

SOFTWARE VERIFICATION, VALIDATION AND TESTING

TESTING DOCUMENTATION

Spotify Automated Testing

Prepared by:
Zara Bahtanović

Proposed to:
Samed Jukić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

21.01.2023

Table of Contents

Table of Contents	2
1. Introduction	3
1.1. About the Project.....	3
1.2. Project Functionalities and Screenshots.....	3
2. Test Plan	6
2.1. Scope.....	6
2.2. Testing Environment and Tools.....	6
3. Test Execution	6
3.1. Sing Up	6
3.2. Login Functionality.....	12
3.3. Playback Functionality.....	15
3.4. Search and Browse Functionality	16
3.5. Localization.....	19
3.6. Accessibility	20
3.7. Radio Functionality.....	21
3.8. Profile Management.....	24
3.9. Podcast Functionality	26
3.10. Library Functionality	29
3.11. Playlist Functionality	29
4. Conclusion	34
4.1. Testing Summary	34
4.2. Final Thoughts	34

1. Introduction

1.1. About the Project

Spotify is a music streaming service that allows users to listen to millions of songs, albums, and playlists on various devices. Founded in 2006, it has over 456 million monthly active users including 196 million paying subscribers. It also allows users to search for music based on artist, album, or genre, to create, edit, and share playlists, follow other users, and discover new music based on their listening habits. The application is available on multiple platforms including web, iOS, and Android. The homepage link is <https://www.spotify.com/>.

1.2. Project Functionalities and Screenshots

The main features of the Spotify project include:

- Music streaming: Users can listen to a wide variety of music from various genres and artists.
- Playlists: Users can create, edit, and share playlists with other users.
- Search: Users can search for music based on artist, album, or genre.
- Radio: Users can listen to radio stations based on a song, artist, album or genre and save the radio station to access it from the library.
- Podcasts: Users can listen to podcasts and access a variety of podcast shows.
- Social media integration: Users can connect their Spotify account with their social media accounts to share music and playlists with friends.
- Offline listening: Users can download music to listen offline.
- Personalization: Spotify offers personalized recommendations for music and podcasts to its users based on their listening history and preferences.

Spotify

Home Search Your Library Create Playlist Liked Songs

FOCUS

- Peaceful Piano
- Deep Focus
- Instrumental Study
- chill lofi study beats
- Jazz Vibes
- Coding Mode
- Beats to think to
- Focus Flow

Spotify Playlists

- Today's Top Hits
- RapCaviar
- All Out 2010s
- Rock Classics
- Chill Hits
- Viva Latino
- Mega Hit Mix
- All Out 80s

Legal Privacy Center Privacy Policy Cookies About Ads Cookies

Premium Support Download Sign up Log in SHOW ALL

This screenshot shows the Spotify homepage. On the left sidebar, there are links for Home, Search, Your Library, Create Playlist, and Liked Songs. Below these are sections for 'FOCUS' and 'Spotify Playlists'. The 'FOCUS' section features eight cards for Peaceful Piano, Deep Focus, Instrumental Study, chill lofi study beats, Jazz Vibes, Coding Mode, Beats to think to, and Focus Flow. The 'Spotify Playlists' section shows eight cards for Today's Top Hits, RapCaviar, All Out 2010s, Rock Classics, Chill Hits, Viva Latino, Mega Hit Mix, and All Out 80s. At the bottom of the sidebar, there are links for Legal, Privacy Center, Privacy Policy, Cookies, About Ads, and Cookies. On the right side, there are links for Premium, Support, Download, Sign up, Log in, and SHOW ALL.

Spotify

Home Search Your Library Create Playlist Liked Songs

My Playlist #5

My Playlist #4

- Unholy (feat. Kim Petras)
- Unholy (feat. Kim Petras)
- Unholy (feat. Kim Petras)

phonk

Install App

Twilight of the Aesir

zake • 2 songs, 7 min 40 sec

PLAYLIST

My Playlist #4

#	TITLE	ALBUM	DATE ADDED	🕒
1	Catch the Sun	xmusologyx	0 seconds ago	2:42
2	Cheaper to Rent	Bjorn Veum	0 seconds ago	4:58

0:01 5:14:20

This screenshot shows the Spotify interface for a user named 'zake'. It displays 'My Playlist #4' which contains two songs: 'Catch the Sun' by xmusologyx and 'Cheaper to Rent' by Bjorn Veum. The total duration of the playlist is 7 minutes and 40 seconds. The Spotify sidebar on the left includes links for Home, Search, Your Library, Create Playlist, and Liked Songs. It also lists 'My Playlist #5' and 'My Playlist #4' along with song snippets from 'Unholy (feat. Kim Petras)' and 'Twilight of the Aesir'. The bottom navigation bar shows playback controls and a progress bar at 0:01 of 5:14:20.

Spotify

Home Search Your Library Create Playlist Liked Songs

My Playlist #5

My Playlist #4

- Unholy (feat. Kim Petras)
- Unholy (feat. Kim Petras)
- Unholy (feat. Kim Petras)

phonk

Install App

Flowers

Miley Cyrus

Today's Top Hits

Miley Cyrus is on top of the Hottest 50!

Spotify • 32,788,749 likes • 50 songs, about 2 hr 30 min

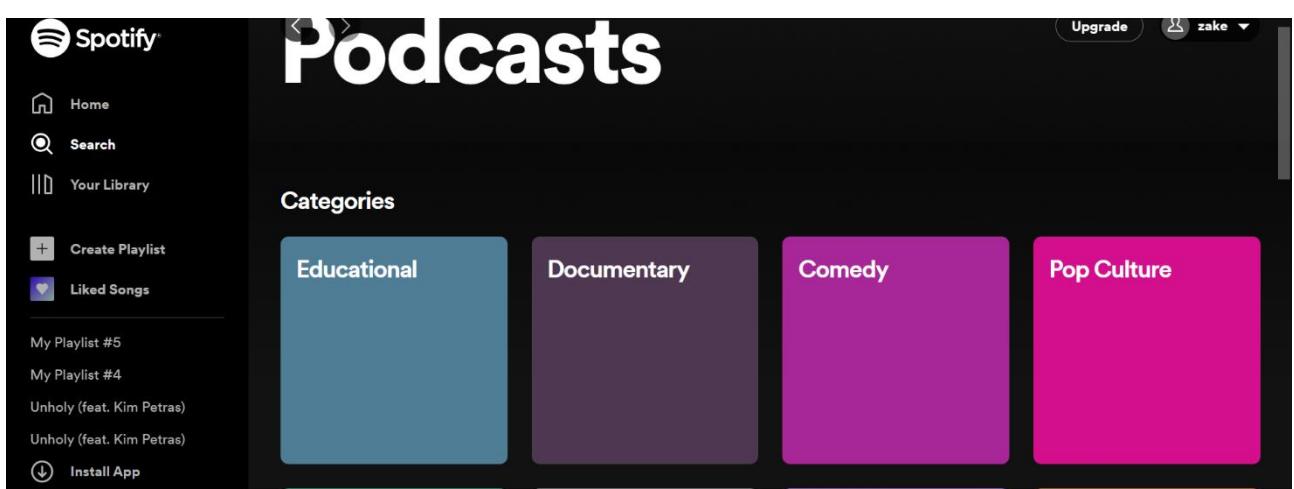
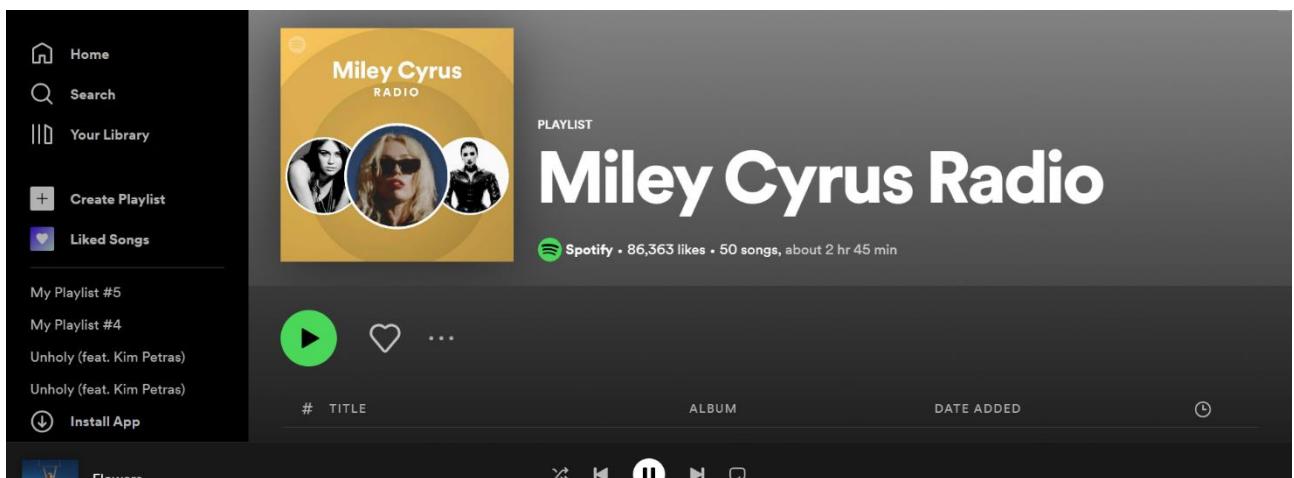
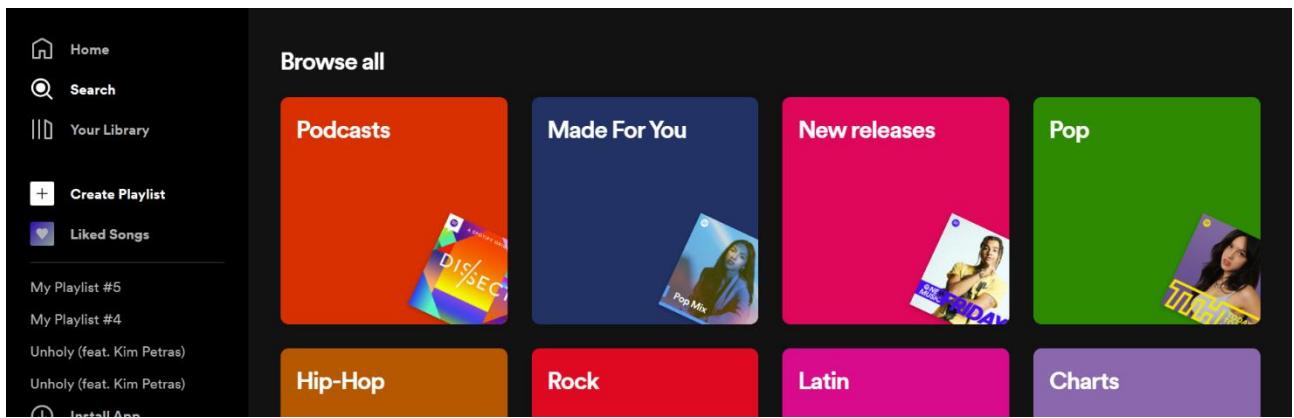
PLAYLIST

Today's Top Hits

#	TITLE	ALBUM	DATE ADDED	🕒
1	Flowers	Flowers	3 days ago	3:20

0:01 3:20

This screenshot shows the Spotify interface for a user named 'zake'. It displays 'Today's Top Hits' which contains one song: 'Flowers' by Miley Cyrus. The song has 32,788,749 likes and a total duration of about 2 hours and 30 minutes. The Spotify sidebar on the left includes links for Home, Search, Your Library, Create Playlist, and Liked Songs. It also lists 'My Playlist #5' and 'My Playlist #4' along with song snippets from 'Unholy (feat. Kim Petras)'. The bottom navigation bar shows playback controls and a progress bar at 0:01 of 3:20.



Pick your Premium

Listen without limits on your phone, speaker, and other devices.

Plan	Price	Accounts
Individual	4.99€/month after offer period	1 account
Duo	6.49€/month after offer period	2 accounts
Family	7.99€/month after offer period	Up to 6 accounts

Individual
4.99€/month after offer period
1 account

- ✓ Ad-free music listening
- ✓ Play anywhere - even offline
- ✓ On-demand playback

Duo
6.49€/month after offer period
2 accounts

- ✓ 2 Premium accounts for a couple under one roof
- ✓ Ad-free music listening, play offline, on-demand playback

Family
7.99€/month after offer period
Up to 6 accounts

- ✓ 6 Premium accounts for family members living under one roof
- ✓ Block explicit music
- ✓ Ad-free music listening, play offline, on-demand playback

GET STARTED

Terms and conditions apply. 1 month free not available for

GET STARTED

Terms and conditions apply. 1 month free not available for

GET STARTED

Terms and conditions apply. 1 month free not available for

2. Test Plan

2.1. Scope

Our main focus will be on functional testing, ensuring that all features of the application are working correctly and as expected. This includes testing the user's ability to create an account, manage it, browse and search for music, podcasts, as well as the ability to manage their playlists. We will also be performing accessibility testing, using tools such as aXe to verify that the website is accessible to as many users as possible. Additionally, we will be testing the website's localization to ensure that it is available in different languages.

However, we will not be testing the recommendation feature as it is beyond the scope of our testing efforts. nor will we be testing the applications responsive design and features that come with the premium version.

2.2. Testing Environment and Tools

In this project, we will be using Selenium WebDriver, a widely-used open-source tool for automating web browsers, as our main testing tool. We will also be using JUnit, a popular open-source testing framework for Java, to write and run our test cases. Additionally, we will be using the selenium-axe library to perform accessibility testing, which allows us to run accessibility checks using the Axe engine. The programming language we will be using is Java. Furthermore, we will be using Maven, a build automation tool for Java projects, to manage our dependencies. We will also be using the selenium-java dependency to run the selenium web driver. The development environment that we will be using is Eclipse. In addition, we will use Git for version control and GitHub for code hosting.

3. Test Execution

3.1. Sing Up

User wants to create an account for the website.

Test Name: Create an Account with Email and Password				
Description: Verify that users can create an account using an email and password				
Pre-condition(s): None				
Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<p>1. Open the Spotify website and click on the “Sign up” button.</p> <p>2. Fill in the required fields.</p> <p>3. Click on the “Sign up” button.</p> <p>4. Assert that the user has been logged in and redirected to the home page</p>	Email=”zaratester717@gmail.com”, Password=uniburch7, Display Name=”zara717”, Date=”02.10.2000”	The user successfully creates an account and is redirected to the home page with that account’s information.	The user successfully creates an account and is redirected to the home page with that account’s information.	PASS

Notes:

```

*SingUpTest.java ×

    @Test
    void singUpTestWithEmailAddress() throws InterruptedException {
        webDriver.get(baseUrl);
        webDriver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

        // cookies:
        webDriver.findElement(By.xpath("//html/body/div[13]/div[3]/div/div[2]/button")).click();

        Thread.sleep(2000);
        //click on the sing up for free button
        webDriver.findElement(By.xpath("//html/body/div[3]/div/div[2]/footer/div[1]/button/span")).click();

        //wait for page to load
        Thread.sleep(2000);
        // asserts that when we click the button we are transferred to the right page
        assertEquals(webDriver.getCurrentUrl(), "https://www.spotify.com/ba/signup?forward_url=https%3A%2F%2Fopen.spotify.com%2F%3F");

        //get all form elements and check if they are intractable
        List<WebElement> formElements=webDriver.findElements(By.className("Input-sc-1gbx9xe-0"));
        for(WebElement element:formElements) {
            assertTrue(element.isDisplayed());
            assertTrue(element.isEnabled());
        }

        //Fill in form
        //email
        webDriver.findElement(By.name("email")).sendKeys("zaretester71732@gmail.com");

        //confirm email
        webDriver.findElement(By.name("confirm")).sendKeys("zaretester71732@gmail.com");

        //password
        webDriver.findElement(By.name("password")).sendKeys("uniburch7");

        //profile name
        String nameString="zara717";
        webDriver.findElement(By.name("displayname")).sendKeys(nameString);

        //day
        webDriver.findElement(By.name("day")).sendKeys("2");

        //month
        Select monthSelect=new Select(webDriver.findElement(By.name("month")));
        monthSelect.selectByVisibleText("October");

        //year
        webDriver.findElement(By.name("year")).sendKeys("2000");

        //assert radioButtons - one is selected then others are not
        List<WebElement> radioButtons = webDriver.findElements(By.cssSelector("input[type='radio'][className='Radio-sc-tr5kfi-0']"));
        for (WebElement radioButton : radioButtons) {
            radioButton.click();
            for (WebElement otherRadioButton : radioButtons) {
                if (!otherRadioButton.equals(radioButton)) {
                    assertFalse(otherRadioButton.isSelected());
                }
            }
        }

        //gender
        webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/fieldset/div/div[2]/label/span[1]")).click();

        //last two buttons
        webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/div[6]/div/label/span[1]")).click();
        webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/div[7]/div/label/span[1]")).click();

        //captcha - manual
        Thread.sleep(12000);
        //gender
        webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/fieldset/div/div[2]/label/span[1]")).click();

        //last two buttons
        webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/div[6]/div/label/span[1]")).click();
        webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/div[7]/div/label/span[1]")).click();

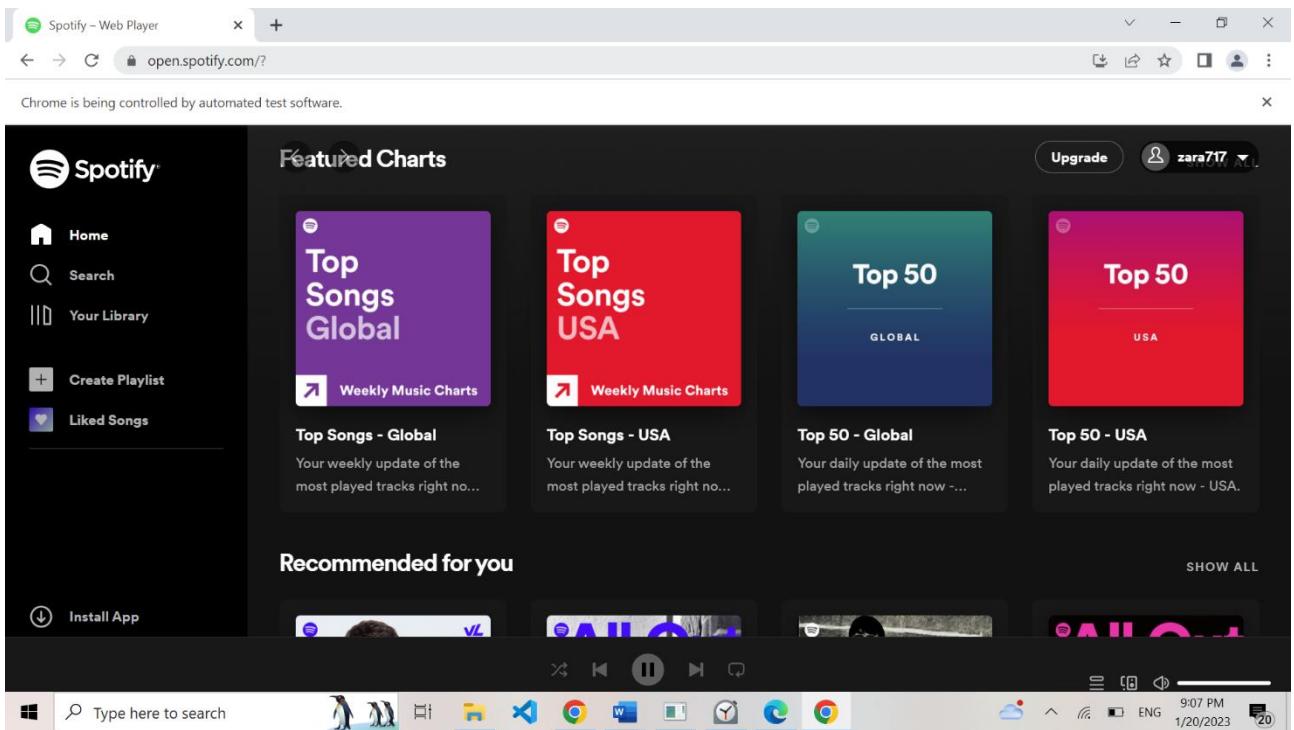
        //captcha - manual
        Thread.sleep(12000);

        //submit
        webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/div[9]/div/button/span[1]")).click();

        //after submit wait until page is loaded
        Thread.sleep(2000);
        //assert that we are singedIn with our account - that profile name we choose is displayed
        String profileUsername=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/button[2]/span")).getText();
        assertEquals(nameString, profileUsername);

        File screenshot = ((TakesScreenshot) webDriver).getScreenshotAs(OutputType.FILE);
        try {
            FileUtils.copyFile(screenshot, new File("Screenshot.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

**Test Name:** Create an Account with Invalid Credentials**Description:** Verify that users cannot create an account using invalid credentials**Pre-condition(s):** None

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Open the Spotify website and click on the “Sign up” button. 2. Click on the “Sing up” button to submit form. 3. Assert that error message is displayed indicating the credentials are invalid.		Error message is displayed indicating that the credentials are invalid.	Error message is displayed indicating that the credentials are invalid.	PASS

Notes:

```

eclipse-workspace - Spotify/src/test/java/SingUpScenario/signUpInvalid.java
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit X Profile.java Playlist.java *signUpInvalid.java X
Finished after 15.218 seconds
Runs: 1/1 Errors: 0 Failures: 0
> signUpInvalid [Runner: JUnit 5] (13.13)
Failure Trace

```

```

    @Test
    void signUpWithInvalidInput() throws InterruptedException {
        webDriver.get(baseUrl);
        webDriver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

        // cookies:
        webDriver.findElement(By.xpath("//html/body/div[13]/div[3]/div/div[2]/button")).click();

        Thread.sleep(2000);
        //click on the sing up for free button
        webDriver.findElement(By.xpath("//html/body/div[3]/div/div[2]/div[2]/footer/div[1]/button/span")).click();

        //wait for page to load
        Thread.sleep(2000);

        //submit
        webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/div[9]/div/button/span[1]")).click();
        Thread.sleep(2000);

        //assert that if i submit with improper input i stay on same page and get error messages
        assertEquals(webDriver.getCurrentUrl(), "https://www.spotify.com/ba/signup?forward_url=https%3A%2F%2Fopen.spotify.com%2F%3F");

        //error messages are displayed
        List<WebElement> errors=webDriver.findElements(By.cssSelector("span.Text-sc-g5kv67-0.jpUeQF"));
        for(WebElement e:errors) {
            e.isDisplayed();
            System.out.print(e.getText()); //error messages
        }
        assertEquals(9,errors.size()); // assert that for every invalid input there is an error message that goes with it
    }
}

```

Test Name: Create an Account with Socials

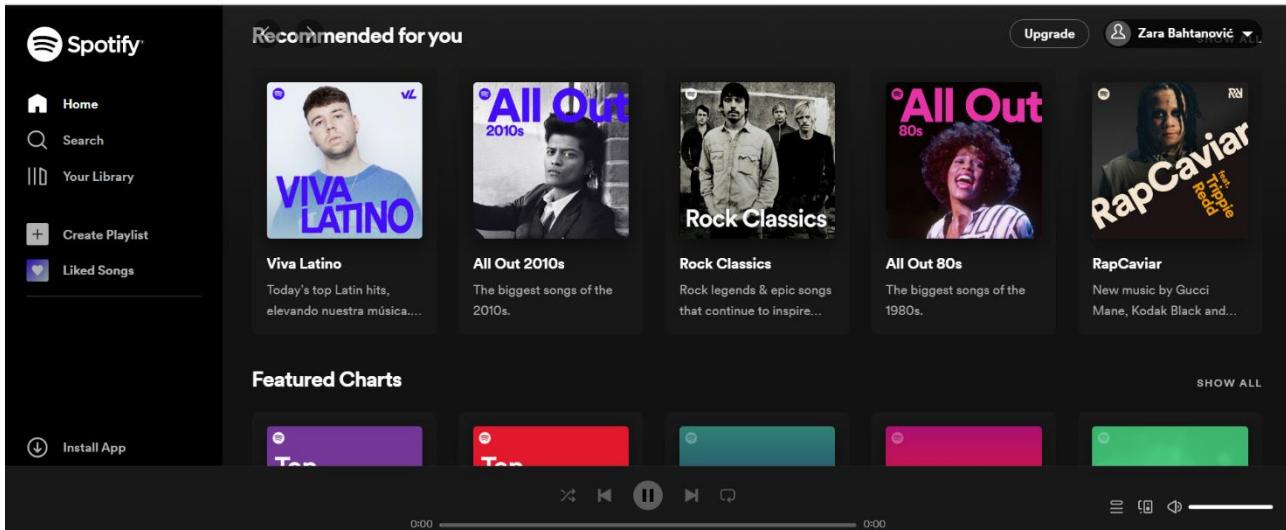
Description: Verify that users create an account using social media accounts

Pre-condition(s): User has a valid social media account

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Open the Spotify website and click on the “Sign up” button. 2. Click on the “Sing up with Google” button. 3. Log in to the Google Account 4. Fill in the form 5. Assert that the user has been logged in and redirected to the home page	Google account credentials	The user is able to successfully sign up using their Google account and is redirected to the home page	The user is able to successfully sign up using their Google account and is redirected to the home page	PASS

Notes: Similar steps for other social media accounts like Facebook

Chrome is being controlled by automated test software.



```
eclipse-workspace - Spotify/src/test/java/SingUpScenario/socials.java
File Edit Source Refactor Navigate Search Project Run Window Help
Profile.java Playlist.java signUpInvalid.java *socials.java SingUpTest.java
[Test
void singUpWithInvalidInput() throws InterruptedException {
    webDriver.get(baseUrl);

    //click on the sing up for free button
    webDriver.findElement(By.xpath("//html/body/div[3]/div/div[2]/div/footer/div[1]/button/span")).click();
    //wait for page to load
    Thread.sleep(2000);

    //sing up with google
    webDriver.findElement(By.partialLinkText("Google")).click();
    Thread.sleep(2000);

    //pick account
    webDriver.findElement(By.xpath("//html/body/div[1]/div[1]/div[2]/div/div[2]/div/div/div[1]/div/form/span/section/div/div/u[2]/div")).click();
    Thread.sleep(2000);

    String nameString=webDriver.findElement(By.name("displayname")).getAttribute("required value");

    //day
    webDriver.findElement(By.name("day")).sendKeys("2");

    //month
    Select monthSelect=new Select(webDriver.findElement(By.name("month")));
    monthSelect.selectByVisibleText("October");

    //year
    webDriver.findElement(By.name("year")).sendKeys("2000");

    //assert radioButtons - one is selected then others are not
    List<WebElement> radioButtons = webDriver.findElements(By.cssSelector("input[type='radio'][className='Radio-sc-tr5kfi-0']"));
    for (WebElement radioButton : radioButtons) {
        radioButton.click();
    }
}

//gender
webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/fieldset/div/div[2]/label/span[1]")).click();
Thread.sleep(2000);

//captcha - manual
Thread.sleep(12000);

//submit
webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/div[6]/div/button")).click();

//after submit wait until page is loaded
Thread.sleep(5000);
//assert that we are singedIn with our account - that profile name we choose is displayed
String profileUsername=webDriver.findElement(By.xpath("//html/body/div[4]/div[2]/div[1]/header/button[2]/span")).getText();
assertEquals(nameString, profileUsername);

File screenshot=((TakesScreenshot) webDriver).getScreenshotAs(OutputType.FILE);
try {
    FileUtils.copyFile(screenshot, new File("Screenshot.png"));
} catch (IOException e) {
    e.printStackTrace();
}
}
```

```
eclipse-workspace - Spotify/src/test/java/SingUpScenario/socials.java
File Edit Source Refactor Navigate Search Project Run Window Help
Profile.java Playlist.java signUpInvalid.java *socials.java SingUpTest.java
[Test
void singUpWithInvalidInput() throws InterruptedException {
    webDriver.get(baseUrl);

    //click on the sing up for free button
    webDriver.findElement(By.xpath("//html/body/div[3]/div/div[2]/div/footer/div[1]/button/span")).click();
    //wait for page to load
    Thread.sleep(2000);

    //sing up with google
    webDriver.findElement(By.partialLinkText("Google")).click();
    Thread.sleep(2000);

    //pick account
    webDriver.findElement(By.xpath("//html/body/div[1]/div[1]/div[2]/div/div[2]/div/div/div[1]/div/form/span/section/div/div/u[2]/div")).click();
    Thread.sleep(2000);

    String nameString=webDriver.findElement(By.name("displayname")).getAttribute("required value");

    //day
    webDriver.findElement(By.name("day")).sendKeys("2");

    //month
    Select monthSelect=new Select(webDriver.findElement(By.name("month")));
    monthSelect.selectByVisibleText("October");

    //year
    webDriver.findElement(By.name("year")).sendKeys("2000");

    //assert radioButtons - one is selected then others are not
    List<WebElement> radioButtons = webDriver.findElements(By.cssSelector("input[type='radio'][className='Radio-sc-tr5kfi-0']"));
    for (WebElement radioButton : radioButtons) {
        radioButton.click();
    }
}

//gender
webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/fieldset/div/div[2]/label/span[1]")).click();
Thread.sleep(2000);

//captcha - manual
Thread.sleep(12000);

//submit
webDriver.findElement(By.xpath("//html/body/div[1]/main/div/div/form/div[6]/div/button")).click();

//after submit wait until page is loaded
Thread.sleep(5000);
//assert that we are singedIn with our account - that profile name we choose is displayed
String profileUsername=webDriver.findElement(By.xpath("//html/body/div[4]/div[2]/div[1]/header/button[2]/span")).getText();
assertEquals(nameString, profileUsername);

File screenshot=((TakesScreenshot) webDriver).getScreenshotAs(OutputType.FILE);
try {
    FileUtils.copyFile(screenshot, new File("Screenshot.png"));
} catch (IOException e) {
    e.printStackTrace();
}
}
```

3.2. Login Functionality

User want to login to their account.

Test Name: Login with Email and Password				
Description: Verify that users can login to their account using their email and password				
Pre-condition(s): User has an exciting account with Spotify				
Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<ol style="list-style-type: none"> 1. Open the Spotify website and click on the “Login” button. 2. Fill in the required fields. 3. Click on the “Login” button. 	Email=“zaratester717@gmail.com”, Password=uniburch7,	User is successfully logged in and redirected to their home page.	User is successfully logged in and redirected to their home page.	PASS
Notes:				



```

@AfterAll
static void tearDownAfterClass() throws Exception {
    webDriver.quit();
}

@Test
void loginTest() throws InterruptedException {
    webDriver.get(baseUrl);

    Thread.sleep(3000);
    // cookies:
    webDriver.findElement(By.xpath("//html/body/div[13]/div[3]/div[2]/button")).click();

    //Login Button
    webDriver.findElement(By.xpath("//html/body/div[3]/div[2]/div[1]/header/div[6]/button[2]/span")).click();
    webDriver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));
    //Login page
    assertTrue(webDriver.findElement(By.id("login-to-continue")).getText().contains("log in"));

    //Input form
    webDriver.findElement(By.id("login-username")).sendKeys("zaratester717@gmail.com");
    webDriver.findElement(By.id("login-password")).sendKeys("uniburch7");

    //submit
    webDriver.findElement(By.xpath("//html/body/div[1]/div/div[2]/div/div[1]/div[3]/div[2]/button/div[1]")).click();

    //successful Login
    String profileUsername=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/button[2]/span")).getText();
    assertEquals("zara717", profileUsername);
}

```

Test Name: Login with Incorrect Credentials				
Description: Verify that users cannot login to their account with incorrect credentials				
Pre-condition(s): User has an exciting account with Spotify				
Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<ol style="list-style-type: none"> 1. Open the Spotify website and click on the “Login” button. 2. Fill in the required fields. 	Email=“zaratester717@gmail.com”, Password= “uniburch”	An error message is displayed and user is not logged in	An error message is displayed and user is not logged in	PASS

3. Click on the “Login” button.

4. Verify that an error message is displayed

Notes:



The screenshot shows a Java test runner interface. At the top, it displays "Runs: 1/1" and "Errors: 0". Below this, a tree view shows a single test case named "LoginFail [Runner: JUnit 5] (12.629 s)" which contains a method "loginWithInvalidInput() (12.629 s)". A "Failure Trace" section is visible below the tree, but its content is mostly blank. To the right of the tree, there is a large amount of Java code for the test. The code uses WebDriver to interact with a web page, specifically targeting elements with XPaths like "/html/body/div[13]/div[3]/div[2]/button" and "/html/body/div[3]/div[2]/div[1]/header/div[6]/button[2]/span". It includes sleep operations, element finds, and assertions. One assertion checks for the text "log in" in an element with id "login-to-continue". Another assertion checks for the text "Incorrect" in an element with a specific XPath. The code ends with a closing brace for the method.

```
Runs: 1/1  Errors: 0  Failures: 0
LoginFail [Runner: JUnit 5] (12.629 s)
  loginWithInvalidInput() (12.629 s)

Failure Trace

}

@Test
void loginWithInvalidInput() throws InterruptedException {
    webDriver.get(baseUrl);

    Thread.sleep(3000);
    // cookies:
    webDriver.findElement(By.xpath("/html/body/div[13]/div[3]/div[2]/button")).click();

    //login Button
    webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div[1]/header/div[6]/button[2]/span")).click();
    webDriver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));
    //login page
    assertTrue(webDriver.findElement(By.id("login-to-continue")).getText().contains("log in"));

    //input form
    webDriver.findElement(By.id("login-username")).sendKeys("zaratestester717@gmail.com");
    webDriver.findElement(By.id("login-password")).sendKeys("uniburch");

    //submit
    webDriver.findElement(By.xpath("/html/body/div[1]/div/div[2]/div/div[1]/div[3]/div[2]/button/div[1]")).click();
    Thread.sleep(3000);

    //error message
    assertTrue(webDriver.findElement(By.xpath("/html/body/div[1]/div/div[2]/div/div[1]/span")).getText().contains("Incorrect"));
}
```

Test Name: Login with Socials

Description: Verify that users can login to their account with proper social media credentials

Pre-condition(s): User has an account that has been linked to a social media account

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Open the Spotify website and click on the “Login” button. 2. Fill in the required fields. 3. Click on the “Login” button.	Google account credentials	The user is successfully logged in using their Google account and is redirected to the home page	The user is successfully logged in using their Google account and is redirected to the home page	PASS

Notes: Similar steps for other social media accounts like Facebook

```

    @Test
    void test() throws InterruptedException {
        webDriver.get(baseUrl);

        //login page
        webDriver.findElement(By.xpath("//html/body/div[3]/div/div[1]/header/div[6]/button[2]/span")).click();

        //assertTrue(webDriver.findElement(By.id("login-to-continue")).getText().contains("log in"));

        //signInviaGoogle
        webDriver.findElement(By.xpath("//html/body/div[1]/div/div[2]/div/div[1]/ul/li[3]/button")).click();
        Thread.sleep(3000);

        //pick account
        webDriver.findElement(By.xpath("//html/body/div[1]/div[1]/div[2]/div/div[2]/div/div[2]/div/div[1]/div/form/span/section/div/div/div[1]/div[1]/div[1]")).click();
        Thread.sleep(5000);

        String profileUsername=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/button[2]/span")).getText();
        assertEquals("Zara Bahtanović", profileUsername); // we have been logged in with account credentials
    }
}

```

Test Name: Verify Logout Functionality

Description: Verify that users can logout from their account

Pre-condition(s): User is logged in to their account

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1.Click profile 2. Select “logout” button. 3. Assert that the user has been logged out.	Google account credentials	The user is successfully logged out.	The user is successfully logged out.	PASS

Notes:

```

    @Test
    void logoutTest() throws InterruptedException {
        webDriver.get(baseUrl);
        Thread.sleep(3000);

        //click profile
        webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/button[2]")).click();
        Thread.sleep(3000);

        //click logout
        webDriver.findElement(By.xpath("//html/body/div[14]/div/div[1]/ul/li[7]/button")).click();
        Thread.sleep(6000);

        //check for new login button
        String loginButtonElement = webDriver.findElement(By.xpath("//html/body/div[3]/div/div[2]/div[1]/header/div[6]/button[2]/span")).getText();
        assertEquals("Log in", loginButtonElement);
    }
}

```

Test Name: Forgotten Password**Description:** Verify that users can reset their password if they have forgotten it.**Pre-condition(s): The user has an exciting account with Spotify**

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<p>1. Open the Spotify website and click on the “Login” button.</p> <p>2. Click on the "forgot password" link.</p> <p>3. Enter a valid email and click the "send" button.</p> <p>4. Confirm that an email has been successfully send</p>	Email="zaratester717@gmail.com",	User is able to reset their password	User is able to reset their password	PASS

Notes:


The screenshot shows a JUnit test run summary and the corresponding Java code for the 'resetPassword' method. The test ran for 13.139 seconds with 1 run, 0 errors, and 0 failures. The code uses WebDriver to interact with a web application, performing steps like navigating to the login page, clicking the forgot password link, entering an email, and sending the password reset email. It then asserts that the message 'We've sent you an email. Just follow the instructions to reset your password.' is present in the response.

```

Finished after 13.139 seconds
Runs: 1/1  Errors: 0  Failures: 0
> restPassword [Runner: JUnit 5] (11.699 s)

void resetPassword() throws InterruptedException {
    webDriver.get(baseUrl);
    Thread.sleep(3000);

    // cookies:
    webDriver.findElement(By.xpath("//html/body/div[13]/div[3]/div/div[2]/button")).click();

    //login Button
    webDriver.findElement(By.xpath("//html/body/div[3]/div/div[2]/div[1]/header/div[6]/button[2]/span")).click();
    webDriver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));

    //login page
    assertTrue(webDriver.findElement(By.id("login-to-continue")).getText().contains("log in"));

    //forgot password
    webDriver.findElement(By.linkText("Forgot your password?")).click();

    //assert that we are at the password reset route
    assertEquals("Password Reset",webDriver.findElement(By.xpath("//html/body/div[1]/div/section/div/h1")).getText());

    //enter email
    webDriver.findElement(By.id("email_or_username")).sendKeys("zara717tester@gmail.com");

    //send
    webDriver.findElement(By.xpath("//html/body/div[1]/div/section/div/form/div[2]/button/div[1]")).click();

    Thread.sleep(3000);
    //successfully send
    String sentString=webDriver.findElement(By.xpath("//html/body/div[1]/div/section/div/p")).getText();
    //System.out.print(sentString);
    assertTrue(sentString.contains("We've sent you an email. Just follow the instructions to reset your password."));

}

```

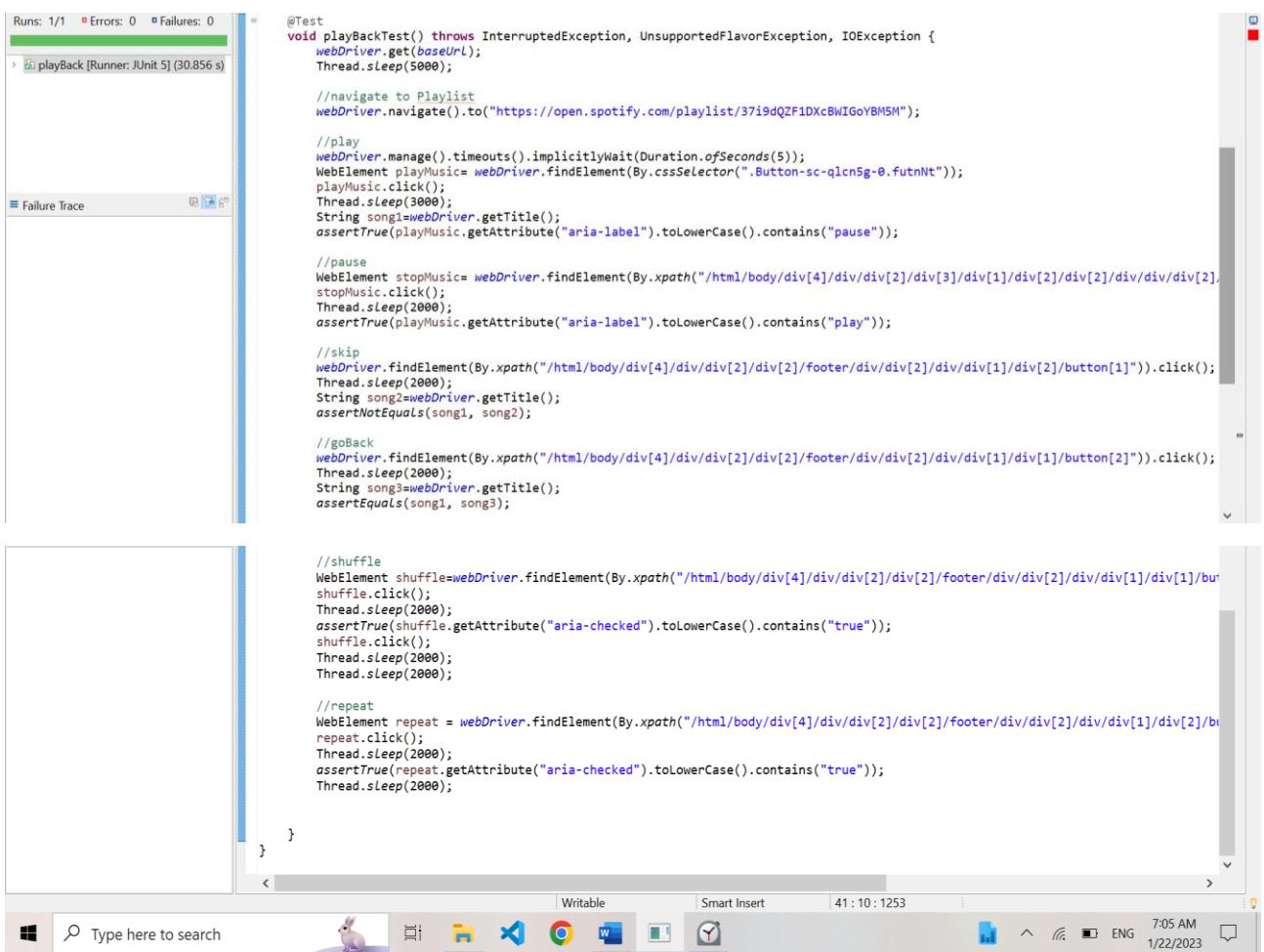
3.3. Playback Functionality

User want to play a song

Test Name: Verify Playback of Playlist**Description:** Test the functionality of playing a playlist, pausing a playlist, skipping a song, shuffling songs, and repeating a song**Pre-condition(s): A user has selected a playlist**

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1.Click “play” 2.Click “pause” 3.Click “skip” 4.Click “go Back” 5. Click “Shuffle” 5.Click “Repeat”		The song plays, pauses, skips, shuffles and repeats.	The song plays, pauses, skips, shuffles and repeats.	PASS

Notes: Similar steps for playing an album, radio, song...



```

Runs: 1/1  Errors: 0  Failures: 0
playBack [Runner: JUnit 5] (30.856 s)

Failure Trace

@Test
void playBackTest() throws InterruptedException, UnsupportedFlavorException, IOException {
    webDriver.get(baseUrl);
    Thread.sleep(5000);

    //navigate to Playlist
    webDriver.navigate().to("https://open.spotify.com/playlist/37i9dQZF1DXcBWIGoYBM5M");

    //play
    webDriver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));
    WebElement playMusic= webDriver.findElement(By.cssSelector(".Button-sc-qlcn5g-0.futnNt"));
    playMusic.click();
    Thread.sleep(3000);
    String song1=webDriver.getTitle();
    assertTrue(playMusic.getAttribute("aria-label").toLowerCase().contains("pause"));

    //pause
    WebElement stopMusic= webDriver.findElement(By.xpath("/html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div[2]/div[1]/div/div[2]"));
    stopMusic.click();
    Thread.sleep(2000);
    assertTrue(playMusic.getAttribute("aria-label").toLowerCase().contains("play"));

    //skip
    webDriver.findElement(By.xpath("/html/body/div[4]/div/div[2]/div[2]/footer/div/div[2]/div[1]/div[1]/button[1]")).click();
    Thread.sleep(2000);
    String song2=webDriver.getTitle();
    assertNotEquals(song1, song2);

    //goBack
    webDriver.findElement(By.xpath("/html/body/div[4]/div/div[2]/div[2]/footer/div/div[2]/div/div[1]/div[1]/button[2]")).click();
    Thread.sleep(2000);
    String song3=webDriver.getTitle();
    assertEquals(song1, song3);

    //shuffle
    WebElement shuffle=webDriver.findElement(By.xpath("/html/body/div[4]/div/div[2]/div[2]/footer/div/div[2]/div/div[1]/button[3]"));
    shuffle.click();
    Thread.sleep(2000);
    assertTrue(shuffle.getAttribute("aria-checked").toLowerCase().contains("true"));
    shuffle.click();
    Thread.sleep(2000);
    Thread.sleep(2000);

    //repeat
    WebElement repeat = webDriver.findElement(By.xpath("/html/body/div[4]/div/div[2]/div[2]/footer/div/div[2]/div/div[1]/div[2]/button[4]"));
    repeat.click();
    Thread.sleep(2000);
    assertTrue(repeat.getAttribute("aria-checked").toLowerCase().contains("true"));
    Thread.sleep(2000);

}
}

Type here to search  Writable  Smart Insert  41 : 10 : 1253  7:05 AM  1/22/2023

```

3.4. Search and Browse Functionality

This test scenario aims to verify that a user can search through the music library for a specific artist, album, song or browse the library through genres.

Test Name: Verify Search Functionality

Description: Test the search functionality and verify the correct results are displayed

Pre-condition(s): A user is logged in to their account

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<ol style="list-style-type: none"> Click the “search” button Enter data into search bar Verify that the correct results are displayed Verify that the results are clickable and lead to correct path 	Search data = “BTS”	The search result should display correct result based on entered data.	The search result should display correct result based on entered data.	PASS

Notes:

```

eclipse-workspace - Spotify/src/test/java/SearchScenario/Search.java
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit Search.java
Finished after 56.316 seconds
Runs: 3/3 Errors: 0 Failures: 0
Search [Runner: JUnit 5] (53.720 s)
  searchTest() [21.081 s]
  categorizeResults() [21.309 s]
  clearRecentSearches() [11.326 s]

Failure Trace
}

@Text
@Order(1)
void searchTest() throws InterruptedException {
    webDriver.get(baseUrl);
    Thread.sleep(3000);

    //search button
    webDriver.findElement(By.cssSelector("#main > div > div.Root__top-container > nav > div.tUwyjggD2n5KVEtP5z1B > ul > li"));
    Thread.sleep(3000);

    //enter smth into search bar
    String searchString="bts";
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/div[3]/div/div/form/input")).sendKeys(searchString);
    Thread.sleep(3000);

    //assert that the correct stuff is displayed based on our search
    String result=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div[2]/main/div[2]/div"));
    assertEquals(searchString, result.toLowerCase());
    assertTrue(webDriver.getCurrentUrl().contains(searchString));

    //Verify that the user can click on a search result and be directed to the correct artist, album, or song page.
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div/div[2]/main/div[2]/div"));
    Thread.sleep(3000);
    String resultHeaderString=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div"));
    assertEquals(searchString, resultHeaderString.toLowerCase());
}
}

```

Test Name: Verify Search by Categorizes

Description: Test the functionality of searching by category (e.g., album, song.)

Pre-condition(s): A user is logged in to their account

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<ol style="list-style-type: none"> Click the “search” button. Enter data into search bar 		The searched data should be correctly filtered and displayed by selected category	The searched data is correctly filtered and displayed by selected category	PASS

3. Test the feature to filter result by category

Notes:

```

eclipse-workspace - Spotify/src/test/java/SearchScenario/Search.java
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit x
Search.java x
Finished after 56.316 seconds
Runs: 3/3 Errors: 0 Failures: 0
Search [Runner: JUnit 5] (53.720 s)
  searchTest() (21.081 s)
  categorizeResults() (21.309 s)
  clearRecentSearches() (11.326 s)

Failure Trace

```

```

@Test
@Order(2)
void categorizeResults() throws InterruptedException {
    //search button
    webDriver.findElement(By.cssSelector("#main > div > div.Root__top-container > nav > div.tUwyjggD2n5KvEtP5z1B > ul > li"));
    Thread.sleep(3000);

    //enter smth into search bar
    String searchString="bts";
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/div[3]/div/div/form/input")).sendKeys(searchString);
    Thread.sleep(3000);

    //filter result by some category
    List<WebElement>searchOptions=webDriver.findElements(By.className("ZWI7jsjzJaR_G8Hy4W6J"));
    for(WebElement element:searchOptions) {
        element.click();
        Thread.sleep(2000);
    }
}

```

Test Name: Verify Clear Search Functionality

Description: Test the functionality of clearing recent searched

Pre-condition(s): A user is logged in to their account and has made at least one prior search

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Click on “search” button 2. Click on “remove” button 3. Verify there are no more “recent searches”		No more “recent searches” displayed	No more “recent searches” displayed	PASS

Notes:

```

eclipse-workspace - Spotify/src/test/java/SearchScenario/Search.java
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit x
Search.java x
Finished after 56.316 seconds
Runs: 3/3 Errors: 0 Failures: 0
Search [Runner: JUnit 5] (53.720 s)
  searchTest() (21.081 s)
  categorizeResults() (21.309 s)
  clearRecentSearches() (11.326 s)

Failure Trace

```

```

    }
}

@Test
@Order(3)
void clearRecentSearches() throws InterruptedException {
    Thread.sleep(3000);
    //search button
    webDriver.findElement(By.cssSelector("#main > div > div.Root__top-container > nav > div.tUwyjggD2n5KvEtP5z1B > ul > li"));
    Thread.sleep(3000);

    //get x button
    List<WebElement> removeElements = webDriver.findElements(By.className("xmJl0s8mcJ3bfhtnoaP1"));
    System.out.print(removeElements.size());

    for(WebElement element : removeElements){
        element.click();
        Thread.sleep(2000);
    }

    //assert there are no more recent searches
    removeElements = webDriver.findElements(By.className("xmJl0s8mcJ3bfhtnoaP1"));
    assertEquals(0, removeElements.size());
    Thread.sleep(2000);
}
}

```

Test Name: Verify Browsing Functionality

Description: Test that user can browse through the music library by some genre/option

Pre-condition(s): A user is logged in to their account

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Click the search link 2. Verify that each prelisted option for “browsing” is clickable and leads to correct path	,	The genres are all valid, clickable and lead to a correct path	The genres are all valid, clickable and lead to correct a path	PASS

Notes:



```

@Test
void testBrowsign() throws InterruptedException {
    WebDriver webDriver;
    webDriver.get("http://www.automationpractice.com");
    Thread.sleep(3000);
    webDriver.findElement(By.cssSelector("main > div > div.Root__top-container > nav > div.tNavJggD2nSkvEtP5z1B > ul > li:nth-child(2) > a > span")).click();
    Thread.sleep(3000);

    JavascriptExecutor jsExecutor=(JavascriptExecutor) webDriver;
    List<WebElement> browseCategories=webDriver.findElements(By.className("en2LrSSfvrgQoajs6ce"));
    List<String> expected=new ArrayList<String>();
    List<String> result=new ArrayList<String>();

    for(int i=1;i<browseCategories.size();i++) {
        if(i==1 || i==2)
            continue;
        expected.add(browseCategories.get(i).getText());
    }

    for(int i=1;i<browseCategories.size();i++) {
        WebElement genre=webDriver.findElement(By.className("rEN7rcpale5GL9z0HQ0R"));
        Thread.sleep(2000);
        assertThat(genre.isDisplayed() && genre.isEnabled());
        Thread.sleep(2000);
        jsExecutor.executeScript("arguments[0].click()",genre);
        Thread.sleep(3000);
        try {
            WebElement genreName=webDriver.findElement(By.className("rEN7rcpale5GL9z0HQ0R"));
            Thread.sleep(2000);
            assertEquals(genreName.isDisplayed() && genreName.isEnabled());
            Thread.sleep(2000);
            String genreString=genre.getText();
            result.add(genreString);
        } catch (NoSuchElementException e) {
            try {
                String genreString = webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div[2]/main/div/div/div/section/div[2]/div/a["+i+"]").getText());
                result.add(genreString);
            } catch (NoSuchElementException e2) {
                //NOOP
            }
        }
        webDriver.navigate().back();
        Thread.sleep(2000);
    }
}

System.out.println(expected);
System.out.println(result);
assertThat(expected.equals(result));
}
}

```

3.5. Localization

This test scenario aims to verify that the website is able to display content in different languages and that the user can switch between languages easily

Test Name: Verify Localization for Different Languages

Description: Test that the website is properly localized for different languages

Pre-condition(s): The website should be available for different languages and user is logged in.

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1.Open website 2.Go to user settings 3.Select a different language from the dropdown menu	Language preference= “de” and “en”	Website's language is changed to the selected language.	Website's language is changed to the selected language	PASS

4. Verify that the website is properly localized for the selected language.

5. Repeat step 3

Notes:

```

Finished after 28.467 seconds
Runs: 1/1    Errors: 0    Failures: 0

localization [Runner: JUnit 5] (26.143 s)
  languageLocalization() (26.143 s)

Failure Trace

```

```

@Test
void languageLocalization() throws InterruptedException {
    webDriver.get(baseUrl);
    Thread.sleep(5000);

    //click profile
    webDriver.findElement(By.xpath("//html/body/div/div[2]/div[1]/header/button[2]")).click();
    Thread.sleep(2000);
    //settings
    webDriver.findElement(By.linkText("Settings")).click();
    Thread.sleep(2000);

    //change language
    String languageString="de";
    Select language=new Select(webDriver.findElement(By.id("desktop.settings.selectLanguage")));
    language.selectByValue(languageString);
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/div[2]/div/div[2]/main/div/div[2]"));
    Thread.sleep(5000);

    //confirm change
    String pageSource=webDriver.getPageSource();
    assertTrue(pageSource.contains("html lang=\""+languageString+"\""));
    Thread.sleep(2000);

    //language
    languageString="en";
    language=new Select(webDriver.findElement(By.id("desktop.settings.selectLanguage")));
    language.selectByValue(languageString);
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/div[2]/div/div[2]/main/div/div[2]"));
    Thread.sleep(3000);

    pageSource=webDriver.getPageSource();
    assertTrue(pageSource.contains("html lang=\""+languageString+"\""));
    Thread.sleep(2000);
}

```

3.6. Accessibility

This test scenario aims to check if the website is accessible for users.

Test Name: Verify Website's Accessibility

Description: Verify that the website is accessible for users with disabilities / compatible with different screen readers / compatible with keyboard navigation

Pre-condition(s): None

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Navigate to the website 2. Run the Axe accessibility tool on the website 3. Check the results for any violations		No accessibility violations found.	Accessibility violations found.	FAIL

Notes:

```

@AfterAll
static void tearDownAfterClass() throws Exception {
    webDriver.quit();
}

@Test
void testAccessibility(){
    webDriver.get(baseUrl);

    AxeBuilder builder = new AxeBuilder();
    Results results = builder.analyze(webDriver);
    List<Rule> violations = results.getViolations();
    if (violations.size() == 0)
    {
        assertTrue(true);
    }
    else
    {
        File reportFile = new File("axe-report");
        AxeReporter.writeResultsToJsonFile(reportFile.getName(), results);
        assertEquals(0, violations.size(), violations.size() + " violations found. See the report at: " + reportFile.getAbsolutePath());
    }
}

```

3.7. Radio Functionality

This test scenario aims to verify that user is able to access and use the radio features, and based on a particular playlist/song/artist that is selected, a new playlist can be created with the same vibes and saved to user's library.

Test Name: Verify Radio Creation from a Playlist				
Description: Test the functionality of creating a radio playlist from a certain playlist				
Pre-condition(s): A user is logged in and has a playlist selected				
Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Select playlist 2. Click the “more” button 3. Click the “go to radio” button. 4. Verify that you are redirected to that playlist’s radio playlist		The user is able create and listen to a playlist based on a specific playlist.	The user is able create and listen to a playlist based on a specific playlist.	PASS
Notes:				

Test Name: Verify Radio Creation from an Artist**Description:** Test the functionality of creating a radio playlist from a certain artist**Pre-condition(s): A user is logged in and has an artist selected**

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Select Artist 2. Click the “more” button 3. Click the “go to radio” button. 4. Verify that you are redirected to that artist’s radio playlist	Name of Artist = John Legend	The user is able create and listen to a playlist based on a specific artist.	The user is able create and listen to a playlist based on a specific artist.	PASS

Notes:


The screenshot shows a JUnit test run interface. The top status bar indicates "Finished after 25.852 seconds". The results summary shows "Runs: 1/1" with "0 Errors" and "0 Failures". Below this, the test class and method are listed: `@Test void testRadiofromArtist() throws InterruptedException {`. The code block contains several assertions and interactions with a WebDriver, including searching for an artist name, clicking a more option, and navigating to a radio playlist. The code ends with an assertion to check if the current URL contains "playlist". The bottom part of the interface shows a "Failure Trace" section which is currently empty.

```

    @Test
void testRadiofromArtist() throws InterruptedException {
    //artist name
    String nameString="john legend";
    webDriver.get(baseUrl);
    Thread.sleep(3000);

    //search button
    webDriver.findElement(By.cssSelector("#main > div > div.Root__top-container > nav > div.tUwyjggD2n5KvEtP5z1B > ul > li:nth-child(1)")).click();
    Thread.sleep(3000);

    //enter artist into search bar
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/div[3]/div/form/input")).sendKeys(nameString);
    Thread.sleep(3000);

    //go to artist
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div[2]/main/div[2]/div/div[1]/div[1]")).click();
    Thread.sleep(4000);

    //more option
    webDriver.findElement(By.className("T0anrkk_QA4IAQL29get")).click();
    Thread.sleep(4000);

    //go to radio
    webDriver.findElement(By.xpath("//html/body/div[15]/div/ul/li[2]/button")).click();
    Thread.sleep(3000);

    //assert
    assertEquals(webDriver.getCurrentUrl().contains("playlist"));
    String h1=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div[2]/main/div/div[1]/div[1]")).getText();
    assertEquals(nameString+" radio", h1.toLowerCase());
}
  
```

Test Name: Verify Saving Radio to Personal Playlist**Description:** Test the functionality of saving a radio playlist to user’s personal playlist collection**Pre-condition(s): A user is logged in and has a radio playlist created.**

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Go to radio playlist 2. Click “more” button 3. Click “add to playlist” button		The user is able save a radio playlist to their personal playlist collection	The user is able save a radio playlist to their personal playlist collection	PASS

4.Click “create new playlist” button

5. Wait for the confirmation message pop-up.

Notes:

```
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit x
radioTest.java x
radioTest [Runner: JUnit 5] (13.143 s)
Runs: 1/1 Errors: 0 Failures: 0
Failure Trace
@Test
void addToPlaylist() throws InterruptedException {
    //more
    webDriver.get("https://open.spotify.com/station/playlist/37i9dQZF1DXcBWIGoYBM5M");
    Thread.sleep(3000);
    webDriver.findElement(By.className("T0anrkk_QA4IAQL29get")).click();
    Thread.sleep(4000);

    //hover over add to playlist
    Actions actions=new Actions(webDriver);
    WebElement addElement=webDriver.findElement(By.xpath("//html/body/div[14]/div/ul/li[2]/button"));
    actions.moveToElement(addElement).perform();
    Thread.sleep(1000);

    //create new playlist
    WebElement createPlaylist=webDriver.findElement(By.xpath("//html/body/div[14]/div/ul/li[2]/div/ul/li[2]/button"));
    createPlaylist.click();
    Thread.sleep(2000);
}
```

3.8. Profile Management

Test scenario to verify that user can view their account and modify it.

Test Name: Verify Account Settings

Description: Test that a user can manage their account settings, including email and push notifications

Pre-condition(s): A user has an account and is logged in

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<ol style="list-style-type: none">Click on the “user” tab.Click on the “settings” link.Update account information.Click on “save changes” button.Verify that the account setting have been successfully changed.		The user can view their settings and successfully update it.	The user can view their settings and successfully update it.	PASS

Notes:

```

@Order(1)
void viewEditProfile() throws InterruptedException{
    String userName="313gmiugyw5oanf2vyq53hj72kfU";
    String newDisplayNameValue="zake";
    //get spotify
    webDriver.get(baseUrl);
    Thread.sleep(3000);

    //click user
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/button[2]")).click();
    Thread.sleep(3000);

    //click profile
    webDriver.findElement(By.linkText("Profile")).click();
    Thread.sleep(3000);
    assertTrue(webDriver.getCurrentUrl().contains("user/"+userName));

    //edit display name
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div[2]/div/div[2]/main/section/div/div[1]/div[1]/div[2]")).click();
    Thread.sleep(3000);

    //model opens
    WebElement editElement=webDriver.findElement(By.id("user-edit-name"));
    Thread.sleep(3000);
    editElement.clear();
    editElement.sendKeys(newDisplayNameValue);

    //save
    webDriver.findElement(By.xpath("//html/body/div[16]/div/div/div/form/div[3]/button")).click();
    Thread.sleep(3000);

    //assert
    String newName=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div[2]/main/section/div/div[1]/div[1]/div[2]")).getText();
    assertEquals(newDisplayNameValue,newName);
}

```

Test Name: Verify Account Settings Update

Description: Verify that a user can update their account settings

Pre-condition(s): A user has an account and is logged in

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<ul style="list-style-type: none"> 1. Click on the “user” tab 2. Click on the “settings” link 3. Click on “edit” link 4. Change gender setting. 5. Verify that the settings have successfully updated. 	Gender = “other”	The account settings are updated and saved successfully	The account settings are updated and saved successfully.	PASS

Notes:

```

Runs: 1/1  Errors: 0  Failures: 0
Profile [Runner: JUnit 5] (18.262 s)
editAccountSetting() (18.262 s)

Failure Trace
@Test
void editAccountSetting() throws InterruptedException{
    //get spotify
    webDriver.get(baseURL);
    Thread.sleep(3000);

    //click user
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[1]/header/button[2]")).click();
    Thread.sleep(3000);

    String handle=webDriver.getWindowHandle();
    //setting
    webDriver.findElement(By.xpath("//html/body/div[14]/div/div/ui/li[1]/button")).click();
    for(String handle:webDriver.getWindowHandles()) {
        if(handle.equals(handle)) {
            webDriver.switchTo().window(handle);
        }
    }

    //click edit profile
    Thread.sleep(3000);
    webDriver.findElement(By.xpath("//html/body/div[1]/div/div/div[2]/div[3]/div[2]/div/article[1]/div/a")).click();
    Thread.sleep(2000);

    //select gender
    Select genderSelect=new Select(webDriver.findElement(By.xpath("//html/body/div[1]/div/div[2]/div[2]/div[2]/div[2]/div/article/section/form/section/div[3]/div[2]/select")));
    String beforeString=genderSelect.getFirstSelectedOption().getText();
    genderSelect.selectByVisibleText("Male");

    //confirm
    Thread.sleep(3000);
    webDriver.findElement(By.xpath("//html/body/div[1]/div/div[2]/div[2]/div[2]/div/article/section/form/div/button")).click();

    //confirm
    Select genderSelect2=new Select(webDriver.findElement(By.xpath("//html/body/div[1]/div/div[2]/div[2]/div[2]/div[2]/div/article/section/form/section/div[3]/div[2]/select")));
    String afterString=genderSelect2.getFirstSelectedOption().getText();
    assertEquals(beforeString, afterString);
}

```

3.9. Podcast Functionality

This test scenario aims to verify that user can subscribe to podcasts, play episodes, un-subscribe, etc...

Test Name: Verify Podcast Subscription				
Description: Verify that a user can subscribe to a podcast				
Pre-condition(s): A user is logged in				
Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Navigate to “Podcast” tab. 2. Pick desired genre 3. Select the first podcast. 3. Click on the “follow” button. 4. Wait for the success pop-up notification and assert that user has successfully subscribed by checking class name.	Genre = “Documentary”	The user is successfully subscribed to the podcast.	The user is successfully subscribed to the podcast.	PASS
Notes:				

Runs: 1/1 Errors: 0 Failures: 0

```

    void testPodcastSubscription() throws InterruptedException {
        webDriver.get(baseUrl);
        Thread.sleep(3000);
        String podcastGenre="Documentary";
        //search button
        webDriver.findElement(By.cssSelector("#main > div > div.Root__top-container > nav > div.tUwyjggD2n5KvEtP5z1B > ul > li:nth-child(1) > a"));
        Thread.sleep(3000);

        //select podcast
        webDriver.findElement(By.linkText("Podcasts")).click();
        Thread.sleep(3000);

        //select genre
        webDriver.findElement(By.linkText(podcastGenre)).click();
        Thread.sleep(3000);

        //select first podcast
        webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div/div[2]/main/div[2]/div[2]/div[1]/div[1]"));
        Thread.sleep(3000);
        assertTrue(webDriver.getCurrentUrl().contains("show"));

        //click follow button
        WebElement followWebElement=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div/div[2]/main/div[2]/div[2]/div[1]/div[1]"));
        followWebElement.click();
        Thread.sleep(5000);
        assertTrue(followWebElement.getAttribute("class").contains("fEbcweEiUPPy2eaIaD3F"));
    }

```

Test Name: Verify Podcast Un-Subscription

Description: Verify that a user can un-subscribe to a podcast

Pre-condition(s): A user is logged in and is subscribed to at least one podcast

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Select podcast. 4. Wait for the success pop-up notification and assert that user has successfully unsubscribed by checking class name.		The user is successfully un-subscribed to the podcast.	The user is successfully un-subscribed to the podcast.	PASS

Notes:

eclipse-workspace - Spotify/src/test/java/podcast/PodcastTest.java

```

File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit Search.java
*PodcastTest.java * Search.java
assertTrue(followWebElement.getAttribute("class").contains("fEbcweEiUPPy2eaIaD3F"));

}

@Test
@Order(2)
void testPodcastUnSubscription() throws InterruptedException {
    //click following button
    WebElement followWebElement=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div/div[2]/main/div[2]/div[2]/div[1]/div[1]"));
    followWebElement.click();
    Thread.sleep(5000);
    assertFalse(followWebElement.getAttribute("class").contains("fEbcweEiUPPy2eaIaD3F"));
}

```

Test Name: Verify Episode Functionality

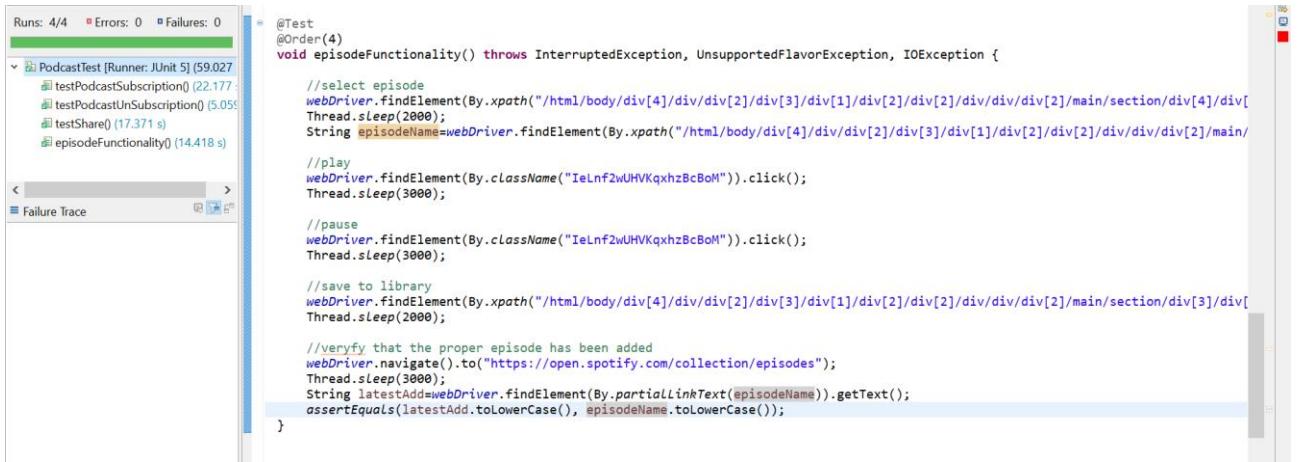
Description: Test the functionality of playing an episode, pausing an episode, adding it to the library

Pre-condition(s): A user has selected an episode

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1.Click "play" 2.Click "pause"		The episode plays, pauses, and can be added to library	The episode plays, pauses, and can be added to library.	PASS

3. Add to library		
4. Verify that the episode has been added to library		

Notes:



Runs: 4/4 Errors: 0 Failures: 0

```
= @Test  
@Order(4)  
void episodeFunctionality() throws InterruptedException, UnsupportedFlavorException, IOException {  
  
    //select episode  
    webDriver.findElement(By.xpath("/html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div[2]/main/section/div[4]/div[  
Thread.sleep(2000);  
String episodeName=webDriver.findElement(By.xpath("/html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div[2]/main/  
  
    //play  
    webDriver.findElement(By.className("IeLnf2wUHVQxhzBcBoM")).click();  
    Thread.sleep(3000);  
  
    //pause  
    webDriver.findElement(By.className("IeLnf2wUHVQxhzBcBoM")).click();  
    Thread.sleep(3000);  
  
    //save to library  
    webDriver.findElement(By.xpath("/html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div/div[2]/main/section/div[3]/div[  
Thread.sleep(2000);  
  
    //verify that the proper episode has been added  
    webDriver.navigate().to("https://open.spotify.com/collection/episodes");  
    Thread.sleep(3000);  
    String latestAdd=webDriver.findElement(By.partialLinkText(episodeName)).getText();  
    assertEquals(latestAdd.toLowerCase(), episodeName.toLowerCase());  
}  
}
```

Test Name: Test Podcast Shareability

Description: Verify that user can generate a link to share podcast

Pre-condition(s):

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<ol style="list-style-type: none">1. Navigate to the podcast you wish to share2. Click on the share button.3. Select the option to share the playlist by copying link3. Assert that the copied link leads to the episode in question		The link to podcast has been successfully copied.	The link to podcast has been successfully copied.	PASS

Notes:

```

    }
    @Test
    @Order(3)
    void testShare() throws InterruptedException, UnsupportedFlavorException, IOException {
        Actions actions=new Actions(webDriver);
        //more
        webDriver.findElement(By.className("T0anrkk_QA4IAQL29get")).click();
        Thread.sleep(4000);

        //share

        WebElement share=webDriver.findElement(By.xpath("//html/body/div[14]/div/ul/li[2]/button"));
        actions.moveToElement(share).perform();
        Thread.sleep(3000);
        //copy link
        Thread.sleep(2000);
        webDriver.findElement(By.xpath("//html/body/div[14]/div/ul/li[2]/div/ul/li[1]/button")).click();
        Thread.sleep(5000);
        //test link
        Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
        Transferable transferable = clipboard.getContents(null);
        String clipboardLink = (String) transferable.getTransferData(DataFlavor.stringFlavor);
        assertTrue( clipboardLink.contains(webDriver.getCurrentUrl()));

        Thread.sleep(4000);
    }
}

```

3.10. Library Functionality

Testing that the user can access their library and see their collection based on certain category

Test Name: Verify Library				
Description: Test the functionality of accessing a library and browsing through categories.				
Pre-condition(s): A user is logged in and has a song/album/playlist selected				
Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1.Click “My Library” link 2. Browse depending on category 3.Verify that the correct link is accessed.		The library is accessed, and the user can see their collection, based on common category.	The library is accessed, and the user can see their collection, based on common category..	PASS
Notes:				

```

    System.setProperty("webdriver.chrome.driver", "C:\Users\zaraab\Downloads\chromedriver\chromedriver.exe");
    ChromeOptions options=new ChromeOptions();
    options.addArguments("--start-maximized");
    options.addArguments("--headless");
    options.addArguments("--user-data-dir=C:\Users\zaraab\AppData\Local\Google\Chrome\User Data");
    webDriver=new ChromeDriver(options);
    baseUrl="https://www.spotify.com";
    actions=new Actions(webDriver);

}

@AfterAll
static void tearDownAfterClass() throws Exception {
    webDriver.quit();
}

@Test
void accessAndBrowseLibrary() throws InterruptedException{
    webDriver.get(baseUrl);
    webDriver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

    //library
    webDriver.findElement(By.partialLinkText("Library")).click();

    //browse by category
    List<WebElement> list=webDriver.findElements(By.className("OEFWODerafYHgp09iL1A"));

    for(WebElement element : list){
        element.click();
        webDriver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
        assertTrue(webDriver.getCurrentUrl().contains(element.getText().toLowerCase()));
        Thread.sleep(2000);
    }
}

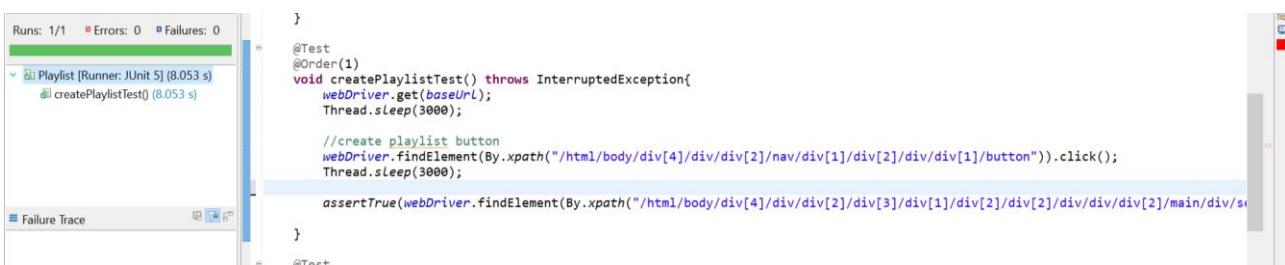
```

3.11. Playlist Functionality

Testing the playlist functionalities, creating a playlist, deleting it, sharing it, adding songs, etc....

Test Name: Verify Playlist Creation**Description:** Verify Playlist creation**Pre-condition(s):** A user is logged in to their account

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Click on the “Create Playlist” button 2. Assert that the library has been created		A new playlist is created and added to the user's library	A new playlist is created and added to the user's library	PASS

Notes:

The screenshot shows a JUnit test runner interface. At the top, it displays "Runs: 1/1" with "Errors: 0" and "Failures: 0". Below this, there is a tree view under the heading "Playlist [Runner: JUnit 5] (8.053 s)". The tree contains a single node "createPlaylistTest() (8.053 s)" which is marked with a green checkmark, indicating success. At the bottom of the interface, there is a "Failure Trace" section which is currently empty.

```
    }
    @Test
    @Order(1)
    void createPlaylistTest() throws InterruptedException{
        webDriver.get(baseUrl);
        Thread.sleep(3000);

        //create playlist button
        webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/nav/div[1]/div[2]/div[1]/button")).click();
        Thread.sleep(3000);

        assertTrue(webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div[2]/main/div/se
    }

    static
```

Test Name: Verify Song Addition to a Playlist**Description:** Test the ability for a user to add songs to an existing playlist**Pre-condition(s):** A user is logged in to their account, and has at least one existing playlist

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
1. Navigate to the playlist where you want to add song. 2. Search for songs you want to add and select them 3. Click on “add” button 4. Verify that the song has been added.	Search song/artist= “bts”	The selected songs are added to the selected playlist	The selected songs are added to the selected playlist	PASS

Notes:

Finished after 10.248 seconds

Runs: 1/1 Errors: 0 Failures: 0

```

    @Test
    @Order(1)
    void createPlaylistTest() throws InterruptedException{
        webDriver.get(baseUrl);
        Thread.sleep(3000);

        //create playlist button
        webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/nav/div[1]/div[2]/div/div[1]/button")).click();
        Thread.sleep(3000);

        assertTrue(webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div[2]/div/div[2]/main/div/sec
    }

    @Test
    @Order(2)
    void addSongs() throws InterruptedException{
        webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div[2]/main/div/section/div[1]
        Thread.sleep(3000);
        List<WebElement>addsElements=webDriver.findElements(By.cssSelector("[aria-label='Add to Playlist']"));
        for(int i=0;i<4;i++) {
            addsElements.get(i).click();
            Thread.sleep(3000);
        }
        sizeSong=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div[2]/div/div[2]/main/div/sec
        assertEquals(0,sizeSong);
    }
}

```

Test Name: Verify Song Deletion from Playlist

Description: Test the ability for a user to delete songs from an existing playlist

Pre-condition(s): A user is logged in to their account

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<p>1. Navigate to the playlist from where you want to delete the song.</p> <p>2. Search for song you want to delete and select it</p> <p>3. Press delete key</p> <p>4. Verify that the song has been deleted.</p>		The selected song is deleted from the selected playlist	The selected song is deleted form the select playlist	PASS

Notes:

Runs: 5/6 Errors: 0 Failures: 0

```

    @Test
    @Order(3)
    void deleteSongs() throws InterruptedException{
        WebElement firstElement=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div[2]/div/div[2]/main/div/sec
        actions.moveToElement(firstElement).click().keyDown(Keys.DELETE).perform();
        Thread.sleep(3000);

        sizeSong=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div/div[2]/div/div[2]/main/div/sec
        assertEquals(0,sizeSong);
    }
}

```

Test Name: Test Playlist Editing

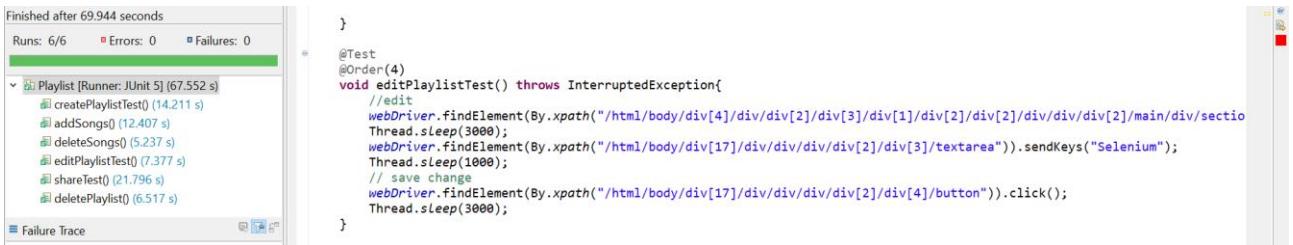
Description: Verify that you can edit playlist

Pre-condition(s): A user is logged in to their account and has at least one playlist

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:

1. Navigate to your playlist	Text= “Selenium”	The playlist has been edited	The playlist has been edited	PASS
2. Click on the edit button				
3. Once a popup appears, add text				
4. Save it				

Notes:



Test Name: Test Sharing Link to a Playlist

Description: Verify sharing link to a playlist

Pre-condition(s): A user is logged in to their account, and has at least one playlist

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<p>1. Navigate to the playlist you wish to share</p> <p>2. Click on the share button located on the playlist.</p> <p>3. Select the option to share the playlist by copying link</p> <p>3. Assert that the copied link leads to the playlist in question</p>	,	The link to playlist has been successfully copied.	The link to playlist has been successfully copied.	PASS

Notes:

Runs: 6/6 Errors: 0 Failures: 0

```

@Test
@Order(5)
void shareTest() throws InterruptedException, UnsupportedFlavorException, IOException{
    Thread.sleep(2000);
    //more
    webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/div[3]/div[1]/div[2]/div[2]/div/div/div[2]/main/div/section[1]"));
    Thread.sleep(5000);
    //share
    WebElement share=webDriver.findElement(By.xpath("//html/body/div[15]/div[1]/div[7]/button"));
    actions.moveToElement(share).perform();
    Thread.sleep(3000);
    //copy link
    Thread.sleep(2000);
    webDriver.findElement(By.xpath("//html/body/div[15]/div[1]/div[7]/div[1]/li[1]/button")).click();
    Thread.sleep(5000);
    //test link
    Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
    Transferable transferable = clipboard.getContents(null);
    String clipboardLink = (String) transferable.getTransferData(DataFlavor.stringFlavor);
    assertTrue( clipboardLink.contains(webDriver.getCurrentUrl()));

    Thread.sleep(4000);
}

```

Test Name: Test Playlist Deletion

Description: Verify the ability for a user to delete an existing playlist

Pre-condition(s): A user is logged in to their account and has at least one playlist

Test Steps:	Test Data:	Expected Result:	Actual Result:	Status:
<p>1.Locate the playlist you want to delete</p> <p>2.Click on the playlist and press the delete key</p> <p>3. Click on the confirmation button in the dialog to delete the playlist.</p> <p>4. Wait for the page to refresh and assert that the playlist is no longer present.</p>		The selected playlist is deleted from the user's library	The selected playlist is deleted from the user's library	PASS

Notes:

Runs: 6/6 Errors: 0 Failures: 0

```

Transferable transferable = clipboard.getContents(null);
String clipboardLink = (String) transferable.getTransferData(DataFlavor.stringFlavor);
assertTrue( clipboardLink.contains(webDriver.getCurrentUrl()));

Thread.sleep(4000);
}

@Test
@Order(6)
void deletePlaylist() throws InterruptedException{

//select and press delete
WebElement playlist=webDriver.findElement(By.xpath("//html/body/div[4]/div/div[2]/nav/div[1]/div[2]/div/div[5]/div/div[4]/div/div[2]/div[1]/div[1]/div[2]/div[2]/div[1]/div[2]/div[2]"));
actions.moveToElement(playlist).click().keyDown(Keys.DELETE).perform();
Thread.sleep(3000);

//confirm
assertTrue(webDriver.findElement(By.xpath("//html/body/div[17]/div/div/div/h2")).getText().toLowerCase().contains("delete"));
webDriver.findElement(By.xpath("//html/body/div[17]/div/div/div/button[2]")).click();

}

```

4. Conclusion

4.1. Testing Summary

Testing Tool	Total Tests	Passed Tests	Failed Tests
Junit, Selenium Web Driver, aXe	32	31	1 – testing Accessibility

4.2. Final Thoughts

In conclusion, the Spotify application was found to be well-implemented, with a user-friendly interface and a wide range of features. However, during testing, some issues were identified in terms of accessibility for users and compatibility with different screen readers, which suggests that some improvements could be made. Recommendations for the site would include conducting further testing in these areas and addressing any issues that are identified.

Overall, the application provides a seamless music streaming experience for users, with the ability to search for music, create playlists, and listen to radio stations.