**SE LAB PROJECT: Song Recommendation System**

**Semester: 5**

**Section: C**

**Subject: Software Engineering**

**Team Members**

Anup Pai (220905424) -Roll No: 45

Anish Sahu (220905522) -Roll No: 56

Aarav Bhandari (220905448) -Roll No: 47

Neeraj Kumar (220905536) -Roll No: 57

**Table of Contents:**

|  |  |
| --- | --- |
| Problem Statement | 3-3 |
| SDLC Model | 4-5 |
| SRS | 6-8 |
| DFD | 9-10 |
| Structure Chart | 11-11 |
| OO Design Diagram | 12-15 |
| Result | 16-17 |

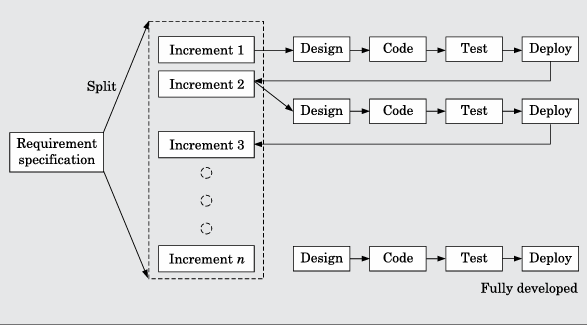
**Problem Statement:**

In recent user satisfaction surveys, less than 30% of users reported being satisfied with the music recommendations provided by the platform. This reflects a significant decline of 20% compared to the same period in the previous year.

The platform aims to increase the satisfaction rate by achieving a recommendation accuracy that resonates with over 90% of its users. However, achieving this goal will require the development of advanced recommendation algorithms, integration of user behavior analysis, and implementation of real-time personalization within the recommendation system.

**Software Development Life Cycle Model**





**Reasons:**

1. Progressive Feature Development: Thanks to the incremental implementation, the developers can introduce elementary features in advance, such as a recommendation engine based purely on genre or average ratings, and more complex things, such as content-based recommendations or collaborative filtering, can be introduced in the course of the project.
2. User feedback integration: Again, the models allow for the rapid deployment of changes, which means it is easy to gather user feedback and change the recommendations according to real user behavior. This is important for recommendation systems to adjust for users' specific preferences.
3. Risk Management and Early Testing: In this model, the system can be tested and updated at every stage of development, thus greatly reducing the possibility of developing fatal flaws. In this way, the developers will be able to validate that the basic recommendation engine is operational before proceeding with other advanced recommendation capabilities.
4. Efficient Use of Resources: The steps of inceptor development focus on the crucial features straight away, enabling effective resource use. This minimizes time wasted in lengthy discussions of what will be done in the future and emphasizes users' changing demands and tastes.
5. Enhanced Scalability: Since it is easier to make the system gradually overextend as opposed to starting with the complete system which makes the system rigid, it is advantageous in instances when more music sources or recommendation algorithm inputs will be needed.

**Software Requirement Specification**

**Purpose:**

Design and develop a song recommendation system which will recommend to users what songs they need based on their preferences, the history of songs they heard, or their mood. The source requirement specification documents all details about features that need to be designed or developed as well as some requirements needed for the total functionality of the system while working with it.

**Project Scope:**

It will allow users to input their preferences based on favourite genres, artists or moods, and thus will provide them with a recommendation. It would also include functionality for song filtering, behaviour tracking, and user likings to further enhance the user experience.

**Environmental Characteristics:**

The system will operate on diverse platforms and devices, from computers, and smartphones to tablets. Its sources of information for such songs to be played off will come as an external feed from its music databases.

**Overall Description**

* *Product Perspective:*

The song recommendation system is designed to be a standalone application that offers users music recommendations based on preferences and behaviour. The system will integrate with external data sources, such as music streaming platforms.

* *Product Features:*

1. Personalized song recommendations.
2. User feedback via song likes and ratings.
3. Optional mood-based song suggestions.
4. Ability to filter songs by genre, artist, or mood
5. Ability to add songs to a playlist

* *User Classes:*

1. General Users: Individuals seeking song recommendations tailored to their likes and interests .
2. Administrators: Responsible for maintaining the system, including data management.

* *Operating Environment:*

The application will run on web and mobile platforms, supporting common operating systems like Windows, Android, and iOS.

* *User Documentation:*

A user guide will be provided, explaining how to interact with the system, including searching a song , filtering songs, and providing feedback.

**External Interface Requirements**

* *User Interfaces:*

The interface will be user-friendly and include options to search, filter, and rate songs. Features such as "Play Next Song" and "Like" buttons will be present, allowing seamless interaction with recommendations.

* *Hardware Interfaces:*

The system will be compatible with standard hardware devices, including desktops, laptops, smartphones, and tablets.

* *Software Interfaces:*

It will interface with external music databases (Spotify, YouTube Music) or a custom playlist for retrieving song data.

* *Communications Interfaces:*

Internet connectivity will be required to pull data from music databases and track user behaviour.

**Non-Functional Requirements**

* *Performance Requirements:*

The system will provide song recommendations within 2-3 seconds based on user inputs like filters and preferences.

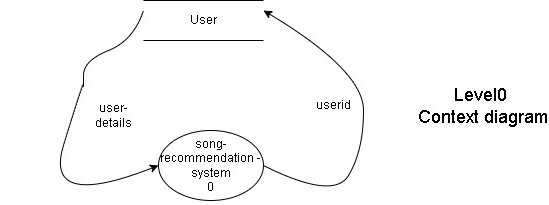
* *Safety Requirements:*

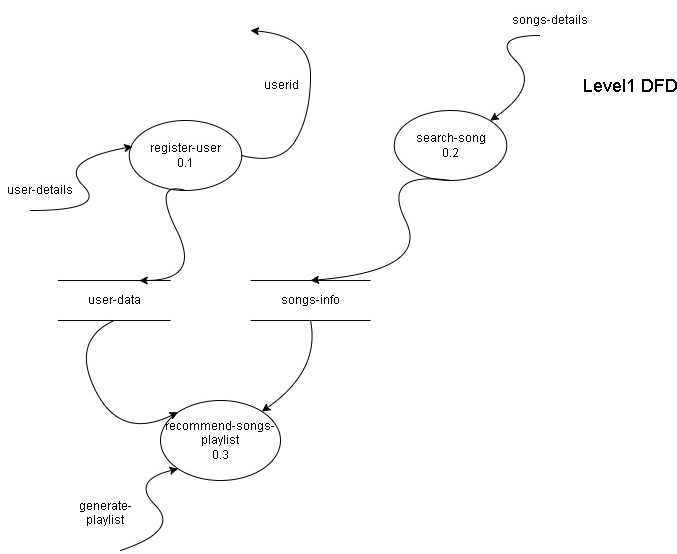
User data, including listening history and preferences, must be securely stored to prevent data loss or unauthorized access.

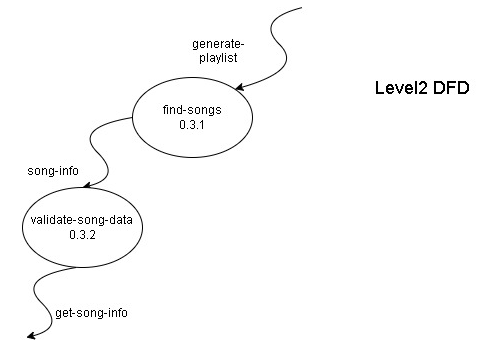
* *Security Requirements:*

Data will be encrypted and stored for purposes of user information, having authenticating mechanisms that are secured to avail personalized features like playlists and recommendations.

**Data Flow Diagram:**



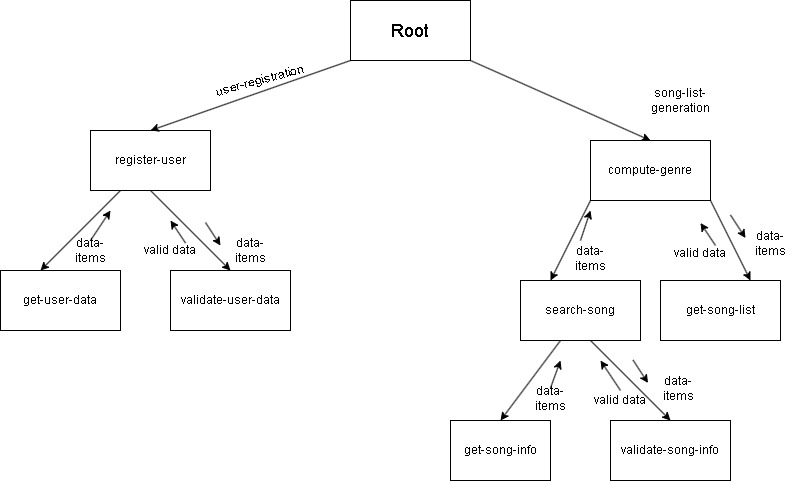




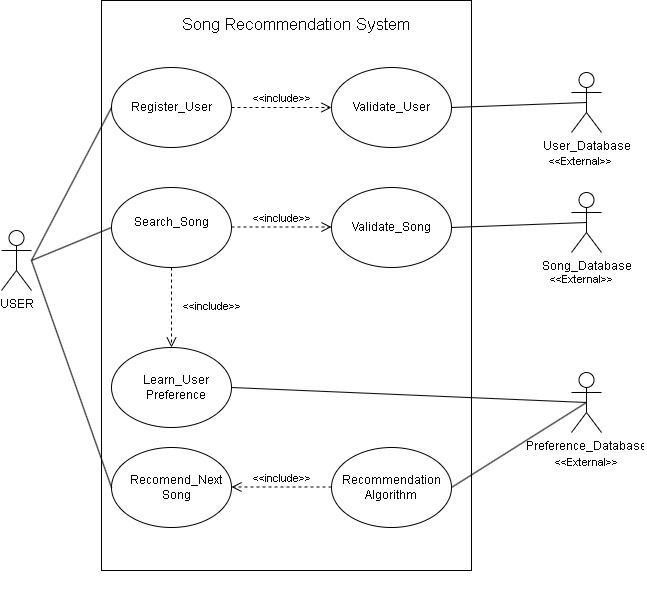
**Data Dictionary for above DFD:**

* userid : integer
* user-details : {userid+name}
* song-details : {songid+songname+genre}
* generate-playlist: {song-details}\*
* get-song-info: song-details

**Structure Chart:**



**Object Oriented Design Diagrams:**

1. **Use Case diagram on Song Recommendation System:** 

**Text description:**

**U1: register-user:** Using this use case, the user can register himself by providing the necessary details.

**Scenario 1: Mainline sequence**

1. User: Select register user option

2. System: display prompt to enter First name, Last name, password and email

3. User: enter the necessary values

4. System: display the generated id and the message that the User has successfully been registered.

**Scenario 2: At step 4 of mainline sequence**

4. System: displays the message that the User has already registered.

**Scenario 3: At step 4 of mainline sequence**

4. System: displays message that some input information have not been entered. The system displays a prompt to enter the missing values.

**U2: Search Song: This use case allows the user to search for songs in the system**.

**Scenario 1: Mainline Sequence**

1. User: Selects the Search Song option.

2. System: Displays a prompt to enter song title, artist, or genre.

3. User: Enters search criteria.

4. System: Searches for songs matching the criteria and displays the results.

5. User: Selects a song from the results to view more details.

**Scenario 2: At step 4 of mainline sequence**

4. System: Displays a message indicating that no songs were found matching the criteria.

**Scenario 3: At step 2 of mainline sequence**

4. System: Displays a message indicating that the input is invalid and prompts the user to enter valid search criteria.

**U3: Learn User Preference:** This use case captures user preferences based on their song searches to enhance recommendations.

**Scenario 1: Mainline Sequence**

1. User: Searches for a song using the Search Song use case.

2. System: Records the searched song and user’s selection.

3. System: Updates user preferences based on the search and selection data.

**Scenario 2: At step 2 of mainline sequence**

2. System: Displays a message indicating that no preferences could be learned from the search (e.g., if no songs were selected).

**Scenario 3: At step 3 of mainline sequence**

3.System: Displays an error message indicating that there was a problem recording preferences.

**U4: Recommend Song:** This use case provides song recommendations based on learned user preferences.

**Scenario 1: Mainline Sequence**

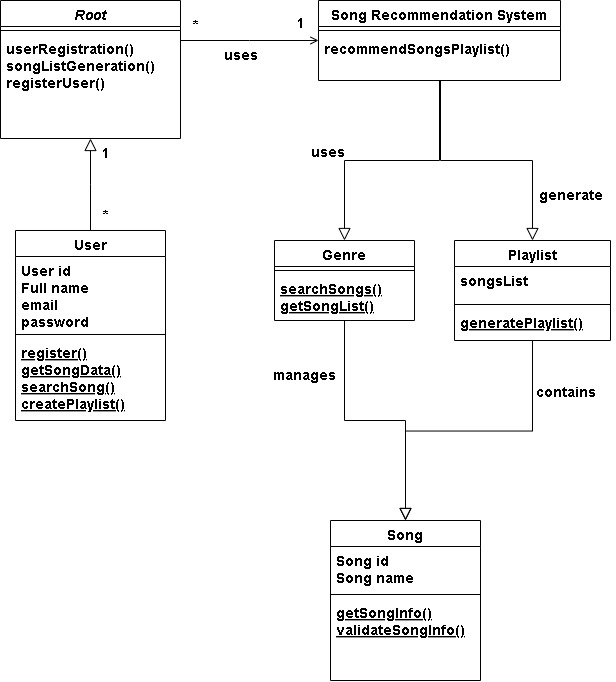
1. User: Selects the Recommend Song option.

2. System: Analyzes recorded user preferences.

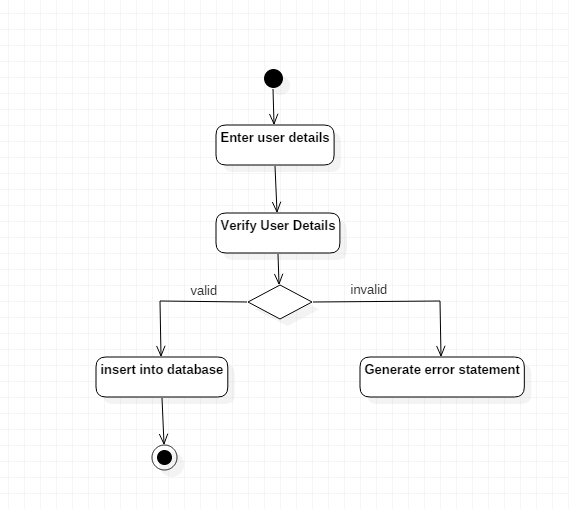
3. System: Displays a list of recommended songs based on preferences.

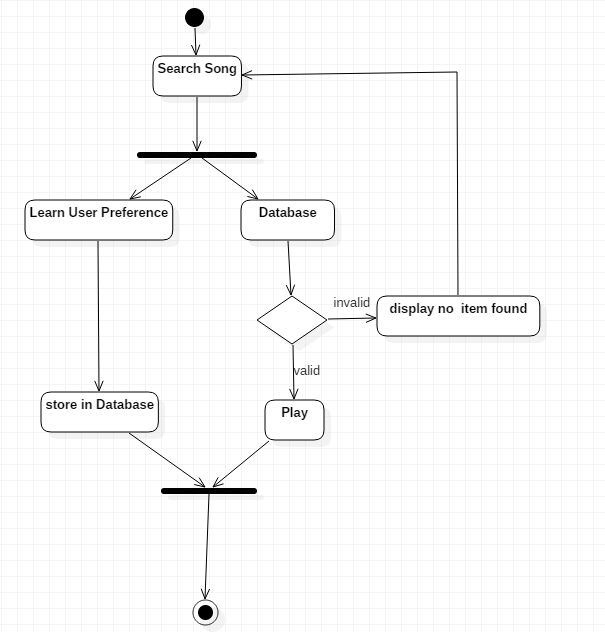
4. User: Selects a recommended song to listen to.

1. **Class Diagram**

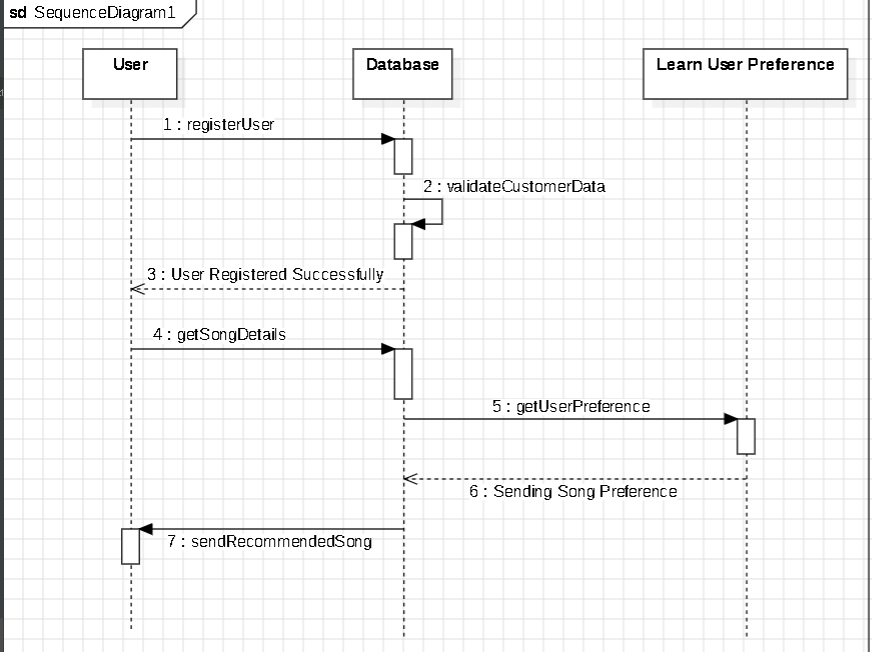


1. **Activity Diagram:**

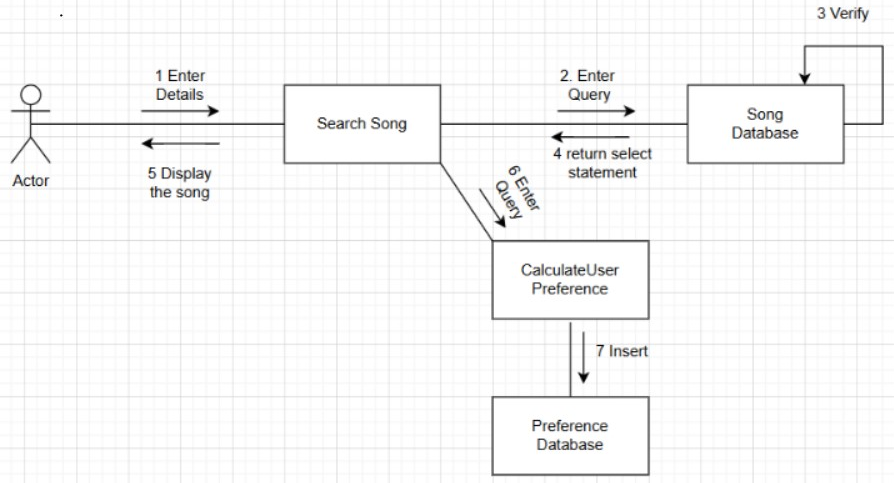




1. **Sequence Diagram:**



1. **Collaboration Diagram:**



User

**Result:**

