* Defines an /api route.
* Reads JSON data from a backend file (let’s call it data.json).
* Returns the data as a JSON response.

### Step 1: Create your data.json file

Create a file called data.json in the same directory as your Flask app. Example content:

json

CopyEdit

[

{"id": 1, "name": "Alice"},

{"id": 2, "name": "Bob"},

{"id": 3, "name": "Charlie"}

]

### Step 2: Create your Flask app (app.py)

python

CopyEdit

from flask import Flask, jsonify

import json

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

@app.route('/api')

def get\_data():

try:

with open('data.json', 'r') as f:

data = json.load(f)

return jsonify(data)

except Exception as e:

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

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

app.run(debug=True)

### Explanation:

* When /api is accessed, the app reads data.json.
* It sends the JSON data back in the response.
* If there's an error reading the file, it returns an error JSON with a 500 status.

### How to run:

1. Make sure you have Flask installed:

bash

CopyEdit

pip install flask

1. Run the app:

bash

CopyEdit

python app.py

1. Access http://localhost:5000/api in your browser or with curl/postman to see the JSON list.

### Prerequisites:

1. You need a MongoDB Atlas cluster with a database and collection ready.
2. Install necessary packages:

bash

CopyEdit

pip install flask pymongo dnspython

### Step 1: Flask app (app.py)

python

CopyEdit

from flask import Flask, render\_template, request, redirect, url\_for

from pymongo import MongoClient

import os

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

# MongoDB Atlas connection string

MONGO\_URI = os.getenv('MONGO\_URI') # Set your connection string in environment variable

client = MongoClient(MONGO\_URI)

db = client['your\_database\_name']

collection = db['your\_collection\_name']

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

def form():

error = None

if request.method == 'POST':

# Example fields from form

name = request.form.get('name')

email = request.form.get('email')

if not name or not email:

error = "Both name and email are required."

else:

try:

doc = {"name": name, "email": email}

collection.insert\_one(doc)

return redirect(url\_for('success'))

except Exception as e:

error = f"An error occurred: {str(e)}"

return render\_template('form.html', error=error)

@app.route('/success')

def success():

return "Data submitted successfully"

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

app.run(debug=True)

### Step 2: Create a folder named templates in the same directory as app.py, and inside it create form.html:

html

CopyEdit

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Submit Data</title>

</head>

<body>

<h1>Submit Data to MongoDB Atlas</h1>

{% if error %}

<p style="color: red;">{{ error }}</p>

{% endif %}

<form method="POST">

<label>Name: <input type="text" name="name" /></label><br /><br />

<label>Email: <input type="email" name="email" /></label><br /><br />

<button type="submit">Submit</button>

</form>

</body>

</html>

### Step 3: Set your MongoDB URI environment variable

Replace the placeholder values with your actual MongoDB URI:

bash

CopyEdit

export MONGO\_URI="mongodb+srv://<username>:<password>@cluster0.mongodb.net/?retryWrites=true&w=majority"

On Windows CMD:

cmd

CopyEdit

set MONGO\_URI="mongodb+srv://<username>:<password>@cluster0.mongodb.net/?retryWrites=true&w=majority"

### How this works:

* User accesses / and sees the form.
* On submit, Flask tries to insert the data into MongoDB Atlas.
* On success, redirect to /success which displays a simple message.
* On failure or missing data, it redisplays the form with an error.