MEHMET YİGİT YİLDİRİM

TABLE OF CONTENTS

I. Introduction

Spotify (Spotify - Web Player: Music for everyone) is one of the world's leading music streaming platforms, offering a broad array of features including personalized music recommendations, curated playlists, podcasts, and a social component allowing users to follow friends and share music. Despite its vast user base and complex feature set, the goal remains to deliver a seamless, intuitive, and enjoyable user experience across all platforms, including the web application.

This report will focus on the Spotify web application - a platform enabling users to access Spotify's services without the need for downloading a dedicated desktop or mobile app. The web application is designed to offer the full suite of Spotify's features, from browsing and streaming music to crafting playlists and following other users. Given its importance and widespread use, it's vital that the web application performs flawlessly, providing the quality experience users have come to expect from Spotify.

To assure the quality of the Spotify web application, a comprehensive testing process is conducted. This testing process, focusing on black-box testing techniques due to the unavailability of the source code, is the primary subject of this report. The report aims to ensure that the web application not only functions as expected but also provides a user-friendly, secure, and consistent experience across a variety of devices and operating systems.

2. Objectives and Cases

2.1 Testing Objectives

The primary objectives of the testing are as follows:

- To ensure that the Spotify (Web, Mobile, Desktop) adheres to all functional requirements as outlined in the software requirements specification. This includes checking the various functionalities such as user login, navigation, search, playback, playlist management, and user account management.
- To validate the user-friendliness and intuitive navigation of the software. The Spotify Web Application should be easy to use and navigate, even for a first-time user.
- To confirm that the software performs as expected under varying conditions and is free from defects. This covers aspects like load time, response time, and handling of potential errors.

2.2 Testing Approach

The testing approach will primarily involve black box testing, where the tester does not have any knowledge of the internal workings of the software. The tester views the software as a "black box", focusing solely on inputs and outputs without concerning themselves with internal behavior. The testing will be based on the functional requirements outlined in the software requirements specification.

2.3 Test Cases

Here is a breakdown of the test cases to be performed during the black box testing phase, and the corresponding test cases that we previously defined:

• **Functional Testing:** This type of testing verifies that each function of the Spotify Web Application operates in conformance with the requirement specification. This includes tests like "Test user login functionality", "Test search functionality", and "Test playback functionality".

• **Non-Functional Testing:** This involves testing the non-functional aspects such as performance, usability, reliability. Test cases like "Test application load time" fall under this category.

• **Usability Testing:** The Spotify Web Application should be user-friendly. Test cases like "Test navigation through the application" and "Test the ease of playlist creation and editing" fall under this category.

• **Compatibility Testing:** This ensures the software can run in different environments, browsers, and devices. "Test application in different web browsers" is an example.

• **Equivalence Partitioning:** It is a software testing technique that divides the input data into partitions of equivalent data from which test cases can be derived. For example, testing the search functionality with valid and invalid inputs.

• **Boundary Value Analysis:** This is a method for designing test cases that focus on the boundary or limit conditions. "Test the system behavior when trying to play an empty playlist" is one such case.

• **Decision Table Testing:** This testing is done to test the system behavior for different input combinations. It is a good way to deal with a combination of inputs, which produces different results. A decision table for "playlist creation and editing options based on user type (Free, Premium)" can be created and tested.

• **Performance Testing:** This testing is done to check the speed, response time, reliability, resource usage, scalability of a software program under their workload. "Test the response time of the search functionality", "Test application load time" are examples.

2.4 USE CASES

Use Case: Interacting with the Spotify Web
Application Title: Interacting with the Spotify Web
Application **Actors:** User, Spotify Web Application
Preconditions: The user is logged into their Spotify account
Normal Flow of Events:

1. The user navigates to different pages/windows, such as home, search, and library.
2. The user searches for songs, albums, artists, and playlists using the search functionality.
3. The user plays, pauses, skips, and seeks within tracks in the playback functionality.
4. The user creates, edits, and deletes playlists, as well as adds and removes tracks from playlists.
5. The user updates their account information in the user account management section.
6. The user interacts with additional features, such as following artists, accessing "Barış Manço " playlists, viewing "Recently Played" items, managing the playback queue, and exploring "Top Charts."
7. The user logs out of their Spotify account.

Alternate Flows:

1. **2A1:** The user searches for a podcast instead of a song, album, artist, or playlist.
    1. The user enters a podcast-related query in the search bar.
    2. The system displays podcast results related to the query.
    3. The user clicks on a podcast result.
    4. The use case continues at step 3 with podcast playback functionality.
2. **3A1:** The user adjusts the volume during playback.
    1. The user drags the volume slider to a specific level.
    2. The system updates the volume according to the user's preference.
    3. The use case continues at step 4.
3. **4A1:** The user rearranges the order of tracks within a playlist.
    1. The user drags a track to a new position within the playlist.
    2. The system updates the order of tracks in the playlist.
    3. The use case continues at step 5.
4. **5A1:** The user resets their password in the user account management section.
    1. The user navigates to the account settings page.
    2. The user clicks on the "Reset Password" button.
    3. The system sends a password reset email to the user's registered email address.
    4. The user clicks on the password reset link in the email and sets a new password.
    5. The use case continues at step 6.
5. **6A1:** The user adds a song to the playback queue from the "Top Charts" section.
    1. The user navigates to the "Top Charts" section.
    2. The user clicks on the "Add to Queue" button for a top-charting song.
    3. The system adds the song to the user's playback queue.
    4. The use case continues at step 7.

This use case provides a comprehensive overview of a user's interaction with the Spotify Web application, including searching, playback, playlist management, user account management, and various additional features. The alternate flows cover scenarios such as podcast searching, volume adjustments, rearranging playlist tracks, password resetting, and adding songs to the playback queue from the "Top Charts" section.

Use Case: Discovering and Following New Music on the Spotify Web
Application Title: Discovering and Following New Music on the Spotify Web
Application **Actors:** User, Spotify Web Application
Preconditions: The user is logged into their Spotify account

Normal Flow of Events:

1. The user navigates to the "Browse" section of the Spotify Web Application.
2. The user explores various featured categories and playlists.
3. The user clicks on a category or playlist that interests them.
4. The user listens to a few songs from the chosen category or playlist.
5. The user discovers a new song, artist, or album they like.
6. The user follows the artist or saves the song or album to their library.
7. The user navigates to the "Radio" section and starts a radio station based on the new song, artist, or album.
8. The user listens to the radio station and discovers more new music.
9. The user saves additional songs, follows more artists, or adds albums to their library based on the radio station's recommendations.
10. The user logs out of their Spotify account.

Alternate Flows:

1. **2A1:** The user discovers new music through the "Genres & Moods" section.
    1. The user navigates to the "Genres & Moods" section.
    2. The user explores various genres and mood categories.
    3. The use case continues at step 3.
2. **4A1:** The user discovers new music through the "Top Charts" section.
    1. The user navigates to the "Top Charts" section.
    2. The user clicks on a top-charting song, album, or playlist.
    3. The use case continues at step 5.
3. **6A1:** The user shares the newly discovered music with a friend.
    1. The user clicks the "Share" button for the new song, artist, or album.
    2. The user chooses a sharing method (e.g., social media, email, or direct message).
    3. The user shares the music with their friend.
    4. The use case continues at step 7.
4. **8A1:** The user adds a song from the radio station to a playlist.
    1. The user clicks the "Add to Playlist" button for the song.
    2. The user chooses an existing playlist or creates a new one to add the song to.
    3. The system adds the song to the chosen playlist.
    4. The use case continues at step 9.
5. **9A1:** The user views the "Recently Played" section to rediscover previously played music.
    1. The user navigates to the "Recently Played" section.
    2. The user clicks on a recently played song, album, or playlist.
    3. The use case continues at step 5 with the user discovering and following new music.

This use case provides a comprehensive overview of a user's journey to discover and follow new music on the Spotify Web application, including browsing featured categories and playlists, listening to radio stations, and saving songs, following artists, or adding albums to their library. The alternate flows cover scenarios such as discovering new music through the "Genres & Moods" and "Top Charts" sections, sharing newly discovered music with a friend, adding a song from a radio station to a playlist, and rediscovering previously played music in the "Recently Played" section.

Use Case: Daily Use of the Spotify Web Application for a Casual User Title: Daily Use of the Spotify Web Application for a Casual User **Actors:** User, Spotify Web Application
Preconditions: The user is logged into their Spotify account

Normal Flow of Events:

1. The user navigates to the Spotify Web Application's homepage.
2. The user explores their personalized "Made For You" playlists, including "Daily Mixes" and "Discover Weekly."
3. The user selects a "Daily Mix" playlist based on their listening preferences.
4. The user listens to the curated playlist while engaging in their daily activities (e.g., working, studying, or exercising).
5. The user "likes" a song they enjoy by clicking the "heart" button, which saves the song to their library.
6. The user adds a song they particularly enjoy to one of their existing playlists or creates a new playlist.
7. The user shares a song, album, or playlist with a friend via social media, email, or direct message.
8. The user explores the "Recently Played" section to revisit previously played music.
9. The user logs out of their Spotify account.

Alternate Flows:

1. **3A1:** The user chooses to listen to their "Discover Weekly" playlist instead of a "Daily Mix."
    1. The user selects their personalized "Discover Weekly" playlist.
    2. The use case continues at step 4.
2. **4A1:** The user enables the "Private Session" mode for their listening session.
    1. The user navigates to their account settings.
    2. The user enables the "Private Session" option.
    3. The user's listening activity is now hidden from their public profile and friend activity.
    4. The use case continues at step 5.
3. **6A1:** The user removes a song from a playlist.
    1. The user navigates to the playlist containing the song they want to remove.
    2. The user clicks the "Remove from this Playlist" option for the specific song.
    3. The system removes the song from the playlist.
    4. The use case continues at step 7.
4. **8A1:** The user browses the "Top Charts" section to explore popular music.
    1. The user navigates to the "Top Charts" section.
    2. The user explores popular songs, albums, or playlists.
    3. The use case continues at step 9.
5. **9A1:** The user follows a new podcast to listen to later.
    1. The user searches for a podcast or navigates to the podcast section.
    2. The user clicks the "Follow" button for a podcast they want to listen to later.
    3. The system adds the podcast to the user's library.
    4. The use case ends after the user logs out.

This use case provides an overview of a casual user's daily interaction with the Spotify Web Application, including exploring personalized playlists, listening to music, liking and adding songs to playlists, sharing music, and revisiting previously played tracks. The alternate flows cover scenarios such as listening to a "Discover Weekly" playlist, enabling the "Private Session" mode, removing a song from a playlist, browsing the "Top Charts" section, and following a new podcast.

Use Case 1: User Log In

1. The user opens the Spotify web application.
2. The system shows the login page.
3. The user enters his/her email and password.
4. The system verifies the login credentials and logs in the user.

Alternate flows: 4A1: The user enters incorrect login credentials.

- The system returns a warning about the incorrect login information and asks to enter the correct one.
- The user enters the correct login information.

4A2: The user enters incorrect login credentials for 3 times.

- The system returns the information about the situation.
- The login access is temporarily blocked.

Use Case 2: User Sign Up

1. The user opens the Spotify web application.
2. The system shows the sign-up page.
3. The user enters his/her email, password, profile name, date of birth, and gender.
4. The system verifies the sign-up information and creates a new user account.

Alternate flows: 4A1: The user enters incorrect sign-up information.

- The system returns a warning about the incorrect sign-up information and asks to enter the correct one.
- The user enters the correct sign-up information.

Use Case 3: User Searches for a Song

1. The user opens the Spotify web application.
2. The system shows the search bar.
3. The user enters the song name.
4. The system shows the search results.

Alternate flows: 4A1: The user enters incorrect song name.

- The system shows no results.

Use Case 4: User Plays a Song

1. The user opens the Spotify web application.
2. The system shows the song list.
3. The user chooses a song and presses the play button.
4. The system plays the song.

Alternate flows: 4A1: The user chooses a song that is not available in his/her region.

- The system shows a warning about the song unavailability.

Use Case 5: User Adds a Song to the Playlist

1. The user opens the Spotify web application.
2. The system shows the song list.
3. The user chooses a song and presses the "add to playlist" button.
4. The system adds the song to the user's playlist.

Alternate flows: 4A1: The user chooses a song that is already in the playlist.

- The system shows a warning about the song duplication.

Test Case ID	Use Case ID	Scenario	Test Input	Expected Output
TC1	1	User logs in with valid credentials	Email: test@example.com, Password: Test@1234	User is logged in successfully
TC2	1	User attempts to log in with an invalid email format	Email: test@example, Password: Test@1234	Error message about invalid email format is displayed
TC3	2	New user signs up with valid details	Email: newuser@example.com, Password: NewUser@1234, Profile Name: New User, Date of Birth: 01/01/2000, Gender: Male	New user account is created successfully
TC4	3	User searches for a song with a valid song name	Song Name: Shape of You	Search results include "Shape of You" by Ed Sheeran
TC5	4	User attempts to play a valid song	Song: Shape of You	"Shape of You" by Ed Sheeran is played
TC6	5	User adds a valid song to the playlist	Song: Shape of You	"Shape of You" by Ed Sheeran is added to the user's playlist
TC7	2	User attempts to sign up with an invalid email format	Email: newuser@example, Password: NewUser@1234, Profile Name: New User, Date of Birth: 01/01/2000, Gender: Male	Error message about invalid email format is displayed
TC8	3	User searches for a song with an invalid/nonexistent song name	Song Name: Some Nonexistent Song	No search results are displayed
TC9	4	User attempts to play a song that's not available in their region	Song: A song not available in user's region	Warning about song unavailability
TC10	5	User attempts to add a song that's already in their playlist	Song: Shape of You	Warning about song duplication in playlist
TC11	1	User attempts to log in with incorrect password	Email: test@example.com, Password: IncorrectPassword	Error message about incorrect login credentials
TC12	2	User attempts to sign up with an email that's already in use	Email: existinguser@example.com, Password: ExistingUser@1234, Profile Name: Existing User, Date of Birth: 01/01/1990, Gender: Female	Warning about existing user information
TC13	1	User attempts to log in with an account that's been blocked	Email: blockeduser@example.com, Password: BlockedUser@1234	Information about blocked login access
TC14	2	User attempts to sign up as underage (under 13)	Email: underageuser@example.com, Password: UnderageUser@1234, Profile Name: Underage User, Date of Birth: 01/01/2015, Gender: Female	Warning about underage user information
TC15	5	User attempts to add a song that's not available in their region to their playlist	Song: A song not available in user's region	Warning about song unavailability
TC16	2	User attempts to sign up with a password that doesn't meet the complexity requirements	Email: weakpassworduser@example.com, Password: weak, Profile Name: Weak Password User, Date of Birth: 01/01/2000, Gender: Male	Warning about weak password, needs to meet complexity requirements
TC17	2	User attempts to sign up with a profile name that exceeds the maximum character limit	Email: longnameuser@example.com, Password: LongNameUser@1234, Profile Name: A very long profile name that exceeds the max character limit, Date of Birth: 01/01/2000, Gender: Male	Warning about profile name length, can't exceed maximum character limit
TC18	3	User searches for a song using special characters	Song Name: @#$$%	Appropriate search results based on special character input, likely no results
TC19	4	User attempts to play a song that's been removed from Spotify's library	Song: A song that's been removed	Warning about song unavailability, has been removed from Spotify's library
TC20	5	User attempts to add a song to a playlist that's been deleted	Song: Shape of You, Playlist: Deleted Playlist	Warning about playlist unavailability, has been deleted

3. Functional Testing

A. Testing Report for sign up action:

- Checkboxes are not necessary in both situation.
- The password can have at least 8 characters and a maximum of 100 characters.
- The password should not be too simple, values such as "12345678" are not accepted.
- User must be over 13 years old.
- Account cannot be opened twice with the same mail.
- "What should we call you?" input can be the same with the other users, special characters can be used. It has to be at least 1 character, there is no character limit, but a maximum of 30 characters are displayed.
- "What's your gender?" One of the buttons here must be selected.
- After selecting Facebook login, if "Not now" is selected, "Something went wrong. Try again."

Method	E-mail Input	Password Input	Profile Name Input	Date of Birth	Gender	C1	C2	Output	Comment	Program Error
Email	validmail	validpw	specialcharactersname	01.01.1980	Male	+	+	Account created successfully.	Special characters in profile name	No
Email	validmail	validpw	validname	01.01.1980	Non-binary	+	+	Account created successfully.	Non-binary gender selected	No
Email	validmail	validpw	validname	13.05.2010	Female	-	+	Sorry, you don't meet Spotify's age requirements.	User exactly 13 years old today	No
Email	validmail	validpw	validname	14.05.2010	Male	+	-	Sorry, you don't meet Spotify's age requirements.	User will be 13 years old tomorrow	Yes
Email	validmail	validpw	validname	01.01.1910	Female	+	+	Account created successfully.	Very old date of birth	No
Email	validmail	validpw	validname	01.01.1905	Male	-	-	Sorry, you don't meet Spotify's age requirements.	Date of birth too old	Yes
Email	validmail	nocharacterspw	validname	01.01.1980	Female	+	+	Your password is too weak. Set a stronger one.	Password with no characters	Yes
Email	validmail	onlyletterspw	validname	01.01.1980	Male	-	-	Account created successfully.	Password with only letters	No
Email	validmail	onlynumberspw	validname	01.01.1980	Male	+	+	Account created successfully.	Password with only numbers	No
Email	validmail	specialcharacterspw	validname	01.01.1980	Male	-	-	Account created successfully.	Password with special characters	No
Email	empty	validpw	validname	01.01.1980	Male	+	+	Please enter your email.	Empty email field	Yes
Google	N/A	N/A	N/A	N/A	N/A	+	+	Account created successfully.	Google login with news and data sharing	No
Google	N/A	N/A	N/A	N/A	N/A	-	-	Account created successfully.	Google login without news and data sharing	No
Facebook	N/A	N/A	N/A	N/A	N/A	+	-	Account created successfully.	Facebook login with news but no data sharing	No
Facebook	N/A	N/A	N/A	N/A	N/A	-	+	Account created successfully.	Facebook login with data sharing but no news	No
Email	validmail	validpw	validname	01.01.1980	Prefer not to say	-	+	Account created successfully.	User prefers not to disclose gender	No
Email	validmail	validpw	validname	01.01.1980	N/A	+	-	Please select a gender.	No gender selection	Yes
Email	validmail	validpw	31charsname	01.01.1980	Male	+	-	Account created successfully.	Profile name exceeds 30 displayed characters, but still accepted	No
Email	validmail	validpw	whitespace	01.01.1980	Female	-	-	Please enter a profile name.	Profile name with only spaces	Yes
Email	duplicatevalidmail	validpw	validname	01.01.1980	Male	+	+	Account already exists with this email.	Duplicate email	Yes
Email	validmail	101charspw	validname	01.01.1980	Male	-	+	Your password is too long.	Password exceeds 100 characters	Yes
Email	validmail	spacespw	validname	01.01.1980	Male	-	-	Your password is too weak. Set a stronger one.	Password with only spaces	Yes

Conditions (input):

- E-mail field is filled.
- E-mail is valid.
- Password field is filled.
- Password is valid (8-100 characters, not too easy).
- Profile name is filled.
- Profile name is valid (1-30 characters).
- Date of birth is filled.
- User's age is valid (13-101 years old).
- Gender is chosen.
- User signs up with Facebook.
- User signs up with Google.

And the resulting actions (output) would be:

- Sign up successfully.
- Show "e-mail field is empty" error.
- Show "invalid e-mail" error.
- Show "password field is empty" error.
- Show "invalid password" error.
- Show "profile name field is empty" error.
- Show "invalid profile name" error.
- Show "date of birth field is empty" error.
- Show "invalid age" error.
- Show "gender not chosen" error.
- Sign up successfully with Facebook.
- Sign up successfully with Google.

Please note that, due to the complexity and the number of conditions, the decision table will have numerous columns (2^11 = 2048 to be precise) if all possible combinations are to be considered. This can become impractical in many cases, and as such, it's often recommended to use techniques such as equivalence partitioning or boundary value analysis to reduce the number of test cases while still maintaining good test coverage.

Due to the high number of conditions, a full decision table would be impractically large. However, I'll provide a simplified decision table, focusing on the main elements. Then, I'll proceed with equivalence partitioning and boundary value analysis for the various input fields.

Conditions \ Actions	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**
1. E-mail field is filled & valid	T	F	T	T	T	T
2. Password field is filled & valid	T	T	F	T	T	T
3. Profile name is filled & valid	T	T	T	F	T	T
4. Date of birth is filled & valid	T	T	T	T	F	T
5. Gender is chosen	T	T	T	T	T	F
6. User signs up with Facebook	F	F	F	F	F	F
7. User signs up with Google	F	F	F	F	F	F
Actions						
1. Sign up successfully	T	F	F	F	F	F
2. Show "e-mail field is empty or invalid" error	F	T	F	F	F	F
3. Show "password field is empty or invalid" error	F	F	T	F	F	F
4. Show "profile name field is empty or invalid" error	F	F	F	T	F	F
5. Show "date of birth field is empty or invalid" error	F	F	F	F	T	F
6. Show "gender not chosen" error	F	F	F	F	F	T

T = True, F = False

Equivalence Partitioning

Equivalence partitioning involves dividing the input data of a software unit into partitions of equivalent data from which test cases can be derived.

E-mail:
Valid Partition: A correctly formatted email address (e.g., test@example.com).
Invalid Partition: Incorrectly formatted email addresses (e.g., test@example, test@.com, test).

Password:
Valid Partition: Passwords containing 8-100 characters, not too easy.
Invalid Partition: Passwords containing less than 8 characters or more than 100 characters, or too easy passwords.

Profile Name:
Valid Partition: Profile names containing 1-30 characters.
Invalid Partition: Profile names containing no characters or more than 30 characters.

Date of Birth:
Valid Partition: Dates of birth that would make the user 13-101 years old.
Invalid Partition: Dates of birth that would make the user less than 13 years old or more than 101 years old.

Gender:
Valid Partition: Choosing any of the provided gender options.
Invalid Partition: Not choosing any gender.

Equivalence Class	Test Input	Expected Output
E1. Valid E-mail	test@example.com	Success Message
E2. Invalid E-mail	test@example	Error Message
E3. Valid Password	Abcd@1234	Success Message
E4. Invalid Password	abcd	Error Message
E5. Valid Profile Name	testprofile	Success Message
E6. Invalid Profile Name		Error Message
E7. Valid Age	Date of Birth: 05/14/2000 (23 years old)	Success Message
E8. Invalid Age	Date of Birth: 05/15/2015 (8 years old)	Error Message
E9. Valid Gender	Male	Success Message
E10. Invalid Gender	None selected	Error Message

Boundary Value Analysis

Boundary Value Analysis is a software testing technique that involves determining the boundary values of valid and invalid partitions.

E-mail:
Lower Boundary: The smallest valid email (a@b.c).
Upper Boundary: Not applicable in this context.

Password:
Lower Boundary: 8 characters.
Upper Boundary: 100 characters.
Profile Name:
Lower Boundary: 1 character.
Upper Boundary: 30 characters. (It can be more but it only displays maximum 30 characters)

Date of Birth:
Lower Boundary: A date that makes the user 13 years old (i.e., 05/14/2010 if the current date is 05/13/2023).
Upper Boundary: A date that makes the user 101 years old (i.e., 05/14/1922 if the current date is 05/13/2023).

Gender:
Must select one.
Lower Boundary: Choosing the first option (male).
Upper Boundary: Choosing the last option (prefer not to say).

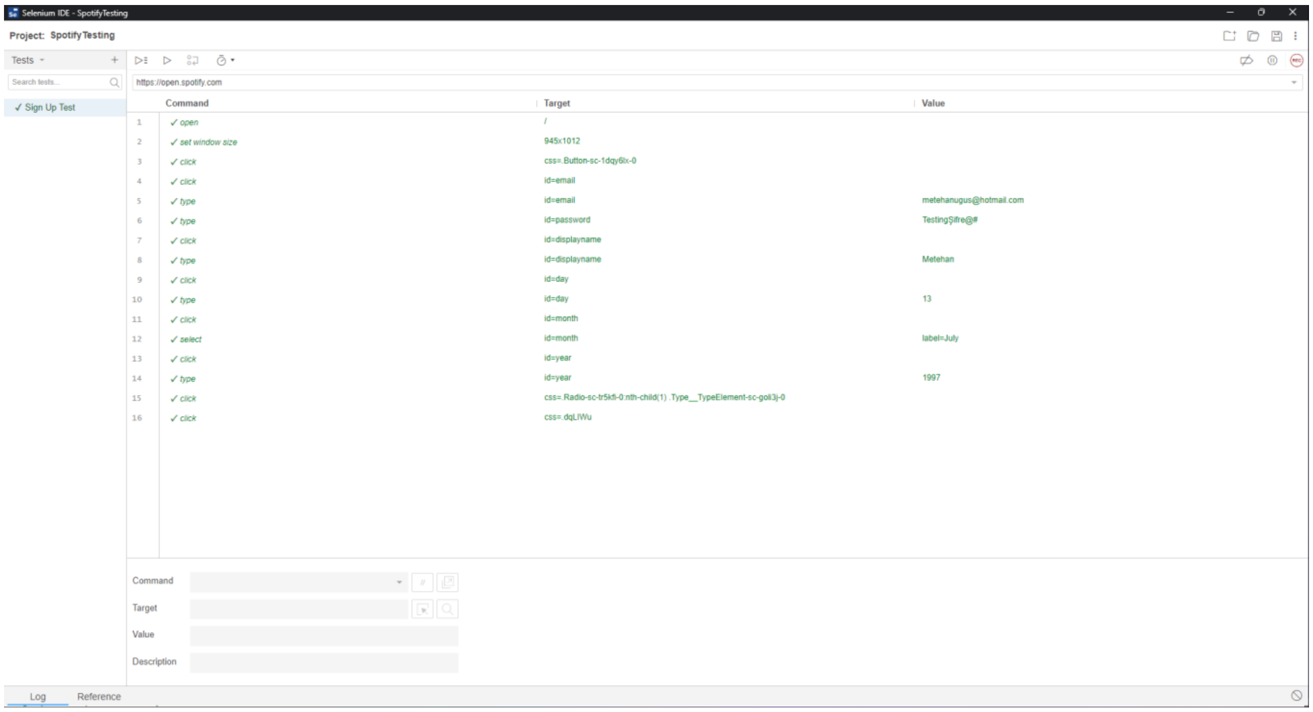
Equivalence Class	Test Input	Expected Output
E1. Valid E-mail	a@b.c	Success Message
E2. Invalid E-mail	ab.c	Error Message
E3. Valid Password (Lower Boundary)	Abcd@123	Success Message
E4. Valid Password (Upper Boundary)	A string of 100 characters	Success Message
E5. Invalid Password	A string of 101 characters	Error Message
E6. Valid Profile Name (Lower Boundary)	a	Success Message
E7. Valid Profile Name (Upper Boundary)	A string of 30 characters	Success Message
E8. Valid Age (Lower Boundary)	Date of Birth: 05/14/2010 (13 years old)	Success Message
E9. Valid Age (Upper Boundary)	Date of Birth: 05/14/1922 (101 years old)	Success Message
E10. Invalid Age	Date of Birth: 05/15/1921 (102 years old)	Error Message
E11. Valid Gender (First Option)	Male	Success Message
E12. Valid Gender (Last Option)	Prefer not to say	Success Message

Please note that boundary value analysis might not apply to all input fields. For instance, for gender, any valid selection is acceptable, and there's no meaningful boundary as it's not a numerical or date range.

For test cases, you should focus on both valid values (from equivalence partitioning) and edge cases (from boundary value analysis). Together, they help ensure a comprehensive coverage of possible inputs.

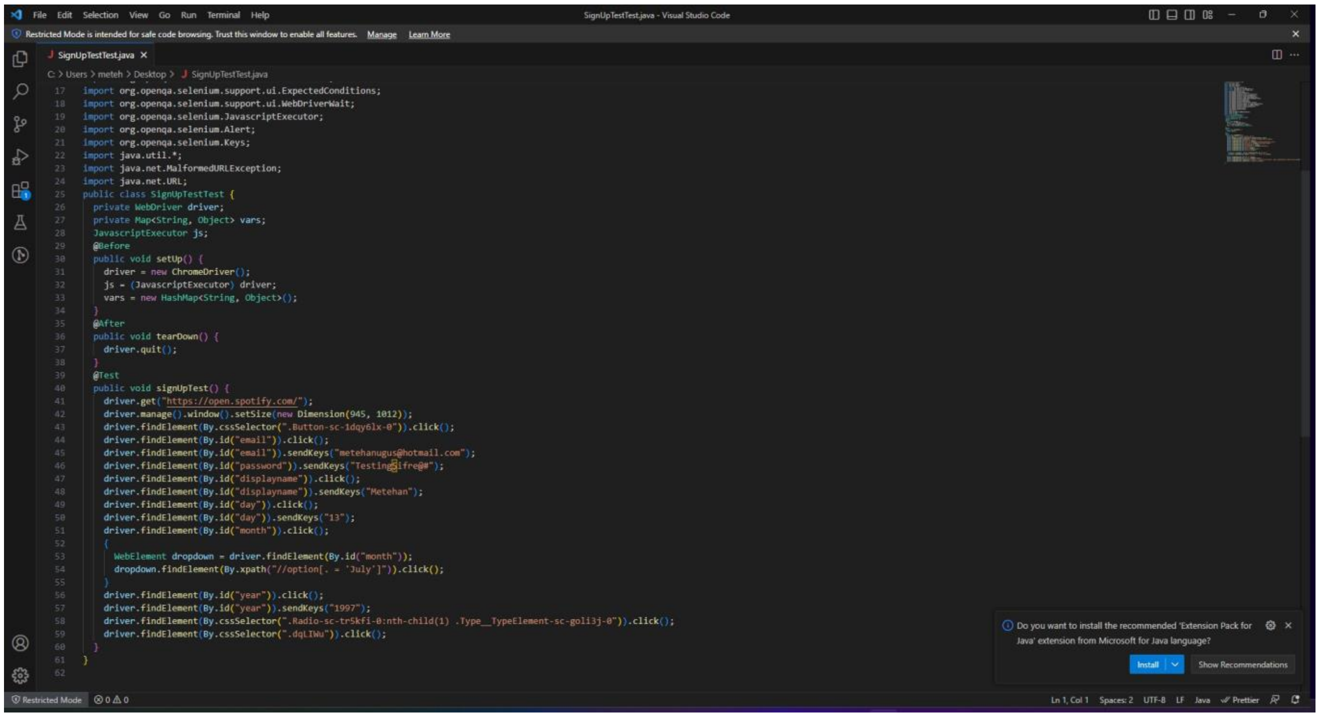
Automation With Selenium

Larger and more understandable images are shared with source codes in the project folder.



https://open.spotify.com

Command	Target	Value
✓ open	/	
✓ set window size	945x1012	
✓ click	css=.Button-sc-1dqy6lx-0	
✓ click	id=email	
✓ type	id=email	metehanugus@hotmail.com
✓ type	id=password	TestingŞifre@#
✓ click	id=displayname	
✓ type	id=displayname	Metehan
✓ click	id=day	
✓ type	id=day	13
✓ click	id=month	
✓ select	id=month	label=July
✓ click	id=year	
✓ type	id=year	1997
✓ click	css=.Radio-sc-tr5kfi-0:nth-child(1) .Type__TypeElement-sc-goli3j-0	
✓ click	css=.dqLIWu	


File Edit Selection View Go Run Terminal Help
SignUpTestTest.java - Visual Studio Code
SignUpTestTest.java
C: > Users > meteh > Desktop > SignUpTestTest.java
Install
Show Recommendations
Restricted Mode
Ln 1, Col 1 Spaces: 2 UTF-8 LF Java Prettier

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

```java
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class SignUpTestTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
  @After
  public void tearDown() {
    driver.quit();
  }
  @Test
  public void signUpTest() {
    driver.get("https://open.spotify.com/");
    driver.manage().window().setSize(new Dimension(945, 1012));
    driver.findElement(By.cssSelector(".Button-sc-1dqy6lx-0")).click();
    driver.findElement(By.id("email")).click();
    driver.findElement(By.id("email")).sendKeys("metehanugus@hotmail.com");
    driver.findElement(By.id("password")).sendKeys("TestingŞifre@#");
    driver.findElement(By.id("displayname")).click();
    driver.findElement(By.id("displayname")).sendKeys("Metehan");
    driver.findElement(By.id("day")).click();
    driver.findElement(By.id("day")).sendKeys("13");
    driver.findElement(By.id("month")).click();
    {
      WebElement dropdown = driver.findElement(By.id("month"));
      dropdown.findElement(By.xpath("//option[. = 'July']")).click();
    }
    driver.findElement(By.id("year")).click();
    driver.findElement(By.id("year")).sendKeys("1997");
    driver.findElement(By.cssSelector(".Radio-sc-tr5kfi-0:nth-child(1) .Type__TypeElement-sc-goli3j-0")).click();
    driver.findElement(By.cssSelector(".dqLIWu")).click();
  }
}
```

Do you want to install the recommended 'Extension Pack for Java' extension from Microsoft for Java language?

B.Testing Report for log in action:

For the login page of the Spotify web application, the decision table will take into account several different inputs and scenarios. Here is a broad outline of what the decision table will look like.
The table would contain inputs such as:

1. Login Method: Email/Username, Google, Facebook, or Apple
2. Email/Username Input: Valid email/username, invalid email/username, blank email/username
3. Password Input: Valid password, invalid password, blank password
4. Remember Me: Selected, not selected

The scenarios we will consider will include:

1. Valid login via Email/Username
2. Invalid Email/Username with valid password
3. Valid Email/Username with invalid password
4. Login via Google
5. Login via Facebook
6. Login via Apple
7. Remember Me selected
8. Remember Me not selected

Possible outputs are:

1. Login successful
2. Incorrect password
3. Account not found
4. Password field is empty
5. Email field is empty

Let's consider each combination of these variables. With 4 login methods, 3 types of email/username input, 3 types of password input, and 2 states of "Remember Me", we have a total of 433*2 = 72 possible scenarios.

However, for login methods other than Email/Username, the email/username input, password input, and "Remember Me" selection are not applicable. So, the actual number of unique scenarios is significantly less. Creating a comprehensive decision table with all these scenarios would be quite large and may be unwieldy. Instead, it might be more useful to create several smaller tables, each focusing on a different aspect of the login process. For example, one table could focus on the different login methods, another on the scenarios involving the Email/Username method, and so on.

This way, you can thoroughly test each part of the login process without getting overwhelmed by the sheer number of possible scenarios.

Email/Username Input	Password Input	Remember Me	Login Successful	Error Message
Valid	Valid	Selected	Yes	No
Valid	Valid	Not Selected	Yes	No
Valid	Invalid	Selected	No	"Incorrect password"
Valid	Invalid	Not Selected	No	"Incorrect password"
Valid	Empty	Selected	No	"Password field is empty"
Valid	Empty	Not Selected	No	"Password field is empty"
Invalid	Valid	Selected	No	"Account not found"
Invalid	Valid	Not Selected	No	"Account not found"
Invalid	Invalid	Selected	No	"Account not found"
Invalid	Invalid	Not Selected	No	"Account not found"
Invalid	Empty	Selected	No	"Account not found"
Invalid	Empty	Not Selected	No	"Account not found"
Empty	Valid	Selected	No	"Email field is empty"
Empty	Valid	Not Selected	No	"Email field is empty"
Empty	Invalid	Selected	No	"Email field is empty"
Empty	Invalid	Not Selected	No	"Email field is empty"
Empty	Empty	Selected	No	"Email field is empty"
Empty	Empty	Not Selected	No	"Email field is empty"

This table only considers login via Email/Username. For login via Google, Facebook, or Apple, the login process will depend on the user's account status with these services, so separate decision tables may be necessary. The "Remember Me" option also doesn't affect the success or failure of the login process, but it can be important for usability testing.

No	Login Method	Email/Username	Password	Remember Me	Login Successful	Incorrect Password	Account Not Found	Password Field Empty	Email Field Empty	Comments
1	Email/Username	Valid	Valid	Y	Y	N	N	N	N	All details are correct, remember me selected.
2	Email/Username	Invalid	Valid	Y	N	N	Y	N	N	Wrong Email/Username entered, rest correct.
3	Email/Username	Valid	Invalid	Y	N	Y	N	N	N	Password incorrect, rest details correct.
4	Email/Username	Blank	Valid	Y	N	N	N	N	Y	Email field left empty, rest entered correctly.
5	Email/Username	Valid	Blank	Y	N	N	N	Y	N	Password field left empty, rest entered correctly.
6	Google	N/A	N/A	N	Y	N/A	N/A	N/A	N/A	Login via Google, no need for email or password.
7	Facebook	N/A	N/A	N	Y	N/A	N/A	N/A	N/A	Login via Facebook, no need for email or password.
8	Apple	N/A	N/A	N	Y	N/A	N/A	N/A	N/A	Login via Apple, no need for email or password.

Equivalence Partitioning:

For this partitioning, we're considering both Email/Username and Password fields. For Emails, we're considering valid and invalid formats, and for Passwords, we're also considering weak (too easy) and strong.

Equivalence Class	Test Input	Expected Output
Valid Email	testuser@gmail.com	Valid Input
Invalid Email	testuser@gmail	Invalid Input - Incorrect Format
Empty Email	""	Invalid Input - Empty Field
Excessive length Email	"a"*101 + "@gmail.com"	Invalid Input - Exceeds Maximum Length
Valid Username	"testuser"	Valid Input
Invalid Username	""	Invalid Input - Empty Field
Excessive length Username	"a"*101	Invalid Input - Exceeds Maximum Length
Valid strong Password	"Password123$"	Valid Input
Invalid weak Password	"password"	Invalid Input - Weak Password
Empty Password	""	Invalid Input - Empty Field
Excessive length Password	"a"*101	Invalid Input - Exceeds Maximum Length

Boundary Value Analysis:

For boundary value analysis, we're looking at the edges of the input fields length, considering the minimum and maximum valid lengths and just below and above these lengths.

Boundary Value Class	Test Input	Expected Output
Minimum valid Email length (3 chars)	"a@b"	Valid Input
Just below minimum Email length (2 chars)	"ab"	Invalid Input - Incorrect Format
Just above maximum Email length (101 chars)	"a"*97 + "@b.c"	Invalid Input - Exceeds Maximum Length
Maximum valid Email length (100 chars)	"a"*96 + "@b.c"	Valid Input
Minimum valid Username length (1 char)	"a"	Valid Input
Just below minimum Username length (0 char)	""	Invalid Input - Empty Field
Just above maximum Username length (101 chars)	"a"*101	Invalid Input - Exceeds Maximum Length
Maximum valid Username length (100 chars)	"a"*100	Valid Input
Minimum valid Password length (8 chars)	"Passw0r$"	Valid Input
Just below minimum Password length (7 chars)	"Passw0r"	Invalid Input - Below Minimum Length
Just above maximum Password length (101 chars)	"a"*100 + "$"	Invalid Input - Exceeds Maximum Length
Maximum valid Password length (100 chars)	"a"*99 + "$"	Valid Input

B. Testing report for search feature

Equivalence Partitioning and Boundary Value Analysis:

Equivalence partitioning is a black-box testing technique that divides the input data of a system into different partitions, where each partition is considered as equivalent. In this technique, test cases are designed to validate the behavior of the system for each partition.
For the search feature of Spotify, we can divide the input data into two partitions:
Valid Partition: This partition includes all the valid inputs that the search feature accepts. It includes alphabets, symbols, special characters, whitespace, and different languages like Turkish, Chinese, etc. and characters like emoji, etc.
Invalid Partition: This partition includes all the invalid inputs that the search feature does not accept. It includes input exceeding the maximum limit of 500 characters.
Based on the requirements and functionality of the search feature, the following equivalence classes can be identified:
Valid search results: artist names, album names, and song names
Invalid search results: non-existent names, incorrect spellings, irrelevant search results

Decision Table ;

Decision	Conditions	Actions
Search query length	0 ≤ length ≤ 500	Return relevant search results
Search query validity	Valid search query	Display search results
	Invalid search query	Display relevant results
Search result validity	Valid search results	Display relevant results

	Invalid search results	Display relevant results
Search result quantity	0 ≤ quantity ≤ 2000	Display relevant search results up to given quantity
	Quantity > 2000	Display relevant results

Use Case Test Table:

Here is a use case test table for the search feature of Spotify:

Test Case ID	**Test Case Description**	**Input Data**	**Expected Output**
TC-01	Search by artist name (valid input)	Adele	List of songs/albums/artists/playlists/profiles with the keyword "Adele"
TC-02	Search by album name (valid input)	1989	List of songs/albums/artists with the keyword "1989"
TC-03	Search by song name (valid input)	Shape of You	List of songs/albums/artists/playlists/profiles with the keyword "Shape of You"
TC-04	Search with whitespace (valid input)	The Beatles	List of songs/albums/artists/playlists/profiles with the keyword "The Beatles"
TC-05	Search with special characters (valid input)	#1	List of songs/albums/artists/playlists/profiles with the keyword "#1"
TC-06	Search with Turkish characters (valid input)	Şarkılar	List of songs/albums/artists/playlists/profiles with the keyword "Şarkılar"

TC-07	Search with Chinese characters (valid input)	中国	List of songs/albums/artists/playlists/profiles with the keyword "中国"
TC-08	Search with invalid characters (invalid input)	😃😃	List of songs/albums/artists/playlists/profiles with the keyword "😃😃"
TC-09	Search with input exceeding maximum limit	A*500	Error message "Input + doesnt found for that"
TC-10	Search with input just below the minimum limit	-1	List of songs/albums/artists/playlists/profiles with the keyword “-1”
TC-11	Search with input just above the maximum limit	A*501	Error message "Input + doesnt found for that"

C. Testing Report for Playlist Management

Decision	Conditions
Create a new playlist	Playlist name includes alphabets, symbols, special characters, different languages, and characters like emoji. Playlist name is not empty and does not exceed 255 characters.
Add a song to a playlist	Playlist exists and has not reached the 10,000 song limit.Playlist does not exist or has reached the 10,000 song limit.If the song already includes the same song, ask the user to add the same song.
Remove a song from a playlist	Playlist exists and the song is in the playlist.Playlist does not exist or the song is not in the playlist.

Delete a playlist	Playlist exists. Playlist does not exist.
Add playlist to your library	Playlist created by the user.Playlist created by someone else.
Share playlist	Share with playlist link. Share on social media account.
Set private playlist	Playlist created by the user.Playlist created by someone else.

And here is a use case test table with valid and invalid values for each condition:

Use Case	Test Data	Expected Result
Create a new playlist	Playlist name: "My Playlist"	Playlist is created successfully.
Create a new playlist	Playlist name: ""	Error message: "Playlist name cannot be empty."
Create a new playlist	Playlist name: "Playlist with more than 255 characters.."	Error message: "Playlist name cannot exceed 255 characters."
Create a new playlist	Playlist name: "Playlist with Turkish characters: İıŞşĞğÜüÖöÇç"	Playlist is created successfully.
Create a new playlist	Playlist name: "Playlist with emoji: 😀😀😀😀😀😀"	Playlist is created successfully.
Add a song to a playlist	Playlist exists and has not reached the 10,000 song limit. Song: "Song Name"	Song is added to the playlist successfully.

Add a song to a playlist	Playlist does not exist. Song: "Song Name"	Error message: "Playlist does not exist."
Add a song to a playlist	Playlist exists and has reached the 10,000 song limit. Song: "Song Name"	Error message: "Playlist has reached the song limit."
Add a song to a playlist	Playlist exists and the song already includes the same song. User selects not to add the same song.	Song is not added to the playlist.
Add a song to a playlist	Playlist exists and the song already includes the same song. User selects to add the same song.	Song is added to the playlist again.
Remove a song from a playlist	Playlist exists and the song is in the playlist. Song: "Song Name"	Song is removed from the playlist successfully.
Remove a song from a playlist	Playlist does not exist. Song: "Song Name"	Error message: "Playlist does not exist."
Remove a song from a playlist	Playlist exists but the song is not in the playlist. Song: "Song Name"	Error message: "Song is not in the playlist."
Delete a playlist	Playlist exists.	Playlist is deleted successfully.
Delete a playlist	Playlist does not exist.	Error message: "Playlist does not exist."
Add playlist to your library	Playlist created by the user.	Playlist is added to the library successfully.

Boundary Value Analysis for Creating a New Playlist:

Test Case ID	Test Case Description	Input	Expected Output
BVA-CNP-1	Test for minimum length of playlist name	"A"	Playlist created successfully
BVA-CNP-2	Test for maximum length of playlist name	"Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam efficitur, ante eu tincidunt."	Playlist name exceeds the maximum length
BVA-CNP-3	Test for invalid characters in playlist name	"My Playlist#"	Playlist name contains invalid characters
BVA-CNP-4	Test for non-Latin characters in playlist name	"我的播放列表"	Playlist name contains non-Latin characters
BVA-CNP-5	Test for blank playlist name	""	Playlist name cannot be blank

Equivalence Partitioning for Adding a Song to a Playlist:

Test Case ID	Test Case Description	Input	Expected Output
EP-ASP-1	Test for adding a song to an existing playlist	Playlist exists, has not reached the 10,000 song limit	Song added successfully
EP-ASP-2	Test for adding a song to a non-existent playlist	Playlist does not exist	Error message displayed
EP-ASP-3	Test for adding a song to a playlist that has reached the 10,000 song limit	Playlist exists, has reached the 10,000 song limit	Error message displayed
EP-ASP-4	Test for adding a song that already exists in the playlist	Playlist exists, song already in playlist	User prompted to confirm adding duplicate song
EP-ASP-5	Test for not adding a song that already exists in the playlist	Playlist exists, song already in playlist, user declines to add duplicate	Song not added to playlist

Equivalence Partitioning for Removing a Song from a Playlist:

Test Case ID	Test Case Description	Input	Expected Output
EP-RSP-1	Test for removing a song from an existing playlist	Playlist exists, song is in playlist	Song removed successfully
EP-RSP-2	Test for removing a song from a non-existent playlist	Playlist does not exist	Error message displayed
EP-RSP-3	Test for removing a song that is not in the playlist	Playlist exists, song is not in playlist	Error message displayed

Boundary Value Analysis for Deleting a Playlist:

Test Case ID	Test Case Description	Input	Expected Output
BVA-DP-1	Test for deleting an existing playlist	Playlist exists	Playlist deleted successfully
BVA-DP-2	Test for deleting a non-existent playlist	Playlist does not exist	Error message displayed

Decision 4: Add playlist to your library

Test Case ID	Test Case Description	Test Data	Expected Results
PLM-TC4.1	Add a playlist created by the user to the library	Playlist created by the user	Playlist should be added to the library
PLM-TC4.2	Add a playlist not created by the user to the library	Playlist not created by the user	Playlist should be added to the library as a followed playlist
PLM-TC4.3	Try to add a playlist that already exists in the library	Playlist already in the library	Playlist should not be added again
PLM-TC4.4	Try to add a playlist with an invalid format or special characters in the name	Playlist name with invalid format or symbols	Error message should be displayed and playlist should not be added to the library

PLM-TC4.5	Try to add a playlist with a name exceeding the maximum character limit	Playlist name exceeding 255 characters	Error message should be displayed and playlist should not be added to the library

Decision 5: Share a playlist

Test Case ID	Test Case Description	Test Data	Expected Results
PLM-TC5.1	Share a playlist by copying the playlist link	Playlist link	Link should be copied to the clipboard
PLM-TC5.2	Share a playlist by posting on social media	Social media platform	Playlist should be shared on the selected social media platform
PLM-TC5.3	Try to share a playlist without selecting any social media platform	No platform selected	Error message should be displayed and playlist should not be shared
PLM-TC5.4	Try to share a playlist that does not exist	Non-existing playlist ID	Error message should be displayed and playlist should not be shared
PLM-TC5.5	Try to share a playlist that is set to private	Private playlist	Error message should be displayed and playlist should not be shared
PLM-TC5.6	Try to share a playlist with invalid characters or symbols in the name	Playlist name with invalid characters/symbols	Error message should be displayed and playlist should not be shared
PLM-TC5.7	Try to share a playlist with a name exceeding the maximum character limit	Playlist name exceeding 255 characters	Error message should be displayed and playlist should not be shared