SOFTWARE DESIGN & ANALYSIS

(CS-324)

ASSIGNMENT # 3

Section: B

Maha Ahsan (18i-0909)

Nameira Rana (18i-0695)

Ghaliyah Masood (18i-0646)

Hadia Chaudhary (18i-0577)

Submitted To

Ma'am Maheen Arshad

Submission Date

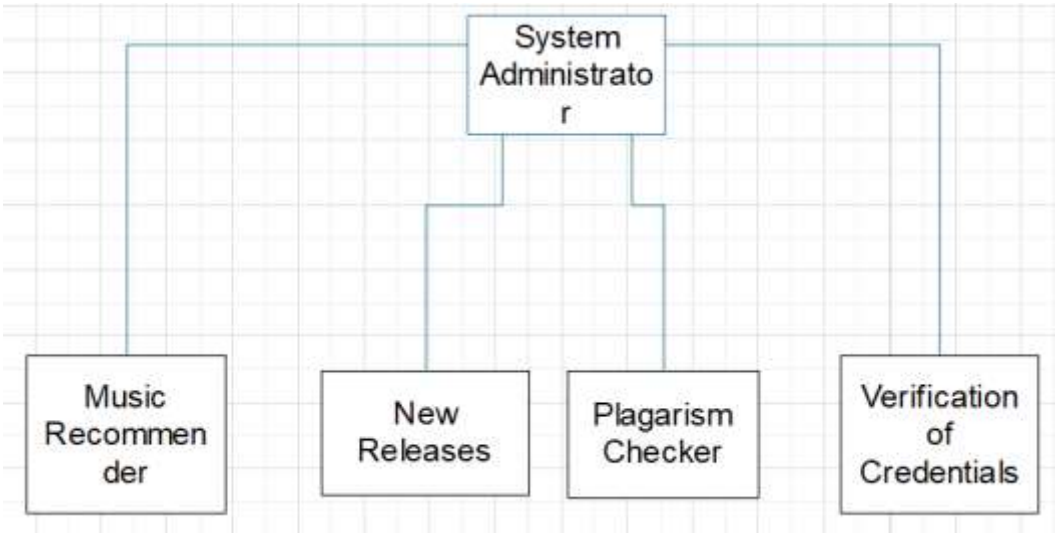
26 October, 2020

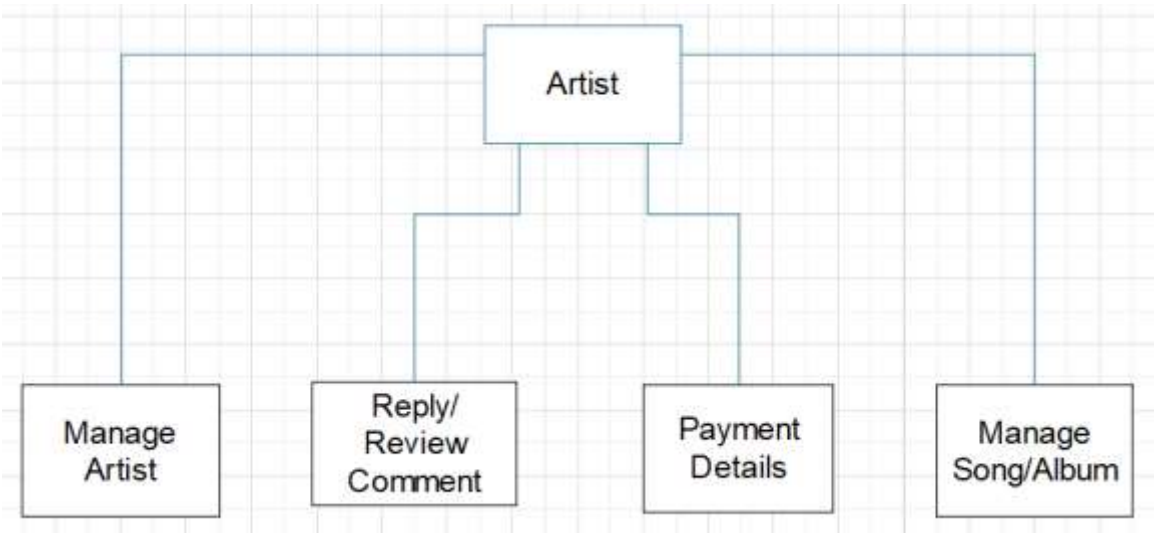
Table of Contents

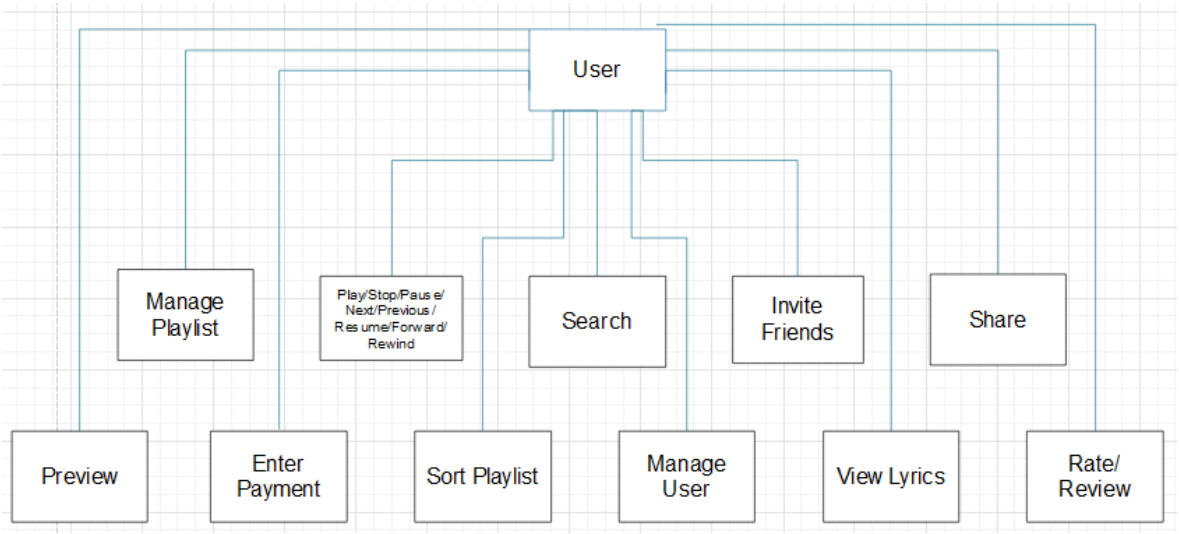
Name: SongZoid

Project Scope/Description:

The goal is to design a Music Management System. The management system will in simple terms allow artists to add their music onto the platform and users to listen to artist's music. The system will basically consist of three components which include database (where all the songs will be kept along with the user's/ artist's data), server (which deals with the calculations/ algorithms of the application) and finally the interface (GUI screen presented to the user/ artist). There will be three handlers of this system which are given below5 with their detailed functionality.







Functional Requirements:

- **Store evaluations:**
  - The server shall receive and store music evaluations

- **Data storing:**
  - The server shall be able to store the newly retrieved data to the database

- **Feedback:**
  - User will give feedback on songs

- **Music buying and sharing:**
  - User can buy and share music with their friends

- **Searching songs/ Albums/ Artist:**
  - User can search for their favorite songs/ albums/ artist from the database

- **Show lyrics:**
  - User can view the lyrics of the song they are listening to

- **Creating playlist:**
  - User can create playlists

- **Sorting playlist/library:**
  - User can sort library/playlist on the basis of name, genre, artist, years (Ascending or descending)

- **User can invite their friends to the app**
  - An invitation link will be the send on the entered email

- **Plagiarism checker**
  - Any song before uploaded will be checked for plagiarism. If the song is plagiarized more than 50%, song will not be uploaded else it will be uploaded

- **Preview songs**
  - User can hear a song snippet before buying it

- **Manage songs/album**
  - Users can handle their songs/albums by retrieving or deleting them
  - Artists can handle their songs/albums by creating/ uploading, retrieving or deleting them

- **Handle a song**
  - User can play/ stop/ pause/ next/ previous/ resume/ forward/ rewind/ stop songs
  - User can put the song(s) on repeat/ shuffle mode

- **Verify credentials**
  - User will verify his payment of a song so he can listen to it for lifetime

- **The software system should be integrated with banking API**

Non-Functional Requirements:

- **Accuracy**
  - Since we will give priority to the accuracy of the software, the performance of the Music Recommender will be based on its accuracy on recommendations

- **Failure handling**
  - System components may fail independently of others. Therefore, system components must be built so they can handle failure of other components they depend on

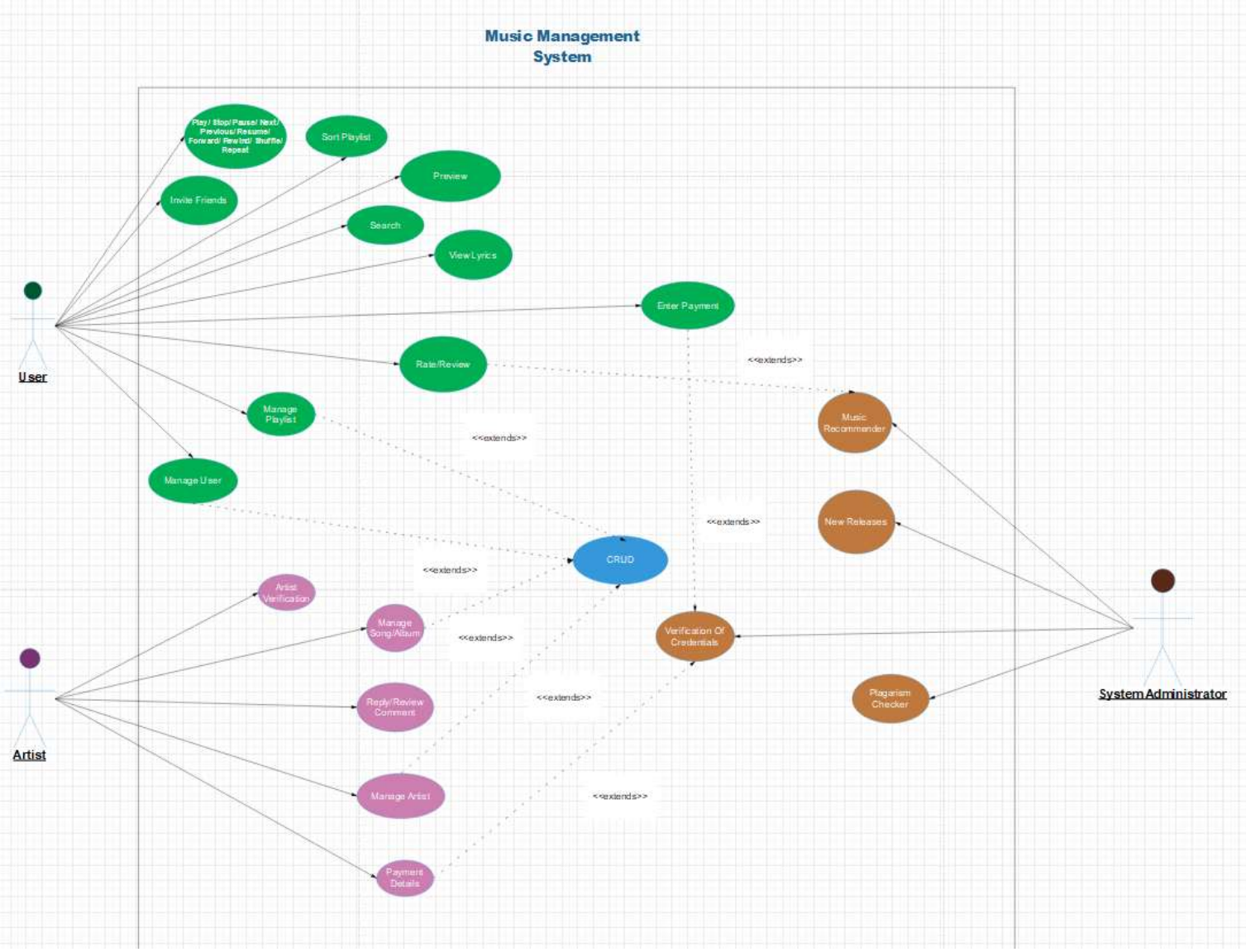
- **Openness**
  - The system should be extensible to guarantee that it is useful for a reasonable period of time

- **Security**
  - Sensitive information (passwords, credit card numbers) should be kept in safe

- **Sound Quality**
  - The sound quality of the songs should be very high. The user should be able to experience high resolution audio.

- **Speed**
  - System must respond to the user's request within 3 seconds.

USE CASE DIAGRAM



ACTOR GOAL LIST:

- **User**
    - Buy music
    - Share Music
    - Make Playlists
    - Preview music
    - Listen to high-quality music
    - Review Music (like or comment)
    - Invite their friends to this application

- **Artist**
    - Release new music
    - Get paid for their content
    - Reply to other user's comments
    - Review other music (like or comment)

- **System**
    - Verify artist
    - Show new releases
    - Music is not plagiarized
    - Secure sensitive information
    - Give Music Recommendations

HIGH-LEVEL USE CASES:

1. Manage Account
2. Preview
3. Manage Playlist
4. Rate/ Review
5. Play/ Stop/ Pause/ Next/ Previous/ Resume/ Forward/ Rewind/ Shuffle/ Repeat song(s)
6. Enter Payment
7. View Lyrics
8. Sort Playlist/ Library
9. Search
10. Share
11. Invite Friends
12. Plagiarism Checker
13. Music Recommender
14. Verify of credentials
15. New Releases
16. Manage Song(s)/ Albums
17. Get Paid
18. Reply/ Review Comments
19. Manage Artist

USE CASE	Manage Account
ACTORS	User
TYPE	Primary
DESCRIPTION	A user can create a new account, update information on their account or delete their account. He can also change the password.

USE CASE	Preview
ACTORS	User
TYPE	Primary
DESCRIPTION	A user wants to listen to a 30 second preview of a song before buying a song. The preview is played. After the preview is played, the user is asked if they want to buy the song or not.

USE CASE	Manage Playlist
ACTORS	User
TYPE	Primary
DESCRIPTION	A user can create a new playlist, update it (add new songs), retrieve or delete their playlist.

USE CASE	Rate/Review
ACTORS	User
TYPE	Primary
DESCRIPTION	The user can either like or dislike a song/album or leave a comment on it.

USE CASE	Play/ Stop/ Pause/ Next/ Previous/ Resume/ Forward/ Rewind/ Shuffle/ Repeat song(s)
ACTORS	User
TYPE	Primary
DESCRIPTION	A user can play a song. He can stop, pause, forward 10s, rewind 10s a playing song. He can also select the next or the previous song while playing a song. Furthermore, he can resume a paused song. He can also put the current song on repeat mode or put all the songs on shuffle mode (randomly selected)

USE CASE	Enter Payment
ACTORS	User
TYPE	Primary
DESCRIPTION	A user wants to buy a song/album. The user clicks on "BUY NOW". If the user's payment details have been previously saved, they only need to enter their pin. After the pin is verified, they are asked to confirm their payment. If their payment details have not been stored, they are asked to enter their card details and pin. After the pin is verified, they are asked to confirm their payment. After their payment is complete, they are asked if they want to save their payment details.

USE CASE	View Lyrics
ACTORS	User
TYPE	Primary

DESCRIPTION	**The user wants to view the lyrics of the song they are listening to. User selects show lyrics options. The lyrics are shown to the user. The lyrics disappear after the song is finished.**

USE CASE	**Sort Playlist/Library**
ACTORS	**User**
TYPE	**Primary**
DESCRIPTION	**A user can sort their playlist/ library according to name, genre, artist, years (Ascending or descending). On completion, the user's playlist is sorted.**

USE CASE	**Search**
ACTORS	**User**
TYPE	**Primary**
DESCRIPTION	**The user can search for a song/album from his library/ playlist. On completion, if the song/album is available it is shown to the user else a "No result found" message is displayed.**

USE CASE	**Share**
ACTORS	**User**
TYPE	**Primary**
DESCRIPTION	**A user wants to share a song/album with their friend. After clicking on share they can type in the email of their friend(user) and share the URL**

	with them. On completion, a URL is shared with their friend(s). The song snippet plays if they have not bought it, else full song plays.

USE CASE	**Invite Friends**
ACTORS	**User**
TYPE	**Primary**
DESCRIPTION	**A user wants to invite their friend(s) to the application. They send them an invite. On completion, an email containing a URL of creating account is sent to their friend(s).**

USE CASE	**Plagiarism Checker**
ACTORS	**System, Artist**
TYPE	**Secondary**
DESCRIPTION	**After an artist adds a new song/album, it is checked for plagiarism. The lyrics of the new song or all the songs in an album are compared with previous records in the database. If a ong has 50% or more content plagiarized then the artist is notified and the song is removed from the application due to copyright issue. If the song is less than 50% plagiarized then it is verified and added to application.**

USE CASE	**Music Recommender**
ACTORS	**System, User**
TYPE	**Primary**

DESCRIPTION	**A user is recommended new/unheard songs based on their previous ratings (type of songs mostly listened) by the system. On completion, the user is provided with a list of songs that would best match their interests.**

USE CASE	**Verify Credentials**
ACTORS	**System, Artist, User**
TYPE	**Secondary**
DESCRIPTION	**When an artist/user adds their payment details the system verifies them before payment is made.**

USE CASE	**New Releases**
ACTORS	**System**
TYPE	**Primary**
DESCRIPTION	**A list of all new song/album releases is maintained.**

USE CASE	**Manage Song/Album**
ACTORS	**Artist**
TYPE	**Primary**
DESCRIPTION	**An artist can add a new song/album. An artist can delete a song/album.**

USE CASE	Get Paid
ACTORS	Artist
TYPE	Primary
DESCRIPTION	An artist wants to get paid for their services. They enter their payment details (card number). Their payment details are verified by the system. Now every time a user buys their song, a percentage of the payment made is transferred to the artists account.

USE CASE	Reply/Review Comments
ACTORS	Artist
TYPE	Primary
DESCRIPTION	An artist can view the comments made about their song/album. An artist can reply to the comments.

USE CASE	Manage Artist
ACTORS	Artist
TYPE	Primary
DESCRIPTION	An artist wants to create an account. They enter their information (email, password, name, etc.). Their information is verified by the system and then their account is created (does any such email really exists?). Any new information or update must be verified by the system.

EXTENDED USE CASES:

1. Get Paid
2. Share Songs
3. Rate/ Review
4. Create Playlist
5. Preview Songs
6. Verify Credentials
7. Plagiarism Checker
8. Music Recommender
9. Manage/Songs/ Album
10. Reply/ Review Comments
11. Play/ Stop/ Pause/ Next/ Previous/ Resume/ Forward/ Rewind/ Repeat/ Shuffle Songs

Use Case Name: Get Paid

Scope: Music Management System

Level: User Goal

Primary Actor: Artist

Stakeholders & Interests:

- **User:**
  - Buy songs
- **Artist:**
  - Will get a percentage of money the user pays per song
- **System:**
  - Will actually transfer the money from user's account to artist's account

Preconditions:

- Artist must be logged in with email address and password

Success Guarantee (Post Condition):

- Money transferred to Artist's account

Main Success Scenario:

Actor/Action	System Response
1. User selects the option to buy song.	

	2. System asks for user's credentials (credit card number and password)
3. User enters credentials	
	4. System verifies it if credentials entered are authentic or not.
	5. System transfers a percentage of that amount to artist 's account whose song is bought.
	6. System displays message "Money Successfully transferred"
	7. System displays message "Song Bought"

Extensions:

- At any time, system crashes
  - Shows crash report
  - System reconstructs prior state
  - System remembers at which users were selected for sharing

- User selects "No"
  - Song is not bought
  - Only preview of songs appears
  - System displays message "Song Not Bought"

- Payment credentials are not verified
  - Song is not bought
  - Only preview of songs appears
  - System displays message "Song Not Bought, Error Occurred! "

Use Case Name: Share Songs

Scope: Music Management System

Level: User Goal

Primary Actor: User

Stakeholders & Interests:

- **User:**
  - Wants to share song(s) with other user(s) through a URL
- **System:**
  - Will actually send the URL to another user's email

Preconditions:

- User must be logged in with email address and password

Success Guarantee (Post Condition):

- URL sent

Main Success Scenario:

Actor/Action	System Response
8. User selects the option to share song.	

	9. Systems display search option to enter user's username.
10. User enters username and marks it until all users are selected.	
	11. System saves the marked users and displays.
	12. Systems sends a URL link to their respective emails

Extensions:

- At any time, system crashes
  - Shows crash report
  - System reconstructs prior state
  - System remembers at which users were selected for sharing

- User selects "No"
  - System does not send the URL to other users
  - System display message "No link send"

Use Case Name: Rate/ Review

Scope: Music Management System

Level: User Goal

Primary Actor: User

Stakeholders & Interests:

- **User:**
  - Like/ Dislike a song
  - Comment on a song
  - Rate a song
  - Reply to other people's comment/ rating

- **System:**
  - Will show the user option on the screen comment section

Preconditions:

- User must be logged in with email address and password

Success Guarantee (Post Condition):

- Like successfully shown on screen
- Comment on song successfully shown on screen
- Rating successfully shown on screen
-

Main Success Scenario:

1. **Like/ Dislike**

Actor/Action	System Response
1. User opens a song to play	
	2. Systems plays that song.
3. User select like button (thumbs up).	
	4. Systems records user's reactions and increments the like number.
	5. Systems display the new number of like on the song.

2. **Comment**

Actor/Action	System Response
1. User opens a song to play	
	2. Systems plays that song.
3. User select comment button.	
	4. System opens a dialogue box for user to write.
5. User writes the comment	
	6. System records the comment
	7. Systems display the comment in the chat box.

3. **Rating**

Actor/Action	System Response
1. User opens a song to play	
	2. Systems plays that song.
3. User select rating button.	

	4. System opens a dialogue box for user to write. (only digits)
5. User writes the number (out of 10)	
	6. System records the rating.
	7. Systems display the rating in the chat box

4. Reply

Actor/Action	System Response
1. User opens a song to play	
	2. System play that song
	3. System is displaying the current comments of people in the chat box
4. User selects the arrow button to reply to specific comment/ rating.	
	5. System open a dialogue box for user to enter.
6. User enter the message	
	7. System records that message
	8. System display that message as a reply to the chosen comment.

Extensions:

- At any time, system crashes
  - Shows crash report
  - System reconstructs prior state
  - System remembers at which users were selected for sharing

- User selects "No"
    - Comment/ Rating/ Like/ Reply not shown on chat box
    - System displays message "Nothing Displayed"

- User selects "Delete"
    - Comment/ Rating/ Like/ Reply deleted from chat box
    - System displays message "Information deleted"

Use case Name: Create playlist

Scope: Music Management System

Level: User Goal

Primary Actor(s): User

Stakeholders and Interest:

- **User:**
  - User can create its own playlist

Preconditions:

- The User must be logged in to their account and be verified

Success Guarantee (Post Condition):

- Playlist is created

Main Success Scenario:

Actor/Action	System Response
1. Artist/ User chooses the menu button.	
	2. System shows a list of songs from user's library.
3. User selects the songs.	

	4. System asks the user to enter the name of playlist.
5. User enter the name.	
6. User selects" Done"	
	7. System created playlist and shows the message "Playlist Created"

Extensions:

- At any time, system crashes
    - Shows crash report
    - System reconstructs prior state
    - To support recovery and correct uploading/deleting/retrieving, ensure all the actions can be retrieved at any scenario

- User selects "No"
    - System displays message "Playlist Not Created"

Use Case Name: Preview songs

Scope: Music Management System

Level: User Goal

Primary Actor(s): User

Stakeholders and Interest:

- **User:**
  - User wants to preview a song.

Preconditions:

- The User must be logged in to their account and be verified

Success Guarantee (Post Conditions):

- "BUY NOW" dialog box appears after previewing one song

Main Success Scenario:

Actor/Action	System Response
1. User searches for the song/album.	
	2. System shows the result related to the search.
3. . User chooses a song.	

	4. System plays the song.
	5. System shows “BUY NOW” dialog box after preview is played.
	6. System asks the user whether they want to buy the song or want to go back to home screen.
7. User selects the to “Go back” .	
	8. System shows the home screen.

Extensions:

- No search found
    - System display an error “No search found”

Use Case Name: Verify Credentials

Scope: Music Management System

Level: User Goal

Primary Actor: User

Stakeholder and Interest:

- **User:**
    - Wants his credential to be verified and be successful so that he can get access to the song for lifetime

- **Artist:**
    - Wants credentials of the user to be verified with no errors and smooth payment to be done. Wants to get notified when someone buys their song with a receipt generated

Precondition:

- System should be verified and authenticated
- User must be logged in with email address and password
- Song/Album which needs to be bought should be selected

Success Scenario (Post Conditions):

- Song/Album successfully bought
- Has been added to the user's playlist

Main Success Scenario:

Actor/Action	System Response
1. User selects the song/album of their interest	
2. They click the option of "BUY NOW"	
	3. Screen opens to ask the user to enter their credentials details.
4. User enters their card number and security code.	
	5. Card is verified by its foreign key.
	6. Save/Not Save option is displayed.
7. User selects the appropriate option.	
	8. If the save option is selected, the system saves users credentials in the database otherwise it does not.
	9. Full song is made available to user
10. User can now access the full song.	

Extension:

- At any time, system crashes
  - System reconstructs prior state
  - User can restart app and ask for recovery
  - Ensure that if anyone is in the middle of buying any song, his credential should not be lost or vulnerable

- At any time if administrator wants to cancel user access
  - He overrides user access and deletes user credential and account.

- Credentials not verified
    - Display Error
    - Ask user to enter again

Special Requirements:

- Font visible for user to read
- Touch screen UI with attractive style
- Language internalization to the text to be displayed

Technology and data variations List:

- Authentication message to be sent on user mail
- Credit information should be entered by the user via keyboard

Frequency of occurrence:

- This service will be frequently used as there will be hundreds of users using this application and buying songs at the same time

Use Case Name: Plagiarism Checker

Scope: Music Management System

Level: System Goal

Primary Actor: Artist

Stakeholder and Interest:

- **Artist:**
  - Wants to make sure they are not uploading copyrighted content.
- **System:**
  - Wants to ensure content uploaded is not stolen.

Precondition:

- Artist must be verified.
- Artist must be logged in with email and password
- Song which needs to be checked should be selected.

Success Scenario (Post Conditions):

1. Song not plagiarized
2. Song successfully checked

Main Success Scenario:

Actor/Action	System Response

1. Artist adds a new Song.	
	2. System asks for plagiarism check.
3. Artist selects okay.	
	4. System checks song lyrics for plagiarism.
	5. System compares the lyrics of the new song with lyrics of songs in the database.
	6. If the song is less than 50% plagiarized then it is verified and the artist is notified.
	7. . Song is made available to users.

Extension:

- At any time, system crashes
  - Shows crash report
  - System reconstructs prior state
  - To support recovery and correct uploading/deleting/retrieving, ensure all the actions can be retrieved at any scenario

- At any time if Artist wants to cancel plagiarism check request
  - Artist cancels request
  - System halt plagiarism check

- Song Plagiarized
  - If the song is 50% plagiarized then it is deleted from the artist's directory.
  - The artist is notified.

Use Case Name: Music Recommender

Scope: Music Management System

Level: User Goal

Primary Actor: User

Stakeholder and Interest:

- **User:**
  - Wants a recommendation of new/unheard songs that might interest them
- **System:**
  - Will display more songs of the type that user has already purchased

Precondition:

- The User must be logged in to their account and be verified

Success Scenario (Post Conditions):

- List of relevant songs successfully provided to the user

Main Success Scenario:

Actor/Action	System Response
1. User opens the recommendation tab.	

	2. System checks the user's profile and list of all songs.
	3. System then checks music that would match the user's type of songs.
	4. System then recommends the user songs previously not previewed or bought by the user.
5. User sees a list of new songs recommendations	

Extension:

- At any time, system crashes
  - Shows crash report
  - System reconstructs prior state
  - To support recovery and correct uploading/deleting/retrieving, ensure all the actions can be retrieved at any scenario

Use case Name: Manage Song/Album

Scope: Music Management System

Level: User Goal

Primary Actor(s): Artist, User

Stakeholders and Interest:

- **Artist:**
  - Wants to create/retrieve/delete or delete his songs
- **User:**
  - User can retrieve/delete their favorite song/ album/ artist

Preconditions:

- The Artist/ User must be logged in to their account and be verified

Success Guarantee (Post Condition):

- Album/ song is created/ uploaded
- Album/ song is deleted
- Album/ song is retrieved

Main Success Scenario:

1. **Create/ Upload:**

Actor/Action	System Response
1. Artist chooses the menu button.	
	2. System shows a list of options.
3. Artist chooses to upload the new album/song.	
	4. System asks the user to select the type(single/album) they want to upload.
5. Artist selects the type.	
	6. System asks the user to upload the song/album and shows the select file dialog box.
7. Artist uploads the file.	
	8. System asks the user to enter the name, date, lyrics, and genre.
9. Artist enters the information.	
10. Artist selects the upload button.	
	11. System ask the user to confirm the uploading
12. Artist confirms the upload.	
	13. System uploads the song/ album and shows the message "Uploaded Successfully".

2. Retrieve:

Actor/Action	System Response
1. Artist/ User chooses the menu button.	
	2. System shows a list of options.

3. Artist/ User chooses to retrieve the album/song.	
	4. System asks the user to enter the song/album.
5. Artist/ User enters the name of the album/song.	
	6. System displays the list of the song related to search.
7. Artists/ User selects the song	
	8. System plays the song

3. Delete:

Actor/Action	System Response
1. Artist/ User chooses the menu button.	
	2. System shows a list of options.
3. Artist/ User chooses to delete the album/song.	
	4. System asks the user to select the song/album they want to delete.
5. Artist/ User selects the song(s).	
	6. System asks the user to confirm the deletion.
7. Artist/ User selects “Yes”.	
	8. System deletes the selected item and displays the message “Deleted Successfully”

Extensions:

- At any time, system crashes
    - Shows crash report
    - System reconstructs prior state
    - To support recovery and correct uploading/deleting/retrieving, ensure all the actions can be retrieved at any scenario

Create:

- Artist uploads the wrong file
    - System shows the uploading file dialog box again
    - Artist selects the new file

- Artist selects “No"
    - File uploading stops
    - System displays the message “Uploading Failed”

Retrieve:

- No search found
    - System display an error “No search found”

Delete:

- Artist selects “No”
    - System does not delete files
    - System displays message “No deleted files”

Use Case Name: Reply/ Review Comments

Scope: Music Management System

Level: User Goal

Primary Actor: Artist

Stakeholders & Interests:

- **Artist:**
  - Like/ Dislike another artist's song
  - Comment on a song
  - Rate another artist's song
  - Reply to other artist's comment/ rating

- **System:**
  - Will show the artist option on the screen comment section

Preconditions:

- Artist must be logged in with email address and password

Success Guarantee (Post Condition):

- Like on another artist's song successfully shown on screen
- Comment on song successfully shown on screen
- Rating successfully shown on screen

Main Success Scenario:

5. **Like/ Dislike**

Actor/Action	System Response
6. Artist opens a song to play	
	7. Systems plays that song.
8. Artist select like button (thumbs up).	
	9. System makes sure that song is not of the artist asking to like it
	10. Systems records user's reactions and increments the like number.
	11. Systems display the new number of like on the song.

6. **Comment**

Actor/Action	System Response
8. Artist opens a song to play	
	9. Systems plays that song.
10. Artist select comment button.	
	11. System opens a dialogue box for user to write.
12. Artist writes the comment	
	13. System records the comment
	14. Systems display the comment in the chat box.

7. **Rating**

Actor/Action	System Response

8. Artist opens a song to play	
	9. Systems plays that song.
10. Artist select rating button.	
	11. System opens a dialogue box for user to write. (only digits)
12. Artist writes the number (out of 10)	
	13. System makes sure that song is not of the artist asking to rate it
	14. System records the rating.
	15. Systems display the rating in the chat box

8. Reply

Actor/Action	System Response
9. User opens a song to play	
	10. System play that song
	11. System is displaying the current comments of people in the chat box
12. User selects the arrow button to reply to specific comment/ rating.	
	13. System open a dialogue box for user to enter.
14. User enter the message	
	15. System records that message
	16. System display that message as a reply to the chosen comment.

Extensions:

- At any time, system crashes
    - Shows crash report
    - System reconstructs prior state
    - System remembers at which users were selected for sharing

- User selects "No"
    - Comment/ Rating/ Like/ Reply not shown on chat box
    - System displays message "Nothing Displayed"

- User selects "Delete"
    - Comment/ Rating/ Like/ Reply deleted from chat box
    - System displays message "Information deleted"

Use Case Name: Play/ Stop/ Pause/ Next/ Previous/ Resume/ Forward/ Rewind/ Repeat/ Shuffle Songs

Scope: Music Management System

Level: User Goal

Primary Actor: User

Stakeholders & Interests:

- **User:**
  - Wants to play/ stop/ pause/ next/ previous/ resume/ forward/ rewind/ repeat/ shuffle stop a song from their library/playlist
- **System:**
  - Will actually play/ stop/ pause/ next/ previous/ resume/ forward/ rewind/ repeat/ shuffle stop a song

Preconditions:

- User must be logged in with email address and password

Success Guarantee (Post Condition):

- Song is played
- Song is paused
- Song is stopped
- Song is resumed
- Song is repeated

- Songs are shuffled
- Next song is played
- Song is rewind by 10s
- Song is forward by 10s
- Previous song is played

Main Success Scenario:

9. Song is Played

Actor/Action	System Response
13. User selects the song of their interest from their library/playlist.	
	14. Systems plays that song.

10. Song is Paused

Actor/Action	System Response
	1. A song is already playing.
2. User presses the pause button.	
	3. System pauses that song.

11. Song is Stopped

Actor/Action	System Response
	1. A song is already playing.
2. User presses the stop button.	
	3. System stops playing that song.

12. Song is Resumed

Actor/Action	System Response

	1. A song is already paused.
2. User presses the resume button.	
	3. System resumes that song.

13. Song is repeated

Actor/Action	System Response
1. User selects the repeat button.	
	2. System puts the song in repeat mode until something else selected. Same song is in repeat mode now.

14. Songs are shuffled

Actor/Action	System Response
1. User selects the shuffle button.	
	2. System puts the song in shuffle mode until something else selected. Songs are randomly selected from playlist/library.

15. Next song is played

Actor/Action	System Response
	1. A song is already playing.
2. User selects the next button,	
	3. System plays the new song from the user's library/playlist

16. Song is rewind by 10s

Actor/Action	System Response

	1. A song is already playing.
2. User selects the rewind button.	
	3. System rewinds the current song by 10s

17. Song is forward by 10s

Actor/Action	System Response
	1. A song is already playing.
2. User selects the forward button.	
	3. System forwards the current song by 10s

18. Previous song is played

Actor/Action	System Response
	1. A song is already playing.
2. User selects the previous button,	
	3. System plays the previous song from the user's library/playlist

Extensions:

- At any time, system crashes
  - Shows crash report
  - System reconstructs prior state
  - System remembers at which point the song stopped playing

Special Requirements:

- Language should be English
- Attractive graphical user interface
- Font and size of font should be visible and clear for the user to read
- Once the song has been selected, it should take long to play it actually

Technology & Data Variations Lists:

- Songs are played by user via a touchscreen or through buttons
- Songs played are listened through a speaker, earphone, or headphone

Frequency of Occurrence:

- This service should be frequently used as there will be hundreds of users using this application and playing songs at the same time