Use Cases

1. creating a room - done

Alisher Pazylbekov

2. joining a room
3. sharing room
4. kicking someone
5. viewing user account
6. changing hosts

Ian Crutcher

7. add song
8. removing a song
9. checking history of room
10. next song (host can skip song)
11. moving songs, getting token back

Omid Anvar

12. searching for rooms
13. adding a friend
14. previewing a song
15. closing the room
16. Logging into website

**Use Case UC1: Creating a Room**

Primary Actor:

- User of website

Stakeholders and Interests:

- User wanting to create a room

Preconditions:

- User is logged in

Success Guarantee:

- Room is created with configuration specified by the user
- User has host privileges in the room

Main Success Scenario:

1. User clicks create room button
2. User prompted with configuration details
3. User selects desired configuration settings and accepts
4. System creates new room with user selected configurations
5. User is greeted with the room screen

Extensions:

3. User fails to give proper specifications
    1. System notifies user about error
    2. User fixes error

Special Requirements:

1. Room should be created within 2,000 ms

Technology and Data Variation: None

List Frequency of Occurrence: Whenever a user wants to create a room

**Use Case UC2: Joining a Room**

Primary Actor:

- User

Stakeholders and Interests:

- User wanting to join a room

Preconditions:

- User is logged in
- At least one room exists

Success Guarantee:

- User joins the chosen room successfully
- User is granted user permissions only

Main Success Scenario:

1. User searches for intended room
2. User clicks intended room
3. User clicks on “Join Room”
4. System adds new user to room
5. User is greeted with the room screen

Extensions:

2. Password is required
    1. User enters password
    2. System validates password
        1. User is notified if password incorrect
1. User is provided with link
    1. User clicks link
    2. User clicks “Join Room”
        1. User prompted with password input
        2. User enters password
        3. System validates password
    3. User successfully joins room
4. User is on room’s blacklist
        1. User is notified that they can not join the room

Special Requirements:

2. User should be added within 2,000 ms

Technology and Data Variation: None

List Frequency of Occurrence: Whenever a user wants to join a room

**Use Case UC3: Sharing a Room**

Primary Actor:

- User

Stakeholders and Interests:

- User wanting to share a room
- Other user that is receiving the shared room link

Preconditions:

- User is logged in
- User is part of a room

Success Guarantee:

- Link is generated
- Link is sent out by user through outside messenger

Main Success Scenario:

1. User clicks on their room
2. User clicks “Share Room” button
3. System generates a link to the room
4. User is prompted with the link
5. User sends out link using an outside messenger

Extensions: Only user errors

Special Requirements:

3. Link should be generated within 2,000 ms

Technology and Data Variation: link is able to be copied and pasted

List Frequency of Occurrence: Whenever a user wants to share a room

**Use Case UC4: Kicking someone from a room**

Primary Actor:

- Host

Stakeholders and Interests:

- Host wanting a member to be kicked

Preconditions:

- User is a host
- User is part of a room

Success Guarantee:

- User is successfully kicked by host

Main Success Scenario:

1. Host clicks on desired user
2. Prompted with user account
3. Clicks “Kick User”
4. Prompted with a “Ban” option
5. Host clicks the button
6. System removes user from room
7. System registers user as a ban user
8. User is successfully kicked from the room

Extensions:

6. Said user has already left the room
    a. Host prompted with error message, user not in room

Special Requirements:

1. Request should be processed within 2,000 ms

Technology and Data Variation: None

List Frequency of Occurrence: Whenever a host wants to kick a user from a room

**Use Case UC5: Viewing a user’s account**

Primary Actor:

- User

Stakeholders and Interests:

- User wanting to see other users’ accounts

Preconditions:

- User is logged in
- User is part of room

Success Guarantee:

- User is able to see details about chosen user’s account

Main Success Scenario:

1. User clicks “Users” button to see list
2. User clicks on desired other user
3. Other user’s account pops up

Extensions:

2. Other user leaves room right when user clicks on their name
    a. User prompted with error message saying other user has left the room

Special Requirements:

4. Other user's account should pop up in under 2,000 ms

Technology and Data Variation: None

List Frequency of Occurrence: Whenever a user wants to see another user's account

**Use Case UC6: Changing hosts**

Primary Actor:

- User

Stakeholders and Interests:

- Host wanting to pass room onto someone else
- Users who want to become hosts

Preconditions:

- User is logged in
- User is a Host of a room

Success Guarantee:

- User passes on Host permissions to another user
- User reverts to user permissions

Main Success Scenario:

1. Host clicks on "Users" button to see list
2. Host clicks "Make Host" button associated with new host
3. System takes away host permissions from original host
4. System gives host permissions to new host
5. Desired user acquires host permissions

Extensions:

4. Desired user leaves/disconnects
    a. System processes fail
    b. Host prompted with error message saying user disconnected

Special Requirements:

5. Permissions should be taken away in under 2,000 ms

Technology and Data Variation: None

List Frequency of Occurrence: Whenever a host wants to make another user a host

**Adding a song to the playlist**

Primary Actor:

- User

Stakeholders and Interests:

- User wanting to add a song to be played in a room

Preconditions:

- User is logged in

- User is currently in a room

Success Guarantee:

- User selected song is added to the playlist

Main Success Scenario

1. User clicks add song button
2. User is prompted to submit a link to a youtube video
3. User enter link to video
4. System checks that link points to a valid video
5. System checks that song is not already in the playlist
5. System adds song to the queue
6. Playlist updates on the website

Extensions:

4. Link does not point to a valid video
    1. User is notified that link did not work
    2. User resubmits video
5. Song is already in the current playlist
    1. System notifies the user that submission of this song is not possible at this time
    2. User can cancel or try another link

Special Requirements: None

Technology and Data Variation: None

Frequency of Occurrence: Whenever a user wants to add a new song into the playlist

**Removing a song**

Primary Actor:

- Host

Stakeholders and Interests:

- Host wanting to delete a song from the playlist

Preconditions:

- User is logged in
- User is currently in a room
- User has Host privileges in a room
- There is a song in the playlist that is not currently being played

Success Guarantee:

- Song is removed from the current playlist

Main Success Scenario:

1. Host clicks on the delete button on a song in the playlist
2. System removes song from playlist
3. Playlist updates on the website

Extensions: None

Special Requirements: None

Technology and Data Variation: None

Frequency of Occurrence: Whenever the host wants to remove a song

**Checking history of a room**

Primary Actor:

- User

Stakeholders and Interests:

- User wanting to check the playlist history of the room he/she is in

Preconditions:

- User is logged in
- User is currently in a room

Success Guarantee:

- History of the current room is displayed

Main Success Scenario:

1. User clicks on history button in the room
2. System collects the history information
3. User is navigated to the history page for the current room

Extensions: None

Special Requirements: None

Technology and Data Variation: None

Frequency of Occurrence: Whenever a user wants to check the room history

**Skip song**

Primary Actor:

- Host

Stakeholders and Interests:

- Host wanting to advance to the next song in the playlist

Preconditions:

- User is logged in
- User is currently in a room
- User has host privileges in the room he/she is in
- There is a song in the queue to be advanced to

Success Guarantee:

- Current playing song is stopped
- Next song is now being played

Main Success Scenario:

1. Host clicks the next button
2. System ends the current song
3. System starts the next song
4. Next song is now being played

Extensions: None

Special Requirements: None

Technology and Data Variation: None

Frequency of Occurrence: Whenever the host wants to skip songs

**Moving songs with tokens**

Primary Actor:

- User

Stakeholders and Interests:

- User wanting to move a song up in the current playlist

Preconditions:

- User is logged in
- User is in a room
- Room is configured with movement ability
- There are songs in the playlist that can be moved up

Success Guarantee:

Song is moved up X positions in the playlist

User receives spent tokens back when song is played

Main Success Scenario:

1. User clicks move up button
2. User is prompted to spend tokens
3. System checks user has enough tokens to spend
4. System accepts and moves song up the same number of positions as tokens spent
5. X tokens are deducted from users total
6. Playlist is updated
7. After song is played, users receives back X tokens

Extensions:

3. User does not have enough tokens
    1. User is notified they do not have enough tokens
    2. User can either re-submit or cancel

Special Requirements: None

Technology and Data Variation: None

Frequency of Occurrence: Whenever a user wants to move a song

**Use Case: Searching for Rooms**

Primary Actor:

- User

Stakeholders and Interests:

- User wants to search for a specific room

Preconditions:

- User must be logged into web application

Success Guarantee:

- User is able to search through a list of available rooms

Main Success Scenario:

1. User uses the search functionality of application to search for a specific room
2. User is able to see a list of rooms that user can potentially join

Extensions: None

Special Requirements: Room search should display list of rooms within 1,500 ms
Technology and Data Variation: None
List Frequency of Occurrence: Whenever a user wants to search for a room

**Use Case: Logging into Website**
Primary Actor:
- User

Stakeholders and Interests:
- User wants to log into web application

Preconditions:
- User must have internet access

Success Guarantee:
- User is able to successfully log into website

Main Success Scenario:
1. User visits music player website
2. User types in username and password
3. System validates User credentials
4. User is greeted by website home screen

Extensions:
3. User does not type in credentials properly
    a. System notifies user that login was unsuccessful due to credentials
    b. User reenters credentials

Special Requirements: Successful logins should take less than 2500 ms
Technology and Data Variation: None
List Frequency of Occurrence: Whenever a User wants to use the website for any reason

**Use Case: Closing the Room**
Primary Actor: Host
Stakeholders and Interests: Host wants to close the room
Preconditions:
1. User must be a Host to close a room
2. Host must be a part of the room host is attempting to close

Success Guarantee: Host is able to successfully close a room
Main Success Scenario:
1. Host closes the room host is a part of
2. Users and Host are no longer part of the room
3. Room no longer exists

Extensions: None
Special Requirements: Closing a room should take place within 3,000 ms
Technology and Data Variation: None
List Frequency of Occurrence: Whenever Host wants to close a specific room

**Use Case: Adding a Friend to Room**

Primary Actor: User, User's friend (another User)
Stakeholders and Interests: User wants to add a friend to User's current group
Preconditions:
1. Both User and User's friend must be logged into application
2. User must already be part of a Room
3. Room must not be at capacity

Success Guarantee: User's friend is able to join room User is in
Main Success Scenario:
1. User invites friend into current Room
2. User's friend types password to enter room (if applicable)
3. System checks if room is at capacity
4. If room is not at capacity User's friend is now part of the room

Extensions:

3. Room is already at capacity

1. System displays message stating room is at capacity and cannot add User's friend

3. User's friend enters in room password incorrectly

1. System notifies User's friend that login was unsuccessful due to credentials

2. User's friend reenters credentials

Special Requirements: User's friend should be added to room within 2,000 ms
Technology and Data Variation: None
List Frequency of Occurrence: Whenever User wants to add another User to room

**Use Case: Previewing a Song**
Primary Actor: User
Stakeholders and Interests: User wants to briefly listen to a song in the queue
Preconditions:
1. User must be part of a room
2. Songs must be added to playlist in room

Success Guarantee: User is able to listen to brief 30 second preview of song
Main Success Scenario:
1. User tries to preview a song in the playlist by clicking preview option
2. Song then plays for 30 seconds

Extensions: None
Special Requirements:
1. Song must be played with 1,500 ms of being clicked
2. Song must play for 30 seconds
3. User must have option to reply song if desired
4. User can preview as many songs as desired

Technology and Data Variation: None
List Frequency of Occurrence: How ever often a user wants to preview a song