# OCR API DOCUMENTATION

## WELCOME TO THE OCR API DOCUMENTATION

This guide provides a comprehensive overview of integrating the OCR API into your Node.js application. With code snippets, detailed instructions, and best practices, you'll have everything you need to extract text from images with precision and efficiency.

## OVERVIEW

The OCR API enables developers to extract printed and handwritten text from images. It supports multi-language text recognition, provides bounding box data for layout analysis, and is optimized for real-time processing.

## FEATURES

- Multi-language support: Recognize text in multiple languages.
- Bounding boxes: Get coordinates of detected text for layout and analysis purposes.
- Wide Image Format Compatibility: Works with PNG, JPEG, BMP, and more.
- Asynchronous Processing: Enables large-scale document handling.

## WORKFLOW DIAGRAM

## HOW IT WORKS?

- The client provides an image URL via the API.
- Azure's OCR engine processes the image and detects text.
- Text is extracted and returned in JSON format, including layout information like bounding boxes.

## SETUP GUIDE

### 1. Prerequisites

- (Optional) Azure subscription for advanced settings and scaling.
- Node.js installed on your system.
- A Postman client or similar tool for testing API calls.

### 2. Environment Setup

Install required dependencies:

> npm install express axios dotenv

Create a `.env` file in your project directory and add the following:

AZURE_VISION_KEY=your-azure-api-key

AZURE_VISION_ENDPOINT=https://your-resource-name.cognitiveservices.azure.com

AZURE_VISION_LOCATION=your-location

### 3. Request Specifications

POST Method - /read-image

URL Endpoint –http://134.209.45.33:3000/read-image

Request body –

| Parameter | Type | Description | Required |
|---|---|---|---|
| image_url | String | URL of the image to process | Yes |

{

"image_url": "https://drive.google.com/uc?id=14x5Q2Ew01Rtsk0qZT-ugU07KCdcr8VHS"

}

Response –

| Field | Type | Description |
|---|---|---|
| page | Integer | Page number of the processed image. |
| lines.text | String | Extracted text. |
| boundingBox | Array | Coordinates of the text area in the image. |


```
[
	{
		"page": 1,
		"lines": [
	{
		"text": "Sample text",
		"boundingBox": [42, 55, 230, 55, 230, 75, 42, 75]
	}
		]
	}
]
```


## 4. Node.js Implementation

```
//Import libraries
const express = require("express");
const bodyParser = require("body-parser");
const axios = require("axios");
require("dotenv").config();

const app = express();
const port = 3000;

app.use(bodyParser.json());
```

```
const endpoint = process.env.AZURE_VISION_ENDPOINT;
const apiKey = process.env.AZURE_VISION_KEY;
const location = process.env.AZURE_VISION_LOCATION;

// OCR API calls
app.post("/read-image", async (req, res) => {
  const { image_url } = req.body;

  if (!image_url) {
    return res
      .status(400)
      .json({ error: "image_url is required in the request body." });
  }

  try {
    const analyzeResponse = await axios.post(
      `${endpoint}/vision/v3.2/read/analyze`,
      { url: image_url },
      {
        headers: {
          "Ocp-Apim-Subscription-Key": apiKey,
          "Ocp-Apim-Subscription-Region": location,
          "Content-Type": "application/json",
        },
      }
    );

    const operationLocation = analyzeResponse.headers["operation-location"];
```

```
  let result;
  for (let i = 0; i < 10; i++) {
    await new Promise((resolve) => setTimeout(resolve, 2000));
    const operationResult = await axios.get(operationLocation, {
      headers: {
        "Ocp-Apim-Subscription-Key": apiKey,
      },
    });
    if (operationResult.data.status === "succeeded") {
      result = operationResult.data;
      break;
    } else if (operationResult.data.status === "failed") {
      return res.status(500).json({ error: "Failure of OCR processing." });
    }
  }

  if (!result) {
    return res.status(408).json({ error: "Timed out while OCR processing." });
  }

  res.status(200).json(result.analyzeResult.readResults);
} catch (error) {
  console.error("Error:", error.message);
  res.status(error.response?.status || 500).json({
    error:
      error.response?.data || "Image could not be processed.",
  });
```

```
  }
});


// Start server
app.listen(port, () => {
  console.log(`Server running at http://134.209.45.33:${port}`);
});
```

## TESTING VIA POSTMAN

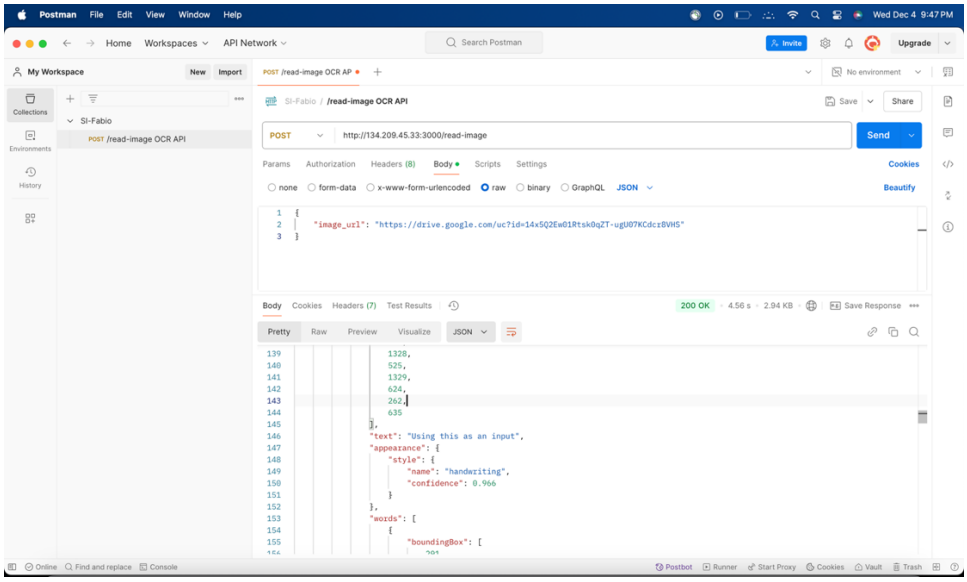
1. Open Postman and create a new request

2. Set the method to POST and the URL as below:
   http://134.209.45.33:3000/read-image

3. Add the following header:
   Content-Type: application/json

4. In the Body tab, choose `raw` and paste the following JSON:
   {
   "image_url": https://drive.google.com/uc?id=14x5Q2Ew01Rtsk0qZT-ugU07KCdcr8VHS
   }

5. Hit send and observe the response



## ERROR HANDLING

- **400**: Ensure that the imageUrl is correct and publicly accessible.
- **408:** This happens when processing takes too long. Retry with a smaller image or after some time.
- **500:** Likely an issue with server configuration. Contact support or review the deployment.

## FAQs

- Q: What image formats are supported?
  A: The API supports JPEG, PNG, and BMP formats.

- Q: How do I handle large images?
  A: Resize or compress the image to ensure efficient processing.

- Q: Is my image data secure?
  A: Yes, data is transmitted securely via HTTPS when hosted on production.

## RESOURCES

- Azure OCR - Optical Character Recognition API Documentation
- Node.js Axios Guide