CS314 Assignment 2: Team Development of a Music Sharing Management System

Due Date: Midnight, October 31, 2013. Submit via RamCT

Assignment Objective

The objective is to give students experience in (1) team development of a medium sized software system, and (2) applying modeling techniques. You must work in a team no smaller than 4 persons and no larger than 5 persons.

Assignment Tasks

You are required to do the following in this assignment:

- Form a team of no larger than 5 persons and no less than 4 persons to design and implement a music sharing system. The problem statement for the system is given below.
- Develop a requirements model for the system. The requirements model must consist of (1) a requirements class model, and (2) use cases for the required major services to be provided by the software.
- Develop a design model for the system. The design model must consist of (1) a design class model, and (2) sequence diagrams that show how each use case is implemented in the design.
- Develop a Java implementation of the design.

Deliverables

You are required to submit the following:

1. A requirements document that consists of
    a. A requirements model: The requirements model must consist of (1) a requirements class model, and (2) use cases for the required major services to be provided by the software.
    b. A statement of all the assumptions you made about the required behavior. The assumptions must not contradict any of the requirements in the problem statement. Each assumption must be presented in the following format:
    Your question: Your answer (the assumption).
    An example of an assumption is given below
    Question: What happens when a friend requests to borrow a song when the owner is

playing the song? Ans: The request is queued. The queue is checked for loan requests after the song has finished playing.

2. A design document that consists of

    a. A design model: The design model must consist of (1) a design class model, and (2) sequence diagrams that show how each use case is implemented in the design.

    b. Design justification: Make the case that your design is of good quality.

3. A Java implementation.

4. Review of each of your team members. A form will be made available via RamCT. Each student must prepare a review of each team member's performance. Include the tasks that the team member was responsible for and how well they fulfilled their responsibilities. Send the reviews to cs314@cs.colostate.edu. **Do not upload them to RamCT as part of your assignment**.

You must submit the Java implementation as an executable JAR file via RamCT.

You must submit electronic forms of the requirements and design documents via RamCT (there will be a separate RamCT dropbox for these documents). You must also submit readable hardcopies of these documents in class on October 31, 2013.

Acknowledgements

The following statement is a slightly modified version of a problem statement written by Trevan Hombs, a past CS517 student. Trevan developed and analyzed a formal design for this system as part of his CS517 project.

Problem Statement

The Music Sharing Management System (MSMS) allows users on a computer to share their music. The MSMS allows users to loan copies of songs to other users in the same manner as a physical copy of music can be loaned. For example, if a friend lends you hard copy of the music on a CD, that hard copy is no longer in the possession of the friend (the owner). This would create a social aspect to music listening that has been lost in the digital age. The MSS would have the following features:

1. Users can establish *friend* relationships with other users in the system. To set up a friend relationship, a user must first invite another user in the system to be a friend. If the invited user accepts then a friend relationship is set up between the two users.
2. Users can browse their friends' music libraries and request to borrow their friends' music.
3. A user can make their entire library available for browsing by anyone in the system or available for browsing by friends only.
4. Users can add new songs, remove songs, search for friends with songs, search for songs within friends, and search for songs outside their circle of friends.
5. Users can organize their songs into playlists. A user can also choose to display their

library sorted by artiste, title, or by genre. Songs that have been loaned out should be marked as not available for playback.

6. There shall be a limit as to how many times a user can borrow the same song from the same friend. There will be a default limit of 3 times, but the owner can change this limit.
7. If a song is loaned to a friend, the owner (the user that owns a song) cannot lend the song to anyone else, nor can the owner play/listen to the song. When a song is currently borrowed, the owner does not technically have that song in his/her library. What I mean by technically is that the borrower can listen to it and the owner cannot. However, anytime the owner wants the song back, the owner can take it back with no questions.
8. Friends could get on a waiting list to borrow a song.
9. Owners can specify how long they will allow a friend to borrow a song. The length can be in units of time or units of song listens.
10. Owners can specify, song-by-song, whether a song has to be approved for borrowing or can be borrowed by friends without approval. Owners can also mark songs as not borrowable.

Each song must have at least the following metadata associated with it:

Song Metadata

1. Name: Name of the song, e.g., “Maria”
2. Artist: Name of performer, e.g., “Branford Marsalis”
3. Album: Name of album containing song, e.g., “Steep Anthology”
4. Year: Year in which song was released, e.g., “2004”
5. Composer: Names of composers. e.g., “Leonard Bernstein, Stephen Sondheim”
6. Genre: Class of music that song belongs to, e.g., “Jazz”

When the system first starts up it reads a file containing data on users and their libraries, and on friend relationships. For example, consider an initial system that consists of three users with user names jweston, mkelly, jbilal, where jweston has password musicQ, mkelly has password sean123, and jbilal has password dean543musiq. In this system jweston is friends with mkelly, and mkelly is also friends with jbilal. Below is a file containing information about the music libraries of these 3 users that will be used to initialize the MSMS.

jweston, musicQ [Maria, Branford Marsalis, Steep Anthology, 2004, Leonard Bernstein Stephen Sondheim, Jazz | Jodys Grind, Bob James, , , , Jazz](mkelly)

mkelly, sean123 [The Bass Man, Marcus Wooten, In Bass Clef, 2013, R Crunk, Alternative | My Way, Uber Daft, Going Past the Future, 2012, , Alternative |The Reason, Heather Headley, ,2012, , R&B](jweston, jbilal)

jbilal, dean543musiq [Breakdown, Gary Clark, Blak and Blu, 2013, Gary Clark, Rock](mkelly)

From the above, each line in the input file contains information for a single user. The information on each line is structured as follows:

Username, Password [song metadata where each is separate by "|"; the metadata items are listed in the order shown under **Song Metadata** shown above] (friends of user)

Note that each song must have a name and artist, but the other information (metadata) is optional.

Notes

1. **System:** For the purposes of this assignment, assume this is a system that resides on a single machine (not realistic, but it avoids you having to develop a distributed/network-based application). Note that the libraries of all users are available for permissible browsing, searching and loaning once the system is running (e.g., a user does not have to be logged in for another user to browse the user's library or to send a request to borrow a song). A user must be logged in to send requests and respond to requests to borrow songs. More than 1 user can be logged in at any given time. You should have a feature that allows the graders to switch users on a single machine so that they can test your software with multiple users logged in.
2. **Out of scope features:**
    a. Your system is not required to actually play songs.
    b. Your system is not required to support buying of songs.
    c. You are not required to implement a distributed system with client and server software.
    d. You are not required to store (persist) the data in a database. You may have a feature that saves the current state of user libraries in a flat file before the program exits. This flat file that could be read in as input when the program is next started. Note that this is not required.
    e. You are not required to support downloading of songs on devices, to the cloud, etc. (you will have the opportunity to explore this extension in Assignment 3).
3. **User Interface:** You are free to design an appropriate user interface. The quality of your user interface will be graded. Specifically, we will count the number of steps needed by the user to complete a task, and we will evaluate how you guide the user in accomplishing his/her tasks (are the interface labels meaningful? Is the feedback given to the user informative?).
4. **Questions & Assumptions:** As usual, the requirements may be incomplete, ambiguous, etc. You can make assumptions about what is required if the information is not stated explicitly in the requirements (this means that you do not have to send questions to the TA). These assumptions must not contradict nor negate any of the requirements stated explicitly. You must state all your assumptions in the requirements document. Each assumptions must be written as a *question:answer* pair. See the requirements document description in the Deliverables section for more information.
5. **Read the Assignment Logistics document:** A document giving guidelines for Assignments 2 and 3 is available on RamCT in the Assignment 2 section.