🎥 MongoDB YouTube Video Manager with PyMongo 🐍 + 🌿

> A terminal-based app that lets you **add**, **view**, **update**, and **delete** YouTube videos using a **MongoDB Atlas database**. Built with 💖 Python + pymongo!

📦 Technologies Used

- **Python 3.13+**
- **MongoDB Atlas** (Cloud DB)
- **PyMongo** (MongoDB Python Driver)
- **BSON** for ObjectId

📁 Project Structure

```
youtube_manager_mongodb.py   # 🧠 Main logic file
.env                         # 🔐 (optional) for storing credentials securely
```

🧪 Step-by-Step Guide

🔧 1. Install Required Package

Install pymongo:

```
pip install pymongo
```

🔗 2. Connect to MongoDB Atlas

```
from pymongo import MongoClient

client = MongoClient("mongodb+srv://CHAI:CHAI@cluster0.lxl3fsq.mongodb.net/",
tlsAllowInvalidCertificates=True)
```

⚠️ **Important:**

- **Never** hardcode credentials (`CHAI:CHAI`) in real apps!
- Use environment variables or `.env` files with `python-dotenv`.

💾 3. Define Database & Collection

```
db = client["ytmanager"]
video_collection = db["videos"]
```

> Here, `ytmanager` is the database name, and `videos` is the collection.

🧠 Feature Functions (CRUD)

➕ Add a Video

```
def add_video(name, time):
    video_collection.insert_one({"name": name, "time": time})
```

📝 Inserts a document like:

```
{ "name": "My Vlog", "time": "12:34" }
```

📋 List All Videos

```
def list_videos():
    for video in video_collection.find():
        print(f"ID: {video['_id']}, Name: {video['name']} and Time:
{video['time']}")
```

☑ Shows all videos with their unique MongoDB `_id`.

✏ Update a Video

```
from bson import ObjectId

def update_video(video_id, new_name, new_time):
    video_collection.update_one({'_id': ObjectId(video_id)}, {"$set": {"name":
new_name, "time": new_time}})
```

⚠ Converts `video_id` from string to `ObjectId`.

❌ Delete a Video

```
def delete_video(video_id):
    video_collection.delete_one({"_id": ObjectId(video_id)})
```

🚨 Your original code had a bug: `delete_video(video_id, name, time)` had extra args. Fixed here!

🚀 Main Menu Loop

```
def main():
    while True:
        print("\n Youtube manager App")
        print("1. List all videos")
        print("2. Add a new video")
        print("3. Update a video")
        print("4. Delete a video")
        print("5. Exit the app")

        choice = input("Enter your choice: ")

        if choice == '1':
            list_videos()
        elif choice == '2':
            name = input("Enter the video name: ")
            time = input("Enter the video time: ")
            add_video(name, time)
        elif choice == '3':
            video_id = input("Enter the video id to update: ")
            name = input("Enter the updated video name: ")
            time = input("Enter the updated video time: ")
            update_video(video_id, name, time)
        elif choice == '4':
            video_id = input("Enter the video id to delete: ")
            delete_video(video_id)
        elif choice == '5':
            break
        else:
            print("Invalid choice")
```

🌀 This is the CLI menu system that ties all features together.

☑️ Final Working Code

```
from pymongo import MongoClient
from bson import ObjectId

client = MongoClient("mongodb+srv://CHAI:CHAI@cluster0.lxl3fsq.mongodb.net/",
tlsAllowInvalidCertificates=True)
```

```python
db = client["ytmanager"]
video_collection = db["videos"]

def add_video(name, time):
    video_collection.insert_one({"name": name, "time": time})

def list_videos():
    for video in video_collection.find():
        print(f"ID: {video['_id']}, Name: {video['name']} and Time:
{video['time']}")

def update_video(video_id, new_name, new_time):
    video_collection.update_one({'_id': ObjectId(video_id)}, {"$set": {"name":
new_name, "time": new_time}})

def delete_video(video_id):
    video_collection.delete_one({"_id": ObjectId(video_id)})

def main():
    while True:
        print("\n Youtube manager App")
        print("1. List all videos")
        print("2. Add a new video")
        print("3. Update a video")
        print("4. Delete a video")
        print("5. Exit the app")

        choice = input("Enter your choice: ")

        if choice == '1':
            list_videos()
        elif choice == '2':
            name = input("Enter the video name: ")
            time = input("Enter the video time: ")
            add_video(name, time)
        elif choice == '3':
            video_id = input("Enter the video id to update: ")
            name = input("Enter the updated video name: ")
            time = input("Enter the updated video time: ")
            update_video(video_id, name, time)
        elif choice == '4':
            video_id = input("Enter the video id to delete: ")
            delete_video(video_id)
        elif choice == '5':
            break
        else:
            print("Invalid choice")

if __name__ == "__main__":
    main()
```

📌 Best Practices (For Production)

🔒 Issue	💡 Better Practice
Hardcoded DB credentials	Use `.env` + `python-dotenv`
`tlsAllowInvalidCertificates=True`	Configure SSL properly or use local MongoDB
No error handling	Add `try/except` for DB operations
No input validation	Sanitize and validate user inputs

🔐 Using `.env` (Optional but Recommended)

```
MONGO_URI=mongodb+srv://username:password@cluster.mongodb.net/
```

```python
from dotenv import load_dotenv
import os
load_dotenv()
client = MongoClient(os.getenv("MONGO_URI"))
```
