Yash Ashokbhai Savaliya

CWID: 861444743

ASSIGNMENT 5: Building a Simple Flask Application with MongoDB

This report documents the development and testing of a Flask-based web application that integrates with MongoDB to store and manage notes. The application allows users to add, view, and delete notes.

Project Requirements

- Implement a Flask application connected to MongoDB.
- Provide routes for:
- Homepage (/): Display all notes from MongoDB.
- Add Note (/add): Provide a form to add a new note.
- Delete Note (/delete/<note_id>): Remove a note by its unique ID.
- Use MongoDB Atlas or a local MongoDB instance.
- Include a well-structured HTML interface (home.html).

Flask Application (app.py)

The app.py file contains:

Flask Setup: Configures the application and initializes MongoDB connection.

Routes:

- Homepage (/): Fetches and displays all notes from MongoDB.
- Add Note (/add): Handles form submission and stores new notes.
- Delete Note (/delete/<note_id>): Deletes a note based on its MongoDB _id.

Setting MongoDB in Terminal:

```
C:\Users\ASHOK>mongo
MongoDB shell version v5.0.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("aac72b8f-439a-415c-8d1a-57d1087396e4") }
MongoDB server version: 5.0.5
```

```
> use note_app
switched to db note_app
> db.createCollection("notes
uncaught exception: SyntaxError: "" literal not terminated before end of script :
@(shell):1:26
> db.createCollection("notes")
{ "ok" : 1 }
> show collections
notes
```

MongoDB Configuration:

```python
from flask import Flask, render_template, request, redirect, url_for
from pymongo import MongoClient
from bson.objectid import ObjectId

app = Flask(__name__)


MONGO_URI = "mongodb://localhost:27017/"
client = MongoClient(MONGO_URI)
db = client["note_app"]
notes_collection = db["notes"]
```

Routes Implememtation:

```python
@app.route("/")
def home():
    notes = list(notes_collection.find())
    return render_template("home.html", notes=notes)
```

```python
@app.route("/add", methods=["POST"])
def add_note():
    full_name = request.form["full_name"]
    cwid = request.form["cwid"]
    if full_name and cwid:
        notes_collection.insert_one({"full_name": full_name, "cwid": cwid})
    return redirect(url_for("home"))
```

```python
@app.route("/delete/<note_id>")
def delete_note(note_id):
    notes_collection.delete_one({"_id": ObjectId(note_id)})
    return redirect(url_for("home"))

if __name__ == "__main__":
    app.run(debug=True)
```

User Interface (home.html)

The home.html file provides a simple yet functional interface for interacting with the application.

Features:

- Table listing all stored notes.
- Form for adding a new note.
- Delete button for each note.

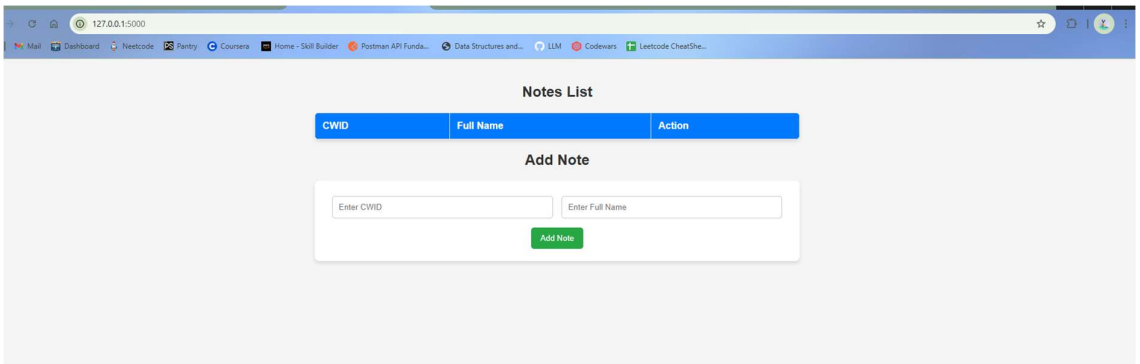
HTML Code:

```html
<h2>Add Note</h2>
<form action="/add" method="POST">
    <input type="text" name="cwid" placeholder="Enter CWID" required>
    <input type="text" name="full_name" placeholder="Enter Full Name" required>
    <br>
    <button type="submit">Add Note</button>
</form>
```

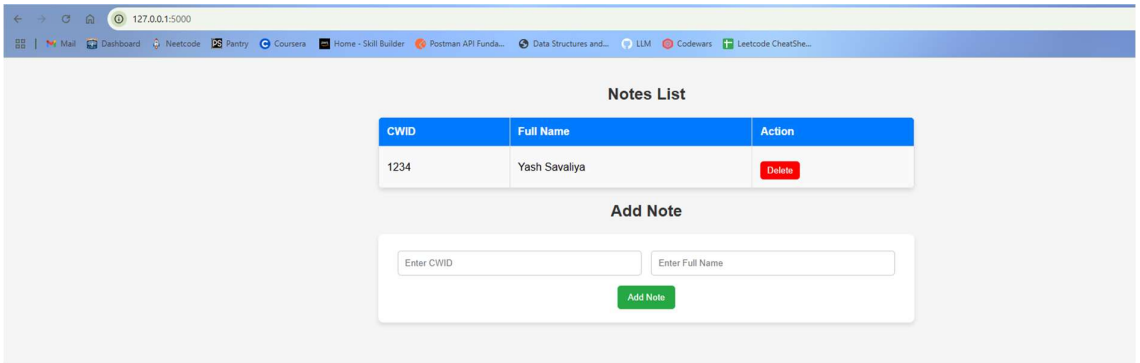
OUTPUT:

Flask app running at port: **http://127.0.0.1:5000/** with home page.

We need to enter **CWID** and **Full Name**.



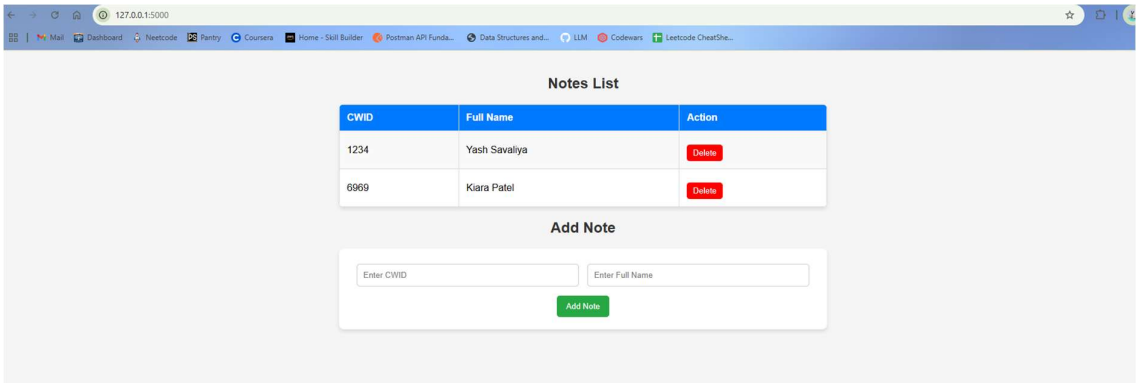
Making an entry by the name Yash Savaliya and CWID 1234:



Checking the MongoDB database through terminal and checking whether new entry has been made by the name Yash Savaliya:

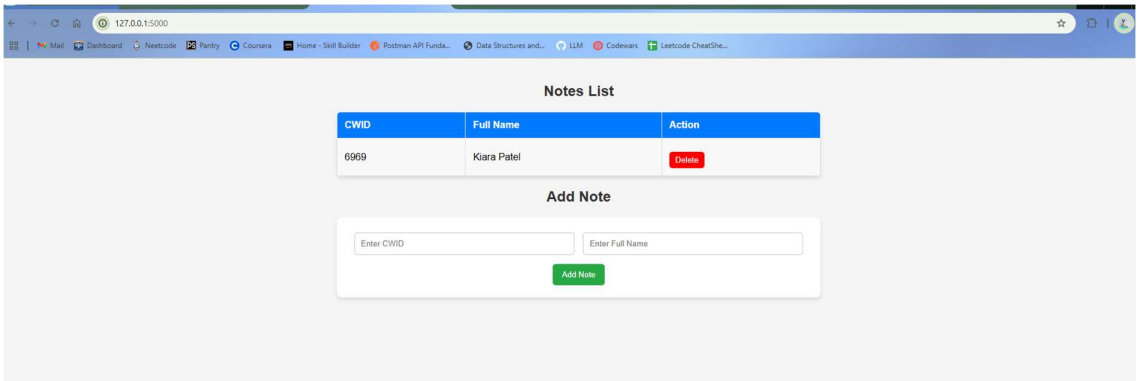
```
> show collections
notes
> db.notes.find().pretty()
{
        "_id" : ObjectId("67db44c894d5c98ef6aabe80"),
        "full_name" : "Yash Savaliya",
        "cwid" : "1234"
}
```

Successfully made a new entry by the name Kiara Patel and CWID: 6969




```
> db.notes.find().pretty()
{
        "_id" : ObjectId("67db48d440a012d634412b20"),
        "full_name" : "Yash Savaliya",
        "cwid" : "1234"
}
{
        "_id" : ObjectId("67db493040a012d634412b21"),
        "full_name" : "Kiara Patel",
        "cwid" : "6969"
}
```

Now, deleting the entry by the name Yash Savaliya:




```
> db.notes.find().pretty()
{
        "_id" : ObjectId("67db493040a012d634412b21"),
        "full_name" : "Kiara Patel",
        "cwid" : "6969"
}
```

CONCLUSION:

This project successfully implemented a **Flask web application** that integrates with **MongoDB** for managing notes. The application meets all specified requirements, including adding, displaying, and deleting notes. The user interface is intuitive, and the MongoDB backend efficiently handles data storage.

Key Learnings:

- How to set up and use **Flask with MongoDB**.
- Handling **routes** and **form submissions** in Flask.
- Using **ObjectId** for MongoDB documents.
- Fetching and displaying data dynamically in Flask templates.