Project 2

• CMUQFlix!
• A Movie Recommendation System
Project 2 Objectives

• Set up a front-end website with PostgreSQL as the back-end

• Allow users to login, “like” movies, and get personalized movies recommendations
Agenda

- **Cookies** in Project 2

- **Recommendation system** in Project 2
Agenda

• **Cookies** in Project 2

• **Recommendation system** in Project 2
Cookies

• Small (text) files stored on your computer

• They help remember information about a user (keeping a session active)
Cookies: How do they Work?

• When a web browser requests a page from the server, the “cookies” for that page are sent as part of the request.

• On the server (your JAVA code!), you will look for cookies in the request.
Cookies: Case 1

- You will be **creating a cookie** when a user **logs in**:

```java
doGet (HttpServletRequest request, HttpServletResponse response) {
    ...
    String username = request.getParameter("username");
    Cookie unameCookie = new Cookie("username", username);
    unameCookie.setMaxAge(3600); /* one hour in s */
    response.addCookie(unameCookie);
    ...
}
```

- This stores the cookie on your computer
Cookies: Case 2

- You will be **checking for a cookie** when a user accesses some page (e.g. index.html):

```java
doGet (HttpServletRequest request, HttpServletResponse response) {
    ...
    Cookie unameCookie = null;
    Cookie allCookies[] = request.getCookies();
    if (allCookies != null)
        for (Cookie c : allCookies)
            if (c.getName().equals("username"))
                unameCookie = c; // username = c.getValue();
    if (unameCookie != null) { \ \ show index.html content }
    else { \ \ show login.html or some login/register form}
    ...
}
```
Cookies: Case 3

• You will be removing a cookie when a user logs out:

```java
doGet (HttpServletRequest request, HttpServletResponse response) {
...
    Cookie unameCookie = null;
    unameCookie = new Cookie("username", null)
    unameCookie.setMaxAge( 0 );
    response.addCookie(unameCookie);
...
}
```
Agenda

• **Cookies** in Project 2

• Recommendation system in Project 2
Recommendation System: Case 1

• If a user \( u \) has not yet “liked” any movies:
  • Display the top 5 “liked” movies in the database
Recommendation System: Case 2

• If a user $u$ has “liked” at least one movie:
  1. Find out what is the user $u$’s “movie clan” is
     • The user’s movie clan is the group of all users of have liked at least one movie $u$ is liked
     • In the figure, Anca’s movie clan would be:
       • Bob
       • Charlie
Recommendation System: Case 2

• If a user $u$ has “liked” at least one movie:
  2. Retrieve all the movies that have been liked by user $u$’s movie clan:

  • In the figure, for Anca, these movies are:
    • Argo
    • Troy
    • Bolt
Recommendation System: Case 2

• If a user $u$ has “liked” at least one movie:

  2. Retrieve all the movies that have been liked by user $u$’s movie clan:

• In the figure, for Anca, these movies are:

  • Argo \((clan\ in\text{-}degree = 3)\)
  • Troy \((clan\ in\text{-}degree = 2)\)
  • Bolt \((clan\ in\text{-}degree = 1)\)
Recommendation System: Case 2

• If a user $u$ has “liked” at least one movie:

  2. Retrieve all the movies that have been liked by user $u$’s movie clan:

• In the figure, for Anca, these movies are:

  • *Argo* (clan in-degree = 3)
  • *Troy* (clan in-degree = 2)
  • *Bolt* (clan in-degree = 1)

Anca has already liked this movie!
Recommendation System: Case 2

- If a user $u$ has “liked” at least one movie:
  2. Retrieve all the movies that have been liked by user $u$’s movie clan:

- The final list of recommendations for Anca:
  - *Troy* \((\text{clan in-degree} = 2)\)
  - *Bolt* \((\text{clan in-degree} = 1)\)

- Top 5 with the largest clan in-degree