Software Requirements and Design Document

For

Group 8

Version 1.0

Authors:
Cameron B
Analia C
Ivan
Daniel
Abbi M

1. Overview (5 points)

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

This project is a website designed to serve as an interactive platform for browsing, ranking, and saving music albums. Users will be able to search for albums by name and access detailed information, such as artist, tracklist, and genre. The platform will support secure login with profiles for users and administrators. When users log in, they will be able to browse a collection of albums, view cover art and titles, and navigate to specific pages with further details and ranking options. Albums can be ranked and saved to individual user libraries, making it easy to track favorites.

2. Functional Requirements (10 points)

*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

1. High: Display home page with login button and search button.
2. High: Allow the user to enter the name of an album and get information about the album such as the artist, tracklist, and genre. The user should also be able to rate said album and save it to their library.
3. High: Allow correct login information to access unique profiles for both users and admin roles
4. High: Display browsing page upon successful login with album covers, artists, and titles
5. Medium: Connect each album to a page with further information and option for users to rank.
6. Low: Logout option

3. Non-functional Requirements (10 points)

*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

1. The database should only be available to admins and users should not be able to see confidential information like other user's passwords.
2. Only real albums should be able to be added to our database/shown to the users.
3. Allowing only people with verified emails to create an account.

4. Use Case Diagram (10 points)

*This section presents the **use case diagram** and the **textual descriptions** of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*
***Textual descriptions of use cases**: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*

5. Class Diagram and/or Sequence Diagrams (15 points)

*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the***

entire system and Sequence Diagrams for the three (3) most important use cases in your system.

*If the main **paradigm** in your system is **not** **Object Oriented** (i.e., you **do not** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for all the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). All the **relationships between classes and their multiplicity** must be shown on the class diagram.*

*A **Sequence Diagram** simply depicts **interaction between objects** (or **functions -** in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

6. Operating Environment (5 points)

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

7. Assumptions and Dependencies (5 points)

List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.