Manage Chat Conversations for a GenAI Chat App using FastAPI

Welcome to the GenAI Chat App Backend Assignment! You're joining a team as a **Python GenAI developer**, and your first task is to implement backend logic for managing chat sessions and messages.

✅ Objective

Implement a FastAPI service to:

- Create new chat sessions
- Add messages to a session
- Retrieve messages from a session

Note: No database required — use in-memory Python data structures.

After completion of the assignment, the expectation is to create a git repository and push the project code.

🚀 (Optional) Starter Code Repository

Use this repo to get started or feel free to create a new project from scratch:

https://github.com/bibektimilsina000/FastAPI-PgStarterKit

⚙️ Prerequisites

- Python

Simulated In-Memory Storage

```
# Session metadata store

session_store = [

   {

       "session_id": 1,
       "session_user": "abc",
       "created_at": "2025-06-30T16:00:00"
   }
]
# Chat history store (session_id -> list of messages)
chat_store = {
   1: [
       {"role": "user", "content": "Hello"},
       {"role": "assistant", "content": "Hi there!"}
   ]
}
```

API Endpoints to implement

Create a New Chat Session

Endpoint:

`POST /sessions`

Request Body

{

  "session_user": "abc"

}

Response

{

  "session_id": 2,

"session_user": "abc"

"created_at": "2025-06-30T16:05:00"

}

Behavior:

- Add validation: Reject empty usernames.
- Normalize username input (Remove trailing spaces and convert to lowercase)
- Assign `session_id = len(session_store) + 1`
- Set `created_at` to current UTC timestamp
- Add new entry to `session_store`
- Initialize empty list in `chat_store[session_id]`

Add a Message to a Session

Endpoint:
`POST /sessions/{session_id}/messages`

Request Body

{

"role": "user",

"content": "What is AI?"

}

Behavior:

- Validates if session exists
- Validates if role is `user` or `assistant`
- Appends message to `chat_store[session_id]`
- Raise Exception if session ID doesn't exist or role is not valid

Get All Messages from a Session

Endpoint:

`GET /sessions/{session_id}/messages`

Response

```
[
  {"role": "user", "content": "Hello"},
  {"role": "assistant", "content": "Hi there!"}
]
```

Behavior:

- Returns full chat history for the session
- Add filtering by role using query param (use List comprehension)
- Raises Exception if session not found

Add Unit Tests

Consider adding tests for the above endpoints using pytest