Caroline Chen yc3756

Cherry Chu ccc2207

Adam Carpentieri ac4409

Haomin Gui hg2533

Assignment T5: Second Iteration

Part 1: User Stories Revisited

1. As a chatroom user, I want to see other participants to see what they are listening to or have previously listened to.

My conditions of satisfaction are: The UI will show the participants current and recently played songs.

This feature was implemented and tested successfully.

2. As a chatroom user, I want to be able to talk to other chatroom users about the music we are listening to and the genre that the chatroom is about.

My conditions of satisfaction are: the user can send messages into the chatroom and see the messages that other users type and send.

This feature was implemented and tested successfully.

3. As an active participant I want to share new music with other users and also listen to shared music from them.

My conditions of satisfaction are: 1. When you share the music, it should appear on other listener's playing queue in their Spotify app. 2. The UI will reflect that a song has been shared in the chat window.

This feature was implemented and tested successfully.

4. As a chatroom user, I want to add songs from other people's recently played song list to my Spotify play queue.

My conditions of satisfaction are: 1. The UI will show the participant an add button next to each song in the recently played song list. 2. After clicking an add button, the corresponding song will be added to the participant's play queue in Spotify.

This feature was implemented and tested successfully.

Part 2: Equivalence Conditions and Boundary Conditions

Discussion:

For functions that involve api calls, we tested the validity of the return object. For example, for refreshRecentlyPlayed() function. We expect the api to return exactly 10 items.

We have three equivalence classes:

1) Return object has less than 10 items
2) Return object has exactly 10 items
3) Return object has more than 10 items

To perform boundary testing on these equivalence partitions, we tested

1) Return object has 9 items (just below 10)
2) Return object has exactly 10 items
3) Return object has 11 items (just above 10)

Valid and Invalid tests on these boundary conditions are detailed below.

For functions that involve object input, we have two equivalence classes:

1) Not Null object (Valid case)
2) Null Object (Invalid case)

We do not have any boundary conditions for object inputs

For functions that involves string input, we have three equivalence classes:

1) Null string (Invalid case)
2) Empty string (Invalid case)
3) Not empty string (Valid case)

To perform boundary testing on these equivalence partitions, we tested

1) String with length == 0
2) String with length == 1 (just above zero)

Note: there are other string inputs that have different boundary conditions. For example, we have the following equivalence classes:

1) Null String (Invalid case)
2) String with less than 15 characters (Invalid case)
3) String with at least 15 characters (Valid case)

We tested the uri string input in addToQueue() the following boundary conditions:

1) String with length == 14
2) String with length == 15 (just above 14)

Detail equivalence partition and boundary condition for each major method:

User.java

refreshRecentlyPlayed()

Equivalence classes:

1) Api return less then 10 items
2) Api return 10 items
3) Api return more than 10 items
4) Null track object
5) Non null track object
6) Null artist object
7) Non null artist object
8) Null track name
9) Empty track name
10) Not empty track name

- [Invalid][1] testResfreshRecentlyPlayedLessThan10()
  - Boundary Condition: total item just below 10 (9 items tested)
  - Boundary Condition: track name just above 0 (length 1 tested)
- [Valid][2, 5, 7, 10] testResfreshRecentlyPlayed()
  - Boundary Condition: = 10
- [Invalid][3] testResfreshRecentlyPlayedMoreThan10()
  - Boundary Condition: just above 10 (11 items tested)
- [Invalid][4] testResfreshRecentlyPlayedNullTrack()
- [Invalid][6] testResfreshRecentlyPlayedNullArtist()
- [Invalid][8] testResfreshRecentlyPlayedNullTrackName()
- [Invalid][9] testResfreshRecentlyPlayedEmptyTrackName()
  - Boundary Condition: track name length == 0

refreshCurrentlyPlaying()

Equivalence classes:

1) Null CurrentlyPlaying object
2) Non Null CurrentlyPlaying object
3) Null track object
4) Non null track object
5) Null artist object
6) Non null artist object
7) Null track name
8) Empty track name
9) Not empty track name

- [Valid][1] testRefreshCurrentlyPlayingNull()
- [Valid][2, 4, 6, 9] testRefreshCurrentlyPlayingNotNull()
  - Boundary Condition: track name length just above 0 (length 1 tested)
- [Invalid][3] testResfreshCurrentlyPlayingNullTrack()
- [Invalid][5] testResfreshCurrentlyPlayingNullArtist
- [Invalid][7] testResfreshCurrentlyPlayingNullTrackName()
- [Invalid][8] testResfreshCurrentlyPlayingEmptyTrackName()
  - Boundary Condition: track name length at 0 (length 0 tested)

addToQueue()

Equivalence classes:

1) Uri length > 14
2) Uri length <= 14
3) Uri contains "spotify:track:”
4) Uri does not contain "spotify:track:”

- [Valid][1][3] testAddToQueueValidUri()
  - Boundary Condition: just above 14 (length: 15)
- [Invalid][2] Test: testAddToQueueShortUri()
  - Boundary Condition: at 14
- [Invalid][1, 4] testAddToQueueLongInvalidUri()
  - Boundary Condition: just above 14 (length: 15)
- [Invalid][2, 4] testAddToQueueShortInvalidUri()
  - Boundary Condition: at 14

Share()

Equivalence class:

1) Non null chatlist
2) Null chatlist
3) Null current track
4) Non null current track
5) Null genre string
6) Non null genre string

- [Valid][1, 4, 6] shareTestOK()
- [Invalid][2] shareTestNullChatlist()
- [Invalid][3] shareTestNullCurrentTrack()
- [Invalid][5] shareTestNullGenreStr()

refreshSpotifyToken()

Equivalence classes:

1) Token length > 0
2) Empty token
3) Null Token

- [Valid][1] testRefreshSpotifyTokenValid()
  - Boundary Condition: just above 0 (tested length = 1)
- [Invalid][2] testRefreshSpotifyTokenEmptyToken()
  - Boundary Condition: at 0
- [Invalid][3] testRefreshSpotifyTokenNullToken()

SqLite.java

The functions in this file interacts with String and Genre object input.
To set equivalence partition for String input, we tested: 1) null string, 2) empty string and 3) string with length > 0.
Boundary conditions for String input are 1) length == 0, and 2) length > 0
And to test Genre object, we tested: 1) null object, 2) not null object. No boundary condition for genre objects

Method:

InsertAuthenticatedUser()

Equivalence class:

1) Not empty username
2) Null Username

3) Empty Username
4) Not empty token
5) Null Token
6) Empty Token
7) Not empty refresh token
8) Null Refresh Token
9) Empty Refresh Token
10) Not empty session id
11) Null Session Id
12) Empty SessionId

Boundary conditions for String input are 1) length == 0, and 2) length > 0

- [Valid][1, 4, 7, 10] testInsertAuthenticatedUserOK()
  - Boundary Condition: length == 1 for all
- [Invalid][2] testInsertAuthenticatedUserNullUsername()
- [Invalid][3] testInsertAuthenticatedUserEmptyUsername()
  - Boundary Condition: length == 0
- [Invalid][5] testInsertAuthenticatedUserNullToken()
- [Invalid][7] testInsertAuthenticatedUserEmptyToken()
  - Boundary Condition: length == 0
- [Invalid][8] testInsertAuthenticatedUserNullRefreshToken()
- [Invalid][9] testInsertAuthenticatedUserEmptyToken()
  - Boundary Condition: length == 0
- [Invalid][11] testInsertAuthenticatedUserNullSessionId()
- [Invalid][12] testInsertAuthenticatedUserEmptySessionId()
  - Boundary Condition: length == 0

UpdateUserAttribute()

Equivalence class:

1) Not empty attribute
2) Null Attribute
3) Empty Attribute
4) Not empty value
5) Null Value
6) Empty Value
7) Not empty username
8) Null Username
9) Empty Username

Boundary conditions for String input are 1) length == 0, and 2) length > 0

- [Valid][1, 4, 7] testUpdateUserAttributeOK()
  - Boundary Condition: length == 1 for all
- [Invalid][2] testUpdateUserAttributeNullAttribute()
- [Invalid][3] testUpdateUserAttributeEmptyAttribute()
  - Boundary Condition: length == 0
- [Invalid][5] testUpdateUserAttributeNullValue()
- [Invalid][6] testUpdateUserAttributeEmptyValue()
  - Boundary Condition: length == 0

- [Invalid][8] testUpdateUserAttributeNullUsername()
- [Invalid][9] testUpdateUserAttributeEmptyUsername()
    - Boundary Condition: length == 0

InsertUserwithGenre()

Equivalence class:

1) Not empty Username
2) Null Username
3) Empty Username
4) Not empty Genre
5) Null Genre
6) Empty Genre

Boundary conditions for String input are 1) length == 0, and 2) length > 0

- [Valid][1, 4] testInsertUserwithGenreOK()
    - Boundary Condition: length == 1 for Username and Genre
- [Invalid][2] testInsertUserwithGenreNullUsername()
- [Invalid][3] testInsertUserwithGenreEmptyUsername()
    - Boundary Condition: length == 0
- [Invalid][5] testInsertUserwithGenreNullGenre()
- [Invalid][6] testInsertUserwithGenreEmptyGenre()
    - Boundary Condition: length == 0

GetUserByName()

Equivalence class:

1) Not empty Username
2) Null Username
3) Empty Username

Boundary conditions for String input are 1) length == 0, and 2) length > 0

- [Valid][1] testGetUserByNameOK()
    - Boundary Condition: length == 1 for Username
- [Invalid][2] testGetUserByNameNullUsername()
- [Invalid][3] testGetUserByNameEmptyUsername()
    - Boundary Condition: length == 0

AuthenticateUser()

Equivalence class:

1) Not null and not empty code
2) Null code
3) Empty code
4) Not empty session id
5) Null session id
6) Empty session id

Boundary conditions for String input are 1) length == 0, and 2) length > 0

- [Valid][1, 4] testAuthenticateUserOld()
    - Boundary Condition: length == 1 for code and session id
- [Valid][1, 4] testAuthenticateUserNew()
    - Boundary Condition: length == 1 for code and session if

- [Invalid][2] testAuthenticateUserNullCode()
- [Invalid][3] testAuthenticateUserEmptyCode()
    - Boundary Condition: code length == 0
- [Invalid][5] testAuthenticateUserNullSessionId()
- [Invalid][6] testAuthenticateUserEmptySessionId()
    - Boundary Condition: session id length == 0

GetUserCount()

Equivalence class:

1) Not empty for username
2) Null username
3) Empty username

Boundary conditions for String input are 1) length == 0, and 2) length > 0

- [Valid][1] testGetUserCountOK()
    - Boundary Condition: username length == 1
- [Invalid][2] testGetUserCountNullUsername()
- [Invalid][3] testGetUserCountEmptyUsername()
    - Boundary Condition: username length == 0

GetGenreUser()

Equivalence class:

1) Not empty username
2) Null username
3) Empty username

Boundary conditions for String input are 1) length == 0, and 2) length > 0

- [Valid][1] testGetGenreUserOK()
    - Boundary Condition: username length == 1
- [Invalid][2] testGetGenreUserNullUsername()
- [Invalid][3] testGetGenreUserEmptyUsername()
    - Boundary Condition: username length == 0

GetUserBySessionId

Equivalence class:

1) Not empty for session id
2) Null session id
3) Empty session id

Boundary conditions for String input are 1) length == 0, and 2) length > 0

- [Valid][1] testGetUserBySessionIdOK()
    - Boundary Condition: username length == 1
- [Invalid][2] testGetUserBySessionIdNullSessionId()
- [Invalid][3] testGetUserBySessionIdEmptySessionId()
    - Boundary Condition: username length == 0

InsertChatRoom

Equivalence class:

1) Non null Genre
2) Null Genre
3) Not empty link
4) Null Link

5) Empty Link
6) Not empty playlist
7) Null Playlist
8) Empty Playlist

Boundary conditions:

String input are 1) length == 0, and 2) length > 0

- [Valid][1, 3, 6] testInsertChatRoomOK()
  - Boundary Condition: length == 1 for all
- [Invalid][2] testInsertChatRoomNullGenre()
- [Invalid][4] testInsertChatRoomNullLink()
- [Invalid][5] testInsertChatRoomEmptyLink()
  - Boundary Condition: link length == 0
- [Invalid][7] testInsertChatRoomNullPlaylist()
- [Invalid][8] testInsertChatRoomEmptyPlaylist()
  - Boundary Condition: playlist length == 0

InsertParticipant()

Equivalence class:

1) Not null genre
2) Null Genre
3) Not empty username
4) Null Username
5) Empty Username
6) Not empty token
7) Null Token
8) Empty Token
9) Not empty refresh token
10) Null Refresh Token
11) Empty Refresh Token
12) Not empty session id
13) Null Session Id
14) Empty SessionId

Boundary conditions:

String input are 1) length == 0, and 2) length > 0

- [Valid][1, 3, 6, 9, 12] testInsertParticipantOK()
  - Boundary Condition: length == 1 for all string
- [Invalid][2] testInsertParticipantNullGenre()
- [Invalid][4] testInsertParticipantNullUsername()
- [Invalid][5] testInsertParticipantEmptyUsername()
  - Boundary Condition: username length == 0
- [Invalid][7] testInsertParticipantNullToken()
- [Invalid][8] testInsertParticipantEmptyToken()
  - Boundary Condition: token length == 0
- [Invalid][10] testInsertParticipantNullRefreshToken()
- [Invalid][11] testInsertParticipantEmptyRefreshToken()
  - Boundary Condition: refresh token length == 0
- [Invalid][13] testInsertParticipantNullSessionId()

- [Invalid][14] testInsertParticipantEmptySessionId()
    - Boundary Condition: session id length == 0

RemoveParticipant()

Equivalence class:

1) Not null genre
2) Null Genre
3) Not empty username
4) Null Username
5) Empty Username

Boundary conditions:

String input are 1) length == 0, and 2) length > 0

- [Valid][1, 3] testRemoveParticipantOK()
    - Boundary Condition: length == 1 for username
- [Invalid][2] testRemoveParticipantNullGenre()
- [Invalid][4] testRemoveParticipantNullUsername()
- [Invalid][5] testRemoveParticipantEmptyUsername()
    - Boundary Condition: username length == 0

RemoveUserGenre()

Equivalence class:

1) Not empty for username
2) Null Username
3) Empty Username

Boundary conditions:

String input are 1) length == 0, and 2) length > 0

- [Valid][1] testRemoveUserGenreOK()
    - Boundary Condition: username length == 1
- [Invalid][2] testRemoveUserGenreNullUsername()
- [Invalid][3] testRemoveUserGenreEmptyUsername()
    - Boundary Condition: username length == 0

GetChatRoomParticipant()

Equivalence class:

1) Not null for Genre object
2) Null Genre

- [Valid][1] testGetChatRoomParticipantOK()
- [Invalid][2] testGetChatRoomParticipantNullGenre()

InsertSong()

Equivalence class:

1) Not empty username
2) Null Username
3) Empty Username
4) Non null time shared
5) Null Time Shared
6) Non null genre
7) Null Genre
8) Not empty song
9) Null song

10) Empty song

Boundary conditions:

String input are 1) length == 0, and 2) length > 0

- [Valid][1, 4, 6, 8] testInsertSongOK()
  - Boundary Condition: username and song length == 1
- [Invalid][2] testInsertSongNullUsername()
- [Invalid][3] testInsertSongEmptyUsername()
  - Boundary Condition: username length == 0
- [Invalid][5] testInsertSongNullTimeShared()
- [Invalid][7] testInsertSongNullGenre()
- [Invalid][9] testInsertSongNullSong()
- [Invalid][10] testInsertSongEmptySong()
  - Boundary Condition: song length == 0

GetChatRoomPlaylist()

Equivalence class:

1) Not null Genre object
2) Null Genre

- [Valid][1] testGetChatRoomPlaylistOK()
- [Invalid][2] testGetChatRoomPlaylistNullGenre()

InsertSession()

Equivalence class:

1) Not empty time
2) Null time
3) Empty time
4) Not empty session id
5) Null session id
6) Empty session id

Boundary conditions:

String input are 1) length == 0, and 2) length > 0

- [Valid][1, 4] testInsertSessionOK()
  - Boundary Condition: time and session id length == 1
- [Invalid][2] testInsertSessionNullTime()
- [Invalid][3] testInsertSessionEmptyTime()
  - Boundary Condition: time length == 0
- [Invalid][5] testInsertSessionNullSessionId()
- [Invalid][6] testInsertSessionEmptySessionId()
  - Boundary Condition: session id length == 0

UserJoin()

Equivalence classes:

1) Non null genre
2) Null genre
3) Not empty username
4) Null username
5) Empty username
6) Not null chatlist
7) Null chatlist

Boundary conditions:
String input are 1) length == 0, and 2) length > 0

- [Valid][1, 3, 6] testUserJoinOldOK()
  - Boundary Condition: username length == 1
- [Invalid][2] testUserJoinNullGenre()
- [Invalid][4] testUserJoinNullUsername()
- [Invalid][5] testUserJoinEmptyUsername()
  - Boundary Condition: username length == 0
- [Invalid][7] testUserJoinNullChatlist()

UserSend()

Equivalence classes:

1) Not empty username
2) Null username
3) Empty username
4) Not empty text
5) Null text
6) Empty text
7) Not null chatlist
8) Null chatlist

Boundary conditions:
String input are 1) length == 0, and 2) length > 0

- [Valid][1, 4, 7] testUserSendOK()
  - Boundary Condition: username and text length == 1
- [Invalid][2] testUserSendNullUsername()
- [Invalid][3] testUserSendEmptyUsername()
  - Boundary Condition: username length == 0
- [Invalid][5] testUserSendNullText()
- [Invalid][6] testUserSendEmptyText()
  - Boundary Condition: text length == 0
- [Invalid][8] testUserSendNullChatlist()

Connect()

Equvalence classes:

1) Null connection
2) Non null connection

- [Valid][1] testConnectOK()
- [Invalid][2] testConnectExists()

GetAllChatrooms()

Equvialence classes:

1) Non null statement
2) Null statement

- [Valid][1] testGetAllChatRoomsOK()
- [Invalid][2] testGetAllChatRoomsNullStmt

GetLatestSession()

Equvalence classes:

3) Non null statement
4) Null statement

- [Valid][1] testGetLatestSession()
- [Invalid][2] testGetLatestSessionNullStmt

[skip] Update()

- Method simply calls getAllChatRooms() that is equivalence partition tested.

ChatRoom.java

AddParticipant()

Equivalence class:

1) not null User object
2) Null User object
3) Not empty username
4) User with null username
5) User with empty username

Boundary condition: username length == 0 and username length > 0

- [Valid][1, 3] addParticipantTestOK()
  - Boundary Condition: username length == 1
- [Invalid][2] addParticipantTestNullUser()
- [Invalid][4] addParticipantTestNullUsername()
- [Invalid][5] addParticipantTestEmptyUsername()
  - Boundary Condition: username length == 0

AddMessage()

Equivalence classes:

1) not null Message object
2) Null Message object
3) Message with not empty content
4) Message with null content
5) Message with empty content

Boundary condition: message content length == 0 and message content length > 0

- [Valid][1, 3] addMessageTestOK()
  - Boundary Condition: message content length == 1
- [Invalid][2] addMessageTestNullMessage()
- [Invalid][4] addMessageTestNullMessageContent()
- [Invalid][5] addMessageTestEmptyMessageContent()
  - Boundary Condition: message content length == 0

AddSong()

Equivalence class:

1) not null Song object
2) Null song object
3) Song with not empty song name
4) Song with null song name
5) Song with empty song name

Boundary condition: song name length == 0 and song name length > 0

- [Valid][1, 3] addSongTestOK()
  - Boundary Condition: song name length == 1
- [Invalid][2] addSongTestNullSong()
- [Invalid][4] addSongTestNullSongName()

- [Invalid][5] addSongTestEmptySongName()
    - Boundary Condition: song name length == 0

Genre.java

IsValidGenre()

Equivalence classes:

1) not empty genre string
2) null genre string
3) empty genre string

Boundary condition: genre string length < 3 and genre string length >= 3

- [Valid][1] isValidGenreTrueTest()
    - Boundary condition: Test at length == 3
- [Invalid][2] isValidGenreFalseTest()
    - Boundary condition: Test at length == 2
- [Invalid][3] isValidGenreNullTest()

MyApi.java

RecentlyPlayed()

Equivalence class:

1) Api object is not null
2) Api object is null

- [Valid][1] recentlyPlayedTestOK()
- [Invalid][2] recentlyPlayedTestNullApi()

CurrentlyPlaying()

Equivalence class:

1) Api object is not null
2) Api object is null

- [Valid][1] currentlyPlayingTestOK()
- [Invalid][2] currentlyPlayingTestNullApi()

AddSong()

Equivalence class:

1) Api object is not null
2) Api object is null

- [Valid][1] addSongTestOK()
- [Invalid][2] addSongTestNullApi()

GetSpotifyTokenFromCode()

Equivalence classes:

1) Non null code
2) null code
3) empty code

Boundary Condition: code length == 0, code length > 0

- [Valid][1] getSpotifyTokenFromCodeTestOK()
    - Boundary Condition: code length at 1
- [Invalid][2] getSpotifyTokenFromCodeNullCode()
- [Invalid][3] getSpotifyTokenFromCodeEmptyCode()
    - Boundary Condition: code length == 0

RefreshSpotifyToken()

Equivalence classes:

1) Non null token
2) null token
3) empty token

Boundary Condition: token length == 0, token length > 0

- [Valid][1] refreshSpotifyTokenTestOK ()
  - Boundary Condition: token length > 0
- [Invalid][2] refreshSpotifyTokenTestNullToken()
- [Invalid][3] refreshSpotifyTokenTestEmptyToken()
  - Boundary Condition: token length == 0

GetEmailFromSpotifyToken()

Equivalence classes:

1) Non null token
2) null token
3) Empty token

Boundary Condition: token length == 0, token length > 0

- [Valid][1] getEmailFromSpotifyTokenTestOK ()
  - Boundary Condition: token length > 0
- [Invalid][2] getEmailFromSpotifyTokenTestNullToken()
- [Invalid][3] getEmailFromSpotifyTokenTestEmptyToken()
  - Boundary Condition: token length == 0

[skip] BuildFormDataFromMap

- Private helper method

<u>ChatList.java</u>

GetChatroomByGenreTest()

Equivalence classes:

1) Not null genre object
2) null genre object

- [Valid][1] getChatroomByGenreTestOK()
- [Invalid][2] getChatroomByGenreTestNullGenre()

GetTotalParticipants()

Equivalence classes:

3) Not null chatroom list
4) null chatroom list

- [Valid][1] getTotalParticipantsTest()
- [Invalid][2] getTotalParticipantsTestNullChatroomList()

RefreshChatList()

Equivalence classes:

1) not null chatroom list
2) chatlist with null chatroom list
3) not null chatrooms
4) chatlist with null chatrooms

- [Valid][1, 3] refreshChatListTestWithToken()
- [Valid][1, 3] refreshChatListTestWithoutToken()

- [Invalid][2] refreshChatListTestNullChatroomList()
- [Invalid][4] refreshChatListTestNullChatroom()

StartChat.java

[skip] InitializeChatList()

- Minor method that calls other db methods to initialize chatlist

[skip] RefreshSongDataRepeatly()

- Helper method that calls thread in Thread.java which is tested

/auth endpoint

Equivalence classes:

1) SessionId in database
2) SessionId not in database

- [Valid][1] authTest()
- [Invalid][2] authTestNoSessionId()

/process_auth

Equivalence classes:

1) Path contains code
2) Path does not contain code
3) SessionId in database
4) SessionId not in database

- [Valid][1, 3] processAuthTest()
- [Invalid][2] processAuthTestNoCode()
- [Invalid][4] processAuthTestNoSessionId()

[skip] /

- Endpoint handle redirection

[skip] /home

- Endpoint handle redirection

[skip] /lobby

- Endpoint handle redirection

[skip] /chatrooms

- Endpoint simply calls getAllChatRooms() that is equivalence partition tested.

/joinroom/:genre

Equivalence classes:

1) Valid Genre object
2) Invalid Genre object

- [Valid][1] joinroomTest()
- [Invalid][2] InvalidJoinroomTest()

/chatroom/:genre

Equivalence classes:

1) Valid genre type
2) Invalid genre type

- [Valid][1] validChatroomGenreTest()
- [Invalid][2] invalidChatroomGenreTest()

/send

Equivalence class:

1) User's genre matches room genre
2) User's genre does not match room genre
- [Valid][1] sendMessageTest
- [Invalid][2] sendMessageInvalidTest

/add

Equivalence classes:

1) Non null song uri
2) Null song uri
- [Valid][1] addSongTest()
- [Invalid][2] addSongTestNoSong()

/share

Equivalence classes:

1) User current track is null
2) User current track is not null
5) [Invalid][1] shareSongNullTest()
6) [Valid][2] shareSongNotNullTest()

/leaveroom/:genre

Equivalence classes:

3) User's genre matches room genre
4) User's genre does not match room genre
7) [Valid][1] lasttoleaveRoomTest()
8) [Invalid][2] InvalidLeaveRoomTest()

Thread.java

Run()

Equivalence classes:

1) Not null database
2) Null database
- [Valid][1] runTestOK()
- [Invalid][2] runTestNullDb()

Link to automatic test suite:
https://github.com/AutumnAudio/AutumnAudio/tree/master/src/test/java

Part 3: Branch Coverage

We achieved a total 98% branch coverage and 95% instruction coverage. The missing branches are caused by the try-catch blocks which catch exceptions in SQL executions and in Spotify API calls.

One example of missing branches in Spotify API calls is that it is possible that the access token we received earlier from Spotify is expired so the API call will throw an exception. To make sure our application works perfectly under this situation, we catched the exception and called another

API to refresh the access token. Since it is difficult for us to simulate this exception, the branch remains uncovered.

Link to coverage report screenshot:
[total]https://github.com/AutumnAudio/AutumnAudio/blob/master/documents/coverage-total-screenshot.png

[controllers]https://github.com/AutumnAudio/AutumnAudio/blob/master/documents/coverage-controllers-classes-screenshot.png

[models]https://github.com/AutumnAudio/AutumnAudio/blob/master/documents/coverage-models-classes-screenshot.png

Link to coverage report: (this interactive html report works by downloading the html file and depending /autumn_audio folder to your local machine)
https://github.com/AutumnAudio/AutumnAudio/blob/master/documents/coverage-report.html

Part 4: Continuous Integration

We use TravisCI for continuous integration in this project. We configure the tool using a file called .travis.yml. Every time a commit is pushed to any branch, a node test for the frontend and a java test for the backend is run. Reports are only deployed when commit is to the *master* branch.

Link to .travis.yml: https://github.com/AutumnAudio/AutumnAudio/blob/master/.travis.yml

The node and java reports from travisCI build are automatically deployed to a separate branch (called travisCI-reports) in our repository.

Link to travisCI reports: https://github.com/AutumnAudio/AutumnAudio/tree/travisCI-reports