WolfMedia

A Media Streaming Service

CSC 540 Database Management Systems

Project Report 3

Team W

Naveen Jayanna - njayann,
Samarth Purushothaman - spurush,
Srilekha Gudipati - sngudipa,
Sai Krishna Teja Varma Manthena - smanthe

February 13, 2023

Revisions to Previous Reports

Report 1 Changes:

1. mainArtist attribute is added to Song in report 1 - highlighted in yellow.

   OLD
   Payment Staff view
   f. Song (albumID, songTitle, playCount, royaltyRate, royaltyPaid)

   Artist's view

   h. Song (albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyPaid, trackNumber)

   Record Label's view

   h. Song (albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyPaid, trackNumber)

   User view

   h. Song (albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, trackNumber)

   NEW
   Payment Staff view

   f. Song (albumID, songTitle, playCount, royaltyRate, royaltyPaid, mainArtitst)

   Artist's view

   h. Song (albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyPaid, trackNumber, mainArtitst)

   Record Label's view

   h. Song (albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyPaid, trackNumber, mainArtitst)

   User view

   h. Song (albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, trackNumber, mainArtitst)

2. Removing extra APIs in Information Processing and Payments: highlighted with yellow and strikethrough.

OLD

Information Processing

- ~~newAccount(accountID, password, typeOfAcct, registrationDate)~~
  ~~Returns Confirmation~~

- ~~updateAccount(accountID, password, typeOfAcct, registrationDate)~~
  ~~Returns Confirmation~~

- ~~deleteAccount(accountID)~~
  ~~Returns Confirmation~~

- ~~newUser(userAccountID, firstName, lastName, phoneNum, email, statusOfSubscription)~~
  ~~Returns Confirmation~~

- ~~updateUser(userAccountID, firstName, lastName, phoneNum, email, statusOfSubscription)~~
  ~~Returns Confirmation~~

- ~~deleteUser(userAccountID)~~
  ~~Returns Confirmation~~

- newArtist(artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabel)
  Returns Confirmation

- updateArtist(artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabel)
  Returns Confirmation

- deleteArtist(artistAccountID)
  Returns Confirmation

- ~~subscribeUserToArtist(userAccountID, artistAccountID)~~
  ~~Returns Confirmation~~

- ~~unsubscribeUserFromArtist(userAccountID, artistAccountID)~~
  ~~Returns Confirmation~~

- newAlbum(albumName, releaseYear, edition)
  Returns albumID

- updateAlbum(albumID, albumName, releaseYear, edition)
  Returns Confirmation

- deleteAlbum(albumID)
  Returns Confirmation

- newSong(albumID, songTitle, mainArtist, duration, genres, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyStatus, collaborators, trackNumber)
  Returns Confirmation

- updateSong(albumID, songTitle, mainArtist, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyStatus, trackNumber)
  Returns Confirmation

- deleteSong(albumID, songTitle)
  Returns Confirmation

- newPodcastHost(podcastHostAccountID, firstName, lastName, phoneNum, email, city)
  Returns Confirmation

- updatePodcastHost(podcastHostAccountID, firstName, lastName, phoneNum, email, city)
  Returns Confirmation

- deletePodcastHost(podcastHostAccountID)
  Returns Confirmation

- newPodcast(podcastHost, podcastName, language, country, episodeCount, genres, rating, sponsors totalSubscribers)
  Returns podcastID

- updatePodcast(podcastID, podcastHost, podcastName, language, country, episodeCount, rating, totalSubscribers)
  Returns Confirmation

- deletePodcast(podcastID)
  Returns Confirmation

- ~~subscribeUserToPodcast(userAccountID, podcastID)~~
  ~~Returns Confirmation~~

- ~~unsubscribeUserFromPodcast(userAccountID, podcastID)~~
  ~~Returns Confirmation~~

- newPodcastEpisode(podcastID, episodeTitle, duration, releaseDate, listeningCount, specialGuests, advertisementCount)

Returns Confirmation

- updatePodcastEpisode(podcastID, episodeTitle, duration, releaseDate, listeningCount, advertisementCount)
  Returns Confirmation

- deletePodcastEpisode(podcastID, episodeTitle)
  Returns Confirmation

- ~~newSponsor(sponsorName)~~
  ~~Return SponsorID~~

- ~~updateSponsor(sponsorName, SponsorID)~~
  ~~Returns Confirmation~~

- ~~deleteSponsor(SponsorID)~~
  ~~Returns Confirmation~~
- ~~newSpecialGuest(guestName)~~
  ~~Return GuestID~~

- ~~updateSpecialGuest(guestName, GuestID)~~
  ~~Returns Confirmation~~

- ~~deleteSpecialGuest(GuestID)~~
  ~~Returns Confirmation~~

- ~~newGenre(genre)~~
  ~~Returns Confirmation~~

- ~~deleteGenre(genre)~~
  ~~Returns Confirmation~~

- ~~addSponsorToPodcast(podcastID, SponsorName, amount)~~
  ~~Returns Confirmation~~

- ~~updateSponsorOfPodcast(podcastID, SponsorName, amount)~~
  ~~Returns Confirmation~~

- ~~removeSponsorFromPodcast(podcastID, SponsorName)~~
  ~~Returns Confirmation~~

- addSongToAlbum(albumID, songTitle, mainArtist, duration, genres, albumID)
  Returns Confirmation

- ~~updateSongOfAlbum(albumID, songTitle, mainArtist, duration, genres, albumID)~~
  ~~Returns Confirmation~~

- addArtistToAlbum(artistAccountID, albumID)
  Returns Confirmation

- updateArtistOfAlbum(artistAccountID, albumID)
  Returns Confirmation

- removeArtistOfAlbum(artistAccountID, albumID)
  Returns Confirmation

- addArtistToRecordLabel(artistAcocuntID, recordLabelAccountID)
  Returns Confirmation

- ~~updateRecordLabelOfArtist(artistAcocuntID, recordLabelAccountID)~~
  ~~Returns Confirmation~~

- addHostToPodcast(podcastHostAccountID, podcastID)
  Returns Confirmation

- ~~updateHostOfPodcast(podcastHostAccountID, podcastID)~~
  ~~Returns Confirmation~~

- ~~removeHostOfPodcast(podcastHostAccountID, podcastID)~~
  ~~Returns Confirmation~~

- addEpisodeToPodcast(podcastID, episodeTitle, duration, releaseDate, listeningCount, specialGuests, advertisementCount)
  Returns Confirmation

- ~~updatePodcastOfEpisode(podcastID, episodeTitle)~~
  ~~Returns Confirmation~~

- ~~addGenresToSong(songID, AlbumID, genres)~~
  ~~Returns Confirmation~~

- ~~updateGenresOfSong(songID, AlbumID, genres)~~
  ~~Returns Confirmation~~

- ~~removeGenresOfSong(songID, AlbumID, genres)~~
  ~~Returns Confirmation~~

- ~~updateGenresofPodcast(podcastID, genres)~~
  ~~Returns Confirmation~~

- ~~addGenresToPodcast(podcastID, genres)~~
  ~~Returns Confirmation~~

- ~~removeGenresToPodcast(podcastID, genres)~~
  ~~Returns Confirmation~~

- ~~addCollaboratorsToSong(songID, AlbumID, collaborators)~~
  ~~Returns Confirmation~~

- ~~updateCollaboratorsOfSong(songID, AlbumID, collaborators)~~
  ~~Returns Confirmation~~

- ~~RemoveCollaboratorsOfSong(songID, AlbumID, collaborators)~~
  ~~Returns Confirmation~~

- ~~ratePodcast(userAccountID, podcastID, rating)~~
  ~~Returns Confirmation~~

- ~~addSpecialGuestsToPodcastEpisode(podcastID, episodeTitle, guestIDs)~~
  ~~Returns Confirmation~~

- ~~updateSpecialGuestsOfPodcastEpisode(podcastID, episodeTitle, guestIDs)~~
  ~~Returns Confirmation~~

- ~~removeSpecialGuestsOfPodcastEpisode(podcastID, episodeTitle, guestID)~~
  ~~Returns Confirmation~~

Maintaining payments

- makeRoyaltyPaymentToRecordLabelForSong(albumId, songTitle, month, year)
  Returns confirmation

- makePaymentToArtistsFromRecordLabel(recordLabelAccountID, albumId, songTitle, month, year)
  Returns confirmation

- makePaymentToPodcastHost(podcastHostAccountID, month, year)
  Returns confirmation

- receivePaymentFromSubscriber(userAccountID, month, year, amt)
  Returns confirmation

- ~~getEarningsOfArtist(artistAccountID, month, year)~~
  ~~Returns earnings of the artist for the given month and year~~

- ~~getEarningsOfPodcastHost(podcastHostAccountID, month, year)~~
  ~~Returns earnings of the podcast host for the given month and year~~

- ~~getEarningsOfRecordLabel(recordLabelAccountID, month, year)~~
  ~~Returns earnings of the record label for the given month and year~~

- ~~getAnnualSubcriptionFeesOfUser(userAccountID, year))~~
  ~~Returns total fees paid by the user in the given year~~

- ~~UpdatePaymentRecordOfUser(userAccountID, month, year, amount)~~
  ~~Returns confirmation~~

- ~~UpdatePodcastHostPaymentRecord(podcastHostAccountID, month, year, amount)~~
  ~~Returns confirmation~~

- ~~UpdateRecordLabelPaymentRecord(recordLabelAccountID, month, year, playCount, amount)~~
  ~~Returns confirmation~~

- ~~UpdateArtistPaymentRecord(artistAccountID, month, year, amount)~~
  ~~Returns confirmation~~

NEW

Information Processing

- newArtist(artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabel)
  Returns Confirmation

- updateArtist(artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabel)
  Returns Confirmation

- deleteArtist(artistAccountID)
  Returns Confirmation

- newAlbum(albumName, releaseYear, edition)
  Returns albumID

- updateAlbum(albumID, albumName, releaseYear, edition)
  Returns Confirmation

- deleteAlbum(albumID)
  Returns Confirmation

- newSong(albumID, songTitle, mainArtist, duration, genres, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyStatus,

collaborators, trackNumber)
Returns Confirmation

- updateSong(albumID, songTitle, mainArtist, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyStatus, trackNumber)
  Returns Confirmation

- deleteSong(albumID, songTitle)
  Returns Confirmation

- newPodcastHost(podcastHostAccountID, firstName, lastName, phoneNum, email, city)
  Returns Confirmation

- updatePodcastHost(podcastHostAccountID, firstName, lastName, phoneNum, email, city)
  Returns Confirmation

- deletePodcastHost(podcastHostAccountID)
  Returns Confirmation

- newPodcast(podcastHost, podcastName, language, country, episodeCount, genres, rating, sponsors totalSubscribers)
  Returns podcastID

- updatePodcast(podcastID, podcastHost, podcastName, language, country, episodeCount, rating, totalSubscribers)
  Returns Confirmation

- deletePodcast(podcastID)
  Returns Confirmation

- newPodcastEpisode(podcastID, episodeTitle, duration, releaseDate, listeningCount, specialGuests, advertisementCount)
  Returns Confirmation

- updatePodcastEpisode(podcastID, episodeTitle, duration, releaseDate, listeningCount, advertisementCount)
  Returns Confirmation

- deletePodcastEpisode(podcastID, episodeTitle)
  Returns Confirmation

- addSongToAlbum(albumID, songTitle, mainArtist, duration, genres, albumID)
  Returns Confirmation

- addArtistToAlbum(artistAccountID, albumID)
  Returns Confirmation

- updateArtistOfAlbum(artistAccountID, albumID)
  Returns Confirmation

- removeArtistOfAlbum(artistAccountID, albumID)
  Returns Confirmation

- addArtistToRecordLabel(artistAcocuntID, recordLabelAccountID)
  Returns Confirmation

- addHostToPodcast(podcastHostAccountID, podcastID)
  Returns Confirmation

- addEpisodeToPodcast(podcastID, episodeTitle, duration, releaseDate, listeningCount, specialGuests, advertisementCount)
  Returns Confirmation

Maintaining payments

- makeRoyaltyPaymentToRecordLabelForSong(albumId, songTitle, month, year)
  Returns confirmation

- makePaymentToArtistsFromRecordLabel(recordLabelAccountID, albumId, songTitle, month, year)
  Returns confirmation

- makePaymentToPodcastHost(podcastHostAccountID, month, year)
  Returns confirmation

- receivePaymentFromSubscriber(userAccountID, month, year, amt)
  Returns confirmation

Application Code

Submitted on Moodle - WolfMedia.java

Transactions

1. **Insert Artist**

   This code block is executing a database transaction that inserts data into two tables: "Account" and "Artist".

   First, the code sets the auto-commit mode of the database connection to false to ensure that both inserts are executed as part of the same transaction.

Then, the code attempts to insert a new account record into the "Account" table using the "insertAccount" function. If the function returns -1, it means that the insertion failed, and the code rolls back the transaction and returns.

Next, the code attempts to insert a new artist record into the "Artist" table using the "insertArtist" function. If this function also returns -1, it means that the insertion failed, the code rolls back the transaction and returns.

If both insertion functions are successful, the code commits the transaction, sets the auto-commit mode back to true, and prints a success message.

If there is any SQL exception during the execution of the transaction, the code catches it, rolls back the transaction, prints an error message, and sets the auto-commit mode back to true.

```
try{
   connection.setAutoCommit(false);
   // register the new account with global 'Account' table first
   if (insertAccount(artistAccountID, "Artist", password) == -1) {
       connection.rollback();
       return;
   }
   // insert the details into Artist table - this will only succeed if the
account details exist in 'Account'
   // because of foreign key constraints
   if (insertArtist(artistAccountID, artistName, status, type, country,
primaryGenre, 0, recordLabelAccountID) == -1) {
       connection.rollback();
       return;
   }
   connection.commit();  //commit only if both statements are executed
   System.out.println("Artist inserted successfully!");
} catch (SQLException e) {
       System.out.println(e.getMessage());
       connection.rollback();  // rollback if there are any errors in the
execution
       System.out.println("Operation Failed. Try Again with Valid Inputs");
} finally {
   connection.setAutoCommit(true);
}
```

2. **Give rating to Podcast**
   This code block is executing a database transaction that updates the rating of a podcast by a user.

   First, the code sets the auto-commit mode of the database connection to false to ensure that both the check for the podcast subscriber and the updates to the two database tables are executed as part of the same transaction.

Then, the code executes a query to check if the user has subscribed to the podcast he wants to rate or not. If the query returns no result, it means that the user is not subscribed to the podcast, and the code prints an error message and returns.

If the user is subscribed to the podcast, the code updates the rating for the podcast in the "PodcastSubscriber" table using the "updatePodcastSubscriberRatingQuery" statement.

After updating the rating for the user in the "PodcastSubscriber" table, the code invokes a function named "updatePodcastRating" that updates the average rating of the podcast in the "Podcast" table based on all the ratings of the podcast subscribers.

If both the update operations are successful, the code commits the transaction, sets the auto-commit mode back to true, and prints a success message.If there is any SQL exception during the execution of the transaction, the code catches it, rolls back the transaction, prints an error message, and sets the auto-commit mode back to true.

```
    connection.setAutoCommit(false);
    // check if the user is subscribed to the podcast he wants to rate or
not
    getPodcastSubscriberQuery.setString(1, podcastID);
    getPodcastSubscriberQuery.setString(2, userAccountID);
    result = getPodcastSubscriberQuery.executeQuery();
    if (!result.next()) {
        System.out.println("Podcast Subscriber not found for given IDs");
        return;
    }
    // update PodcastSubscriber table with the rating the user wants to give
    updatePodcastSubscriberRatingQuery.setDouble(1, rating);
    updatePodcastSubscriberRatingQuery.setString(2, userAccountID);
    updatePodcastSubscriberRatingQuery.setString(3, podcastID);
    updatePodcastSubscriberRatingQuery.executeUpdate();

    //Update average rating for the given podcast considering all existing
ratings.
    updatePodcastRating(podcastID);
    connection.commit();  //commit if both operations are successful
    System.out.println(" Rating is added and Podcast rating is updated
successfully!");
} catch (SQLException e) {
    System.out.println(e.getMessage());
    connection.rollback();  // rollback if there is any error in any of the
operations
    System.out.println("Operation Failed. Try Again with Valid Inputs");
} finally {
    connection.setAutoCommit(true);
}
```

Design Decisions:

The system has a admin menu that gives the user the following options: “Artist”, “Song”, “Album”, “Record Label”, “Podcast”, “Podcast Host”, “Podcast Episode”, “User”, “Account”, “Reports”, “Payments” and “Genres”. When prompted, the user enters a number 0-12 corresponding with what kind of action they would like to take. For each menu option there is a submenu displayed that has specific operations listed. Our system has a main class (Wolfmedia.java) that facilitates switching between the main menu options and submenu’s to deal with the next set of tasks.

Since our user is having an admin role, the user will be given with options that can insert or delete data as well. Our system takes input from the user, depending on which option the user has selected and performs the necessary operations. The user receives appropriate messages on the success/failures of the transactions. The user will also be able to list all contents of all tables through these menu’s itself instead of navigating to/from the database UI. The user can navigate through different Menus through exit/ case 0 i.e, he will be prompted with the sub-menu he is in as long as he doesn’t input 0 to go back to the previous menu.

FUNCTIONAL ROLES

Part 1:

Software Engineer: Teja (Prime), Samarth (Backup)
Database Designer/Administrator: Naveen (Prime), Srilekha(Backup)
Application Programmer: Srilekha (Prime), Naveen(Backup)
Test Plan Engineer: Samarth(Prime), Teja (Backup)

Part 2:

Software Engineer: Srilekha (Prime), Naveen(Backup)
Database Designer/Administrator: Samarth (Prime), Teja (Backup)
Application Programmer: Teja(Prime), Samarth(Backup)
Test Plan Engineer: Naveen (Prime), Srilekha(Backup)

Part 3:

Software Engineer: Naveen (Prime), Samarth(Backup)
Database Designer/Administrator: Teja (Prime), Srilekha(Backup)
Application Programmer: Samarth(Prime), Naveen(Backup)
Test Plan Engineer: Srilekha (Prime), Teja(Backup)