Flask-Mongo Assignment Submission

Question 1: Flask API using JSON Data

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.

🔹 **Objective:**

Create an API endpoint `/api` using Flask that:

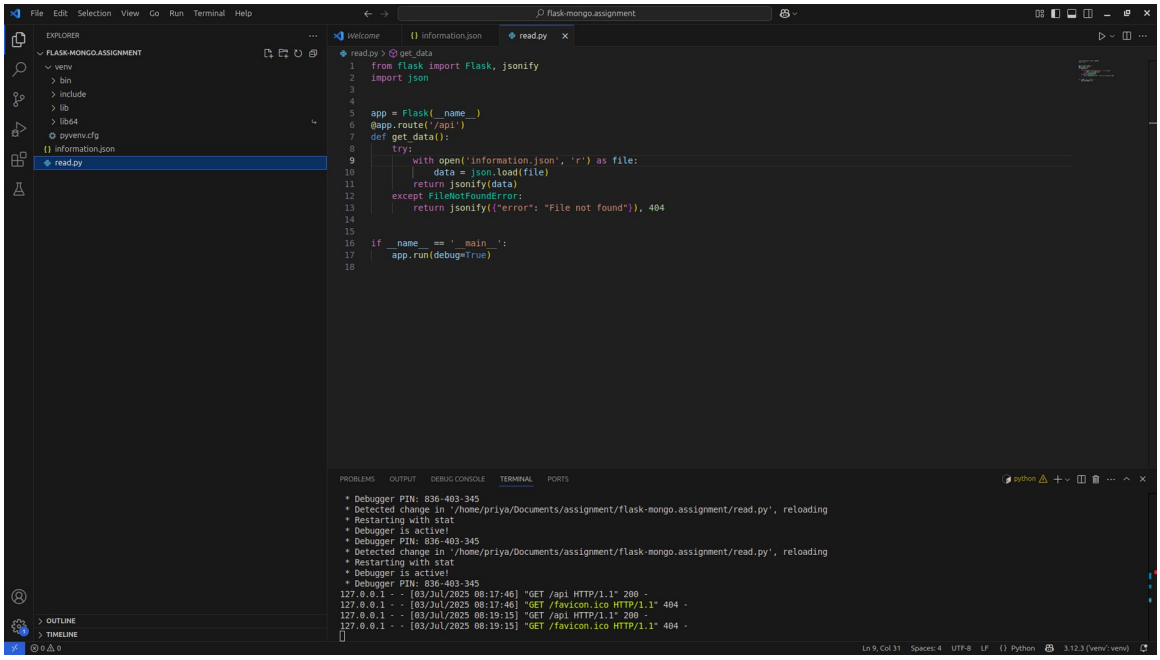
- Reads data from a file (`information.json`)
- Returns it as JSON using the `jsonify()` method

`read.py` — Flask Application Code

```
from flask import Flask, jsonify
import json

app = Flask(__name__)

@app.route('/api')

def get_data():

    try:

        with open('information.json', 'r') as file:

            data = json.load(file)

            return jsonify(data)

    except FileNotFoundError:

        return jsonify({"error": "File not found"}), 404

if __name__ == '__main__':

    app.run(debug=True)
```

Explanation of the Code:

Line	Purpose
`from flask import Flask, jsonify`	Imports Flask for web app and `jsonify` to return JSON responses
`import json`	Allows us to read `.json` files
`app = Flask(__name__)`	Initializes the Flask app
`@app.route('/api')`	Defines the API route `/api`
`with open('information.json', 'r') as file:`	Opens the file in read mode
`data = json.load(file)`	Loads the JSON data from the file
`return jsonify(data)`	Returns the JSON data as an API response
`except FileNotFoundError:`	If the file is missing, returns error JSON with 404
`app.run(debug=True)`	Runs the app with debug mode ON



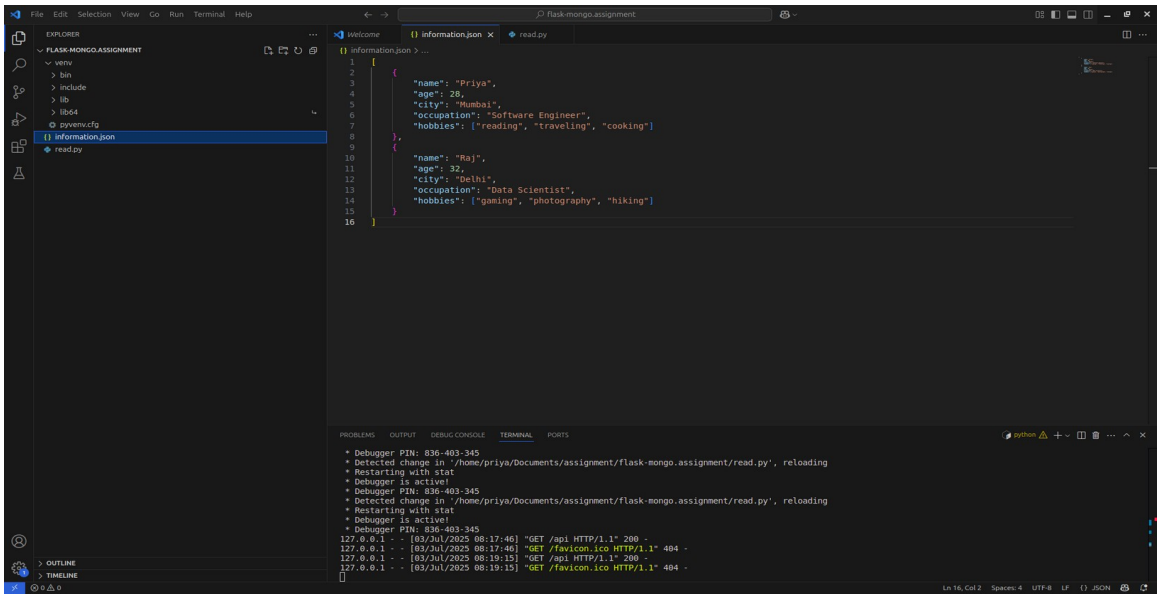
`information.json` — Data File


```
[
  {
    "name": "Priya",
    "age": 28,
    "city": "Mumbai",
    "occupation": "Software Engineer",
    "hobbies": ["reading", "traveling", "cooking"]
  },
  {
    "name": "Raj",
    "age": 32,
    "city": "Delhi",
    "occupation": "Data Scientist",
    "hobbies": ["gaming", "photography", "hiking"]
  }
]
```

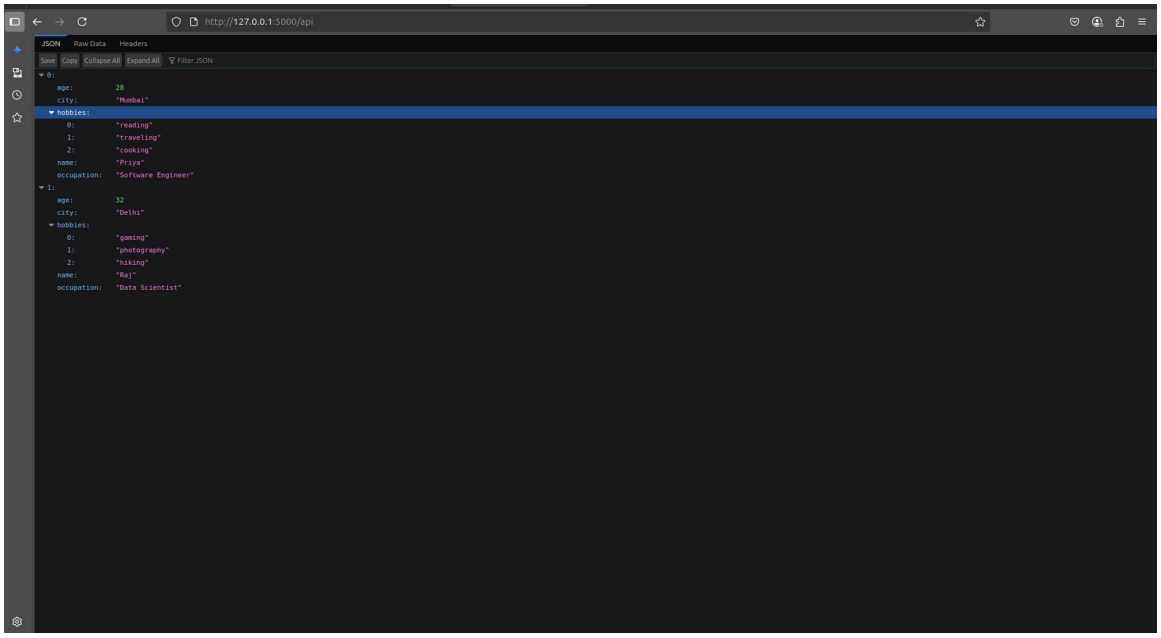

Explanation:
This file contains a **list of dictionaries**, each representing a person with:

- `name`, `age`, `city`, `occupation`
- `hobbies`: A list of strings

This structure simulates a **basic database-like storage in a file**, ideal for demo APIs.


File Edit Selection View Go Run Terminal Help
flask-mongo.assignment
EXPLORER
FLASK-MONGO.ASSIGNMENT
venv
bin
include
lib
lib64
pyvenv.cfg
information.json
read.py
Welcome
information.json
read.py
information.json > ...
[
{
"name": "Priya",
"age": 28,
"city": "Mumbai",
"occupation": "Software Engineer",
"hobbies": ["reading", "traveling", "cooking"]
},
{
"name": "Raj",
"age": 32,
"city": "Delhi",
"occupation": "Data Scientist",
"hobbies": ["gaming", "photography", "hiking"]
}
]
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
python
* Debugger PIN: 836-403-345
* Detected change in '/home/priya/Documents/assignment/flask-mongo.assignment/read.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 836-403-345
* Detected change in '/home/priya/Documents/assignment/flask-mongo.assignment/read.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 836-403-345
127.0.0.1 - - [03/Jul/2025 08:17:46] "GET /api HTTP/1.1" 200 -
127.0.0.1 - - [03/Jul/2025 08:17:46] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [03/Jul/2025 08:19:15] "GET /api HTTP/1.1" 200 -
127.0.0.1 - - [03/Jul/2025 08:19:15] "GET /favicon.ico HTTP/1.1" 404 -
OUTLINE
TIMELINE
Ln 16, Col 2
Spaces: 4
UTF-8
LF
{} JSON

Output


http://127.0.0.1:5000/api
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
0:
age: 28
city: "Mumbai"
hobbies:
0: "reading"
1: "traveling"
2: "cooking"
name: "Priya"
occupation: "Software Engineer"
1:
age: 32
city: "Delhi"
hobbies:
0: "gaming"
1: "photography"
2: "hiking"
name: "Raj"
occupation: "Data Scientist"

Question 2: Signup Form with MongoDB

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.

- **Objective:**
  - Create a web form that collects **name** and **email**
  - Insert this data into **MongoDB Atlas**
  - On successful submission, redirect to a "Data submitted successfully!" page
  - If there's an error (e.g., empty input or invalid email), show the error **on the same page without redirecting**

app.py — Main Flask App

```
from flask import Flask, render_template, request, redirect, url_for
from datetime import datetime
from dotenv import load_dotenv
import os
import pymongo


load_dotenv()
MONGO_URI = os.getenv('MONGO_URI')
client = pymongo.MongoClient(MONGO_URI)
db = client.best
collection = db['flask_mongo_assignment2']
```

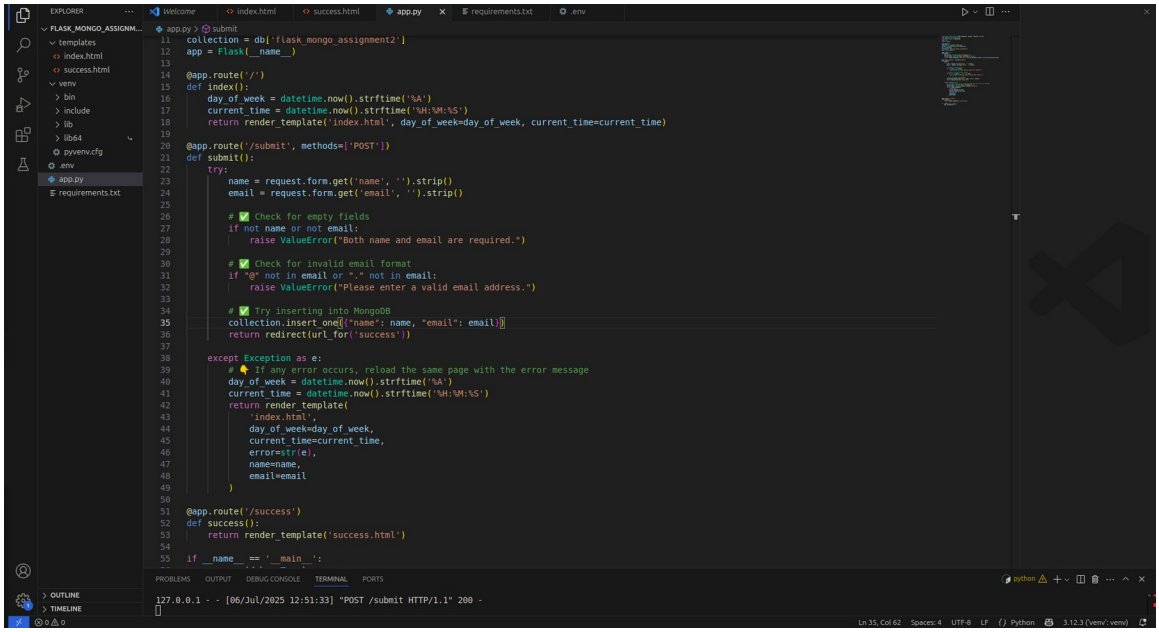
```
app = Flask(__name__)


@app.route('/')
def index():
    day_of_week = datetime.now().strftime('%A')
    current_time = datetime.now().strftime('%H:%M:%S')
    return render_template('index.html',
day_of_week=day_of_week, current_time=current_time)


@app.route('/submit', methods=['POST'])
def submit():
    try:
        name = request.form.get('name', '').strip()
        email = request.form.get('email', '').strip()


        if not name or not email:
            raise ValueError("Both name and email are
required.")


        if "@" not in email or "." not in email:
            raise ValueError("Please enter a valid email
address.")
```

```
        collection.insert_one({"name": name, "email": email})
        return redirect(url_for('success'))

    except Exception as e:
        day_of_week = datetime.now().strftime('%A')
        current_time = datetime.now().strftime('%H:%M:%S')
        return render_template(
            'index.html',
            day_of_week=day_of_week,
            current_time=current_time,
            error=str(e),
            name=name,
            email=email
        )

@app.route('/success')
def success():
    return render_template('success.html')

if __name__ == '__main__':
```

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

Explanation:

Part	Description
MongoDB Connection	Using `pymongo`, you connect to MongoDB Atlas using a `.env` stored URI
Route `/`	Loads the sign-up form and dynamically shows **day and time**
Route `/submit`	Validates form input, inserts into MongoDB, handles errors gracefully
Error Handling	If form is empty or email is invalid → Shows error message on the same form
Route `/success`	Shows success message upon successful submission

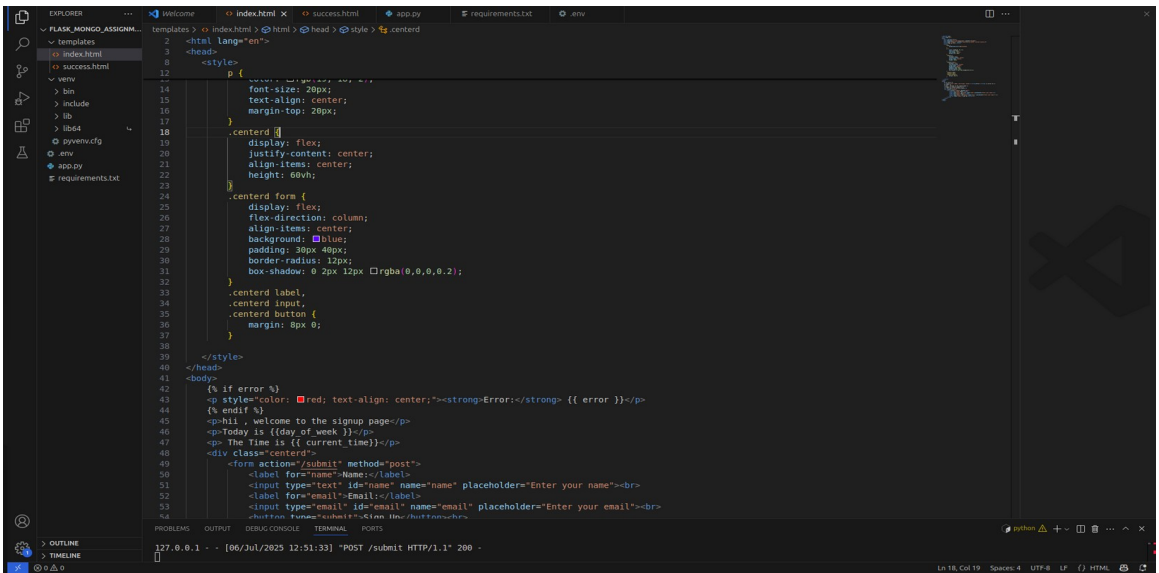


index.html — Frontend Form Page

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="UTF-8">
  <title>Sign up form </title>
  <style>
      body { background-color:whitesmoke; }
      p { text-align: center; font-size: 20px; }
      .centerd { display: flex; justify-content: center; align-items: center; height: 60vh; }
      .centerd form {
          display: flex; flex-direction: column; align-items: center;
          background: blue; padding: 30px 40px; border-radius: 12px;
          box-shadow: 0 2px 12px rgba(0,0,0,0.2);
      }
      .centerd label, .centerd input, .centerd button { margin: 8px 0; }
  </style>
</head>
<body>
  {% if error %}
  <p style="color: red; text-align: center;"><strong>Error:</strong> {{ error }}</p>
```

```
    {% endif %}
    <p>hii , welcome to the signup page</p>
    <p>Today is {{day_of_week}}</p>
    <p> The Time is {{ current_time }}</p>
    <div class="centerd">
        <form action="/submit" method="post">
            <label for="name">Name:</label>
            <input type="text" name="name" placeholder="Enter your name">
            <label for="email">Email:</label>
            <input type="email" name="email" placeholder="Enter your email">
            <button type="submit">Sign Up</button>
        </form>
    </div>
</body>
</html>
```



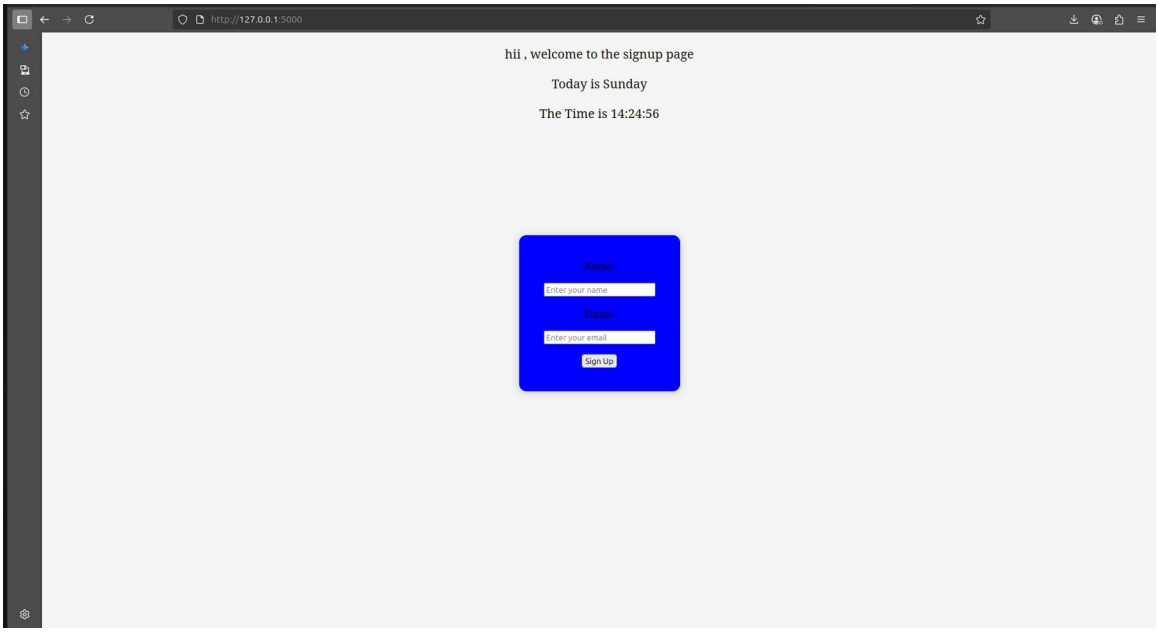
success.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Submission Status</title>
</head>
<body>
    <h1 style="text-align: center; color: green;">Data submitted successfully!</h1>
</body>
</html>
```

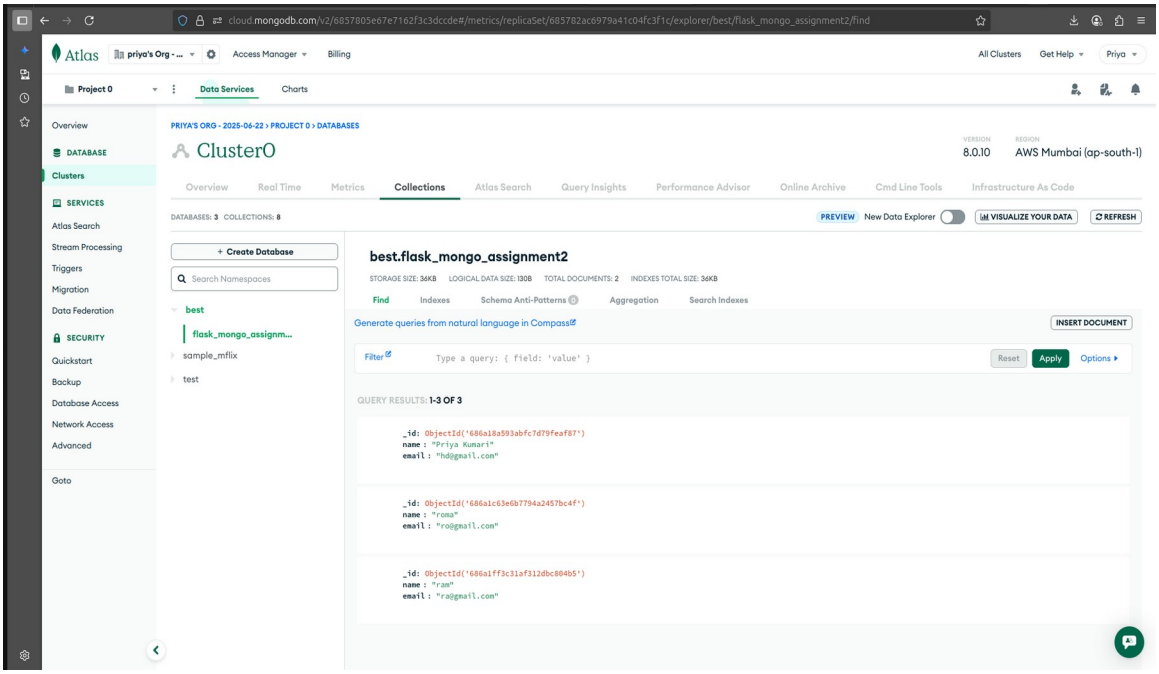
requirements.txt

```
flask
pymongo
dnspython
python-dotenv
```

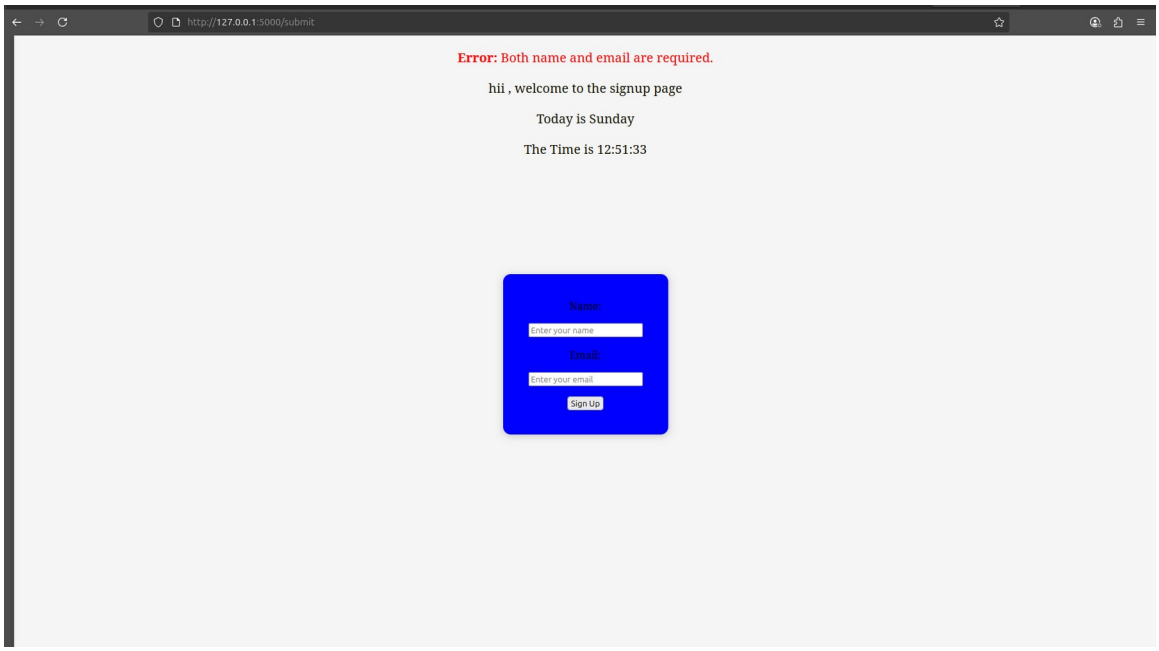
Successfully built a working form in Flask



Connected to a remote MongoDB Atlas database



Errors are shown on the same page (not redirected)



Data successfully submitted and confirmed by redirecting to success page