🔥 AI Movie Character Chatbot - Internship Ladder Challenge 🚀

🏆 Goal: Progressively Build & Scale a Backend Chatbot

This internship challenge is **designed as a ladder**, where you start with a **simple implementation** and progressively tackle **more complex, real-world challenges**.

Each **milestone** gets harder, and you **earn points** for each level completed.

🎺 Scoring System

🏅 Level	🎯 Task	🏆 Points
Level 1	Basic API Chatbot	**10**
Level 2	Store & Retrieve Movie Script Data	**20**
Level 3	Implement RAG with Vector Search	**30**
Level 4	Scale System to Handle High Traffic	**40**
Level 5	Optimise for Latency & Deploy	**50**
BONUS	Fine-Tune AI / Add Voice / WebSockets	**20-50**

🏆 **Top Performers:** If you **score 100+ points**, you are among the **top candidates** and may receive **direct internship offers**.

📚 Movie Script Sources

For this project, you may find useful movie scripts at the following websites:

1. **IMSDb (Internet Movie Script Database)** - https://www.imsdb.com/
2. **SimplyScripts** - https://www.simplyscripts.com/
3. **The Script Lab** - https://thescriptlab.com/

📌 Submission Guidelines

✅ **Where to Submit:**

- Candidates should **submit their work via GitHub**.
- Provide a **GitHub repo link** containing:
  - Source code
  - README file
  - API Documentation

✅ **What to Include in the Submission:**

- **README.md** explaining how to set up and run the project.
- **API documentation** (Postman collection, OpenAPI schema, or Markdown file).
- **Screenshots / Demo Video** (if applicable).

🚀 Level 1: Basic API Chatbot (10 Points)

🖊️ **Task:** Build a simple backend that allows users to chat with a movie character.

✅ **Requirements:**

- Use **OpenAI GPT API** (or an open-source LLM like Llama2).
- Create a **simple REST API**:
  - `POST /chat` – Accepts `{character, user_message}` and returns a response.
- Responses should **mimic the character's personality** (even if manually defined).

🎯 **Goal:** Show basic API integration & response handling.

⏳ **Expected Time:** ~2-3 hours

🚀 Level 2: Store & Retrieve Movie Script Data (20 Points)

🖊️ **Task:** Instead of relying solely on the AI model, store **real movie dialogues** and use them for responses.

✅ **Requirements:**

- Scrape or download movie scripts from
- Store **dialogues per character** in a **database (PostgreSQL, MongoDB, or SQLite).**
- Modify `POST /chat` to:
    - First, **search the database** for an **exact or closely matching line**.
    - If found, return that line.
    - If not found, generate a response using AI.

🎯 **Goal:** Improve realism by retrieving **actual character dialogues** instead of generating everything from scratch.

⏳ **Expected Time:** ~5-7 hours

🚀 Level 3: Implement RAG with Vector Search (30 Points)

🖊️ **Task:** Improve chat accuracy using **Retrieval-Augmented Generation (RAG)** with **semantic search**.

✅ **Requirements:**

- Convert movie scripts into **vector embeddings** (use OpenAI Embeddings API, FAISS, Pinecone, or ChromaDB).
- Store embeddings in a **vector database**.
- Modify `POST /chat`:
    - **Retrieve the most relevant dialogue using a similarity search**.
    - Pass the **retrieved dialogue as context** to the AI model before generating a response.

🎯 **Goal:** Make AI **more accurate** by giving it **real dialogues** before generating responses.

⏳ **Expected Time:** ~10-12 hours

🏗️ Level 4: Scale System to Handle High Traffic (40 Points)

🖊️ **Task:** Optimize the backend to handle **100,000+ API requests per second**.

✅ **Requirements:**

- Add **caching (Redis) to speed up dialogue retrieval**.
- Implement **rate limiting (5 requests/sec per user)** to prevent abuse.
- Use **async programming (Celery, Task Queues, or FastAPI's async)** to improve performance.
- Run **load tests (using Locust or K6)** to benchmark performance.

🎯 **Goal:** Ensure the chatbot can **handle large user loads** without breaking.

⏳ **Expected Time:** ~12-15 hours

🏁 Level 5: Optimize for Latency & Deploy (50 Points)

🖊️ **Task:** Reduce response times to **under 500ms** and deploy the chatbot.

✅ **Requirements:**

- Implement **WebSockets** for **real-time chat** instead of REST APIs.
- Deploy the backend on **AWS, DigitalOcean, or Vercel**.
- Add **monitoring tools (Prometheus + Grafana)** to track performance.
- Store **chat history per user** in a database.

🎯 **Goal:** Make the chatbot **fast, scalable, and production-ready**.

⏳ **Expected Time:** ~15-20 hours

Frequently Asked Questions:

Will keep updating FAQs as questions come.

Q: How to ask questions?
A: You can ask by commenting on this document or by asking in the internshala chat.

Q: Do we need to follow the Levels sequentially?
A: No, Build in order you like.