1. Create a Flask application with an /api route. When this route is accessed, it should return a JSON list. The data should be stored in a backend file, read from it, and sent as a response.
2. from json import loads
3. from dotenv import load\_dotenv
4. import requests, datetime
5. import os
6. import sys
7. from pymongo import MongoClient
8. from flask import Flask, json, jsonify, request, render\_template # Importing flask module in the project is mandatory
9. # Flask is a web framework for building web applications in Python.
10. #render\_template is used to render HTML files
11. # jsonify is used to convert data to JSON format
12. # request is used to handle request data
13. load\_dotenv()  # Load environment variables from .env file
14. app = Flask(\_\_name\_\_)   # Creating the instance of Flask app
15. port = int(os.environ.get('PORT', 5010))
16. # MongoDB connection setup
17. mongo\_uri = os.getenv('MONGO\_URI')
18. mongo\_uri = "mongodb+srv://sunilsiddharth\_db\_user:WWKk7zwPkasZYYDd@mycluster.skvw5ew.mongodb.net/?retryWrites=true&w=majority&appName=MyCluster"
19. if not mongo\_uri:
20. print("Error: MONGO\_URI environment variable not set.")
21. sys.exit(1)
22. client = MongoClient(mongo\_uri)
23. db = client.get\_database('Sunil\_Test')  # Replace with your database name
24. users\_collection = db.get\_collection('myCollection')  # Replace with your collection name
25. # If db and collection do not exist, MongoDB will create them when you insert the first document.
26. @app.route("/") # route to display the home page
27. def home(): # function to display the home page
28. #return "Hello, this is a Flask Microservice"
29. return render\_template("SignUpForm.html") # to render HTML file
30. @app.route("/signup", methods=["POST"])  # route with allowed methods as POST
31. def signup(): # function to handle the sign-up form submission
32. # Extract form data
33. username = request.form.get('username')
34. email = request.form.get('email')
35. password = request.form.get('password')
36. # Insert user data into MongoDB
37. user\_data = {
38. 'username': username,
39. 'email': email,
40. 'password': password,  # In a real application, never store plain passwords
41. 'signup\_date': datetime.datetime.utcnow()
42. }
44. users\_collection.insert\_one(user\_data)
45. # Here, you would typically save the user data to a database
46. # For demonstration, we'll just return a success message
48. return jsonify({
49. 'message': 'User signed up successfully!',
50. 'username': username,
51. 'email': email
52. }), 201 # 201 Created status code
53. @app.route("/api", methods=["GET"])  # route to get all users
54. def api(): # function to fetch all users
55. users = list(users\_collection.find({}, {'\_id': 0, 'password': 0, 'signup\_date': 0}))  # Exclude \_id and password fields
57. # Write the users list to a temporary file and read it back
58. # This step is generally unnecessary but included as per the original code's logic
59. file=open('temp.txt','w')
60. file.write(str(users))
61. file.close()
62. file=open('temp.txt','r')
63. users=file.read()
64. users = users.replace("\'", "\"")  # Replace single quotes with double quotes for valid JSON
65. file.close()
66. python\_object = json.loads(users)
67. return jsonify(python\_object), 200 # 200 OK status code
68. if \_\_name\_\_ == "\_\_main\_\_": # on running python app.py
69. app.run(debug=True, host="0.0.0.0", port=port) # run the flask app on the given port number

2. Create a form on the frontend that, when submitted, inserts data into MongoDB Atlas. Upon successful submission, the user should be redirected to another page displaying the message **"Data submitted successfully"**. If there's an error during submission, display the error on the same page without redirection.

from json import loads

from dotenv import load\_dotenv

import requests, datetime

import os

import sys

from pymongo import MongoClient

from flask import Flask, json, jsonify, request, render\_template # Importing flask module in the project is mandatory

# Flask is a web framework for building web applications in Python.

#render\_template is used to render HTML files

# jsonify is used to convert data to JSON format

# request is used to handle request data

load\_dotenv()  # Load environment variables from .env file

app = Flask(\_\_name\_\_)   # Creating the instance of Flask app

port = int(os.environ.get('PORT', 5010))

# MongoDB connection setup

mongo\_uri = os.getenv('MONGO\_URI')

mongo\_uri = "mongodb+srv://sunilsiddharth\_db\_user:WWKk7zwPkasZYYDd@mycluster.skvw5ew.mongodb.net/?retryWrites=true&w=majority&appName=MyCluster"

if not mongo\_uri:

    print("Error: MONGO\_URI environment variable not set.")

    sys.exit(1)

client = MongoClient(mongo\_uri)

db = client.get\_database('Sunil\_Test')  # Replace with your database name

users\_collection = db.get\_collection('myCollection')  # Replace with your collection name

# If db and collection do not exist, MongoDB will create them when you insert the first document.

@app.route("/") # route to display the home page

def home(): # function to display the home page

    #return "Hello, this is a Flask Microservice"

    return render\_template("SignUpForm.html") # to render HTML file

@app.route("/signup", methods=["POST"])  # route with allowed methods as POST

def signup(): # function to handle the sign-up form submission

    # Extract form data

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

    email = request.form.get('email')

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

    # Insert user data into MongoDB

    user\_data = {

        'username': username,

        'email': email,

        'password': password,  # In a real application, never store plain passwords

        'signup\_date': datetime.datetime.utcnow()

    }

    users\_collection.insert\_one(user\_data)

    # Here, you would typically save the user data to a database

    # For demonstration, we'll just return a success message

    return jsonify({

        'message': 'Data Submitted successfully',

        'username': username,

        'email': email

    }), 201 # 201 Created status code

if \_\_name\_\_ == "\_\_main\_\_": # on running python app.py

    app.run(debug=True, host="0.0.0.0", port=port) # run the flask app on the given port number