Documentation: Azure Cognitive Services OCR Integration Project

Mohammed Waseem Tabrez

801404522

Introduction

This document provides a comprehensive overview of the **Azure Cognitive Services OCR Integration Project**. The project leverages Azure's powerful **Computer Vision API** to extract text from images using Optical Character Recognition (OCR). This API can analyze images to detect and extract printed or handwritten text, making it a valuable tool for text extraction from scanned documents, street signs, business cards, and more.

Table of Contents

---

1. Overview

Project Description

The **Azure Cognitive Services OCR Integration Project** connects a Node.js application to the **Azure Computer Vision API**, specifically the OCR functionality. This API extracts text from images by leveraging machine learning and computer vision techniques. The project demonstrates how to integrate the Azure API into a Node.js application, enabling the extraction of text from images hosted on remote servers.

Key Technologies Used:

- **Node.js**: A JavaScript runtime used for building the server-side application.
- **Express.js**: A web framework for Node.js used to handle API routes and server-side logic.
- **Axios**: A promise-based HTTP client used for making requests to the Azure Cognitive Services API.
- **Swagger**: A tool for documenting the API and making it easy for developers to interact with the endpoints.

---

2. Azure Cognitive Services OCR

What is OCR?

Optical Character Recognition (OCR) is a technology used to recognize text within images. It converts different types of documents, such as scanned paper documents, PDFs, or images taken by a digital camera, into editable and searchable data. OCR can detect both printed and handwritten text in images.

Overview of Azure OCR

Azure's **Computer Vision API** provides OCR functionality, allowing users to extract text from images. It is part of the **Azure Cognitive Services** suite and is designed to be easy to use with minimal setup required.

The OCR API has the ability to analyze images asynchronously, providing both **printed text** (e.g., books, posters) and **handwritten text** recognition. Azure Cognitive Services offers multiple services, including **Image Analysis**, **Text Analytics**, and **Face Recognition**, all of which can be accessed via REST API endpoints.

- **Azure Cognitive Services OCR Features**:

- Detects printed text in images from URLs or local files.
- Supports multiple languages and character sets.
- Can process large and complex images, such as forms or documents with mixed content (images and text).

For more information on Azure OCR, refer to the [official documentation](#).

---

3. Technologies Used

Node.js

Node.js is a JavaScript runtime used to build scalable network applications. It is built on the V8 JavaScript engine, which is known for its high performance. Node.js allows us to handle asynchronous operations, which is ideal for making HTTP requests to Azure's OCR API.

Express.js

Express.js is a minimalist web framework for Node.js that simplifies routing, middleware integration, and server setup. In this project, Express handles HTTP requests and routes, enabling easy communication with the Azure OCR API.

Axios

Axios is a promise-based HTTP client that allows making asynchronous requests. In this project, Axios is used to send requests to the Azure API for OCR operations and to fetch results.

Swagger

Swagger is an open-source tool used for API documentation. It provides an interactive UI that allows developers to view and test API endpoints directly from the browser. This makes the process of understanding and interacting with the API much easier.

---

4. Project Setup

Prerequisites

Before getting started, make sure you have the following installed on your system:

- **Node.js**: [Download Node.js](#)
- **Postman**: [Download Postman](#)

Additionally, you need to have:

- **Azure Cognitive Services OCR Subscription Key** and **Endpoint URL**.

Environment Setup

1. Clone the repository:
2. git clone https://github.com/Alitabrez786123/System-Integration-Final-Project.git
3. cd System-Integration-Final-Project

4. Install dependencies:
5. npm install
6. Update the following values in your index.js file:
    - Replace YOUR_SUBSCRIPTION_KEY with your actual subscription key.
    - Replace YOUR_REGION with the appropriate region (e.g., eastus).
7. const region = 'eastus'; // Your region
8. const subscriptionKey = 'YOUR_SUBSCRIPTION_KEY'; // Your subscription key
9. const endpoint = 'https://YOUR_REGION.api.cognitive.microsoft.com/vision/v3.2/read/analyze';

---

5. Execution of the Project

Running the Server

1. Run the application using Node.js:
2. node index.js
3. The server will run on http://localhost:10000.

Testing with Postman

To test the OCR functionality with **Postman**:

1. Open Postman and set the request type to **POST**.
2. Set the request URL to http://localhost:10000/ocr.
3. In the **Body** tab, select **raw** and choose **JSON** as the content type.
4. Provide the following sample request body:
5. {
6. "imageUrl": "https://your-image-url-here"
7. }
8. Send the request and you should receive the extracted text as a response.

Swagger Documentation Access

Swagger provides an easy-to-use interface for testing the API. Once the server is running, access the Swagger documentation by visiting:

http://localhost:10000/api-docs

This page allows you to interact with the /ocr endpoint and test the OCR functionality.

---

6. Server Log Output

When the server is running and you make a successful POST request, you will see logs in the terminal indicating the extracted text. For example:

Extracted Text:

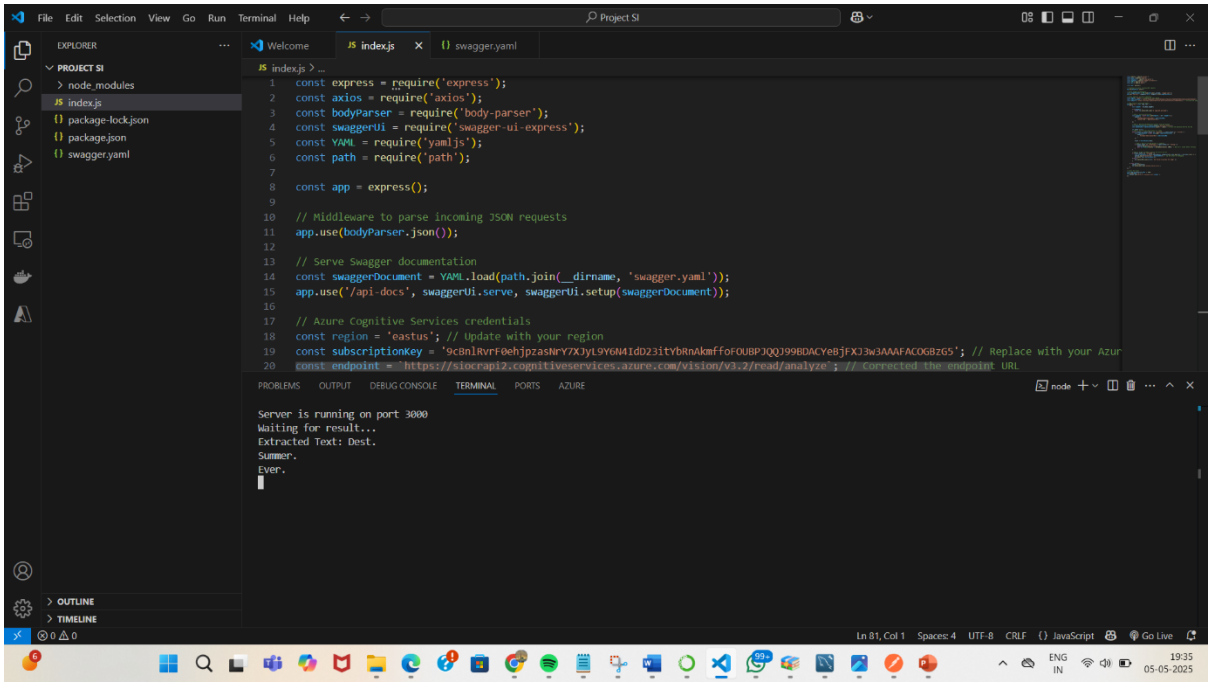
Pennsylvania

COMMERCIAL

DRIVER'S LICENSE

...

Screenshot: Server Running Status



---

7. Challenges and Solutions

Challenge 1: Handling Asynchronous OCR Results

One challenge faced was the asynchronous nature of the OCR processing. Azure's OCR API processes images in the background, and we needed to poll for results. To address this, the solution included setting up a loop to check for the succeeded status of the OCR process until it completed.

Challenge 2: Handling Invalid Responses

Another challenge was handling errors returned from the API, such as invalid or inaccessible URLs. The error handling mechanism was added to return appropriate error responses to the user.

---

8. Deployment of the API on Render

To deploy the OCR API on Render, we've chosen Render as the cloud platform for hosting our service. Render provides automatic scaling, efficient deployment, and a simple configuration process, making it an ideal choice for quickly hosting web services.

1. **Linking the GitHub Repository**

Deployment and Accessing the API

- The deployment will be live at the URL:

https://system-integration-final-project.onrender.com

- **Important:** This URL will only respond to POST requests sent via **Postman** or another API client. It is not a simple GET request URL; to get OCR results, you must send a POST request with the image URL in the body.

2. Test the API using Postman

- Open **Postman** and send a POST request to https://system-integration-final-project.onrender.com/ocr with a JSON body containing the imageUrl of the image you want to process. For example:

{

"imageUrl": "https://images.unsplash.com/photo-1494871262121-49703fd34e2b?fm=jpg&q=60&w=3000&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8Nnx8dGV4dHxlbnwwfHwwfHx8MA%3D%3D"

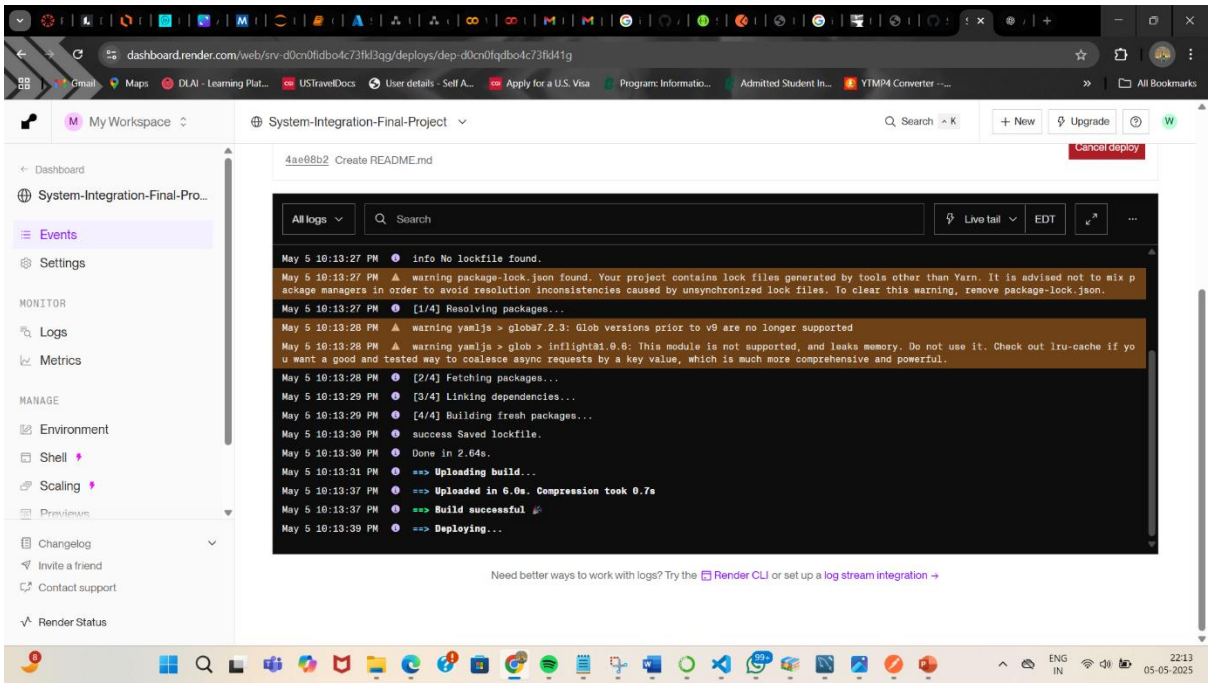
}

- When the request is sent, the API will process the image and return the text extracted from the image, if successful.

3. Monitor and Manage the Service

- Render provides live logs and monitoring tools where you can see the real-time status of your service, monitor request errors, and check deployment logs.

Render Deployment Screenshot:



---

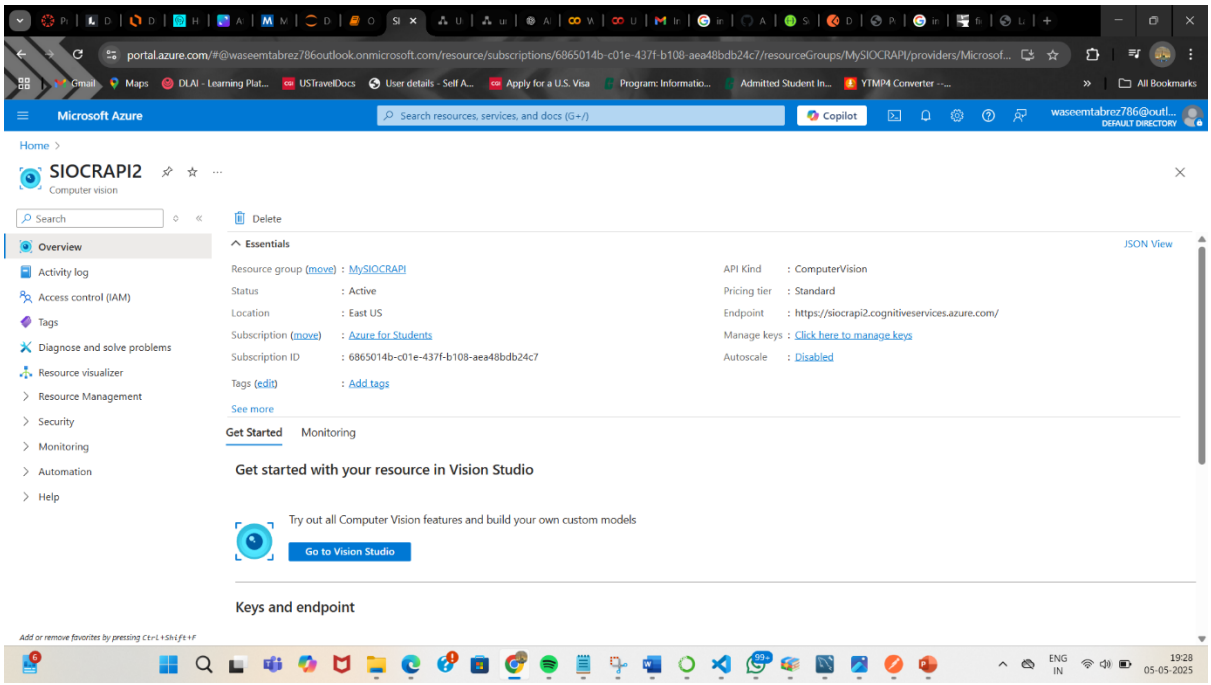
9. Conclusion

This project successfully integrates Azure's Cognitive Services OCR API with a Node.js backend application, allowing users to extract text from images using a simple API. The use of **Axios** for making requests and **Swagger** for API documentation improves the overall user experience for interacting with the OCR service.

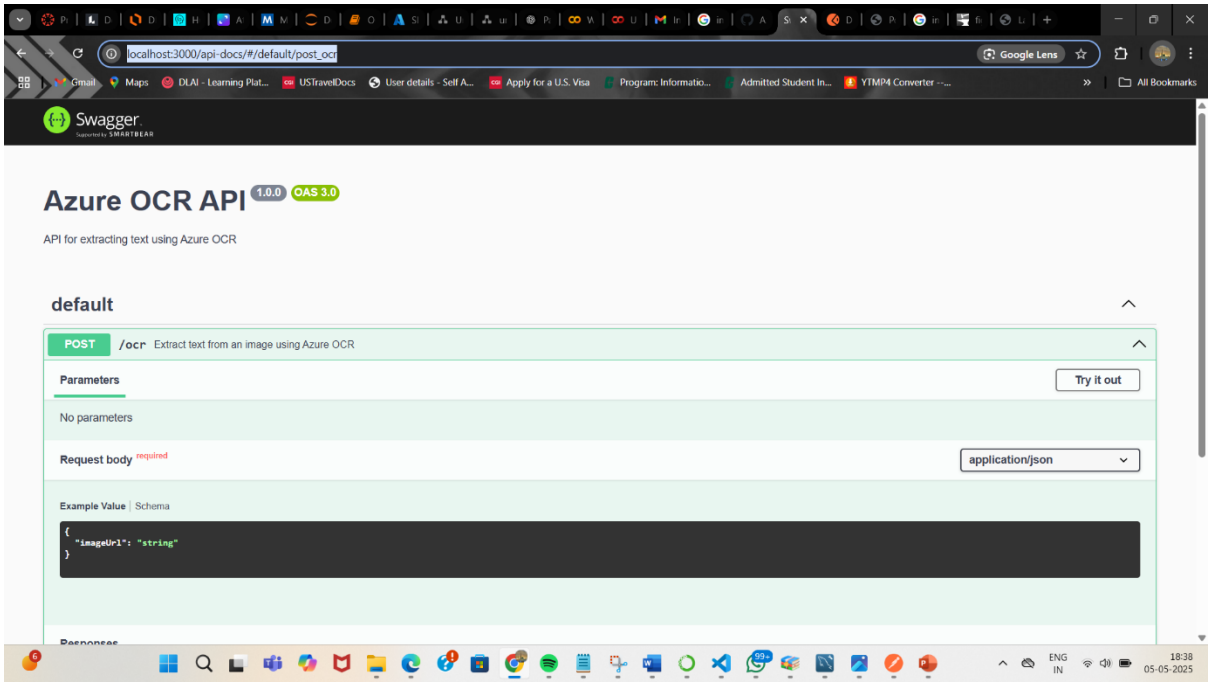
---

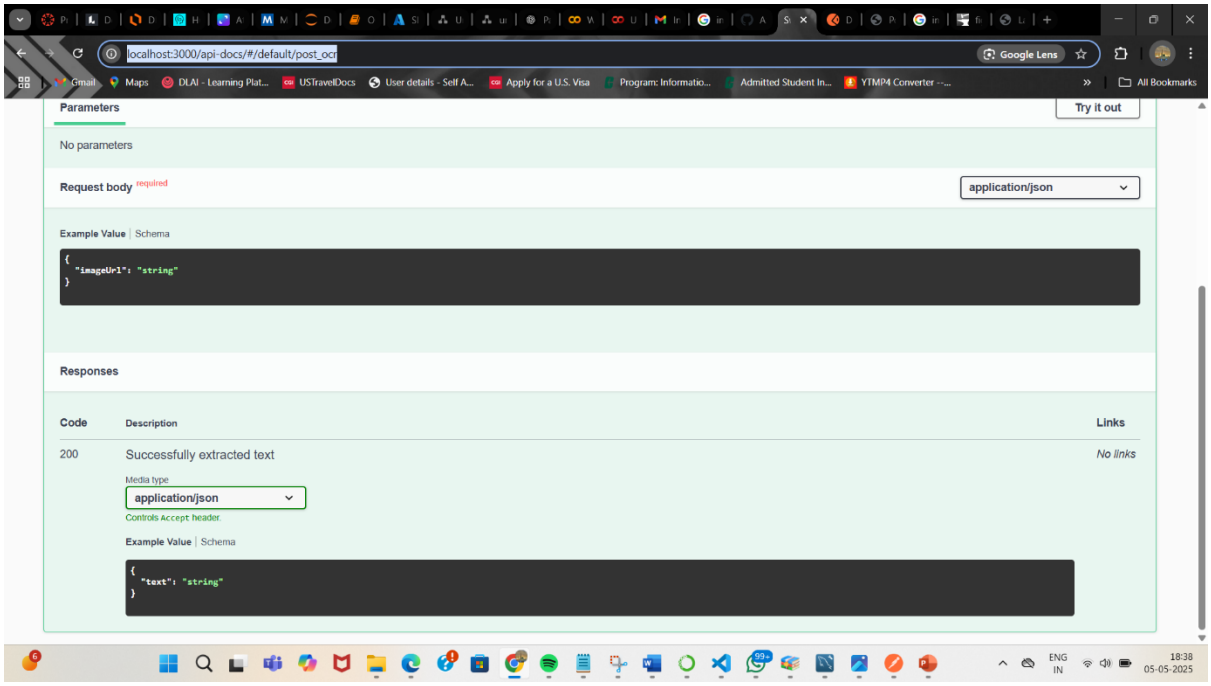
10. Screenshots and Demo

Azure OCR Computer Vision Dashboard

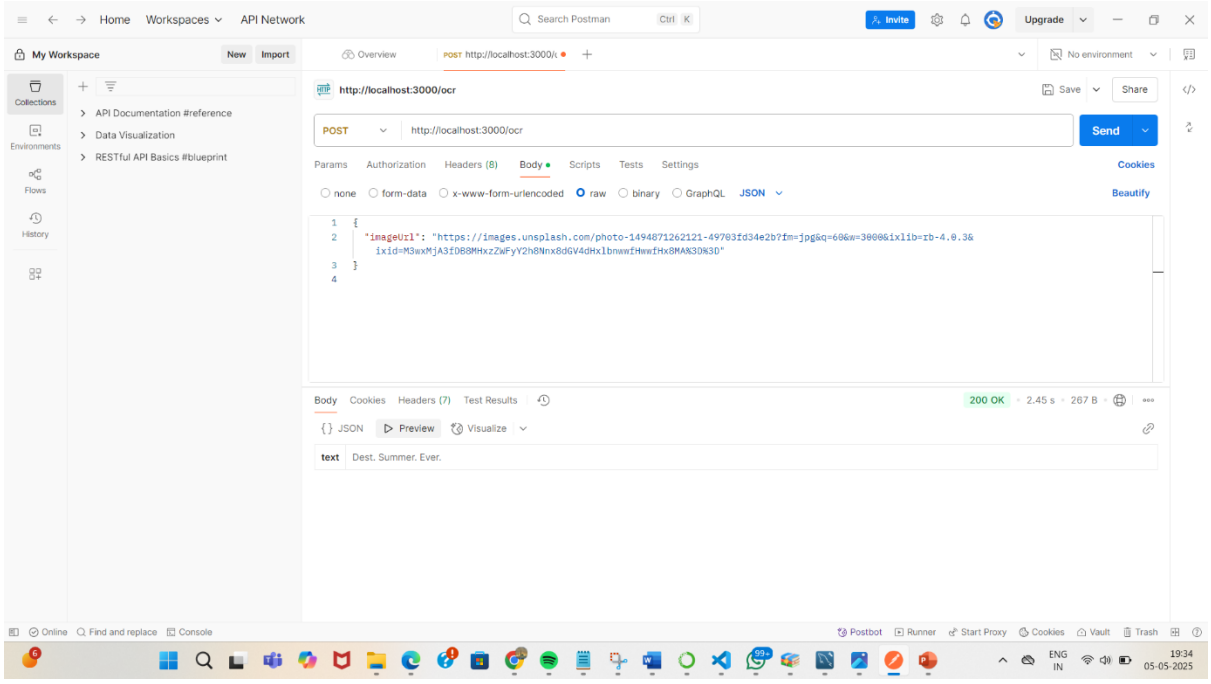


Swagger Documentation





Postman Execution



VScode Server log output