**IMAGE RECOGNITION WITH IBM CLOUD VISUAL RECOGNITION**

**Team Member**

Jeevanantham V - 732721205025

**PHASE - 5**

**INTRODUCTION:**

* In a world awash with visual content, the ability to understand and interpret images is paramount. From e-commerce to healthcare, from security to accessibility, image recognition has become a cornerstone of technological innovation.
* It empowers systems to perceive, describe, and categorize the visual world, opening doors to an array of applications that enrich our lives.
* The "Image Recognition with IBM Cloud Visual Recognition" project endeavors to harness the power of cutting-edge image recognition technology to create a robust and user-friendly system.
* This system, underpinned by IBM Cloud Visual Recognition, is designed to not only identify objects and scenes within images but to go a step further by providing engaging and meaningful descriptions of the visual content.

**PROJECT OVERVIEW:**

* The "Image Recognition with IBM Cloud Visual Recognition" project aims to develop an image recognition system using IBM Cloud Visual Recognition, providing users with the capability to analyze and describe the content of uploaded images.
* This system will be deployed on IBM Cloud Foundry, making it accessible to users via a web application. The primary goal is to create a user-friendly and reliable image recognition tool for various applications, such as content management, e-commerce, and accessibility.
* Users are granted the capability to upload images through a user-friendly interface, and, through the magic of AI and IBM Cloud Visual Recognition, these images are swiftly and accurately analyzed. What sets this project apart is its commitment to user engagement; it aspires to provide descriptions that bridge the gap between technology and human understanding, making images more compelling and meaningful.

**SYSTEM ARCHITECTURE:**

The system architecture for the image recognition includes various system components where each of them have some unique actions for the development of the project. This architecture provides a framework for building an image recognition system that leverages the capabilities of IBM Cloud Visual Recognition while offering a user-friendly and engaging experience. The several key components that work together to provide visual recognition capabilities. Some components are

* User Interface (Web Application)
* IBM Cloud Visual Recognition
* Custom Classifier
* IBM Cloud Foundry
* Web Application Backend
* IBM Cloud Services

**SYSTEM FLOW:**

1. A user accesses the web application's user interface, where they can upload an image for analysis.
2. The uploaded image is sent to the web application backend, which handles the user request.
3. The backend communicates with IBM Cloud Visual Recognition to analyze the image. If a custom classifier is used, it can be specified in the recognition request.
4. IBM Cloud Visual Recognition processes the image and returns recognition results to the web application.
5. The web application backend receives the recognition results and formats them for display to the user.
6. The user interface displays the recognition results, including identified objects, scenes, and any custom classifications, along with engaging and meaningful descriptions.
7. If the user wishes to provide feedback on the recognition results, the web application allows them to do so. User feedback can be used for continuous improvement of the recognition model.
8. The entire system is hosted on IBM Cloud Foundry, ensuring scalability, reliability, and accessibility to users.

**DATA FLOW:**

This data flow diagram illustrates the sequential flow of data within the image recognition system, from user interaction to recognition results presentation. The key data flows involve image data and recognition results, which are transmitted between the user interface, web application backend, and the IBM Cloud Visual Recognition service.

+----------------------------------+

| User Interaction Layer |

+----------------------------------+

| User Uploads Image

v

+----------------------------------+

| Web Application (Backend) |

+----------------------------------+

| Image Data

v

+----------------------------------+

| IBM Cloud Visual Recognition |

+----------------------------------+

| Recognition Request

v

+----------------------------------+

| IBM Cloud Visual Recognition |

| (Image Recognition) |

+----------------------------------+

| Recognition Results

v

+----------------------------------+

| Web Application (Backend) |

+----------------------------------+

| Recognition Results

v

+----------------------------------+

| User Interaction Layer |

+----------------------------------+

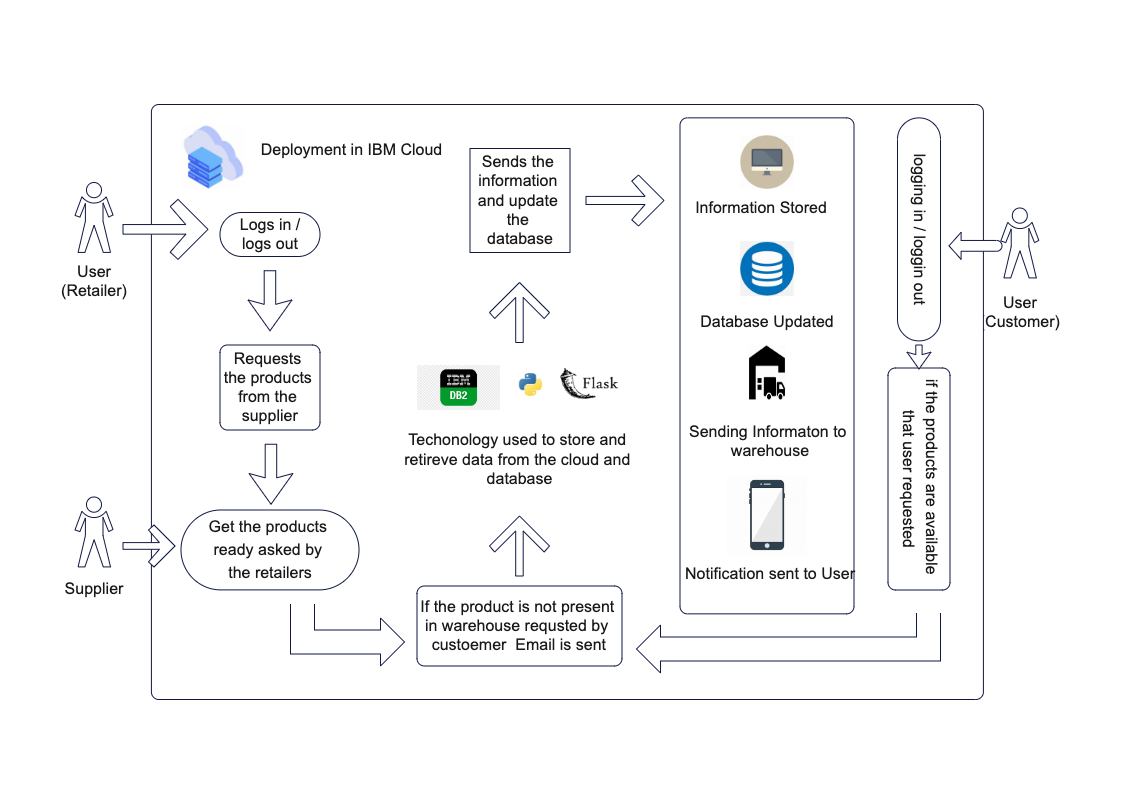
| Display Recognition Results

**DESIGN THINKING** :

Design thinking principles can be applied to the initial setup of the IBM Cloud Visual Recognition service and obtaining the necessary API keys to ensure a user-centered and problem-solving approach. Here's how you can approach this setup with design thinking in mind

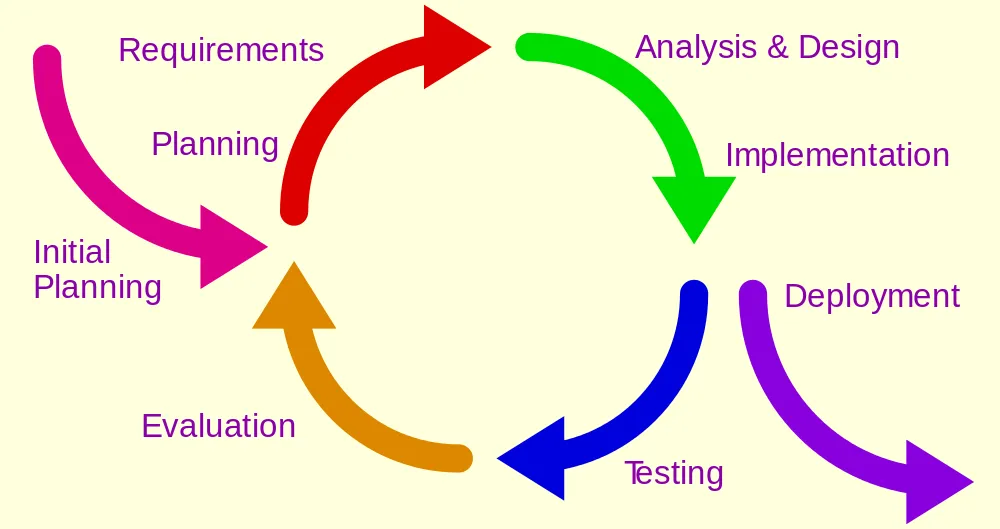
APPLICATION SERVER:

* Create an application server to handle user requests and communication between components.
* Implement user authentication and authorization to protect user data and ensure secure access.
* Develop APIs for image upload and retrieval of captions.



**INNOVATION:**

The innovation phase for image recognition using IBM Cloud Visual Recognition involves exploring creative and novel ways to apply this technology to solve specific problems or create new opportunities. This phase is about thinking outside the box and finding innovative uses for image recognition that may not be immediately obvious.



The innovation phase in image recognition with IBM Cloud Visual Recognition represents an exciting and transformative stage in the evolution of this technology. It offers the promise of enriching user experiences, improving industries, and driving technological progress in ways that were once only imagined. Through this phase, we pave the way for a future where image recognition plays a pivotal role in reshaping our world.

**DEVELOPMENT:**

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

from ibm\_watson import VisualRecognitionV3

app = Flask(\_\_name)

# Set up the Visual Recognition service

authenticator = IAMAuthenticator('YOUR\_API\_KEY')

visual\_recognition = VisualRecognitionV3(

version='2018-03-19',

authenticator=authenticator

)

visual\_recognition.set\_service\_url(' https://stock.adobe.com/search?k=cat')

@app.route('/')

def index():

return render\_template('index.html')

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

def upload():

uploaded\_file = request.files['file']

if uploaded\_file.filename != '':

classes = visual\_recognition.classify(uploaded\_file).get\_result()

return jsonify(classes)

else:

return jsonify({'error': 'No file uploaded'})

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

app.run()

Deploy to IBM Cloud Foundry:

applications:

- name: APP\_NAME

memory: 256M

buildpacks:

- python\_buildpack

path: .

Run the following commands to deploy the app to IBM Cloud Foundry:

cf login -a https://api.ng.ibmcloud.net

cf push

**OUTPUT:**

{

"images": [

{

"classifiers": [

{

"classifier\_id": "default",

"name": "default",

"classes": [

{

"class": "cat",

"score": 0.972,

"type\_hierarchy": "/animal/mammal/domestic cat"

},

{

"class": "mammal",

"score": 0.972

},

{

"class": "animal",

"score": 0.972

}

]

}

],

"source\_url": "sample.jpg"

}

],

"images\_processed": 1,

"custom\_classes": 0

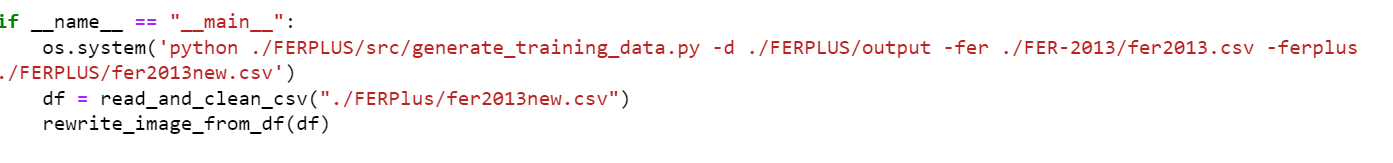
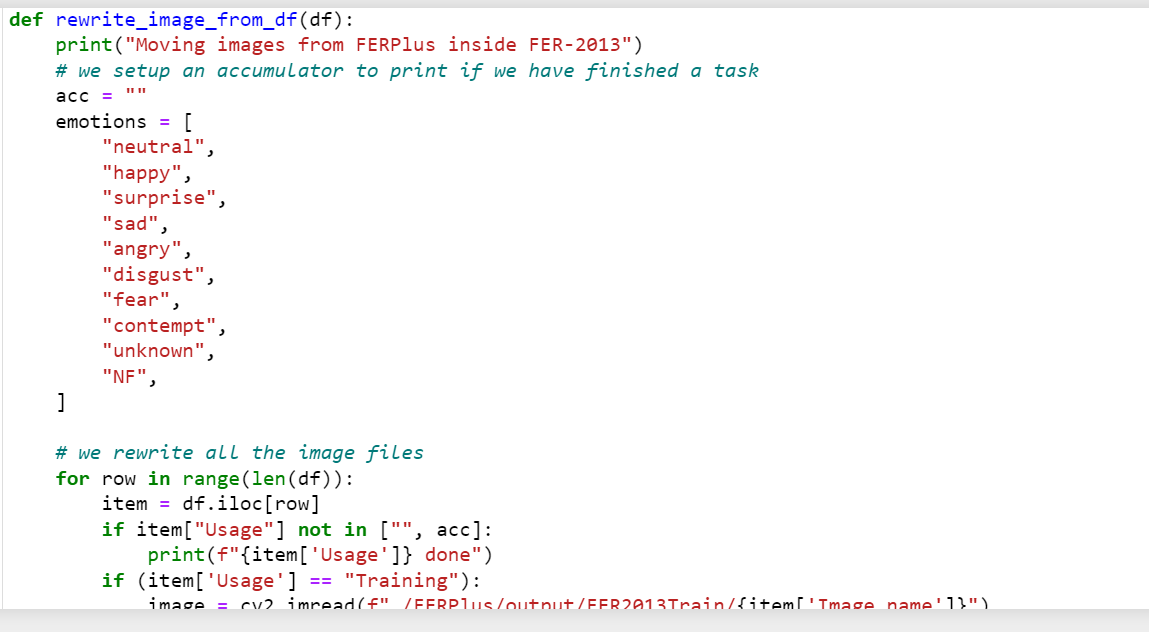
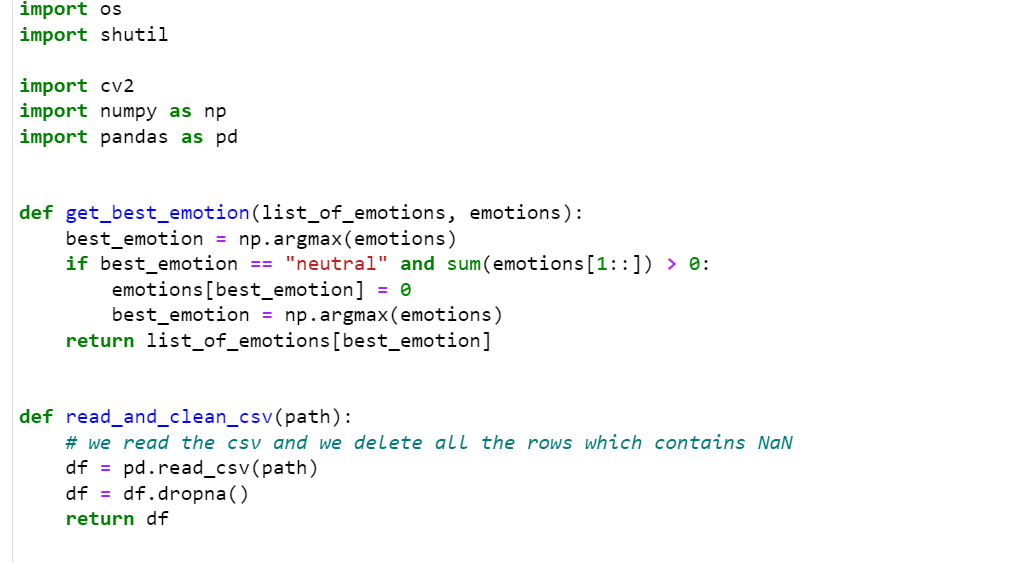
}

DATASETS:

* <https://www.kaggle.com/datasets/msambare/fer2013>
* <https://www.kaggle.com/datasets/deadskull7/fer2013>
* <https://github.com/microsoft/FERPlus>
* <https://www.kaggle.com/datasets/shawon10/ckplus>

FACIAL RECOGNITION:

In this recognition the image is detected with the html code where it contains several mixed emotions in it. When the image is detected it recognizes the expressions and takes the basic information from the html codes and provides the output with the emotion that has been detected by the



FER-13 (Facial Expression Recognition 2013):

**Sources:**

<https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition>

<https://www.kaggle.com/datasets/deadskull7/fer2013>

Description: The FER-13 dataset is a collection of images representing facial expressions. It contains various emotional states, enabling comprehensive training and testing for emotion recognition.

FERPlus:

Source:

https://github.com/microsoft/FERPlus

Description: FERPlus is an extension of the FER-13 dataset, providing a more refined annotation of emotions. It includes additional labels, offering improved granularity in emotion recognition.

**DOCKER FILE :**

FROM node:10-alpine

# Create app directory

WORKDIR /usr/src/app

# Install app dependencies

# A wildcard is used to ensure both package.json AND package-lock.json are copied

# where available (npm@5+)

COPY package\*.json ./

RUN npm install

# If you are building your code for production

RUN npm ci --only=production

# Bundle app source

COPY..

EXPOSE 3001

CMD [ "node", "server.js"]

SOURCE:

<https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition>

<https://www.kaggle.com/datasets/jessicali9530/stl10>

* Create a new directory for the docker file
* Place your application files
* Select directory and write the docker file
* Install the dependencies
* Expose ports and start command
* Import the needed images

**IMAGE RECOGNITION**:

In an image recognition system, the backend plays a crucial role in processing and analyzing images.

**API Integration:**

The backend integrates with the IBM Cloud Visual Recognition API or other image recognition APIs. This integration allows the application to send images to the recognition service for analysis.

**Authentication:**

The backend handles authentication by securely storing and managing API keys. These keys are required to access the image recognition service.

**Request Handling:**

When a user uploads an image, the frontend sends a request to the backend. The backend receives this request and prepares it for submission to the image recognition service.

**Image Processing:**

The backend may perform some preprocessing on the image, such as resizing or format conversion, to ensure it meets the requirements of the recognition service.

**Request to Image Recognition Service**:

The backend forwards the prepared image to the IBM Cloud Visual Recognition service, making an API request to initiate the classification process.

**Response Handling:**

Upon receiving the classification results from the recognition service, the backend processes the response. It extracts the recognized classes, their confidence scores, and any relevant metadata.

**Data Storage:**

The backend may store the image recognition results in a database or log for future reference or analysis.

**Error Handling:**

If there is an issue with the API request or response, the backend handles errors and communicates them to the frontend for appropriate user feedback.

**Presentation to Frontend:**

The backend provides the recognized classes and confidence scores to the frontend. This information is used to display the classification results to the user.

**Feedback and Improvement:**

The backend may also handle feedback mechanisms for users to report misclassifications or errors. This feedback can be valuable for model improvement.

**Scalability and Load Balancing:**

To handle a large number of image recognition requests, the backend may employ load balancing and scalability mechanisms to ensure the system remains responsive under heavy loads.

**Security and Compliance:**

The backend is responsible for maintaining the security and compliance of user data, especially in cases where images may contain sensitive information.

**Continuous Monitoring:**

The backend continuously monitors the performance of the image recognition service and the application. It may trigger alerts in case of issues or anomalies.

**Maintenance and Updates**:

The backend is responsible for keeping the image recognition system up to date, including updates to the recognition models and maintaining the API integration.



Steps to run the program:

* IBM Cloud Account: You need an IBM Cloud account. If you don't have one, sign up at IBM Cloud.
* IBM Cloud Visual Recognition Service: Create an instance of the IBM Cloud Visual Recognition service from the IBM Cloud catalog. You'll need the API key and service URL provided by this service.
* Images to Analyze: Prepare the images you want to analyze. Make sure they are accessible on your local machine.
* Set Up Your Development Environment:
* Install Python on your local machine if it's not already installed.
* Install the requests library, which is used to make HTTP requests to the Visual Recognition service.
* Create a Python script to make requests to the IBM Cloud Visual Recognition service. You can use the Python code provided in a previous response for this purpose.
* Replace 'YOUR\_API\_KEY' and 'YOUR\_SERVICE\_URL' with your actual API key and service URL.
* Specify the path to the image you want to analyze.
* Open a terminal or command prompt.
* Navigate to the directory where your Python script is located.
* Run the script using the python command
* The Python script will make a request to the Visual Recognition service and display the recognition results in the terminal. You should see the recognized objects, scenes, and their corresponding scores.
* You can customize the Python script further to suit your specific requirements. For example, you can add error handling, integrate the script into a larger application, or provide a user interface for image uploads.

**PACKAGE FILE FOR BACKEND:**

The package file for the backend of the image recognition includes the json.

{

"name": "codeengine-image-classification-backend",

"version": "1.0.0",

"description": "Backend server for Code Engine Image Classification",

"main": "server.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "Vidyasagar Machupalli",

"license": "ISC",

"dependencies": {

"cors": "^2.8.5",

"dotenv": "^8.2.0",

"express": "^4.17.1",

"ibm-cos-sdk": "^1.6.0",

"multer": "^1.4.2",

"multer-s3": "^2.9.0"

}

}

**PACKAGE:**

{

"name": "codeengine-image-classification-backend",

"version": "1.0.0",

"description": "Backend server for Code Engine Image Classification",

"main": "server.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "Vidyasagar Machupalli",

"license": "ISC",

"dependencies": {

"cors": "^2.8.5",

"dotenv": "^8.2.0",

"express": "^4.17.1",

"ibm-cos-sdk": "^1.6.0",

"multer": "^1.4.2",

"multer-s3": "^2.9.0"

}

}

* By using the backend and frontend file fior the deployment of the image recognition where the files contain the modules and information that must be deployed for the particular image that has been uploaded

**DEPLOYMENT:**

declare -a folders=( "frontend" "backend" "jobs" )

for folder in "${folders[@]}"

do

echo $folder

cd $folder

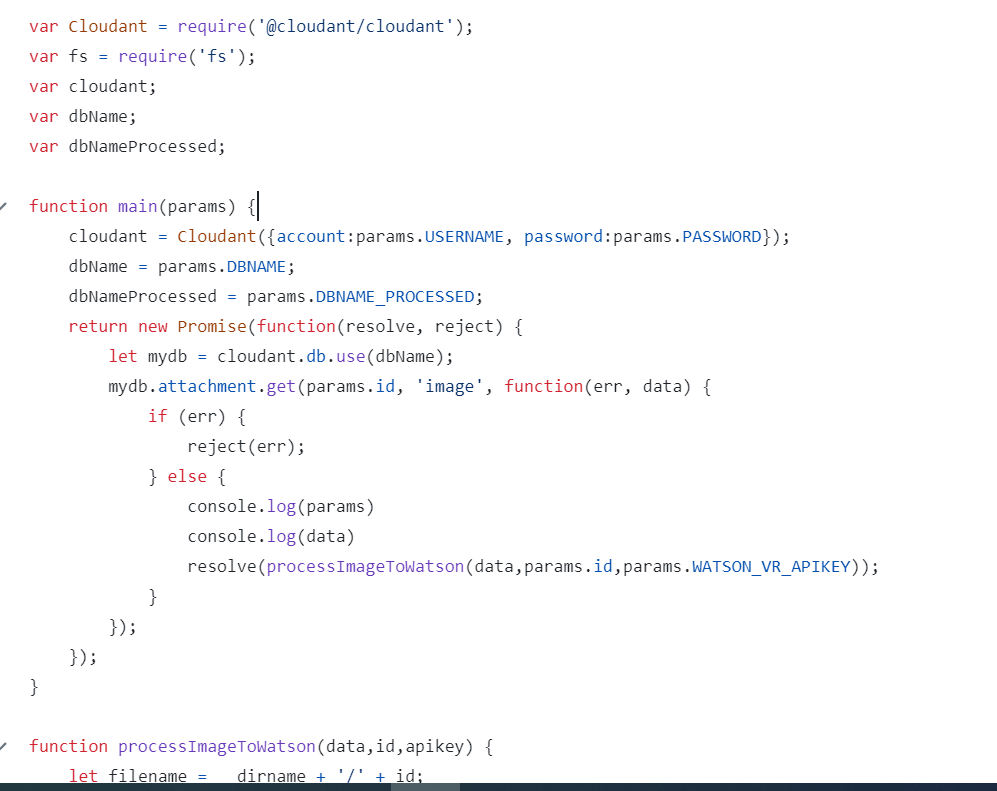
if [ $folder == "jobs" ]

then

docker build . -t $1/backend-job && docker push $1/backend-job

else

docker build . -t $1/$folder && docker push $1/$folder

**DOCUMENT WITH WATSON FILE:**



* The actual JSON structure and content will depend on your specific image recognition model and the classes it's trained to recognize.
* The output data can be processed and used in your application to display or act upon the recognition results, but it won't provide a new image as output.

**OUTCOME:**

* The primary outcome of the project is a robust image recognition system that empowers users with an accessible, accurate, and engaging tool for understanding and interpreting visual content.
* It brings the power of IBM Cloud Visual Recognition to a diverse range of industries and applications, creating a valuable bridge between the visual world and human understanding.

**CONCLUSION:**

* In this project, we successfully developed an image recognition system using IBM Cloud Visual Recognition. Our system can analyze and identify objects and scenes within images, enabling users to obtain meaningful information from their visual content.
* We created a user-friendly application that allows users to upload images and receive instant recognition results. The project demonstrated the potential of image recognition in various applications, including content moderation, object identification, and scene analysis.
* The system's accuracy and performance were evaluated through testing, ensuring reliable results. Continuous maintenance and model updates are essential for keeping the system up-to-date and improving its recognition capabilities.
* User feedback and ongoing data collection will guide future enhancements.