ONLINE MUSIC PLAYER APPLICATION FOR ARTISTS AND LISTENERS

2018503561

MN- CT
Anna University, MIT Campus

Online Music Player Application for Artists and Listeners

Project Description

Project Title	***Online Music Player Application for Artists and Listeners***
Project Leader	***K.Soundarya***

This project deals with an online music player run in a web page that is created for artists, or the producers of the music, and for the listeners who use this application, or the consumers.

Unique functionalities exist for the artists and the listeners respectively, along with the common functionalities that will exist.

The listeners and the artists will have to register prior to usage.

This application is dynamic and interactive.

SRS Document

Table of Contents

Revision History

Name	Date	Reason for Changes	Version

1. Introduction

1.1 Purpose

The online music player is an application that is used to digitize and monetize music released by artists to the listeners through an online registration process which is available separately for both the artists and the listeners. It aims at minimizing the illegal use of music and improving the interaction of the artists with respect to the listeners, and branding.

1.2 Intended Audience and Reading Suggestions

This SRS is mainly developed for the project development team. In this team there are the project manager, developer, coder, tester and documentation writer and the user of the project too.

Users (Artists and Listeners)

This document is intended to ensure that it satisfies the needs of the customers, the artists and the listeners who would use the application.

Project Manager

This SRS document is also very important for the project manager as it helps in cost estimation which can be performed by referring to the SRS document and it contains all the information that is required for the project plan.

Project Developer

The project developer will refer to the SRS document to ensure that the product developed is as per the needs of the customers.

Tester

The tester reads the SRS document to ensure that the requirements are understandable based on the functionality specified so that she or he can test the software and validate it’s working.

Document Writer

The document writer reads the SRS document and understands the exact requirements so that they can write user manuals without any errors based on the SRS document.

Maintenance

The SRS document helps the maintenance engineers to understand functionality of the system. A clear knowledge of the functionality is needed to design and code.

1.3 Product Scope

Registration into the portal is the first step to access/post music in the application.

Artists

For registration, the artists will have to first provide information such as their name, label/distributor name, and all other necessary details, along with a valid account number, after which their names will be verified in the application as an artist. If in case the artist isn't a part of any label, or do not have a music distributor yet, they will have to fill another form, to choose one provided by the software application team itself, compulsorily, before registering as an artist in the music player application.

Listeners

The users will have to first register for free, i.e., fill details like their name, email ID, username in the application, password, and other details, after which they can use the application. An optional provision for paid membership is also provided, for which the consumers will have to pay a certain amount of money to enjoy certain privileges. This can either be opted at the initial registration, or later on too.

1.4 Definitions, Acronyms and the Abbreviations

- Administrator - Refers to the super user who is the Central Authority who has been vested with the privilege to manage the entire system. It can be any higher official in the Senior Development Team of the application's company.
- User - The listeners and the artists who wish to make use of the application.
- Label/Music Distributor – The legal team behind the artists, who manages such brands and trademarks, coordinates the production, manufacture, distribution, marketing, promotion, and enforcement of copyright for sound recordings and music videos.
- Music Player - Refers to this Online Music Player Application platform.
- HTML - Markup Language used for creating web pages.
- HTTP - Hyper Text Transfer Protocol.
- TCP/IP – Transmission Control Protocol/Internet Protocol is the communication protocol used to connect hosts on the Internet.
- JSON - JavaScript Object Notation
- API - Application programming interface

1.5 References

https://developer.spotify.com/

2. Overall Description

2.1 Product Perspective

The online music player application acts as an interface between the 'artists' and the 'listeners.' This system tries to make the interface as simple, and at the same time, as dynamic as possible. Thus, legalization, monetization and digitization of music by creators are enforced through this application.

2.2 Product Function

- Secure Registration of information by the Applicants.
- Provide personalized dashboards depending on the type of the user.
- Provide access to advertisers.
- Provisions for inclusion of upgradation to the premium version of the application, to the listeners.
- Public/private profiles can follow other public/private profiles. In case of private accounts, user will be prompted to accept request or to ignore request. Listeners have the option of keeping their profiles private or public, whereas the artists' profile is public by default.

2.3 Operating Environment

Particulars	Client System	Server System
Operating System	Windows/Linux/Macintosh	Linux
Processor	Intel or AMD	Intel or AMD
Hard Disk	1 GB	1 TB
RAM	256 MB	8 GB

2.4 Design and Implementation Constraints

The applicants require a computer to submit their information. The artist, who is currently registering must be directed to the distributor selection form link if it's found out that they don't initially have one.

The applicants require a computer to submit their information. Although security is given high importance, there is always a chance of intrusion in the web world which requires constant monitoring.

2.5 Assumptions and Dependencies

- The Applicants must have basic knowledge of computers and English Language.
- The listeners may be required to pay for the premium version.
- The artists labels/distributors as well as the artists' bank account numbers will be needed.
- Each User must have a User ID and password.
- Internet connection is a must
- Proper browsers should be installed in the user's system

3. External Interface Requirements

3.1 User Interfaces

- Artists – The producers to the contents of this application, they can publish their albums, singles, collaborations and playlists of songs to the platform, for the listeners.
- Listeners – She/he can browse for artists/songs/albums/playlists in the application.
- Music distributor/Label – This is the external agency related to the artists. It is absolutely necessary for the artists to have a distributor or to be in a label. If not, the application will prompt the artists to fill a form to choose one from the many that the application itself provides.

3.2 Hardware Interfaces

Web APIs are necessary. Servers will have to be placed in a secure place.

3.3 Software Interfaces

• Front End Client - The online interface is built using HTML and JavaScript engines.

• Web Server - Glassfish application server (Oracle Corporation).

• Back End - Oracle database.

3.4 Communications Interfaces

This application uses internet. Hence, it uses HTTP for transmission of data. This protocol allows easy interaction between client and server. The objects will be stored in the JSON format.

4. System function

- Secure Registration of information by the Applicants.
- Add songs

- Add followers
- Add playlists
- Initiate follow requests

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Login into the system will take seconds.
- Registration process can be lengthy.
- Individual network speed should also be considered.

5.2 Safety Requirements

The user details are to be maintained at high privacy. Since the application is accessible only through the internet, the application should be made hack-proof with strict security protocols. Also, the

5.3 Software Quality Attributes

Availability

Anyone with an internet connection and an account in the portal can easily use the interface. The music player can be accessed in Mac, Windows and Linux OS.

Maintainability

Since music from artists are released in large numbers on a day-day basis, the application requires daily maintenance.

Usability

Listeners with an account can use the application. Artists with a distributor/label can only use the application.

5.4 Business Rules

- Advertisements are to be screened after every ten songs played by the listener in the application, and strictly only to the non-premium versions. All users with premium version will not have this feature.
- Any song that doesn’t meet the community guidelines can be taken down from the platform to protect the listeners’ and the application’s sentiments.
- One distributor or a label can host one or more artists.
- The artists’ work will be paid for only through their label/distributor.

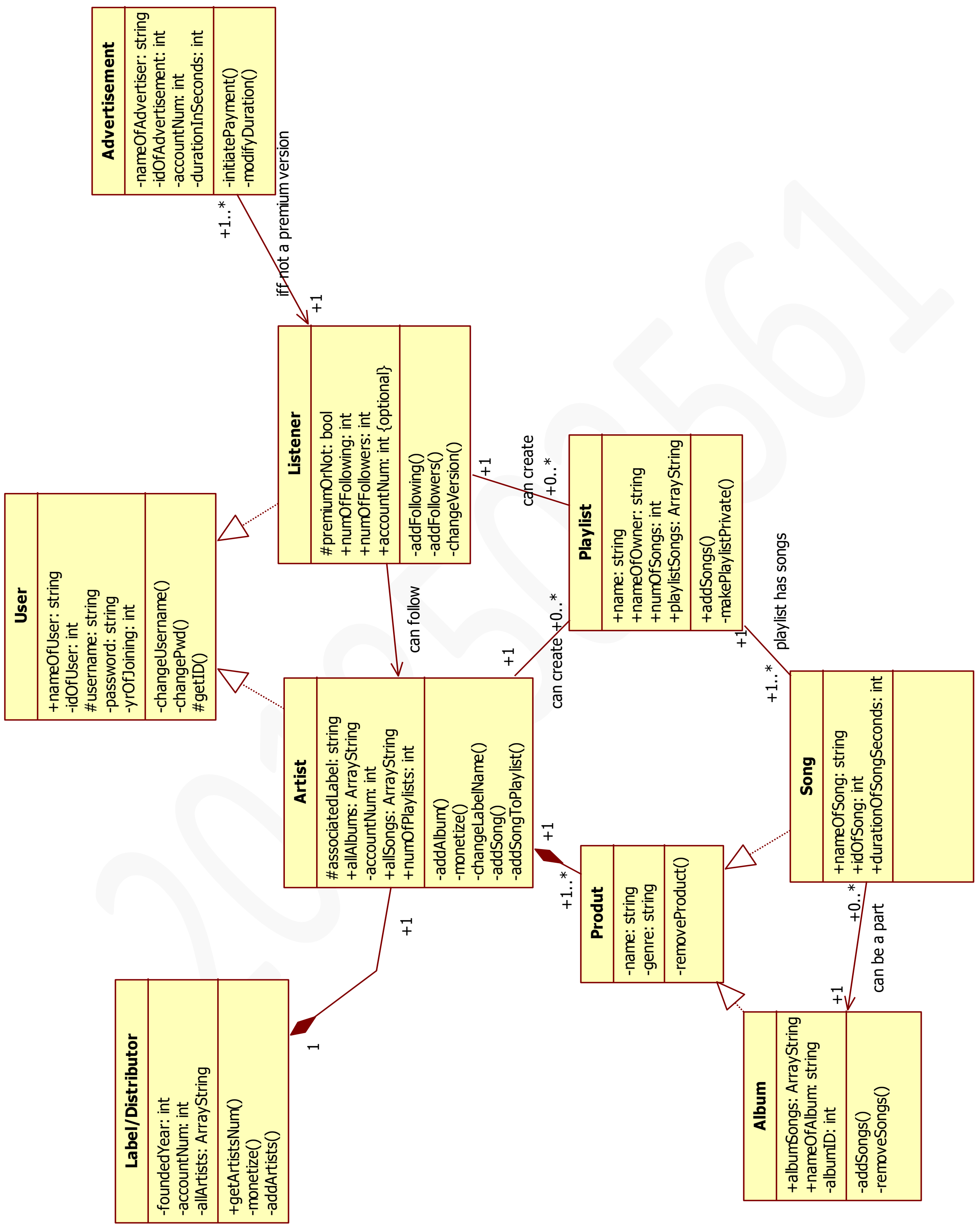
- The work of artists is monetized if and only if the average stream of the artist's work by the listeners is more than a thousand. Until then the work will not be paid for by the application.

Appendix A: Glossary

Definitions, Acronyms and Abbreviation:

- SRS: Software Requirement Specification
- Client/User: The entity who will be using the application
- Server: A system that runs in Linux that monitors the application
- HTTP: Hyper Text Transfer Protocol
- RAM: Random Access Memory
- Username: Unique name given to each account of digital library
- Password: Unique word given to each user as a secret code
- HTML, JavaScript, JSON, API

Class Diagram



The above Class Diagram has the following classes with attributes and functions:

1. User (nameOfUser, idOfUser, username, password, yrOfJoining, changeUsername(), changePwd(), getID())
2. Listener (premiumOrNot, numOfFollowing, numOfFollowers, accountNum, addFollowing(), addFollowers(), changeVersion())
3. Label/Distributor(foundedYear, accountNum, allArtists, getArtistsNum(), monetize(), addArtists())
4. Artist (associatedLabel, allAlbums, accountNum, allSongs, numOfPlaylists, addAlbum(), monetize(), changeLabelName(), addSong(), addSongToPlaylist())
5. Advertisement (nameOfAdvertiser, idOfAdvertiser, accountNum, durationInSeconds, initiatePayment(), modifyDuration())
6. Product (name, genre, removeProduct())
7. Album (albumSongs, nameOfAlbum, albumID, addSongs(), removeSongs())
8. Song (nameOfSong, idOfSong, durationOfSongSeconds)
9. Playlist (name, nameOfOwner, numOfSongs, plalistSongs, addSongs(), makePlaylistPrivate())

Inception

1. Vision

The vision of the project is to create a dynamic and interactive web page where music artists and listeners can discover each other easily.

The artists have a better chance of promoting their music through this application, and it gives them an opportunity to legally monetize their music.

The listeners who use this application to discover new songs will find it very easy to search for songs, as well as artists, and get a hands-on access to the playlists created by their favorite artists.

2. Scope

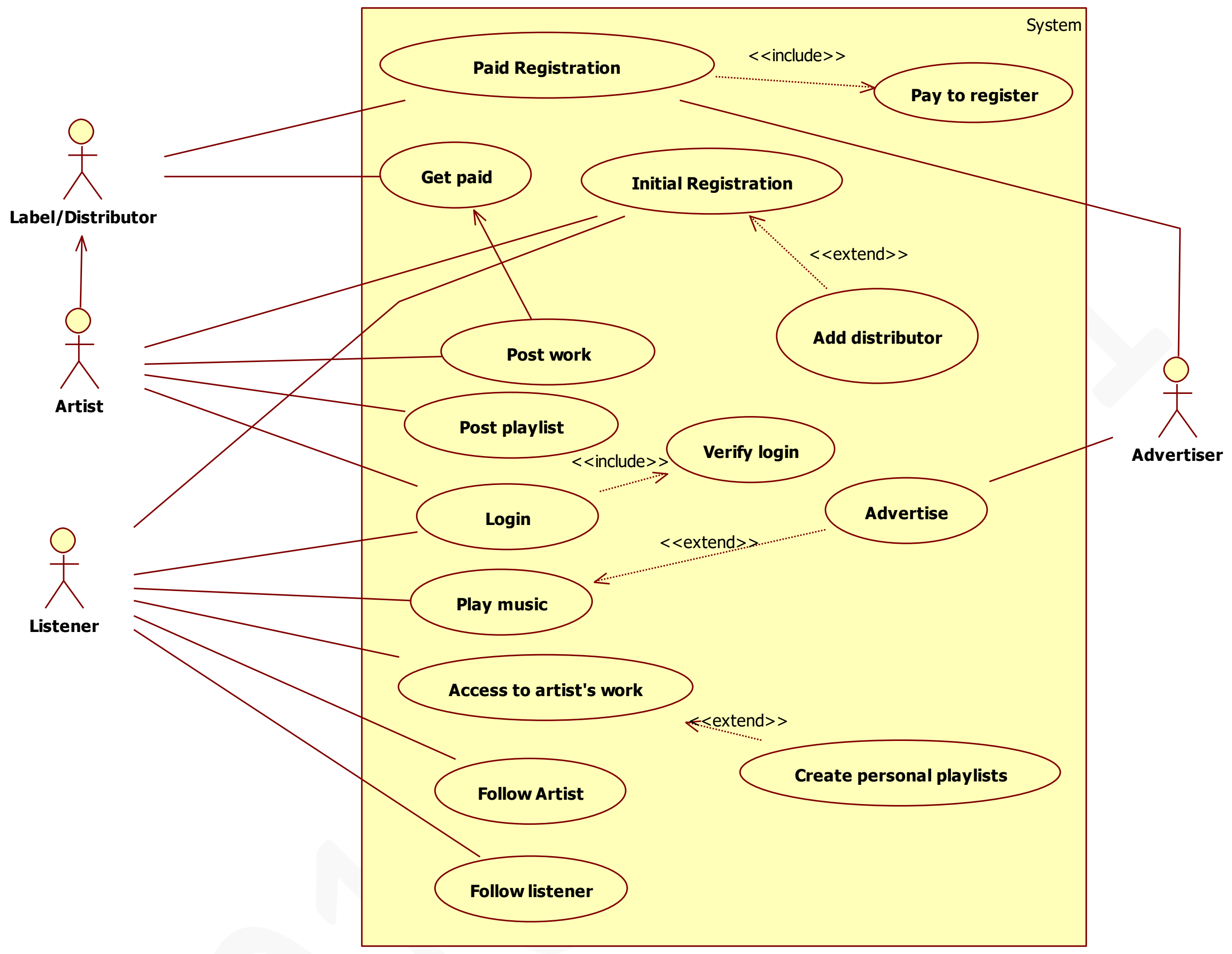
The music player application is a dynamic, and interactive web-page where artists can post their new/old singles, albums, or even collaborations done with other artists. Listeners who are in the look-out for new songs from their favorite artists, or for discovering new artists and expanding their musical horizons can easily achieve so through this application.

Usage of the application is restricted to only those who register into the application. This initial registration is different for listeners and artists. Both the artists and the listeners will have to fill up basic information about them (which will not violate their privacy), as well as the username and the password to be used in the application. The artists however will have to provide information on their label/distributor, and compulsorily choose one from the application's provided list if they don't have one initially. The listeners on the other hand, will have the option of paying for the premium version of the application.

Advertisers can easily advertise their products in this application to the non-premium version listeners. However, the advertisers have to register too, prior to usage. Listeners can follow artists as well as other listeners.

This application hence strives to be an interactive platform wherein the users can make use of the easy usability and manageability of the application to their own benefits.

3. Business Use-Case



The Primary actors are artist, listener and the label/distributor that manages the artist. The secondary actor is the advertiser, who cannot advertise without any non-premium version listener listening to music. The above business use case provides the most basic use cases that are expected to be included in the project.

The label/distributor has to pay to register in the application. This is one of the ways the application earns money. In the same way, the advertisers will have to pay to the application for advertising.

The listeners have an option of paying; she/he may choose to pay to have the premium version of the application, which has additional functionalities than that of the non-premium version. The artists on the other hand, are tied with one label/distributor who will pay them. As and when the former's works gain more view or/and the artist gains more following, the associated label would be paid by the application.

The Four Questions of Inception

1. Feasible?

Financial Feasibility

The type of investment chosen is bootstrapping initially. Building a simple application with the basic features initially, and then expanding based on the demand is to be opted. Even though bootstrapping involves a high risk, the platform's demand is expected to grow due to the lesser number of privately owned music player applications, with such vast and elaborate features. Increase in demand can further facilitate the attainment of loan from banks.

Initially, risk is involved, though considering good marketing, better UI/UX designs and addition of features based on feedback from the users, as well as government subsidies and personal debts, the application can be said to be financially feasible.

Market Feasibility

Demand for applications such as the one being proposed are fairly high in demand, because it gives the artists a place to digitally post their work. Any listener can also listen to the songs in the application before making a decision of buying any album, which saves their money if they aren't interested in the particular album. Many companies have realized the demand and come out with products in this domain. The main competitors would be Apple Music and Spotify, which are doing already good in the market. This application would strive to be less expensive and more creative, as well as user friendly than the other two platforms, and will give more focus to data protection and security of the users involved. Privacy would be given utmost importance, which are not so looked upon in the previous platforms, and which are of high demand in the market.

Thus, the product is market feasible.

So, to conclude the application is feasible, with high risk present initially.

2. Buy or/and Build?

The application can be built from scratch. The team behind the application is highly skilled and the confidence to build stems from this fact. A basic working prototype can be built first to ensure that all features are errorless, and with time advancements can be made. In the future maybe options of buying other businesses are an option, as the product grows and demand increases.

3. Rough Unreliable Range of Cost

Around Rs. 1.5lakh. Since the application mostly requires a laptop, and endless amount of coding, the building part of the project requires very little amount to start. Hiring developers is also considered for this rough unreliable cost which comes to around Rs. 75,000. Electronic gadgets required are to be bought with the remaining cost, and only if absolutely necessary owing to a tight budget.

After the application has been built with the most basic functionalities, the cost required would be around Rs. 10lakhs. This is because, the application has to pay to the artists as and when their work gets monetized. This is into consideration after all the other sources of revenue.

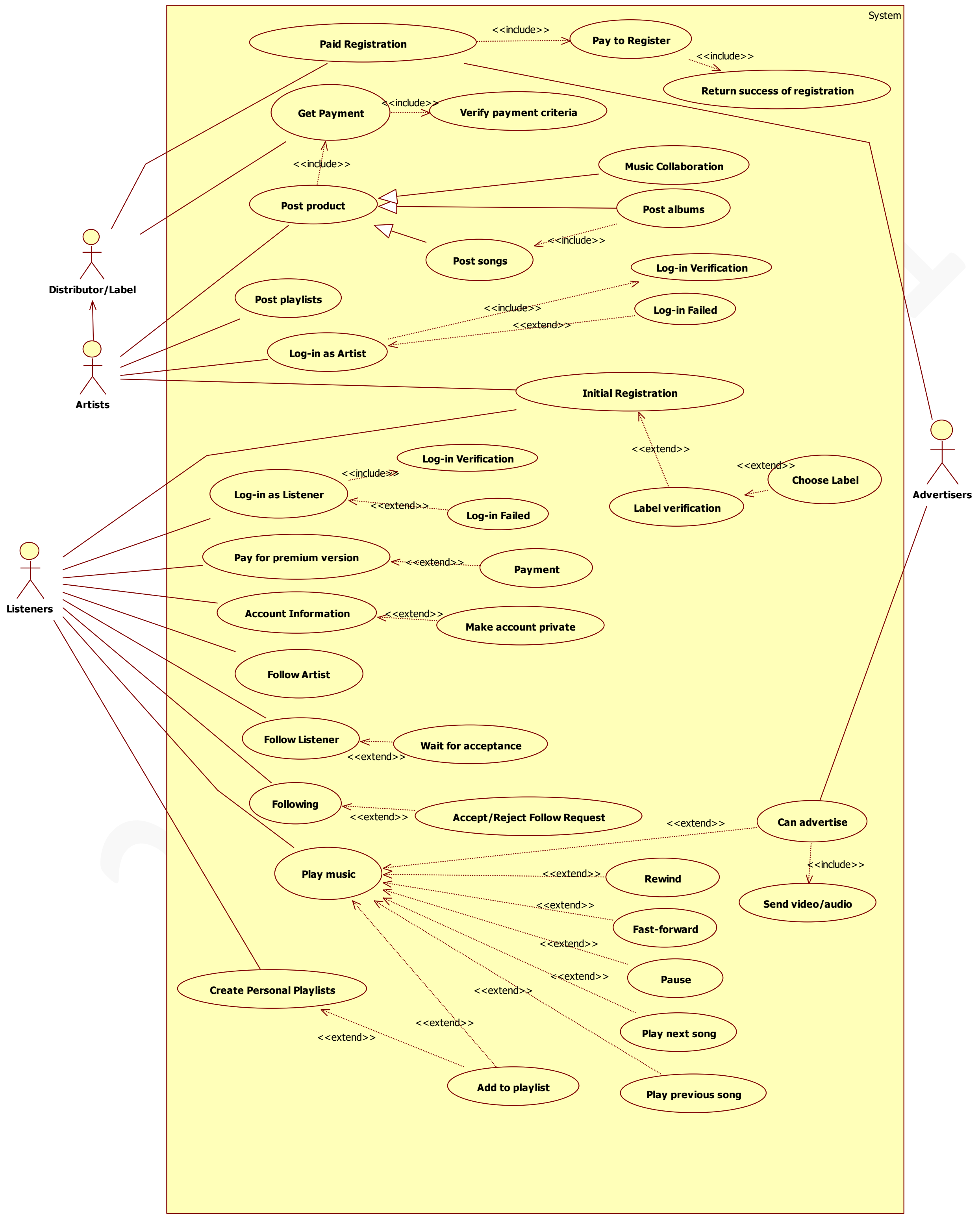
So, the overall rough unreliable range of cost would be around Rs. 1 lakh to Rs. 10 lakhs.

4. Should We Proceed or Stop?

The project can be continued.

Continuous and consistent effort incorporating self-financing, personal debt, sweat equity, minimized operating costs, inventory minimization, subsidy financing, and sales from the ongoing development of the project can be heavily relied upon to continue with the project. The growth of the project is estimated to only grow, after the initial, scratch buildup of the application.

Use Case Diagram



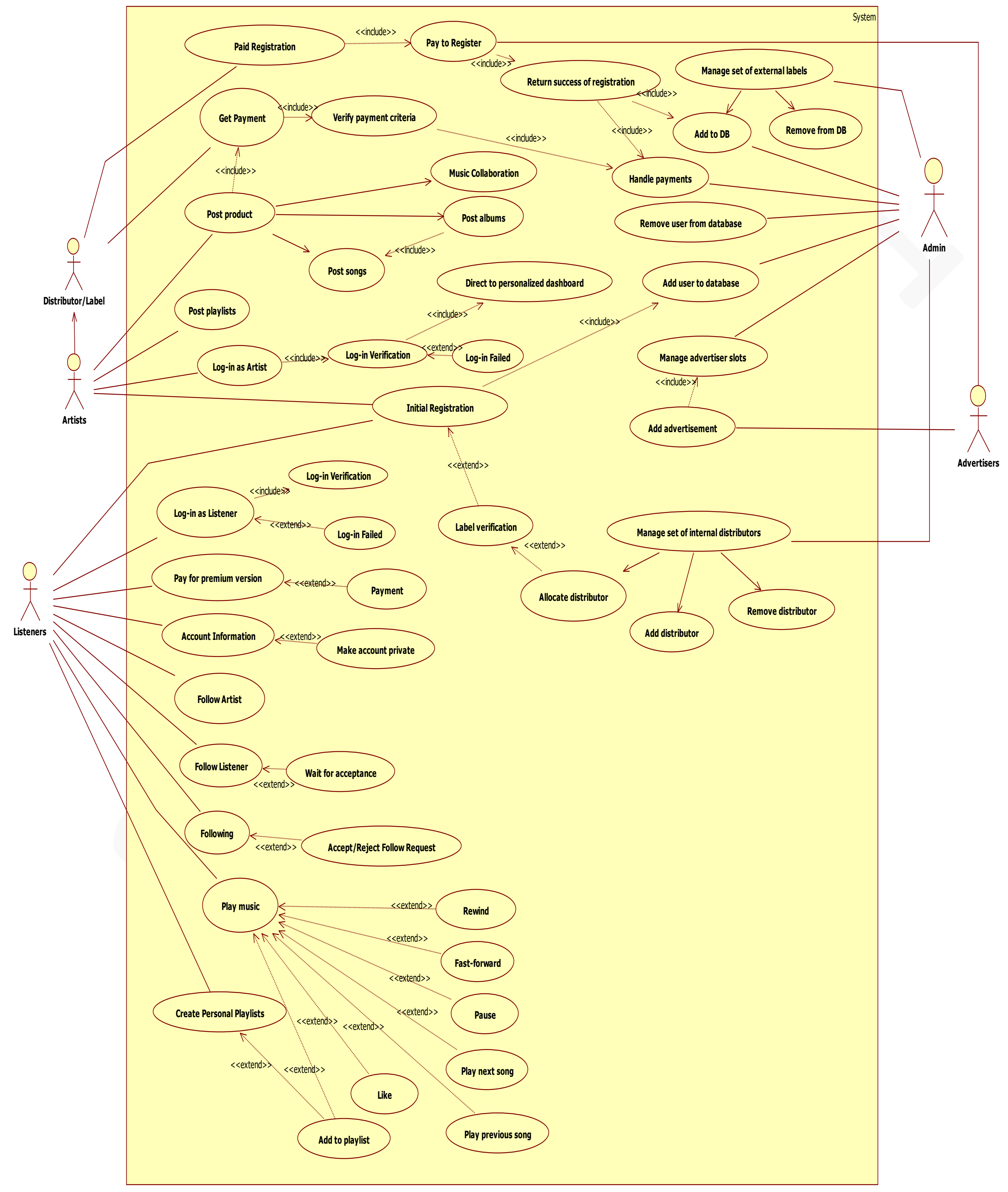
Description of any two Use Cases of the above Use Case Diagram:

Actors: Distributor/Label, Artists, Listeners, Advertisers

Use Case UC1	Initial Registration
Scope	Online Music Application
Level	User Goal
Goal in context	To register into the application before using it
Actors	Primary Actors: Artists Listeners
	Secondary Actor: Distributor /Label
Steps: Basic Flow	**Step 1:** Start. **Step 2:** Open the webpage of the application. **Step 3:** Click on the “Register if New” button. **Step 4:** Based on the type of user, click on “Register as Artists” or “Register as Listeners” button. **Step 5:** Fill in the basic information that is provided. The “*” mark indicates that the field has to be filled up without fail. **Step 5a:** In the case of an artist who doesn’t have a tie up with a label/distributor, click on the link provided to choose one provided by the application first. **Step 5b:** In the case of a listener, one can choose to pay for the premium version, or not. **Step 6:** Fill in a valid username. **Step 7:** Fill in a valid, strong password with at least 8 characters, at least one special character, at least one upper case alphabet and at least one lowercase alphabet. **Step 8:** Click on the Submit button after verifying all details to proceed into the portal. **Step 9:** End.

Use Case UC2	Play Music
Scope	Online Music Application
Level	User Goal
Goal in context	To play music that is available in the application
Actors	Primary Actors: Listeners
	Secondary Actor: Advertiser Artists Distributor/Label
Steps: Basic Flow	**Step 1:** Start. **Step 2:** Open the webpage of the application. **Step 3:** Log-in as listener into the application. **Step 4:** Select/search for a song to play **Step 5:** Click on the play button. **Step 6:** The rewind button can be clicked on if the user wants to move backward to one position of the song. **Step 7:** The fast-forward button can be clicked on if the user wants to move forward to one position of the song. **Step 8:** The Pause button can be clicked to stop playing the song for a brief period of time. **Step 8a:** Click on the Play button to resume playing the song from the initially left time-stamp. **Step 9:** Click on the “Add to Playlist” button to add the particular song to a playlist created by the listener. **Step 9a:** Click on the create playlist button to insert the song into a newly created playlist. **Step 10:** Click on the play next song button to play the next song in the album. **Step 11:** Click on the play previous song button to play the previous song in the album. **Step 12:** If in case the listener is a non-premium version user, and she/he gets an advertisement, watch/hear it fully because the application doesn’t allow skipping of advertisements for the former. **Step 13:** End.

Refined Use Case Model



The refined use case model is shown above. Other important and extended functionalities have been added. Another actor, the administrator(admin) has been added to the use case model with associated use cases such as managing the set of external and the internal (i.e., tied up with the application) labels/distributors, addition and removal of users and provision of slots to advertisers.

Description of any two Use Cases of the above Use Case Diagram:

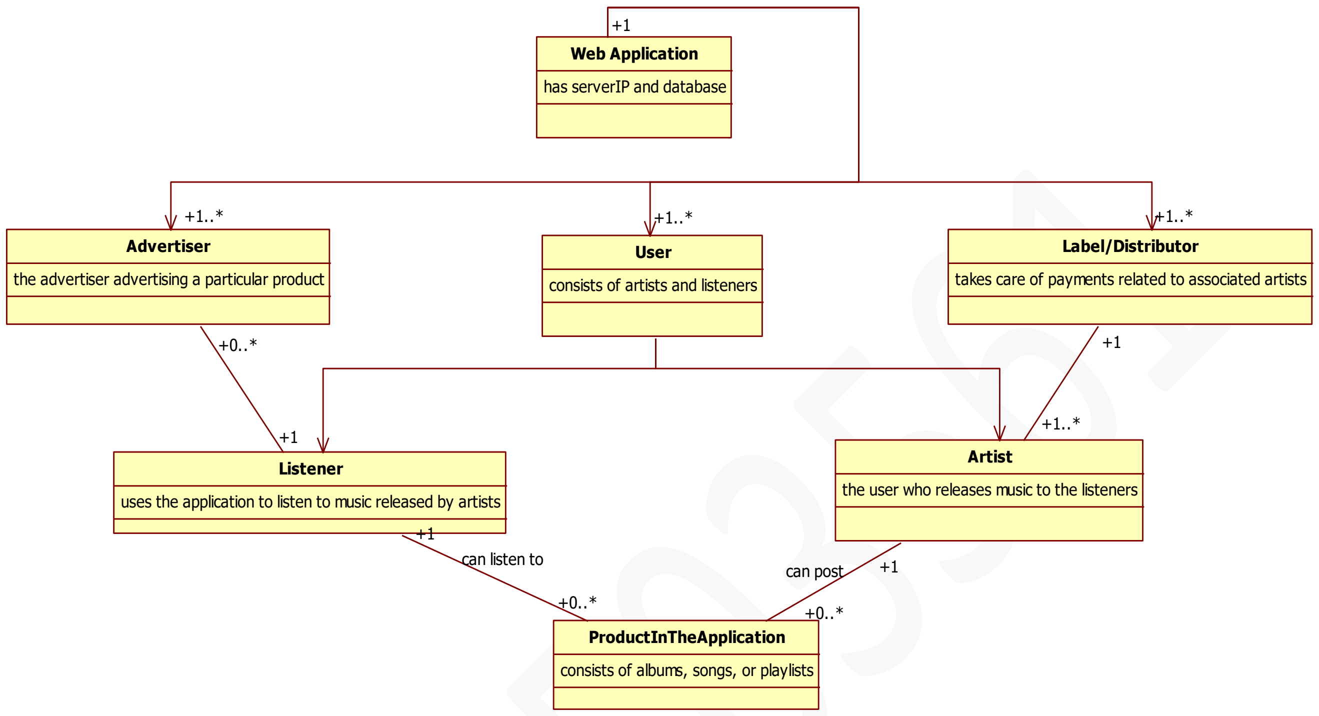
Actors: Distributor/Label, Artists, Listeners, Advertisers, Admin.

Use Case UC1	Initial Registration
Scope	Online Music Application
Level	User Goal
Goal in context	To register into the application before using it
Actors	Primary Actors: Artists Listeners
	Secondary Actor: Distributor /Label Admin
Steps: Basic Flow	**Step 1:** Start. **Step 2:** Open the webpage of the application. **Step 3:** Click on the "Register if New" button. **Step 4:** Based on the type of user, click on "Register as Artists" or "Register as Listeners" button. **Step 5:** Fill in the basic information that is provided. The "*" mark indicates that the field has to be filled up without fail. **Step 5a:** In the case of an artist who doesn't have a tie up with a label/distributor, click on the link provided to choose one provided by the application first. **Step 5b:** In the case of a listener, one can choose to pay for the premium version, or not. **Step 6:** Fill in a valid username. **Step 7:** Fill in a valid, strong password with at least 8 characters, at least one special character, at least one upper case alphabet and at least one lowercase alphabet. **Step 8:** Click on the Submit button after verifying all details to proceed into the portal.

	Step 9: Admin has to add the user to the database. **Step 10:** End.

Use Case UC2	Play Music
Scope	Online Music Application
Level	User Goal
Goal in context	To play music that is available in the application
Actors	Primary Actors: Listeners
	Secondary Actor: Advertiser Artists Distributor/Label Admin
Steps: Basic Flow	**Step 1:** Start. **Step 2:** Open the webpage of the application. **Step 3:** Log-in as listener into the application. **Step 4:** Select/search for a song to play **Step 5:** Click on the play button. **Step 6:** The rewind button can be clicked on if the user wants to move backward to one position of the song. **Step 7:** The fast-forward button can be clicked on if the user wants to move forward to one position of the song. **Step 8:** The Pause button can be clicked to stop playing the song for a brief period of time. **Step 8a:** Click on the Play button to resume playing the song from the initially left time-stamp. **Step 9:** Click on the “Add to Playlist” button to add the particular song to a playlist created by the listener. **Step 9a:** Click on the create playlist button to insert the song into a newly created playlist. **Step 10:** Click on the play next song button to play the next song in the album. **Step 11:** Click on the play previous song button to play the previous song in the album. **Step 12:** End.

Basic Domain Model


+1
Web Application
has serverIP and database
+1..*
Advertiser
the advertiser advertising a particular product
+1..*
User
consists of artists and listeners
+1..*
Label/Distributor
takes care of payments related to associated artists
+0..*
+1
+1
+1..*
Listener
uses the application to listen to music released by artists
Artist
the user who releases music to the listeners
+1
can listen to
+0..*
can post
+1
+0..*
ProductInTheApplication
consists of albums, songs, or playlists

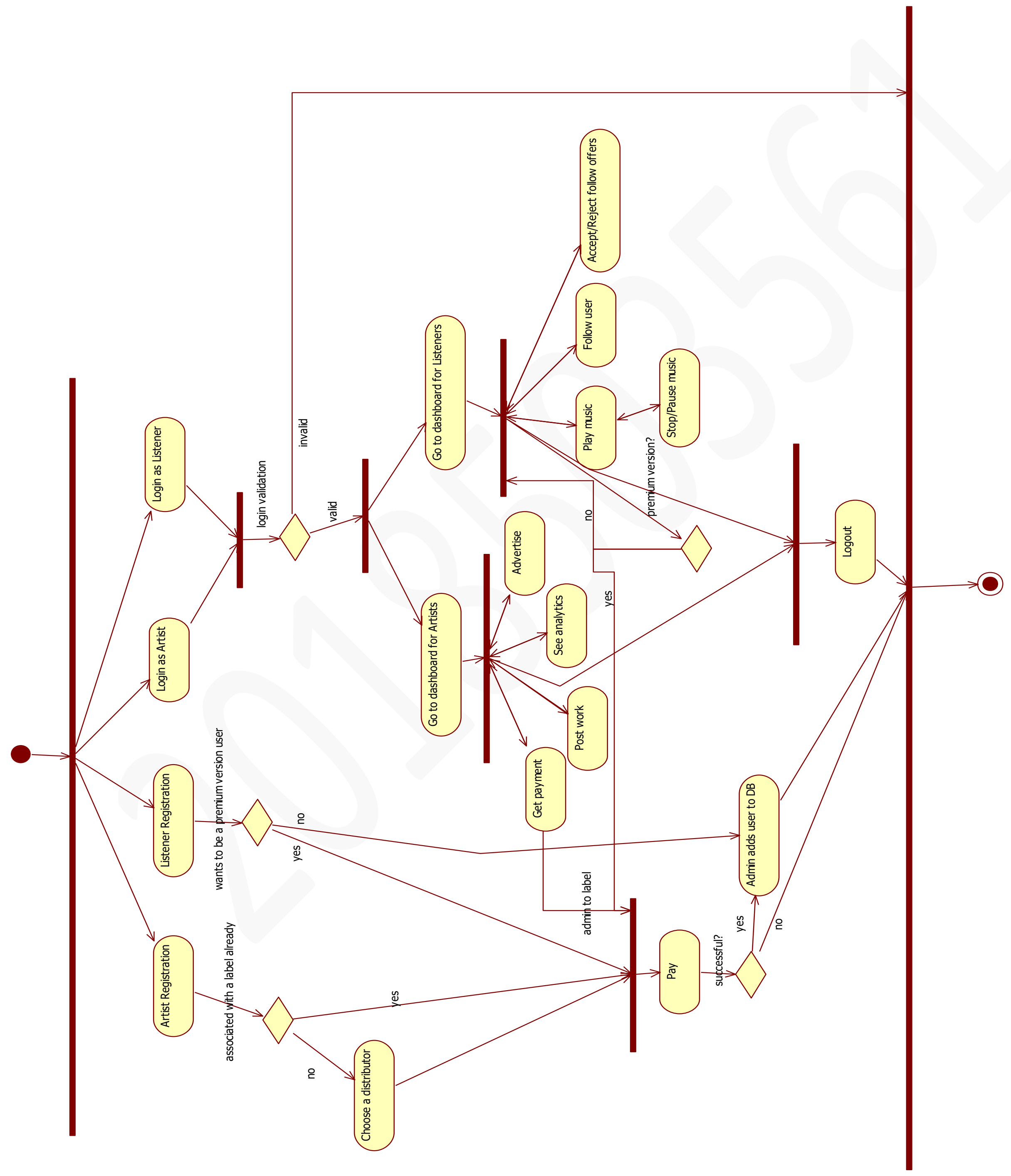
Description:

The above basic domain model consists of classes such as

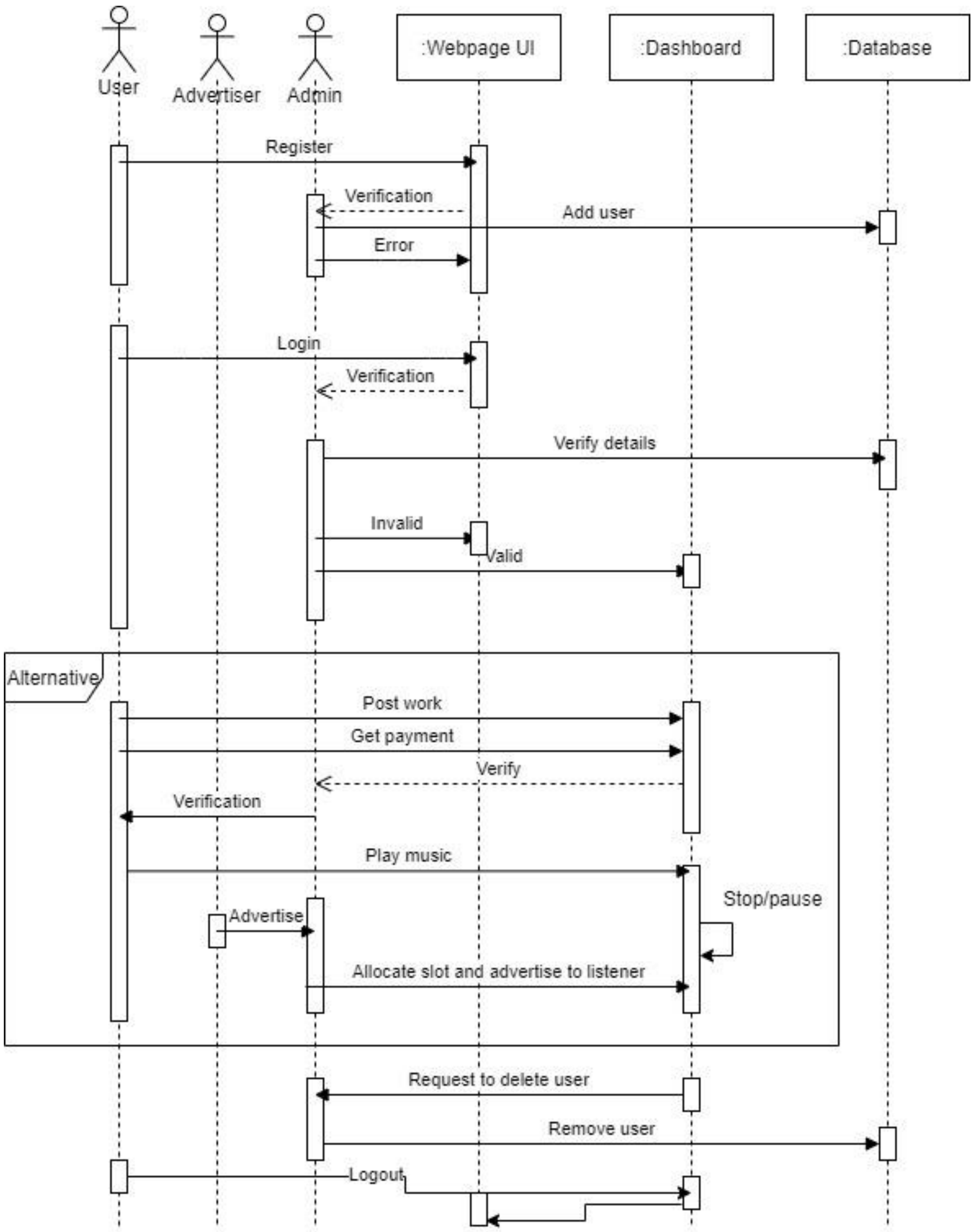
1. Web application : The web application itself with the server IP and the database used.
2. Advertiser : The advertiser advertising a particular product.
3. User : Consists of artists and listeners
    a. Listener : The user who uses the application to listen to music released by artists.
    b. Artist : The user who releases music to the listeners.
4. Label/Distributor : Takes care of payments related to associated artists .
5. ProductInTheApplication : Artist's works such as albums, songs and playlists.

Dynamic Models

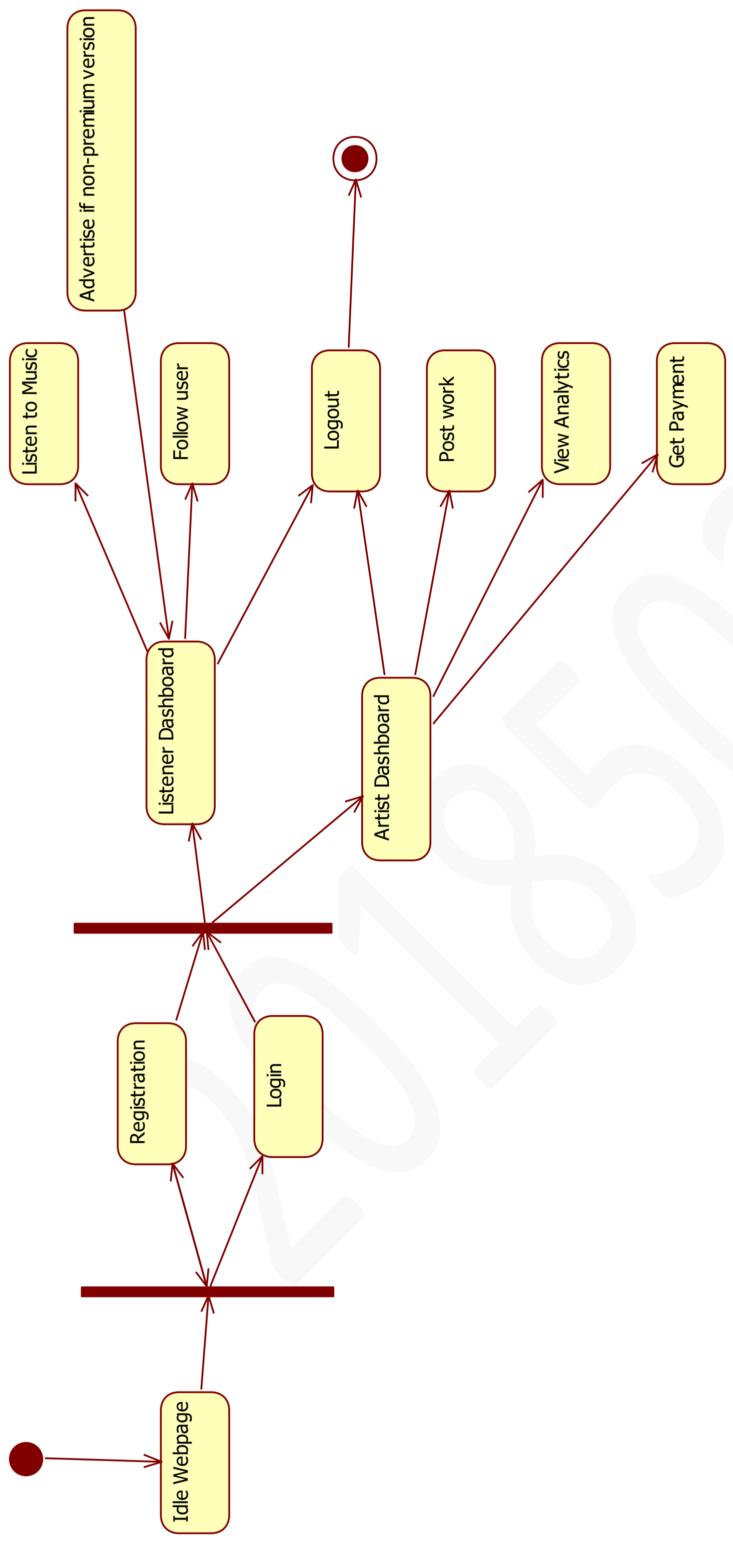
Activity Diagram



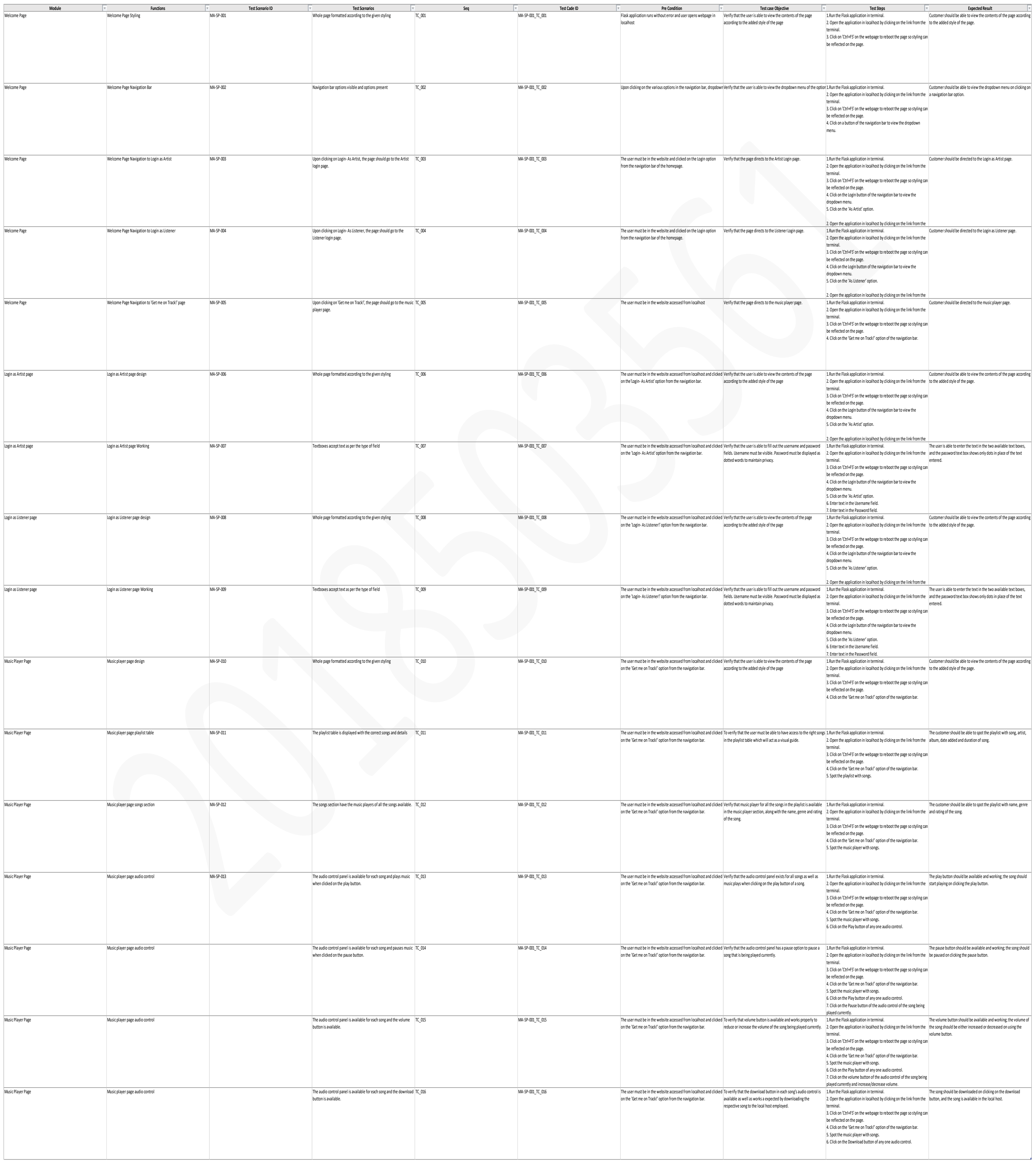
Sequence Diagram



State Diagram



Testcases



Module	Functions	Test Scenario ID	Test Scenarios	Seq	Test Cade ID	Pre Condition	Test case Objective	Test Steps	Expected Result
Welcome Page	Welcome Page Styling	MA-SP-001	Whole page formatted according to the given styling	TC_001	MA-SP-001_TC_001	Flask application runs without error and user opens webpage in localhost	Verify that the user is able to view the contents of the page according to the added style of the page	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page.	Customer should be able to view the contents of the page according to the added style of the page.
Welcome Page	Welcome Page Navigation Bar	MA-SP-002	Navigation bar options visible and options present	TC_002	MA-SP-001_TC_002	Upon clicking on the various options in the navigation bar, dropdown	Verify that the user is able to view the dropdown menu of the option	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on a button of the navigation bar to view the dropdown menu.	Customer should be able to view the dropdown menu on clicking on a navigation bar option.
Welcome Page	Welcome Page Navigation to Login as Artist	MA-SP-003	Upon clicking on Login- As Artist, the page should go to the Artist login page.	TC_003	MA-SP-001_TC_003	The user must be in the website and clicked on the Login option from the navigation bar of the homepage.	Verify that the page directs to the Artist Login page.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the Login button of the navigation bar to view the dropdown menu. 5. Click on the 'As Artist' option. 2. Open the application in localhost by clicking on the link from the	Customer should be directed to the Login as Artist page.
Welcome Page	Welcome Page Navigation to Login as Listener	MA-SP-004	Upon clicking on Login- As Listener, the page should go to the Listener login page.	TC_004	MA-SP-001_TC_004	The user must be in the website and clicked on the Login option from the navigation bar of the homepage.	Verify that the page directs to the Listener Login page.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the Login button of the navigation bar to view the dropdown menu. 5. Click on the 'As Listener' option. 2. Open the application in localhost by clicking on the link from the	Customer should be directed to the Login as Listener page.
Welcome Page	Welcome Page Navigation to 'Get me on Track!' page	MA-SP-005	Upon clicking on 'Get me on Track!', the page should go to the music player page.	TC_005	MA-SP-001_TC_005	The user must be in the website accessed from localhost	Verify that the page directs to the music player page.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the 'Get me on Track!' option of the navigation bar.	Customer should be directed to the music player page.
Login as Artist page	Login as Artist page design	MA-SP-006	Whole page formatted according to the given styling	TC_006	MA-SP-001_TC_006	The user must be in the website accessed from localhost and clicked on the'Login- As Artist' option from the navigation bar.	Verify that the user is able to view the contents of the page according to the added style of the page	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the Login button of the navigation bar to view the dropdown menu. 5. Click on the 'As Artist' option. 2. Open the application in localhost by clicking on the link from the	Customer should be able to view the contents of the page according to the added style of the page.
Login as Artist page	Login as Artist page Working	MA-SP-007	Textboxes accept text as per the type of field	TC_007	MA-SP-001_TC_007	The user must be in the website accessed from localhost and clicked on the 'Login- As Artist' option from the navigation bar.	Verify that the user is able to fill out the username and password fields. Username must be visible. Password must be displayed as dotted words to maintain privacy.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the Login button of the navigation bar to view the dropdown menu. 5. Click on the 'As Artist' option. 6. Enter text in the Username field. 7. Enter text in the Password field.	The user is able to enter the text in the two available text boxes, and the password text box shows only dots in place of the text entered.
Login as Listener page	Login as Listener page design	MA-SP-008	Whole page formatted according to the given styling	TC_008	MA-SP-001_TC_008	The user must be in the website accessed from localhost and clicked on the 'Login- As Listener!' option from the navigation bar.	Verify that the user is able to view the contents of the page according to the added style of the page	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the Login button of the navigation bar to view the dropdown menu. 5. Click on the 'As Listener' option. 2. Open the application in localhost by clicking on the link from the	Customer should be able to view the contents of the page according to the added style of the page.
Login as Listener page	Login as Listener page Working	MA-SP-009	Textboxes accept text as per the type of field	TC_009	MA-SP-001_TC_009	The user must be in the website accessed from localhost and clicked on the 'Login- As Listener!' option from the navigation bar.	Verify that the user is able to fill out the username and password fields. Username must be visible. Password must be displayed as dotted words to maintain privacy.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the Login button of the navigation bar to view the dropdown menu. 5. Click on the 'As Listener' option. 6. Enter text in the Username field. 7. Enter text in the Password field.	The user is able to enter the text in the two available text boxes, and the password text box shows only dots in place of the text entered.
Music Player Page	Music player page design	MA-SP-010	Whole page formatted according to the given styling	TC_010	MA-SP-001_TC_010	The user must be in the website accessed from localhost and clicked on the 'Get me on Track!' option from the navigation bar.	Verify that the user is able to view the contents of the page according to the added style of the page	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the 'Get me on Track!' option of the navigation bar.	Customer should be able to view the contents of the page according to the added style of the page.
Music Player Page	Music player page playlist table	MA-SP-011	The playlist table is displayed with the correct songs and details	TC_011	MA-SP-001_TC_011	The user must be in the website accessed from localhost and clicked on the 'Get me on Track!' option from the navigation bar.	To verify that the user must be able to have access to the right songs in the playlist table which will act as a visual guide.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the 'Get me on Track!' option of the navigation bar. 5. Spot the playlist with songs.	The customer should be able to spot the playlist with song, artist, album, date added and duration of song.
Music Player Page	Music player page songs section	MA-SP-012	The songs section have the music players of all the songs available.	TC_012	MA-SP-001_TC_012	The user must be in the website accessed from localhost and clicked on the 'Get me on Track!' option from the navigation bar.	Verify that music player for all the songs in the playlist is available in the music player section, along with the name, genre and rating of the song.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the 'Get me on Track!' option of the navigation bar. 5. Spot the music player with songs.	The customer should be able to spot the playlist with name, genre and rating of the song.
Music Player Page	Music player page audio control	MA-SP-013	The audio control panel is available for each song and plays music when clicked on the play button.	TC_013	MA-SP-001_TC_013	The user must be in the website accessed from localhost and clicked on the 'Get me on Track!' option from the navigation bar.	Verify that the audio control panel exists for all songs as well as music plays when clicking on the play button of a song.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the 'Get me on Track!' option of the navigation bar. 5. Spot the music player with songs. 6. Click on the Play button of any one audio control.	The play button should be available and working; the song should start playing on clicking the play button.
Music Player Page	Music player page audio control		The audio control panel is available for each song and pauses music when clicked on the pause button.	TC_014	MA-SP-001_TC_014	The user must be in the website accessed from localhost and clicked on the 'Get me on Track!' option from the navigation bar.	Verify that the audio control panel has a pause option to pause a song that is being played currently.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the 'Get me on Track!' option of the navigation bar. 5. Spot the music player with songs. 6. Click on the Play button of any one audio control. 7. Click on the Pause button of the audio control of the song being played currently.	The pause button should be available and working; the song should be paused on clicking the pause button.
Music Player Page	Music player page audio control		The audio control panel is available for each song and the volume button is available.	TC_015	MA-SP-001_TC_015	The user must be in the website accessed from localhost and clicked on the 'Get me on Track!' option from the navigation bar.	To verify that volume button is available and works properly to reduce or increase the volume of the song being played currently.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the 'Get me on Track!' option of the navigation bar. 5. Spot the music player with songs. 6. Click on the Play button of any one audio control. 7. Click on the volume button of the audio control of the song being played currently and increase/decrease volume.	The volume button should be available and working; the volume of the song should be either increased or decreased on using the volume button.
Music Player Page	Music player page audio control		The audio control panel is available for each song and the download button is available.	TC_016	MA-SP-001_TC_016	The user must be in the website accessed from localhost and clicked on the 'Get me on Track!' option from the navigation bar.	To verify that the download button in each song's audio control is available as well as works a expected by downloading the respective song to the local host employed.	1.Run the Flask application in terminal. 2. Open the application in localhost by clicking on the link from the terminal. 3. Click on 'Ctrl+F5' on the webpage to reboot the page so styling can be reflected on the page. 4. Click on the 'Get me on Track!' option of the navigation bar. 5. Spot the music player with songs. 6. Click on the Download button of any one audio control.	The song should be downloaded on clicking on the download button, and the song is available in the local host.

Conclusion

Thus, the document for the OOAD Mini Project for the online music player application has been created and implemented successfully.