Functional Requirements

Feature Name:Create Accounts		Priority:High
Description: Using email and creating password would allow user to create their account.		
Inputs:User info, data	Outputs: Giving the user access to likes, comments,	Processes: User will go through a form that will ask for credentials such as an email, username,and password. Credentials will then be stored in a database.
Dependencies: User information(email, username, password), database		

Feature Name: Sharing Songs		Priority: High
Description: Users will be able to send a song to their feed, allowing other users to see it as a post.		
Inputs: Link, data	Outputs: A post created and shared to friends feed.	Processes: User will click ‘Plus’ button, will create a post, post a link, click send. Post will be created and will show to others feed.
Dependencies: Needs Spotify Web API		

Feature Name: Linking Spotify		Priority: High
Description: Users will be able to sign in to link their account with Spotify. Enables users to find music from spotify library and be able to have songs shared via spotify		
Inputs: User Spotify Account login	Outputs: Successful Link between Spotify	Processes: Button that will prompt user for login, Sign in with their login, successful login

Dependencies: Spotify Web API

Feature Name: Login page	Priority: High	
Description: before a user is able to use all of the websites' features, such as liking a post/piece of music(albums, songs, playlists), commenting on other user's posts, and looking at other user's profiles.		
Inputs: user credentials(email, username, password).	Outputs: comments being displayed once clicked, and access to collections(likes/favorites).	Processes: If a user attempts to interact with other users or music while they are not logged in, the website will direct them to the login page where they will either log in or be directed to create an account.
Dependencies:User data, database, backend APIs		

Feature Name: Main page	Priority: High	
Description: User will see Trending music from Spotify. This will show before the user has logged in. After logging in, User will see what they have liked, friends posts, and music they have selected. Have a Search Bar for looking for music, a 'add friend' button. The Main Feed that the user can scroll to see more posts. Trending section/page to show what is trending in music in general. Profile picture in top right.		
Inputs: Login Credentials	Outputs: Main feed	Processes: Login to User's account with the credentials. Automatically shown to main page
Dependencies: Database storing the login credentials. Access to main web page.		

Feature Name: Searching and adding other Users	Priority: High
Description: User will click the 'add friend' button. Type in the other user's name. Will output a list of users matching what the user typed. Click the symbol to add the other	

user. Symbol most likely to be a plus button.		
Inputs: inputs based on User's wording	Outputs: Lists of matching inputs and plus buttons next to names	Processes: User clicks 'add friend' button. Type in keywords (usually the other User's name/username), Looks at matching inputs in list. Click plus button.
Dependencies: Searching function, database, backend APIs		

Feature Name: Comments/likes	Priority: normal	
Description: when users see a post on their main feed, they should be able to comment and like the post		
Inputs: text, clicking on the like button	Outputs: displaying the new comment/like	Processes: user hits the like button or clicks on the comment button, creates a new comment, and
Dependencies: database, backend APIs		

Feature Name: Searching for songs	Normal	
Description: Users should be able to search for songs and/or other users directly from the main page		
Inputs: text	Outputs: A list of songs/accounts that match the input text	Processes: search bar, results list
Dependencies: Spotify Web API		

Feature Name: User Profile Page	Priority: Normal
Description: a dedicated profile page which will display the user's information such as username, profile picture, and posts	

Inputs: clicking on someone's username	Outputs: user profile page	Processes: when users click on another user's username, they will be taken to another page which will display that user's information and posts
Dependencies: database, backend APIs		

Feature Name: Create Playlists	Priority: Normal	
Description: Give users the ability to create and add playlists by implementing Spotify's API to secure song titles and song snippets to add to a playlist. The information taken from Spotify's API which would include song titles and functions from the API will then be stored in a database connected to a user's account to allow them to play snippets and display a song's cover from TuneShare..		
Inputs: Song titles	Outputs: display song cover, title. And have the option to play a snippet from the song.	Processes:A user would first create an empty playlist and use the search bar dedicated to searching for music to search for songs they would like to add.Once a user has found a song they would like to add to a playlist, a user would select the option to add a song to a playlist. Once a song has been added to a playlist the image of the songs cover would be displayed on the created playlist's page and the option to play a snippet would appear alongside the cover image.
Dependencies: song title, Spotify API, database.		

Feature Name: Admin	Priority: normal

Description: special account that will be able to delete posts, accounts, etc from other users		
Inputs: Account made	Outputs: Being able to delete posts, accounts, and monitor the site.	Processes: Backend
Dependencies: Database, Backend		

Feature Name: Artist accounts	Priority: Low	
Description: these will be special accounts that can only be created by artists, they will be able to upload songs directly		
Inputs: user credentials(email, username, password).	Outputs: An artist account	Processes: Artists will be able to create their accounts, and upload their songs so that other users can listen to them
Dependencies: database, backend APIs		

Feature Name: Spotify Stats	Priority: Low	
Description: User profiles can display the user's spotify stats such as the latest played songs, most played songs/artists, etc. This should be hidden by default since it can be considered private information.		
Inputs:	Outputs	Processes:
Dependencies: Spotify Web API		

Non- Functional Requirements

Features	**Non-Functional Req.**
Create Accounts	● An email cannot have more than one account linked to it ● Usernames should be unique ● Usernames should be at most 20 characters long ● User IDs (automatically generated on the backend) should be unique ● Passwords minimum requirement have 8 characters, need to have 1 number and 1 special symbol
Sharing Songs	● User posts will only be viewable to the user and the user's friends, via the feed ● The user must be logged in to make a post ○ Otherwise, they will be directed to register page
Linking Spotify	● A user can only link one Spotify account to their TuneShare account ● This will pull data from the user's Spotify account
Searching for Songs:	● Display at most 8 results to a dynamic page ○ Load 8 more results with "load more" option
Spotify stats:	● Data from the user's linked Spotify account ○ Most listened to songs, recent listens…etc ● Hidden by default, with the option to display them
Comments/likes	● A user can only like a post once ● The total number of likes on a user's post will be viewable on the feed to

	the user, and the user's friends ● Comments should be under 1000 characters ● Comments will be viewable under each post ○ Initially one comment, with the option to load/view all comments
Main Page	● Display the 5 most recent posts ● Load next 5 after scrolling down ● Load old posts once all new content has been viewed
User Profile Page	● Page load time ● Profile picture loading time ● Data retrieval time
User Search & add	● 100 Character limit ● Username search load time
Artist accounts	● Verified Artist accounts ● Can upload songs no more than 50MB in size
Create Playlists	● 1000 songs max per playlist

Quality Attributes:

- **Reliability**
  - Availability - The system should be available 24/7 with minimal downtime
  - Data Consistency- Data such as posts and comments, should be consistent even after modifications.
- **Usability**
  - User Experience (UX)- The interface should be as user friendly as possible with clear navigation.
- **Maintainability**
  - Code Maintenance- Code should be consistent and have the option for easy feature additions and bug fixing.
  - Documentation- Detailed documentation for code, APIs, and database.
- **Security**
  - Authentication and Authorization- Secure sign up, login, profile management processes.
  - Password hashing following SHA256 standards + salting

System Performance:

- **Response Time**
    - Pages and posts should load in under 3 seconds on average
- **Throughput**
    - The system should be able to handle up to 100 requests per second
- **Resource Utilization**
    - CPU Utilization
        - Servers need to handle many user requests at the same time.
            - Posts,likes,comments,feed generation
    - Memory (RAM)
        - Memory usage for caching accessed data
            - Catching profiles,main feed,trending
        - Could impact database load and response time

Constraints:

- **Technical Constraints**
    - Response time due to network latency could affect the loading time requirements.
    - Data Storage capacity might be a limiting factor
    - Retrieval speed limitations when handling data sets
    - Third Party Integrations impose rate limits and downtime risks
- **Operational Constraints**
    - Restrictions when updates or deployments made
    - Maintenance windows and system down times could affect user experience
- **Platform Constraint**
    - Browser compatibility could be an issue (Chrome, Safari, Firefox)