Stevie TheTV Repository Documentation

Comprehensive documentation for the Stevie The TV project (also called the AI Movie & TV Show Companion). This guide explains what the project does, the technologies that were used to build it, and how the pieces of the repository fit together.

---

1. Project Overview

- **Purpose:** Stevie TheTV is a spoiler-aware AI copilot that watches along with a viewer by reading subtitle files and answering questions about what just happened in a film or TV episode.
- **High-level workflow:** A viewer uploads a video plus matching '.srt' file via the web UI; FastAPI stores the files and metadata; when the viewer asks a question the backend fetches the subtitle context up to the current timestamp, injects any watch history, and calls an LLM (OpenAI, Ollama, or Groq) to generate a spoiler-safe answer.
- **Key capabilities:**
    - Upload, list, stream, and delete local media inside the browser without extra tooling.
    - Subtitle-aware chat assistant that automatically trims context to the preceding ~5 minutes to keep prompts focused.
    - Watch history tracking so the assistant remembers how far the user progressed and what else they have seen.
    - Works with a local Ollama model by default but can switch to Groq or OpenAI via environment variables.
    - Lightweight FastAPI backend plus single-page HTML/CSS/JS frontend so it is easy to deploy on hobby clouds.

---

2. Technology Stack

Layer	Technologies	Notes
Backend runtime	**FastAPI**, **Uvicorn**, **Python 3.10+**	'run_server.py' loads 'env' files and boots Uvicorn. API is defined in 'movie_companion/server/api.py'.

Domain logic	'movie_companion' package	Modules include 'assistant.py', 'llm.py', 'library.py', 'history.py', 'subtitles.py', and 'time_utils.py'.
AI providers	OpenAI SDK, HTTP for Ollama and Groq	Provider selected via 'CompanionConfig'/ 'LLMSettings'.
Data parsing	'pysrt' for subtitles, 'json' for metadata	JSON files stored in 'data/'.
Frontend	Vanilla HTML, CSS, and ES6	Served directly by FastAPI using 'StaticFiles'.
Build/Tooling	No bundler required	All dependencies pinned in 'requirements.txt'. Windows helper script 'run_server.bat' is provided.

Export to Sheets

---

3. Repository Layout

- `run_server.py` / `run_server.bat`: Launchers
- `movie_companion/`: Python package (backend + domain logic)
- `web/static/`: Frontend assets (HTML/CSS/JS/images)
- `data/`: JSON metadata + watch history
- `media/`: Uploaded videos and subtitle files
- `DEPLOYMENT*.md`: Hosting guides
- `README.md`: Quickstart
- `SYSTEM_PROMPT_EXAMPLE.md`: Custom prompt instructions

---

4. Backend Architecture

4.1 Entry Points

- **run_server.py**: Loads '.env.local' and 'env' using 'python-dotenv', builds the FastAPI app, and runs 'uvicorn' on 'PORT' (defaults to 8000).
- **run_server.bat**: Windows helper that activates the repo directory and runs 'python run_server.py'.

4.2 FastAPI Application ('movie_companion/server/api.py')

- Configures directories under the project root ('data', 'media/videos', 'media/subtitles', 'web/static').
- Adds permissive CORS middleware and mounts '/static' to serve the frontend.
- **Routes:**
  - `GET /health`: Quick status check.
  - `GET /videos`: List uploaded items.
  - `POST /videos`: Multipart upload handler for title, video, and '.srt'.
  - `GET /videos/{id}/stream` & `/subtitles`: Stream stored assets.
  - `DELETE /videos/{id}`: Delete metadata and on-disk files.
  - `GET /context`: Fetch subtitle context up to a timestamp.
  - `POST /ask`: Accept question, build 'CompanionConfig', and offload to thread pool.

---

5. Frontend Application ('web/static')

- **index.html**: Single-page layout featuring a top bar, upload drawer, video player, and chat sidebar.
- **styles.css**: Modern CSS with variables, responsive layout, and animated chat bubbles.
- **app.js**: Pure vanilla JavaScript controller that handles library interactions, fetches API data, syncs subtitles in-browser, and manages chat scrolling/retries.

---

6. Running the Project Locally

1. Install Python 3.10+ and (optionally) Ollama.
2. Create and activate a virtual environment.
3. `pip install -r requirements.txt`.
4. Export one of the supported API keys (`OPENAI_API_KEY`, `GROQ_API_KEY`, or run `ollama serve`).
5. `python run_server.py`.
6. Navigate to `http://localhost:8000`, upload media, and start chatting.