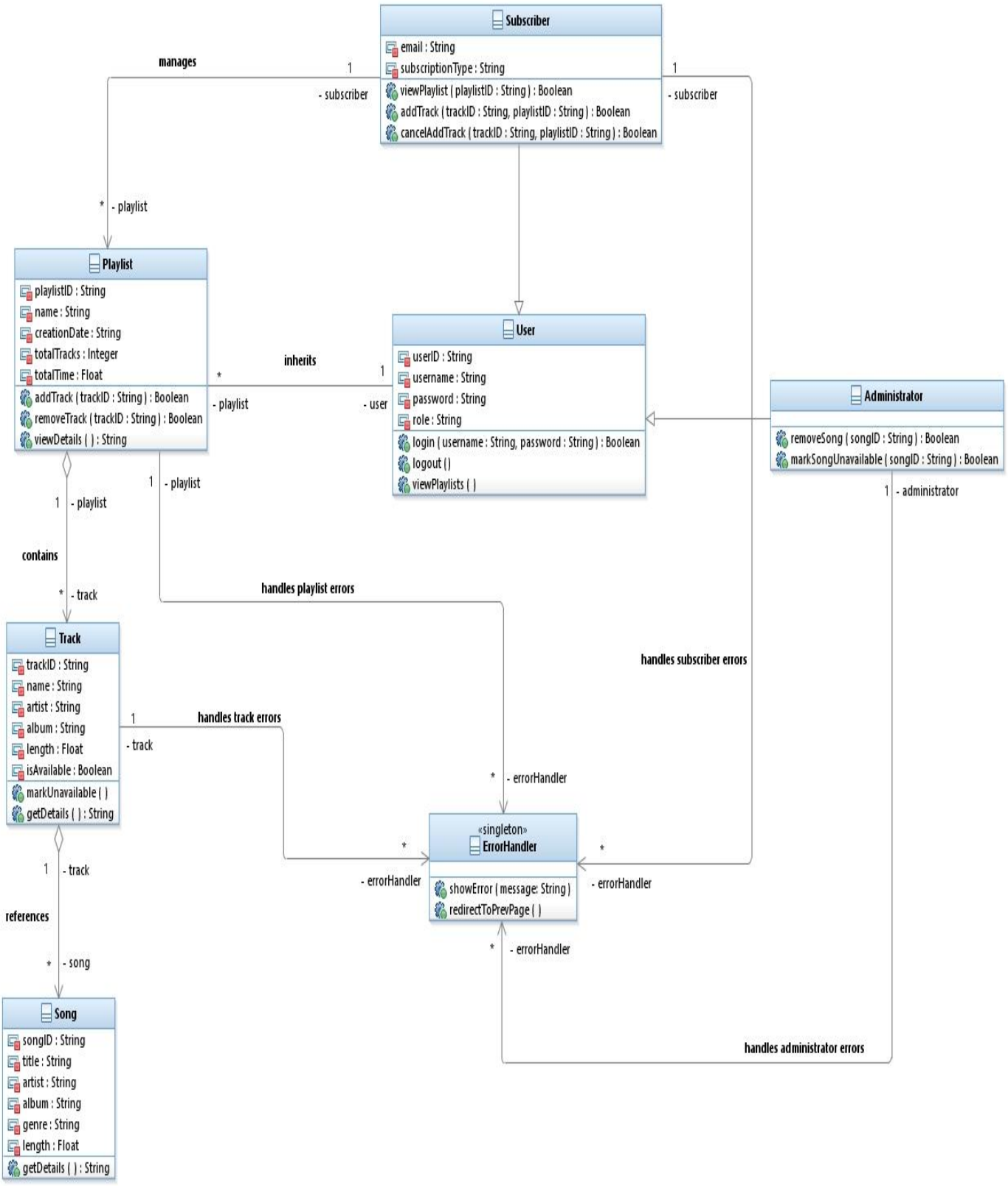
2024-11-08

SOFTWARE ENGINEERING 3

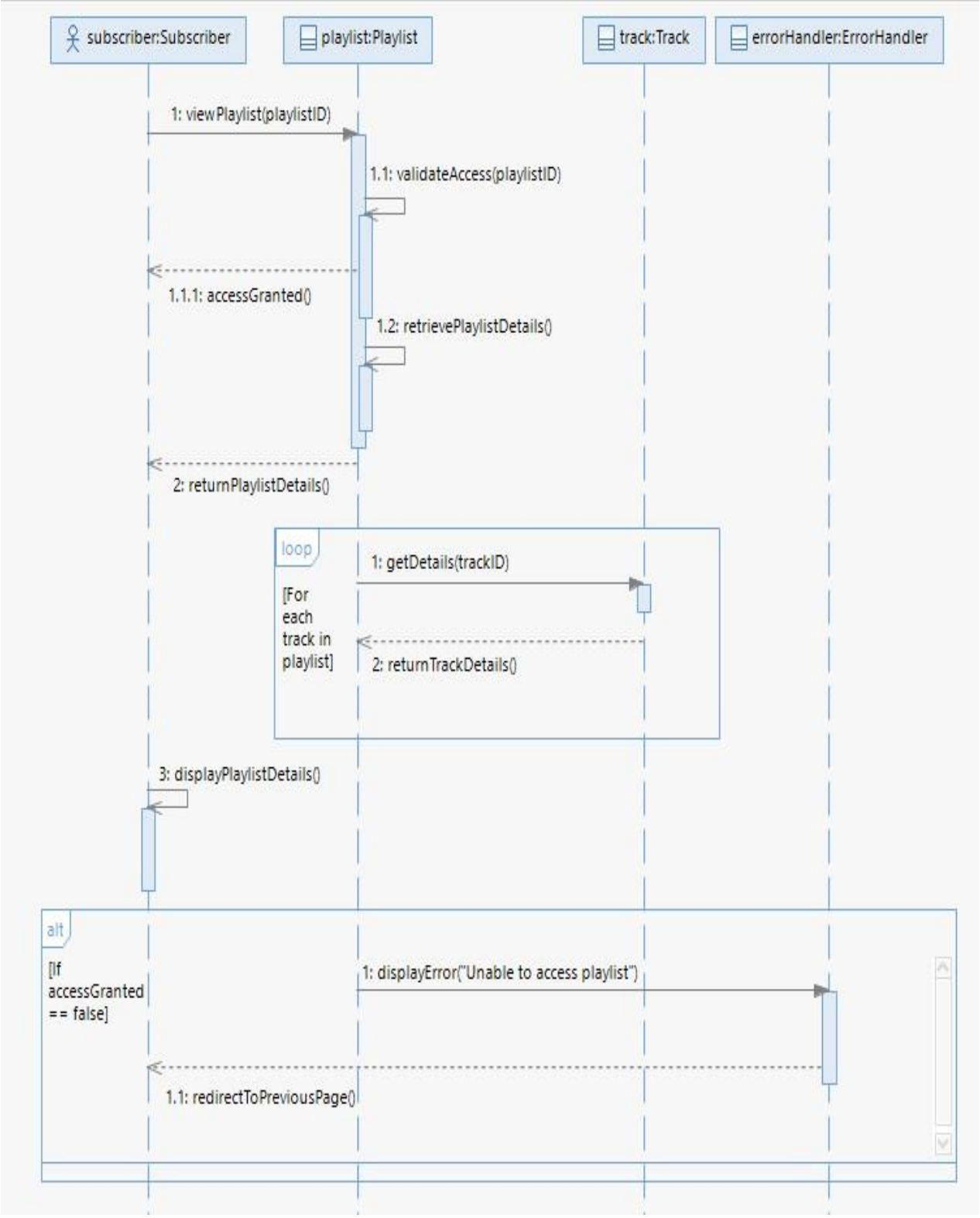
CA 1 ASSIGNMENT – MUSIC PLAYER APP

Paulina Czarnota C21365726

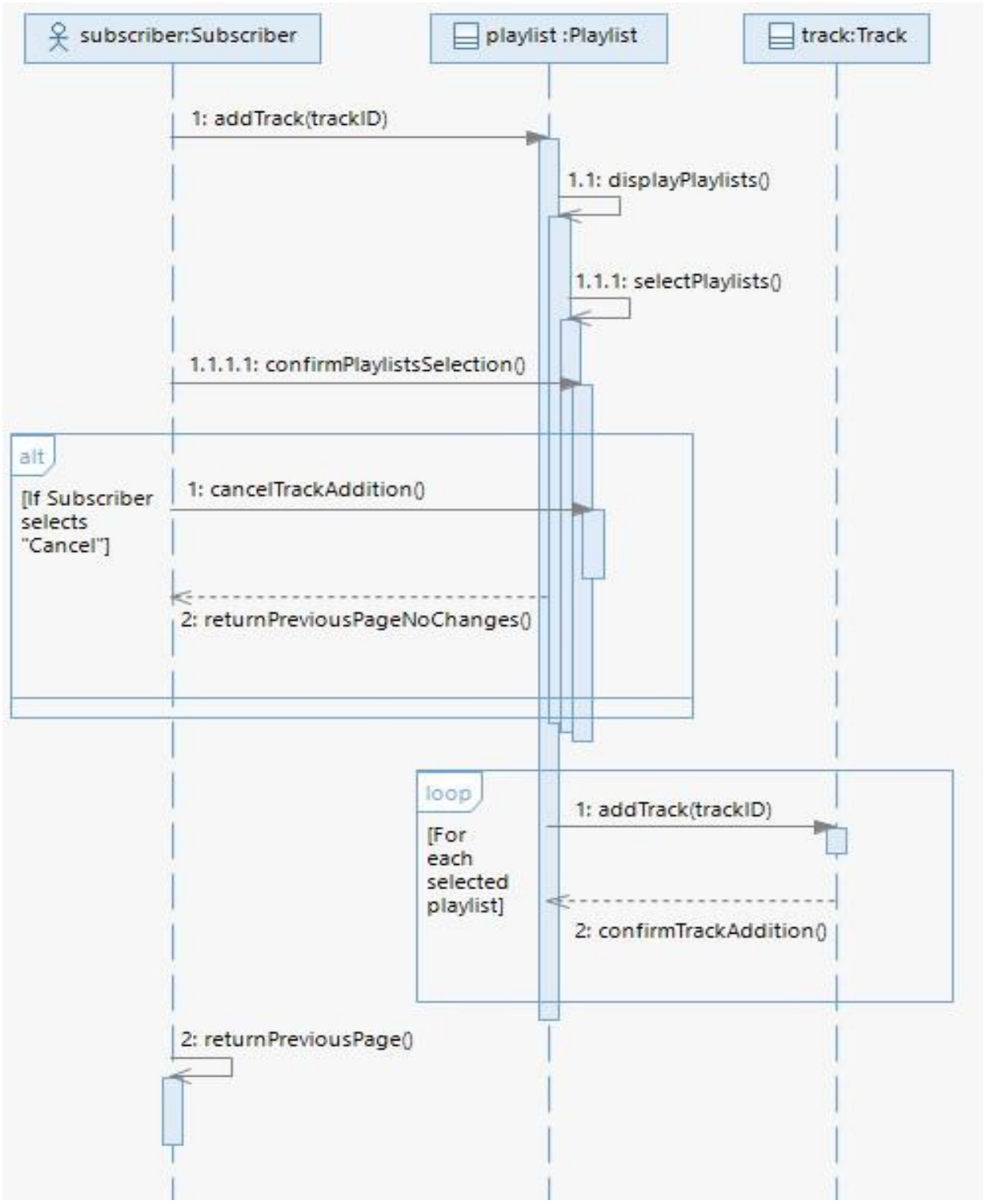
CLASS DIAGRAM



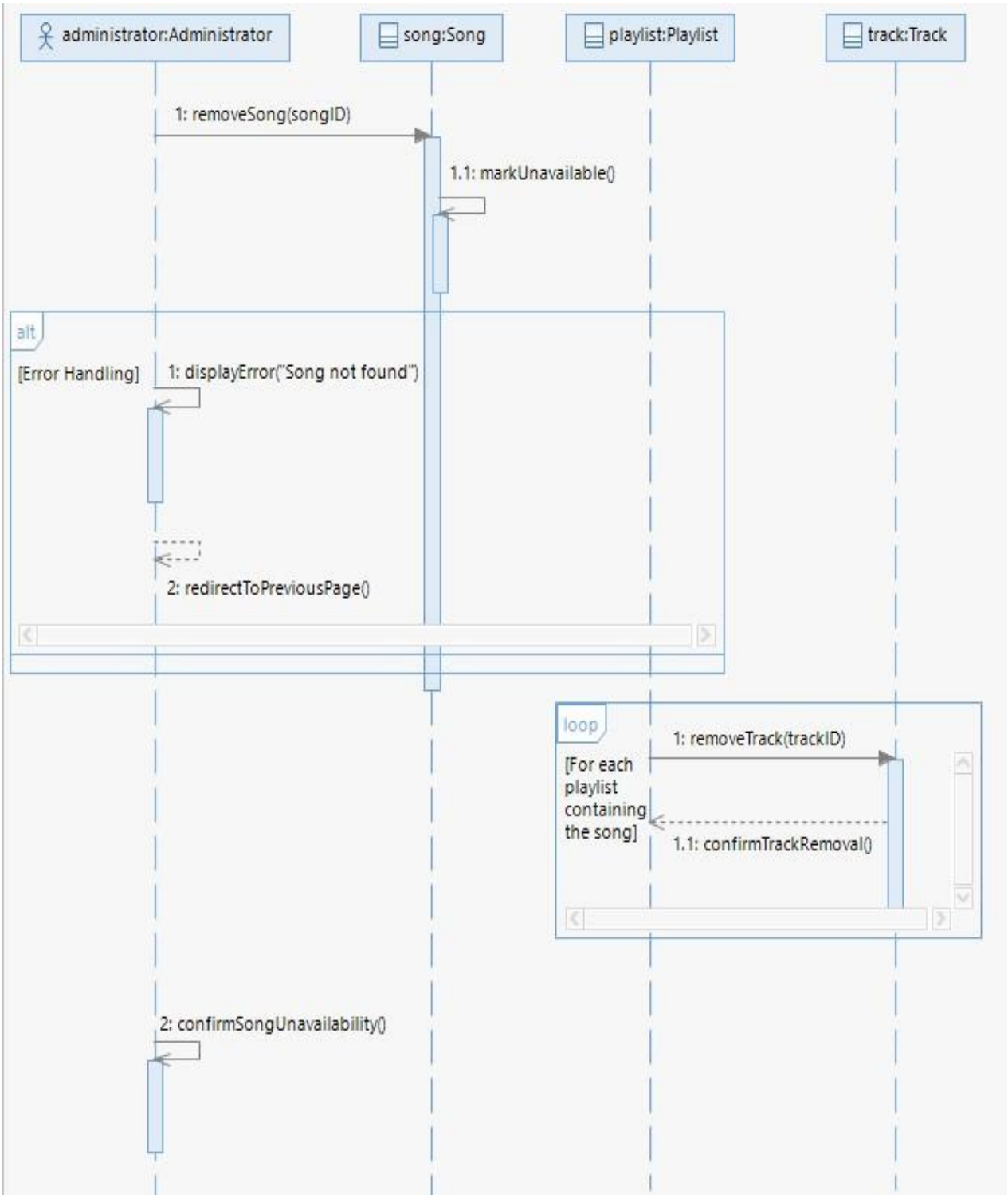
1. SEQUENCE DIAGRAM: VIEW PLAYLIST



2. SEQUENCE DIAGRAM: ADD TRACK



3. SEQUENCE DIAGRAM: REMOVE SONG



CLASS DIAGRAM EXPLANATION

This class diagram models the music playlist application structure, focusing on entities and interactions to support playlist management and administrative controls over songs:

1. **User and Subclasses:**
   - The User class is a base class representing all users, with attributes like userID, username, and password, and methods for login and logout.
   - Subscriber inherits from User, adding attributes like email and subscriptionType, and methods such as viewPlaylist and addTrack for managing playlists.
   - Administrator inherits from User, with methods specific to song management, like removeSong and markSongUnavailable.

2. **Playlist:**
   - Represents a subscriber's playlist and includes attributes like playlistID, name, creationDate, and totalTracks. Methods like addTrack and removeTrack allow the subscriber to modify the playlist.

3. **Track and Song:**
   - Track represents a song instance within a specific playlist, containing trackID, name, artist, and isAvailable.
   - Song holds the actual music data, including metadata like title, artist, and genre. Multiple tracks can reference the same song across different playlists.

4. **ErrorHandler:**
   - A singleton class responsible for centralized error management, with methods like showError and redirectToPrevPage to handle errors uniformly.

5. **Relationships:**
   - A Subscriber can manage multiple Playlists.
   - Each Playlist contains multiple Tracks, each linked to a single Song.
   - ErrorHandler is accessible by all classes involved in operations where errors might occur, providing consistent error handling.

SEQUENCE DIAGRAMS EXPLANATIONS

1. Sequence Diagram: View Playlist

- **Purpose**: This diagram illustrates how a Subscriber views the details of a specific playlist.
- **Flow**:
  1. The Subscriber invokes viewPlaylist(playlistID), which calls the Playlist class to check access permissions with validateAccess.
  2. If access is granted, the system retrieves playlist details with retrievePlaylistDetails, which includes the playlist's metadata.
  3. A loop iterates over each track in the playlist, calling getDetails on each Track to fetch song information.
  4. The playlist and track details are displayed to the Subscriber.
  5. If access is denied, ErrorHandler displays an error message and redirects the Subscriber to the previous page.
- **Multiplicity & Relationships**:
  - Subscriber interacts with multiple Playlists (1-to-many).
  - Playlist contains multiple Tracks (1-to-many).
  - ErrorHandler provides centralized error handling.

2. Sequence Diagram: Add Track

- **Purpose**: This sequence diagram describes how a Subscriber adds a track to one or more playlists.
- **Flow**:
  1. The Subscriber initiates addTrack(trackID), prompting the system to display all available playlists via displayPlaylists.
  2. The Subscriber selects one or more playlists and confirms the selection.
  3. For each selected playlist, a loop iterates and calls addTrack(trackID) on both the Playlist and Track classes, adding the track to the playlist.
  4. If the Subscriber cancels the operation, cancelTrackAddition() is called, and no changes are made.
  5. After adding the track(s), the system returns the Subscriber to the previous page.

- **Multiplicity & Relationships**:
  - Subscriber can manage multiple Playlists (1-to-many).
  - Each Track can be added to multiple Playlists (many-to-many).
  - Error handling is triggered if the user cancels the operation.

3. Sequence Diagram: Remove Song

- **Purpose**: This diagram models the steps an Administrator takes to remove a song and mark it as unavailable.
- **Flow**:
  1. The Administrator selects the removeSong(songID) option, which calls markUnavailable on the Song to set its availability to false.
  2. If the song ID is invalid, ErrorHandler displays a “Song not found” error message and redirects the Administrator to the previous page.
  3. If the song is valid, the system proceeds to iterate through each playlist containing the song, calling removeTrack(trackID) to remove the track from all playlists.
  4. The Administrator confirms the song's unavailability with confirmSongUnavailability().
- **Multiplicity & Relationships**:
  - An Administrator interacts with multiple Songs (1-to-many).
  - Each Song can be referenced by multiple Tracks across playlists (1-to-many).
  - Error handling is managed by ErrorHandler for cases where the song ID is invalid.