### **SOFTWARE REQUIREMENTS SPECIFICATION(SRS)**

**Link to wiki Page:** [**https://github.com/ShivaniPandula/MusicMix\_GDP-05/wiki/Software-requirements-specification-(SRS)**](https://github.com/ShivaniPandula/MusicMix_GDP-05/wiki/Software-requirements-specification-(SRS))

### **Project Information**

# Project Charter

# Project Title: MusicMix

* **Project Start Date:** 04/30/2024
* **Projected Finish Date:** NA
* **Budget Information:** $100,000
* **Project Manager:** Srinadh Bokkisam

## **Project Objectives:**

* Current platforms like Gaana lack certain essential features that could greatly enhance user satisfaction and ease of use. These missing features not only diminish the attractiveness of the platform but also hinder users' ability to discover and enjoy music seamlessly.

## **Motivation for solving the problem:**

1. **Enhanced User Experience:**
   * By providing a seamless and feature-rich music streaming platform, users can enjoy a more satisfying experience. This includes easy access to a vast library of songs, personalized playlists, and curated recommendations, all contributing to a more enjoyable music listening experience.
2. **Improved Music Discovery:**
   * Current platforms often lack robust music discovery features, making it challenging for users to explore new artists and genres. By offering curated playlists, radio stations, and personalized recommendations, MusicMix can help users discover new music tailored to their tastes, leading to increased engagement and satisfaction.
3. **Convenience and Accessibility:**
   * Offline playback support ensures that users can enjoy their favorite music even without an internet connection, catering to users who may not always have access to reliable internet services. This convenience enhances the accessibility of the platform and allows users to enjoy uninterrupted music playback anytime, anywhere.
4. **Personalization and Engagement:**
   * Integration with music recommendation algorithms enables MusicMix to offer personalized music suggestions based on users' listening history and preferences. This personalization fosters a deeper connection between users and the platform, increasing user engagement and retention.
5. **Security and Privacy:**
   * By prioritizing secure user authentication, data encryption, and protection against unauthorized access, MusicMix ensures that users' personal information and music preferences are safeguarded. This instills trust in the platform and reassures users that their data is being handled responsibly.

## **Functionality Requirements:**

* <https://github.com/ShivaniPandula/MusicMix_GDP-05/wiki/Functional-Requirements-List-(Iteration-2)>

## **Roles and Responsibilities:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name and Signature** | **Role** | **Position** | **Contact Information** |
| Srinadh Bokkisam | Business Analyst | Project Manager | [S565718@nwmissouri.edu](mailto:S565718@nwmissouri.edu) |
| Sayendranadh Chowdary | Developer | Software Engineer | [S564837@nwmissouri.edu](mailto:S564837@nwmissouri.edu) |
| Shivaram Reddy | Developer | Software Engineer | [S566581@nwmissouri.edu](mailto:S566581@nwmissouri.edu) |
| Shivani Goud | Developer | Software Engineer | [S564911@nwmissouri.edu](mailto:S564911@nwmissouri.edu) |
| Raj Kumar | Quality Assurance | Team Member | [S567114@nwmissouri.edu](mailto:S567114@nwmissouri.edu) |
| Sathvik Reddy | Quality Assurance | Team Member | [S566920@nwmissouri.edu](mailto:S566920@nwmissouri.edu) |

# About the Developers

**GitHub UserName:** ShivaniPandula **Link to GitHub:** <https://github.com/ShivaniPandula>  
**Strengths:** Attention to detail: Being meticulous helps in writing clean, error-free code and debugging.  
**Interests:** Programming in various languages such as Python, Java, C++, etc., to develop software, applications, or websites.

**GitHub UserName:** Shivaramreddypalla  
**Link to GitHub:** <https://github.com/Shivaramreddypalla>  
**Strengths:** Proficient Android developer with 1.5 years of experience, skilled in Java and Kotlin, excelling in UI/UX design and problem-solving.  
**Interests:** Passionate about optimizing app performance and user experiences, eager to explore cross-platform development and engage with developer communities.

**GitHub UserName:** Srinadh1998-hub  
**Link to GitHub:** <https://github.com/Srinadh1998-hub>  
**Strengths:** Worked as an Angular developer for 2.8 years, Highly proficient in CSS & JavaScript, Little bit of Knowledge on reactJS.  
**Interests:** Creating real-time Webpages which are user friendly using AngularJS, Unit testing with Jasmine and karma, Playing Cricket.

**GitHub UserName:** rajkumarchigurupati  
**Link to GitHub:** <https://github.com/rajkumarchigurupati>  
**Strengths:** Worked as a Data analyst for 2 years  
**Interests:** Working with Datasets, Creating Apps.

**GitHub UserName**: sathvikrdy09  
**Link to GitHub:** <https://github.com/sathvikrdy09>  
**Strengths:** Worked as an SAP Developer 9 months, Highly proficient in Java and Dbms.  
**Interests:** Creating real-time Applications, Cooking Food and Playing Games.

**GitHub UserName:** sayendra99  
**Link to GitHub:** <https://github.com/sayendra99>  
**Strengths:** Worked as an Api Developer 10months, Highly proficient in Java,python,

Htmls,css,Bootstrap,Javascript ,Nodejs and Dbms.  
**Interests:** Creating real-time Applications,Exploring new Inovatives technologies and Playing Games.

**Problem Statement**

Current platforms like Gaana lack certain essential features that could greatly enhance user satisfaction and ease of use. These missing features not only diminish the attractiveness of the platform but also hinder users' ability to discover and enjoy music seamlessly.

**Identified Issues:**

**1. Missing Sections for Recently Played and Most Played Songs:**

Users often want to quickly access songs they've recently played or their most-played tracks. Including these sections would improve the navigational ease and user satisfaction.

**2. No Song Recognition by Audio:**

Have you ever heard a song playing somewhere and wished you could instantly identify it? Gaana lacks a feature for song recognition based on audio input, depriving users of a convenient tool for discovering new music.

**3. Limited Recommended Songs:**

Gaana's recommendation system falls short in providing personalized suggestions based on users' listening habits. This limitation prevents users from discovering new favorites tailored to their preferences.

**Impact on User Experience:**

1. **Harder to Discover New Music**: The absence of features like lyric search and song recognition impedes users' ability to explore and discover new music effortlessly.
2. **Less Interactive Experience**: Gaana's lack of voice search and sections for recently played and most played songs diminishes the interactive and intuitive nature of the platform's navigation.
3. **Missed Opportunities for Personalization**: Without robust recommendation capabilities, users may feel that they are not receiving tailored music suggestions, leading to a less personalized experience.

**Conclusion:**

By addressing these identified issues and implementing innovative solutions, Music Mix aims to revolutionize the music streaming experience, making it more engaging, intuitive, and personalized for users. Join us on this journey to unlock the full potential of music discovery and enjoyment.

### **Design**

# Use Cases

**Use Case 1: User Registration**

* **Goal:** A new user wants to register an account on the MusicMix web application.
* **User Input:** User provides personal information (name, email), chooses a username and password.
* **System Actions:**
  + The system validates the user's input data.
  + The system creates a new user account.
* **Output:** User receives a confirmation message, and their account is registered.

**Use Case 2: User Login**

* **Goal:** A registered user wants to log in to their account.
* **User Input:** User enters their username and password.
* **System Actions:**
  + The system verifies the user's credentials.
  + If valid, the system grants access to the user's account.
* **Output:** User gains access to their account dashboard.

**Use Case 3: Music Playback**

* **Goal:** A user wants to stream music tracks, albums, or playlists.
* **User Input:** User selects a song, album, or playlist for playback.
* **System Actions:**
  + The system retrieves the selected music content.
  + The system streams the music with high-quality audio.
* **Output:** User can listen to the selected music content.

**Use Case 4: Playlist Creation**

* **Goal:** A user wants to create a personalized playlist.
* **User Input:** User selects songs to add, edit , Delete or existing playlist.
* **System Actions:**
  + The system allows the user to create a new playlist or select an existing one.
  + The system adds the selected songs to the playlist.
* **Output:** User's playlist is created or updated with the selected songs.

**Use Case 5: Music Discovery**

* **Goal:** A user wants to discover new music through curated playlists, radio stations, or recommendations.
* **User Input:** User explores featured playlists, trending tracks, or genre-based recommendations.
* **System Actions:**
  + The system presents curated playlists, trending tracks, and genre-based recommendations.
  + The system suggests new music based on user preferences and listening history.
* **Output:** User discovers new music content to explore and enjoy.

**Use Case 6: Offline Playback**

* **Goal:** A user wants to listen to music without an internet connection.
* **User Input:** User selects music for offline playback.
* **System Actions:**
  + The system allows the user to download selected music content for offline listening.
  + The system stores downloaded music locally on the user's device.
* **Output:** User can listen to downloaded music offline, without requiring an internet connection.

**Use Case 7: Account Settings Update**

* **Goal:** A user wants to update their account settings and preferences.
* **User Input:** User edits their account information, notification settings, or password.
* **System Actions:**
  + The system validates and updates the user's account settings.
* **Output:** User's account settings and preferences are updated successfully.

**Use Case 8: Social Sharing**

* **Goal:** A user wants to share music with friends or followers on social media platforms.
* **User Input:** User selects music content to share and chooses a social media platform.
* **System Actions:**
  + The system generates a shareable link or post with the selected music content.
  + The system posts the shareable link or content to the chosen social media platform.
* **Output:** User's selected music content is shared with friends or followers on social media.

**Use Case 9: Search and Filtering**

* **Goal:** A user wants to search for specific music content and apply filters for refined results.
* **User Input:** User enters search keywords and selects filtering options.
* **System Actions:**
  + The system searches for music content matching the entered keywords.
  + The system filters search results based on user-selected criteria (e.g., artist, genre).
* **Output:** User sees refined search results matching their criteria.

**Use Case 10: Admin User Management**

* **Goal:** An administrator wants to delete a user account that is no longer needed in the system.
* **User Input:** Administrator accesses the user management section and selects the user account to be deleted.
* **System Actions:**
  + The system prompts the administrator to confirm the deletion action to prevent accidental removal.
  + The system removes the selected user account from the database, including any associated permissions and access rights.
* **Output:** The administrator receives confirmation that the user account has been successfully deleted from the system, and the user will no longer be able to log in or access any resources.

**Use Case 11: Admin Content Management**

* **Goal:** An administrator needs to manage the music content in the application.
* **User Input:** Administrator accesses the music management section in the admin dashboard.
* **System Actions:**
  + The system provides options for adding, deleting, or editing music content, including tracks, albums, and playlists.
* **Output:** The administrator receives confirmation of successful actions, ensuring that the music content in the application is effectively managed according to their requirements.

**Use Case 12: User Feedback**

* **Goal:** Users want to provide feedback, suggestions, or report issues to the MusicMix platform administrators.
* **User Input:** Users navigate to the feedback section of the application and submit their feedback.
* **System Actions:**
  + The system provides a dedicated feedback form where users can input their comments, suggestions.
* **Output:** Users feel heard and valued as their feedback is acknowledged and addressed by the MusicMix platform administrators, leading to improvements and enhancements based on user input.

**UseCase Diagram:**

A diagram of a person's relationship

Description automatically generated

# Functional Requirements List

**Customer Functional Requirements:**

1. **User Authentication and Registration:**
   * The system MUST provide a user registration process.
   * The system MUST allow registered users to log in.
   * The system SHOULD provide a "Remember Me" option for user convenience during login.
2. **Password Reset:**
   * The system MUST allow user to reset their passwords securely.
   * Password reset emails MUST be sent to the customer's registered email address.
3. **User Profile Management:**
   * The system MUST enable users to create and update their profiles.
   * Users MUST be able to view and edit their personal information (e.g., name, address) in their profiles.
   * Users SHOULD be able to upload profile pictures.
4. **Music Discovery and Playback:**
   * Users MUST have access to a vast library of music content, including Artists, albums, and playlists.
   * Playback controls for playing music tracks, along with viewing additional details such as album artwork, SHOULD be provided.
5. **Playlist Creation and Management:**
   * Users MUST be able to create personalized playlists by adding tracks from the music library.
   * They MUST have the ability to edit, reorder, and delete playlists, with options for sharing with others.
6. **Social Interaction:**
   * Users SHOULD be able to like favorite artists, friends, and influencers to stay updated on their latest releases and activities.
   * Sharing music, playlists, and recommendations with friends through social media or within the MusicMix community SHOULD be facilitated.
7. **Personalized Recommendations:**
   * Users SHOULD receive personalized music recommendations based on listening history, preferences, and behavior.
   * Exploring curated playlists, trending tracks, and genre-based recommendations for discovering new music MUST be supported.
8. **Offline Playback:**
   * Users MUST have the capability to download music tracks, albums, or playlists for offline playback when the internet connection is not available.

**Admin Functional Requirements:**

1. **User Management:**
   * The system MUST allow administrators to create, edit, and delete user accounts.
   * User authentication logs and activity history MAY be accessible to administrators for security monitoring.
2. **Content Management:**
   * Administrators MUST be able to add, edit, or remove music content from the library, including tracks, albums, and playlists.
   * They MAY have the capability to modify metadata and artwork associated with music content.
3. **Support and Communication:**
   * The system MAY offer customer support tools for administrators to respond to user inquiries, feedback, and complaints.
   * Communication channels for announcements, updates, and service maintenance MAY be maintained by administrators.

# Non‐functional Requirements List

### **Performance:**

* **Streaming Efficiency**: Music tracks, albums, and playlists should stream with minimal buffering and playback interruptions, ensuring a smooth listening experience.
* **Response Time**: User interactions within the application, such as browsing, searching, and playback controls, should have a minimal response time for optimal usability.
* **Scalability**: The application should be scalable to handle increased traffic and data volume as the user base grows over time.

### **Security:**

* **Data Encryption**: All sensitive user data, including personal music preferences and session tokens, must be encrypted both in transit and at rest.
* **Authentication**: Secure authentication mechanisms should be implemented to protect user accounts from unauthorized access, including robust password hashing and protecting it.
* **Authorization**: Role-based access control should be enforced to ensure that users can only access content and features appropriate to their role.
* **Data Privacy**: Compliance with data protection regulations to safeguard user privacy and ensure transparent handling of personal data.

### **Availability:**

* **Continuous Availability**: The MusicMix application should be available 24/7 with minimal downtime for maintenance or updates, ensuring uninterrupted access to music content.
* **Fault Tolerance**: The application should gracefully handle errors and failures, preventing data loss and maintaining service availability even in the face of unexpected issues.

### **Usability:**

* **Intuitive Interface**: The user interface should be intuitive and user-friendly, with clear navigation and easy access to key features such as playlist creation, music discovery, and playback controls.
* **Mobile Responsiveness**: The application should be responsive and optimized for various devices, including smartphones and tablets, to provide a consistent user experience across platforms.

### **Code Documentation:**

* **Comprehensive Documentation**: The codebase should be well-documented to facilitate understanding and maintenance by developers, including inline comments, README files, and API documentation.

**Data Management Plan**

**Data Management Plan for MusicMix**: Summary of Data to Store:

**Users Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| User ID | Unique identifier for each user. |
| Song ID | Unique identifier for each song |
| Username | User's chosen username for the platform. |
| Email | User's email address for communication/login. |
| Password | Hashed and salted password for authentication. |
| Gender | Gender of the Person |
| DOB | Date of Birth of person |
| PhoneNumber | Phone Number of a Person |

### **Playlist Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Playlist ID | Unique identifier for each playlist. |
| User ID | Foreign key linking to the playlist creator. |
| Details | Details of the Playlist. |

### **Song Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Song ID | Unique identifier for each song. |
| Song Name | Name of the song. |
| Album ID | Foreign key linking to the album the song belongs to. |
| Artist ID | Foreign key linking to the artist who performed the song. |
| Genre | Genre classification of the song. |
| Lyrics | Text containing the lyrics of the song. |
| Duration | Length of the song in seconds. |

### **Album Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Album ID | Unique identifier for each album. |
| Album Name | Name of the album. |
| Release Date | Date when the album was released. |

### **Artist Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Artist ID | Unique identifier for each artist. |
| Artist Name | Name of the artist. |
| Bio | Biography or description of the artist. |

### **Admin Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Admin ID | Unique identifier for each admin user. |
| User Name | Username of the admin user. |
| Password | Hashed and salted password for authentication. |

### **Song Artist**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Song Artist ID | Unique identifier for each song Artist. |
| Song ID | Unique identifier for each song. |
| Artist ID | Unique identifier for each Artist. |

### **Album Artist**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Album Artist | Name of the Artist in the Album. |
| Album ID | Unique Identifier for each album. |
| Artist ID | Unique identifier for each Artist. |

### **Song Play List**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Song Play List ID | Unique identifier for the song playlist. |
| Song ID | Unique identifier for each song. |
| Play List ID | Unique identifier for each playlist. |

### **Feedback**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Feedback ID | Unique identifier for the Feedback. |
| Feedback Text | Textual content of the feedback provided by the user. |
| User Id | Foreign key linking to the User. |

A diagram of a computer

Description automatically generated with medium confidence

# Relationships

## **User - Playlist**

* One-to-Many relationship: One user can have multiple playlists.
* Foreign key: User ID in the Playlist table.

## **User - Song**

* Many- to-one relationship: Many Users can have single song.

## **Playlist - SongPlayList**

* one-to-Many relationship: A playlist can contain multiple songs, and a song can belong to multiple playlists.

## **Song - SongPlayList**

* One-to-Many relationship: A song can be in multiple SongPlayList and a Song can belong

## **Song - Album**

* Many-to-One relationship: Many songs can belong to one album.
* Foreign key: Album ID in the Song table.

## **Song - SongArtist**

* One-to-Many relationship: Many songs can be performed by one artist.
* Foreign key: Artist ID in the Song table.

## **Artist - SongArtist**

* One-to-Many relationship: An Artist can Sing many Songs.

## **Album - AlbumArtist**

* One-to-Many relationship: One album can be created by Many artist.
* Foreign key: Artist ID in the Album table.

## **Artist - AlbumArtist**

* one-to-Many relationship: one Artist Can be in many in AlbumArtist Table.

## **User - feedback**

* one-to-Many relationship: one user can have zero or many feedbacks.

# Initial Plans to Secure Data

## **Authentication and Authorization:**

* Implement secure authentication mechanisms like OAuth or JWT for user login.
* Use role-based access control (RBAC) to restrict access to certain features or data based on user roles (e.g., admin, regular user).

## **Data Encryption:**

* Encrypt sensitive data like passwords and any other personal information using industry-standard encryption algorithms (e.g., bcrypt for passwords).
* Ensure that data transmission between the client and server is encrypted using HTTPS.

## **Input Validation and Sanitization:**

* Implement input validation and sanitization to prevent SQL injection, XSS (Cross-Site Scripting), and other common security vulnerabilities.
* Validate user inputs on the client-side and server-side to ensure data integrity.

## **Database Security:**

* Apply database security measures such as using strong database credentials, restricting database access to authorized users only, and regularly updating the database management system (DBMS) to the latest version with security patches.

## **Backup and Disaster Recovery:**

* Regularly backup the database to prevent data loss in case of system failures, cyber attacks, or other unforeseen events.
* Implement disaster recovery plans to quickly restore the system and data in case of emergencies.

## **Logging and Monitoring:**

* Enable logging of user activities, database transactions, and system events to track suspicious activities and detect security breaches.
* Set up real-time monitoring tools to continuously monitor system performance and security metrics.

## **Security Training and Awareness:**

* Provide security training and awareness programs for developers, administrators, and users to educate them about common security threats, best practices, and compliance requirements.
* Encourage users to use strong, unique passwords and enable multi-factor authentication (MFA) for added security.

# Mapping of Functional Requirements to Data Storage

### **User Authentication Management:**

* Store user data (e.g., username, email, password hash) in the User table.
* Session tokens for "Remember Me" functionality can be stored in the User table or a separate Session table.
* Store password reset tokens and associated expiration timestamps in the User table.
* Store User profile information (e.g., name, address) in the User table.

### **Music Management:**

* Store music content information (e.g., artists, albums, playlists) in respective tables (Artist, Album, Playlist).
* Store user-created playlists and their track associations in a Playlist table.
* Additional playlist metadata (e.g. details) may be stored in the Playlist table.

### **Social Interaction Management:**

* Store user-follow relationships in a Follow table with references to the User table.

**List of Proposed Prototypes**

1. **User Registration Prototype:**
   * Mockup of the user registration form.
   * Screens for input validation and error handling during registration.
   * Confirmation screen upon successful registration.
2. **User Login Prototype:**
   * Mockup of the user login page.
   * Screens for authentication and error handling during login.
3. **Homepage with Music Categories Prototype:**
   * Design of the homepage layout displaying music categories.
   * Sample music category cards and navigation.
4. **Music Search and Listing Prototype:**
   * Mockups for the music search bar and search results.
   * Music listing screens with album artwork, descriptions, and playback options.
5. **Playlist Management Prototype:**
   * Mockups for the playlist management interface.
   * Screens for creating, editing, and deleting playlists.
6. **User Profile Management Prototype:**
   * Design of the user profile page.
   * Screens for editing personal information and profile picture.
   * Option to view and manage liked songs or playlists.
7. **Password Reset Prototype:**
   * Screens for initiating the password reset process.
   * Screens for generating and sending password reset emails.
   * Reset password form with validation checks.

### **Meeting Minutes**

# Client Meeting Minutes I

## **Who was present**

* Shivani Goud Pandula
* Shivaram Reddy Palla
* Srinadh Bokkisam
* Sayendranadh Chowdary Devabhaktineni
* Rajkumar Chigurupati
* Sathvik Cheruku

## **Meeting agenda**

* Discussing the problem Statement, Requirements, and proposed solution for the MusicMix.

## **Specific questions asked and Specific answers given**

* Srinadh: About the Interface Requirement and Requirements for the personalized recommendation
* Client Answer: Basic interface should be implemented and for recommendations suggested to have personalized songs based on region, genre etc.
* Shivani: Do we need the Application to have offline songs?
* Client Answer: Add a feature to download the songs
* Shivaram: Is SQL database is good enough?
* Client Answer: Depending on the size of your datasets decide your database like MongoDB, Oracle etc.
* Sayendranadh: Audio search advanced feature will be added
* Client Answer: Recommended a few search algorithms to be implemented.
* RajKumar: Lyrics will be displayed in the interface
* Client Answer: Suggested to add the Lyrics feature
* Sathvik: Description of the song should be displayed or not?
* Client Answer: Suggested to add the feature

## **Action items:** what have you decided you will show the client in your next meeting

* Pictorial Representation of our Interface

## **When is the next meeting:**

* Yet to be Decided