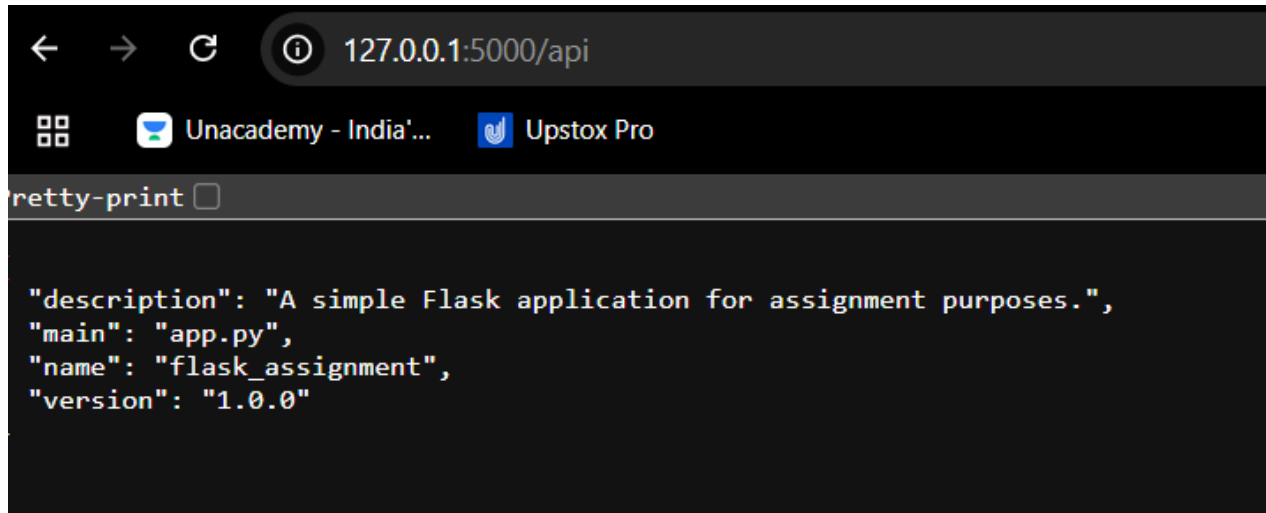


1. Create a Flask application with an `/api` route. When this route is accessed, it should return a JSON list. The data should be stored in a backend file, read from it, and sent as a response.

```
app.py × text.json
flask assignment > app.py > ...
1  from flask import Flask, jsonify
2  import json
3
4  app = Flask(__name__)
5
6  @app.route('/api')
7  def home():
8
9      data = json.load(open('text.json'))
10     return jsonify(data)
11
12 if __name__ == '__main__':
13     app.run(debug=True)
14
15
```

```
app.py text.json ×
flask assignment > text.json > ...
1  {
2      "name": "flask_assignment",
3      "version": "1.0.0",
4      "description": "A simple Flask application for assignment purposes.",
5      "main": "app.py"
6  }
```



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:5000/api`. Below the address bar, there are tabs for "Unacademy - India..." and "Upstox Pro". The main content area displays a JSON response from a REST client, with the text `pretty-print` and a checkbox. The JSON data is as follows:

```
"description": "A simple Flask application for assignment purposes.",
"main": "app.py",
"name": "flask_assignment",
"version": "1.0.0"
```

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.



A screenshot of a code editor showing Python code for a Flask application. The code is as follows:

```
flask assignment > 2 > frontend > app.py > ...
1  from flask import Flask, render_template, request
2  import requests
3
4  BACKEND_URL = 'http://localhost:6000'
5
6  app = Flask(__name__)
7
8  @app.route('/')
9  def home():
10     return render_template('index.html')
11
12  @app.route('/submit', methods=['POST'])
13  def submit():
14     form_data = dict(request.form)
15     requests.post(BACKEND_URL + '/submit', json=form_data)
16     return "Data Submitted Successfully"
17
18  if __name__ == '__main__':
19     app.run(host = '0.0.0.0', port = 5000, debug=True)
```

```

flask assignment > 2 > backend > app.py > ...
1  from flask import Flask, request
2  from pymongo.mongo_client import MongoClient
3  from pymongo.server_api import ServerApi
4  from dotenv import load_dotenv
5  import os
6
7  load_dotenv()
8  MONGO_URI = os.getenv('MONGO_URI') # Load the MongoDB URI from the .env file
9
10 uri = MONGO_URI # Ensure you have the correct MongoDB URI in your .env file
11
12 client = MongoClient(uri, server_api=ServerApi('1'))
13
14 db = client.test
15 collection = db['flask_assignment']
16
17 app = Flask(__name__)
18
19 @app.route('/submit', methods=['POST'])
20 def submit():
21     form_data = request.get_json()
22     collection.insert_one(form_data)
23
24     return "Data Submitted Successfully"
25
26 if __name__ == '__main__':
27     app.run(host = '0.0.0.0', port = 6000, debug=True)
28

```

## Calorie Calculator

**Age**

**Gender**

Select
▼

**Weight (kg)**

**Height (cm)**

**Activity Level**

Select
▼

Submit

## Data Submitted Successfully

The screenshot displays the ClusterO MongoDB management console. The interface includes a top navigation bar with the ClusterO logo, version (8.0.11), and region (AWS Mumbai (ap-south-1)). Below this is a secondary navigation bar with tabs for Overview, Real Time, Metrics, Collections (selected), Atlas Search, Query Insights, Performance Advisor, Online Archive, Cmd Line Tools, and Info. The main content area shows the 'test.flask\_assignment' collection. On the left, a sidebar lists the database 'test' and the collection 'flask\_assignment'. The main panel displays collection statistics: STORAGE SIZE: 36KB, LOGICAL DATA SIZE: 217B, TOTAL DOCUMENTS: 2, and INDEXES TOTAL SIZE: 36KB. It also provides tabs for Find (selected), Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A search bar with the placeholder 'Type a query: { field: 'value' }' is present, along with 'Reset', 'Apply', and 'Options' buttons. Below the search bar, two document entries are visible, each with fields like '\_id', 'age', 'gender', 'weight', 'height', and 'activity'. A chat bubble icon is located in the bottom right corner.

ClusterO

VERSION: 8.0.11 REGION: AWS Mumbai (ap-south-1)

Overview Real Time Metrics Collections Atlas Search Query Insights Performance Advisor Online Archive Cmd Line Tools Info

DATABASES: 1 COLLECTIONS: 1

+ Create Database

Search Namespaces

test

flask\_assignment

test.flask\_assignment

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 217B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Generate queries from natural language in Compass

INSERT DOCUMENT

Filter Type a query: { field: 'value' } Reset Apply Options

```
{
  "_id": ObjectId('6883fc32739e116786bee41e'),
  "age": "26",
  "gender": "male",
  "weight": "70",
  "height": "175",
  "activity": "Super Active"
}
```

```
{
  "_id": ObjectId('6883fd17739e116786bee41f'),
  "age": "25",
  "gender": "male",
  "weight": "65"
}
```

**Submission Guidelines -:** Attach Screenshots or command along with explanation and submit in doc(google doc or microsoft doc) format also attach github repo link