

## COMMANDS:

`./client --server (ip of server) -- port (port being used)`

`./client --server 127.0.0.1 --port 8080`

`./server --load (path of dataset) --port (port number being used)`

`./server --load ../ml-100k --port 8080`

## TO BUILD:

```
g++ -o client client.cpp -std=c++17
g++ -o server server.cpp -std=c++17 -pthread
```

```
kehale@kehale-fedora:~/code/engine/suggestion_engine/server$ g++ -o client client.cpp
-std=c++17
```

```
kehale@kehale-fedora:~/code/engine/suggestion_engine/server$ g++ -o server server.cpp
-std=c++17 -pthread
```

## For AWS Deployment:

Using the same server but on aws ec2 instance located in india for closeness.

Everything on server remains the same, we need to allow the port number from instance settings.

GUI based frontend is written in streamlit python (need to install it to run)

Need to replace the instance ipv4 and port number for proper connection and working

## Movie Scoring

To rank movies for a specific user, the system calculates a score using three components: **Genre Match**, **Quality**, and **Popularity**.

**Formula:**

$$\text{FinalScore} = \text{GenreMatch} \times \text{AvgRating} \times (1 + \text{PopularityBoost})$$

When a user rates a movie, the system updates their preference for that movie's genres. This determines how much the user "likes" a specific genre.

**Formula:**

$$S_{\text{new}} = S_{\text{old}} + (R \times (R - \mu) \times W_{\text{genre}})$$

- S: The user's affinity score for a specific genre.
- R: The rating the user gave the movie (1.0 to 5.0).
- $\mu$ : The user's average rating across all movies they have seen.
- $(R - \mu)$ : The **deviance**. If positive, the user liked it more than usual; if negative, they disliked it.
- $W_{\text{genre}}$ : Weighting factor. The system gives the **first** genre listed for a movie **1.5x** weight, and others **1x**.

To rank movies for a specific user, the system calculates a score using three components: **Genre Match**, **Quality**, and **Popularity**.

**Formula:**

$$\text{FinalScore} = \text{GenreMatch} \times \text{AvgRating} \times (1 + \text{PopularityBoost})$$

### Component 1: Genre Match (Gmatch)

$$\text{Gmatch} = i \sum (S_{\text{user},i} \times W_i)$$

- Sum of the user's scores (S) for all genres the movie belongs to.
- **Note:** The system heavily biases the "primary" genre (the first one in the list) with a **1.5x** multiplier, while others get **1x**.

## Component 2: Popularity Boost

$$\text{Score} = \text{AvgRating} \times \ln(1 + \text{TotalRatings})$$

- Uses the natural logarithm ( $\ln$ ) to smooth out the numbers. A movie with 1,000 ratings isn't "10 times better" than one with 100, but it is significantly more reliable. The division by 10 scales this down so popularity doesn't completely overpower personal preference.

When recommending popular movies generally (not personalized):

It gives top rated movies from different genres to try to learn user preferences