Movielens API

Consider the Movielens 20M Dataset from Movielens.

NOTE: There are different Movielens datasets with different file structure. Please make sure you are using the `*Movielens 20M*` dataset.

Technology stack

- Python 3.9+
- Django 4+
- Django Rest Framework (DRF) with Viewsets
- pytest

Core Requirements

- API application exists with database backend by choice
- Persistence (database) models that support the Movielens 20M dataset exist
- Dataset data is loaded into models
- Can list movies (`GET /api/movies/`)
- Movie information includes `movieId`
- Movie information includes movie `title`
- Movie information includes a `tags` attribute which is a string comma-separated list of tags
- Movie information includes a `genres` attribute which is a list of strings
- Movie list is paginated
- Can specify movies order by title
- Can retrieve movie (`GET /api/movies/{movieId}/`)
- Can create movie (`POST /api/movies/`)
- Can update movie (`PUT /api/movies/{movieId}`)
- Can update movie using partial update (`PATCH /api/movies/{movieId}/`)
- User can rate a movie (`POST /api/movies/{movieId}/rate/`)
- User cannot cast multiple votes for the same move. Attemt to do so results in user error.
- Only authenticated users can cast a vote
- Only authenticated users can perform actions
- Can filter movie list to include only movies having given tag
- Can filter movie list to include only movies from particular genre

Nice to have

- Movie information includes `rating` - a decimal field with average ratings cast on the movie
- Excecutable documentation (e.g. OpenAPI Swagger interface)
- Movie list and retrieve use a database view
- Can list the links for a given movie
- Management command to export the movie list

Topics for discussion

- Django and Django REST Framework:
  - DRF Viewsets - model viewset mixins
  - Model Serializers
- Object-oriented Python
  - Inheritance
  - Polymorphism
  - Multiple Inheritance
- Test Driven Development TDD
- Testing with `pytest`
  - fundamentals
  - fixtures
  - parameterized tests
  - test coverage - measuring, applicability
- Testing Django REST Framework API with pytest-django package
  - API/URL testing using `client` fixture
  - View testing using `request factory`
  - Best practices for testing DRF applications
- Coding practices (e.g. static code analysis, linting and formatting)
- Devops culture and practices
- Build and package for Python

Implement as many of the requirements as you find appropriate to demonstrate enough knowledge and understanding of the listed topics for discussion. Application should be provided in a code repository (github?) or code dump with a README file explaining how to start the application.

Ability to disucss how the requirements are implemented and/or are going to be implemented is curcial.