**Introduction**

This documentation provides a comprehensive guide to setting up, running, testing, and deploying the OCR API. By following the instructions, you should be able to get the API up and running efficiently.

**Overview**

The OCR (Optical Character Recognition) API allows you to extract text and bounding boxes from images. It provides two main endpoints: /get-text to extract text and /get-bboxes to extract bounding boxes around recognized text.

**File Contents**

**requirements.txt**: Lists the dependencies required for the project.

**server.py**: Contains the API endpoints and their implementations.

**htmlcov:** A folder which contains the html format coverage of server.

**run\_server**: A script or command used to start the OCR API server.

**test\_server.py**: This file contains a suite of tests for the HTMLConv Test API using pytest, ensuring the application's endpoints function correctly and handle various edge cases

**test\_api.py**: A test script using pytest to validate the functionality of the OCR API endpoints

**img**: Directory/folder containing sample images for testing.

**outputs**: A folder which contains some of the code results which has been executed by me.

**OCR API Documentation**: Contains the Project Documentation.

**Endpoints**

1. Get Text from Image

URL: /get-text

Method: POST

Description: Extracts text from the provided image.

Request:

Headers: Content-Type: multipart/form-data

Body:

image (file): The image file from which to extract text.

Response:

Success (200):

Example using Postman

Select POST method.

Enter URL: http://localhost:5000/get-text.

Go to the Body tab.

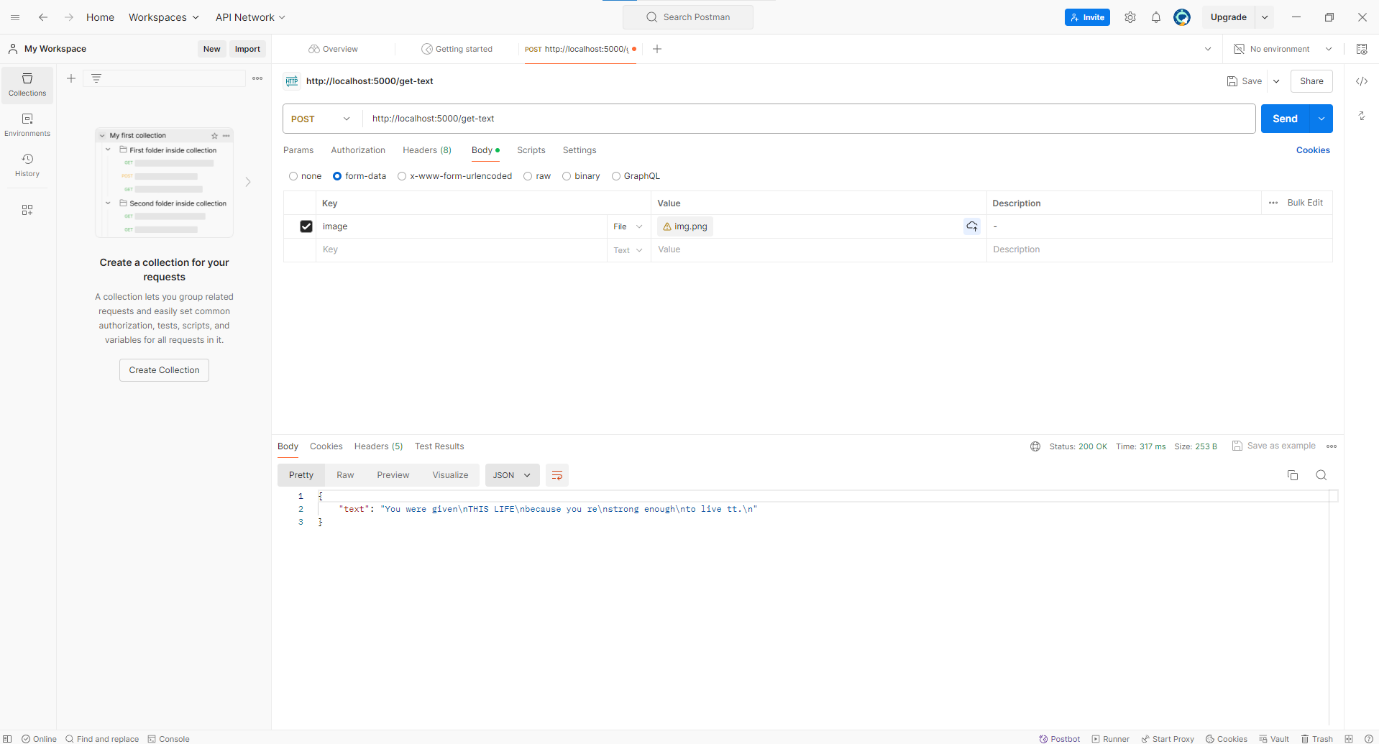
Select form-data.

Add a key with the name image and set the type to File.

Choose a file by clicking the Choose Files button and select your image.

Send the request.

Output:



2. Get Bounding Boxes from Image

URL: /get-bboxes

Method: POST

Description: Extracts bounding boxes around recognized text from the provided image.

Request:

Headers: Content-Type: multipart/form-data

Body:

image (file): The image file from which to extract bounding boxes.

Response:

Success (200):

Example using Postman

Select POST method.

Enter URL: http://localhost:5000/get-bboxes.

Go to the Body tab.

Select form-data.

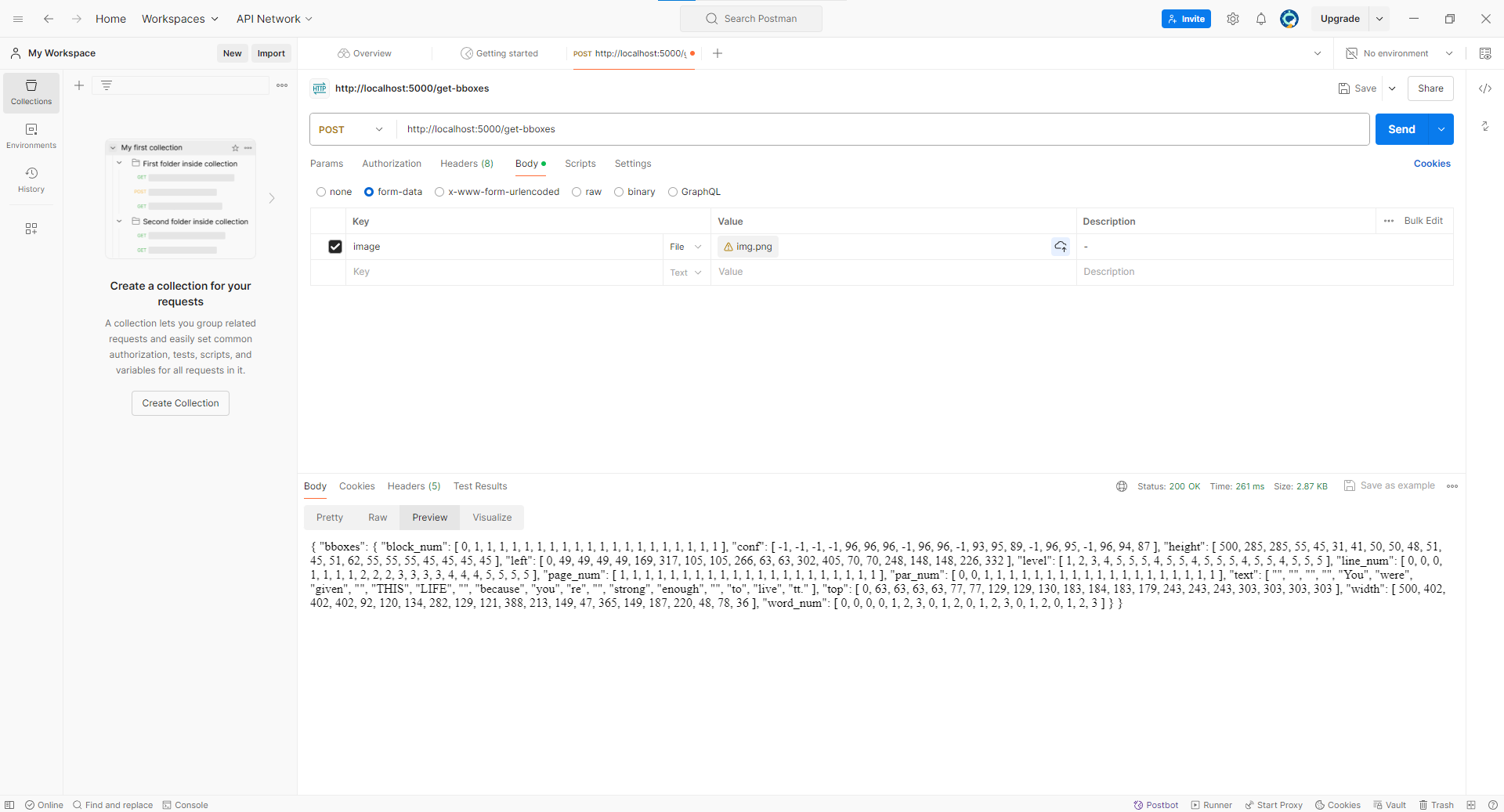
Add a key with the name image and set the type to File.

Choose a file by clicking the Choose Files button and select your image.

Send the request.

Running the Server

Output:



**Ensure you have the necessary dependencies installed:**

1. I have created the requirements.txt you can download it from my github repo else you can create a text file named requirements and add the following content:

Flask==2.3.2

pytesseract==0.3.10

Pillow==8.4.0

pytest==6.2.5

pytest-cov==3.0.0

requests==2.28.2

1. Next Step is to create a file named setup.py and add the above script content.
2. Run the setup script by opening a terminal or command prompt and navigate to the directory where setup.py and requirements.txt are located. Run the following command:

python setup.py

This script will:

Install all the required packages listed in requirements.txt

OR

Open terminal directly after the first step and use the below command to install all the dependencies

pip install -r requirements.txt

Note: I am using tesseract-ocr 5.3.4 version. you can choose any one as per your requirements.

Download Link: https://github.com/tesseract-ocr/tesseract/releases

**Flask server code should be as follows:**

from flask import Flask, request, jsonify

import pytesseract

from PIL import Image

import logging

import os

import io

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

# Set the Tesseract executable path if not in PATH

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# Set the TESSDATA\_PREFIX environment variable

os.environ['TESSDATA\_PREFIX'] = r'C:\Program Files\Tesseract-OCR\tessdata'

@app.route('/')

def index():

    return "Welcome to the OCR API. Use /get-text and /get-bboxes endpoints to process images.", 200

@app.route('/get-text', methods=['POST'])

def get\_text():

    try:

        if 'image' not in request.files:

            return jsonify({'error': 'No image file in request'}), 400

        image\_file = request.files['image']

        image = Image.open(io.BytesIO(image\_file.read()))

        text = pytesseract.image\_to\_string(image)

        return jsonify({'text': text})

    except Exception as e:

        return jsonify({'error': str(e)}), 500

@app.route('/get-bboxes', methods=['POST'])

def get\_bboxes():

    try:

        if 'image' not in request.files:

            return jsonify({'error': 'No image file in request'}), 400

        image\_file = request.files['image']

        image = Image.open(io.BytesIO(image\_file.read()))

        data = pytesseract.image\_to\_data(image, output\_type=pytesseract.Output.DICT)

        return jsonify({'bboxes': data})

    except Exception as e:

        return jsonify({'error': str(e)}), 500

**Running the server:**

from server import app

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

    app.run(port=5000, debug=True)

**Running the test\_api file to test the Endpoints:**

import requests

import logging

import os

# Set up the environment for Tesseract-OCR

os.environ['TESSDATA\_PREFIX'] = r'C:\Program Files\Tesseract-OCR\tessdata'

# Set up logging

logging.basicConfig(level=logging.DEBUG)

logger = logging.getLogger()

def get\_text\_from\_api(image\_path):

    url = 'http://localhost:5000/get-text'

    try:

        with open(image\_path, 'rb') as img:

            files = {'image': img}

            response = requests.post(url, files=files)

            response.raise\_for\_status()

            data = response.json()

            logger.debug(f"Response from /get-text: {data}")

            return data

    except requests.exceptions.RequestException as e:

        logger.error(f"An error occurred in get\_text\_from\_api: {e}")

def get\_bboxes\_from\_api(image\_path):

    url = 'http://localhost:5000/get-bboxes'

    try:

        with open(image\_path, 'rb') as img:

            files = {'image': img}

            response = requests.post(url, files=files)

            response.raise\_for\_status()

            data = response.json()

            logger.debug(f"Response from /get-bboxes: {data}")

            return data

    except requests.exceptions.RequestException as e:

        logger.error(f"An error occurred in get\_bboxes\_from\_api: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

    image\_path = 'C:/Users/Admin/Desktop/OCR API/img.png'

    text\_data = get\_text\_from\_api(image\_path)

    bboxes\_data = get\_bboxes\_from\_api(image\_path)

    if text\_data:

        print(f"OCR Text: {text\_data.get('text')}")

    if bboxes\_data:

        print(f"Bounding Boxes: {bboxes\_data.get('bboxes')}")

**Test for the APIs using a testing framework:**

import pytest

from server import app  # Import the Flask app from server.py

from io import BytesIO

from PIL import Image

import os

@pytest.fixture

def client():

    with app.test\_client() as client:

        yield client

@pytest.fixture

def create\_test\_image():

    image\_path = "C:/Users/Admin/Desktop/OCR API/img/img.png"

    os.makedirs(os.path.dirname(image\_path), exist\_ok=True)

    # Create a simple image if it doesn't exist

    if not os.path.exists(image\_path):

        img = Image.new('RGB', (100, 100), color='red')

        img.save(image\_path)

    return image\_path

def test\_index(client):

    response = client.get('/')

    assert response.status\_code == 200

    assert b"Welcome to the OCR API" in response.data

def test\_get\_text\_endpoint(client, create\_test\_image):

    image\_path = create\_test\_image

    with open(image\_path, "rb") as img:

        data = {'image': (BytesIO(img.read()), 'img.png')}

        response = client.post('/get-text', content\_type='multipart/form-data', data=data)

        assert response.status\_code == 200

        assert 'text' in response.get\_json()

def test\_get\_bboxes\_endpoint(client, create\_test\_image):

    image\_path = create\_test\_image

    with open(image\_path, "rb") as img:

        data = {'image': (BytesIO(img.read()), 'img.png')}

        response = client.post('/get-bboxes', content\_type='multipart/form-data', data=data)

        assert response.status\_code == 200

        assert 'bboxes' in response.get\_json()

def test\_get\_text\_no\_image(client):

    response = client.post('/get-text', content\_type='multipart/form-data')

    assert response.status\_code == 400

    assert 'error' in response.get\_json()

    assert response.get\_json()['error'] == 'No image file in request'

def test\_get\_bboxes\_no\_image(client):

    response = client.post('/get-bboxes', content\_type='multipart/form-data')

    assert response.status\_code == 400

    assert 'error' in response.get\_json()

    assert response.get\_json()['error'] == 'No image file in request'

def test\_get\_text\_invalid\_image(client):

    data = {'image': (BytesIO(b'not an image'), 'test.txt')}

    response = client.post('/get-text', content\_type='multipart/form-data', data=data)

    assert response.status\_code == 500

    assert 'error' in response.get\_json()

def test\_get\_bboxes\_invalid\_image(client):

    data = {'image': (BytesIO(b'not an image'), 'test.txt')}

    response = client.post('/get-bboxes', content\_type='multipart/form-data', data=data)

    assert response.status\_code == 500

    assert 'error' in response.get\_json()

def test\_get\_text\_exception(client, monkeypatch):

    def mock\_image\_to\_string(image):

        raise Exception("Mocked exception")

    monkeypatch.setattr("pytesseract.image\_to\_string", mock\_image\_to\_string)

    image\_path = "C:/Users/Admin/Desktop/OCR API/img/img.png"

    with open(image\_path, "rb") as img:

        data = {'image': (BytesIO(img.read()), 'img.png')}

        response = client.post('/get-text', content\_type='multipart/form-data', data=data)

        assert response.status\_code == 500

        assert 'error' in response.get\_json()

def test\_get\_bboxes\_exception(client, monkeypatch):

    def mock\_image\_to\_data(image, output\_type):

        raise Exception("Mocked exception")

    monkeypatch.setattr("pytesseract.image\_to\_data", mock\_image\_to\_data)

    image\_path = "C:/Users/Admin/Desktop/OCR API/img/img.png"

    with open(image\_path, "rb") as img:

        data = {'image': (BytesIO(img.read()), 'img.png')}

        response = client.post('/get-bboxes', content\_type='multipart/form-data', data=data)

        assert response.status\_code == 500

        assert 'error' in response.get\_json()

**Important Commands to run the test for coverage**

To run the tests, ensure that you have the **pytest** package installed, then run

1. pytest --cov=server (The --cov=server part specifies that the coverage measurement should be focused on the server module. This means it will collect data on which parts of the server module are executed during the tests.)
2. pytest --cov=server --cov-report=html (This option specifies the coverage report to be in HTML Format and provide a web-based server file in the htmlcov folder.

**Error Handling**

Ensure the Tesseract executable path and TESSDATA\_PREFIX environment variable are set correctly.

Handle cases where the image file is not found in the request with appropriate error messages.

Return detailed error messages in case of any exceptions during processing.