CSC 540(001) Database management concepts and Systems

WOLFMEDIA

A MEDIA STREAMING SERVICE

Team - G

- **Arun Srinivasan Parthasarathy, apartha4**
- **Kiron Jayesh, kjayesh**
- **Adittya Soukarjya Saha, asaha4**
- **Arun Kumar Ramesh, arames25**

Assumptions:

1. Users' payments can be done only for the entire month. For instance, if a user registers/subscribes at the end of May, the user will have to pay the subscription for the entire month of May.
2. There can only be one main artist for one song.
3. An artist can belong to only one record label.
4. All songs in the album will be made by the same artist.
5. Royalty payment to songs will be made on the first of every month.
6. The podcast payment is made in lump sum to the podcast hosts. The responsibility of splitting the payment lies among the podcast hosts.
7. There cannot be two songs released by the same artist on the same date.
8. No two albums can have the same name if they have the same artist.
9. Podcasts cannot have the same name if they are hosted by the same set of hosts.

1. Problem Statement:

The main idea is to build a database for a media streaming service similar to that of Spotify and Apple Music. This database would be used by the management and the administration present in the company. Apart from keeping track of the basic information for Songs, Artists, Labels, Podcasts, etc., there is also a need to keep track of the payment information for podcast hosts, labels, users, and artists. There are multiple tasks and operations that need to be performed: information processing, maintaining the information for Podcasts, Songs, Artists, Podcasts, etc. Each of these tasks would be performed using API calls from a programming language (Python).

The main reason for using a database over a text file is that we need to maintain reports of each payment that is made to the podcast, labels, and artists. If we just needed to keep the information stored for logs, then it would be more suitable to use a text file. Since the media streaming service finance team would require us to analyze the trends that are present in the database we need an efficient way to store the data. Moreover, the use of text files would make understanding relations between such huge chunks of data very complex. A simple text file on the hand would not have any constraints on the data, and pretty much any kind of data could easily be appended to them.

2. Intended Users:

- **WolfMedia Song Team:** Manages each song, record label, artist and album present in the database.
- **WolfMedia Podcast Team:** Manages each podcast, its hosts and episodes that are present in the database.
- **WolfMedia Finance Team:** Analyzes the amount of money coming in from the users via the monthly subscriptions. Also, analyzes the amount of money which is going out to be paid to the Podcast Hosts, Songs and Artists.
- **WolfMedia Administrator**: Oversees all the information that is present in the database. Has complete access to create, update, delete any entity in the database.

3. Five Main Entities:

- **Song:** Title, Duration, Play Count, Release Date, Language, Royalty Paid, Released Country, Royalty Rate, Genres
- **Artist:** ArtistID**,** Name, Type, Country, Primary Genre, Monthly Listeners, Status
- **Podcast:** Name, Country, Episode Count, Genres, Language, Rating, Total Subscribers, Sponsors
- **Finance:** PaymentID which is the attribute in the Parent Entity of Song Payment, User Payment, Podcast Payment
- **Podcast Hosts:** HostID, FirstName, LastName, Email, Phone, City

4. Real Life Situations where this Database would be used:

Situation 1: An artist under a record label would like to register a new song to the media streaming service. He would contact the management of the streaming service and ask them to add this particular song to the database so subscribers can listen to it. Furthermore, he would like to add this song to a new album as the title track (For example: Track Number 1).

Situation 2: Continuing from the previous scenario**,** a month would have passed after this song had been added to the database. The finance team would have to use the money earned from the users' subscriptions to pay the Record Label of the Artist and the Artist himself.

5. Application Programming Interfaces:

Information Processing:

- createSong(songTitle, mainArtistID, duration, genres, releaseDate, releaseCountry, language, royaltyRate, collabIDs, royaltyPaidStatus)
  return confirmation if song is created

- updateSong(songTitle, duration, genre, releaseDate, releaseCountry, language, royaltyRate, royaltyPaidStatus, mainArtistID)
  return confirmation if song is updated
  For values not being updated, the same value would be sent in the param list.

- deleteSong(songTitle, mainArtistID)
  return confirmation if song is deleted

- createArtist(artistID, artistName, status, type, country, primaryGenre)
  return confirmation if artist is created

- updateArtist(artistId, artistName, status, type, country, primaryGenre, monthly listener)
  return confirmation if artist is updated
  For values not being updated, the same value would be sent in the param list.

- deleteArtist(artistID)
  return confirmation if artist is deleted

- createPodcast(hostIDs, podcastName, language, country, episodeCount, genres, rating, sponsor, totalSubscribers)
  return confirmation if podcast is created

- createPodcastHost(hostID, firstName, lastName, phone, email, city)
  return confirmation if podcast host is created

- createPodcastEpisode(episodeNumber, podcastName, hostID, episodeTitle, duration, releaseDate, listeningCount, advCount)
  return confirmation if podcast episode is created

- assignHostToPodcast(hostID, podcastName)
  assigns a host to a podcast

- updateHost(hostID, firstName, lastName, phone, email, city)
  return confirmation if host is updated
  For values not being updated, the same value would be sent in the param list.

- updatePodcast(hostID, podcastName, language, country, episodeCount, genre, rating, sponsor, totalSubscriber)
  return confirmation if podcast is updated
  For values not being updated, the same value would be sent in the param list.

- updatePodcastEpisode(episodeNumber, podcastName, hostID, episodeTitle, duration, releaseDate, listeningCount, advCount)
  return confirmation if podcast episode is updated
  For values not being updated, the same value would be sent in the param list

- deletePodcast(podcastName, hostID)
  returns confirmation if podcast is deleted

- deleteHost(hostID)
  return confirmation if host is deleted

- deleteEpisode(episodeNumber, podcastName, hostID)
  return confirmation if episode is deleted

- createAlbum(songTitle -> List of songs, songReleaseDate -> List of release dates, mainArtistID albumName, releaseYear, Edition)
  Here we need to put a set of songs and release dates along with the main Artist ID as a set of albums get added to an album
  returns confirmation if album is created

- assignSongToAlbum(songTitle, songReleaseDate, albumName, mainArtistID)
  assigns a song to an Album

- createRecordLabel(labelID, labelName)
  returns confirmation if record label is created

- assignArtistToLabel(labelID, ArtistID)
  assigns artist to label

- deleteLabel(labelID)
  returns confirmation if label is deleted

- createUser(userID, firstName, lastName, phone, email, registrationDate, subscriptionStatus, monthlyFee)
  returns confirmation if user is created

- deleteUser(userID)
  returns confirmation if user is deleted

- createSpecialGuest(specialGuestID, guestName, episodeID, podcastName, hostID)
  return confirmation if special guest is created

- updateSpecialGuest(specialGuestID, guestName, episodeID, podcastName, hostID)
  return confirmation if special guest is updated
  For values not being updated, the same value would be sent in the param list

- deleteSpecialGuest(specialGuestID)

- updateUser(userID, firstName, lastName, phone, email, registrationDate, subscriptionStatus, monthlyFee)
  return confirmation if user is updated
  For values not being updated, the same value would be sent in the param list

Maintaining Record:

- findSongsGivenAlbum(albumName)
  returns all songs in an Album from different artist who have the same album name

- findSongsGivenArtist(artistID)
  returns all songs given an artist

- findPodcastEpisodes(podcastName)
  returns all episodes from a podcast who have that particular name(different hosts

might be present)

- updateTotalSubscribers(podcastName, hostID, subscribers)
  return confirmation if total subscribers is updated
  For values not being updated, the same value would be sent in the param list

- updateRatings(podcastName, hostID, rating)
  return confirmation if rating is updated
  For values not being updated, the same value would be sent in the param list

- updateMonthlyListeners(artistID, monthlyListener)
  return confirmation if monthly listener is updated
  For values not being updated, the same value would be sent in the param list

- updateListeningCount(episodeNumber, podcastName, hostID, listeningCount)
  return confirmation if listening count is updated
  For values not being updated, the same value would be sent in the param list

Maintaining Payments:

- makeSongPayment(startDate, endDate, songName, mainArtistID, releaseDateSong, playCount, paymentID)
  makes a payment for a song to artist and label for a particular period using the royalty rate

- makePodcastPayment(podcastName, hostID, paymentFee, bonusFee, paymentID)
  makes a payment for a podcast depending on a flat fee

- userPayments(paymentCycleDate, paymentID)
  users pay the finance team

Reports:

- monthlyPlayCountGivenSong(songName, mainArtistID)
  Reports monthly play count for a given song

- monthlyPlayCountGivenAlbum(albumName)
  Reports monthly play count for a given album’s songs for all the albums with the same name

- monthlyPlayCountGivenArtist(artistID)
  Reports monthly play count for a given artist's songs

- paymentsToHost(startDate, EndDate, hostID)
  Reports the payments made to a host for a given time frame

- paymentsToArtist(startDate, EndDate, artistID)
  Reports the payments made to an artist for a given time frame

- paymentsToLabel(startDate, EndDate, labelID)
  Reports the payments made to a label for a given time frame

- revenueEarnedPerMonth()
  Reports the revenue earned per month for the media streaming service

- revenueEarnedPerYear()
  Reports the revenue earned per year for the media streaming service

- reportSongGivenArtist(artistID)
  Reports all the songs given an artist

- reportSongGivenAlbum(albumName)
  Reports all the songs given an album(songs from albums with the same name might also be given)

- reportEpisodeGivenPodcast(podcastName)
  Reports all the podcast episodes given a podcast(episodes from podcasts with the same name might also be given)

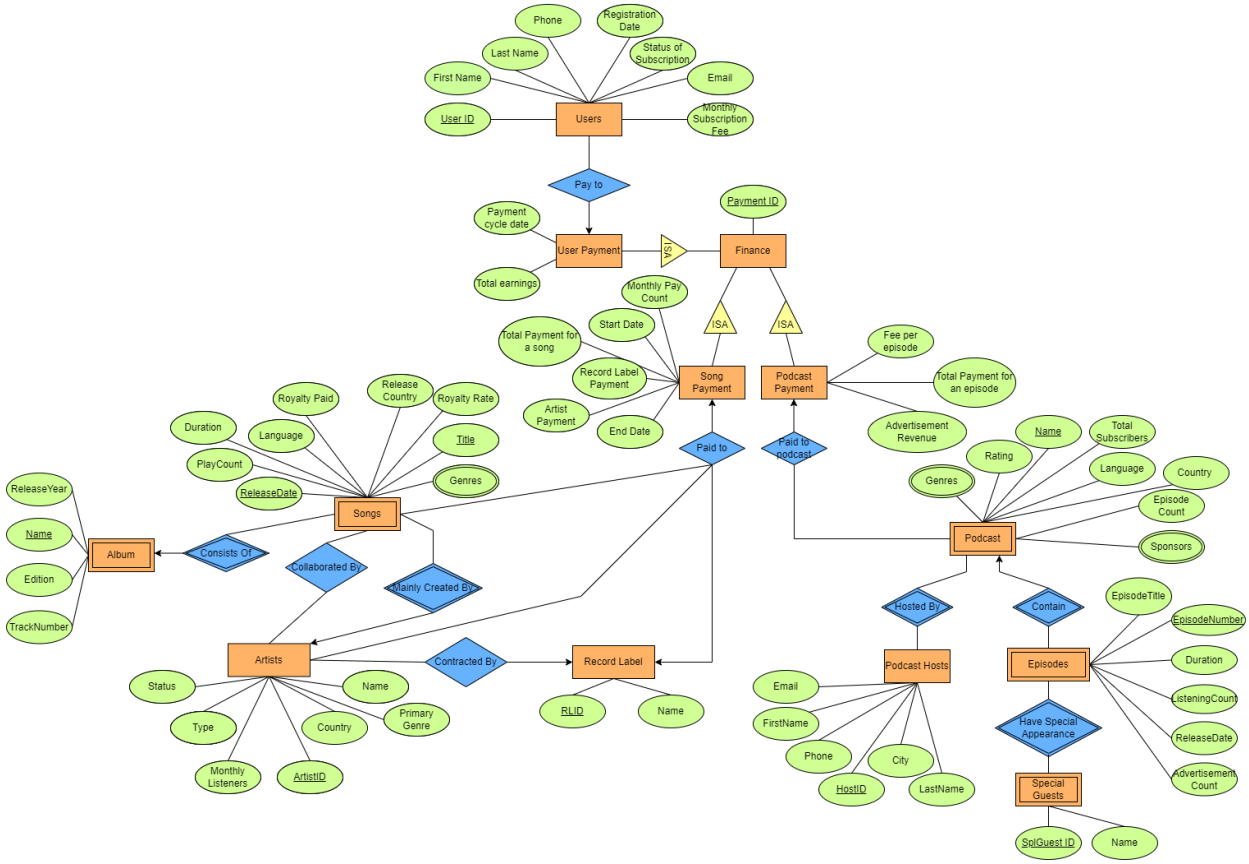
6. Description of Views:

- **WolfMedia Song Team View:** Has access to the songs and their metadata. They can view the songs, their album, their artists, and the artists' record labels. This will enable the team to effectively manage the songs and their related data. The team does not need access to Podcasts and Finance related information.
- **WolfMedia Podcast Team View:** Has access to podcasts, hosts, episodes, and special guests. This will allow the team to manage the podcasts and related data effectively. They do not have access to the songs and finance-related information.

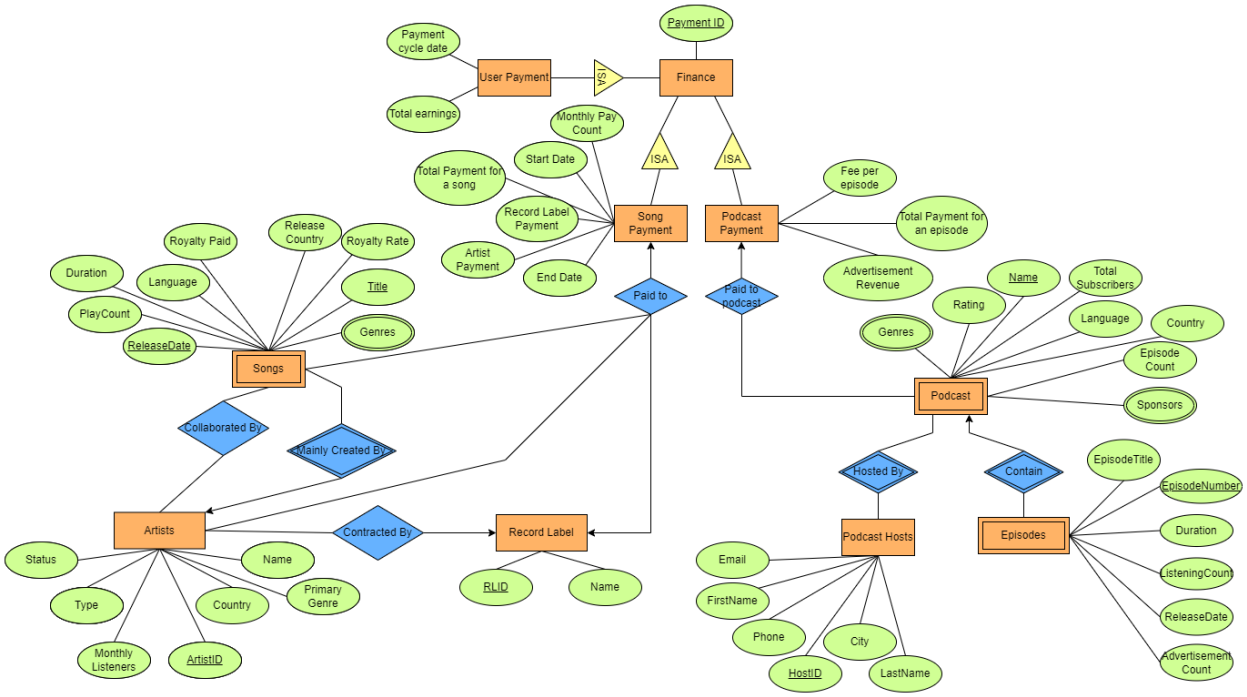
- **<u>WolfMedia Finance Team View</u>:** They can access the billing and financial information. They can access and manage User Payments. The royalties and the fees paid related to songs and podcasts, respectively.
- **<u>WolfMedia Administrator Video</u>**: Has complete access to all data in the system. The administrator can view the details of all songs, artists, and record labels, and information related to podcasts, and all the payment information.

7. Local ER Diagrams:

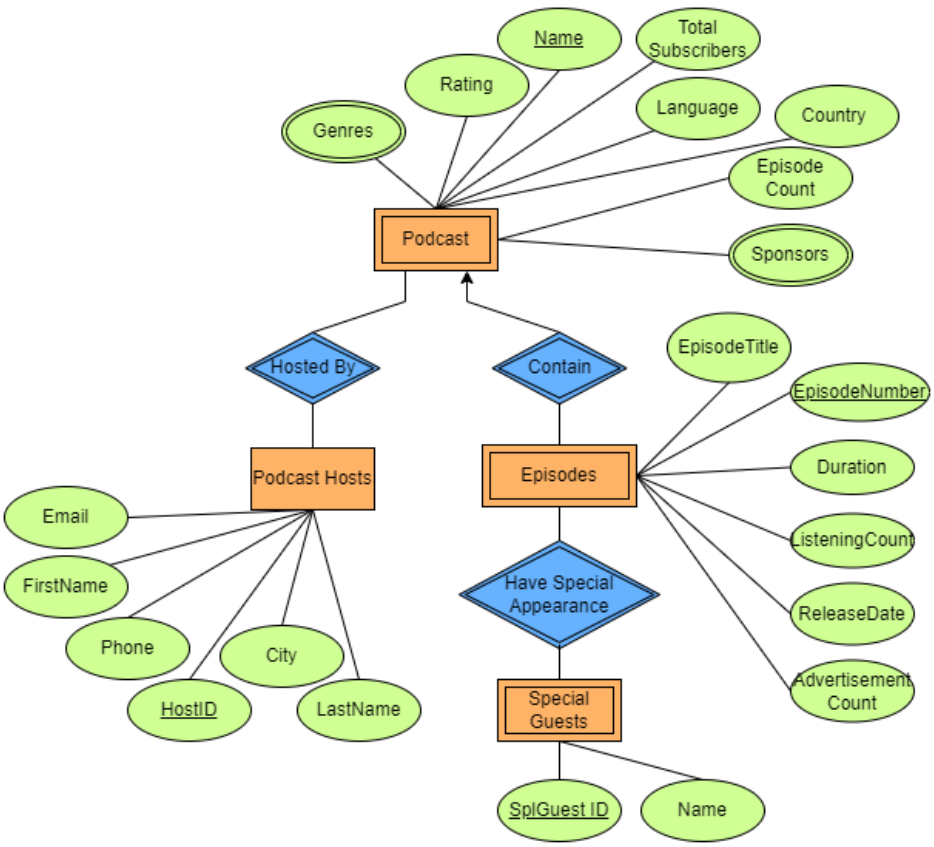
<u>Administrator View:</u>



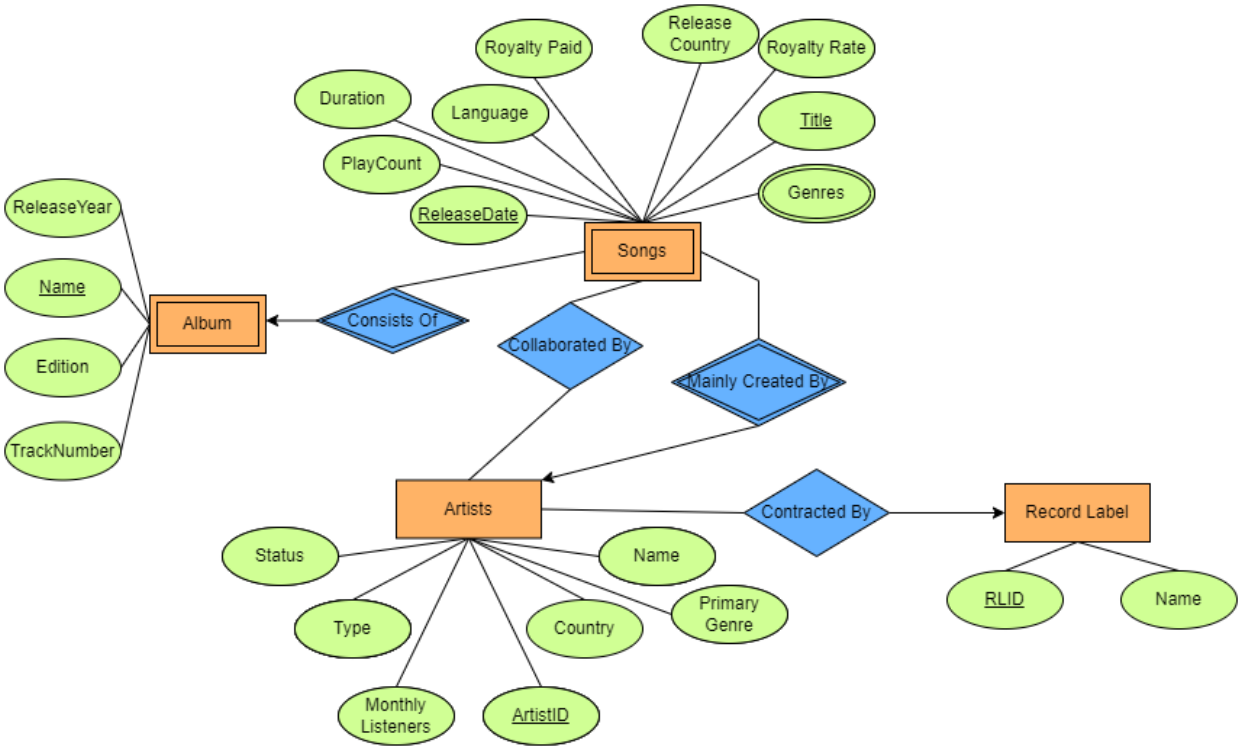
Finance Team View:



Podcast Team View:



Song Team View:



8. Description of Local ER Diagrams:

- The WolfMedia System consists of four teams which include Podcast, Finance, Administrator, and Song teams. Each team has its own functionalities and data access.
- Users use the WolfMedia Streaming service to listen to songs and podcasts. They are uniquely identified by User ID. They also pay monthly payments to use the streaming service.
- The user's payment is recorded in the User Payment entity. User Payment inherits its primary key 'Payment ID' from 'Finance'.
- Each user can pay only 'one' payment in a month.
- A song in the database can only be identified by its Title, release date, and its main artist.
- Each album contains one or more songs, and each song is assigned a track number. The albums are identified by its name and the artist who created it. It can also be identified when given a particular song name, its release date and its main artist. Also, a song can only be attributed to one album.
- Album is made as a weak entity since it cannot exist independently without a song. An album is created only if a song is assigned to it.

- A song can be made by multiple artists but every song has a main artist through which it is identified along with its title. The artists are identified by Artist ID.
- An artist can create many songs, whereas a song can be made by only one main artist even though there can be multiple collaborators.
- The Record Label is identified by its Record Label ID. An artist can be a part of only one record label at a time. In contrast, a record label can have many artists in its contract.
- Song Payment is where the payment process to artists and record labels happen and it has Payment ID as its unique identifier which is inherited from Finance.
- The 'Total Payment for a song' is calculated based on the royalty rate of the song and its monthly play count.
- A podcast is identified by its name and the podcast hosts' ID.
- The Podcast is hosted by one or more hosts. The Podcast Hosts are uniquely identified by a HostId. A particular host can host one or more podcasts.
- Each podcast has one or more episodes. The episodes are identified by the episode number, given the name and host IDs of the podcast.
- An episode can have special guests and each of them are identified by Special Guest ID.
- A particular special guest can feature in one or more podcast episodes. Similarly, an episode can contain zero or more special guests.
- Podcast Payment is where the payment process to podcast happens and it has Payment ID as its unique identifier which is inherited from Finance.
- The Advertisement Count and the Flat fee for each episode determine the Total payment for an episode.
- Each payment: user, song, or podcast records a specific transaction in the database.

9. Local Relational Schemas:

Administrator View:

Users(UserID, firstName, lastName, phone, registrationDate, statusOfSubscription, email, monthlySubscriptionFee)
PayTo(UserID, PaymentID)
UserPayment(PaymentID, PaymentCycleDate, TotalEarnings)
Finance(PaymentID)
SongPayment(PaymentID, MonthlyPayCount, StartDate, TotalPaymentForSong, RecordLabelPayment, ArtistPayment, EndDate)
PaidTo(Title, ReleaseDate, ArtistID, RLID)

Songs(Title, ReleaseDate, ArtistID, Genres, RoyaltyRate, RoyaltyPaid, ReleaseCountry, Language, Duration, PlayCount)
Artists(ArtistID, Name, PrimaryGenre, Country, Type, Status, MonthlyListeners)
Album(Name, Title, ReleaseDate, ArtistID, ReleaseYear, Edition, TrackNumber)
RecordLabel(RLID, Name)
CollaboratedBy(ArtistID, Title, ReleaseDate)
ContractedBy(ArtistID, RLID)

PodcastPayment(PaymentID, FeePerEpisode, TotalPaymentForEpisode, AdvertisementRevenue)
PaidToPodcast(PaymentID, Name)
Podcast(PaymentID, Name, Genres, Rating, TotalSubscribers, Language, Country, EpisodeCount, Sponsors)
PodcastHosts(PaymentID, PodcastName, HostID, FirstName, LastName, Phone, Email, City)
Episodes(PaymentID, EpisodeNumber, PodcastName, EpisodeTitle, Duration, ListeningCount, ReleaseDate, AdvertisementCount)
SpecialGuests(PaymentID, SplGuestID,EpisodeNumber, PodcastName, Name)

Song Team View:
Songs(Title, ReleaseDate, ArtistID, Genres, RoyaltyRate, RoyaltyPaid, ReleaseCountry, Language, Duration, PlayCount)
Artists(ArtistID, Name, PrimaryGenre, Country, Type, Status, MonthlyListeners)
RecordLabel(RLID, Name)
Album(Name, Title, ReleaseDate, ArtistID, ReleaseYear, Edition, TrackNumber)
ContractedBy(ArtistID, RLID)
CollaboratedBy(ArtistID, Title, ReleaseDate)

Podcast Team View:
Podcast(PaymentID, Name, Genres, Rating, TotalSubscribers, Language, Country, EpisodeCount, Sponsors)
PodcastHosts(PaymentID, PodcastName, HostID, FirstName, LastName, Phone, Email, City)
Episodes(PaymentID, EpisodeNumber, PodcastName, EpisodeTitle, Duration, ListeningCount, ReleaseDate, AdvertisementCount)
SpecialGuests(PaymentID, SplGuestID,EpisodeNumber, PodcastName, Name)

Finance Team View:
UserPayment(PaymentID, PaymentCycleDate, TotalEarnings)
Finance(PaymentID)
SongPayment(PaymentID, MonthlyPayCount, StartDate, TotalPaymentForSong, RecordLabelPayment, ArtistPayment, EndDate)
PaidTo(Title, ReleaseDate, ArtistID, RLID)

Songs(Title, ReleaseDate, ArtistID, Genres, RoyaltyRate, RoyaltyPaid, ReleasedCountry, Language, Duration, PlayCount)
Artists(ArtistID, Name, PrimaryGenre, Country, Type, Status, MonthlyListeners)
RecordLabel(RLID, Name)
CollaboratedBy(ArtistID, Title, ReleaseDate)
ContractedBy(ArtistID, RLID)

PodcastPayment(PaymentID, FeePerEpisode, TotalPaymentForEpisode, AdvertisementRevenue)
PaidToPodcast(PaymentID, Name)
Podcast(PaymentID, Name, Genres, Rating, TotalSubscribers, Language, Country, EpisodeCount, Sponsors)
PodcastHosts(PaymentID, PodcastName, HostID, FirstName, LastName, Phone, Email, City)
Episodes(PaymentID, EpisodeNumber, PodcastName, EpisodeTitle, Duration, ListeningCount, ReleaseDate, AdvertisementCount)

10. Local Schema Documentation:

Entity Sets to Relations:

1. The entities that were designed earlier were converted to relations. The strong entities like 'Users', 'Artists', 'Record Label', 'Podcast Hosts', and 'Finance' were directly made into relations having the same attributes as their entities.
2. There are multiple weak entities which include 'Songs', 'Album', 'Podcast', 'Episodes', and 'Special Guests'. The weak entities are decomposed into relations that contain a partial primary key(s) and a primary key derived from the strong entity to which it is connected with a weak relation (double diamond). For instance, In the 'Songs' relation, 'Artist ID' is derived from Artists, while 'Title' and 'Release Date' serve as partial primary keys. In order to access the data of a weak entity relation, it is necessary to access the primary key(s) of the connecting strong entity(ies). For instance, to access a particular data in the 'Episodes' relation, we need to specify the 'EpisodeNumber', 'Name' from 'Podcast' and 'HostID' from

'Hosts'. Similarly, to access a specific data from 'Songs' relation, we need to specify, 'Title' and 'ReleaseDate' and also "ArtistID" from 'Artists'.

3. Certain entities are inherited from a parent entity to reduce redundancy and achieve effective relations. The children entity will contain attributes of the parent entity along with unique attributes of its own. For example, User Payment is inherited from Finance, and its attributes are 'PaymentID' (primary key, derived from Finance), 'Payment Cycle Date', and 'Total Earnings'.

Relationships to Relations:

All the relationships are converted into a relation. Thus, we have ‘Consists Of’, ‘Collaborated By’, ‘Mainly Created By’, ‘Contracted By’, ‘Paid to’, ‘Paid to podcast’, ‘Pay to’, ‘Hosted By’, ‘Contain’, ‘Have Special Appearance’ present as relations in our schema. Weak relationships are not converted into a relation. Also, it is worth mentioning that, although the relation ‘Collaborated By’ links a weak entity ‘Songs’ and ‘Artists’, it is not considered a weak relationship since ‘Songs’ are identified from the main artists which are represented by the relation ‘Mainly Created By’. All the appropriate attributes necessary to represent the particular relation are derived from the connecting entities. For instance, the ‘Contracted By’, which connects ‘Artists’ and ‘Record Label’ contains ArtistID and RLID as attributes in its relation.