Equivalence Partition and Boundary Testing

User.java

1. refreshRecentlyPlayed()

* this function calls the SpotifyAPI and return 10 items. The equivalence classes are:

1. [Invalid] api returns less than 10 items
   * Boundary Condition: just below 10 (9 items tested)
   * Test: testResfreshRecentlyPlayedLessThan10
2. [Valid] api returns exactly 10 items
   * return ok message and continue
   * Test: testResfreshRecentlyPlayed
3. [Invalid] api returns more than 10 items
   * Boundary Condition: just above 10 (11 items tested)
   * Test: testResfreshRecentlyPlayedMoreThan10

* If api returns 10 items, we checks the validity of items in the returned data structure. The equivalence classes are (Given valid api response):

1. [Invalid] Null track object
   * Test: testResfreshRecentlyPlayedNullTrack
2. [Valid] Non null track object
   * Test: testResfreshRecentlyPlayed

(Given non null track object)

1. [Invalid] Null artist object
   * Test: testResfreshRecentlyPlayedNullArtist
2. [Valid] Non null artist object (Given non null track object)
   * Test: testResfreshRecentlyPlayed
3. [Invalid] Null track name
   * Test: testResfreshRecentlyPlayedNullTrackName
4. [Invalid] Non null but empty track name (Track name string size = 0)
   * Boundary Condition: at 0 (length 0 tested)
   * Test: testResfreshRecentlyPlayedEmptyTrackName
5. [Valid] Non null and non empty track name (Track name string size > 0)
   * Boundary Condition: just above 0 (length 1 tested)
   * Test: testResfreshRecentlyPlayed
6. refreshCurrentlyPlaying()

The equivalence classes are:

1. [Valid] api return null CurrentlyPlaying object
   * Indicates no songs are playing right now
   * Test: testRefreshCurrentlyPlayingNull
2. [Valid] api returns non CurrentlyPlaying object
   * Indicates a song is playing right now
   * Test: testRefreshCurrentlyPlayingNotNull

(Given non null currentlyPlaying object)

1. [Invalid] Null track object
   * Test: testResfreshCurrentlyPlayingNullTrack
2. [Valid] Non null track object
   * Test: testResfreshCurrentlyPlayingNotNull

(Given non null track object)

1. [Invalid] Null artist object
   * Test: testResfreshCurrentlyPlayingNullArtist
2. [Valid] Non null artist object (Given non null track object)
   * Test: testResfreshCurrentlyPlayingNotNull
3. [Invalid] Null track name
   * Test: testResfreshCurrentlyPlayingNullTrackName
4. [Invalid] Non null but empty track name (Track name string size = 0)
   * Boundary Condition: at 0 (length 0 tested)
   * Test: testResfreshCurrentlyPlayingEmptyTrackName
5. [Valid] Non null and non empty track name (Track name string size > 0)
   * Boundary Condition: just above 0 (length 1 tested)
   * Test: testResfreshCurrentlyPlayingNotNull
6. addToQueue(String uri)

The equivalence classes for the input string are:

1. [Valid] uri length is > 14 and contains "spotify:track:”
   * Boundary Condition: just above 14 (length: 15)
   * Test: testAddToQueueValidUri
2. [Invalid] uri length is <= 14
   * Boundary Condition: at 14
   * Test: testAddToQueueShortUri
3. [Invalid] uri length is > 14 but not contain "spotify:track:”
   * Test: testAddToQueueInvalidUri
4. refreshSpotifyToken()

This function call the SpotifyAPI to refresh token. The equivalence classes are:

1. [Valid] token has length > 0
   * Boundary Condition: just above 0 (tested length = 1)
   * Test: testRefreshSpotifyTokenValid
2. [Invalid] Empty token
   * Boundary Condition: at 0
   * Test: testRefreshSpotifyTokenEmptyToken
3. [Invalid] Null token
   * Test: 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 null and not empty for all
2. Null Username
3. Empty Username
4. Null Token
5. Empty Token
6. Null Refresh Token
7. Empty Refresh Token
8. Null Session Id
9. Empty SessionId

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

* [Valid][1] testInsertAuthenticatedUserOK()
  + Boundary Condition: length == 1 for all
* [Invalid][2] testInsertAuthenticatedUserNullUsername()
* [Invalid][3] testInsertAuthenticatedUserEmptyUsername()
  + Boundary Condition: length == 0
* [Invalid][4] testInsertAuthenticatedUserNullToken()
* [Invalid][5] testInsertAuthenticatedUserEmptyToken()
  + Boundary Condition: length == 0
* [Invalid][6] testInsertAuthenticatedUserNullRefreshToken()
* [Invalid][7] testInsertAuthenticatedUserEmptyToken()
  + Boundary Condition: length == 0
* [Invalid][8] testInsertAuthenticatedUserNullSessionId()
* [Invalid][9] testInsertAuthenticatedUserEmptySessionId()
  + Boundary Condition: length == 0

UpdateUserAttribute()

Equivalence class:

1. Not null and not empty for all
2. Null Attribute
3. Empty Attribute
4. Null Value
5. Empty Value
6. Null Username
7. Empty Username

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

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

InsertUserwithGenre()

Equivalence class:

1. Not null and not empty for Username and Genre
2. Null Username
3. Empty Username
4. Null Genre
5. Empty Genre

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

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

AuthenticateUser()

Equivalence class:

1. Not null and not empty for code and session id
2. Null code
3. Empty code
4. Null session id
5. Empty session id

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

* [Valid][1] testAuthenticateUserOld()
  + Boundary Condition: length == 1 for code and session id
* [Valid][1] testAuthenticateUserNew()
  + Boundary Condition: length == 1 for code and session if
* [Invalid][2] testAuthenticateUserNullCode()
* [Invalid][3] testAuthenticateUserEmptyCode()
  + Boundary Condition: code length == 0
* [Invalid][4] testAuthenticateUserNullSessionId()
* [Invalid][5] testAuthenticateUserEmptySessionId()
  + Boundary Condition: session id length == 0

GetUserCount()

Equivalence class:

1. Not null and 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 null and 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] testGetGenreUserOK()
  + Boundary Condition: username length == 1
* [Invalid][2] testGetGenreUserNullUsername()
* [Invalid][3] testGetGenreUserEmptyUsername()
  + Boundary Condition: username length == 0

GetUserBySessionId

Equivalence class:

1. Not null and 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. Not null and not empty for Genre, link and playlist
2. Null Genre
3. Null Link
4. Empty Link
5. Null Playlist
6. Empty Playlist

Boundary conditions:

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

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

InsertParticipant()

Equivalence class:

1. Not null and not empty for all String and object
2. Null Genre
3. Null Username
4. Empty Username
5. Null Token
6. Empty Token
7. Null Refresh Token
8. Empty Refresh Token
9. Null Session Id
10. Empty SessionId

Boundary conditions:

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

* [Valid][1] testInsertParticipantOK()
  + Boundary Condition: length == 1 for all string
* [Invalid][2] testInsertParticipantNullGenre()
* [Invalid][3] testInsertParticipantNullUsername()
* [Invalid][4] testInsertParticipantEmptyUsername()
  + Boundary Condition: username length == 0
* [Invalid][5] testInsertParticipantNullToken()
* [Invalid][6] testInsertParticipantEmptyToken()
  + Boundary Condition: token length == 0
* [Invalid][7] testInsertParticipantNullRefreshToken()
* [Invalid][8] testInsertParticipantEmptyRefreshToken()
  + Boundary Condition: refresh token length == 0
* [Invalid][9] testInsertParticipantNullSessionId()
* [Invalid][10] testInsertParticipantEmptySessionId()
  + Boundary Condition: session id length == 0

RemoveParticipant()

Equivalence class:

1. Not null and not empty for all String and object
2. Null Genre
3. Null Username
4. Empty Username

Boundary conditions:

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

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

RemoveUserGenre()

Equivalence class:

1. Not null and not empty for all 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 null and not empty for all string and object
2. Null Username
3. Empty Username
4. Null Time Shared
5. Null Genre
6. Null song
7. Empty song

Boundary conditions:

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

* [Valid][1] testInsertSongOK()
  + Boundary Condition: username and song length == 1
* [Invalid][2] testInsertSongNullUsername()
* [Invalid][3] testInsertSongEmptyUsername()
  + Boundary Condition: username length == 0
* [Invalid][4] testInsertSongNullTimeShared()
* [Invalid][5] testInsertSongNullGenre()
* [Invalid][6] testInsertSongNullSong()
* [Invalid][7] 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 null and not empty for all time and session id
2. Null time
3. Empty time
4. Null session id
5. Empty session id

Boundary conditions:

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

* [Valid] testInsertSessionOK()
  + Boundary Condition: time and session id length == 1
* [Invalid] testInsertSessionNullTime()
* [Invalid] testInsertSessionEmptyTime()
  + Boundary Condition: time length == 0
* [Invalid] testInsertSessionNullSessionId()
* [Invalid] testInsertSessionEmptySessionId()
  + Boundary Condition: session id length == 0

[skip] GetLatestSession()

* Helper method for testing purpose

[skip] Update()

* Method that call getAllChatRooms()

[skip] Connect()

* Minor method Setup database connection

ChatRoom.java

AddParticipant() – has User object input

Equivalence class:

1. not null and not empty User object
2. Null object
3. User with null username
4. User with empty username

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

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

AddMessage() – has Message object input

Equivalence classes:

1. not null and not empty Message object
2. Null object
3. Message with null content
4. Message with empty content

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

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

AddSong() – has Song object input

Equivalence class:

1. not null and not empty Song object
2. Null object
3. Song with null song name
4. Song with empty song name

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

* [Valid][1] addSongTestOK()
  + Boundary Condition: song name length == 1
* [Invalid][2] addSongTestNullSong()
* [Invalid][3] addSongTestNullSongName()
* [Invalid][4] addSongTestEmptySongName()
  + Boundary Condition: song name length == 0

Genre.java

IsValidGenre() – has genre String input

Equivalence classes:

1. not null and not empty string
2. null string
3. empty 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()

SpotifyAPI.java

[skip] RecentlyPlayed()

* Helper method that simply calls API

[skip] CurrentlyPlaying()

* Helper method that simply calls API

[skip] AddSong()

* Helper method that simply calls API

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

ChatList.java

[skip] Size()

* Minor method that returns size of array

[skip] GetChatroomByGenreTest()

* Minor method that get values from map

GetTotalParticipants()

Equivalence classes:

1. Not null chatroom list
2. null chatroom list

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

RefreshChatList()

Equivalence classes:

1. chatlist object with not null chatroom list and chatrooms
2. chatlist with null chatroom list
3. chatlist with null chatrooms

* [Valid][1] refreshChatListTestWithToken()
* [Valid][1] refreshChatListTestWithoutToken()
* [Invalid][2] refreshChatListTestNullChatroomList()
* [Invalid][3] refreshChatListTestNullChatroom()

StartChat.java

[skip] InitializeChatList()

* Minor methods that calls other db methods to initialize chatlist

[skip] RefreshSongDataRepeatly()

* Private methods that calls refreshChatList() in ChatList.java