Uncovering the Hidden Treasures of the Mushroom Kingdom: A Classification Analysis

PROJECT ID : NM2023TMID15019

FACULTY MENTOR : Ms.M.Sasirekha

Project Member :

- D. Samrat
- R. Akash
- K. Aswin Raj
- A. Dinakaran
- K. Thennavan

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER NO. TITLE PAGE NO.

CHAPTER 1

INTRODUCTION

1.1 Project Overview:

This project aims to classify different species of mushrooms based on their images. The classification is done using deep learning methods, specifically transfer learning using popular models Inception. The focus is on identifying the cap, gills underside of cap, and astern, which are key features for the optical recognition of mushroom species. The identified species are Boletus, Lactarius, and Russula.

The classification task involves three major categories of mushrooms: Boletus, Lactarius, and Russula. These categories encompass a wide range of species found across different regions of our planet. Boletus mushrooms are known for their distinctive cap shapes and pore-covered undersides, while Lactarius mushrooms often exhibit vibrant colors and produce a milky latex when damaged. Russula mushrooms, on the other hand, showcase diverse cap and stem characteristics and are an intriguing group to explorewhich are found in various habitats like forests, fields, and decomposing logs. Mushrooms have different shapes, sizes, and colors and are used for food, medicine, and other purposes. By leveraging deep learning techniques and transfer learning, this project aims to improve the accuracy and efficiency of mushroom species classification.

1.2 Purpose :

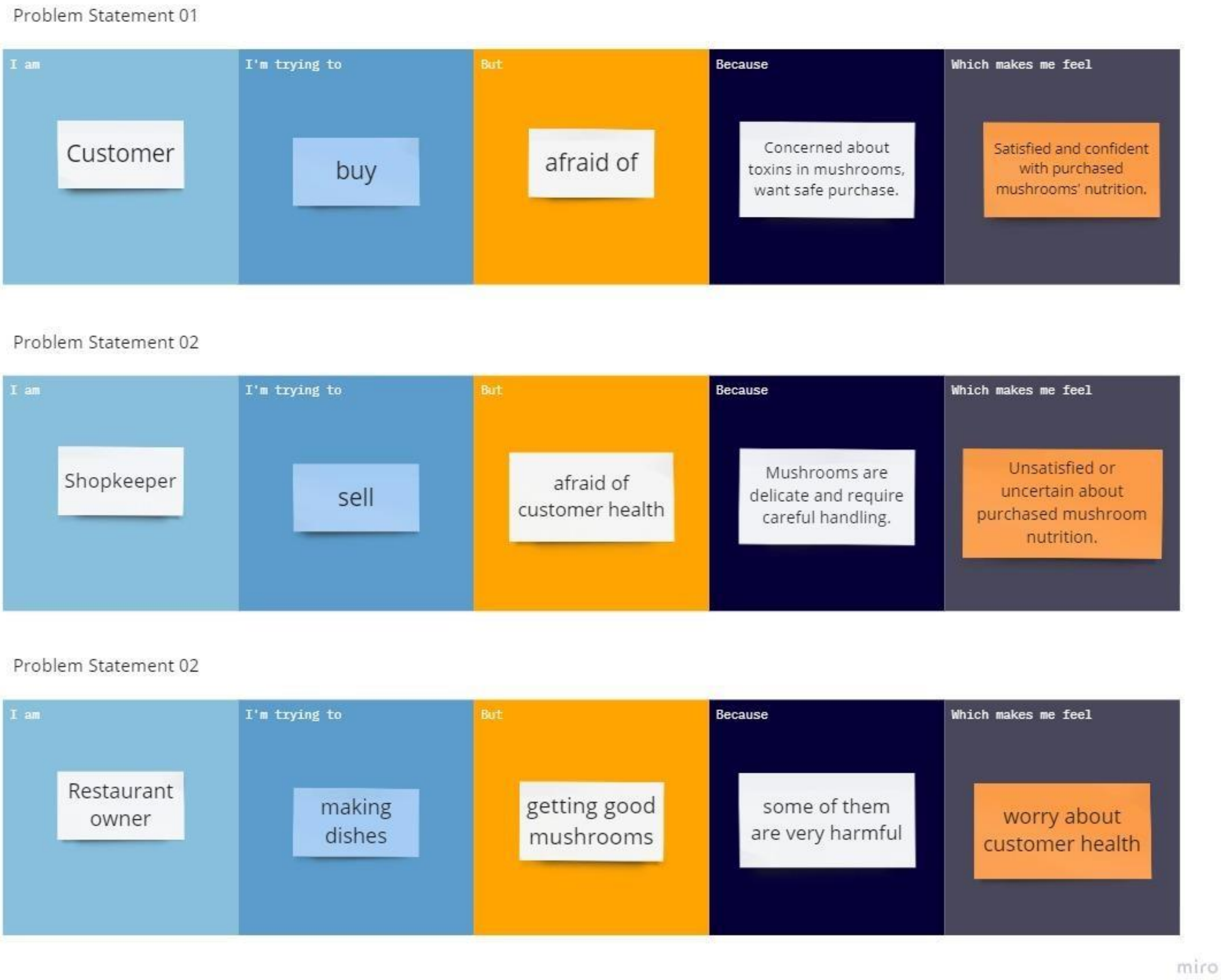
The purpose of this project is to develop a robust and accurate system for optical recognition and classification of mushroom species based on their visual characteristics. By leveraging deep learning techniques and transfer learning, this project aims to enhance the efficiency and accuracy of mushroom species identification. The project not only contributes to the field of mycology but also holds ecological significance by aiding in the study and conservation of mushroom species. Additionally, the system has practical applications in culinary and medicinal domains, enabling the identification of edible and medicinal mushrooms. Overall, this project serves as a comprehensive exploration of deep learning and transfer learning methods in the context of mushroom species recognition, showcasing their potential in image analysis and classification tasks.

CHAPTER 2

IDEATION & PROPOSED SOLUTION

2.1 Problem Statement Definition :

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Customer	I want to buy mushrooms	I am getting scare	I am concerned about the potential risks of consuming wild mushrooms and want to make sure that the mushrooms I purchase are free from harmful toxins.	Unsatisfied or uncertain about purchased mushroom nutrition.
PS-2	Shopkeeper	I want to sell mushrooms	I concern about customer health	Mushrooms are a delicate and perishable product that require careful handling and storage.	Stressed about ensuring the quality and availability of my mushroom inventory and concerned about meeting the demands of my customers.
PS-3	Restaurant owner	Making dishes	I am struggling to get fresh, good high-quality mushrooms	some of them are very harmful	Worry about Customer health .



Problem Statement 01

I am	I'm trying to	But	Because	Which makes me feel
Customer	buy	afraid of	Concerned about toxins in mushrooms, want safe purchase.	Satisfied and confident with purchased mushrooms' nutrition.

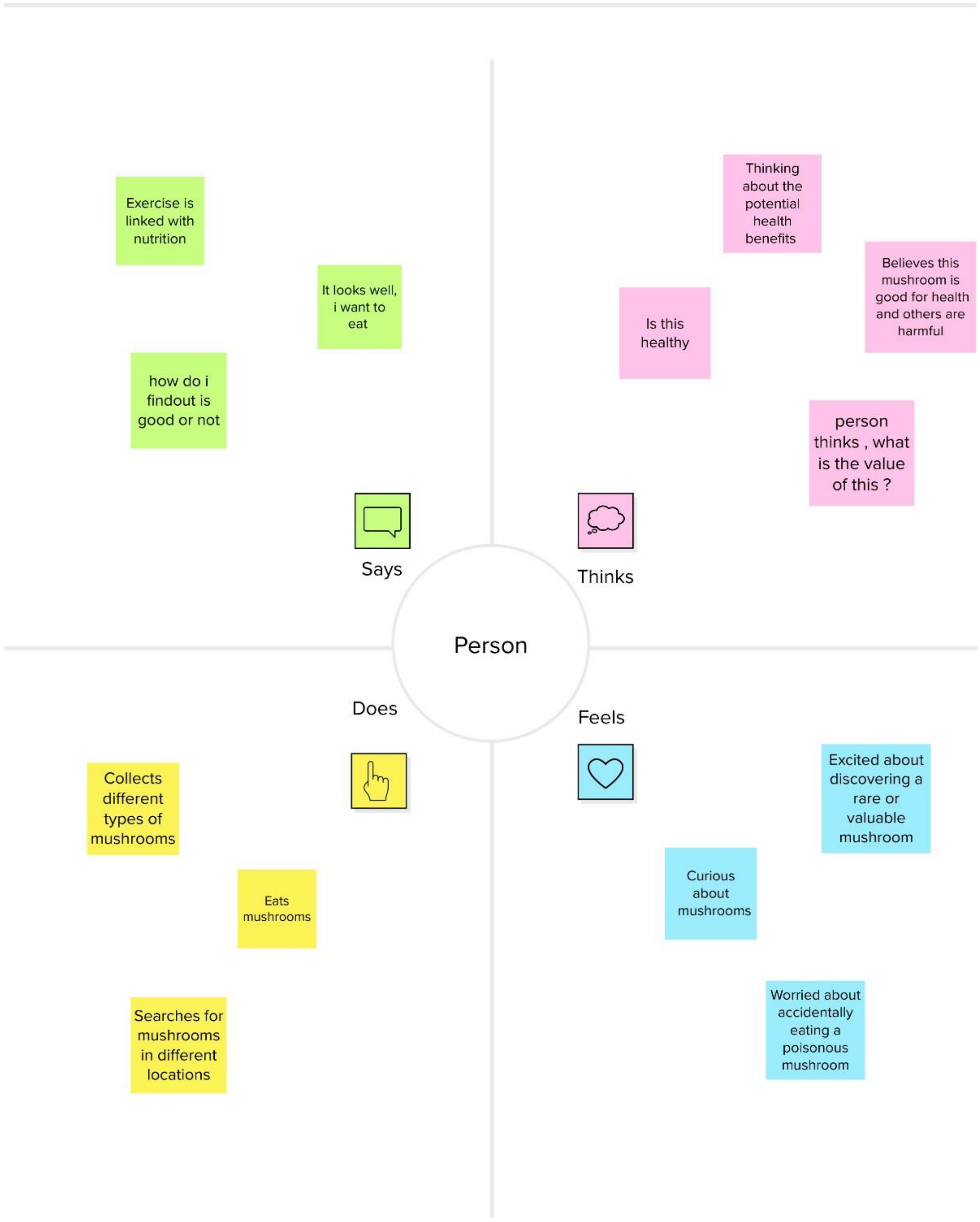
Problem Statement 02

I am	I'm trying to	But	Because	Which makes me feel
Shopkeeper	sell	afraid of customer health	Mushrooms are delicate and require careful handling.	Unsatisfied or uncertain about purchased mushroom nutrition.

Problem Statement 02

I am	I'm trying to	But	Because	Which makes me feel
Restaurant owner	making dishes	getting good mushrooms	some of them are very harmful	worry about customer health

2.2 Empathy Map Canvas :


Exercise is linked with nutrition
It looks well, i want to eat
how do i findout is good or not
Says
Thinking about the potential health benefits
Believes this mushroom is good for health and others are harmful
Is this healthy
person thinks , what is the value of this ?
Thinks
Person
Does
Feels
Collects different types of mushrooms
Eats mushrooms
Searches for mushrooms in different locations
Excited about discovering a rare or valuable mushroom
Curious about mushrooms
Worried about accidentally eating a poisonous mushroom

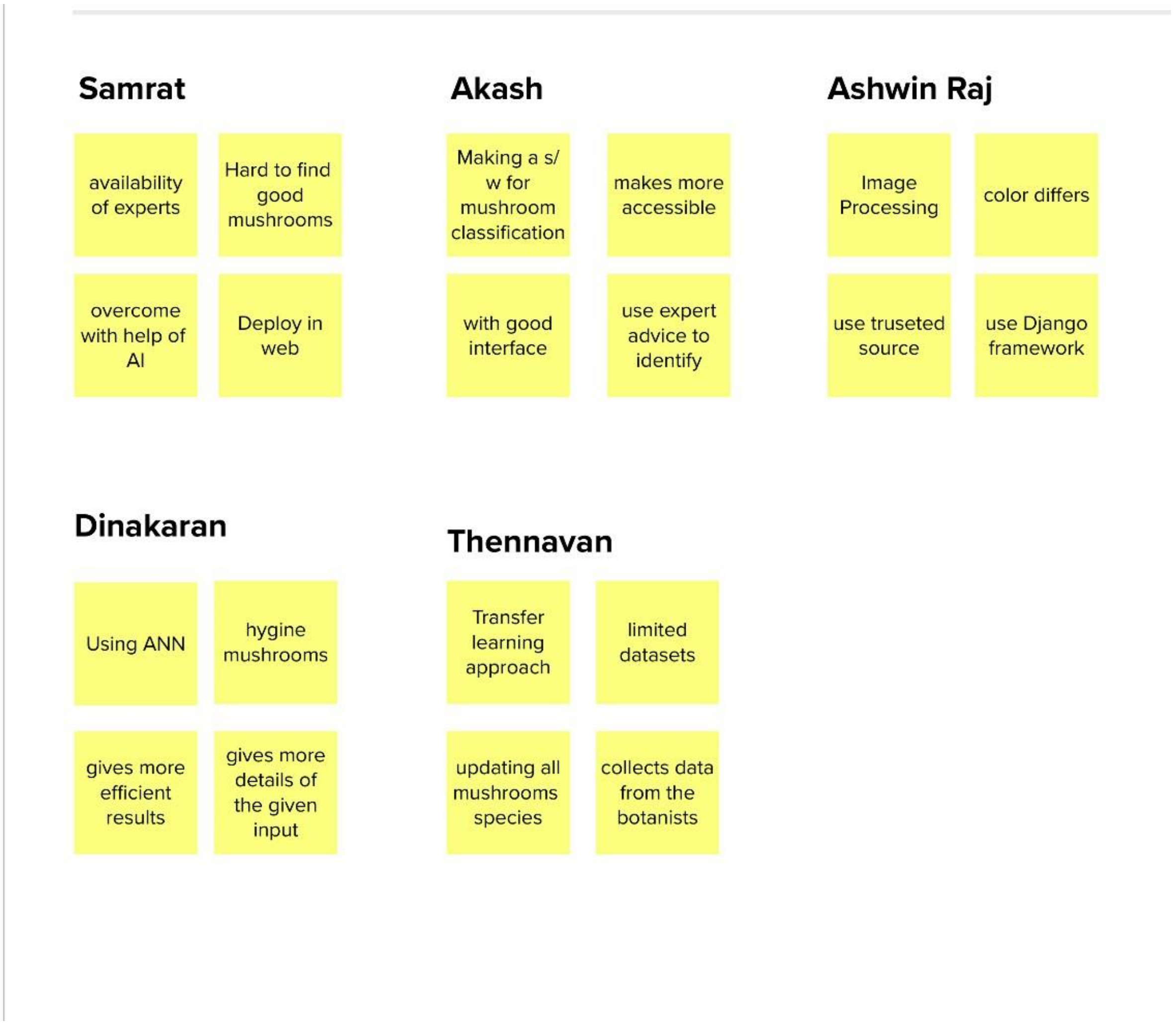
2.3 IDEATION & BRAINSTORMING :

i. Problem Statement :

Mushrooms are a valuable resource, but classification and identification are challenging due to the complexity of their physical features. A deep learning classification analysis using transfer learning techniques can provide a more efficient and accurate way to identify and classify mushrooms, unlocking their potential for human health and environmental sustainability.

Problem : How we can classify different types of mushrooms using deep learning methods and image analysis, in order to identify their species and make them more accessible and identifiable to people ?

ii. Brainstorm, Idea listening and Grouping



Samrat

availability of experts

Hard to find good mushrooms

overcome with help of AI

Deploy in web

Akash

Making a s/w for mushroom classification

makes more accessible

with good interface

use expert advice to identify

Ashwin Raj

Image Processing

color differs

use truseted source

use Django framework

Dinakaran

Using ANN

hygine mushrooms

gives more efficient results

gives more details of the given input

Thennavan

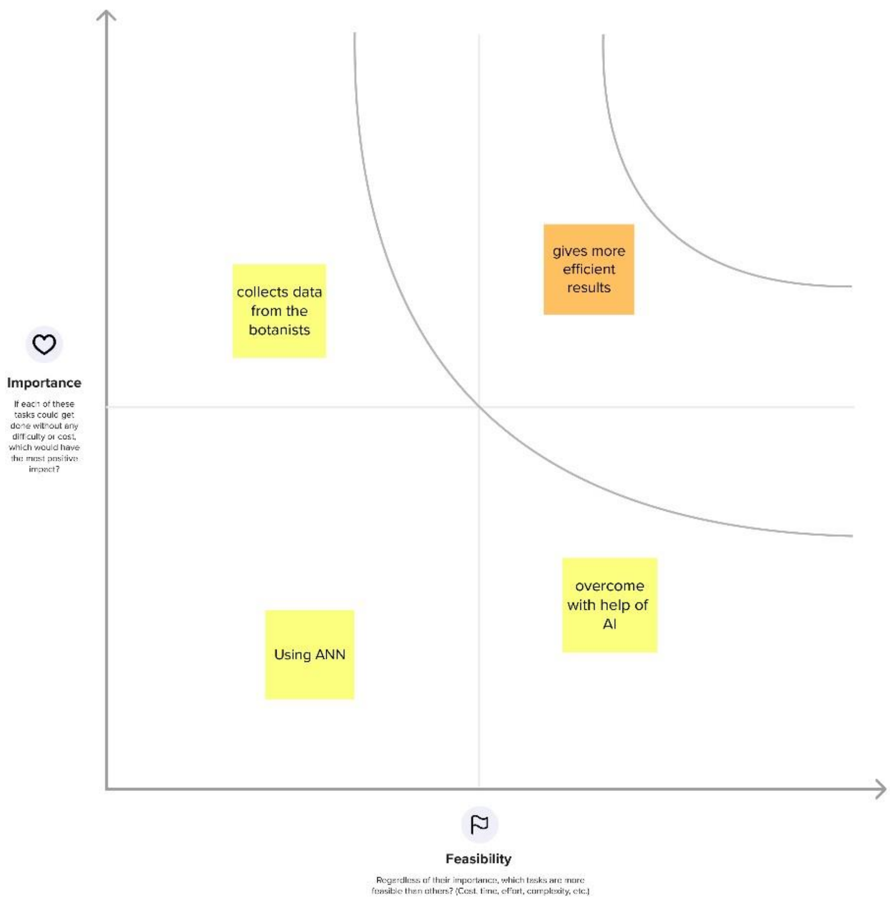
Transfer learning approach

limited datasets

updating all mushrooms species

collects data from the botanists

iii. Idea Prioritization


collects data from the botanists
gives more efficient results
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
overcome with help of AI
Using ANN
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

2.4 Proposed Solution :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The mushroom kingdom is home to a diverse range of mushrooms, but many of these remain unidentified or understand. Therefore, there is a need to classify the mushrooms based on their features and characteristics for better understanding and utilization.
2.	Idea / Solution description	Our proposed solution is to develop a classification analysis system for mushrooms based on their physical features, habitat, and other characteristics. We will use machine learning algorithms to identify and classify the mushrooms accurately. The system will allow users to input images or descriptions of mushrooms, and the system will provide a classification of the mushrooms along with detailed information about species of that mushroom.
3.	Novelty / Uniqueness	While there are existing mushroom classification systems, they rely heavily on human experts' knowledge and experience, which can be time-consuming and expensive. Our proposed system uses machine learning algorithms to automate the classification process, reducing the need for human intervention, and making it more efficient and cost-effective.
4.	Social Impact / Customer Satisfaction	Our classification analysis system will benefit a broad range of users, including mushroom enthusiasts, conservationists, and scientists.
5.	Business Model (Revenue Model)	Our model is open source and it can accessible for everyone. Ultimate this is for the educational purpose
6.	Scalability of the Solution	Our classification analysis system is highly scalable, and we can expand it to include more features and characteristics as we gather more data.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Functional Requirement :

FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	User Registration	Allow users to register an account.
FR-2	User Confirmation	Send confirmation email to verify user's registration.
FR-3	Mushroom Classification	• Implement deep learning algorithms for mushroom analysis. • Use transfer learning techniques(inception V3, resnet50V2, Xception) • Train the models to classify mushrooms into Boletus, Lactarius, and Russula categories
FR-4	Image Processing	Develop image processing module to extract features from mushroom images.
FR-5	Prediction and Classification	Perform prediction and classification of mushrooms based on the trained models
FR-6	User Interface	Design a user friendly interface for users to interact with the system

3.2 Non-Functional Requirement :

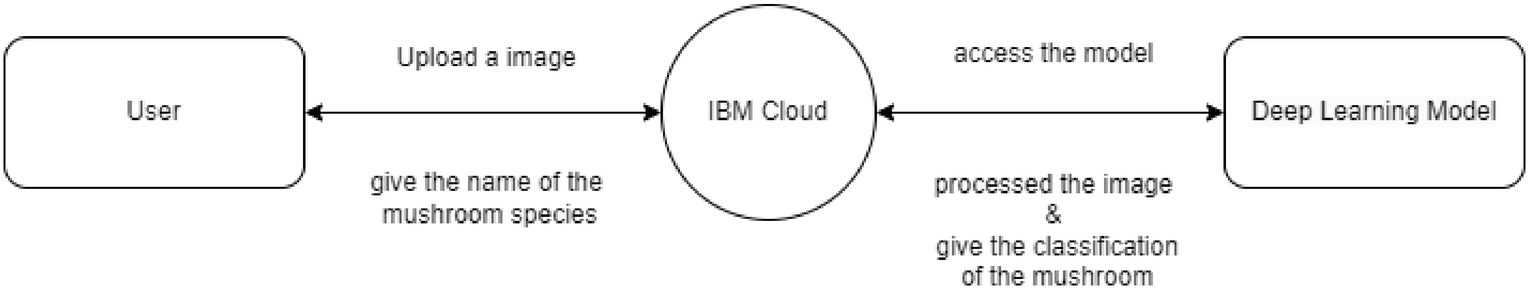
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system should be intuitive and easy to use for both technical and non-technical users.
NFR-2	**Security**	Implement security measures to protect user data and prevent unauthorized access.
NFR-3	**Reliability**	Ensure the system is stable and reliable, with minimal downtime or errors.
NFR-4	**Performance**	The system should have fast processing and response times for image classification.
NFR-5	**Availability**	The system should be available for users to access and use at all times.
NFR-6	**Scalability**	The system should be able to handle a growing number of users and mushroom images efficiently.

CHAPTER 4

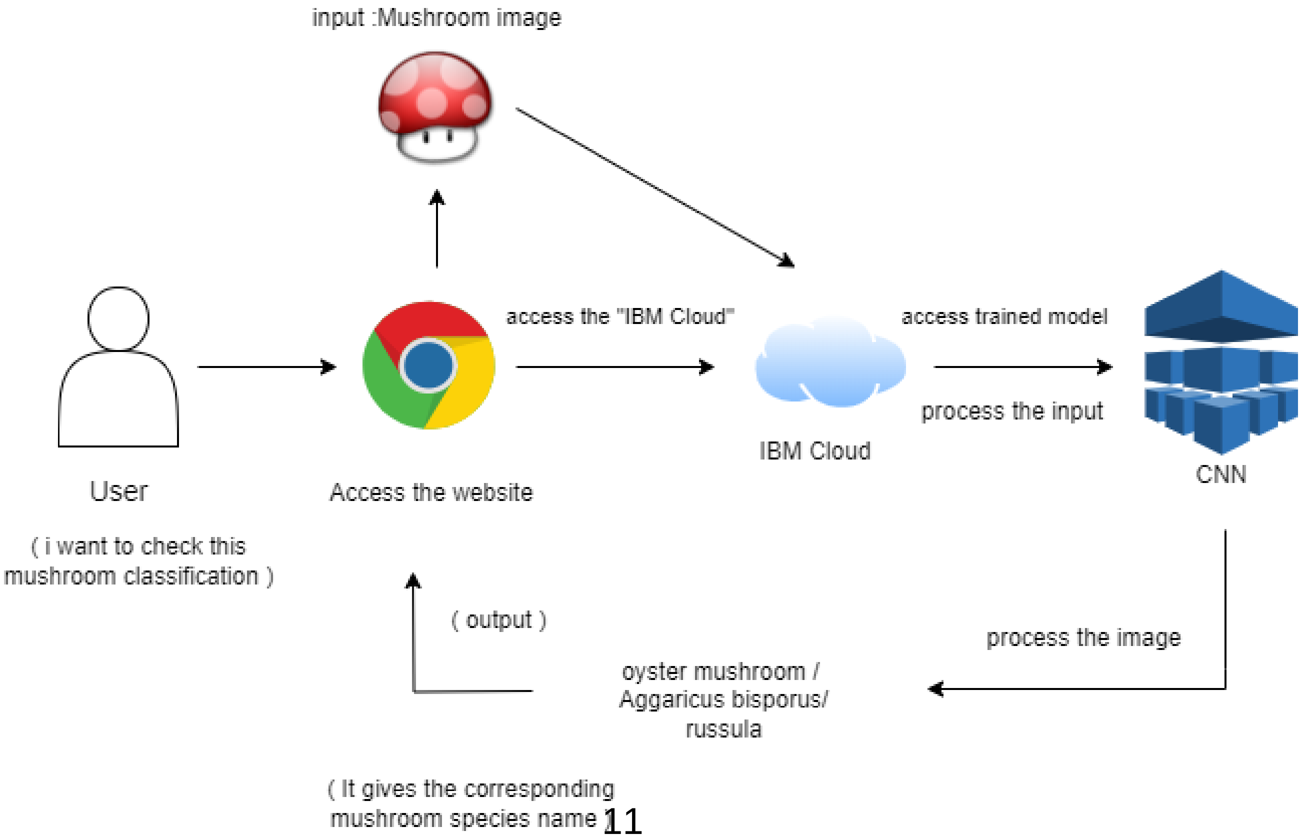
PROJECT DESIGN

4.1 Data Flow Diagrams :

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.


User
Upload a image
give the name of the mushroom species
IBM Cloud
access the model
processed the image & give the classification of the mushroom
Deep Learning Model

4.2 Solution & Technical Architecture :


input :Mushroom image
User
(i want to check this mushroom classification)
Access the website
access the "IBM Cloud"
IBM Cloud
access trained model
process the input
CNN
process the image
oyster mushroom / Aggaricus bisporus/ russula
(output)
(It gives the corresponding mushroom species name)

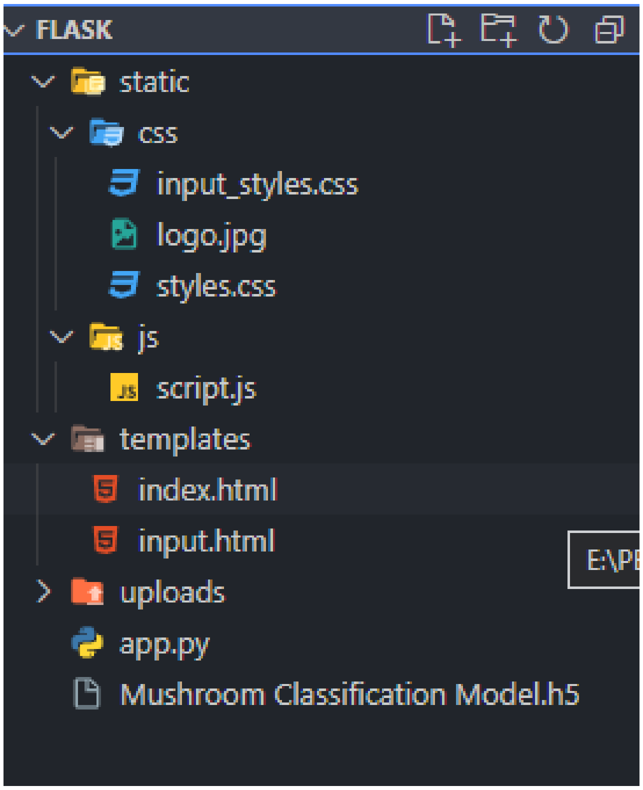
4.3 User Stories :

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Samrat
	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Samrat
	Registration	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard	Low	Akash
	Registration	USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard	Medium	Akash
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register for the application.	High	Aswin Raj
	Dashboard	USN-6	As a user, I can view mushroom classification	I can view my mushrooms classification results	Medium	Aswin Raj
Customer (Web user)	Registration	USN-7	As a user, I can register for the application in website	I can access the webssite	High	Dinakaran
	Login	USN-8	I can login the website	I can access the application	High	Dinakaran
	Mushroom Classification	USN-9	I can upload the mushroom image	I can know info about the mushroom images	High	Thennavan
	Dashboard	USN-10	As a user, I can view mushroom classification	I can view my mushrooms classification results	Medium	Thennavan

CHAPTER 5

CODING & SOLUTIONING

PROJECT STRUCTURE :



```
FLASK
  static
    css
      input_styles.css
      logo.jpg
      styles.css
    js
      script.js
  templates
    index.html
    input.html
  uploads
  app.py
  Mushroom Classification Model.h5
```

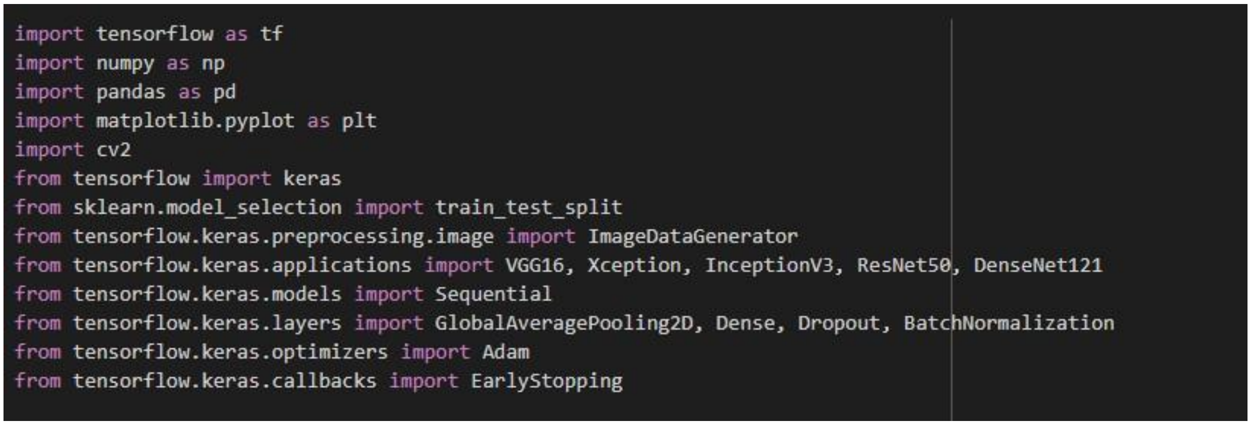
Uncovering the Hidden Treasures of the Mushroom Kingdom: A Classification Analysis Project structure, it have the static(css,js), templates(HTML),uploads,app.py file and transfer learning model.

5.1 Feature 1 :

Uncovering the Hidden Treasures of the Mushroom Kingdom: A Classification Analysis Project has function that user uploads any mushroom image it will give the classification of the mushroom, with their family name like “Boletus, Lactarius & Russula.”

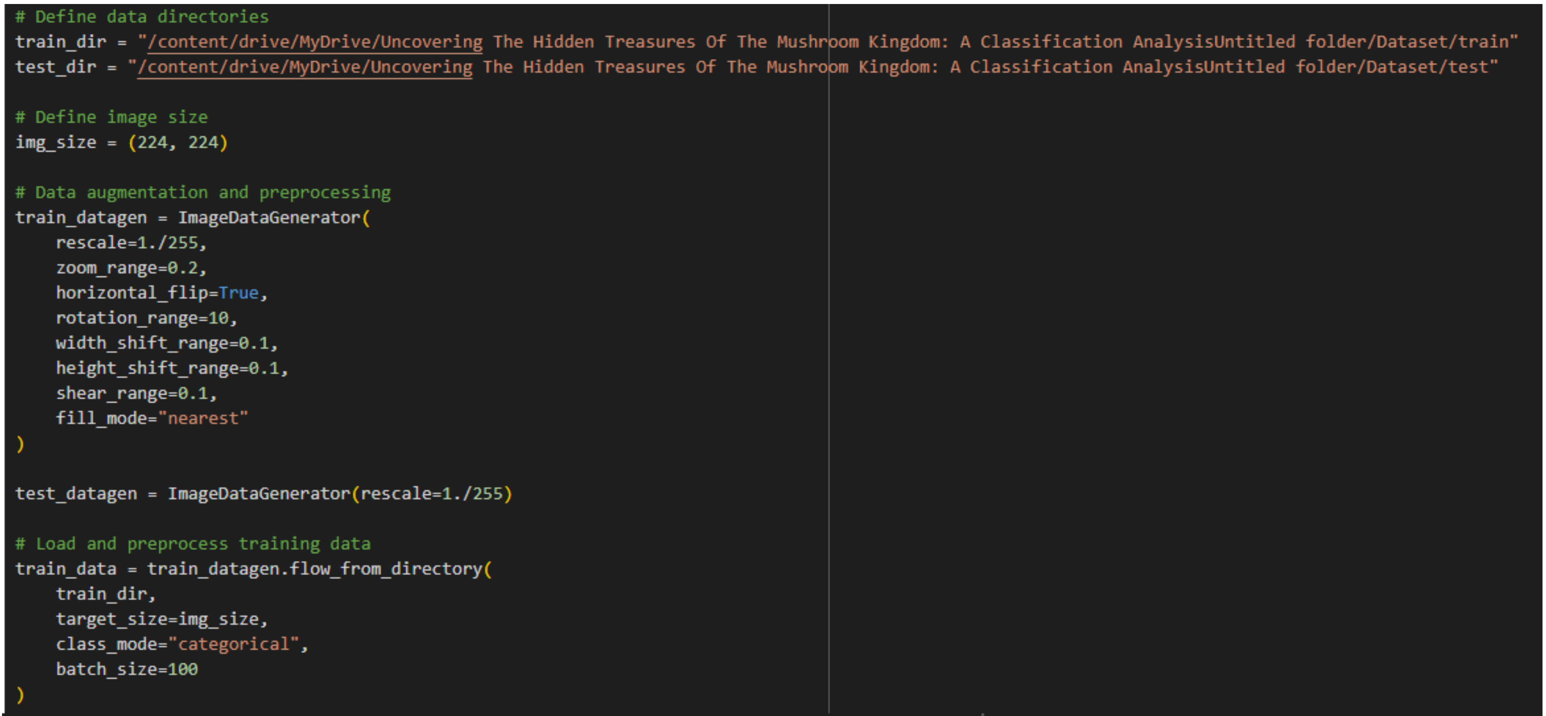
Coding :

Importing required libraries

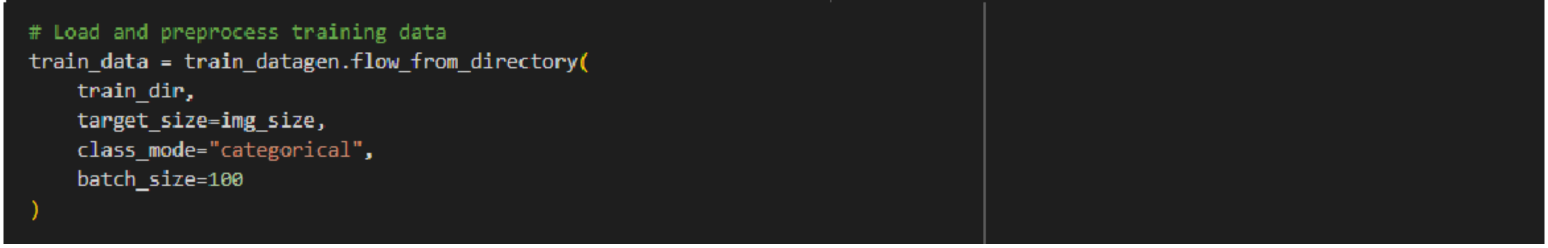


```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
from tensorflow import keras
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16, Xception, InceptionV3, ResNet50, DenseNet121
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
```

Define dataset directory and data augmentation

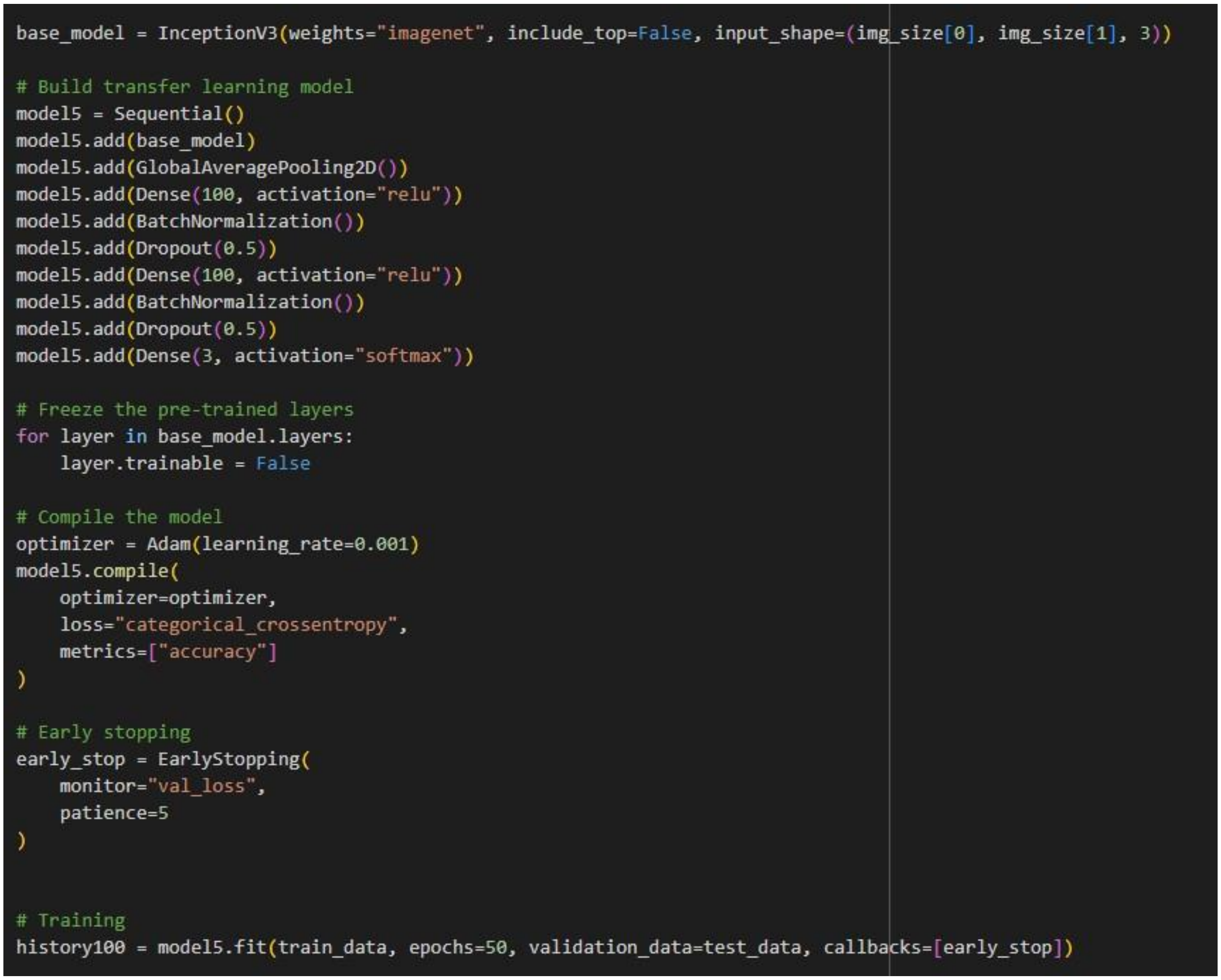


```
# Define data directories
train_dir = "/content/drive/MyDrive/Uncovering The Hidden Treasures Of The Mushroom Kingdom: A Classification AnalysisUntitled folder/Dataset/train"
test_dir = "/content/drive/MyDrive/Uncovering The Hidden Treasures Of The Mushroom Kingdom: A Classification AnalysisUntitled folder/Dataset/test"

# Define image size
img_size = (224, 224)

# Data augmentation and preprocessing
train_datagen = ImageDataGenerator(
    rescale=1./255,
    zoom_range=0.2,
    horizontal_flip=True,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    fill_mode="nearest"
)

test_datagen = ImageDataGenerator(rescale=1./255)

# Load and preprocess training data
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    class_mode="categorical",
    batch_size=100
)
```



```
# Load and preprocess training data
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    class_mode="categorical",
    batch_size=100
)
```

Model building with InceptionV3 pretrained model, adding layer & Training the model :



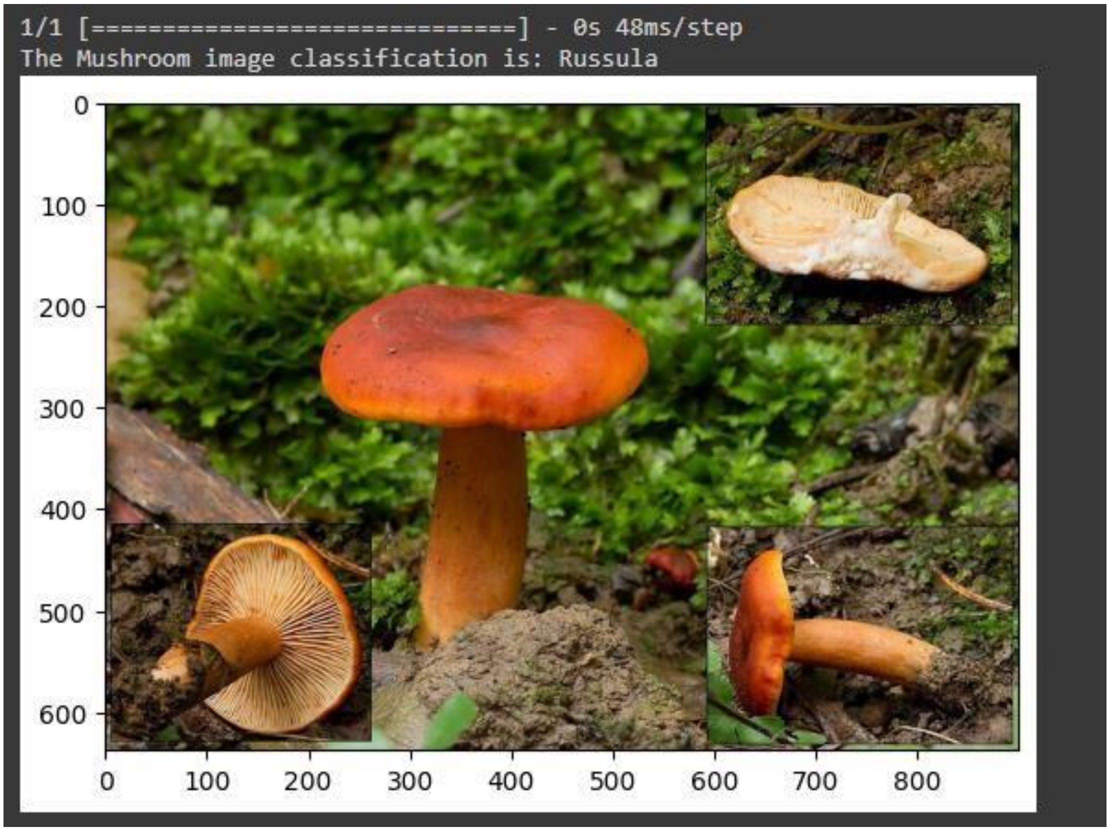
```
base_model = InceptionV3(weights="imagenet", include_top=False, input_shape=(img_size[0], img_size[1], 3))

# Build transfer learning model
model5 = Sequential()
model5.add(base_model)
model5.add(GlobalAveragePooling2D())
model5.add(Dense(100, activation="relu"))
model5.add(BatchNormalization())
model5.add(Dropout(0.5))
model5.add(Dense(100, activation="relu"))
model5.add(BatchNormalization())
model5.add(Dropout(0.5))
model5.add(Dense(3, activation="softmax"))

# Freeze the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
optimizer = Adam(learning_rate=0.001)
model5.compile(
    optimizer=optimizer,
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

# Early stopping
early_stop = EarlyStopping(
    monitor="val_loss",
    patience=5
)


# Training
history100 = model5.fit(train_data, epochs=50, validation_data=test_data, callbacks=[early_stop])
```

This model is build with pretrained model “InceptionV3” and adding the layers like 3 dense layers , GlobalAveragePooling2D. A Dense layer with 100 units and ReLU activation function is added. BatchNormalization layer is added to normalize the activations of the previous layer.

Dropout layer is added to prevent overfitting by randomly dropping 50% of the connections.

A Dense layer with 3 units and softmax activation function is added as the final output layer.

Predicting the input image :



```
1/1 [==============================] - 0s 48ms/step
The Mushroom image classification is: Russula
```

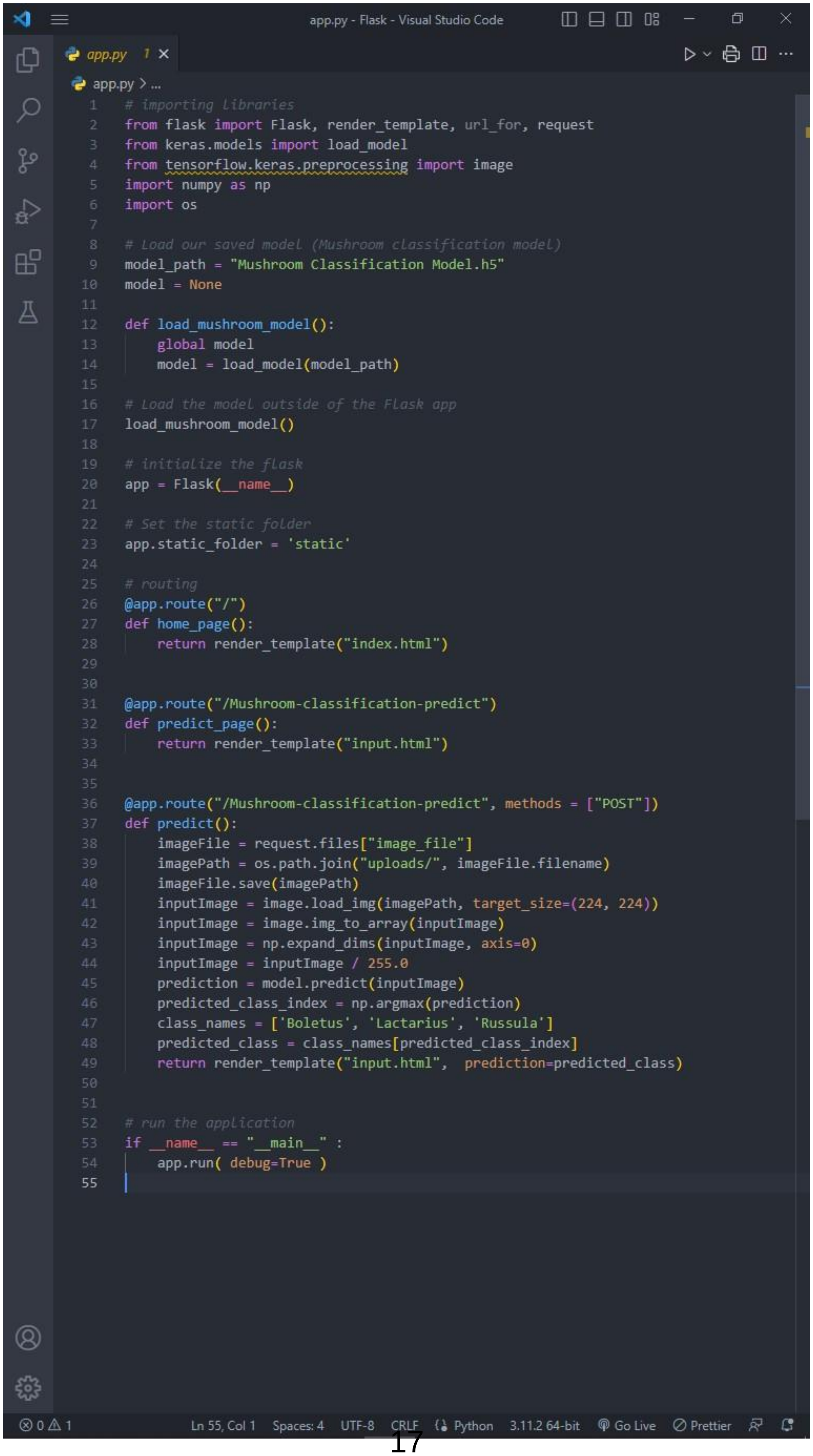
If user upload image, the model will classify the images then process the image and give the mushroom classification (Output).

5.2 Feature 2 :

In Existing projects it doesn’t show the upload file, but our project overcome it .We integrated Machine learning to flask. Flask is a webFramework development.

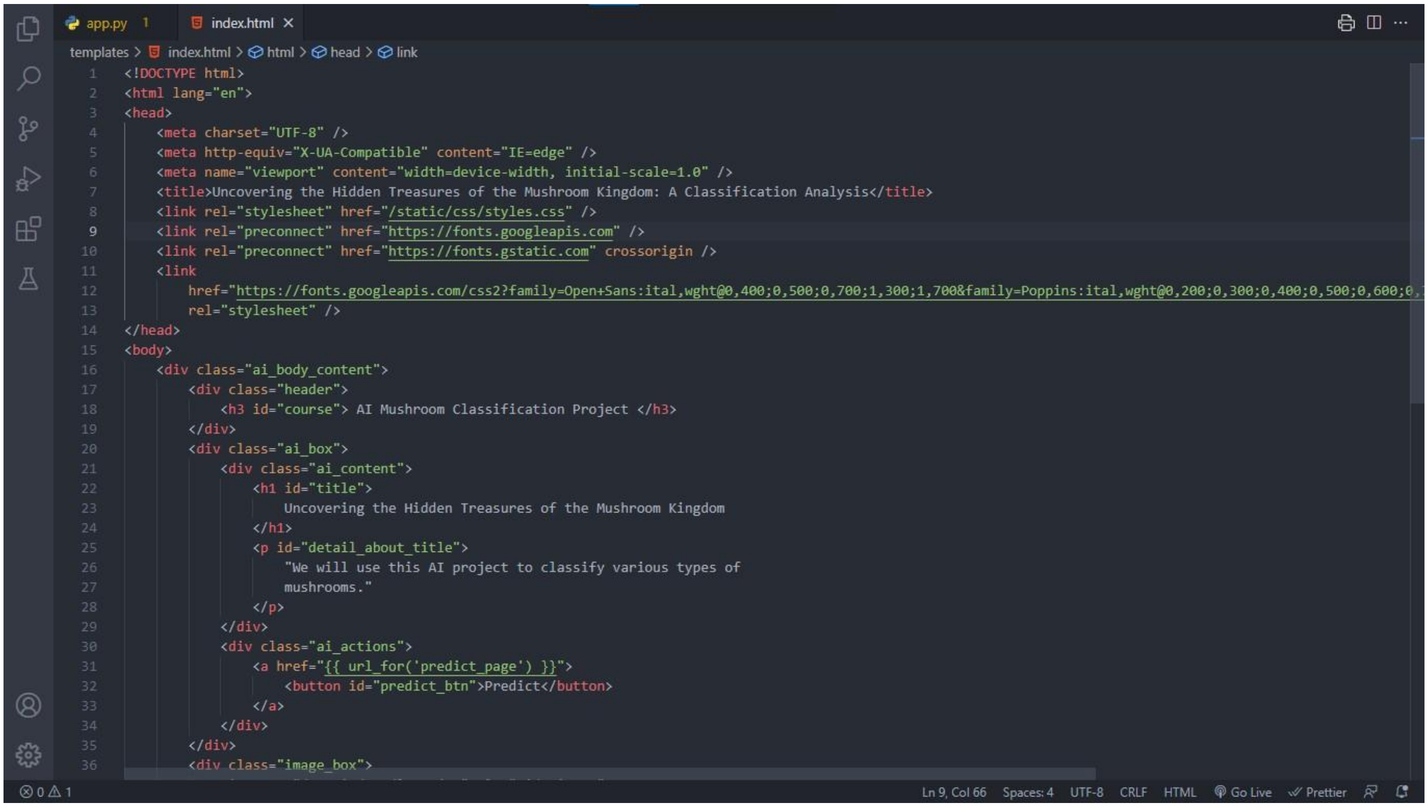
Uncovering the Hidden Treasures of the Mushroom Kingdom: A Classification Analysis Project structure, it have the static(css,js), templates(HTML),uploads,app.py file and transfer learning model.

Flask App :



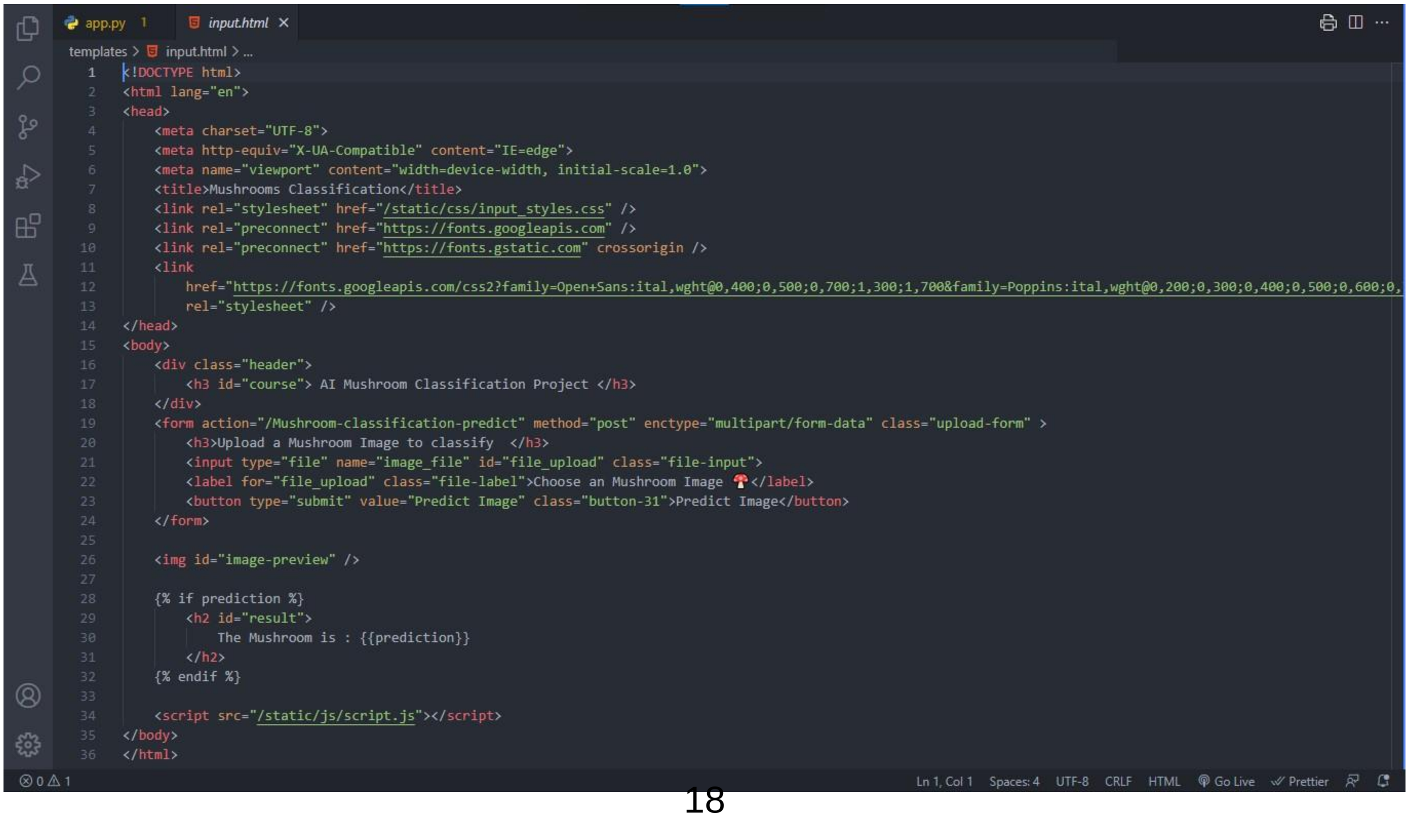
```python
# importing libraries
from flask import Flask, render_template, url_for, request
from keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import os

# Load our saved model (Mushroom classification model)
model_path = "Mushroom Classification Model.h5"
model = None

def load_mushroom_model():
    global model
    model = load_model(model_path)

# Load the model outside of the Flask app
load_mushroom_model()

# initialize the flask
app = Flask(__name__)

# Set the static folder
app.static_folder = 'static'

# routing
@app.route("/")
def home_page():
    return render_template("index.html")


@app.route("/Mushroom-classification-predict")
def predict_page():
    return render_template("input.html")


@app.route("/Mushroom-classification-predict", methods = ["POST"])
def predict():
    imageFile = request.files["image_file"]
    imagePath = os.path.join("uploads/", imageFile.filename)
    imageFile.save(imagePath)
    inputImage = image.load_img(imagePath, target_size=(224, 224))
    inputImage = image.img_to_array(inputImage)
    inputImage = np.expand_dims(inputImage, axis=0)
    inputImage = inputImage / 255.0
    prediction = model.predict(inputImage)
    predicted_class_index = np.argmax(prediction)
    class_names = ['Boletus', 'Lactarius', 'Russula']
    predicted_class = class_names[predicted_class_index]
    return render_template("input.html",  prediction=predicted_class)


# run the application
if __name__ == "__main__" :
    app.run( debug=True )

```

index.html :


app.py 1
index.html
templates > index.html > html > head > link
Ln 9, Col 66 Spaces: 4 UTF-8 CRLF HTML Go Live Prettier
⊗ 0 ⚠ 1

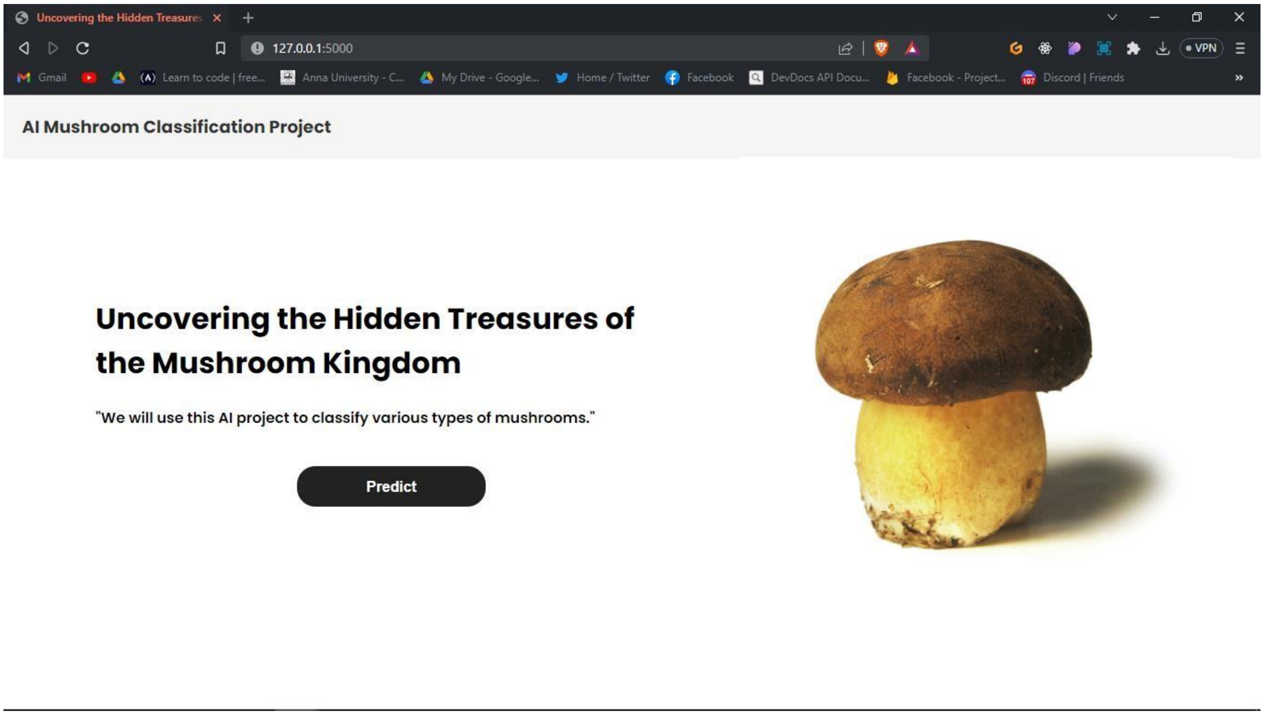
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Uncovering the Hidden Treasures of the Mushroom Kingdom: A Classification Analysis</title>
    <link rel="stylesheet" href="/static/css/styles.css" />
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
        href="https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,400;0,500;0,700;1,300;1,700&family=Poppins:ital,wght@0,200;0,300;0,400;0,500;0,600;0,
        rel="stylesheet" />
</head>
<body>
    <div class="ai_body_content">
        <div class="header">
            <h3 id="course"> AI Mushroom Classification Project </h3>
        </div>
        <div class="ai_box">
            <div class="ai_content">
                <h1 id="title">
                    Uncovering the Hidden Treasures of the Mushroom Kingdom
                </h1>
                <p id="detail_about_title">
                    "We will use this AI project to classify various types of
                    mushrooms."
                </p>
            </div>
            <div class="ai_actions">
                <a href="{{ url_for('predict_page') }}">
                    <button id="predict_btn">Predict</button>
                </a>
            </div>
        </div>
        <div class="image_box">
```

input.html :


app.py 1
input.html
templates > input.html > ...
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live Prettier
⊗ 0 ⚠ 1

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mushrooms Classification</title>
    <link rel="stylesheet" href="/static/css/input_styles.css" />
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
        href="https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,400;0,500;0,700;1,300;1,700&family=Poppins:ital,wght@0,200;0,300;0,400;0,500;0,600;0,
        rel="stylesheet" />
</head>
<body>
    <div class="header">
        <h3 id="course"> AI Mushroom Classification Project </h3>
    </div>
    <form action="/Mushroom-classification-predict" method="post" enctype="multipart/form-data" class="upload-form" >
        <h3>Upload a Mushroom Image to classify  </h3>
        <input type="file" name="image_file" id="file_upload" class="file-input">
        <label for="file_upload" class="file-label">Choose an Mushroom Image 🍄</label>
        <button type="submit" value="Predict Image" class="button-31">Predict Image</button>
    </form>

    <img id="image-preview" />

    {% if prediction %}
        <h2 id="result">
            The Mushroom is : {{prediction}}
        </h2>
    {% endif %}

    <script src="/static/js/script.js"></script>
</body>
</html>
```

TESTING :



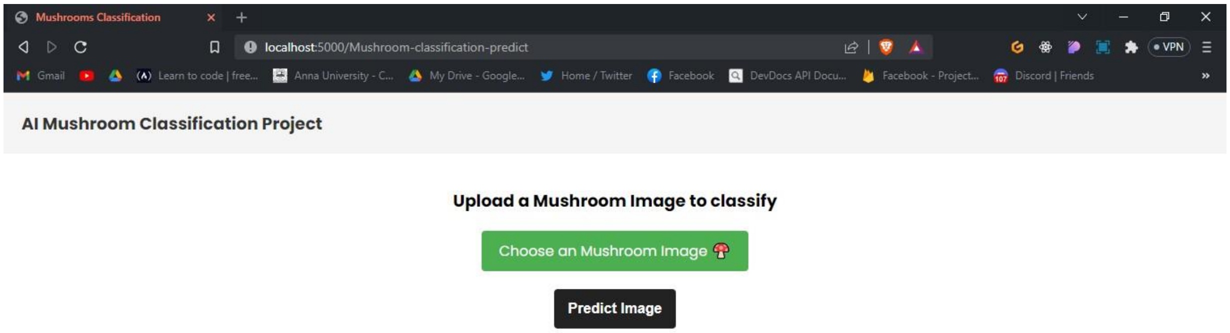
AI Mushroom Classification Project

Uncovering the Hidden Treasures of the Mushroom Kingdom

"We will use this AI project to classify various types of mushrooms."

The interface of the Uncovering the Hidden Treasures of the Mushroom Kingdom: A Classification Analysis. It have button to predict page to classifiying the mushroom image.

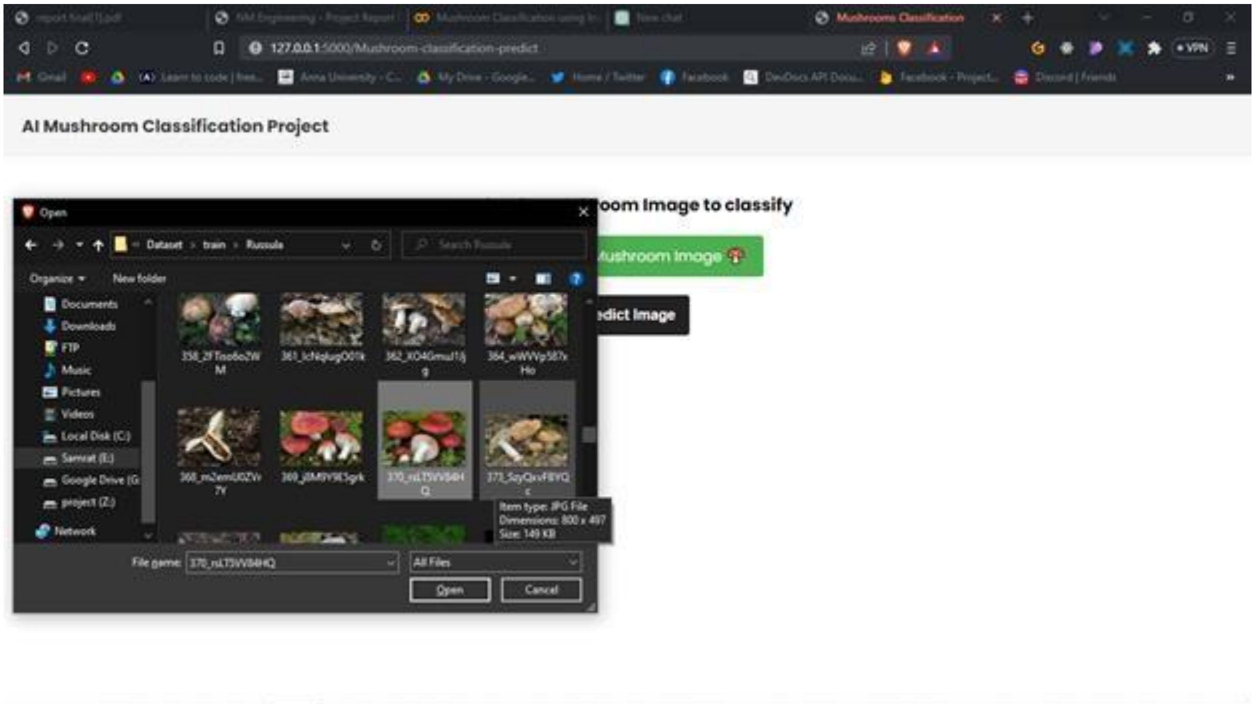
Predict Image :



AI Mushroom Classification Project

Upload a Mushroom Image to classify

SELECT IMAGE :



127.0.0.1:5000/Mushroom-classification-predict

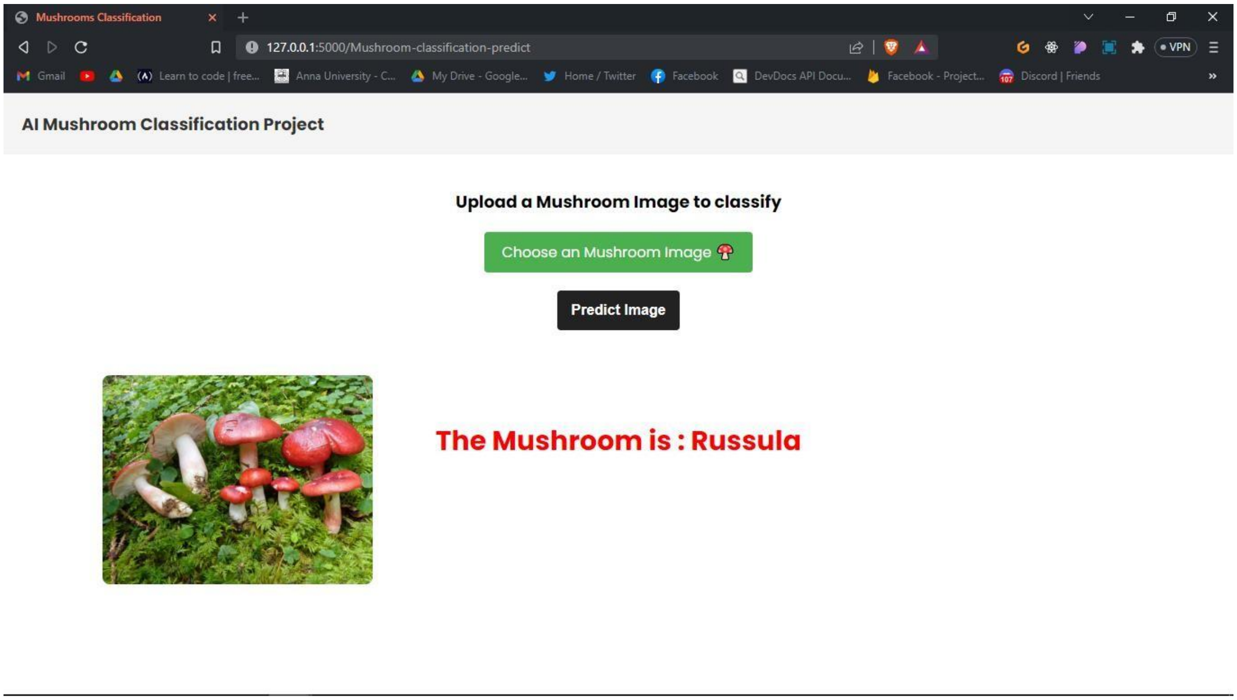
AI Mushroom Classification Project

oom Image to classify

File name: 370_rsLT5VVB4HQ

Item type: JPG File
Dimensions: 800 x 497
Size: 149 KB

OUTPUT PAGE :



127.0.0.1:5000/Mushroom-classification-predict

AI Mushroom Classification Project

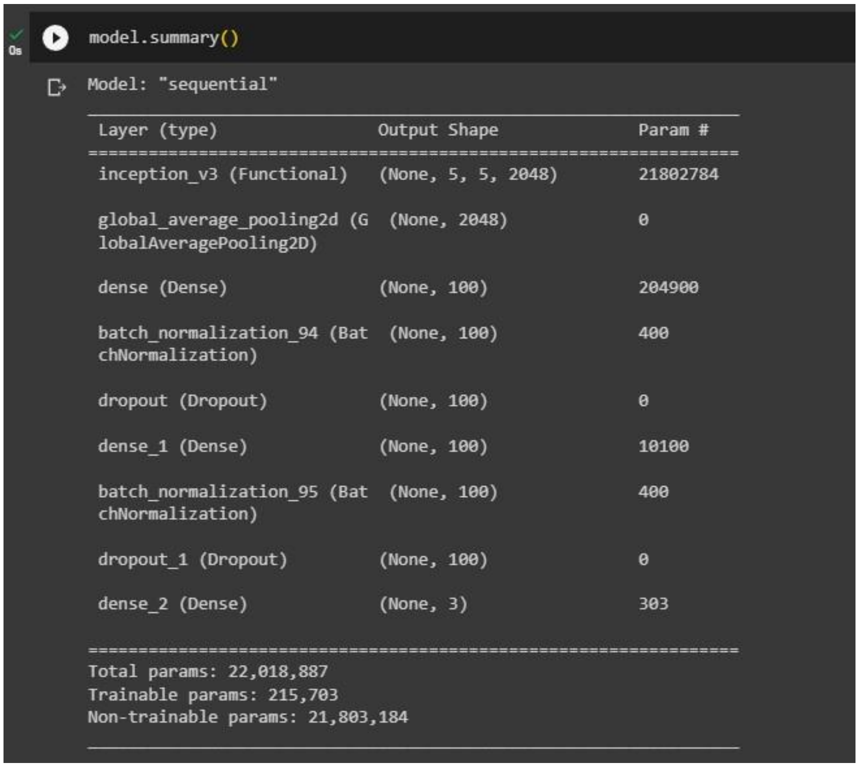
Upload a Mushroom Image to classify

The Mushroom is : Russula

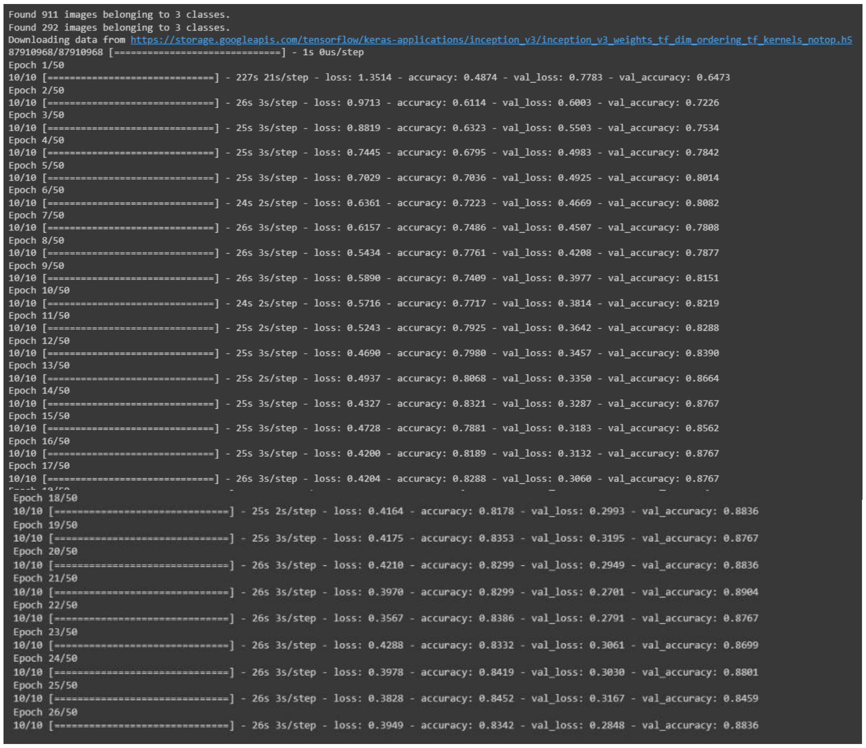
CHAPTER 6

RESULTS

6.1 Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Model Summary	-	

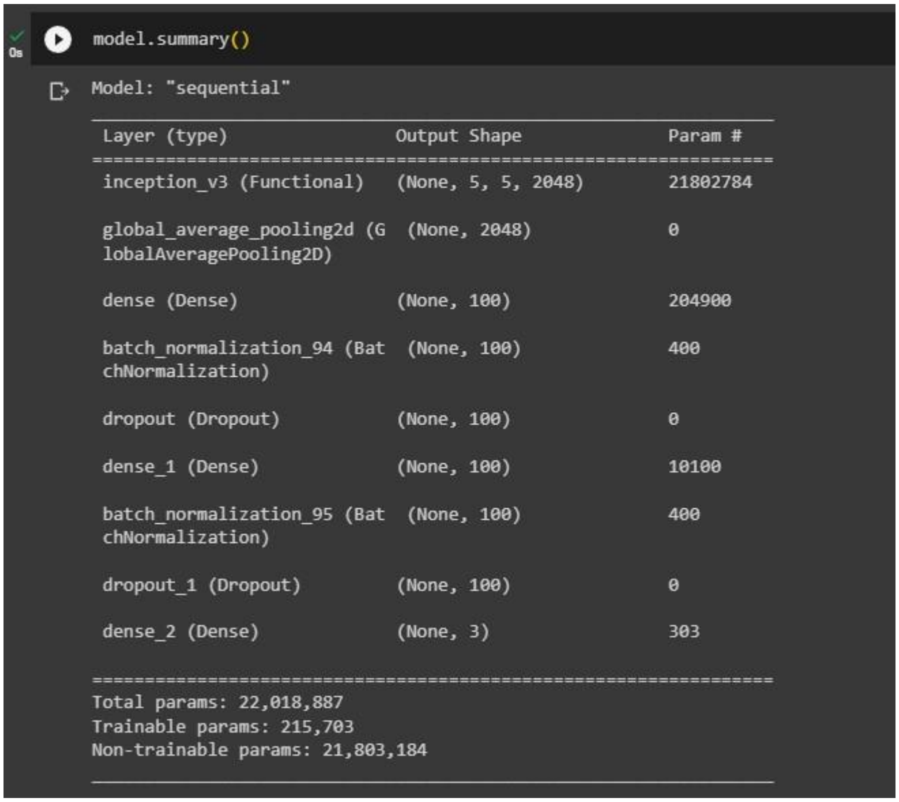
```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 100)               204900

 batch_normalization_94 (Bat  (None, 100)              400
 chNormalization)

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 100)               10100

 batch_normalization_95 (Bat  (None, 100)              400
 chNormalization)

 dropout_1 (Dropout)         (None, 100)               0

 dense_2 (Dense)             (None, 3)                 303

=================================================================
Total params: 22,018,887
Trainable params: 215,703
Non-trainable params: 21,803,184
_________________________________________________________________
```

2.	Accuracy	**Training Accuracy** : 83.42% **Validation Accuracy** **:** 88.36%	

```
Found 911 images belonging to 3 classes.
Found 292 images belonging to 3 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 1s 0us/step
Epoch 1/50
10/10 [==============================] - 227s 21s/step - loss: 1.3514 - accuracy: 0.4874 - val_loss: 0.7783 - val_accuracy: 0.6473
Epoch 2/50
10/10 [==============================] - 26s 3s/step - loss: 0.9713 - accuracy: 0.6114 - val_loss: 0.6003 - val_accuracy: 0.7226
Epoch 3/50
10/10 [==============================] - 25s 3s/step - loss: 0.8819 - accuracy: 0.6323 - val_loss: 0.5503 - val_accuracy: 0.7534
Epoch 4/50
10/10 [==============================] - 25s 3s/step - loss: 0.7445 - accuracy: 0.6795 - val_loss: 0.4983 - val_accuracy: 0.7842
Epoch 5/50
10/10 [==============================] - 25s 3s/step - loss: 0.7029 - accuracy: 0.7036 - val_loss: 0.4925 - val_accuracy: 0.8014
Epoch 6/50
10/10 [==============================] - 24s 2s/step - loss: 0.6361 - accuracy: 0.7223 - val_loss: 0.4669 - val_accuracy: 0.8082
Epoch 7/50
10/10 [==============================] - 26s 3s/step - loss: 0.6157 - accuracy: 0.7486 - val_loss: 0.4507 - val_accuracy: 0.7808
Epoch 8/50
10/10 [==============================] - 26s 3s/step - loss: 0.5434 - accuracy: 0.7761 - val_loss: 0.4208 - val_accuracy: 0.7877
Epoch 9/50
10/10 [==============================] - 26s 3s/step - loss: 0.5890 - accuracy: 0.7409 - val_loss: 0.3977 - val_accuracy: 0.8151
Epoch 10/50
10/10 [==============================] - 24s 2s/step - loss: 0.5716 - accuracy: 0.7717 - val_loss: 0.3814 - val_accuracy: 0.8219
Epoch 11/50
10/10 [==============================] - 25s 2s/step - loss: 0.5243 - accuracy: 0.7925 - val_loss: 0.3642 - val_accuracy: 0.8288
Epoch 12/50
10/10 [==============================] - 25s 3s/step - loss: 0.4690 - accuracy: 0.7980 - val_loss: 0.3457 - val_accuracy: 0.8390
Epoch 13/50
10/10 [==============================] - 25s 2s/step - loss: 0.4937 - accuracy: 0.8068 - val_loss: 0.3350 - val_accuracy: 0.8664
Epoch 14/50
10/10 [==============================] - 25s 3s/step - loss: 0.4327 - accuracy: 0.8321 - val_loss: 0.3287 - val_accuracy: 0.8767
Epoch 15/50
10/10 [==============================] - 25s 3s/step - loss: 0.4728 - accuracy: 0.7881 - val_loss: 0.3183 - val_accuracy: 0.8562
Epoch 16/50
10/10 [==============================] - 25s 3s/step - loss: 0.4200 - accuracy: 0.8189 - val_loss: 0.3132 - val_accuracy: 0.8767
Epoch 17/50
10/10 [==============================] - 26s 3s/step - loss: 0.4204 - accuracy: 0.8288 - val_loss: 0.3060 - val_accuracy: 0.8767
Epoch 18/50
10/10 [==============================] - 25s 2s/step - loss: 0.4164 - accuracy: 0.8178 - val_loss: 0.2993 - val_accuracy: 0.8836
Epoch 19/50
10/10 [==============================] - 25s 3s/step - loss: 0.4175 - accuracy: 0.8353 - val_loss: 0.3195 - val_accuracy: 0.8767
Epoch 20/50
10/10 [==============================] - 26s 3s/step - loss: 0.4210 - accuracy: 0.8299 - val_loss: 0.2949 - val_accuracy: 0.8836
Epoch 21/50
10/10 [==============================] - 26s 3s/step - loss: 0.3970 - accuracy: 0.8299 - val_loss: 0.2701 - val_accuracy: 0.8904
Epoch 22/50
10/10 [==============================] - 26s 3s/step - loss: 0.3567 - accuracy: 0.8386 - val_loss: 0.2791 - val_accuracy: 0.8767
Epoch 23/50
10/10 [==============================] - 26s 3s/step - loss: 0.4288 - accuracy: 0.8332 - val_loss: 0.3061 - val_accuracy: 0.8699
Epoch 24/50
10/10 [==============================] - 26s 3s/step - loss: 0.3978 - accuracy: 0.8419 - val_loss: 0.3030 - val_accuracy: 0.8801
Epoch 25/50
10/10 [==============================] - 26s 3s/step - loss: 0.3828 - accuracy: 0.8452 - val_loss: 0.3167 - val_accuracy: 0.8459
Epoch 26/50
10/10 [==============================] - 26s 3s/step - loss: 0.3949 - accuracy: 0.8342 - val_loss: 0.2848 - val_accuracy: 0.8836
```

PERFORMANCE SCREENSHOTS :

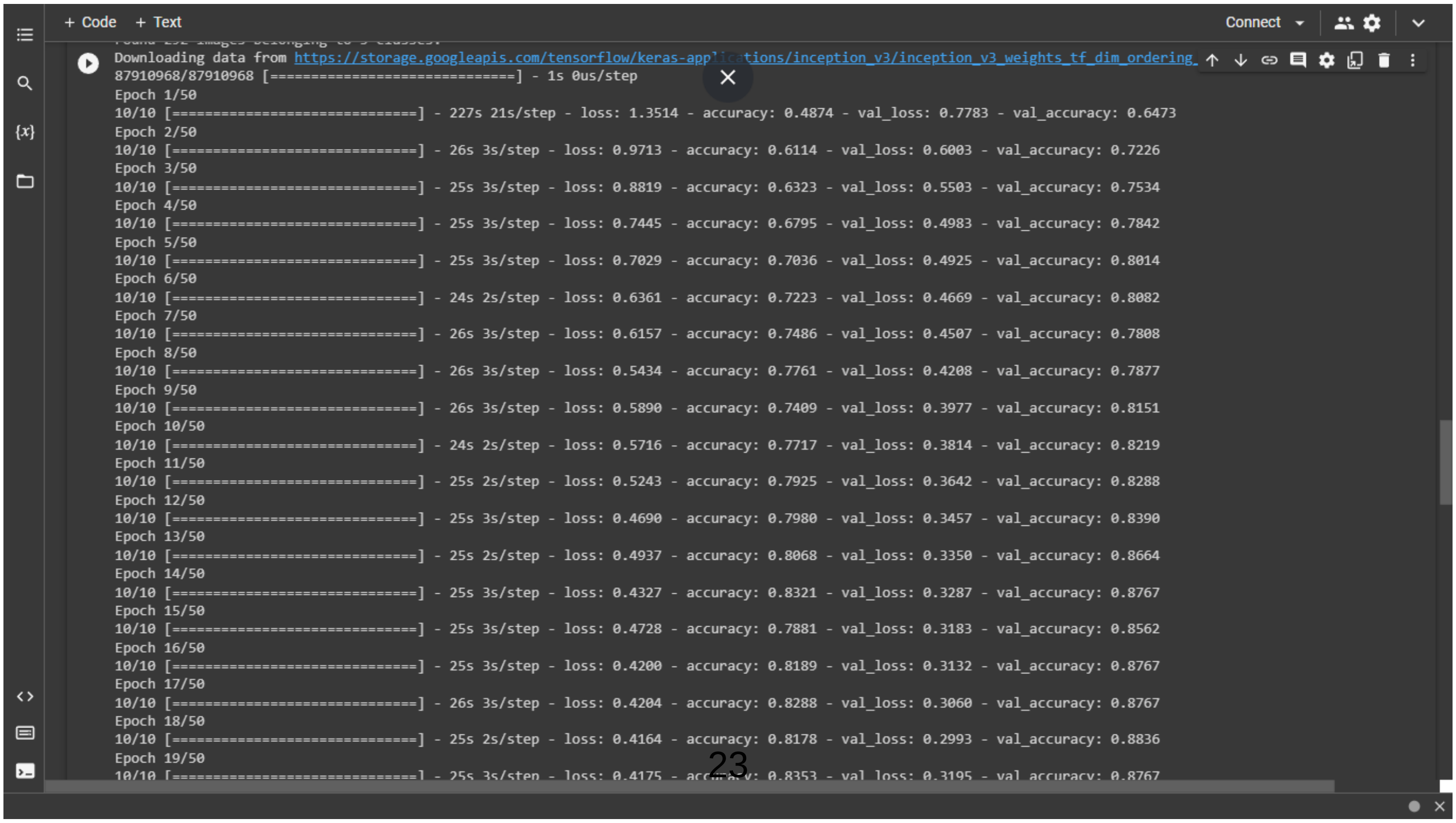
Summary :



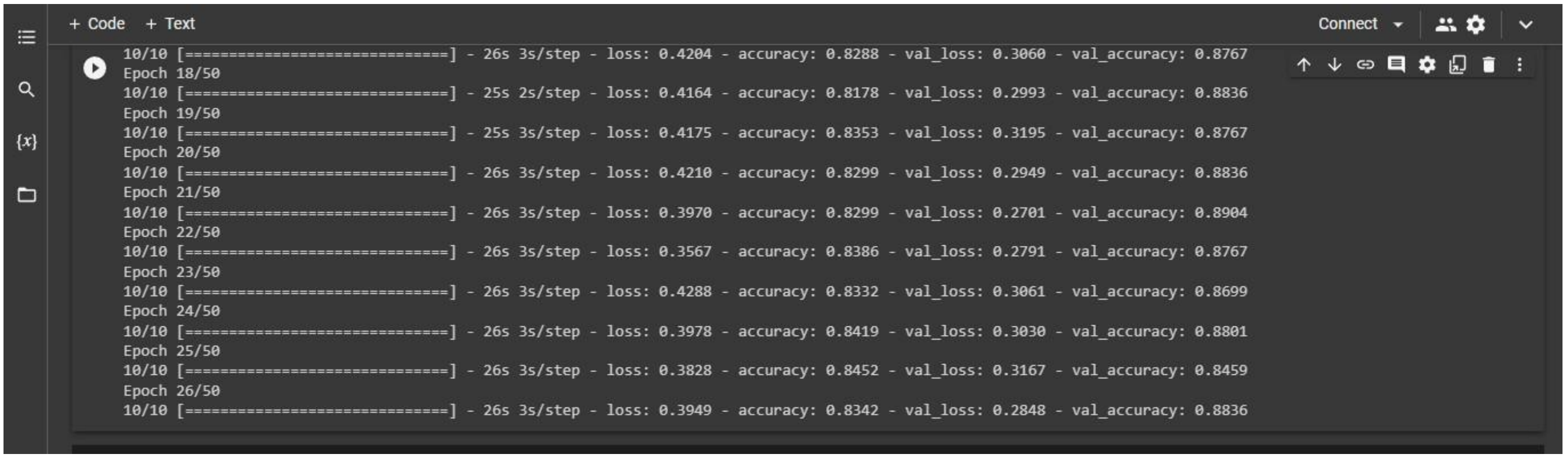
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 100)               204900

 batch_normalization_94 (Bat  (None, 100)              400
 chNormalization)

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 100)               10100

 batch_normalization_95 (Bat  (None, 100)              400
 chNormalization)

 dropout_1 (Dropout)         (None, 100)               0

 dense_2 (Dense)             (None, 3)                 303

=================================================================
Total params: 22,018,887
Trainable params: 215,703
Non-trainable params: 21,803,184
_________________________________________________________________
```

ACCURACY :



```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_
87910968/87910968 [==============================] - 1s 0us/step
Epoch 1/50
10/10 [==============================] - 227s 21s/step - loss: 1.3514 - accuracy: 0.4874 - val_loss: 0.7783 - val_accuracy: 0.6473
Epoch 2/50
10/10 [==============================] - 26s 3s/step - loss: 0.9713 - accuracy: 0.6114 - val_loss: 0.6003 - val_accuracy: 0.7226
Epoch 3/50
10/10 [==============================] - 25s 3s/step - loss: 0.8819 - accuracy: 0.6323 - val_loss: 0.5503 - val_accuracy: 0.7534
Epoch 4/50
10/10 [==============================] - 25s 3s/step - loss: 0.7445 - accuracy: 0.6795 - val_loss: 0.4983 - val_accuracy: 0.7842
Epoch 5/50
10/10 [==============================] - 25s 3s/step - loss: 0.7029 - accuracy: 0.7036 - val_loss: 0.4925 - val_accuracy: 0.8014
Epoch 6/50
10/10 [==============================] - 24s 2s/step - loss: 0.6361 - accuracy: 0.7223 - val_loss: 0.4669 - val_accuracy: 0.8082
Epoch 7/50
10/10 [==============================] - 26s 3s/step - loss: 0.6157 - accuracy: 0.7486 - val_loss: 0.4507 - val_accuracy: 0.7808
Epoch 8/50
10/10 [==============================] - 26s 3s/step - loss: 0.5434 - accuracy: 0.7761 - val_loss: 0.4208 - val_accuracy: 0.7877
Epoch 9/50
10/10 [==============================] - 26s 3s/step - loss: 0.5890 - accuracy: 0.7409 - val_loss: 0.3977 - val_accuracy: 0.8151
Epoch 10/50
10/10 [==============================] - 24s 2s/step - loss: 0.5716 - accuracy: 0.7717 - val_loss: 0.3814 - val_accuracy: 0.8219
Epoch 11/50
10/10 [==============================] - 25s 2s/step - loss: 0.5243 - accuracy: 0.7925 - val_loss: 0.3642 - val_accuracy: 0.8288
Epoch 12/50
10/10 [==============================] - 25s 3s/step - loss: 0.4690 - accuracy: 0.7980 - val_loss: 0.3457 - val_accuracy: 0.8390
Epoch 13/50
10/10 [==============================] - 25s 2s/step - loss: 0.4937 - accuracy: 0.8068 - val_loss: 0.3350 - val_accuracy: 0.8664
Epoch 14/50
10/10 [==============================] - 25s 3s/step - loss: 0.4327 - accuracy: 0.8321 - val_loss: 0.3287 - val_accuracy: 0.8767
Epoch 15/50
10/10 [==============================] - 25s 3s/step - loss: 0.4728 - accuracy: 0.7881 - val_loss: 0.3183 - val_accuracy: 0.8562
Epoch 16/50
10/10 [==============================] - 25s 3s/step - loss: 0.4200 - accuracy: 0.8189 - val_loss: 0.3132 - val_accuracy: 0.8767
Epoch 17/50
10/10 [==============================] - 26s 3s/step - loss: 0.4204 - accuracy: 0.8288 - val_loss: 0.3060 - val_accuracy: 0.8767
Epoch 18/50
10/10 [==============================] - 25s 2s/step - loss: 0.4164 - accuracy: 0.8178 - val_loss: 0.2993 - val_accuracy: 0.8836
Epoch 19/50
10/10 [==============================] - 25s 3s/step - loss: 0.4175 - accuracy: 0.8353 - val_loss: 0.3195 - val_accuracy: 0.8767
```



```
10/10 [==============================] - 26s 3s/step - loss: 0.4204 - accuracy: 0.8288 - val_loss: 0.3060 - val_accuracy: 0.8767
Epoch 18/50
10/10 [==============================] - 25s 2s/step - loss: 0.4164 - accuracy: 0.8178 - val_loss: 0.2993 - val_accuracy: 0.8836
Epoch 19/50
10/10 [==============================] - 25s 3s/step - loss: 0.4175 - accuracy: 0.8353 - val_loss: 0.3195 - val_accuracy: 0.8767
Epoch 20/50
10/10 [==============================] - 26s 3s/step - loss: 0.4210 - accuracy: 0.8299 - val_loss: 0.2949 - val_accuracy: 0.8836
Epoch 21/50
10/10 [==============================] - 26s 3s/step - loss: 0.3970 - accuracy: 0.8299 - val_loss: 0.2701 - val_accuracy: 0.8904
Epoch 22/50
10/10 [==============================] - 26s 3s/step - loss: 0.3567 - accuracy: 0.8386 - val_loss: 0.2791 - val_accuracy: 0.8767
Epoch 23/50
10/10 [==============================] - 26s 3s/step - loss: 0.4288 - accuracy: 0.8332 - val_loss: 0.3061 - val_accuracy: 0.8699
Epoch 24/50
10/10 [==============================] - 26s 3s/step - loss: 0.3978 - accuracy: 0.8419 - val_loss: 0.3030 - val_accuracy: 0.8801
Epoch 25/50
10/10 [==============================] - 26s 3s/step - loss: 0.3828 - accuracy: 0.8452 - val_loss: 0.3167 - val_accuracy: 0.8459
Epoch 26/50
10/10 [==============================] - 26s 3s/step - loss: 0.3949 - accuracy: 0.8342 - val_loss: 0.2848 - val_accuracy: 0.8836
```

Training Accuracy : **83.42 %**

Validation Accuracy : 88.36 %

CHAPTER 7

ADVANTAGES & DISADVANTAGES

ADVANTAGES :

- Good User – Friendly interface
- Less complexity
- Optical recognition
- High-performance classifiers: Deep-learning methods (InceptionV3)
- Wide range of mushrooms image classification
- High speed mushroom image classification
- Useful for all kind of peoples

DISADVANTAGES :

- Less accessibility
- Limited scope (Classifies only 3 kinds of mushrooms only).

CHAPTER 8

CONCLUSION

In conclusion, the project focused on the optical recognition and classification of various mushroom species using deep-learning methods. By leveraging transfer learning techniques and Inception V3 model, the project aimed to achieve high-performance classification accuracy. The classification of mushrooms has a wide range of applications, including food, medicine, conservation, and ecological research. By accurately identifying mushroom species based on their physical features, The project also contributes to the advancement of mycology as a scientific discipline and enthusiasts.

Overall, this project holds promise for further advancements in mushroom-related research and applications.

CHAPTER 9

FUTURE SCOPE

- **Expanding of classification system :**

  The current project focuses on three major categories of mushrooms (Boletus, Lactarius, and Russula). There is potential for expanding the classification system to include more mushroom species from various regions around the world. This expansion would enhance the knowledge base and contribute to a more comprehensive understanding of mushroom diversity.

- **Developing a mobile app** :

  Creating a user-friendly mobile application based on the trained models would make mushroom identification and classification more accessible to a wider audience.

- **Live Camera Detection :**

  This innovative approach eliminates the need for capturing and uploading mushroom images each time, as users can simply activate the camera feature, allowing for automatic real-time classification of mushrooms as soon as they are encountered.

CHAPTER 10

APPENDIX

Source Code :

https://github.com/naanmudhalvan-SI/PBL-NT-GP-14794-1682843285/tree/master/3.%20Final%20Deliverables/Flask

GitHub & Project Video Demo Link :

Git Repo Link :

https://github.com/naanmudhalvan-SI/PBL-NT-GP-14794-1682843285.git

Project Demo Link :

(Demonstration video)

https://drive.google.com/file/d/1BP3lPeCDkyoR62QLl3aMWFQV4NTlsq3f/view?usp=sharing