

**IMAGE RECOGNITION USING IBM CLOUD VISUAL RECOGNITION**

DEVELOPMENT 1



**IMAGE RECOGNITION USING IBM CLOUD VISUAL RECOGNITION**

**Problem Statement:**

* Develop an image recognition system using IBM Cloud Visual Recognition.
* Share your passion for photography by uploading images and watch as the system accurately classifies and describes content.
* Craft engaging visual stories with the help of AI-generated captions.
* Connect with your audience through captivating visuals and compelling narratives!

**Problem Definition:**

* The project involves creating an image recognition system using IBM Cloud Visual Recognition.
* The goal is to develop a platform where users can upload images, and the system accurately classifies and describes the image contents.
* This will enable users to craft engaging visual stories with the help of AI-generated captions, enhancing their connection with the audience through captivating visuals and compelling narratives.

WORK FLOW DIAGRAM

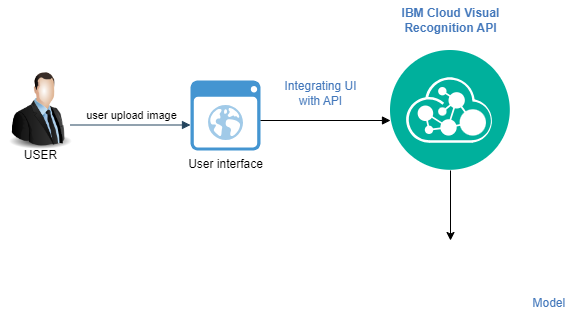
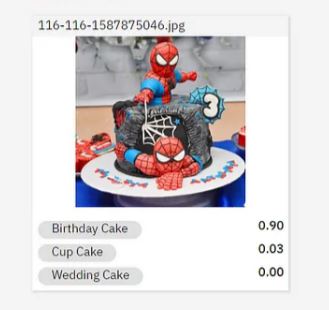


IMAGE CLASSIFICATION



Caption generation

DEVELOPMENT

SETUP IBM CLOUD SERVICE

1. Create the Watson Studio Service

* Log in to IBM Cloud and go to the catalog section.
* Search for the “Watson Studio” service (Type it in the search engine or find it in the “AI” section) and create the service.

2. Create the Visual Recognition Service

* Within IBM Cloud go to the catalog section.
* Search for the “Watson studio” service (Write it in the search engine or find it in the “AI” section) and create the service.
* Go to "Manage" and click on "Create a custom model"
* The “Object Storage” service will be created automatically.

3. Create the model a model in Visual Recognition

Within the Visual Recognition panel, perform the following steps:

• In the "Classify Images" section we click on “Create Model”

• We select previously created services (Watson Studio and Object Storage).

• We click on the “Create” button.

4. Add files to the project and train the model

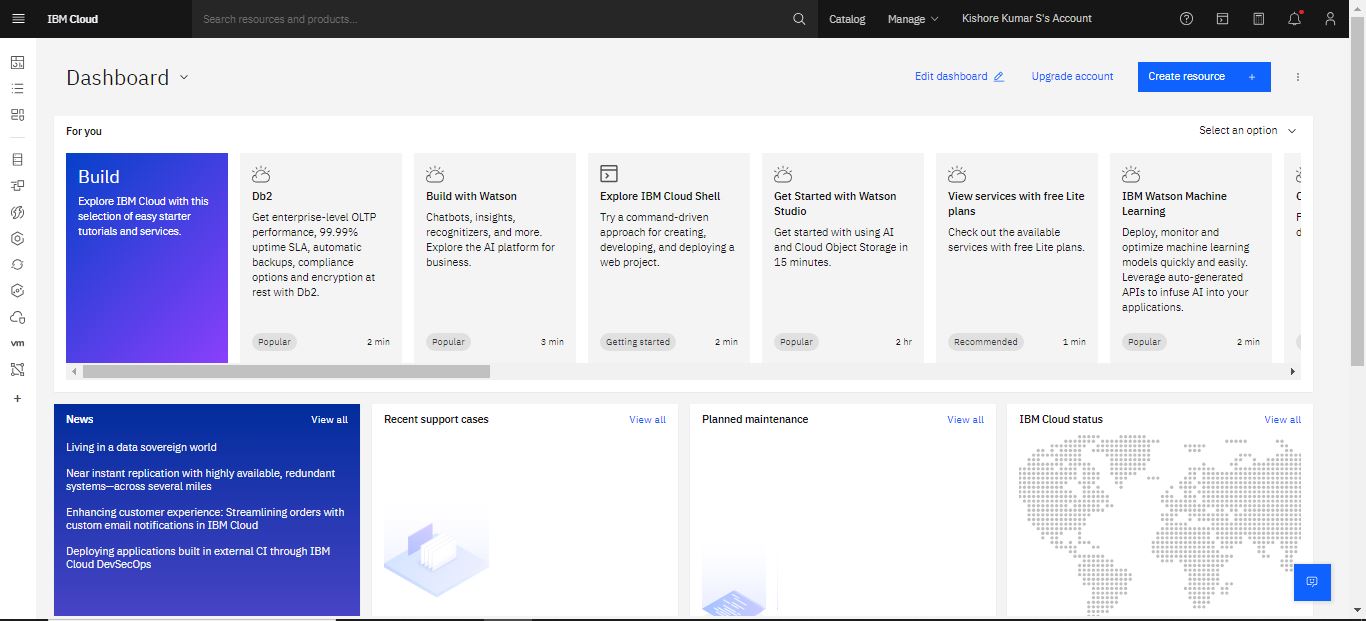
* We changed the name of the Model to “Types of Emotions”
* We add the .zip files found in the folder “Emotion images” in the “Upload to Project” area
* We select all the imported files and click on “Add to model”
* Once the photos are loaded we will see that “Ready to train” appears and we will click on “Train model”.
* Once the training is finished we can test it with the file which is located in the "Test Images" folder

5. Get service credentials

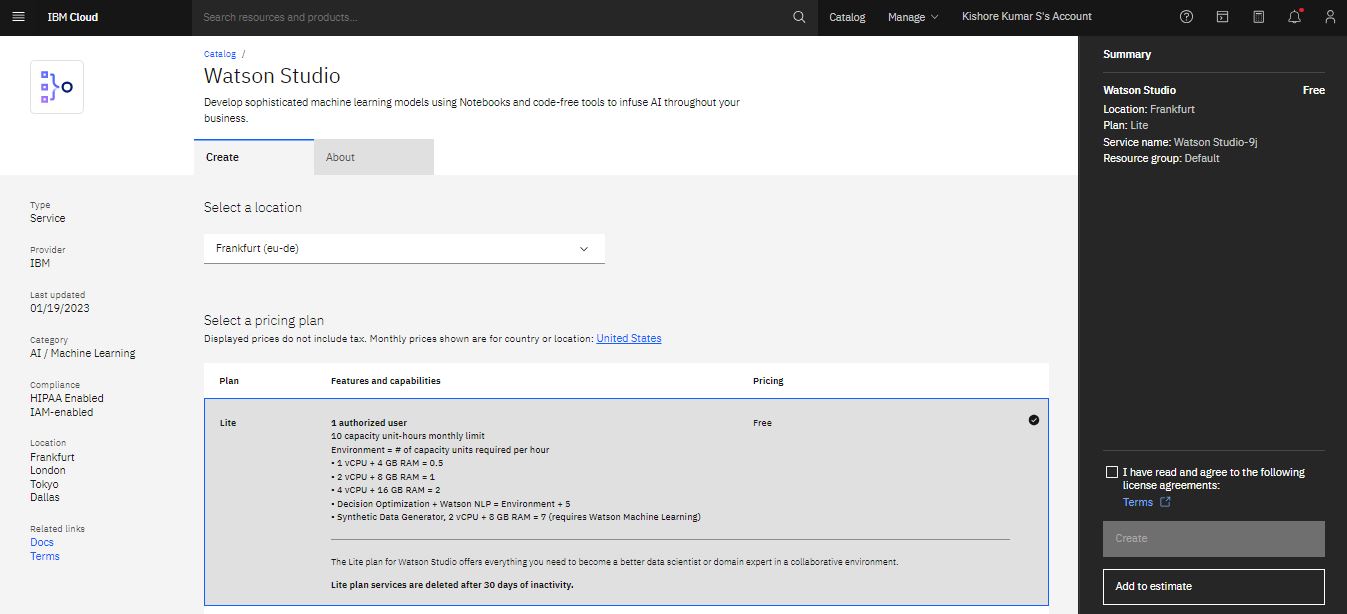
* Open a new tab in your browser to access IBM Cloud.
* In the control panel, click Services > watson
* Go to the "Service Credentials" tab, go to the "writer" credentials, and click on view credentials.
* Copy the credentials into the text editor of your choice so we can use them later

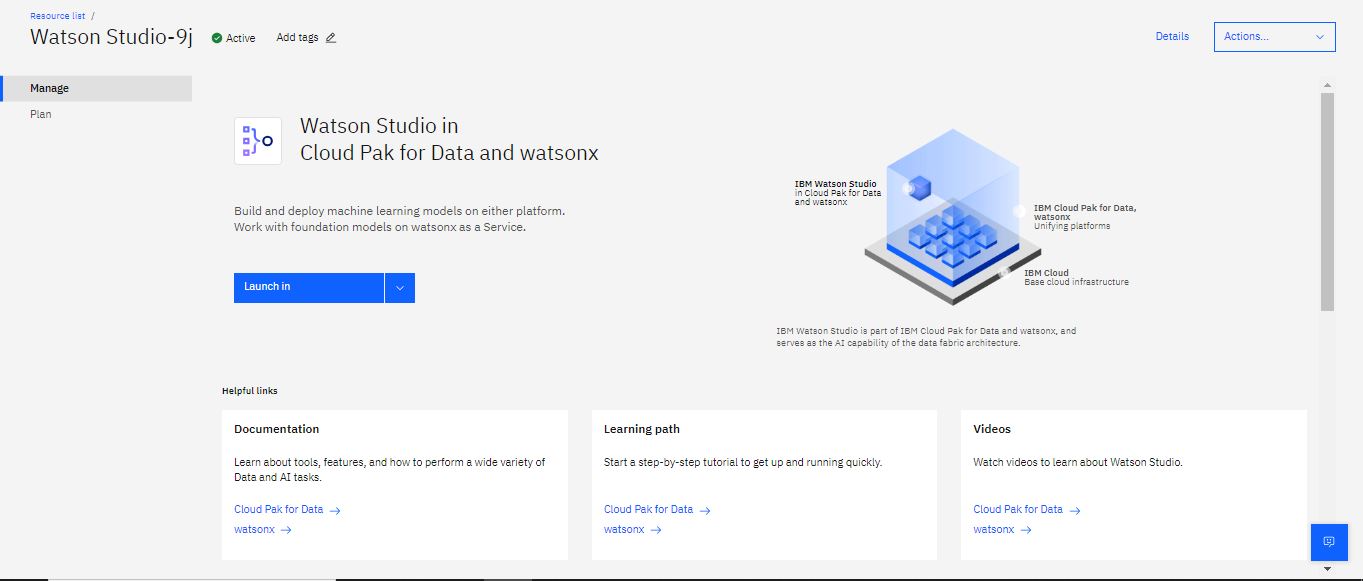
Step1:

* Log in to IBM Cloud and go to the catalog section.
* Search for the “Watson Studio” service (Type it in the search engine or find it in the “AI” section) and create the service.



Step2:

* Within IBM Cloud go to the catalog section.
* Search for the “Watson studio” service (Write it in the search engine or find it in the “AI” section) and create the service.

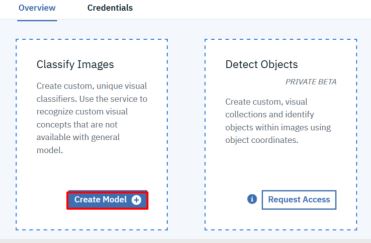


Step 3:

• In the "Classify Images" section we click on “Create Model”

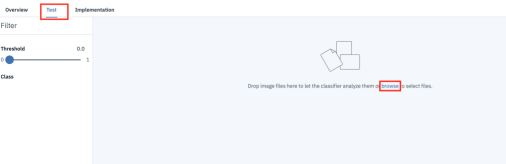
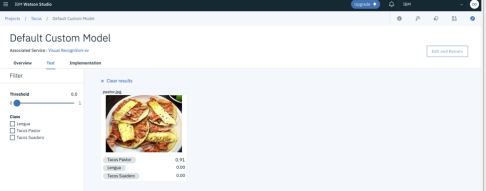
• We select previously created services (Watson Studio and Object Storage).

• We click on the “Create” button.



* Once the photos are loaded we will see that “Ready to train” appears and we will click on “Train model”.
* Once the training is finished we can test it with the file which is located in the "Test Images" folder

Step 4:

* Increase the threshold to get accurate output

Step 5:

Get the API keys associated with your Visual Recognition service. These keys will be used in the project.

Simple web interface

Creating a simple web app involves developing a basic user interface that allows users to upload images and view AI-generated captions. For simplicity, an HTML, CSS, and JavaScript-based web app.

HTML:

<!DOCTYPE html>

<html>

<head>

<title>Image Classification</title>

<link rel="stylesheet" type="text/css" href="/static/style.css">

<meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

<h1>Image Classification Web App</h1>

<form action="http://localhost:5000/classify" method="POST" enctype="multipart/form-data" class="form-container">

<input type="file" name="image" accept=".jpg, .jpeg, .png, .gif, .bmp, .tif" class="file-input" id="image-input">

<div id="image-container">

<img id="image-preview" alt="Choosen Image">

</div>

<input type="submit" value="Classify Image" class="classify-button">

</form>

<div id="results"></div>

<script src="/static/script.js"></script>

<script>

document.querySelector('#file-upload').addEventListener('change', function () {

const fileName = this.files[0].name;

document.querySelector('#file-name').textContent = fileName;

})

document.querySelector('form').addEventListener('submit', function(event) {

event.preventDefault();

const formData = new FormData(this);

fetch('/classify', {

method: 'POST',

body: formData,

})

.then(response => response.json())

.then(data => {

const resultsDiv = document.getElementById('results');

resultsDiv.innerHTML = '';

if ('results' in data) {

data.results.forEach(result => {

const p = document.createElement('p');

p.textContent = `Class: ${result.class}, Score: ${result.score}`;

resultsDiv.appendChild(p);

});

} else if ('error' in data) {

const p = document.createElement('p');

p.textContent = data.error;

resultsDiv.appendChild(p);

}

});

});

</script>

</body>

</html>

this HTML code is a front-end part of a web application, and the actual image classification logic is expected to be implemented on the server side, as indicated by the form's action attribute, which points to "<http://localhost:5000/classify.>" This server would be responsible for processing the uploaded image and returning the classification results.

CSS:

/\* Add your custom styles here \*/

body {

font-family: Arial, sans-serif;

background-image: url('back2.jpg'); /\* Relative path to the image file \*/

background-size: cover ;

background-repeat: no-repeat;

}

h1 {

text-align: center;

}

form {

text-align: center;

margin: 20px auto;

}

.form-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: space-between;

height: 200px; /\* Adjust the height as needed \*/

}

.file-input {

background-color: #007BFF;

color: #fff;

padding: 10px 20px;

border: none;

border-radius: 100px;

cursor: pointer;

font-size: 18px;

transition: background-color 0.3s ease;

}

.file-input:hover {

background-color: #0056b3;

}

/\* Style the "Classify Image" button \*/

.classify-button {

background-color: #4CAF50;

color: white;

padding: 12px 24px;

border: none;

border-radius: 100px;

cursor: pointer;

font-size: 18px;

}

/\* Hover effect for the "Classify Image" button \*/

.classify-button:hover {

background-color: #45a049;

}

/\* Style for the image container \*/

#image-container {

text-align: center;

margin: 20px auto 10px;

padding: 0px;

}

#image-preview {

max-width: 60px;

max-height: 60px;

display: flex;

align-items: end;

}

input[type="file"] {

margin-top: 10px;

padding: 10px;

}

#results {

text-align: center;

margin: 20px auto;

}

p {

font-size: 16px;

}

@media (max-width: 768px) {

.file-name {

padding: 8px 16px;

font-size: 16px;

}

.image-preview {

max-width: 40px;

max-height: 40px;

}

.classify-button {

padding: 10px 20px;

font-size: 16px;

}

Overall, these styles create a visually appealing and responsive design for your image classification web app. The HTML structure is correctly integrated with these styles for the desired visual outcome.

Script.js

document.getElementById('image-input').addEventListener('change', function () {

const imagePreview = document.getElementById('image-preview');

const file = this.files[0];

if (file) {

const reader = new FileReader();

reader.onload = function (e) {

imagePreview.src = e.target.result;

};

reader.readAsDataURL(file);

} else {

imagePreview.src = '';

}

});

It provides a real-time image preview functionality, allowing users to see the selected image before submitting it for classification. The preview image will be displayed in the HTML element with the ID image-preview.

App.py:

import os

import json

from flask import Flask, render\_template, request, jsonify

from ibm\_watson import VisualRecognitionV3

from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator

app = Flask(\_\_name\_\_)

# Replace these with your actual IBM Cloud credentials

api\_key = "BmpffSkPjESnRKDYCoHVKaPQSSHCxrVQU8YS3-WOildA"

url = "<https://api.us-south.visual-recognition.watson.cloud.ibm.com/instances/acfa1ca5-43c1-4b38-9a3d-9eff8477b8f8/v3/classify?url=>"

# Initialize the Watson Visual Recognition service

authenticator = IAMAuthenticator(api\_key)

visual\_recognition = VisualRecognitionV3(

version="2018-03-19",

authenticator=authenticator

)

visual\_recognition.set\_service\_url(url)

# Define a route for the home page

@app.route('/')

def index():

return render\_template('index.html')

# Define a route to handle image classification

@app.route('/classify', methods=['POST'])

def classify\_image():

image = request.files['image']

if not image:

return jsonify({'error': 'No image provided'})

with image as image\_file:

classes = visual\_recognition.classify(

image\_file,

threshold='0.6',

owners=["me"]

)

if classes is not None and 'images' in classes:

results = []

for image in classes['images']:

if 'classifiers' in image:

for classifier in image['classifiers']:

for cls in classifier['classes']:

class\_name = cls['class']

score = cls['score']

results.append({'class': class\_name, 'score': score})

return jsonify({'results': results})

else:

return jsonify({'error': 'No classes found in the image'})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

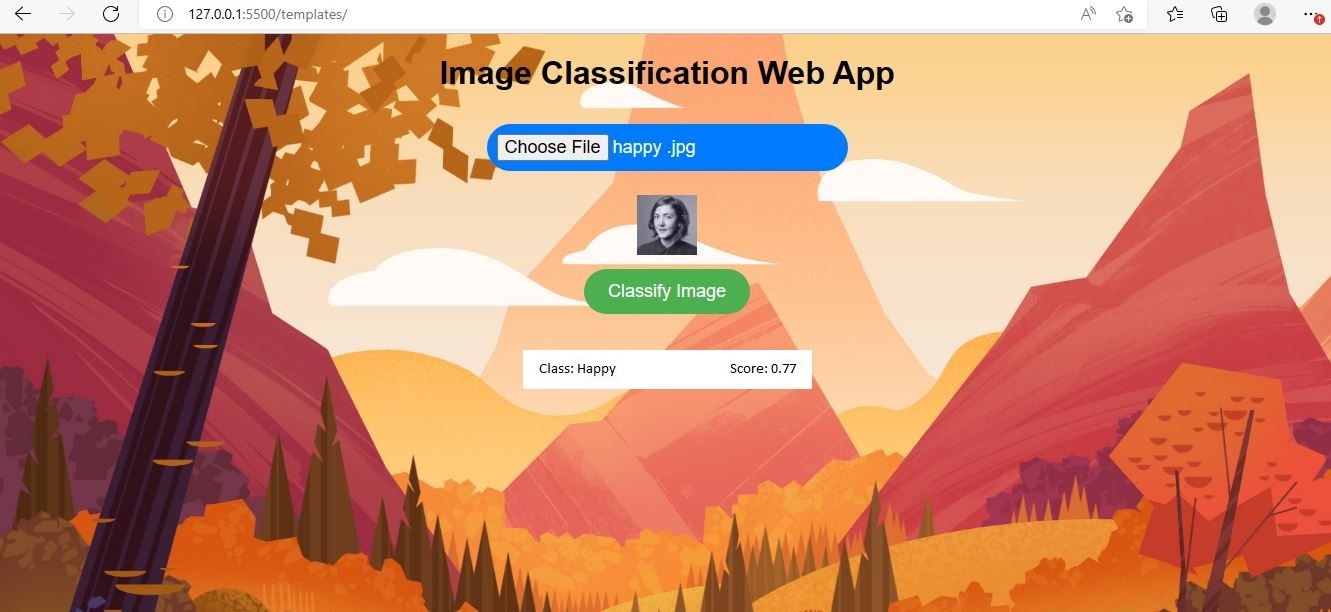
The application serves as a simple web interface for image classification using IBM Watson Visual Recognition. Users can upload images, and the app will return classification results based on the provided image. This include some modules

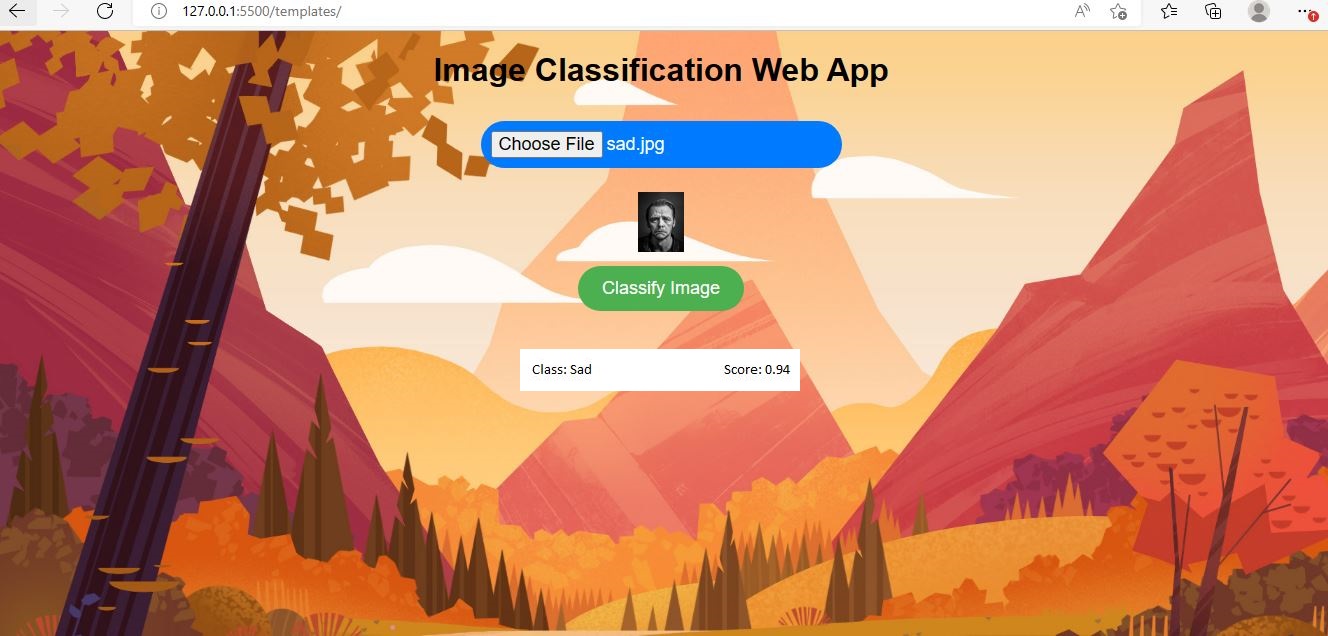
* os: Imports the os library for working with the operating system.
* json: Imports the json library for handling JSON data.
* Flask: Imports the Flask library to create a web application.
* request: Imports the request object from Flask to handle HTTP requests.
* jsonify: Imports the jsonify function from Flask to return JSON responses.
* VisualRecognitionV3: Imports the VisualRecognitionV3 class from the ibm\_watson module for working with IBM Watson Visual Recognition.
* IAMAuthenticator: Imports the IAMAuthenticator class from the ibm\_cloud\_sdk\_core.authenticators module for authenticating with IBM Watson services.

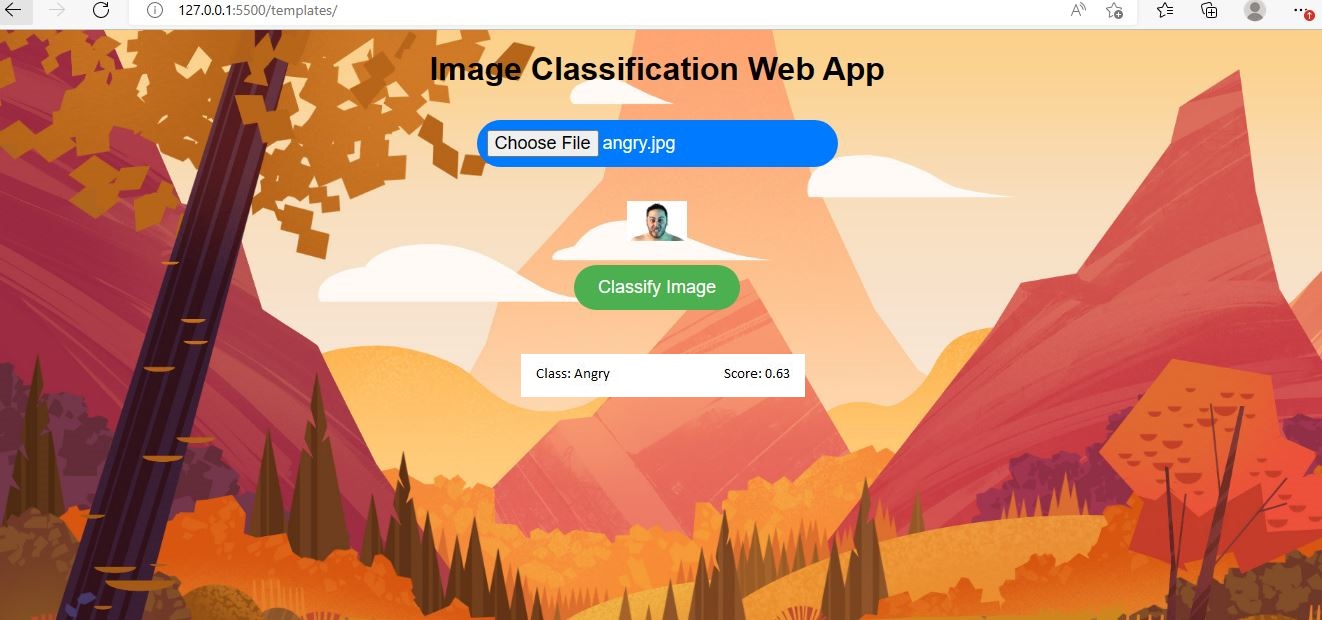
Result:

* Users access the web application through a web browser and are presented with a web page where they can upload images.
* After uploading an image, the web application sends the image to the IBM Watson Visual Recognition service for classification.
* The classification is performed with a specific confidence threshold and owner settings.
* Once the classification is complete, the results are extracted from the response, including class names and scores.
* The results are displayed to the user on the web page.

Output 1:



Output 2:

Output 3:

Conclusion:

It represents a small functional web application for image classification powered by IBM Watson Visual Recognition and built using the Flask web framework. It offers users an easy way to upload images for analysis, with the classification results displayed in a user-friendly manner.

The project's main components, including the Flask web application, IBM Watson Visual Recognition integration, and HTML template, work together to deliver the image classification service. Users can interact with the web interface to upload their images, and the application leverages the capabilities of IBM Watson to classify those images, presenting the results with class names and corresponding confidence scores.

Overall, this project provides a solid starting point for leveraging AI and machine learning capabilities in a web application, making image classification accessible to users in a straightforward and interactive manner.