ChatGPT

Movie Catalog API

Overview

Movie Catalog API is a simple FastAPI application that lets you record and manage a list of movies (or books). It uses SQLite for storage, SQLAlchemy for persistence and Pydantic for request/response validation. The app is designed as a learning project and demonstrates typical CRUD operations [1].

Features

- **Health check:** view a root endpoint that confirms the API is running [1].
- **List movies:** get all movies with optional keyword search and sorting by rating or year [2].
- **Retrieve a movie:** fetch a single movie by its ID [3].
- **Create a movie:** add a new movie entry [4].
- **Update a movie:** modify an existing movie entry; partial updates are supported [5].
- **Delete a movie:** remove a movie by ID [6].

Setup

1. **Install dependencies**:
   Use pip (inside a virtual environment is recommended) to install FastAPI, Uvicorn, SQLAlchemy and Pydantic:

```
pip install fastapi "uvicorn[standard]" sqlalchemy pydantic
```

1. **Run the application**:
   From the project directory, start the server with:

```
python -m uvicorn main:app --reload
```

The server will start on `http://127.0.0.1:8000` and reload automatically when you change code.

Usage

Interactive documentation

FastAPI automatically generates interactive API documentation. Navigate to `http://127.0.0.1:8000/docs` to explore and call endpoints directly in your browser. Each route defined in `main.py` [1] will be listed with its parameters and example responses.

Endpoints

Below is a summary of the available endpoints. Each corresponds to a function defined in `main.py` [1] and implemented using CRUD helpers [7].

Method	Path	Description
GET	/	Health check – returns a running messa
GET	/movies/	List movies with optional search and s
GET	/movies/{id}	Retrieve a movie by ID
POST	/movies/	Create a new movie
PUT	/movies/{id}	Update an existing movie
DELETE	/movies/{id}	Delete a movie by ID

Example requests

Create a new movie:

```
POST /movies/
Content-Type: application/json

{
  "title": "Inception",
  "director": "Christopher Nolan",
  "year": 2010,
  "rating": 9.0,
  "watched": true
}
```

List movies with search and sort:

```
GET /movies/?search=inception&sort_by=rating
```

Retrieve a movie:

```
GET /movies/1
```

Update a movie:

```
PUT /movies/1
{
  "rating": 9.5
}
```

Delete a movie:

```
DELETE /movies/1
```

Data model

The API persists movies in an SQLite database via SQLAlchemy. The `Movie` model contains the fields `id`, `title`, `director`, `year`, `rating` and `watched` [8]. CRUD helper functions encapsulate create, read, update and delete operations [7].

Notes

- The default SQLite database file (`movies.db`) is created in the project directory the first time the app runs [8].
- The API uses Pydantic to validate and serialize request and response bodies. In Pydantic v2 you may see a warning about `orm_mode`; to silence it, replace `orm_mode = True` with `from_attributes = True` in your schema definitions.

Enjoy building and experimenting with your Movie Catalog API!

---

[1] [2] [3] [4] [5] [6] GitHub
https://github.com/chambies2015/Movie-Catalog/blob/main/main.py

[7] GitHub
https://github.com/chambies2015/Movie-Catalog/blob/main/crud.py

[8] GitHub
https://github.com/chambies2015/Movie-Catalog/blob/main/database.py