ECE:5995 Modern Databases – Fall 2025

Final Project – Polyglot recommendation system

Due Friday Dec 12th, 2025, 11:59pm.

For this project we will use the movielens collection (100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users). We'll only use the movies and ratings csv files available on ICON.

The movies file includes movie titles, genres, release years, and unique movie IDs. The ratings file records user ratings for movies, with each rating associated with a user ID and a movie ID.

The goal of the project is to design a recommendation system that suggests movies to users, specifically movies they have not yet rated or seen.

You are free to choose suitable design strategies and database systems that support an optimal recommendation system.

Part 1. Upload the data into your database.

- Select an appropriate primary database system and design a suitable schema for the MovieLens data.
- Import the movies and ratings data into your database.
- Create appropriate indexes and constraints to support efficient query performance.

Part 2. Data design pattern and recommendation strategy.

- Design the data model to support movie recommendation queries.
- Extend or structure the data to incorporate similarity or preference metrics.

- Implement a recommendation approach using queries in the language supported by your chosen database system. Possible strategies include content-based filtering, collaborative filtering, latent factor-based models, or hybrid approaches.
- Retrieve and provide the top 5 recommended movies for a user, ensuring they are movies the user has not previously rated.

Part 3. Integration with Redis for caching and search.

- Write a Python script to extract the complete movie list from the primary database and load it into Redis. This is a one-time operation.
- In Redis, store each movie as a hash using the key format movie:<movieID>.
- Each stored hash must include the title, genre, and computed average rating.
- Create a full text search index over the movie: prefix on the Redis database (https://redis.io/docs/latest/commands/ft.create/).
- Sample code for Redis full text search indexing:

```
FT.CREATE movies_index ON HASH PREFIX 1 "movie:" SCHEMA title TEXT
```

Part 4. User application requirements.

All operations must be persistent. The application must support the following features:

1. Prompt the user to enter their user ID:
   If a username exists in the database, display a welcome message with the name.
   If not, prompt the user to enter their name and store it in the primary database.
2. Retrieve the list of movies the user has rated along with their ratings from the primary database.
   Cache this information in Redis using a structure of your choosing.
   Apply an expiration time to the cache and provide justification for your design choice.
3. Support movie title search in Redis using full-text search.
   Limit search results to at most 10 movies.
   For each returned movie, provide:
   title, genre, average rating, whether the user has seen it, and the user's rating if applicable.

4. Display the top 5 recommended movies the user has not seen or rated.
5. Allow the user to submit a rating for any recommended movie.
   Store the new rating in the primary database and update the average rating stored in Redis.
6. Ensure that all database connections are properly closed before the application exits.

Submission requirements:

1. **Zip file submission (due Friday, December 12th, 2025):**
   - Source code for the entire project.
   - Documentation/README.txt containing:
     - A concise file, akin to a "user's manual".
     - This file should describe the language version, program package(s) in a few lines, how to install it, how to use it, and how to run a demo. Important: A grader must be able to run and reproduce your results.

2. **Final report, presentation slides, and video demo submission (due Monday, December 15th, 2025):**
   The final report should include:
   - Explanation of database system choice and motivation behind the choice.
   - Data schema and indexing strategies for each system used.
   - Justification of design decisions including the recommendation method, similarity metrics, and Redis caching approach.
   - Discussion of results and performance.
   - Lessons learned.

   The final presentation should include the relevant information from your final report and a video demo of your project. The presentation should be no more than 10 mins (7-8 mins would be ideal).

3. **Team survey submission (individual, due Tuesday, December 16th, 2025):**

- Required for group submissions.
- Provide comments on each team member’s contributions, including your own, along with a score reflecting the level of contribution.