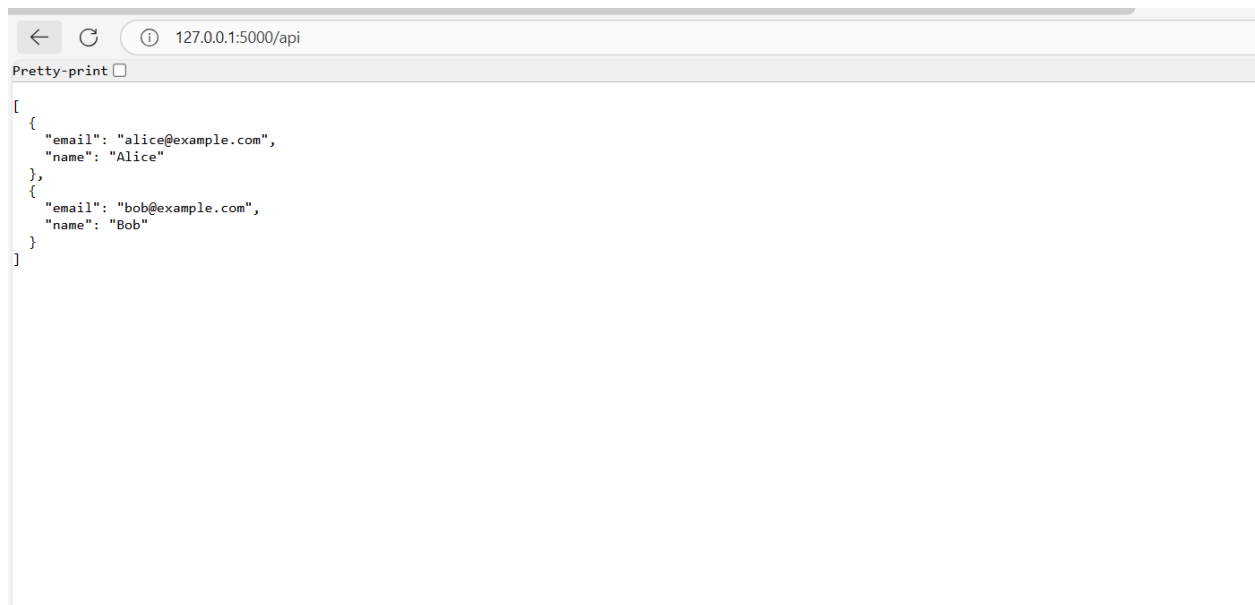


Assignment 3

Outputs

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.



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:5000/api`. Below the address bar, there is a "Pretty-print" checkbox. The main content area of the browser displays a JSON array containing two objects, each with "email" and "name" fields. The first object represents Alice and the second represents Bob.

```
[
  {
    "email": "alice@example.com",
    "name": "Alice"
  },
  {
    "email": "bob@example.com",
    "name": "Bob"
  }
]
```

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.

Data submitted successfully

[Go back](#)

With Error

Submit Data

bad auth : authentication failed, full error: {'ok': 0, 'errmsg': 'bad auth : authentication failed', 'code': 8000, 'codeName': 'AtlasError'}

Name:

Email:

Process

1. Set up MongoDB database
2. Create [app.py](#), form.html, success.html files and update Mongo DB credential in [app.py](#)
3. Create a Conda environment then install flask and mongo
4. Run - python [app.py](#)
5. Access the UI on local server - <http://127.0.0.1:5000/>
6. Access Json file details on - <http://127.0.0.1:5000/api>
7. Enter details like First name and Email address and hit submit then verify in mongodb database
8. Modify the password to replicate error scenario

Set Up MongoDB Atlas

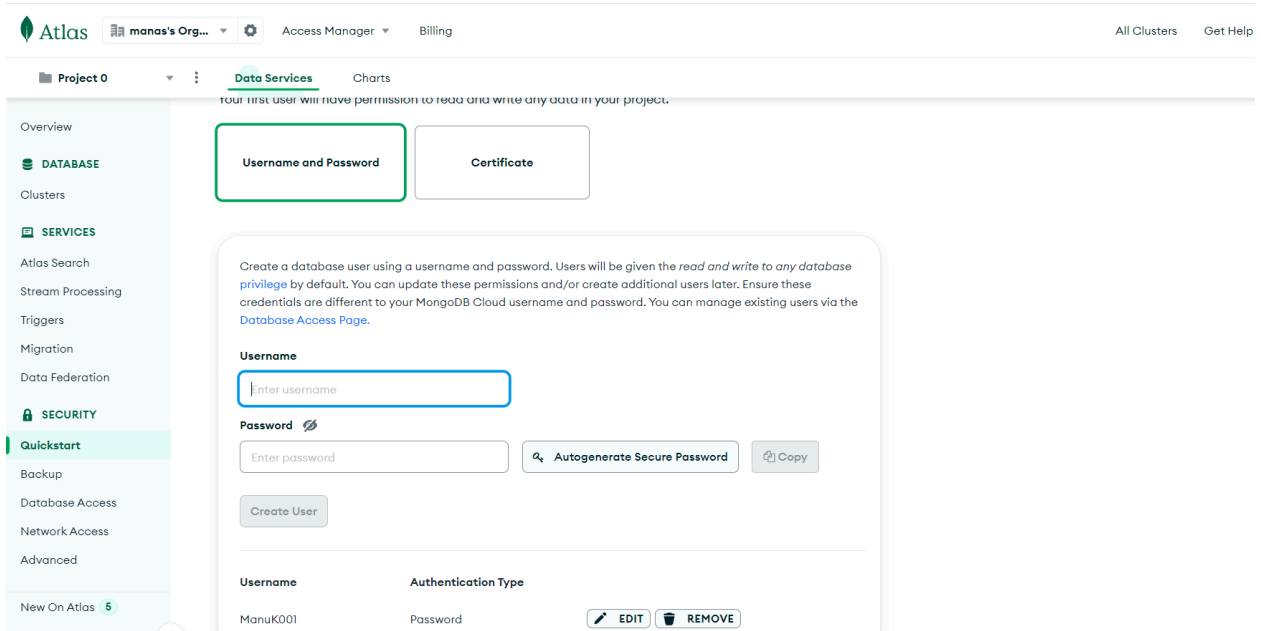
Step 1

1. Go to [MongoDB Atlas](#) and create a free account.
2. Create a new Cluster (use the free tier).
3. Click Database Access, add a new user with Read/Write privileges.
4. Go to Network Access, add your current IP address or allow access from anywhere (0.0.0.0/0) for testing purposes.

In Clusters, click Connect → Connect your application, copy the connection string (e.g.):

`mongodb+srv://<username>:<password>@cluster0.mongodb.net/?retryWrites=true&w=majority`

5. Replace `<username>` and `<password>` with your actual credentials.



Step 2: Install **pymongo** in Your Conda Environment

conda activate your-env-name

pip install pymongo

pip install flask

```
(base) C:\Users\mmoha>conda env list

# conda environments:
#
base                * C:\Users\mmoha\miniconda3
testflask           C:\Users\mmoha\miniconda3\envs\testflask
tfmodels            C:\Users\mmoha\miniconda3\envs\tfmodels

(base) C:\Users\mmoha>conda activate testflask

(testflask) C:\Users\mmoha>cd C:\Users\mmoha\OneDrive\Documents\DevopsWork\Assignment 3

(testflask) C:\Users\mmoha\OneDrive\Documents\DevopsWork\Assignment 3>python3 app.py
Traceback (most recent call last):
  File "C:\Users\mmoha\OneDrive\Documents\DevopsWork\Assignment 3\app.py", line 1, in <module>
    from flask import Flask, jsonify, request, render_template
ModuleNotFoundError: No module named 'flask'

(testflask) C:\Users\mmoha\OneDrive\Documents\DevopsWork\Assignment 3>pip install flask
Collecting flask
  Downloading flask-3.1.1-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.9.0 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.2.1-py3-none-any.whl.metadata (2.5 kB)
Collecting itsdangerous>=2.2.0 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting markupsafe>=2.1.1 (from flask)
  Downloading MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl.metadata (4.1 kB)
Collecting werkzeug>=3.1.0 (from flask)
  Using cached werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Collecting colorama (from click>=8.1.3->flask)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading flask-3.1.1-py3-none-any.whl (103 kB)
Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Downloading click-8.2.1-py3-none-any.whl (102 kB)
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
```

Step 3: Connect Flask to MongoDB Atlas

#Update your **app.py**:

```
from flask import Flask, request, render_template
```

```
from pymongo import MongoClient
```

```
app = Flask(__name__)
```

Connect to MongoDB Atlas

```
client =
```

```
MongoClient("mongodb+srv://<username>:<password>@cluster0.mongodb.net/?retryWrites=true&w=majority")
```

```
db = client["formdb"]
```

```
collection = db["submissions"]
```

Replace **<username>** and **<password>** with your actual credentials. You can also use environment variables for safety.

1. Flask App with **/api** Route

Let's start with a simple backend that returns JSON data from a file.

Folder structure

Assignment 3/

├─ app.py

├─ data.json

└─ templates/

 ├─ form.html

 └─ success.html

data.json

```
[  
    {"name": "Alice", "email": "alice@example.com"},  
    {"name": "Bob", "email": "bob@example.com"}  
]
```

app.py

```
from flask import Flask, jsonify, request, render_template  
  
import json  
  
from pymongo import MongoClient  
  
app = Flask(__name__)  
  
# MongoDB Atlas setup  
  
client = MongoClient("YOUR_MONGODB_ATLAS_URI")  
  
db = client["your_database"]  
  
collection = db["your_collection"]  
  
@app.route('/api')  
  
def api():
```

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

```
    data = json.load(f)
```

```
return jsonify(data)
```

```
@app.route('/')
```

```
def form():
```

```
    return render_template('form.html')
```

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

```
def submit():
```

```
    try:
```

```
        name = request.form['name']
```

```
        email = request.form['email']
```

```
        collection.insert_one({'name': name, 'email': email})
```

```
        return render_template('success.html', message="Data submitted successfully")
```

```
    except Exception as e:
```

```
        return render_template('form.html', error=str(e))
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```



```
File Edit Selection ... Search
C:\Users\minoha> OneDrive\Documents\DevopsWork\Assignment3> app.py
1 from flask import Flask, jsonify, request, render_template
2 import json
3 from pymongo import MongoClient
4
5 app = Flask(__name__)
6
7 # MongoDB Atlas setup
8 client = MongoClient("mongodbsrv://Manu001:Manu001@clustermanuug.rytqit.mongodb.net/?retryWrites=true&w=majority&appName=ClusterManuAug")
9 db = client["formdb"]
10 collection = db["submissions"]
11
12 @app.route('/api')
13 def api():
14     with open('data.json', 'r') as f:
15         data = json.load(f)
16     return jsonify(data)
17
18 @app.route('/')
19 def form():
20     return render_template("form.html")
21
22 @app.route('/submit', methods=['POST'])
23 def submit():
24     try:
25         name = request.form['name']
26         email = request.form['email']
27         collection.insert_one({'name': name, 'email': email})
28         return render_template("success.html", message="Data submitted successfully")
29     except Exception as e:
30         return render_template("form.html", error=str(e))
31
32 if __name__ == '__main__':
33     app.run(debug=True)
```

2. Frontend Form + Success/Error Handling

form.html

<!DOCTYPE html>

<html>

<head><title>Submit Form</title></head>

<body>

<h2>Submit Data</h2>

{% if error %}

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

{% endif %}

<form method="POST" action="/submit">

Name: <input type="text" name="name">

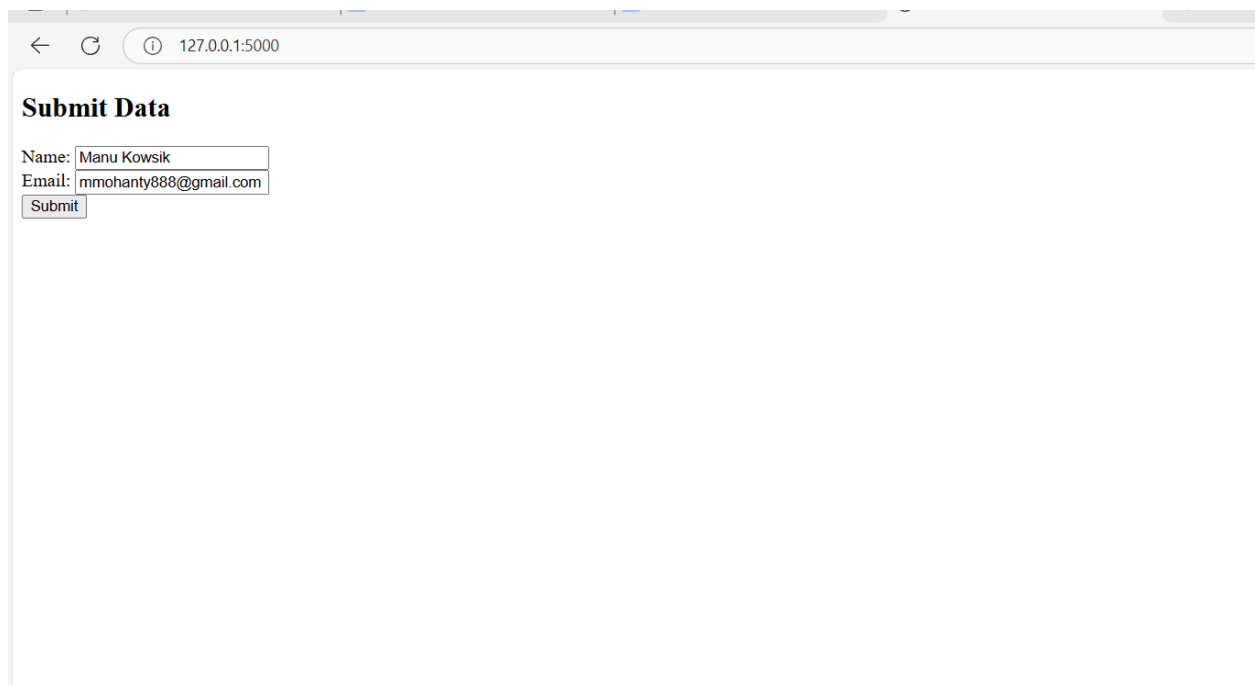
Email: <input type="text" name="email">

<input type="submit" value="Submit">

</form>

</body>

</html>



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. The page content is titled 'Submit Data' and contains a form with two text input fields: 'Name:' with the value 'Manu Kowsik' and 'Email:' with the value 'mmohanty888@gmail.com'. Below these fields is a 'Submit' button.

[success.html](#)

```
<!DOCTYPE html>
```

```
<html>
```

```
<head><title>Success</title></head>
```

```
<body>
```

```
<h2>{{ message }}</h2>
```

```
<a href="/">Go back</a>
```

```
</body>
```

```
</html>
```

Data submitted successfully

[Go back](#)