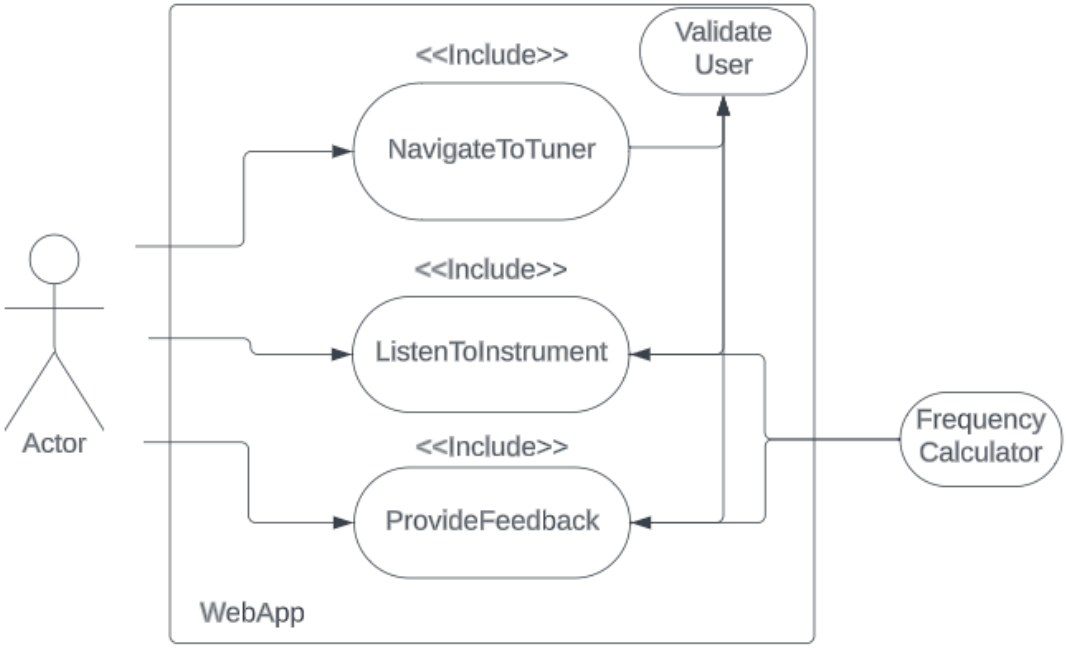
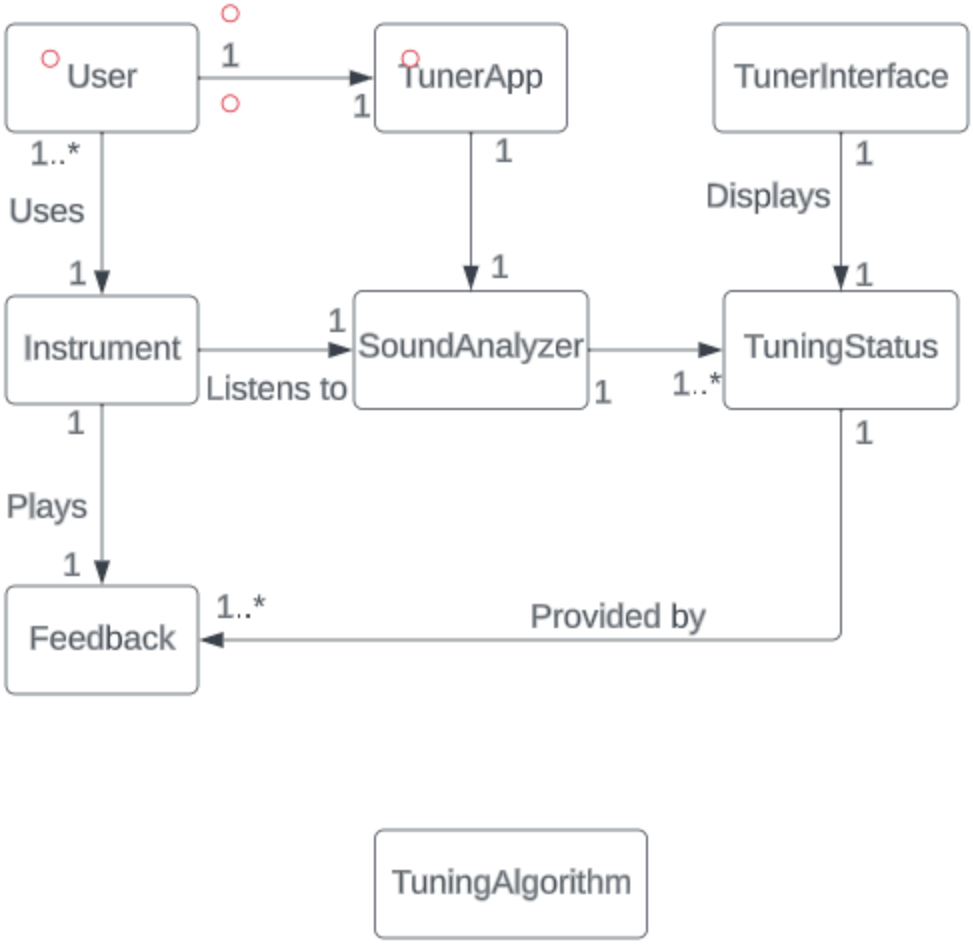
Use Case 1: In-App Tuner

Success Scenario:

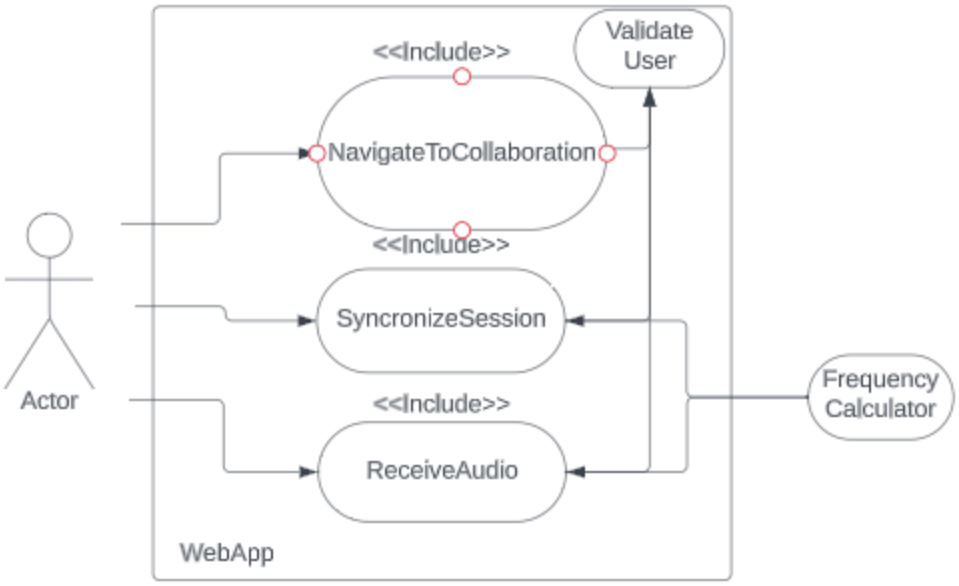
1. User opens the app.
2. User navigates to the "Tuner" section.
3. User activates the tuner feature.
4. The app's tuner interface is displayed.
5. User plays a musical instrument.
6. The app listens to the instrument's sound and displays the current tuning status.
7. The app provides visual and auditory feedback to help the user tune the instrument.
8. User successfully tunes their instrument.

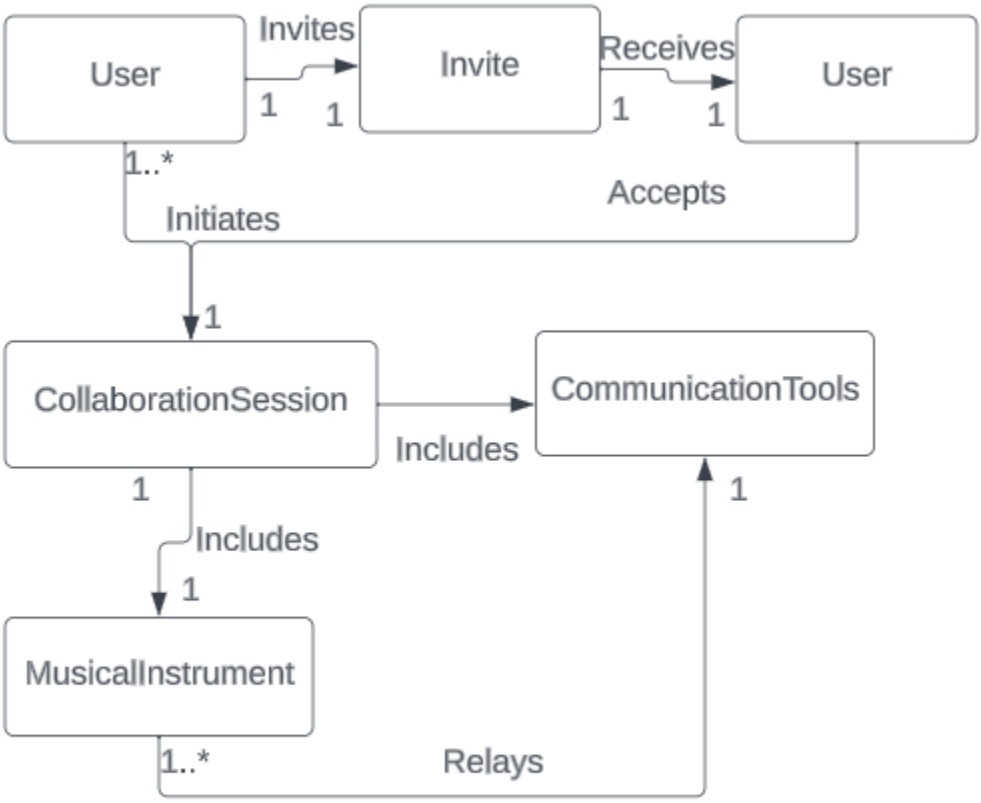




Use Case 2: Collaboration Feature

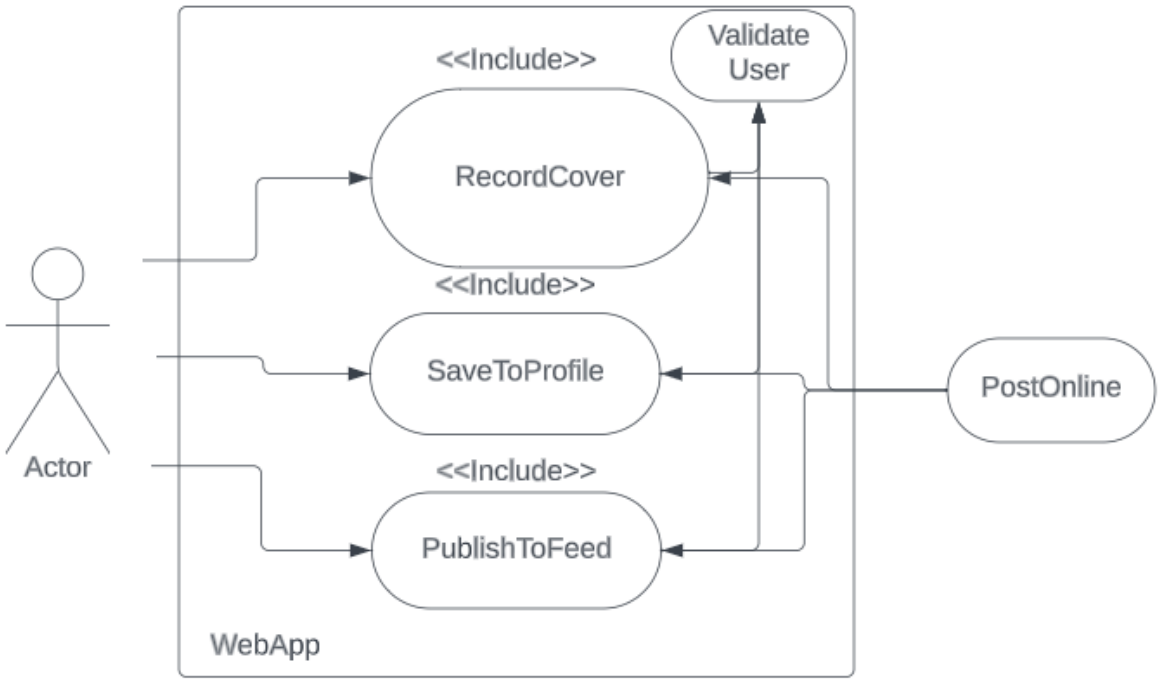
Success Scenario:
1. User logs into the app.
2. User navigates to the "Collaboration" section.
3. User initiates a collaboration session and invites friends.
4. Friends accept the invitation and join the session.
5. The app synchronizes the musical session, allowing users to play together in real-time.
6. Users can see and hear each other's performances.
7. Users successfully collaborate on a song requiring multiple instruments.
8. The app provides tools for communication during the collaboration session.

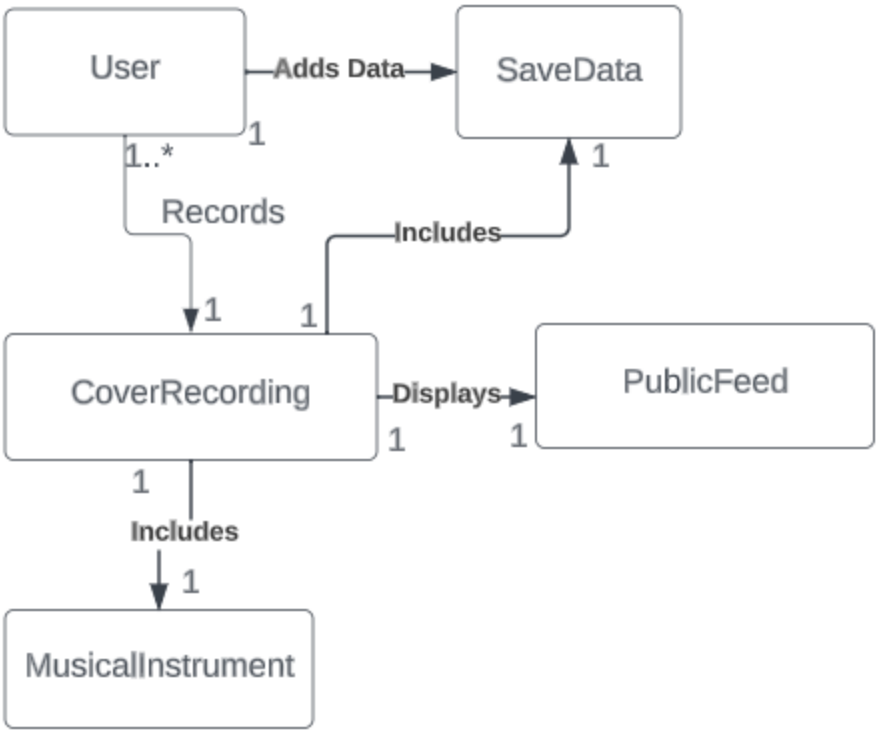

Validate User
<<Include>>
NavigateToCollaboration
<<Include>>
SyncronizeSession
<<Include>>
ReceiveAudio
Frequency Calculator
Actor
WebApp


User
Invites
1
1
Invite
Receives
1
1
User
1..*
Initiates
Accepts
1
CollaborationSession
Includes
CommunicationTools
1
1
Includes
1
MusicalInstrument
1..*
Relays

Use Case 3: Save and Post Covers

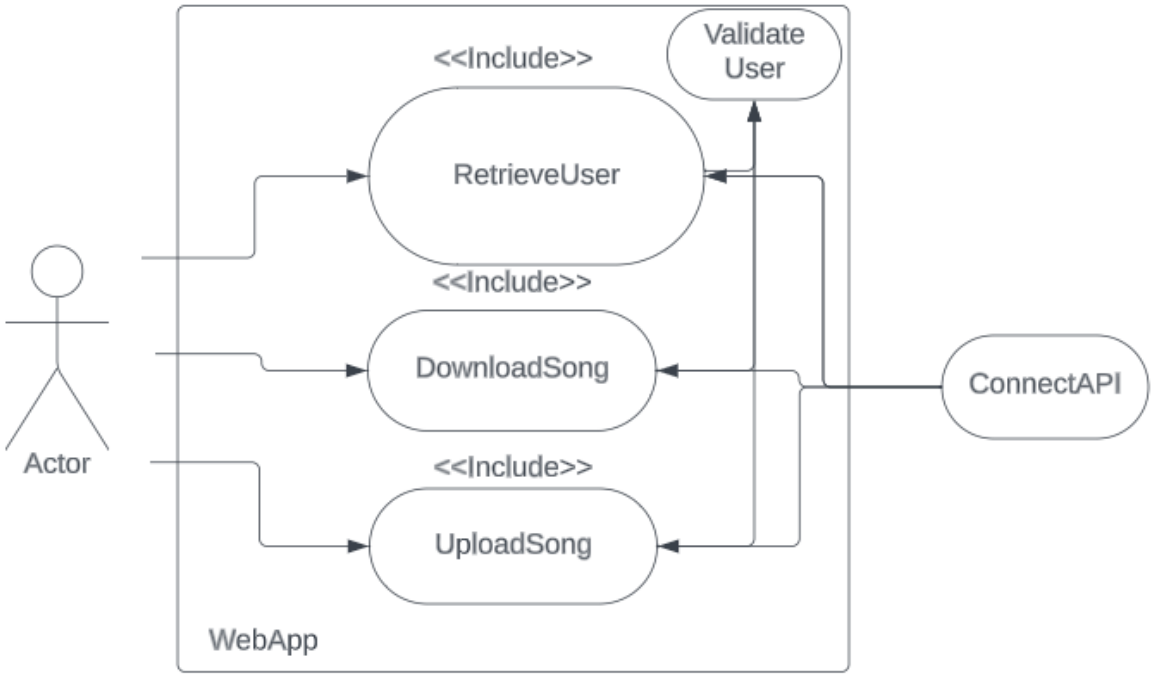
Success Scenario:
1. User logs into the app.
2. User records a cover of a song within the app.
3. User adds metadata such as song title, artist, and description.
4. The app saves the recorded cover to the user's profile.
5. User selects the option to post the cover.
6. The app publishes the cover to a public feed for discovery by other users and creators.
7. Other users can listen to and engage with the posted covers.
8. Users can like, comment on, and share covers.

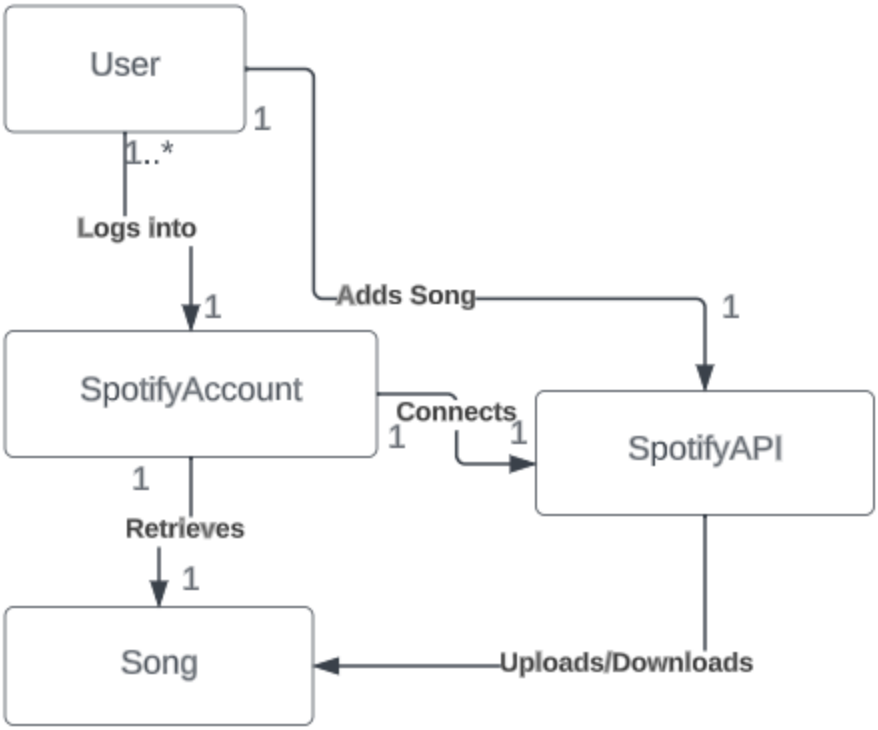




Use Case 4: Integration with Spotify

Success Scenario:
1. User logs into the app.
2. User connects their Spotify account to the app.
3. The app securely establishes a connection with Spotify's API.
4. The app retrieves the user's Spotify playlists and song library.
5. User selects a song from Spotify to play or cover within the app.
6. The app automatically uploads the chosen song for the user to work on.
7. Users can seamlessly switch between their app-created content and Spotify tracks.
8. User selects a song from their library to upload to Spotify.
9. The app uploads the chosen song to their Spotify profile.


Validate User
<<Include>>
RetrieveUser
<<Include>>
DownloadSong
ConnectAPI
Actor
<<Include>>
UploadSong
WebApp



Non-functional requirements:

- **Security**: User passwords are salted to ensure website security. Unique salts are used for each password.
- **Integration**: The app should be able to securely connect to external services like Spotify.
- **Performance**: The app should provide real-time synchronization and low-latency audio feedback during collaboration sessions.
- **Scalability**: The app should handle a growing user base and an increasing number of posted covers.

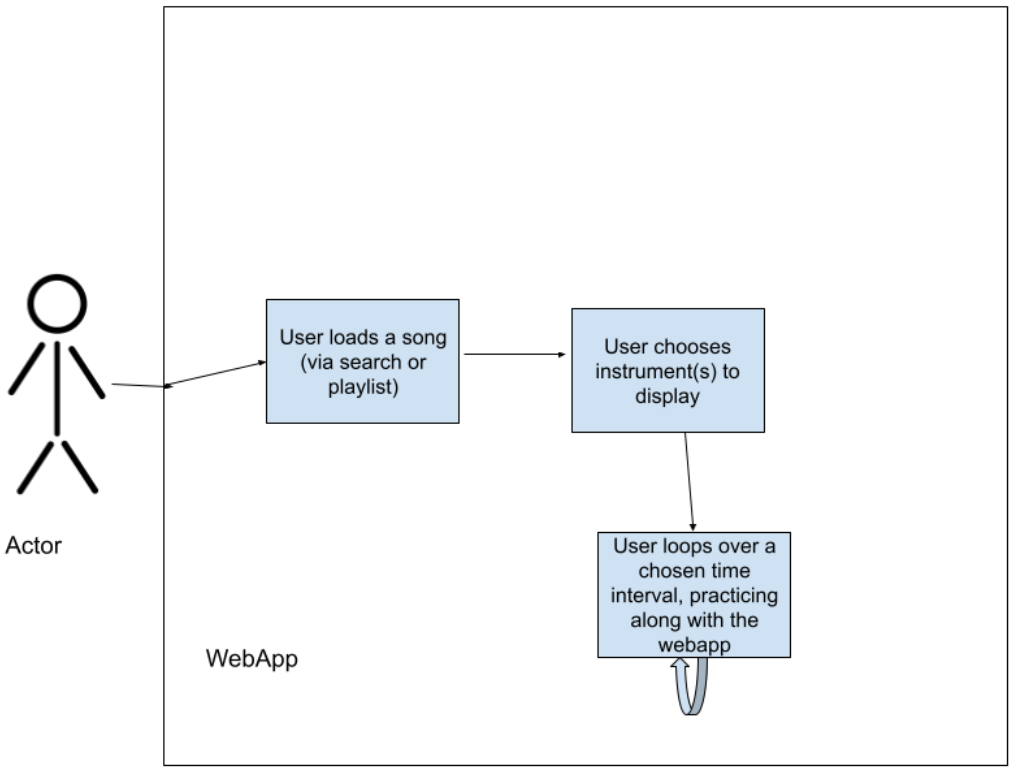
This requirements description outlines four key use cases and includes non-functional requirements related to security, integration, performance, and scalability to ensure a robust and user-friendly music app.

Use Case 5: Practicing a song
Success Scenario:

1. User opens the app.
2. User navigates to the "Practice" section.
3. User searches and loads a song OR loads a song from their playlist.
5. User sets tempo on metronome
6. User clicks play on timeline
7. User can set time intervals to practice and repeat
8. User can click exit to quit practicing the song.

Nonfunctional:

- Minimalist design
- Reliable connection
- Efficient instrument rendering
- Secure login



Use Case 5: Upload Song

1. User logs into the app
2. User opens up songs menu
3. User clicks on edit playlist
4. User clicks add song
5. A menu pops up to either upload an audio file
6. User uploads audio file
7. Audio file is now in the server available for other users to see

Use Case 6: user reports a bug

1. User logs into the app
2. User notices a bug in the web app
3. User goes to the main menu/ home screen and opens report a bug menu
4. User puts in required information and the bug report is sent

Use Case 7: User checks compatible instruments

1. User logs into the app
2. User goes to opens up the instrument menu
3. User opens up compatible instruments menu
4. User realizes the instrument they are planning to buy is not compatible

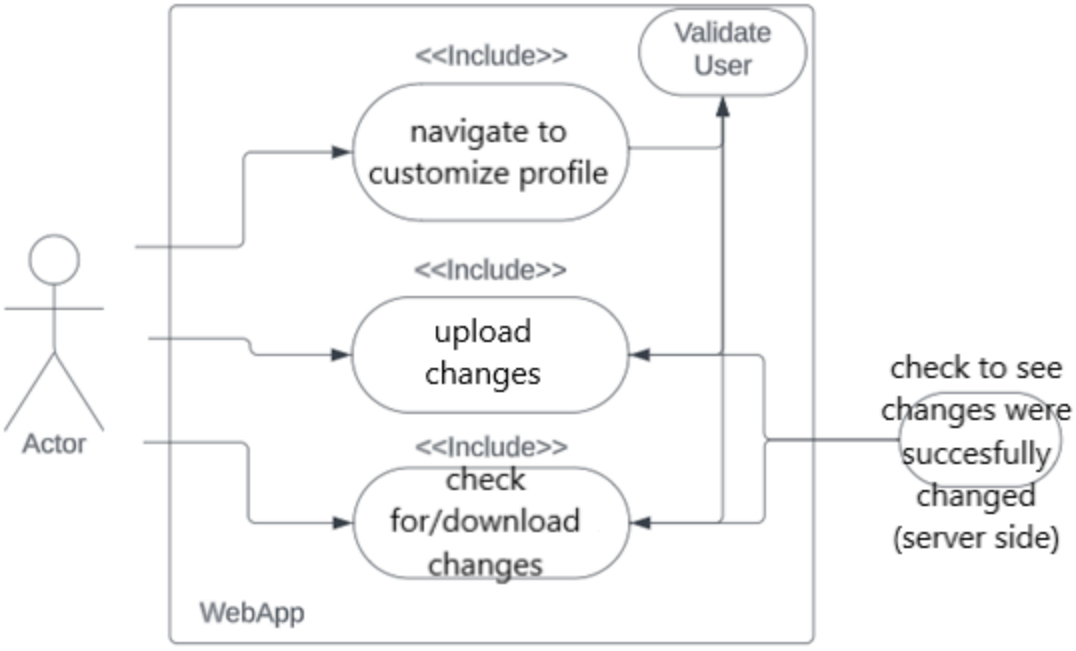
Use Case 8: Customize Profile

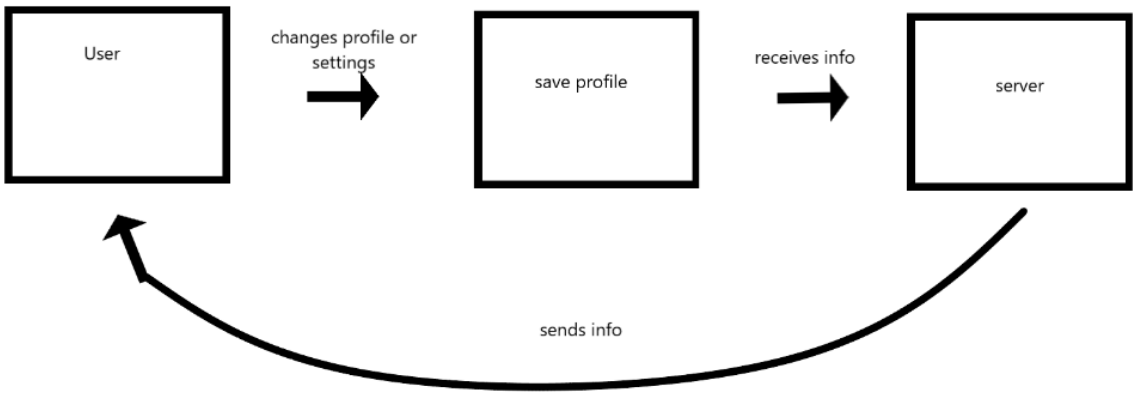
Success Scenario:

1. User logs in the app.
2. User opens up the customize profile
3. User changes desired settings and clicks
4. User modifies the profile, and clicks save
4. User exits the app
5. User logs back into the app and the changes were saved

Nonfunctional:

- Profile changes take less than 2 seconds to save





Use Case 9: Logging played songs
Success scenario:

1. User logs into the web app
2. User selects a song they would like to play
3. User plays and completes the song successfully
4. System records they have completed the song

5. System remembers they completed the song playing the bass
6. This data is saved to their profile

Use Case 10: Leaderboard
Success scenario:

1. User logs into web app
2. User plays a song and completes the song
3. User is given a score based on their playing
4. After finishing the song a leaderboard is displayed showing the scores of their friends as well as the top scores
5. User is also able to navigate to a separate section for leaderboards
6. There is a leaderboard for friends and all people that displays their lifetime points earned

Use Case 11: Feedback

1. User logs into web app
2. User selects a song
3. User selects their instrument and connects it to the web app
4. User starts playing song
5. Display shows visual feedback as the player correctly hits/misses the correct notes and timing
6. This data is collected and goes into a score compiled at the end of the song to grade your playing of the song

Use Case 12: Song Suggestions

1. User is logged into their account for the first time
2. User is prompted to select their instruments of choice
3. User is prompted to select their favorite artists, songs, and genres of music
4. This data is recorded
5. Over time, the songs the user has played, completed, or abandoned are recorded
6. The data is compiled together using a learning algorithm to suggest songs that are similar to songs they have played, suggests new music that is popular, or trending music that fits their selected favorite genres, artists, instruments
7. These suggestions are displayed when the artist finishes a song, as a suggestion for the next song to play.

Non-functional Requirement

- Web app is intuitive to use without any previous music experience

Use Case 13: Metronome
Success Scenario:

1. The user logs into the web app and selects the instrument they want to practice, in this case, the guitar.
2. The user navigates to the song library, where they choose a song they wish to practice.
3. The user selects the metronome feature for the chosen song, which is clearly labeled and accessible from the song details page.
4. The app presents the user with options to customize the metronome settings, allowing them to adjust the beats per minute (BPM) to match their skill level and comfort.
5. The user sets the BPM to their desired tempo using a user-friendly slider.
6. As the selected song begins, the metronome starts ticking according to the user's chosen BPM, providing a clear and audible beat.
7. The user also sees a visual indicator on the screen, ensuring they maintain perfect timing throughout the practice.
8. The app simultaneously displays chord progressions, sheet music, or finger placements relevant to the song, aiding the user in playing along accurately.
9. If the user encounters difficulty with a particular part of the song, they have the option to pause the metronome and practice that section repeatedly until they feel confident.
10. The app provides instant feedback and helpful tips, showing the user where they can improve
11. The user practices the selected song, guided by the metronome, until they feel confident and satisfied with their performance.
12. The app records the user's practice session, including the BPM settings used and the time spent on the song.

Use Case 14: Live Interactive Tutorials

1.The user signs up for the "web app" with the goal of learning to play the guitar.
2.Upon creating an account and selecting the guitar as their instrument of choice, the web app guides the user through a setup process to determine their skill level, starting from complete beginner.
3.The web app displays a personalized dashboard for the user, featuring recommended tutorials based on their skill level and musical interests.
4.The user chooses a beginner's tutorial on basic guitar chords, marked with clear instructions for new learners.
5.The tutorial starts with a friendly and encouraging introduction video from a skilled guitar instructor, explaining the importance of the topic.
6. As the user progresses through the tutorial, the web app provides step-by-step instructions, both in video and text format, to ensure they understand the techniques involved.

Use Case 15: Customizing User Settings

1.The user logs into the "web app" to access their personalized account.
2. The user navigates to the settings menu, where they can tailor their experience by adjusting various preferences.

3. Within the settings, the user finds options for customizing notifications, allowing them to choose which types of notifications they wish to receive, such as updates, new content, or community interactions.
4. The user can also specify the frequency and delivery method for these notifications, ensuring they receive updates in a way that suits their needs.
5. Preference settings are easily accessible, allowing the user to fine-tune their user experience, such as adjusting the app's theme, font size, or display preferences, enhancing their comfort while using the app.
6. Language settings provide the user with the flexibility to select their preferred language for the app's interface and content, promoting inclusivity and accessibility for a global user base.
7. After customizing their settings, the user saves their preferences, and the web app promptly applies the selected options.
8. The user's settings are now tailored to their preferences, ensuring a personalized and user-friendly experience.
9. Any changes in settings, such as language preferences or social media integration, are immediately reflected in the user's interface.

Use Case16 : Genre Sorting and Song Saving

Success Scenario:

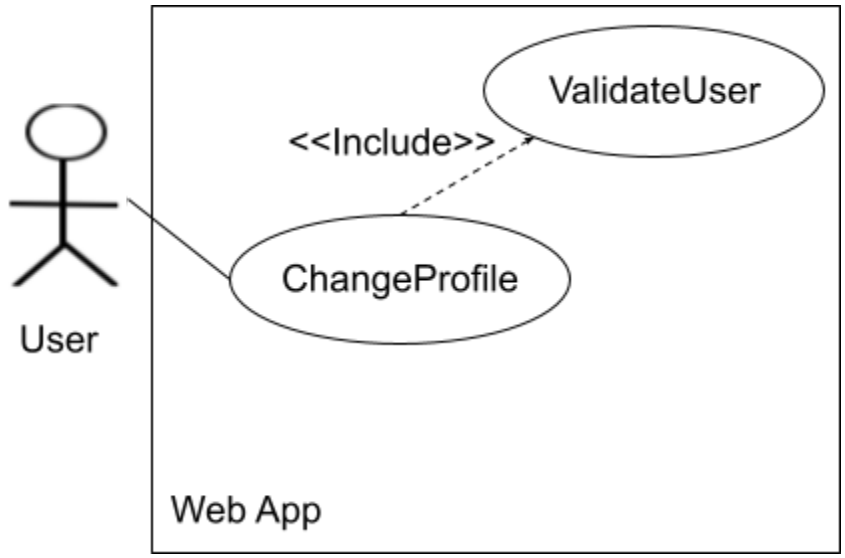
1. The user opens the app to explore and organize their music collection.
2. In the app's search function, the user selects the "Genre" filter to narrow down their search for a specific type of music.
3. The user chooses the "Rock" genre, and the app instantly populates a list of rock songs from their music library.
4. Scrolling through the results, the user finds a song that they want to add to their playlist.
5. With a simple tap on the "Save" button next to the song, the app stores the song in their saved songs collection.
6. The user repeats this process to save several more rock songs that they plan to include in a playlist.
7. The user then navigates to their "Playlists" section, where they can create a new playlist or select an existing one.
8. The user selects the "Create New Playlist" option and gives it a name, such as "Rock Classics."
9. They go back to their saved songs collection and start adding their favorite rock songs to the newly created "Rock Classics" playlist.
10. After customizing their playlist, the user enjoys listening to their curated collection of rock music, all neatly organized in one place

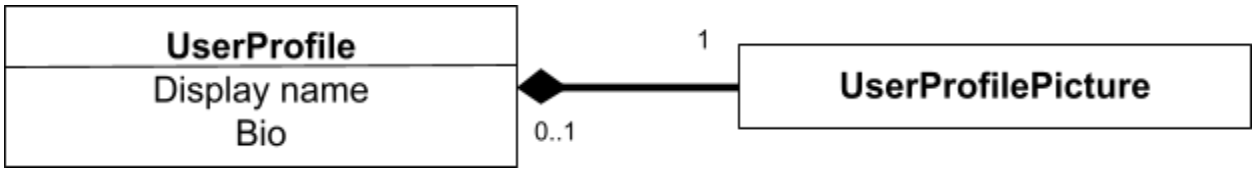
Non Functional Requirement:
Responsiveness: The app must provide a smooth and seamless user experience, ensuring quick loading times and smooth transitions between screens, with a focus on minimal lag and user interface responsiveness.

Bandwidth Efficiency: The app should be designed to optimize data usage and be considerate of users with limited bandwidth. It should prioritize efficient data transmission, minimizing unnecessary data transfer and reducing the strain on users' internet connections.

Use Case 21: Change Profile
Success Scenario

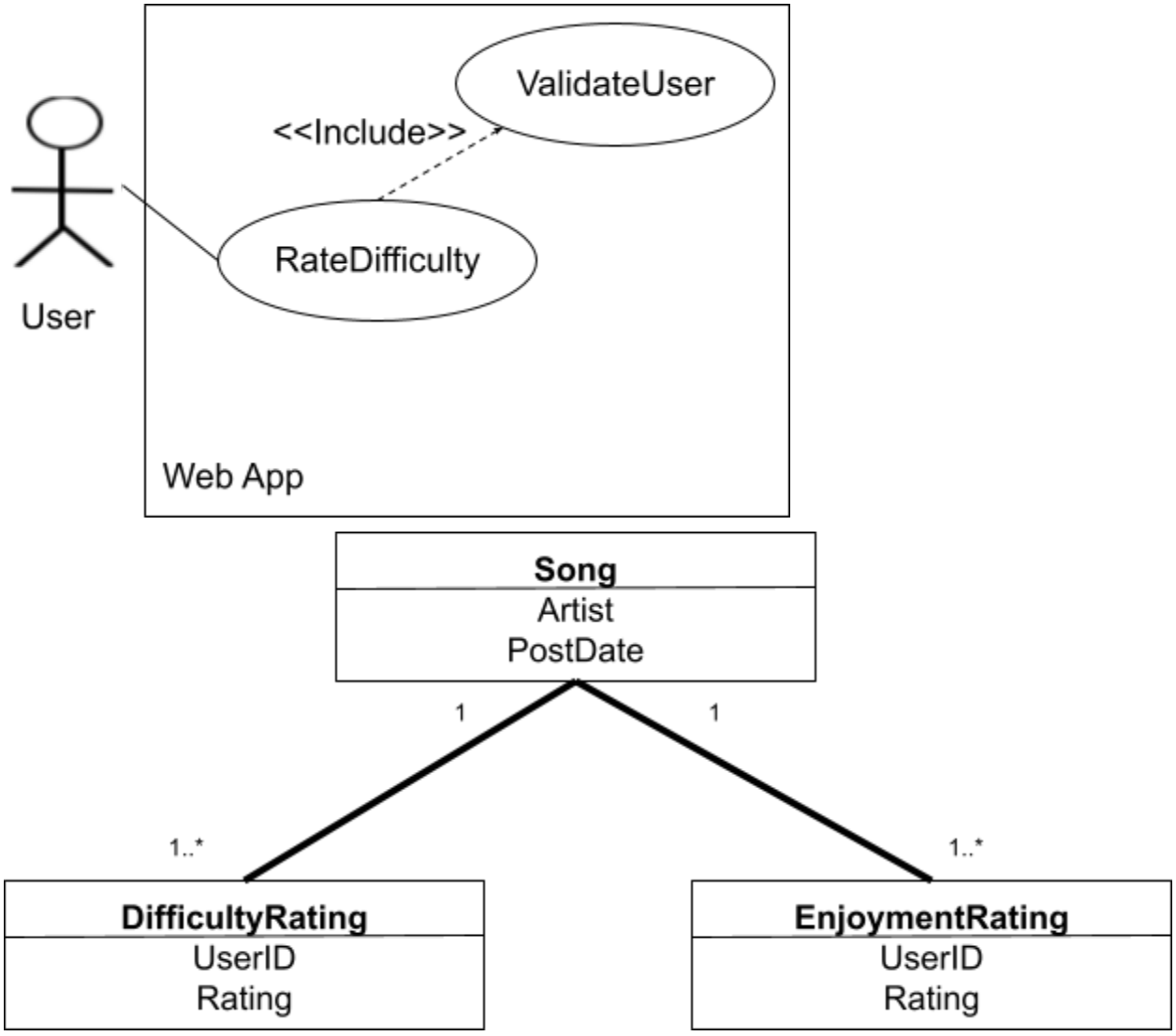
1. User logs into account
2. User selects the profile button
3. User selects the edit profile button
4. User changes the information they would like (profile picture, display name, bio)
5. User clicks save changes





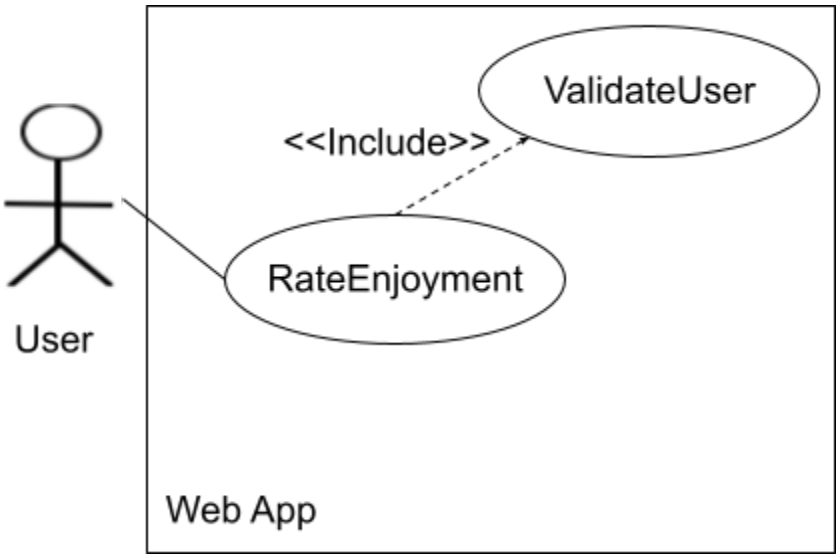
Use Case 22: Rate Song Difficulty
Success Scenario

1. User logs into account
2. User locates a song
3. If the user has played it before, prompt user to rate the difficulty
4. User selects 1 to 5 stars in 0.5 start increments
5. User selects to post their rating



Use Case 23: Rate Song Enjoyment
Success Scenario

1. User logs into account
2. User locates a song
3. If the user has played it before, prompt user to rate the enjoyment
4. User selects 1 to 5 stars in 0.5 start increments
5. User selects to post their rating



Use Case 24: Generate Shareable Link
Success Scenario

1. User logs into account
2. User navigates to a previous recording
3. User selects the generate shareable link button
4. User copies the link