

1. Each user can create a class UserAccount. This class contains the following attributes:
2. userName of type string
3. userID of type int, which are unique for each user
4. userEmail of type string
5. password of type string
6. displayName of type string
7. profilePicture of type byte
8. playListID of type int
9. playList, which is a linked list of class songs

The user can perform the following operations:

1. bool registerUser(string userName, string userEmail, string password): Registers a new user with the provided credentials, which would then be saved to the server. The operation will return true if the registration is successful, otherwise a false will be returned along with a failed to register message.
2. bool login(string userNameOrEmail, string password): The operation will first asks the user to enter a username or email and a password. It will then compare the entered information with the existing username, email, and password stored in the server and returns true if the login is successful.
3. bool updateProfile(int userID, byte profilePicture, string displayName): Updates the user's profile information, such as display name and profile picture. Returns true if the update is successful. If only the profilePicture or displayName was entered, the operation will only update the entered information and retains the other attribute.
4. UserPreferences getPreferences(userID: int): Retrieves the user's preferences, such as preferred genres and customization settings.
5. int addPlaylist(int userID, cusPlaylist playlistDetails): Creates a new playlist for the user and returns its unique identifier.
6. bool customizePlaylist(int userID, int playlistID, cusPlaylist playlistDetails): Customizes an existing playlist based on user preferences. Returns true if customization is successful.
7. List<Playlist> getPlaylists(int userID): Return a list of playlists created by the user.
8. Each song in the library belongs to the songs class. A songs class contains a songType, songName, and songArtist of type string, as well as songDuration and songBPM of type int. The songs class can perform getters functions such as getSongType, getSongName, getSongArtist, getSongDuration, and getSongBPM without taking any parameters.
9. A UserAccount class can have multiple cusPlaylist. A cusPlaylist contains an array of songs which was added by the user. The cusPlaylist can perform many getter operations such as getSongType, getSongName, getSongArtist, getSongDuration, and getSongBPM. These operations take the requested song in the playlist as parameter.
10. To add and remove songs from a playlist, the class PlaylistManager will be used. It contains no attribute but has two operations. The first operation is addSongToPlaylist, which accept a playlistID from the UserAccount and a song of class songs as parameter. It will return true if the song is successfully added to the playlist. The second operation is removeSongFromPlaylist, which also accept a playlistID and a song as parameter. The operation will return true if the song is removed from the playlist.
11. User can have control of a song’s playback by using the class PlaybackController. The class contains no attribute but has the following operations:
12. bool playPlaylist(int playlistID): initiate or resume the playback of the selected playlist and return true if the operation is successful.
13. bool pausePlayback(): pause the current playback and return true if successful.
14. bool skipTrack(): skips to the next songs in the playlist and return true if successful.
15. To integrate the system with Spotify, the class SpotifyAPIManager will be used. It contains no attributes and has the following operations:
16. AuthenticationRequest initiateSpotifyAuthProcess(): Initiates the Spotify authentication process and returns an authentication request.
17. AuthorizationRequest requestSpotifyAuthorization(List<Scope> scopes): Requests authorization from Spotify with specified scopes and returns an authorization request.
18. List<SpotifyTrack> fetchSpotifyCatalog(string query): Sends a request to Spotify's catalog API to retrieve a list of matching tracks.
19. bool addTrackToSpotifyUserPlaylist(int playlistID, int trackID): Adds a track to the user's Spotify playlist and returns true if successful.
20. List<SpotifyPlaylist> fetchSpotifyUserPlaylists(): Retrieves a list of the user's Spotify playlists.
21. The AI can generate a custom playlist for the user by using the generatePlayList class. This class contains a favSongType and favArtist of type string; a favSongDuration; a favSongBPM of type int; and a linked list of class songs named recommendSongs. The SortSongs class can performs the following operations based on an array of cusPlayList as parameter:
22. favSongType, which return the string favSongType
23. favArtist, which return the string favArtist
24. favSongDuration, which return the int favSongDuration
25. favSongBPM, which return the int favSongBPM

In addition, the SortSongs class can perform recommend song operation, which takes all of the class’s attribute and return a recommendSongs of class songs

1. The SessionManager class will be used to manage each login session on a devide. It contains no attribute but has two operations. The first operation is storeSessionData, which accepts a unique userID of type int and a sessionData as parameter. The operation will return true if it manages to store the data on the device. The second operation is getSessionData, which uses the userID as parameter and return the sessionData that has already been stored using the first operation.
2. Different users can take part in creating and managing a playlist through the class CollaborativePlaylist. The class contains no attribute but has two operations. The first operation is addContributor, which accept the playlistID of a particular playlist and a userID of the user taking part in the contribution. The operation will return true if it has successfully added the user. The second operation is removeContributor, which also accept a playlistID and a userID of the user being removed from accessing the playlist. It will return true if the operation is successful.
3. The class UIContoller will handle all information and interfaces being displayed to the user. The class has no attributes and has the following operations:
4. void displayLoginPage(): Display the login page on the device.
5. void displayRegistrationPage(): Display the registration page after the user choose to sign up for a new account.
6. void displayUserProfile(int userID): Displays the user's profile page based on the userID provided.
7. void displayPlaylistGenerationPage(): Display the playlists that is generated by the AI.
8. void displaySpotifyIntegrationPage(): Display the Spotify integration page.
9. void displayCustomizationPage(int playlistID): Display the playlist customization page, which allows the user to add/remove songs or change the playlist’s name.
10. void displayCollaborativePlaylistPage(int playlistID): void: Display the collaborative playlist page and the name of all users contributing to the playlists.
11. void displayPlaybackPage(int playlistID): void: Display the playback page, which include the song’s name, the song’s length, the pause and unpause button, and the skip button
12. To manage all the users, playlists, and the collaborations between playlists, the class DatabaseManager is used. It contains no attributes and has the following operations:
13. bool storeUser(UserAccount userDetails): Stores user data in the database and returns true if successful.
14. UserAccount getUser(string usernameOrEmail): Retrieves user data from the database based on the username or email.
15. int storePlaylist(cusPlaylist playlistDetails): Stores playlist data in the database and returns the unique playlist identifier.
16. cusPlaylist getPlaylist(int playlistID): Retrieves playlist data from the database based on its identifier.
17. bool storeCollaborativePlaylist(CollaborativePlaylist playlistDetails): Stores collaborative playlist data in the database and returns true if successful.
18. CollaborativePlaylist getCollaborativePlaylist(int playlistID): Retrieves collaborative playlist data from the database based on its identifier.
19. To change the template of a playlist, the class TemplateManager will be used. It contains no attribute and has an operation named applyTemplate. The operation takes a cusPlaylist and a template as parameter and will return the same playlist with the updated template.