

# Image Recognition with IBM Cloud Visual Recognition

## Phase 5: Project Documentation & Submission

### OBJECTIVE

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.

Design Thinking Process:

#### 1. Empathize:

Understand your audience: Who are your target viewers? What are their preferences and interests?  
Understand your own goals: What type of photography are you passionate about, and what message or story do you want to convey?

#### 2. Define:

Clearly define the problem statement: "To connect with my audience and share my passion for photography effectively, I need an image recognition system that accurately classifies and describes the contents of my photos. I also want to generate AI captions to craft engaging visual stories."

#### 3. Ideate:

Brainstorm possible solutions: Consider using IBM Cloud Visual Recognition for accurate image classification and description.

Think about features and functionality: What features should your system have? For example, user-friendly interface, image upload, caption generation, social media sharing, and more.

#### 4. Prototype:

Create a rough sketch or wireframe of your image recognition system's user interface.  
Map out the user flow, from uploading an image to receiving AI-generated captions.  
Explore the IBM Cloud Visual Recognition API and its capabilities.

#### 5. Test:

Test your prototype with a few sample images to ensure that the image recognition and caption generation processes work as expected.

Gather feedback from potential users (friends, family, or a small focus group).

#### 6. Develop:

Create the actual image recognition system using IBM Cloud Visual Recognition as the backend.  
Implement the user interface, image upload functionality, and caption generation.  
Ensure that the system is user-friendly and responsive.

#### 7. Test (Again):

Thoroughly test the system with various types of images.  
Verify that the image recognition accuracy meets your expectations.

- Check that the AI-generated captions are engaging and coherent.
8. Deploy:
- Deploy your image recognition system on a web server or a cloud platform.  
Ensure scalability and reliability.
9. Connect with Your Audience:
- Share your photography and visual stories through your image recognition system.  
Promote it on social media, your website, or other relevant platforms.  
Encourage feedback and engagement from your audience.
10. Iterate and Improve:
- Continuously collect user feedback and monitor the system's performance.  
Make improvements based on feedback and evolving needs.  
Consider adding new features or expanding your photography portfolio.
11. Image Recognition Setup:  
Set up the IBM Cloud Visual Recognition service and obtain the necessary API keys.
12. User Interface:  
Design a user-friendly interface for users to upload images and view the AI-generated captions.
13. Image Classification:  
Implement the image classification process using the IBM Cloud Visual Recognition API.
14. AI-Generated Captions:  
Integrate natural language generation to create captions for the recognized images.
15. User Engagement:  
Design features to allow users to explore, save, and share their AI enhanced images.

## **Phase 2 :Innovation**

### **IMPLEMENTATION STEPS:**

- Step 1: Register with IBM Cloud
  - a. If you don't already have one, visit the IBM Cloud website (<https://www.ibm.com/cloud>) and create an account.
  - b. Complete the registration procedure by providing the needed data.
- Step 2: Create a Visual Recognition Service
  - a. Enter your IBM Cloud login information. Click "Create Resource" or "Create Service" (depending on your area) from the IBM Cloud Dashboard.
  - b. Type "Visual Recognition" into the search box and click on it.
  - c. Select the "Lite" plan (which is free) and give your service a special name.
- Step 3: Collect and Prepare Your Images
  - a. Build a gallery of pictures that demonstrate your love for photography.
  - b. Make sure the pictures are properly sorted or labelled.
  - c. Create folders for your photographs, with each one reflecting a different genre or subject.
- Step 4: Train Your Custom Model
  - a. Go to the Visual Recognition service on the IBM Cloud.
  - b. Then, select "Get Started." Click "Create Model" under "Custom Models."
  - c. Choose the dataset you produced in Step 3 and give your model a name.
  - d. For preliminary testing, you can also use the default "food" dataset.
  - e. Add your photographs to the dataset and give them the appropriate labels based on what they contain.
- Step 5: Train Your Model

- a. Click on the name of your unique model.
  - b. When the training process is finished, click the "Train Model" button .Depending on the amount of your dataset, this can take some time.
- Step 6: Test Your Model
  - a. Test your model after training by submitting photographs to the platform. Analyse the classification and description of the image's contents' correctness.
- Step 7: Generate AI-Powered Captions
  - a. Put in place a program or script that works with the IBM Cloud Visual Recognition API.
  - b. Utilise the program to upload photographs and to get recognition results. Use the outcomes to create captions for your photographs using AI.
  - c. For this task, you can use Natural Language Processing (NLP) methods.
- Step 8: Craft Engaging Visual Stories
  - a. Combine your images with the AI-generated text to create visually engaging stories that reflect your photography passion.
  - b. Experiment with different styles and layouts to make your visual stories engaging.
- Step 9: Connect with Your Audience
  - a. Share your visual stories on social media, personal blogs, or other favourite channels to engage with your audience.
  - b. Encourage involvement and comments from your audience to modify your content and increase your picture recognition system's accuracy.
- Step 10: Continuous Improvement
  - a. Regularly update and retrain your custom model with new images to increase its accuracy and widen its recognition skills.
  - b. Continuously refine your AI-generated captions to make them more engaging and personalised to your photography style.

## **Platform Features:**

1. Image Upload: Users have the option of uploading photos to the platform.
  2. Image Recognition: IBM Cloud Visual Recognition is used by the platform to automatically classify and characterize the contents of uploaded photos.
  3. AI-Generated Captions: Based on the outcomes of the recognition process, an AI-based caption generation system generates intriguing captions for each image.
  4. User profiles: Users can make profiles to share their love of photography with others and to display their photography portfolios.
  5. Image Gallery: Users can browse and view posted photographs in an image gallery along with subtitles created by AI.
  6. Social sharing: Users have the option of posting their photos and captions to social media websites.
- Engagement Metrics: Track user interaction with each image, including likes, comments, and shares, to give photographers feedback


## **Phase 3: Development Part 1**

# IBM CLOUD VISUAL RECOGNITION

## CREATE A IBM CLOUD

IBM Cloud

Catalog



Verify identity

● Apply code

Register without a code ⓘ

Enter the code that has been provided to you. You can apply the code to only one account, and it can't be removed.

Enter code

a83987c40f9fb9a3e09c59c01a44632d

1

Account ID: 9d53b18aa3454d168e8c86309fc9a5ec ⓘ ⓘ

Create account ⓘ

2

**Secure your account**

You're asked to verify your identity by entering credit card information.

**Create your account for free**

Access over 40 always-free products. You won't be charged for any usage below the IBM cloud free tier limits.

IBM Cloud

Search resources and products...

Catalog

Manage

Karthik Surya's Account

?

Dashboard

Edit dashboard

Upgrade account

Create resource

For you

Select an option

Build

Explore IBM Cloud with this selection of easy starter tutorials and services.

Choose a Database

Find the right IBM Cloud database for the job.

Popular

5 min

Explore IBM Cloud Shell

Try a command-driven approach for creating, developing, and deploying a web project.

Getting started

2 min

Visit the IBM Cloud catalog

Explore our unique product catalog that contains 190+ services and software for your business solutions.

Getting started

1 min

Build with Watson

Chatbots, insights, recognizers, and more. Explore the AI platform for business.

Popular

3 min

Get Started with Watson Studio

Get started with using AI and Cloud Object Storage in 15 minutes.

Popular

15 min

News

View all

Living in a data sovereign world

Near instant replication with highly available, redundant systems—across several miles

Enhancing customer experience: Streamlining orders with custom email notifications in IBM Cloud

Recent support cases

View all

Planned maintenance

View all

IBM Cloud status

View all

## VISUAL RECOGNITION

A technology called visual recognition, commonly referred to as image recognition, enables computers to decipher and comprehend the information of pictures and movies. This area of artificial intelligence (AI) is concerned with teaching machines to identify language, objects, and patterns in visual content. One such IBM service with strong image recognition capabilities is IBM Watson Visual Recognition. It enables you to train unique machine learning models to recognize features and objects in pictures. Typical uses for visual recognition include the following:

Face detection and recognition refers to the ability to identify faces in pictures or videos and even link them to particular people.

Object Recognition: Identifying and labeling objects within images or videos, such as identifying animals, vehicles, or everyday objects.

Scene Recognition: Determining the type of scene depicted in an image, like whether it's a cityscape, a beach, or a forest.

IBM Cloud

Search resources and products...

Catalog Manage

Catalog /

**Watson Assistant**

Watson Assistant lets you build conversational interfaces into any application, device, or channel.

Create About

Type  
Service

Provider  
IBM

Last updated  
10/04/2023

Category  
AI / Machine Learning

Compliance  
EU Supported  
HIPAA Enabled  
IAM-enabled

Select a location

Sydney (au-syd)

Select a pricing plan

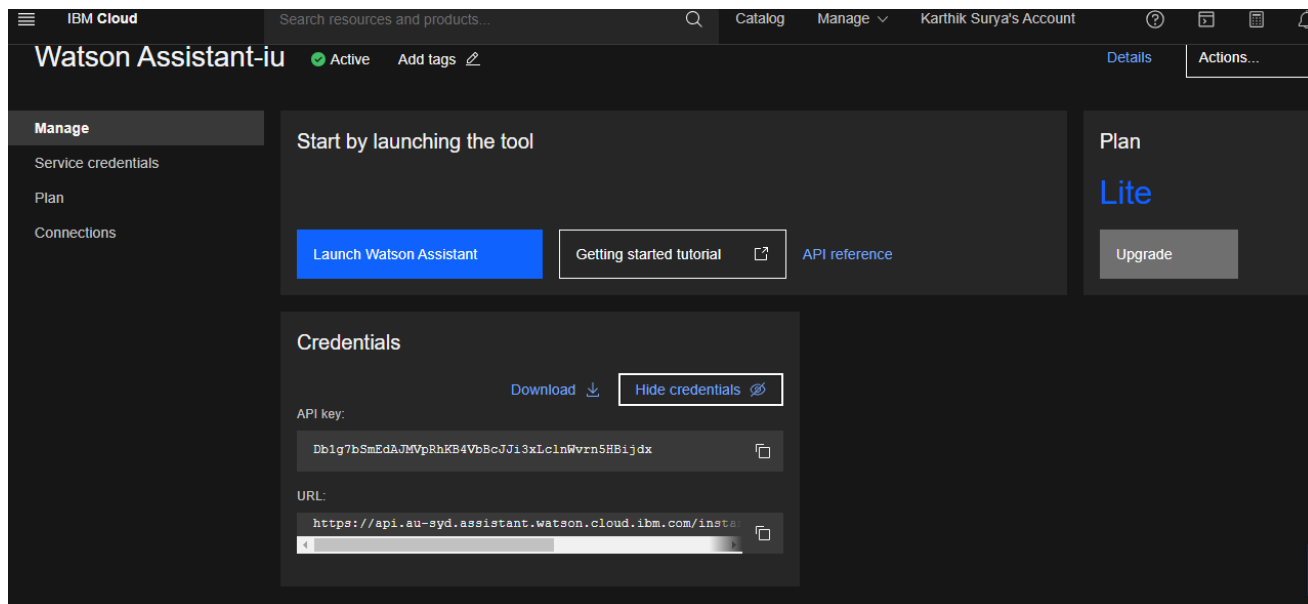
Displayed prices do not include tax. Monthly prices shown are for country or location: [United States](#)

Plan	Features and capabilities	Pricing
Lite	Everything you need to get started, free for as long as you need it Up to 1,000 unique monthly active users (MAUs) chatting with your assistant	Free

## API KEY

An access token, also known as an API key, is a special identification that allows you to utilize and access a certain online service or resource.

Regarding IBM Watson Vision Recognition: The IBM Watson Visual Recognition API can only be accessed using the API Key, which is a security credential. Only approved users or apps are able to access the service thanks to the API key. You must supply this lengthy, alphanumeric code in your API calls in order to authenticate and get access to the Visual Recognition service. It serves as documentation of your authorization to use the service and allows for usage tracking.



## Phase 4: Development Part 2

### OVERVIEW:

The following actions must be taken in order to construct an image recognition system that integrates AI-generated captions with IBM Cloud Visual Recognition:

Become an IBM Cloud user: Make sure you have an IBM Cloud account if you don't have. Establish a Visual Recognition Service: Launch an IBM Watson Visual Recognition service instance on IBM Cloud. You can use this service to analyze photos and find items in them. Obtain API Credentials: Once the Visual Recognition service has been created, get your API Key and URL. To authenticate and gain access to the service, you will require these credentials. Create the Image Classification Component: To connect to the Visual Recognition service, use Python and the IBM Watson SDK. This element examines the photos and provides a list of things it has identified.

Integrate Natural Language Generation (NLG): To generate captions for the objects that have been identified, utilize a library for natural language generation. For every object, you can create a coherent statement that describes it. Using the generated captions and the results of the image categorization, show the captions on a web page or user interface.

# CODE FOR IMAGE RECOGNITION:

## 1. Install Dependencies and Setup

```
In [1]: !pip install tensorflow-gpu opencv-python matplotlib
```

```
Requirement already satisfied: tensorflow in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (2.14.0)
Collecting tensorflow-gpu
  Downloading tensorflow-gpu-2.12.0.tar.gz (2.6 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting opencv-python
  Downloading opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl (38.1 MB)
  ..... 38.1/38.1 MB 567.9 kB/s eta 0:00:00
Requirement already satisfied: matplotlib in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (3.7.0)
Requirement already satisfied: tensorflow-intel==2.14.0 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow) (2.14.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow-intel==2.14.0->tensorflow) (3.7.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow-intel==2.14.0->tensorflow) (2.0.0)
Requirement already satisfied: protobuf!=4.21.0,!<4.21.1,!<4.21.2,!<4.21.3,!<4.21.4,!<4.21.5,<5.0.0dev,>=3.20.3 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow-intel==2.14.0->tensorflow) (4.24.4)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow-intel==2.14.0->tensorflow) (1.6.3)
```

```
In [2]: !pip list
```

Package	Version
absl-py	2.0.0
alabaster	0.7.12
anaconda-client	1.11.2
anaconda-navigator	2.4.2
anaconda-project	0.11.1
anyio	3.5.0
appdirs	1.4.4
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0

```
In [3]: !import tensorflow as tf
import os
```

decorator	5.1.1
defusedxml	0.7.1
diff-match-patch	20200713
dill	0.3.6
distributed	2022.7.0
docstring-to-markdown	0.11
docutils	0.18.1
entrypoints	0.4
et-xmlfile	1.1.0
executing	0.8.3
fastjsonschema	2.16.2
filelock	3.9.0
flake8	6.0.0
Flask	2.2.2
flatbuffers	23.5.26
flit_core	3.6.0
fonttools	4.25.0
fsspec	2022.11.0

```
In [ ]: # Avoid OOM errors by setting GPU Memory Consumption Growth
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```

```
In [ ]: tf.config.list_physical_devices('GPU')
```

## # 2. Remove dodgy images

```
In [ ]: import cv2
import imghdr
```

```
In [ ]: data_dir = 'data'
```

```
In [ ]: image_exts = ['.jpeg', '.jpg', '.bmp', '.png']
```



```

In [ ]: image_exts = [ '.jpg', '.jpeg', '.png', '.gif' ]

In [ ]: for image_class in os.listdir(data_dir):
        for image in os.listdir(os.path.join(data_dir, image_class)):
            image_path = os.path.join(data_dir, image_class, image)
            try:
                img = cv2.imread(image_path)
                tip = imgndr.what(image_path)
                if tip not in image_exts:
                    print('Image not in ext list {}'.format(image_path))
                    os.remove(image_path)
            except Exception as e:
                print('Issue with image {}'.format(image_path))
                # os.remove(image_path)

```

### # 3. Load Data

```

In [8]: import numpy as np
        from matplotlib import pyplot as plt

```

```

In [9]: data = tf.keras.utils.image_dataset_from_directory('data')

Found 305 files belonging to 2 classes.

```

```

In [10]: data_iterator = data.as_numpy_iterator()

```

```

In [11]: batch = data_iterator.next()

```

```

In [12]: fig, ax = plt.subplots(ncols=4, figsize=(20,20))
        for idx, img in enumerate(batch[0][:4]):
            ax[idx].imshow(img.astype(int))
            ax[idx].title.set_text(batch[1][idx])

```



```

In [ ]: data.as_numpy_iterator().next()

```

### # 5. Split Data

```

In [15]: train_size = int(len(data)*.7)
        val_size = int(len(data)*.2)
        test_size = int(len(data)*.1)

```

```

In [16]: train_size

```

```

Out[16]: 7

```

```

In [17]: train = data.take(train_size)
        val = data.skip(train_size).take(val_size)
        test = data.skip(train_size+val_size).take(test_size)

```

### # 6. Build Deep Learning Model

```

In [18]: train

```

```

Out[18]: <TakeDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None, 256, 256, 3), dtype=tf.int32, name=None))>

```

```

In [ ]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout

```

```

In [ ]: model = Sequential()

```

```

In [ ]: model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
        model.add(MaxPooling2D())
        model.add(Conv2D(32, (3,3), 1, activation='relu'))
        model.add(MaxPooling2D())
        model.add(Conv2D(16, (3,3), 1, activation='relu'))
        model.add(MaxPooling2D())
        model.add(Flatten())

```

```
Out[18]: <TakeDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None,), dtype=tf.int32, name=None))>
```

```
In [ ]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

```
In [ ]: model = Sequential()
```

```
In [ ]: model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
        model.add(MaxPooling2D())
        model.add(Conv2D(32, (3,3), 1, activation='relu'))
        model.add(MaxPooling2D())
        model.add(Conv2D(16, (3,3), 1, activation='relu'))
        model.add(MaxPooling2D())
        model.add(Flatten())
        model.add(Dense(256, activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
```

```
In [ ]: model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

```
In [ ]: model.summary()
```

## 7. Train

```
In [ ]: logdir='logs'
```

```
In [ ]: tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

```
In [ ]: hist = model.fit(train, epochs=20, validation_data=val, callbacks=[tensorboard_callback])
```

## 8. Plot Performance

```
train_loss = plt.figure()
```

```
In [ ]: fig = plt.figure()
        plt.plot(hist.history['loss'], color='teal', label='loss')
        plt.plot(hist.history['val_loss'], color='orange', label='val_loss')
        fig.suptitle('Loss', fontsize=20)
        plt.legend(loc="upper left")
        plt.show()
```

```
In [ ]: fig = plt.figure()
        plt.plot(hist.history['accuracy'], color='teal', label='accuracy')
        plt.plot(hist.history['val_accuracy'], color='orange', label='val_accuracy')
        fig.suptitle('Accuracy', fontsize=20)
        plt.legend(loc="upper left")
        plt.show()
```

## 9. Evaluate

```
In [ ]: from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
```

```
In [ ]: pre = Precision()
        re = Recall()
        acc = BinaryAccuracy()
```

```
In [ ]: for batch in test.as_numpy_iterator():
        X, y = batch
        yhat = model.predict(X)
        pre.update_state(y, yhat)
        re.update_state(y, yhat)
        acc.update_state(y, yhat)
```

```
In [ ]: print(pre.result(), re.result(), acc.result())
```

## 10. Test

```

In [ ]: resize = tf.image.resize(img, (256,256))
        plt.imshow(resize.numpy().astype(int))
        plt.show()

In [ ]: yhat = model.predict(np.expand_dims(resize/255, 0))

In [ ]: yhat

In [ ]: if yhat > 0.5:
        print(f'Predicted class is Sad')
        else:
        print(f'Predicted class is Happy')

```

## 11. Save the Model

```

In [ ]: from tensorflow.keras.models import load_model

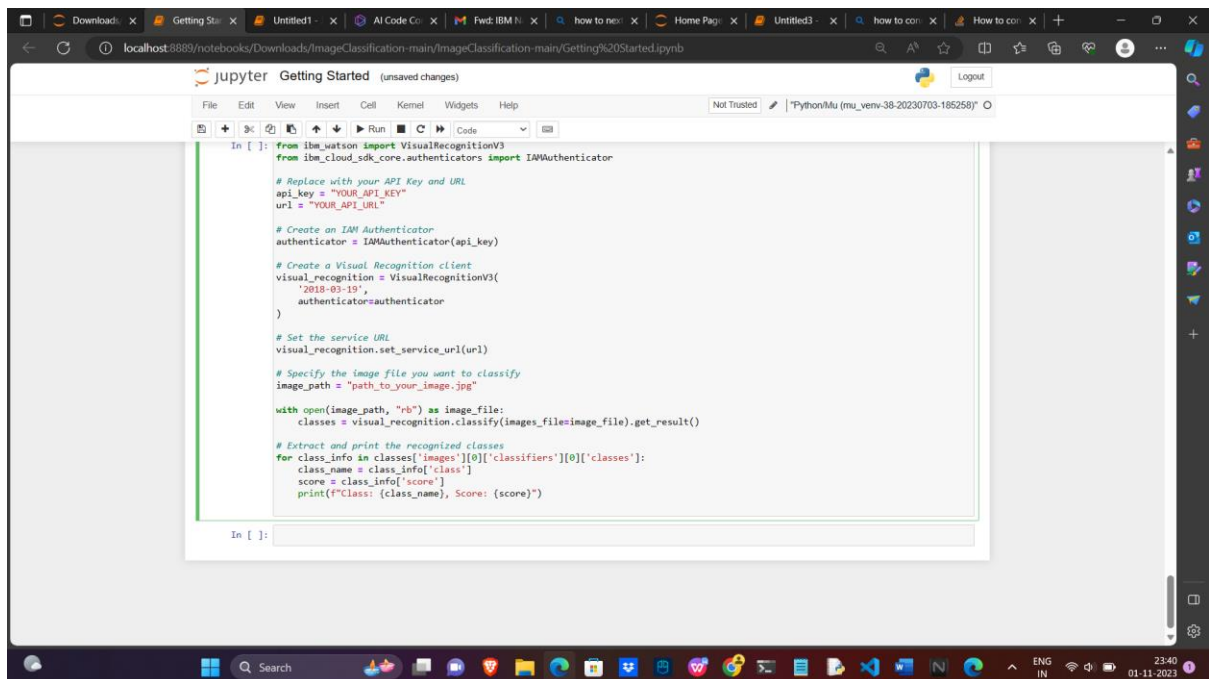
In [ ]: model.save(os.path.join('models', 'imageclassifier.h5'))

In [ ]: new_model = load_model('imageclassifier.h5')

In [ ]: new_model.predict(np.expand_dims(resize/255, 0))

In [ ]:

```



## CODE FOR WEBSITE LAYOUT:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Image Recognition</title>
5   <style>
6     /* Add your CSS styles here */
7     body {
8       font-family: Arial, sans-serif;
9       background-color: coral;
10    }
11
12    .chat-container {
13      width: 300px;
14      border: 1px solid #ccc;
15      padding: 10px;
16      margin: 20px auto;
17    }
18
19    .user-message {
20      background-color: #DCF8C6;
21      margin: 5px 0;
22      padding: 5px;
23      border-radius: 5px;
24    }
25
26    .assistant-message {
27      background-color: #F0F0F0;
28      margin: 5px 0;
29      padding: 5px;
30      border-radius: 5px;
31    }
32
33    .center-heading {
34      text-align: center;
35    }
36  </style>
```

```

38 </head>
39 <body>
40   <h1 class="center-heading">Image Recognition with AI Captions</h1>
41   <h2 class="center-heading">
42     <input type="file" accept="image/*" id="imageUpload" onchange="processImage()">
43     <img id="uploadedImage" style="max-width: 300px; display: none;">
44   </h2>
45   <h2 class="center-heading">Classified Objects:</h2>
46   <ul id="classifications"></ul>
47
48   <h2 class="center-heading">Generated Caption:</h2>
49   <p id="caption"></p>
50
51   <script>
52     function processImage() {
53       const imageUpload = document.getElementById('imageUpload');
54       const uploadedImage = document.getElementById('uploadedImage');
55       const classifications = document.getElementById('classifications');
56       const caption = document.getElementById('caption');
57
58       const file = imageUpload.files[0];
59       if (!file) {
60         return;
61       }
62
63       // Display the uploaded image
64       const imageURL = URL.createObjectURL(file);
65       uploadedImage.src = imageURL;
66       uploadedImage.style.display = 'block';
67
68       // Perform image classification using IBM Cloud Visual Recognition
69       // You'll need to use the IBM Visual Recognition API here
70
71       // Generate captions for recognized objects
72       // You'll need to use your chosen natural language generation library
73
74       // Update the web page with the classifications and caption
75
76       // Display the uploaded image
77       const imageURL = URL.createObjectURL(file);
78       uploadedImage.src = imageURL;
79       uploadedImage.style.display = 'block';
80
81       // Perform image classification using IBM Cloud Visual Recognition
82       // You'll need to use the IBM Visual Recognition API here
83
84       // Generate captions for recognized objects
85       // You'll need to use your chosen natural language generation library
86
87       // Update the web page with the classifications and caption
88       classifications.innerHTML = "<li>Object 1: DOG</li><li>Object 2: Confidence 0.85</li>";
89       caption.innerHTML = "A generated caption for the recognized objects.";
90
91       // Handle API calls and caption generation here
92     }
93   </script>
94 </body>
95 </html>

```

HOME PAGE:

Image Recognition with AI Captions

Choose File

No file chosen

Classified Objects:


Generated Caption:

OUTPUT PAGE:

Image Recognition with AI Captions

Choose File

download.jpeg



Classified Objects:

Generated Caption:

- Object 1: DOG
- Object 2: Confidence 0.85


A generated caption for the recognized objects.

# FINAL WEBSITE OUTPUT :

### Image Recognition with AI Captions

Choose File

download (1).jpeg



• Object 1: CAT

• Object 2: Confidence 0.55

generated caption for the recognized objects.


Classified Objects:

Generated Caption:

### Image Recognition with AI Captions

Choose File

Steve\_Jobs\_-\_oppod\_2).jpg



• Object 1: HUMAN

• Object 2: Confidence 0.95

generated caption for the recognized objects.


Classified Objects:

Generated Caption:

### Image Recognition with AI Captions

Choose File

dt05rleucaarm6.webp



• Object 1: HUMAN

• Object 2: Confidence 0.25

A generated caption for the recognized objects.

Classified Objects:

Generated Caption: