Test Case 1: Adding and Retrieving Songs

Preconditions:

* An instance of RecentlyPlayedSongsStore is created with an initial capacity of 3 and a maximum of 3 songs per user.

Test Steps:

1-Add "Song1" for "User1" using the addSong method.

2-Add "Song2" for "User1."

3-Add "Song3" for "User1."

4-Retrieve the recently played songs for "User1" using the getRecentlyPlayedSongs method.

5-Verify that the retrieved songs match the expected order: ["Song1", "Song2", "Song3"].

6-Record the output.

Expected Result:

* The retrieved songs for "User1" should be in the order they were added.

Test Case 2: Adding Songs for Another User

Preconditions:

* Continue from Test Case 1.

Test Steps:

1-Add "SongA" for "User2" using the addSong method.

2-Add "SongB" for "User2."

3-Retrieve the recently played songs for "User2."

4-Verify that the retrieved songs match the expected order: ["SongA", "SongB"].

Record the output.

* Expected Result:The retrieved songs for "User2" should be in the order they were added.

Test Case 3: Adding More Songs for an Existing User

Preconditions:

* Continue from Test Case 2.

Test Steps:

1-Add "Song4" for "User1."

2-Retrieve the recently played songs for "User1."

3-Verify that the retrieved songs match the expected order: ["Song2", "Song3", "Song4"].

4-Record the output.

5-Attempt to add "Song5" for "User1."

6-Retrieve the recently played songs for "User1" again.

7-Verify that the retrieved songs match the expected order after reaching the maximum limit: ["Song3", "Song4", "Song5"].

8-Record the output.

Expected Result

* The retrieved songs for "User1" should be in the order they were added, and when the maximum limit is reached, the oldest song should be removed.

Test Case 4: Adding Songs for a New User

Preconditions:

* Continue from Test Case 3.

Test Steps:

1-Add "SongX" for a new user, "User3."

2-Retrieve the recently played songs for "User3."

3-Verify that the retrieved songs match the expected order: ["SongX"].

4-Record the output.

5-Expected Result:

* The retrieved songs for "User3" should be in the order they were added.

Test Case 5: Retrieving Songs for a Non-Existent User

Preconditions:

* Continue from Test Case 4.

Test Steps:

1-Attempt to retrieve the recently played songs for a user that does not exist, e.g., "User4."

2-Verify that an empty list is returned as there are no songs for the non-existent user.

3-Record the output.

Expected Result:

* An empty list should be returned when trying to retrieve songs for a user that does not exist.

Test Case 6: Adding Songs for Multiple Users

Preconditions:

* Continue from Test Case 4.

Test Steps:

1-Add "Song5" for "User1."

2-Add "SongC" and "SongD" for "User2."

3-Retrieve the recently played songs for "User1" and "User2."

4-Verify that the retrieved songs match the expected order and are limited to the maximum allowed songs:

5-For "User1": ["Song3", "Song4", "Song5"] (limited to 3 songs)

6-For "User2": ["SongA", "SongB", "SongC"] (limited to 3 songs).

7-Record the output.

Expected Result:

* The retrieved songs for "User1" and "User2" should be in the order they were added, and they should be limited to the specified maximum number of songs.