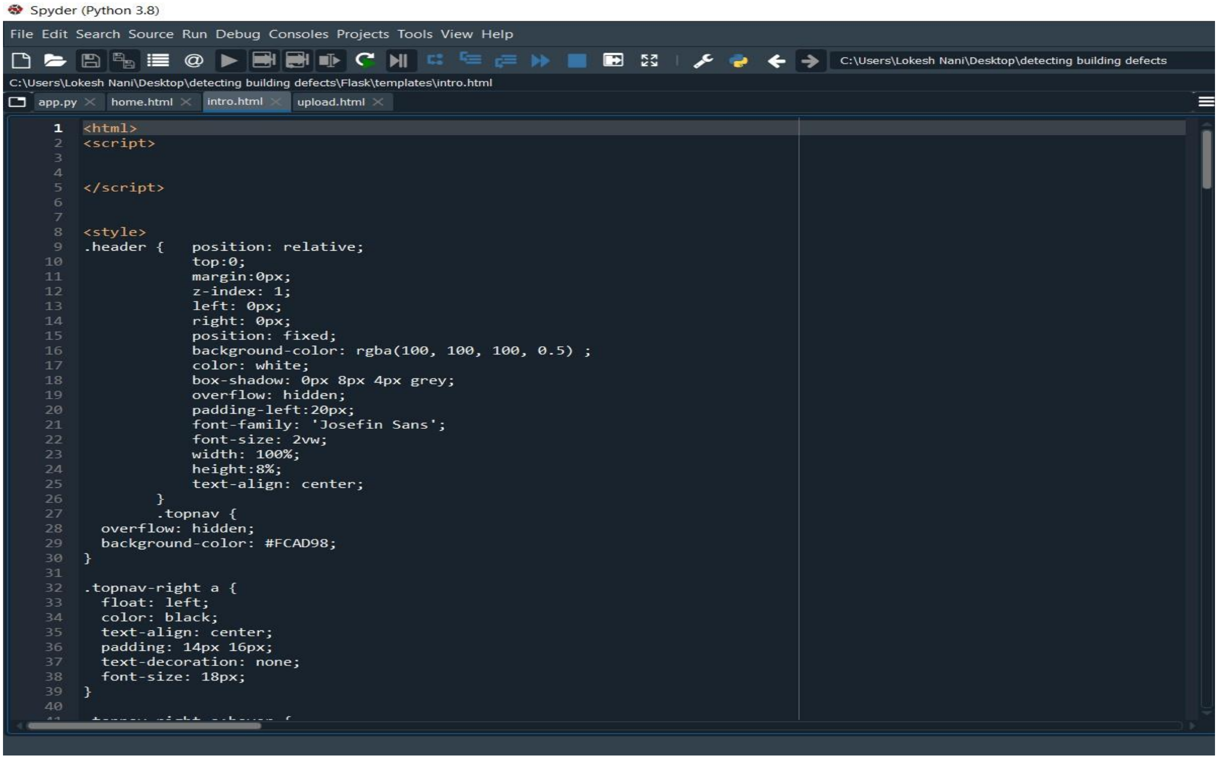
```
                    <div class="card-back">
                        <h3>Cracks</h3>
                        <p></p>
                    </div>
                </div>
            </div>


<div class="container">

        <div class="cards">

            <div class="card">
                <div class="card-inner">
                    <div class="card-front">
                        <img src="https://images.unsplash.com/photo-1606983044731-827bb9051339?ixid=MXwxMjA3fDB8MHxwaG90b
                            alt="">
<div class="text-block">
    <h2>Flakes</h2>
    <p>A small, flat, very thin piece of something, typically been peeled off from a larger piece.</p>
  </div>

                    </div>
                    <div class="card-back">
                        <h3>Flakes</h3>
                        <p></p>
                    </div>
                </div>
            </div>



<div class="container">

        <div class="cards">

            <div class="card">
                <div class="card-inner">
                    <div class="card-front">
                        <img src="https://images.unsplash.com/photo-1562146724-9f263331ac03?ixid=MXwxMjA3fDB8MHxwaG90by1w
                            alt="">
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\home.html
app.py
home.html
intro.html
upload.html

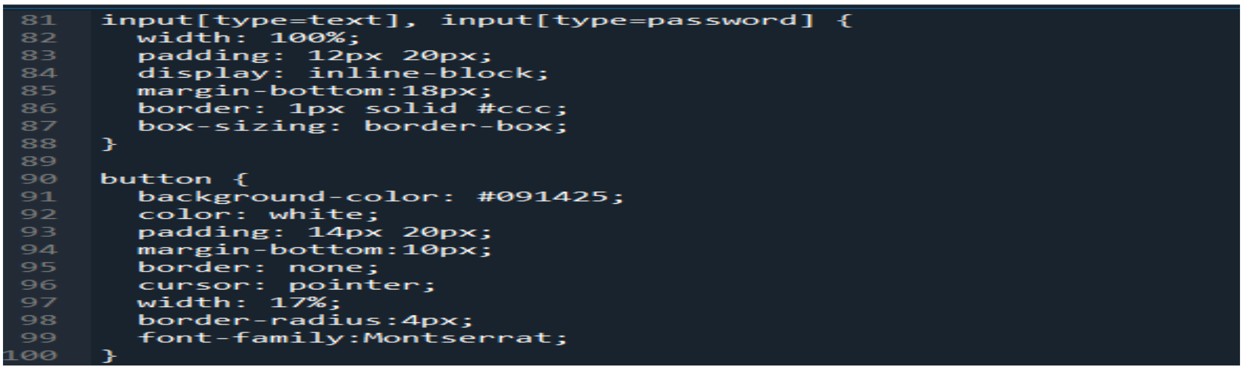
```
    <p>A small, flat, very thin piece of something, typically been peeled off from a larger piece.</p>
  </div>

                    </div>
                    <div class="card-back">
                        <h3>Flakes</h3>
                        <p></p>
                    </div>
                </div>
            </div>



<div class="container">

        <div class="cards">

            <div class="card">
                <div class="card-inner">
                    <div class="card-front">
                        <img src="https://images.unsplash.com/photo-1562146724-9f263331ac03?ixid=MXwxMjA3fDB8MHxwaG90by1w
                            alt="">
<div class="text-block">
    <h2>Roof</h2>
    <p>Defects in roofs are often due to Inappropriate design .</p>
  </div>

                    </div>
                    <div class="card-back">
                        <h3>Roof</h3>
                        <p></p>
                    </div>
                </div>
            </div>




</body>
<html>
```

Figure 11: Home.html page is the code for home page of our Web Application

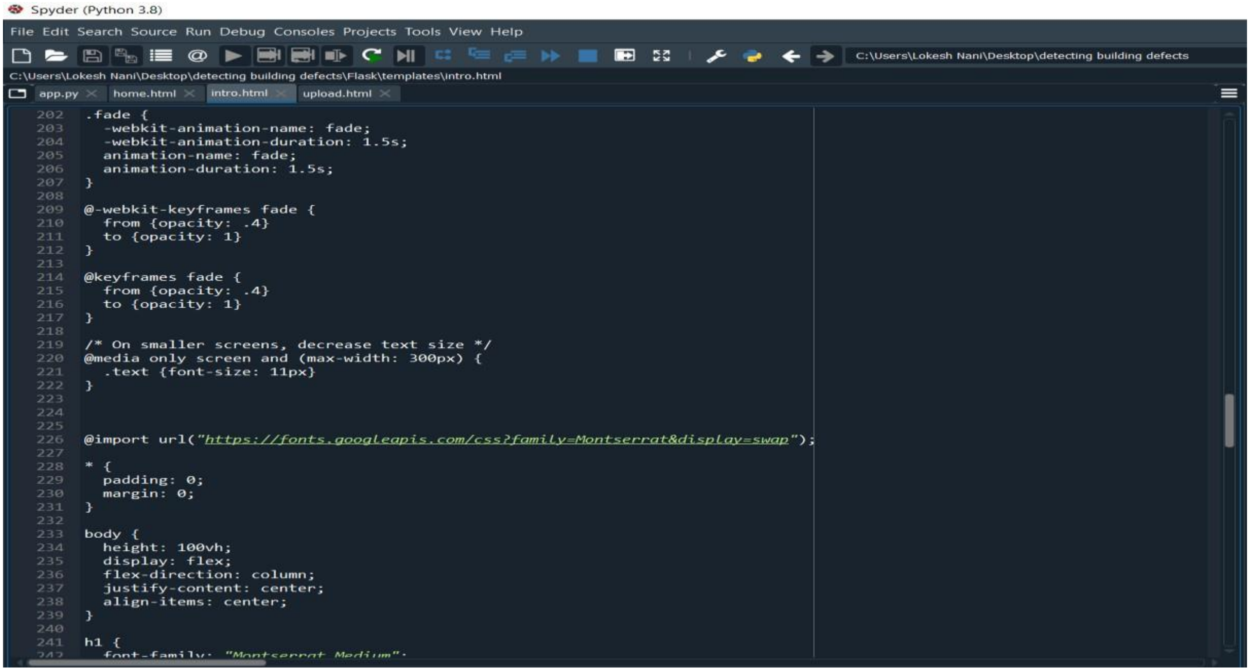
- **INTRO.HTML:**


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\intro.html
app.py
home.html
intro.html
upload.html

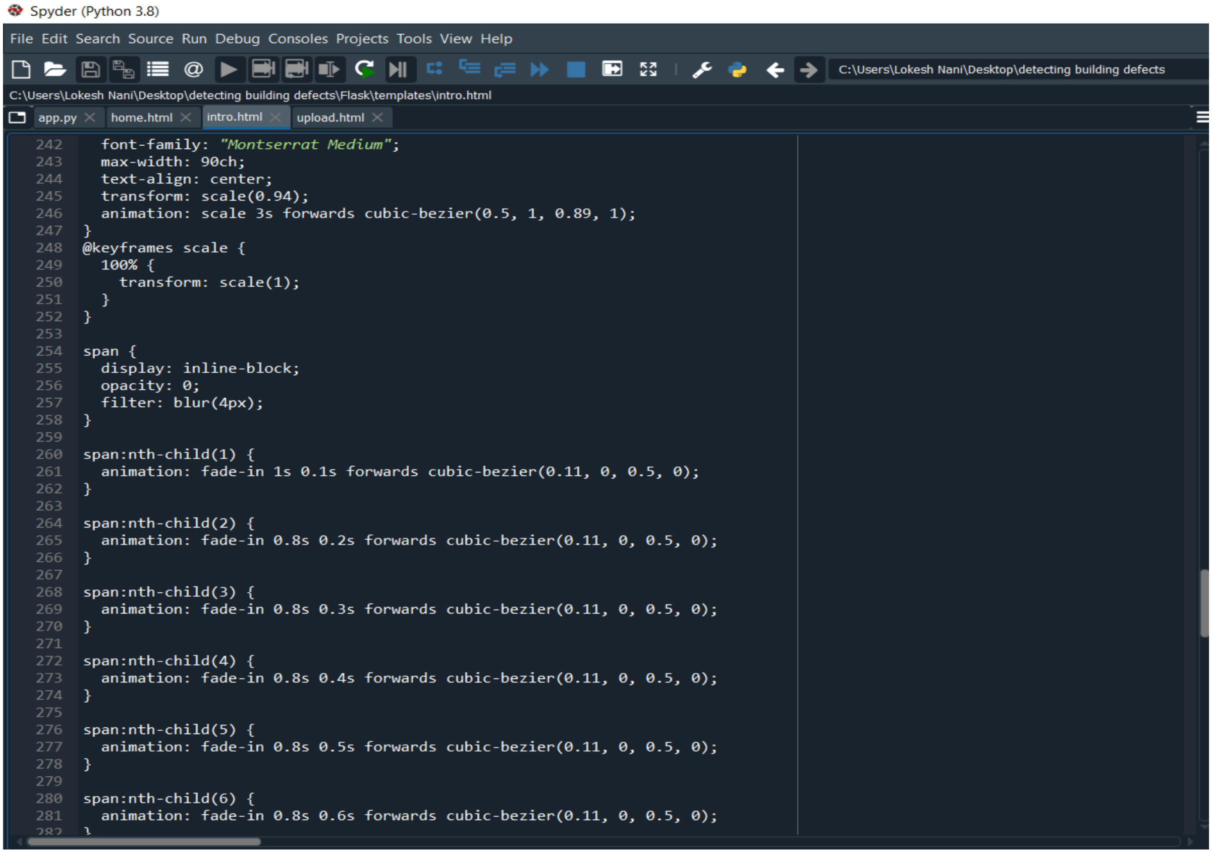
```
<html>
<script>


</script>


<style>
.header {   position: relative;
            top:0;
            margin:0px;
            z-index: 1;
            left: 0px;
            right: 0px;
            position: fixed;
            background-color: rgba(100, 100, 100, 0.5) ;
            color: white;
            box-shadow: 0px 8px 4px grey;
            overflow: hidden;
            padding-left:20px;
            font-family: 'Josefin Sans';
            font-size: 2vw;
            width: 100%;
            height:8%;
            text-align: center;
        }
        .topnav {
  overflow: hidden;
  background-color: #FCAD98;
}

.topnav-right a {
  float: left;
  color: black;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
}

.topnav-right a:hover {
```



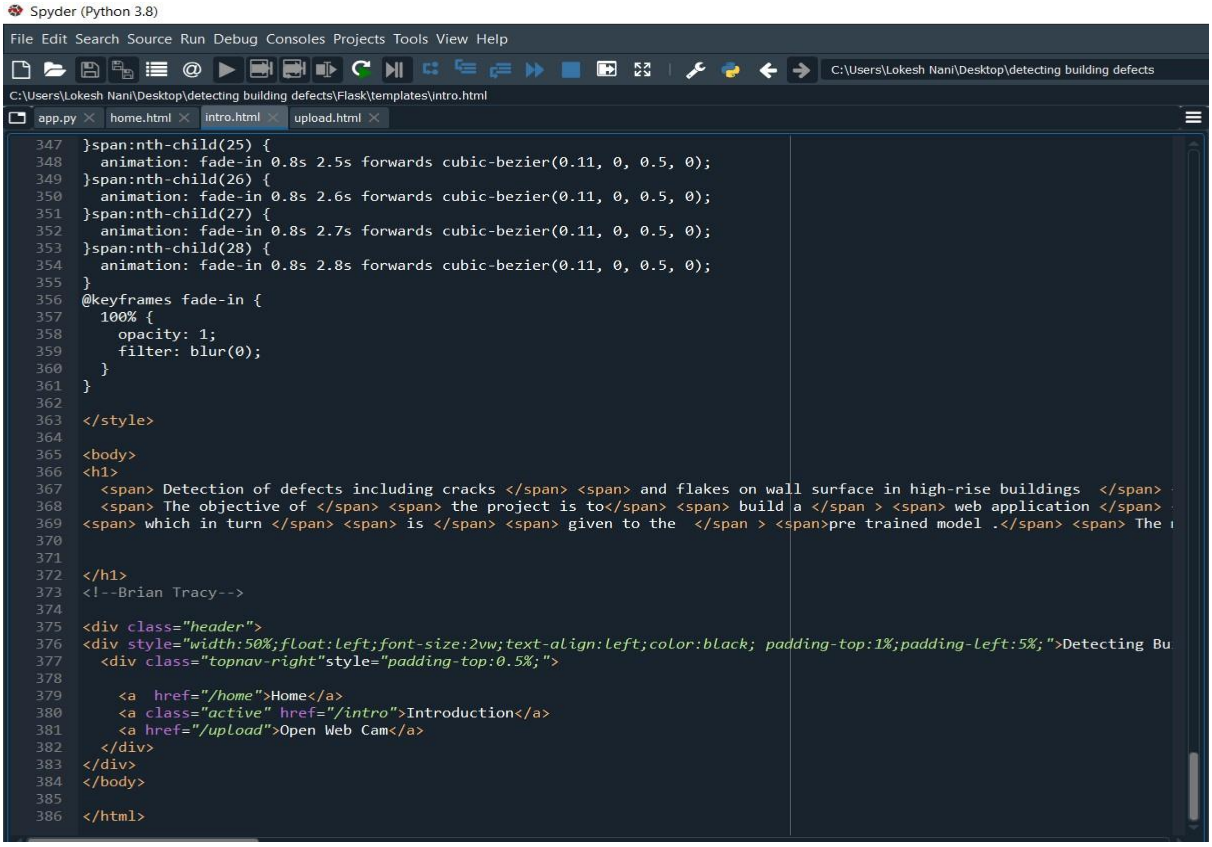
```
input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

button {
  background-color: #091425;
  color: white;
  padding: 14px 20px;
  margin-bottom:10px;
  border: none;
  cursor: pointer;
  width: 17%;
  border-radius:4px;
  font-family:Montserrat;
}
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\intro.html
app.py
home.html
intro.html
upload.html

```
.fade {
  -webkit-animation-name: fade;
  -webkit-animation-duration: 1.5s;
  animation-name: fade;
  animation-duration: 1.5s;
}

@-webkit-keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

@keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
  .text {font-size: 11px}
}



@import url("https://fonts.googleapis.com/css?family=Montserrat&display=swap");

* {
  padding: 0;
  margin: 0;
}

body {
  height: 100vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}

h1 {
  font-family: "Montserrat Medium";
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\intro.html
app.py
home.html
intro.html
upload.html

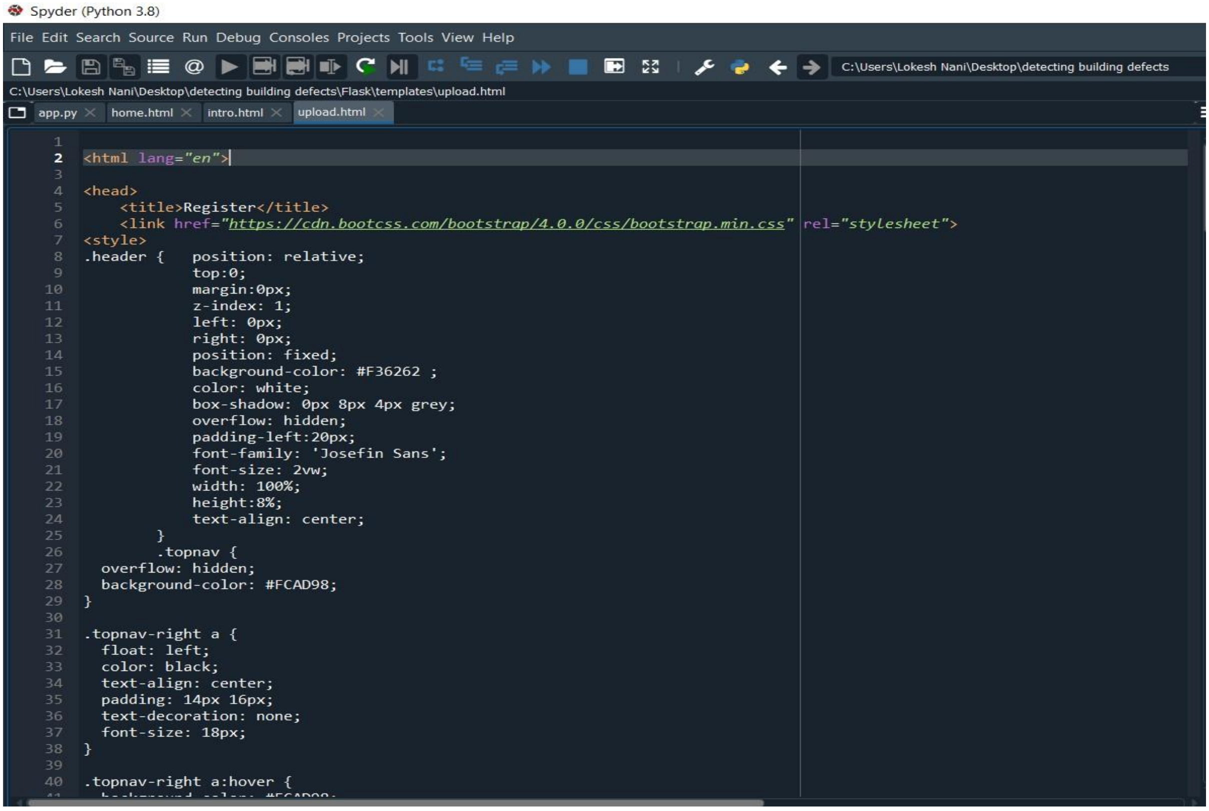
```
  font-family: "Montserrat Medium";
  max-width: 90ch;
  text-align: center;
  transform: scale(0.94);
  animation: scale 3s forwards cubic-bezier(0.5, 1, 0.89, 1);
}
@keyframes scale {
  100% {
    transform: scale(1);
  }
}

span {
  display: inline-block;
  opacity: 0;
  filter: blur(4px);
}

span:nth-child(1) {
  animation: fade-in 1s 0.1s forwards cubic-bezier(0.11, 0, 0.5, 0);
}

span:nth-child(2) {
  animation: fade-in 0.8s 0.2s forwards cubic-bezier(0.11, 0, 0.5, 0);
}

span:nth-child(3) {
  animation: fade-in 0.8s 0.3s forwards cubic-bezier(0.11, 0, 0.5, 0);
}

span:nth-child(4) {
  animation: fade-in 0.8s 0.4s forwards cubic-bezier(0.11, 0, 0.5, 0);
}

span:nth-child(5) {
  animation: fade-in 0.8s 0.5s forwards cubic-bezier(0.11, 0, 0.5, 0);
}

span:nth-child(6) {
  animation: fade-in 0.8s 0.6s forwards cubic-bezier(0.11, 0, 0.5, 0);
}
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\intro.html
app.py
home.html
intro.html
upload.html

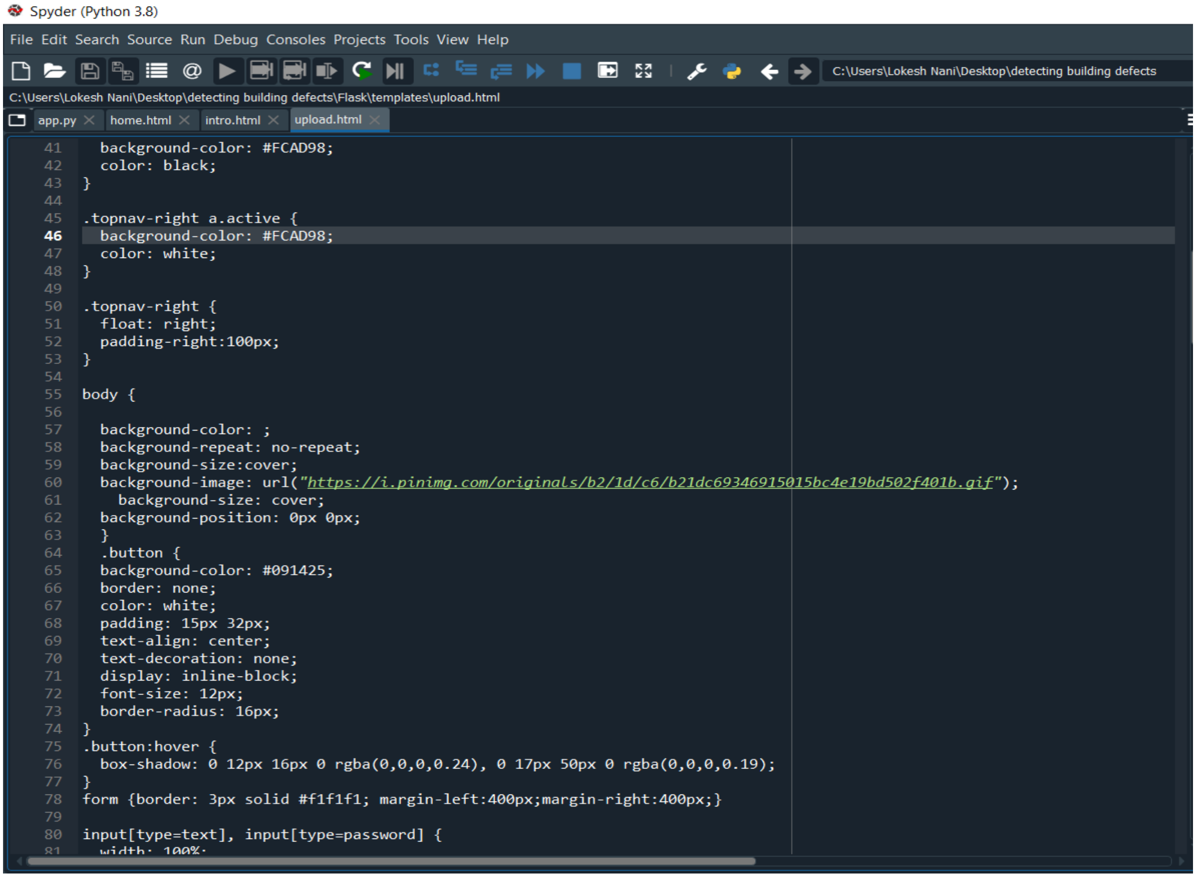
```
}span:nth-child(25) {
  animation: fade-in 0.8s 2.5s forwards cubic-bezier(0.11, 0, 0.5, 0);
}span:nth-child(26) {
  animation: fade-in 0.8s 2.6s forwards cubic-bezier(0.11, 0, 0.5, 0);
}span:nth-child(27) {
  animation: fade-in 0.8s 2.7s forwards cubic-bezier(0.11, 0, 0.5, 0);
}span:nth-child(28) {
  animation: fade-in 0.8s 2.8s forwards cubic-bezier(0.11, 0, 0.5, 0);
}
@keyframes fade-in {
  100% {
    opacity: 1;
    filter: blur(0);
  }
}

</style>

<body>
<h1>
  <span> Detection of defects including cracks </span> <span> and flakes on wall surface in high-rise buildings  </span>
  <span> The objective of </span> <span> the project is to</span> <span> build a </span > <span> web application </span>
<span> which in turn </span> <span> is </span> <span> given to the  </span > <span>pre trained model .</span> <span> The


</h1>
<!--Brian Tracy-->

<div class="header">
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:black; padding-top:1%;padding-left:5%;">Detecting Bu
  <div class="topnav-right"style="padding-top:0.5%;">

    <a  href="/home">Home</a>
    <a class="active" href="/intro">Introduction</a>
    <a href="/upload">Open Web Cam</a>
  </div>
</div>
</body>

</html>
```

Figure 12. Intro.html is the page Which displays an introduction about the project

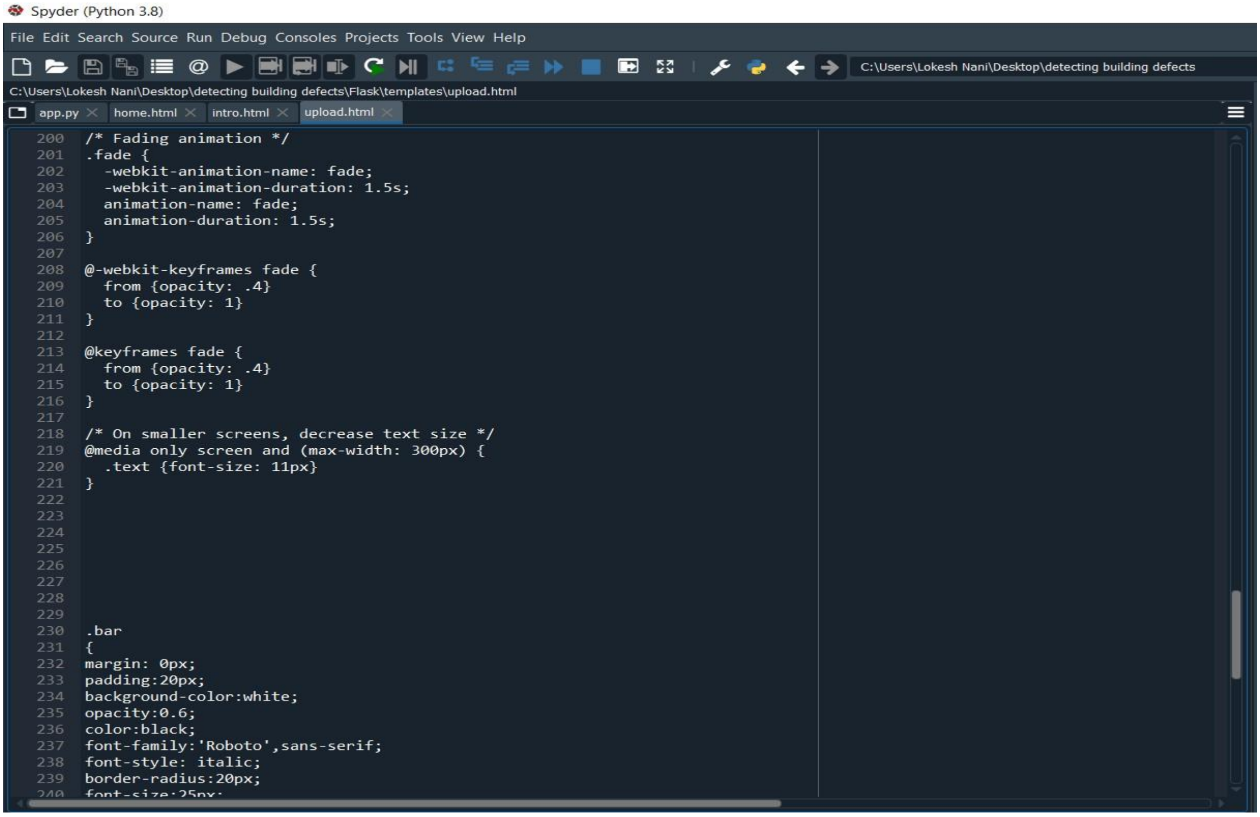
- **UPLOAD.HTML:**


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\upload.html
app.py
home.html
intro.html
upload.html

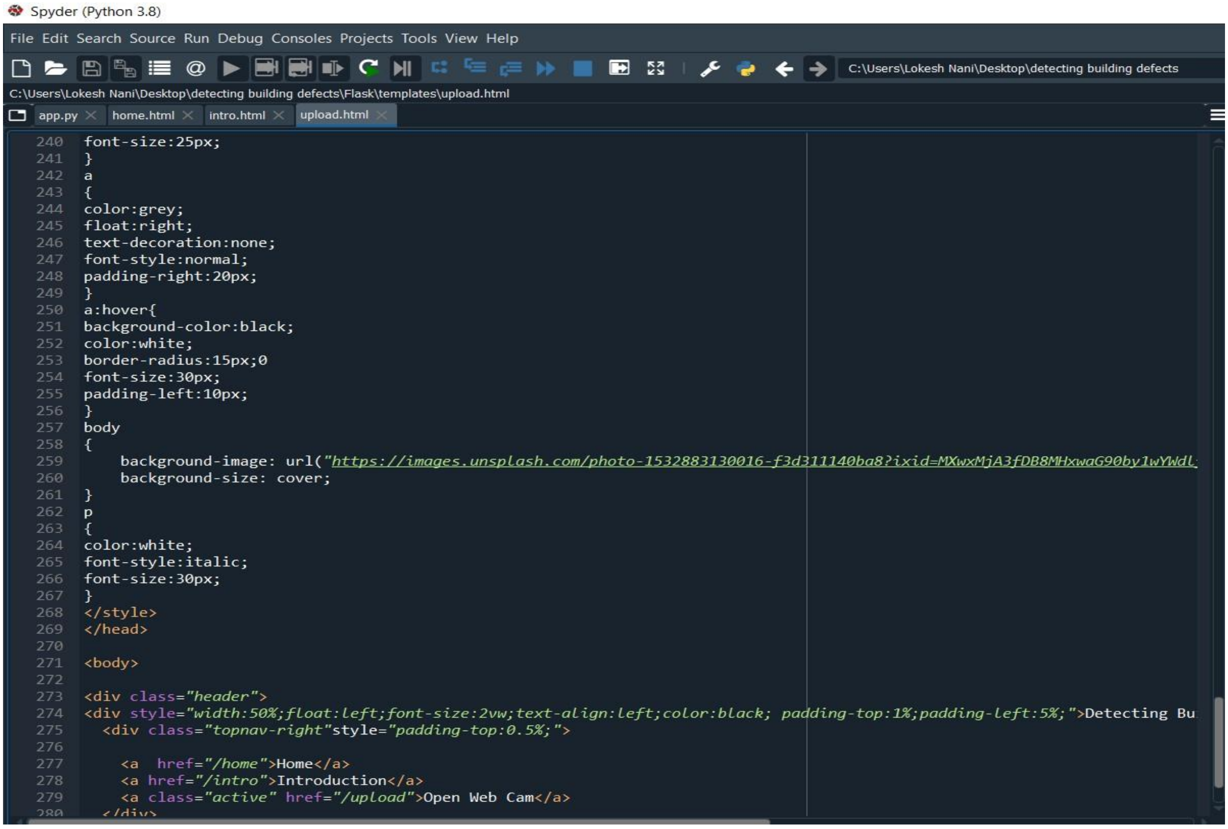
```

<html lang="en">

<head>
    <title>Register</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
<style>
.header {   position: relative;
            top:0;
            margin:0px;
            z-index: 1;
            left: 0px;
            right: 0px;
            position: fixed;
            background-color: #F36262 ;
            color: white;
            box-shadow: 0px 8px 4px grey;
            overflow: hidden;
            padding-left:20px;
            font-family: 'Josefin Sans';
            font-size: 2vw;
            width: 100%;
            height:8%;
            text-align: center;
        }
        .topnav {
  overflow: hidden;
  background-color: #FCAD98;
}

.topnav-right a {
  float: left;
  color: black;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
}

.topnav-right a:hover {
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\upload.html
app.py
home.html
intro.html
upload.html

```
  background-color: #FCAD98;
  color: black;
}

.topnav-right a.active {
  background-color: #FCAD98;
  color: white;
}

.topnav-right {
  float: right;
  padding-right:100px;
}

body {

  background-color: ;
  background-repeat: no-repeat;
  background-size:cover;
  background-image: url("https://i.pinimg.com/originals/b2/1d/c6/b21dc69346915015bc4e19bd502f401b.gif");
    background-size: cover;
  background-position: 0px 0px;
  }
  .button {
  background-color: #091425;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 12px;
  border-radius: 16px;
}
.button:hover {
  box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

input[type=text], input[type=password] {
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\upload.html
app.py
home.html
intro.html
upload.html

```
/* Fading animation */
.fade {
  -webkit-animation-name: fade;
  -webkit-animation-duration: 1.5s;
  animation-name: fade;
  animation-duration: 1.5s;
}

@-webkit-keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

@keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
  .text {font-size: 11px}
}









.bar
{
margin: 0px;
padding:20px;
background-color:white;
opacity:0.6;
color:black;
font-family:'Roboto',sans-serif;
font-style: italic;
border-radius:20px;
```

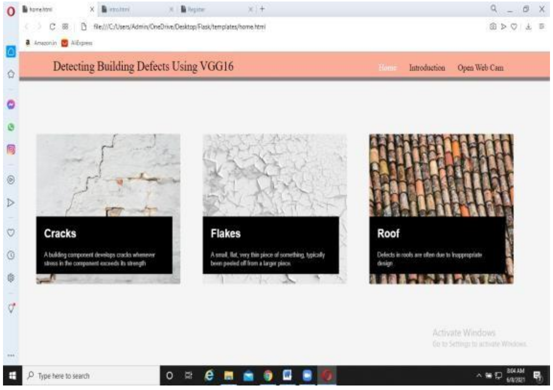

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Lokesh Nani\Desktop\detecting building defects
C:\Users\Lokesh Nani\Desktop\detecting building defects\Flask\templates\upload.html
app.py
home.html
intro.html
upload.html

```
font-size:25px;
}
a
{
color:grey;
float:right;
text-decoration:none;
font-style:normal;
padding-right:20px;
}
a:hover{
background-color:black;
color:white;
border-radius:15px;0
font-size:30px;
padding-left:10px;
}
body
{
    background-image: url("https://images.unsplash.com/photo-1532883130016-f3d311140ba8?ixid=MXwxMjA3fDB8MHxwaG90by1wYWdl
    background-size: cover;
}
p
{
color:white;
font-style:italic;
font-size:30px;
}
</style>
</head>

<body>

<div class="header">
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:black; padding-top:1%;padding-left:5%;">Detecting Bu
  <div class="topnav-right"style="padding-top:0.5%;">

    <a  href="/home">Home</a>
    <a href="/intro">Introduction</a>
    <a class="active" href="/upload">Open Web Cam</a>
  </div>
```

Figure 13. Upload.html is the page which displays an Emergency alert

3. CONCLUSION

- **We finally created three web pages in order to translate text from one language to the other language.**
    - In the output language section, we included executing local host in browser.
    - An image is given as an input from the web cam and the output is in the form of text whether it is cracks, flakes or roof.
    - Here are the screenshot images of our project.



Detecting Building Defects Using VGG16

Cracks

A building component develops cracks whenever stress in the component exceeds its strength

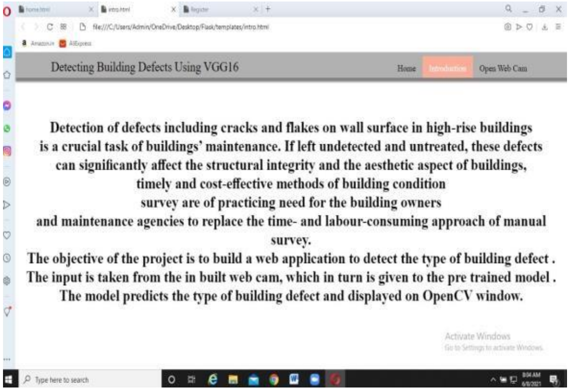
Flakes

A small, flat, very thin piece of something, typically been peeled off from a larger piece.

Roof

Defects in roofs are often due to inappropriate design.

Figure 14. Displays the home page



Detecting Building Defects Using VGG16

Detection of defects including cracks and flakes on wall surface in high-rise buildings is a crucial task of buildings' maintenance. If left undetected and untreated, these defects can significantly affect the structural integrity and the aesthetic aspect of buildings, timely and cost-effective methods of building condition survey are of practicing need for the building owners and maintenance agencies to replace the time- and labour-consuming approach of manual survey.

The objective of the project is to build a web application to detect the type of building defect . The input is taken from the in built web cam, which in turn is given to the pre trained model . The model predicts the type of building defect and displayed on OpenCV window.

Figure 15. Displays an introduction about the project

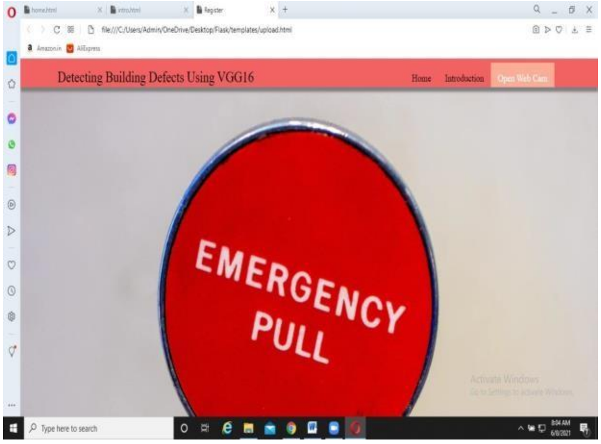
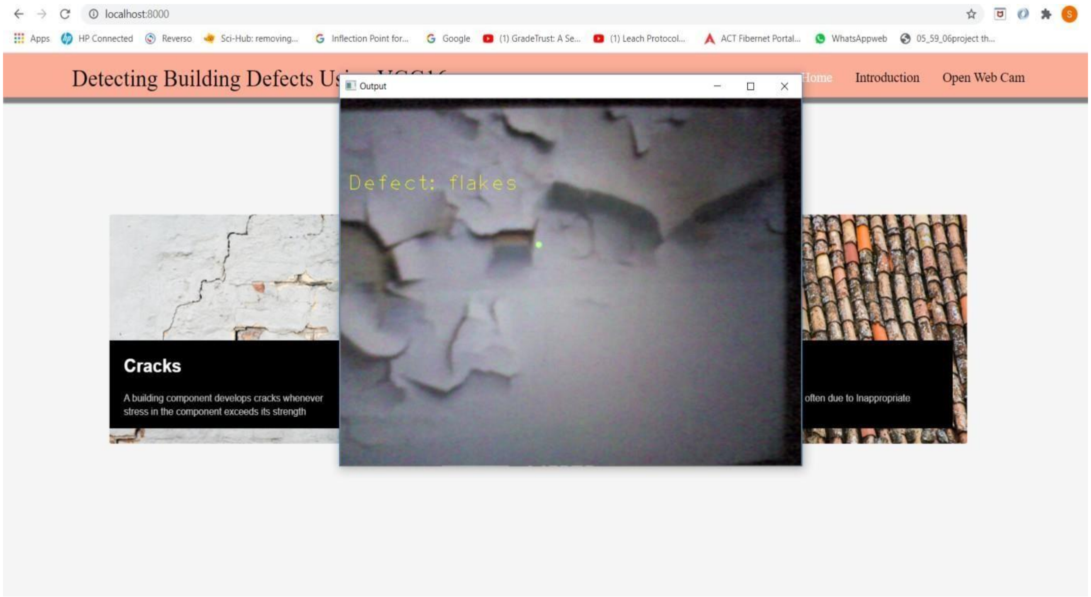

home.html
intro.html
Register
file:///C:/Users/Admin/OneDrive/Desktop/Flask/templates/upload.html
Amazon.in
AliExpress
Detecting Building Defects Using VGG16
Home
Introduction
Open Web Cam
EMERGENCY
PULL
Activate Windows
Go to Settings to activate Windows.
Type here to search
8:04 AM
6/8/2021

Figure 16. Displays an Emergency alert

These are the screenshots of the output of our project. When we have given the wall images through our webcam as well as our disc. The output is displayed in text form.



A building component develops cracks whenever stress in the component exceeds its strength

often due to Inappropriate

Figure 17. Output of our project predictions displayed like this

4.ADVANTAGES

- Little to no equipment needed
- Easy to train
- Portable
- Minimum Part Preparation
- Perfect prediction of the building defects and very accurate performance calculations.

5.DISADVANTAGES

- Surface indications only
- Generally, only able to detect large flaws
- Possible misinterpretation of flaws
- User should have the idea on all the parameters and units of each parameter.

6.APPLICATIONS

- Thermography methodologies for detecting energy related building defects
- Time-lapse thermography for building defect detection
- Improvingthedetectionofthermalbridgesinbuildingsviaon-siteinfrared thermography
- Geophysical methods for defect detection in architectural structures.

7.FUTURE SCOPE

For the future works, the challenges and limitation that we were facing this in paper will be addressed. The presented paper had to set a number of limitations, i.e., firstly, multiple types of the defects are not considered at once. This means that the images considered by the model belonged to only one category.

Secondly, only the images with visible defects are considered. Thirdly, consideration of the extreme lighting and orientation, e.g., low lighting, too bright images, are not included in this study. In the future works, these limitations will be considered to be able to get closer to the concept of a fully automated detection.

Through fully satisfying these challenges and limitations, our present work will be evolved into a software application to perform real-time detection of defects using vision sensors including drones. The work will also be extended to cover other models that can detect other defects in construction such as cracks, structural movements, spalling and corrosion. Our long-term vision includes plans to create a large, open-source database of different building and construction defects which will support world-wide research on condition assessment of built assets.

8.BIBILOGRAPHY

1. Agdas, D.; Rice, J. A.; Martinez, J. R.; Lasa, I. R., Comparison of visual inspection and structural health monitoring as bridge condition assessment methods. Journal of Performance of Constructed Facilities 2015, 30, (3), 04015049.
2. Shamshirband, S.; Mosavi, A.; Rabczuk, T., Particle swarm optimization model to predict scour depth around bridge pier. arXiv preprint arXiv:1906.08863 2019.
3. Zhang, Y.; Anderson, N.; Bland, S.; Nutt, S.; Jursich, G.; Joshi, S., All printed strain sensors: Building blocks of the aircraft structural health monitoring system. Sensors and Actuators A: Physical 2017, 253, 165-172.
4. Noel, A. B.; Abdaoui, A.; Elfouly, T.; Ahmed, M. H.; Badawy, A.; Shehata, M. S., Structural health monitoring using wireless sensor networks: A comprehensive survey. IEEE Communications Surveys & Tutorials 2017, 19, (3),1403-1423.
5. Kong, Q.; Allen, R. M.; Kohler, M. D.; Heaton, T. H.; Bunn, J., Structural health monitoring of buildings using smartphone sensors. Seismological Research Letters 2018, 89, (2A), 594-602.
6. Song, G.; Wang, C.; Wang, B., Structural health monitoring (SHM) of civil structures. In Multidisciplinary Digital Publishing Institute: 2017.
7. Annamdas, V. G. M.; Bhalla, S.; Soh, C. K., Applications of structural health monitoring technology in Asia. Structural Health Monitoring 2017, 16, (3), 324-346.
8. Lorenzoni, F.; Casarin, F.; Caldon, M.; Islami, K.; Modena, C., Uncertainty quantification in structural health monitoring: Applications on cultural heritage buildings. Mechanical Systems and Signal Processing 2016,66, 268-281.
9. Maguire, M.; Dorafshan, S.; Thomas, R. SDNET2018: A concrete crack image dataset for machine learning applications. Browse Datasets 2018.

9.HELP FILE

PROJECT EXECUTION:

STEP-1: Go to **Start,** search and launch **ANACONDA NAVIGATOR.**

STEP-2: After launching of **ANACONDA NAVIGATOR,** launch **JUPYTER NOTEBOOK.**

STEP-3: Open **"Major project code" IPYNB file.**

STEP-4: Then run all the cells.

STEP-5: All the **data preprocessing**, **training and testing**, **model building**, **accuracy** of the model can be showcased.

STEP-6: And a pickle file will be generated.

STEP-7: Create a Folder named **FLASK** on the **DESKTOP.** Extract the pickle file into this Flask Folder.

STEP-8: Extract all the html files (home.html, intro.html, upload.html) and python file(app.py) into the **FLASK Folder** and extract this flask into project file.

STEP-9: Then go back to **ANACONDA NAVIGATOR** and the launch the **SPYDER.**

STEP-10: After launching Spyder, give the path of **FLASK FOLDER**.

STEP-11: Open all the app.py and html files present in the Flask Folder.

STEP-12: After running of the app.py, open **ANACONDA PROMPT** and follow the below steps:

cd File Path→click enter

python app.py→click enter (We could see running of files).

STEP-13: Then open **BROWSER,** at the URL area type **—localhost:8000".**

STEP-14: Home page of the project will be displayed.

STEP-15: Click on **—introduction.** it will be displayed few sentences of introduction of our project.

STEP-16: Then click on **—Open Web Cam,** integrated web cam to capture the video frame of building defect then the type of building defect is identified and showcased on the OpenCV window and emergency pull is initiated.