Ratings & Suggestions

Software Engineering Principles • Group Project 2
Person-Hour Estimate: _______

In this activity, your team will design and demonstrate an application that allows people to provide product ratings and makes product recommendations for them. We will refer to the people who use the program as “users”. The following text refers to the “products” in the database but your implementation should use a specific category of product such as books, music, movies, games, restaurants, TV shows, etc.

Collecting, storing, and reporting user ratings

Your application will manage information about a set of *products* and a set of *users.* Users will *rate* the products. Any given user probably doesn’t have experience with all the products so will only have ratings for some of them. All data must be saved between application runs so that users can connect multiple times and accumulate ratings.

For each product, the application must store relevant and appropriate identifying information for the kind of product you have chosen (such as author and title if the products are books). Note that there are often multiple products with the same values of several attributes.

Likewise, identifying information must be stored for users (name, account-id, etc.). You might also choose to implement user passwords. It must be possible to add new users to the system dynamically.

For each user, the application needs to maintain the rating for each product that the user has rated thus far. Think carefully about an appropriate rating system. A returning user may reconnect to the system and rate more products and change past ratings.

The application must produce statistics about a given product’s ratings, such as the number of ratings, the average rating, the median rating, and the like.

Making predictions and suggesting products for users

When you browse or make purchases at some online sites (such as Amazon.com and Netflix.com), they suggest other products you might like. The displayed page may state, “other users who bought this item also bought …” The better a site is at making predictions that actually match the users’ tastes, the happier the users will be and the more likely they are to trust the recommendations.

The complexity of the prediction algorithm can range from simple comparisons to complex statistical models. Additional information about making recommendations will be provided in a separate document.

Project Planning

Carefully consider your options for software development life cycle model and their characteristics. Establish meaningful, realistic, and tangible milestones. Consider how to most effectively and realistically utilize the collective resources. Anticipate team dysfunctions and develop concrete guidelines for addressing and resolving potential conflict.

Basic requirements

Many details are intentionally unspecified. Part of your assessment is the quality of and justifications for the decisions you make.

Products

The application must keep track of at least the following information about each product in the system:

- Unique identification: This value uniquely identifies a product.
- Common name: There may be multiple products with the same common name.
- Attributes: A collection of additional attributes typically associated with the product type.

Users

Your program must keep track of at least the following information about each user:

- Name: The name of the user, e.g., "Stymie Hourgang". Multiple users may have the same name.
- Account: A unique account name in the system. No two users may have the same account name.

Your program must also keep track of which user is currently logged in, if any. Upon program startup or immediately after a user logs out, there is no user currently logged in.

Ratings

Your program must keep track of product ratings.

- Each rating has three pieces of information: the product, the user, and the rating.
- Although the data on which you test your program will likely be somewhat small, in a real system there would be a great many products, a great many users, and no user would have rated a very large fraction of the products. If you picture the ratings data as a table (rows being users and columns being products, with rating values in the cells of the table), most of the table would be empty. This is known as “sparse” data. Keep this in mind when designing your structure for storing ratings.

Graphical User-Interface (GUI)

Your program must use a GUI for all input and output exchanges with the user.

Your program must provide feedback indicating the result of every operation the user requests. For example, if the user tries to add a new user with an account name that is already in the system, it might display the message “Sorry — that account name is already in use. Please try a different name.” You must decide how to handle the situations that might arise.

Operations

When it starts, the application must load all the existing data (as created in previous runs of the application) from a data file. If the file doesn’t exist — because this is an initial run of the program — the application must initialize all the data structures as appropriate. Prior to exiting, the application must have automatically saved the updated data to the data file for use in the next application run.

Your application must provide the following basic operations for the user.

- **Log in**
  If there are no passwords, anyone can log in by using a valid id. If there are passwords, they must be verified for successful log in.
- **Log out**
- **View all your own ratings**
- **Rate a product**
- **See recommended products**
- **Quit**
- **Load external data**
  Read data from a file that may not have been created by this application. The user specifies the file. Each person in a ratings file must be created as a new user, if that user doesn’t already exist. Product and rating data must be loaded into the application’s data structures.
- **Add a new product**
- **Add a new user**
  If no “admin” user, this is allowed no matter who is logged in, and even if no one is logged in. If an “admin” user exists, then only that user may add a new user.

Expected Extensions

Here are some potential extensions to the basic application.

- Allow users to add new products to the system.
- Add *administrator* role and access.
  - Only the administrator may request and view overall statistics from the system.
  - Only the administrator may add new users.
  - Only the administrator may add new products to the system.
- Enhance the efficiency of producing summary ratings for a particular product.
- Make suggestions for friendships that are based on the similarity between two users' ratings.
- Implement the database using a DBMS such as SQLite, Java DB (Derby), Berkeley DB JE, or MySQL.

Deliverables

Vision & Plan

- Team members' names; Team name; Product name
- Vision statement (must include a context diagram)
- Life-cycle model and rationale for its adoption
- Plan that includes concrete milestones

Designs

- System architecture
- User interface design
- API Specification
- User interface demonstration prototype

Implementation

- Level 1: Users can rate products and new users can be added to the system. The application can display a list of products, sorted by average rating.
- Level 2: The project is a working, maintainable, installable application that runs correctly. It allows a user to rate products and also makes suggestions about other products that user might like by displaying a list of the top 6 recommendations.

Documentation (Team)

- Documentation of the quantity and distribution of contributions (by team member & activity)
- Installation, operation, and maintenance information
- Test plan and test cases
- List of at least 4 problems encountered during development, along with solution details for each problem

Documentation (Individual)

- Description of what was learned and discovered during planning activities
- Description of what was learned and discovered during design activities
- Reflection, observations, and insights gleaned during the project
- Peer and self assessment of quantity and quality of contributions

Presentation (Team)

- A presentation of the application and the project documentation by the project team to the instructor. The presentation must
  - address which algorithms and components were developed by each team member
  - reflect upon the management of the project and individual contributions
  - demonstrate the application created by the team
- Note that all individuals may be required to:
  - discuss the algorithms and components they developed
  - reflect upon the management of the project and their individual contributions to project management
  - demonstrate knowledge of the software created by the team