# CS374 – Intro to Database Management

# Application Development Project

# Rubric for Second Deliverable

## Group Member #1: Michelle Lie

## Group Member #2: Salim Dhahri

## Group Member #3: Esther Ng

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Requirements | Points | Awarded |
| Description of Application | * An overview of your application * System requirements (e.g. hardware, DBMS, other software) * A detailed description of your application * Are there features that will not be implemented? What are they, and why won’t you fulfill them? | 10 |  |
| Project Management -Schedule | * Detailed schedule of who will do what part of project, by when | 5 |  |
| Logical Diagram | * Logical diagram in UML or E-R * Discussion of how your data model will satisfy the needs of your application * Discussion of alternative designs that you did not do (and why) | 15 |  |
| Queries Required | * Required queries in English (not SQL) * What entities and/or relationships are required for each query? * How will each query satisfy the needs of your application | 15 |  |
| Grammar, punctuation, syntax, and references | * Follow rules from the Penguin handbook on writing * References as appropriate (e.g. if you are modeling your application after an existing application, make note of that) | 5 |  |

**Description of Application**

* Overview: Our application will allow users to store movies they’ve watched before, as well as storing information about movies and directors. Each movie will have data stored about its genre, director, overall user ratings, length, actors, release date, and availability on streaming platforms. Users can rate the movies and add to their watched list, which will allow the application to recommend movies or directors to the user based on their ratings or watch history.
* System requirements: We are planning to use mySQL to store database tables, and Python or C++ for embedded SQL, manipulating and outputting data to the user.
* Detailed description: Users will add the movies they have watched, with ability to rate the movie. The user can also find the other top-rated movies in a specific genre, other movies a specific director has made, movies released in a specific year, and more. Also, we are planning to implement embedded SQL to create an interface for users to interact with, to retrieve the queries
* Features not implemented:
  + Future development could include more collaboration between users, such as creating a common movie “playlist/watch list”

**Project Management – Schedule**

* By November 21: Make the database for the project, including all the tables with the data that contains all the attributes
* Over Thanksgiving Break: each group member comes up with mySQL query for specific categories
* November 28: Project status update due
* Between November 28-December 10: Implement embedded SQL with interactive interface for users to select queries
* December 10-14: Create final presentation and demo, work on individual writeup
* December 14: Final Presentation – final project, demo, and individual writeup due

**Logical Diagram**

**A screen shot of a movie

Description automatically generated**

* This design helps lay out our project's objectives in a way that emphasizes the relationships between different tables.
  + To start with, “Directed” is a two-way relationship between a movie and its director since a director directs a movie and consequently, a movie is directed by a director.
  + Additionally, when it comes to the “Watched” relationship, it connects a user to a movie since the user watches the movie which adds them to their rated and watched lists, also a movie’s rating will be based on the “Watched relationship”
* Other designs we did not do included incorporating DateWatched and UserRating as attributes within Users. However, we realized this design would make updating a new user more difficult. Incorporating it as part of the relationship between Movie and Users allows for DateWatched and UserRating to be attributes that are more easily manipulated and used in queries.

**Queries Required**

* Retrieve all the information about a specific user
  + **Requires user table**
  + Satisfies requirements of application by ensuring information about a specific user can be pulled – in an actual application, this would support features like being able to look up a friend, for example
* Retrieve all the information about a movie that a specific user watched
  + **Requires watched entities and user**
  + Allows application to recommend movies to user based on watch history
* Retrieve all the information about a director of a specific movie
  + **Requires director and movie table**
  + Allows application to find movies based on director
* Retrieve all the movies in a specific [genre, year, length, director, etc.]
  + **Requires movie table**
  + Allows application to recommend movies to user of a specific attribute

Embedded SQL

* Sort movies based on ratings
  + Sort movies in a specific [genre, year, length, director, etc.]
* Recommend top movies based on a user’s highest rated movie or director
* Find the average rating for a specific movie based on users’ input/rating