APOLLO

HOME AUDIO SYSTEM

Charles-William Desroches
Courtney Wright
Laurent Chenet
Saif Elkholy
Ece Pidik

System Requirements

The following requirements are presented in order of priority, followed by their respective requirement IDs

1. The Apollo shall maintain an organized library of songs, grouped by artist or album.
ID: R01

2. The Apollo shall allow the user to play a song, album, or playlist to one or more desired locations in the home.
ID: R02

3. The Apollo shall provide the ability to add a song by uploading a sound file and specifying the song's title, duration, position in album, genre, release date, and artist name.
ID: R03

4. The Apollo shall provide the ability to add an album by specifying the album's name, genre, release date, duration, and number of songs, as well as uploading or specifying sound files to be placed in the album.
ID: R04

5. The Apollo shall save playlists, room locations, and the song library when not in use.
ID: R05

6. The Apollo shall provide the ability to add a location where audio is to be played in the home.
ID: R06

7. The Apollo shall allow the user to create playlists, by saving songs in a specific order.
ID: R07

8. The Apollo shall provide the ability to control the volume or mute the sound temporarily in any connected location individually.
ID: R08

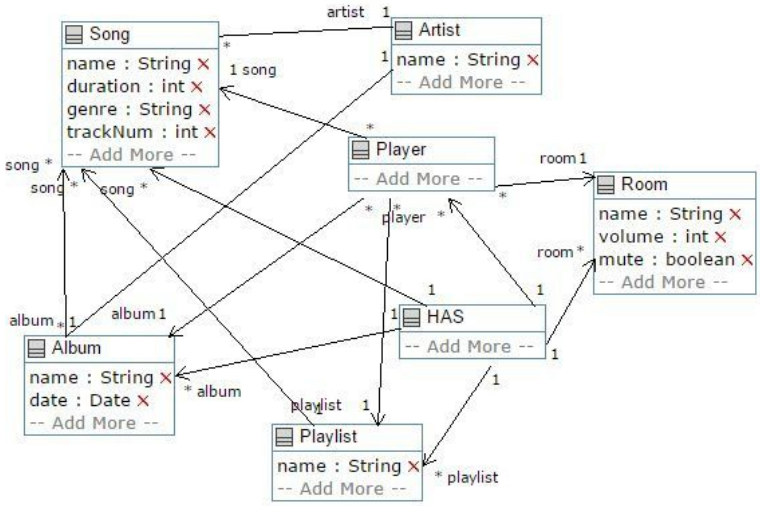
9. The Apollo shall be a responsive application, performing user commands with minimal delay and saving songs and locations input by the user instantly.
ID: R09

10. The Apollo shall be a free application.
ID: R10

11. Updates applied to the Apollo UI shall be constructive and explanatory.
ID: R11

Domain Model

```
class Song {
  name;
  int duration;
  genre;
  int trackNum;
  * -- 1 Artist artist;
}

class Artist {
  name;
  1 -- * Album album;
}

class Album {
  name;
  Date date;
  1 -> * Song song;
}

class Playlist {
  name;
  1 -> * Song song;
}

class Room {
  name;
  int volume;
  boolean mute;
}

class Player {
  * -> 1 Song song;
  * -> 1 Room room;
  * -> 1 Album album;
  * -> 1 Playlist playlist;
}

class HAS {
  singleton;
  1 -> * Player player;
  1 -> * Room room;
  1 -> * Song song;
  1 -> * Album album;
  1 -> * Playlist playlist;
}
```



Use Cases

Title: *Add Album*
Actors: *User*
Requirements: *R04*
Main Scenario:

1. System shows the Add Album form.
2. User enters the name of the album, the genre, the release date, and the number of songs on the album.
3. User provides the location on the machine of every song in the album.
4. The system adds the album to the song library, showing under its artist as well as independently with other albums in the library.
5. System shows the Add Album form.

Alternatives:

4a. One or more of the required information fields are empty or contain only spaces
4a1. System shows one or more of the following error messages: “Cannot have empty fields!” (use case continues at step 1)
4b. The release date entered is not a valid date (i.e. date entered is in the future)
4b1. System shows error message “Incorrect date format!” (use case continues at step 1)
4d. The song location is not valid
4d1. System shows error message “Song location not valid.” (use case continues at step 1)

Title: *Add Song*
Actors: *User*
Requirements: *R03*
Main Scenario:

1. System shows Add Song form.
2. User enters the name of the song, duration, position in album, genre, release date, and artist name, as well as the location of the sound file on the machine
3. System adds the song to the library, saved under the artist and album indicated.
4. System shows Add Song form.

Alternatives:

3a. One or more of the required information fields are empty or contain only spaces
3a1. System shows one or more of the following error message: “Cannot have empty fields!” (use case continues at step 1)
3b. The release date entered is not a valid date
3b1. System shows error message “Incorrect date format!” (use case continues at step 1)
3c. The song location is not valid
3c1. System shows error message “Song location not valid” (use case continues at step 1)

Title: *Create Playlist*
Actors: *User*
Requirements: *R07*
Main Scenario:

1. System shows Create Playlist form.
2. User enters name of playlist.
3. System displays full library of songs, allowing the user to check off the ones they would like included in their playlist
4. System adds the playlist.
5. System shows Create Playlist form.

Alternatives:

4a. The field for Playlist name is empty or contains only spaces.
4a1. System shows error message “Playlist name cannot be empty!”

Title: *Add Location*
Actors: *User*
Requirements: *R06*
Main Scenario:

1. System shows Add Location form
2. User enters name of location.
3. System creates the room in the system.
4. System shows Add Location form.

Alternatives:

3a. The field for name of location is empty or contains only spaces.
3a1. System shows error message “Room Location name cannot be empty!”

Title: *Volume Control*
Actors: *User*
Requirements: *R08*
Main Scenario:

1. System shows Volume Control Menu.
2. User selects location from list of room locations
3. User changes volume to the desired setting or selects to mute the sound.
4. System enacts volume changes in room.
5. System shows Volume Control Menu.

Alternatives:

None

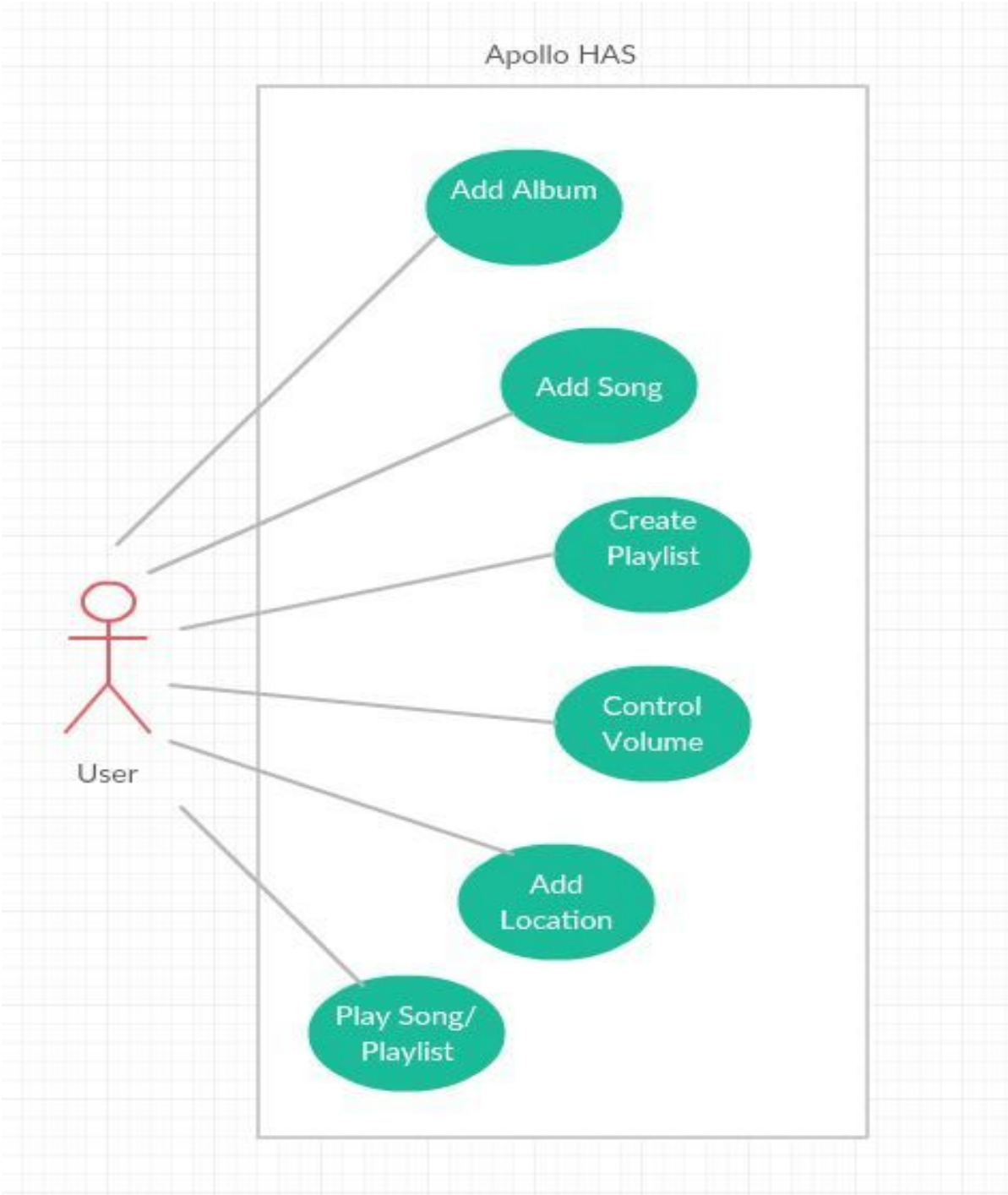
Title: *Play Song/Playlist to Location*
Actors: *User*
Requirements: *R02*
Main Scenario:

1. System shows library.
2. User selects List of Songs, List of Playlists, List of Albums.
3. User selects song to play
   3.1 User selects desired song from list of songs
   3.2 User selects desired album, then selects song in album
   3.3 User selects desired playlist
4. User selects location for song to be played.
5. System plays song in specified location.
6. System shows library.

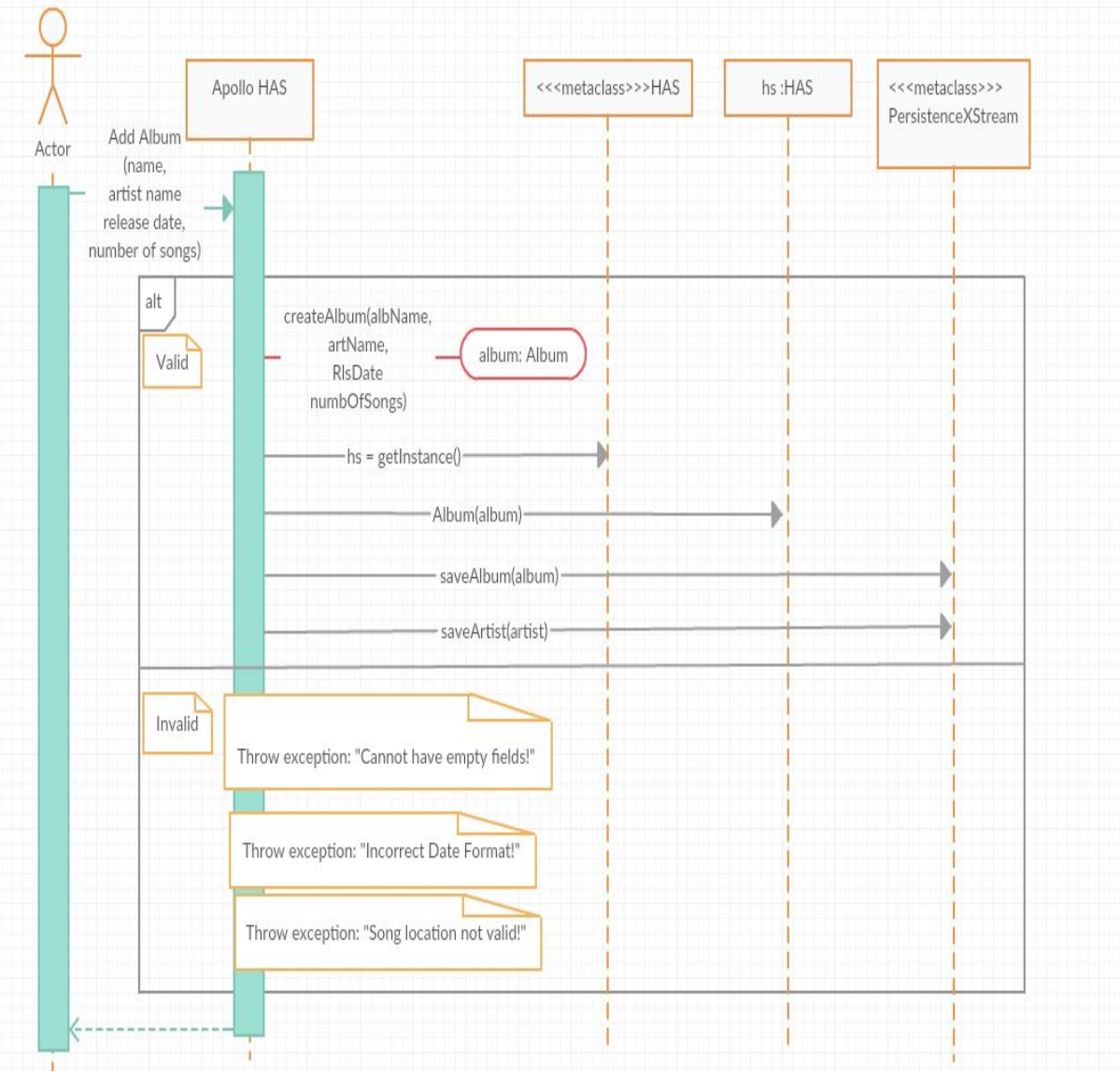
Alternatives

5a. User does not specify a song to play
5a1. System shows error message “Please choose a song to play!” (Use case continues at step 3
5b. User does not specify a location
5b1. System shows error message “Please choose a location to play the song!” (Use case continues at step 4)

Use Case Diagram



Sequence Diagram



Work Plan

The work shall be split evenly among the five team members. Three members will spearhead an application (Desktop, Mobile, Web), with the remaining two members acting as “floaters” who write tests and documentation, and double check or help debug code.

PATCH 1 - Delivered February 22nd at 11:30 pm
REQUIREMENTS ADDRESSED: R03, R04, R05

-Implementation of Add Album, and by consequence, Add Song, and the Persistence functionality. 65%
-Java and PHP code to be generated from conceived Umple domain model 15%
-Use cases defined and use case diagram generated.10%
-Requirements-Level Sequence Diagram for Add Album function generated. 10%

PATCH 2 - Delivered March 7th at 11:30 pm
REQUIREMENTS ADDRESSED: R01, R02

-Architecture for full system proposed. 50%
-Block Diagram generated. 20%
-Discuss and propose OO approach to system design. 30%

PATCH 3 - Delivered March 21st at 11:30 pm
REQUIREMENTS ADDRESSED: R01, R09

-Description of unit testing. 25%
-Description of component testing. 25%
-Description of system testing. 25%
-Description of performance/stress testing. 25%

PATCH 4 - Delivered March 28th at 11:30 pm
REQUIREMENTS ADDRESSED: R06, R07, R08, R10, R11

-Description of release pipeline proposed. 50%
-Finish implementing remaining features. Ensure all use cases function properly. 50%

PATCH 5 - Delivered April 11th-14th
REQUIREMENTS ADDRESSED: R11

-Present the Apollo HAS 100%

PATCH 6 - Delivered April 15th at 11:30 pm
REQUIREMENTS ADDRESSED: R01-R11

-Following presentation and peer review, improve, add, or change non-functional or clunky features, and make the Apollo more aesthetically pleasing. 100%
-Submit finished product source code for evaluation.