**IMAGE RECOGNITION WITH IBM CLOUD VISUAL**

**RECOGNITION**

PROBLEM DEFINITION:

The project involves creating an image recognition system using IBM Cloud Visual Recognition. In this project, users can upload images, and the system accurately classifies and describes the image contents.

Project’s Objective:

Our project’s objective is to upload the images and recognizing it using IBMs cloud visual recognition.

DESIGN THINKING:

Image recognition setup:

* Collect a large dataset of images.
* Rescale images to a consistent size. This ensures uniformity in input data for the model.
* Divide the dataset into training, validation, and test sets.
* Train the chosen model using the training data.
* Use appropriate metrics to evaluate the model’s performance.
* Test the trained model on unseen data to ensure it generalizes well.
* Design the user interface of your application for image recognition.

*User interface:*

* Users can use visual recognition to categorize images.
* If the system involves capturing images, integrate the camera function seamlessly. Ensure high-quality image capture.
* Display comprehensive details, including images, descriptions, and specifications.
* If applicable, enable users to create accounts to save their favorite products or create wish lists.
* Allow users to manage their privacy settings, especially regarding data collection and targeted advertisements.
* Conduct usability testing with real users to identify any pain points and areas for improvement.

*Image classification:*

* Gather a diverse set of images for each category you want to recognize.
* Evaluate the results and iterate on the training process if needed. You might need to add more diverse data or tweak the parameters to improve accuracy.
* Use the Visual Recognition API endpoints to classify images in real-time as they are uploaded to your application.
* Collect feedback and use it to improve the model. You can retrain the model with new data to enhance its accuracy over time.

*AI-Generated captions:*

* Gather a diverse set of images that you want to recognize.
* Use the API key provided by IBM Cloud Visual Recognition to authenticate your requests.
* Send the images to the Visual Recognition API for analysis. The API will return JSON data containing information about the recognized images.

*User engagement:*

* Ensure the user interface is intuitive and easy to navigate.
* Improve the accuracy of recognition algorithms to provide reliable results.
* Allow users to customize recognition models based on their specific needs.
* Ensure data security and privacy of user-uploaded images.
* Create online forums or communities where users can share their experiences and tips regarding recognition.
* Offer periodic reports on recognition performance and suggestions for improvement.

Platforms Layout:

* Sign up and Log in to IBM Cloud.
* Create a Visual Recognition Service.
* Get API Credentials.
* Develop your Application.
* User Interface (UI) Design.
* Enhancements and Customizations.
* Testing.
* Deployment.
* Documentation and User Guides.
* Security and Compliance.

Features :

* Custom Image Classification.
* Pre-Trained Models.
* Visual Tagging.
* Explicit Content Detection.
* Face Detection.
* Text Recognition.
* Data Enrichment.
* Quality and Compliance.

Technical implementation:

* Visual Inspection: Identifying parts as defective or non-defective in manufacturing can quickly inspect thousands of parts on an assembly line.
* Image Classification: Categorizing images based on the image content. This is especially useful in applications such as image retrieval and recommender systems in e-commerce.
* Automated Driving: The ability to recognize a stop sign or a pedestrian in an image is crucial to autonomous driving applications.
* Robotics: Image recognition can be used by robots to identify objects and enhance autonomous navigation by identifying locations or objects on their path.

Support Vector Machines

* SVMs describe features by making histograms of images. They use a sliding detection window technique by moving around the image. The algorithm then takes the test picture and compares the trained histogram values with the ones of various parts of the picture to check for close matches.

Bag of Features

* Bag of Features models like Scale Invariant Feature Transformation (SIFT) does pixel-by-pixel matching between a sample image and its reference image. The trained model then tries to pixel match the features from the image set to various parts of the target image to see if matches are found.
* Some other machine learning models widely used in computer vision include:
* Regression Algorithms
* Instance-based Algorithms
* Regularization Algorithms
* Decision Tree Algorithms
* Bayesian Algorithms
* Clustering Algorithms

Step 1: Create an IBM account

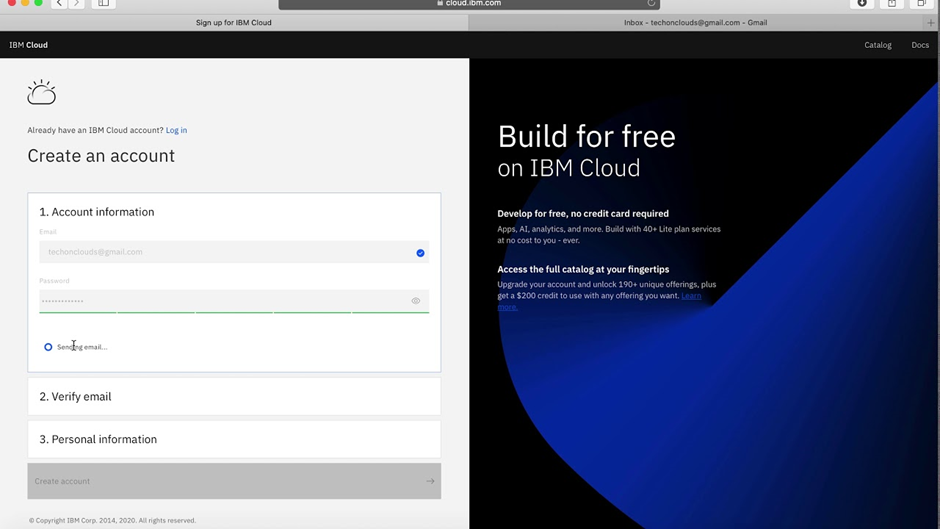
• Enter your email address, and create a password.

• Complete any necessary verification steps.

• Provide additional information.

• Read and accept the terms and conditions of using IBM Cloud services.

• Click on the "Create Account"



Step 2: Set up of Visual recognition service and obtain API keys

• Go to the IBM Cloud Catalog.

• In the catalog, search for "Visual Recognition".

• Click on the Visual Recognition service to configure it.

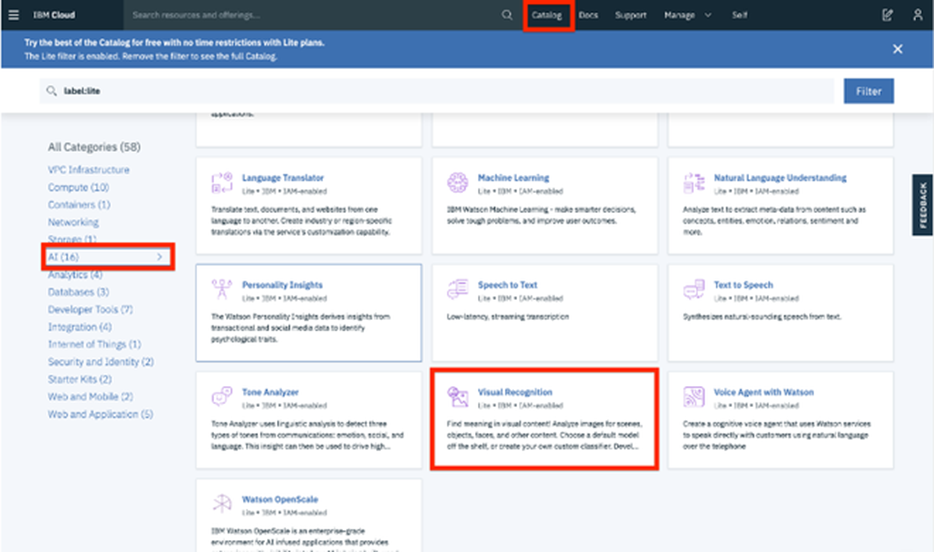
• Provide a unique name for your service instance.

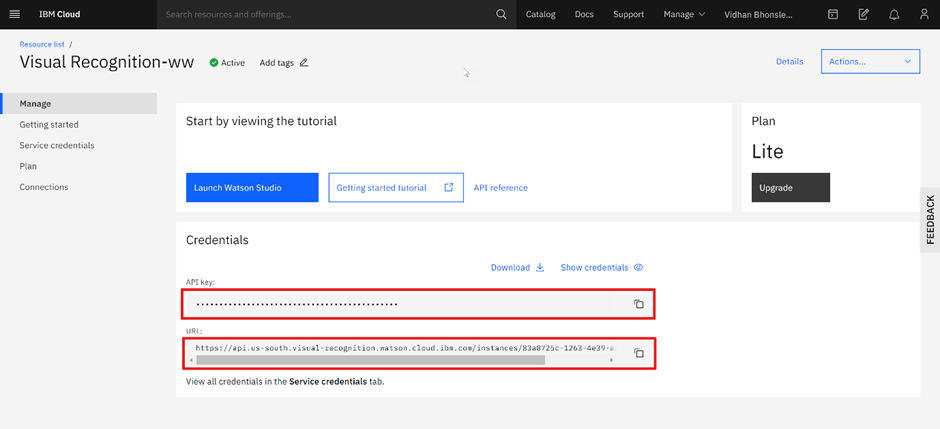
• Choose a region for your service (e.g., Dallas, London).

• Once the service instance is created, go to the service dashboard.

• Find the "Manage" tab, and from there, you can get your API key and the service URL.

• Explore the API documentation provided by IBM Cloud for Visual Recognition.



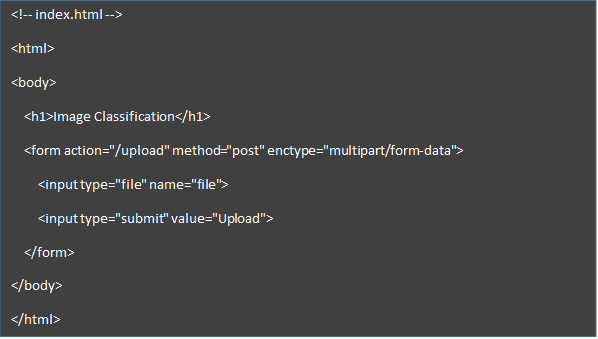


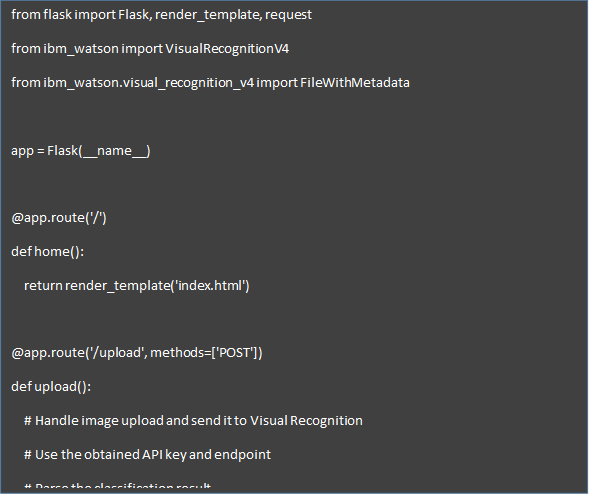
Step 3: Design of simple web interface

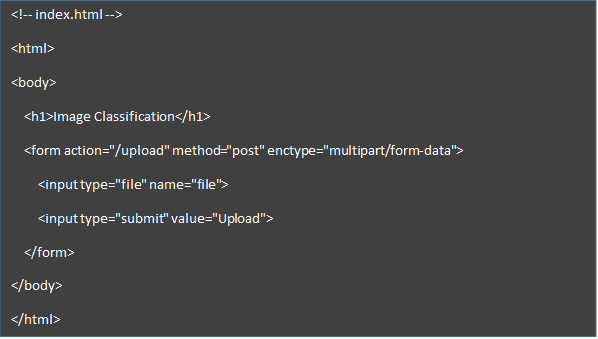
• Navigate to the "Catalog" and find the "Visual Recognition" service.

• Once the Visual Recognition service is created, obtain the API key and endpoint from the IBM Cloud dashboard.

• Install Necessary Libraries: Install Flask for building the web application





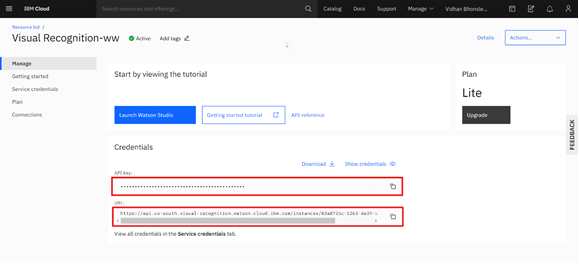


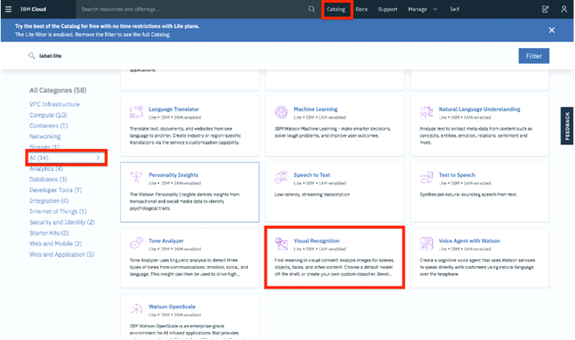
**Given problem statement:**

* Continue building the image recognition system by integrating IBM Cloud Visual Recognition and AI-generated captions.
* Implement the image classification process using the IBM Cloud Visual Recognition API.
* Use natural language generation captions for the recognized images.

Step 1: Integrating IBM Cloud Visual Recognition with AI-generated caption

* Use the IBM Watson Visual Recognition tool to train your model. Upload your images and provide appropriate labels for each category.
* Train the model using your dataset. IBM Watson Visual Recognition will use machine learning algorithms to learn from your data.
* After training the model, you can use it to analyze new images and generate predictions.
* When you receive predictions (labels) for an image from Visual Recognition, use a natural language processing (NLP) model.
* Use the Visual Recognition API to classify images.
* Combine the labels and generated captions to create a meaningful description of the image.





Step :2 IMPLEMENTATION

import requests

import openai

# IBM Visual Recognition API Endpoint and API Key

VISUAL\_RECOGNITION\_API\_ENDPOINT = "YOUR\_VISUAL\_RECOGNITION\_API\_ENDPOINT"

VISUAL\_RECOGNITION\_API\_KEY = "YOUR\_VISUAL\_RECOGNITION\_API\_KEY"

# OpenAI API Key

OPENAI\_API\_KEY = "YOUR\_OPENAI\_API\_KEY"

# Function to get image labels from IBM Visual Recognition

def get\_image\_labels(image\_url):

response = requests.post(

VISUAL\_RECOGNITION\_API\_ENDPOINT,

params={"version": "YYYY-MM-DD"},

headers={"apikey": VISUAL\_RECOGNITION\_API\_KEY},

files={"images\_file": open(image\_url, "rb")},

)

return response.json()

# Function to generate captions using OpenAI API

def generate\_captions(labels):

prompt = "Describe the following labels: " + ", ".join(labels)

response = openai.Completion.create(

engine="text-davinci-002",

prompt=prompt,

max\_tokens=100,

n=1,

stop=None,

apiKey=OPENAI\_API\_KEY

)

return response.choices[0].text.strip()

# Example usage

image\_url = "URL\_TO\_YOUR\_IMAGE"

labels = get\_image\_labels(image\_url)

captions = generate\_captions(labels)

print("Labels:", labels)

print("Generated Caption:", captions)