Intermediate Submission 1 Group 49

Miguel Lebrun, Kiril Aleksiev, David Mandado, Marko Spyrou, Victor Handzhiev, Stoyan Meshov
Eindhoven University of Technology, The Netherlands
2111152, 2108399, 2138549i, 2106426, 2159384, 1978276, Group 49

1 INTERMEDIATE SUBMISSION 1

1.1 Project Idea

The project consists of a desktop application that allows users to search for songs, albums and artists (reviewables), rate them, and write reviews. Users can log into an account, browse reviewables, and share their thoughts through comments and star ratings as well as following other users to see their reviews. Reviews are displayed under reviewables, and a users reviews can be seen under their profile, making it easy to discover opinions from other users. The user is also recommended songs similar to their highly rated ones.

1.2 Use-Case Diagram

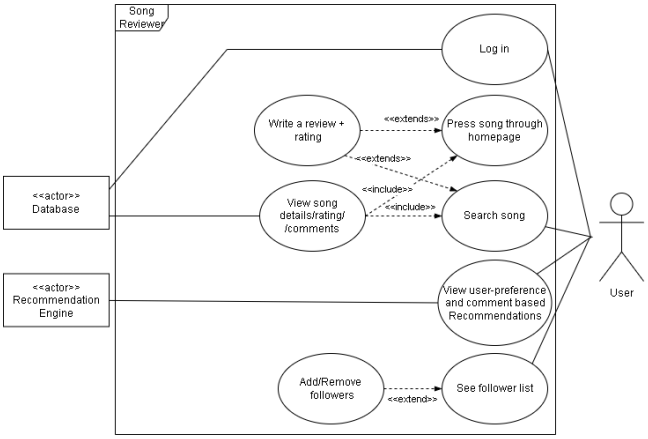

Figure 1: Revised use-case diagram: plain association lines, *User* directly linked to *Search Song*, and social/recommendation features separated into distinct use cases.

Figure 1 shows the interactions between the user and the system's primary functionality. The user can search for reviewables and write a review + rating. Additional features include viewing a reviewable's details, following other users, and getting personalized recommendations. Some actions, like writing a review, require first searching for or selecting a song. The system communicates with a mocked database for all song metadata.

1.3 Use-Case Description

The primary use case is "Write a Review," since it integrates multiple interactions and reflects the core purpose of the application. Table 1 below fully describes both main, alternative, and exception flows.

1.4 Requirements

Functionality requirements:

(1) The system must store all user, review and reviewable metadata persistently in CSV files located in `src/main/resources/db/`.
(2) The system must be able to read and parse data from these CSV files to display accurate information in the application.
(3) After a successful login/sign up, the system must display personalized reviewable recommendations based on user reviews and ratings.
(4) The system must allow users to submit a star rating (1-5) and a text review for any displayed reviewable.
(5) The system must display all existing reviews for a reviewable, sorted in descending order by number of likes.
(6) The system must allow users to follow/unfollow other users.
(7) Users must be able to search for reviewables by name.

Quality requirements (SMART):

(1) **Search latency:** When a user searches for a reviewable, search results must appear within ≤ 4 seconds under average conditions (Intel i5 or equivalent).
(2) **Review responsiveness:** The system must display the submitted review within 3 seconds of clicking "Submit".
(3) **Responsive layout:** The GUI must render and remain interactive across the the main desktop platforms (Windows 10/11, MacOS, Debian-based Linux (Ubuntu)).
(4) **Startup time**: The application must launch and become interactive within 10 seconds on a typical consumer laptop (Intel i5 or equivalent)

Constraints:

(1) The system must use CSV files as tables for persistence storage, simulating a database located in `src/main/resources/db/` without relying on any cloud-based back-end services.
(2) The system must be implemented as a JavaFX desktop application using maven and must not include external dependencies (libraries).

Table 1: Use Case Description – *Write a Review*

Identifier	UC-Review-001
Name	Write a Review
Authors	Miguel Lebrun, Kiril Aleksiev, David Mandado, Marko Spyrou, Victor Handzhiev, Stoyan Meshov
Priority	Must
Summary	A user submits a star rating (1–5) and text review for a selected song.
Actors	User, Database
Trigger	The user initiates the "Write a Review" action on a song.
Preconditions	The song exists in the Database.
Postconditions	The review is persisted and associated with both the song and the user.
Result	The submitted review (rating + comment) is stored in the database persistently. The review is displayed under the reviewable it is associated with.
Main Scenario	(1) User logs/signs into the application. (2) User searches for or selects a reviewable. (3) System retrieves reviewable metadata from mocked database. (4) User clicks into the "Write Review" dialog. (5) User enters a star rating and writes a text comment. (6) System saves the review to the database.
Alternative Scenarios	2a. User selects a reviewable from home page "Recommendations" instead of typing a query. 2b. User selects a reviewable from the profile of another followed user instead of typing a query.
Exception Scenarios	1e. User inputs erroneous login data → Show a notification to the user with an explanation of the error. 6e. Database I/O exception on save → Alert the user that the review wasnt saved.
Requirements	F1, F2, F3, F4, F5, F6, Q1, Q2, Q3, C1, C2