**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

Contents

[1. Introduction 3](#_Toc125259096)

[1.1. About the Project 3](#_Toc125259097)

[1.2. Project Functionalities and Screenshots 3](#_Toc125259098)

[2. Test Plan 3](#_Toc125259099)

[2.1. Scope 3](#_Toc125259100)

[2.2. Testing Environment and Tools 4](#_Toc125259101)

[3. Test Execution 4](#_Toc125259102)

[3.1. Sing Up 4](#_Toc125259103)

[3.2. Login Functionality 6](#_Toc125259104)

[3.3. Playback Functionality 8](#_Toc125259105)

[3.4. Search and Browse Functionality 9](#_Toc125259106)

[3.5. Localization 11](#_Toc125259107)

[3.6. Accessibility 12](#_Toc125259108)

[3.7. Radio Functionality 13](#_Toc125259109)

[3.8. Profile Management 16](#_Toc125259110)

[3.9. Podcast Functionality 17](#_Toc125259111)

[3.10. Library Functionality 17](#_Toc125259112)

[3.11. Playlist Functionality 18](#_Toc125259113)

[4. Conclusion 22](#_Toc125259114)

[4.1. Testing Summary 22](#_Toc125259115)

[4.2. Final Thoughts 22](#_Toc125259116)

# 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/**](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.

# 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, browse and search for music, playlists, and podcasts, as well as the ability to follow and unfollow other users, view and edit their profile, and manage their playlists. We will also be performing accessibility testing, using tools such as aXe to verify that the website is accessible. 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

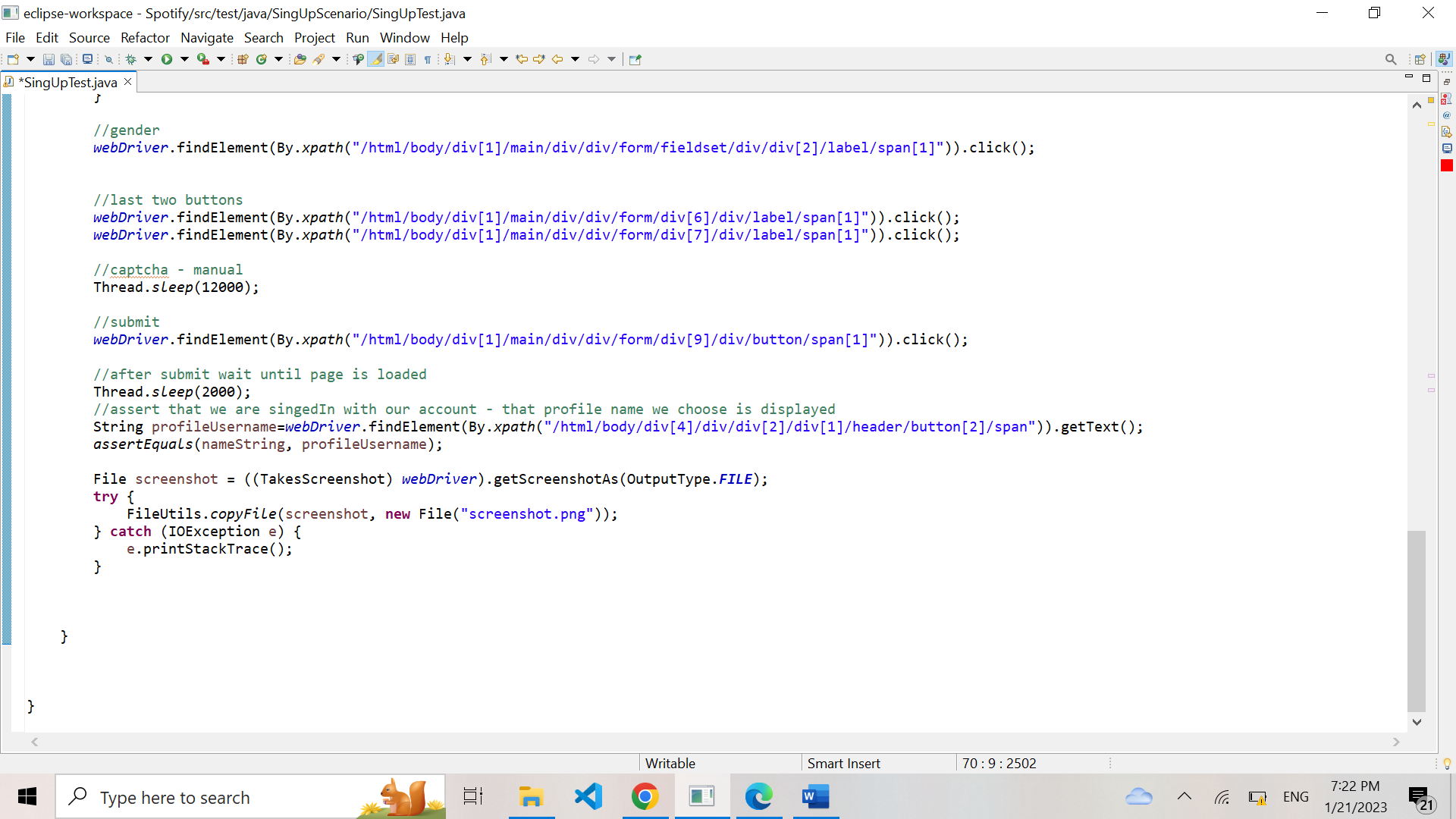
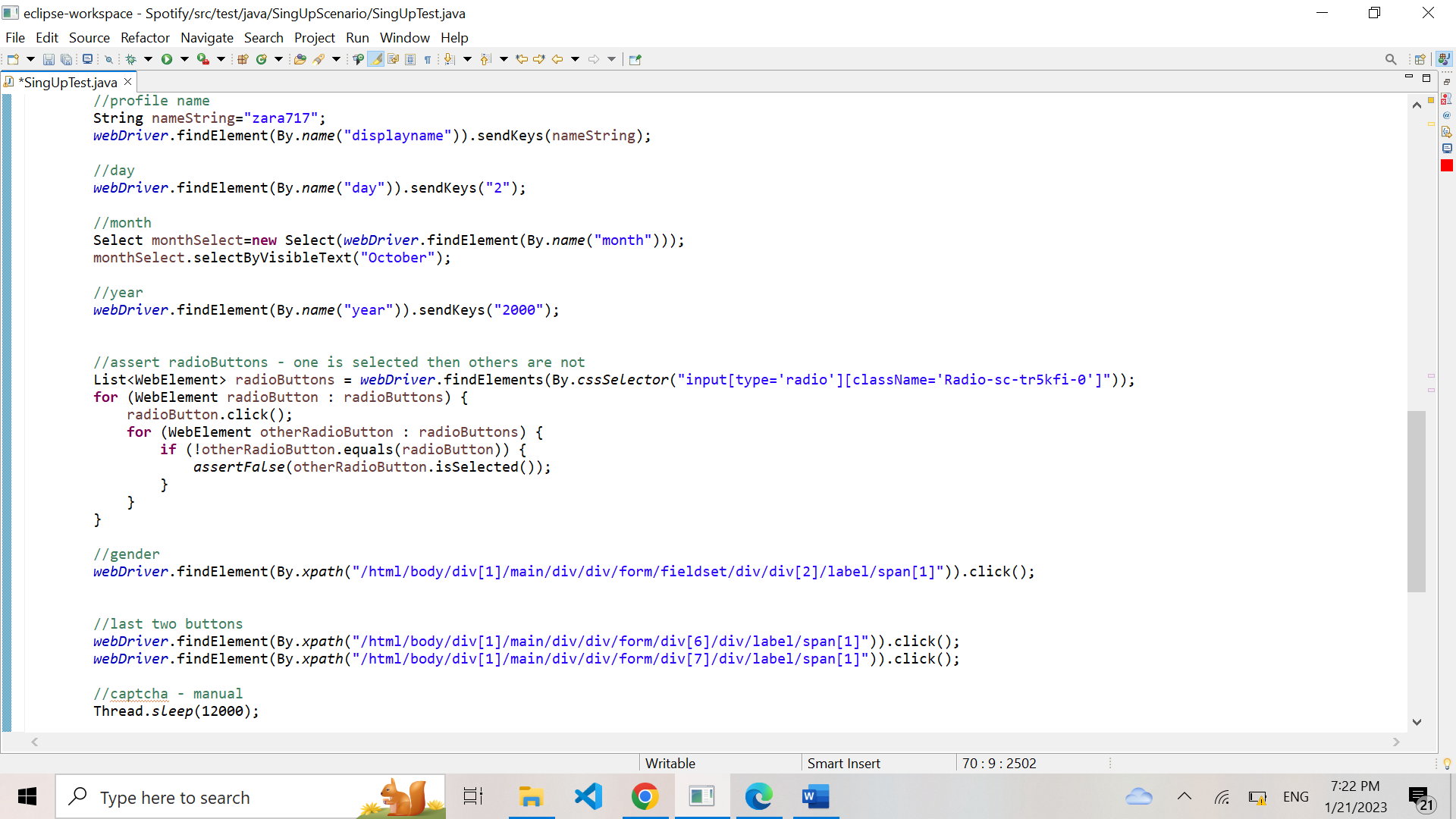
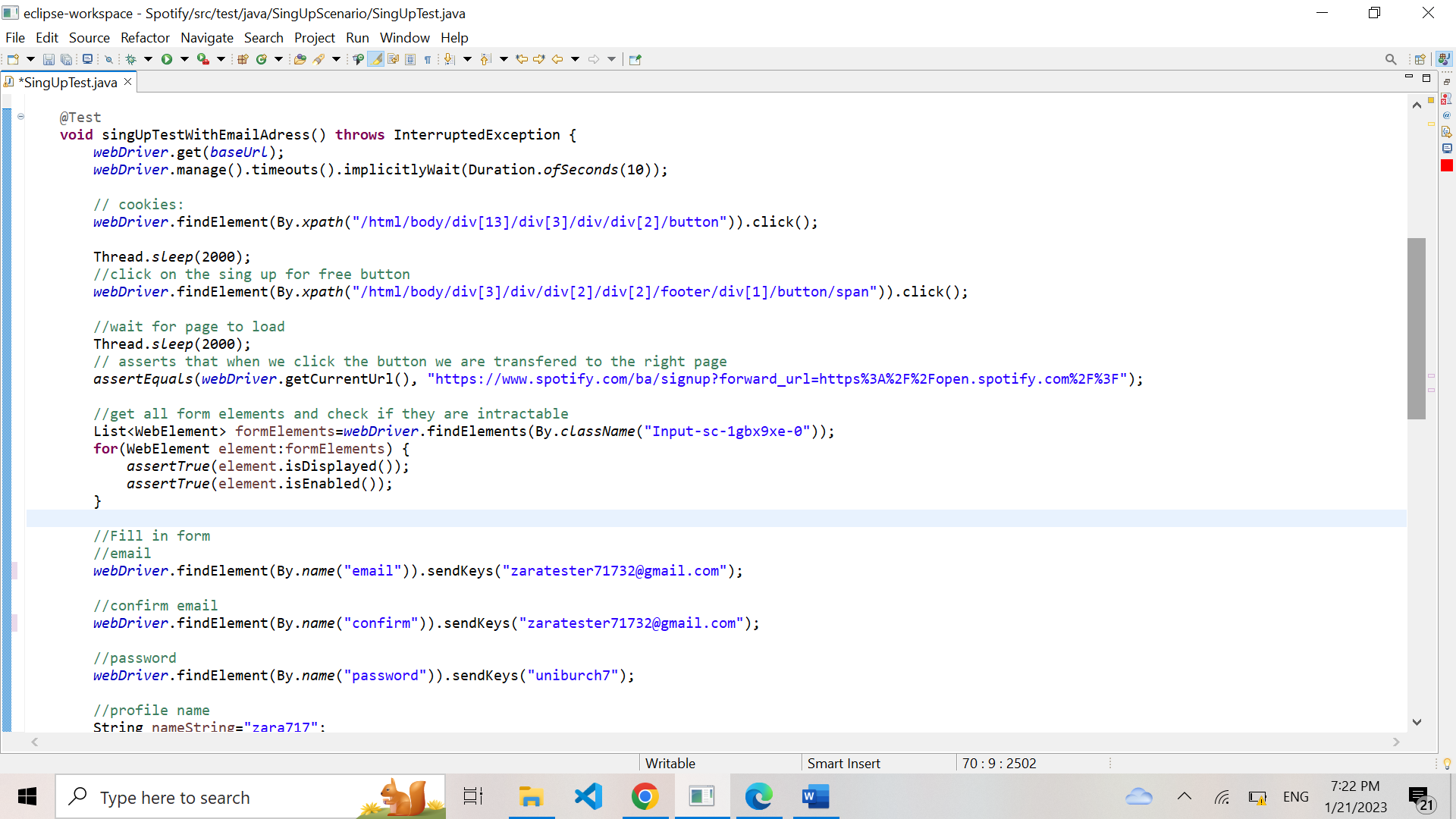
In this project, we will be using Selenium WebDriver, a widely-used open-source tool for automating web browsers, as our main testing tool. We will also be using JUnit, a popular open-source testing framework for Java, to write and run our test cases. Additionally, we will be using the selenium-axe library 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.

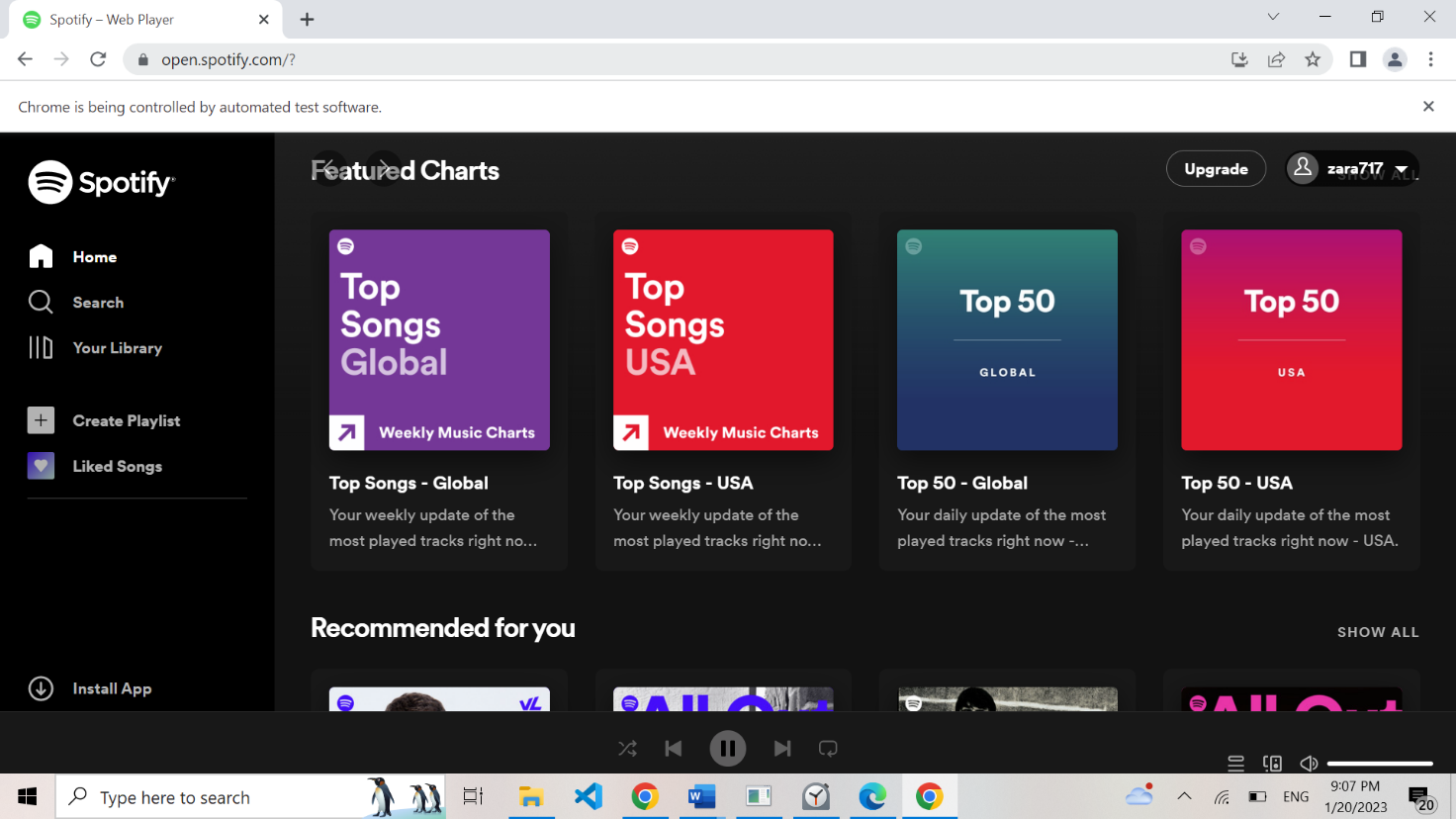
# 3. Test Execution

## 3.1. Sing Up

User want to create a personalized account for Spotify.

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

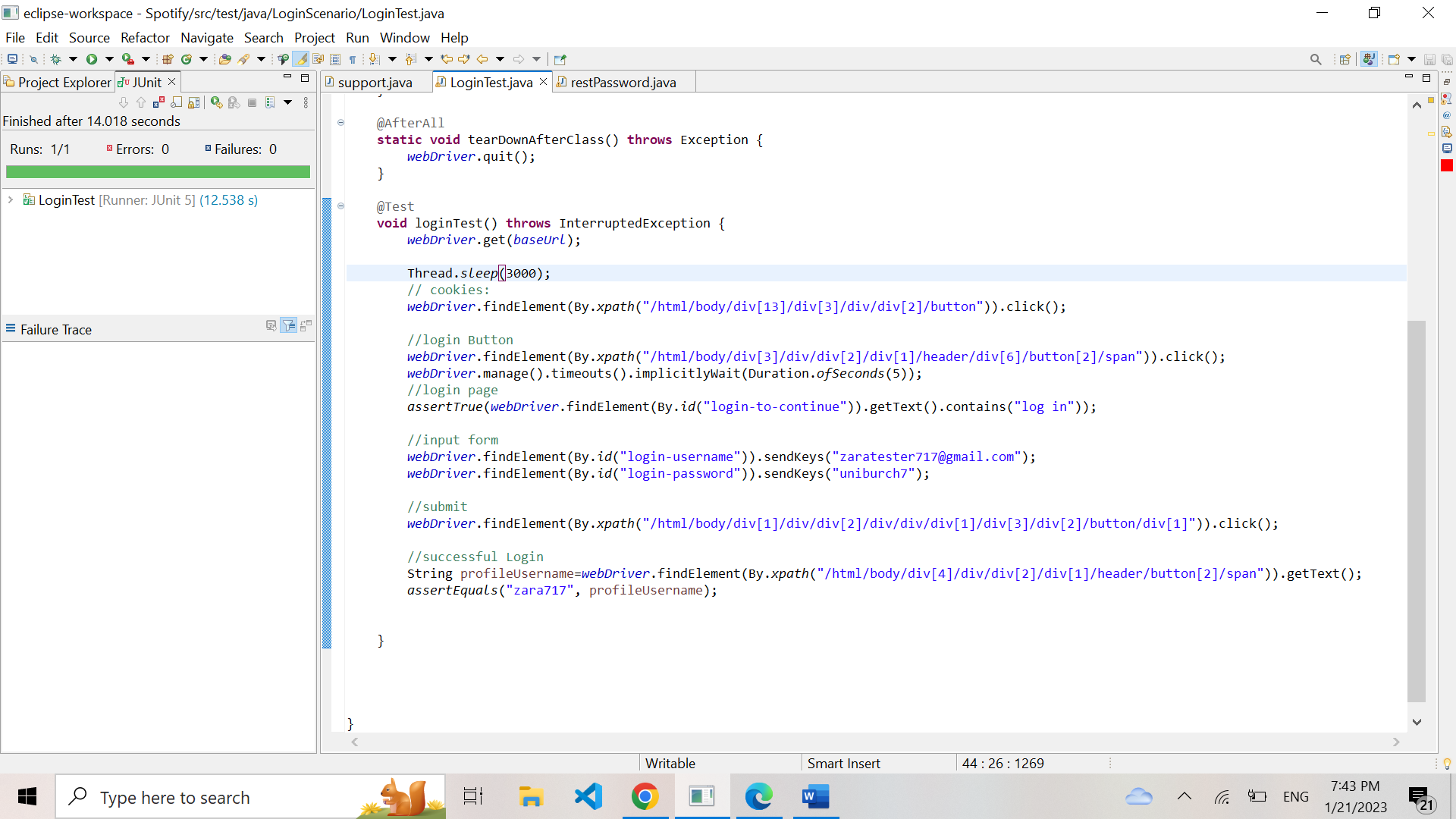




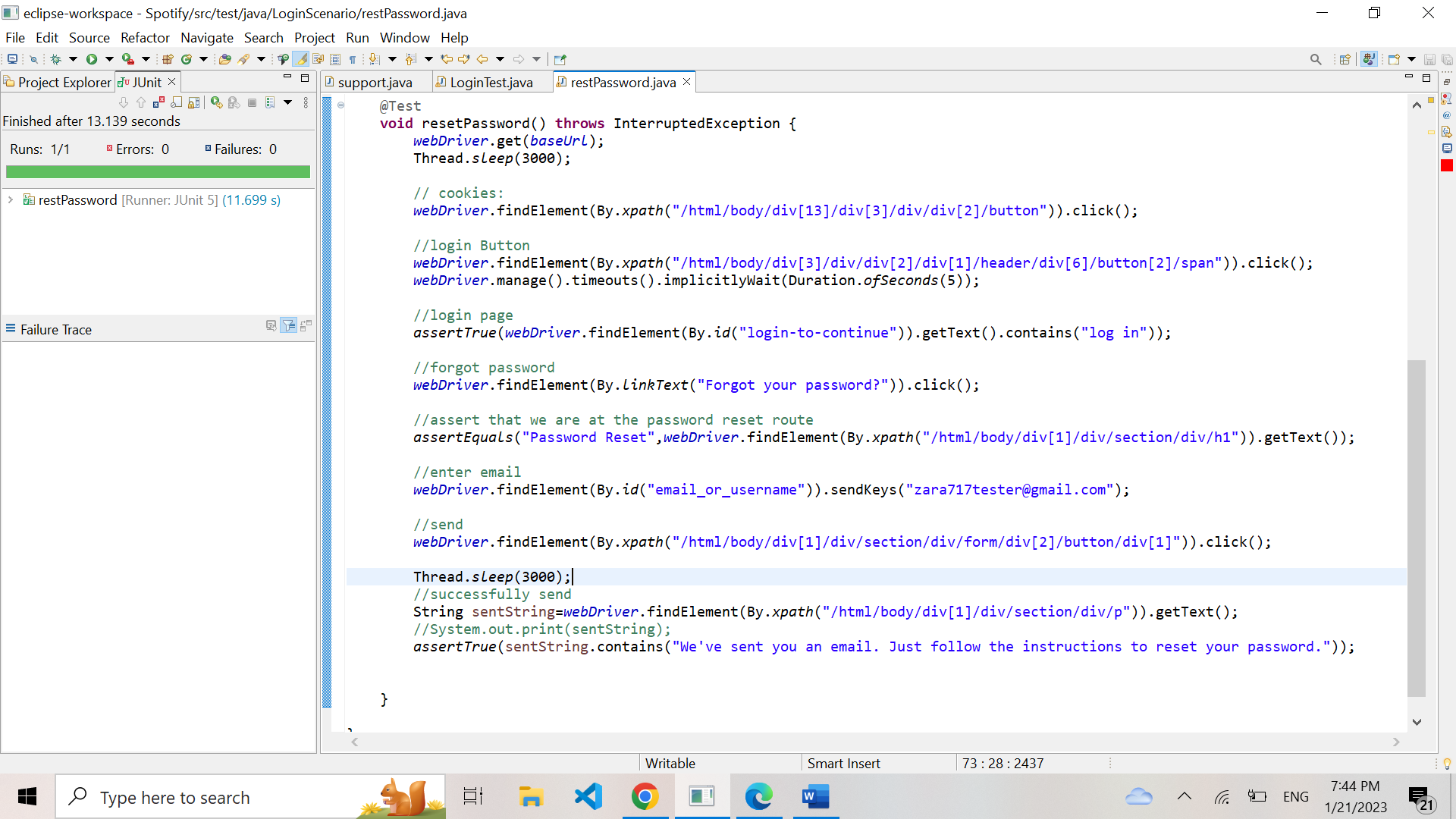
## 3.2. Login Functionality

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): The user has an exciting account with Spotify** | | | | |
| **Test Steps:**  1.Open the Spotify website and click on the “Login” button.  2. Fill in the required fields.  3. Click on the “Login” button. | **Test Data:**  Email=” zaratester717@gmail.com”,  Password=uniburch7, | **Expected Result:**  User is successfully logged in and redirected to their home page. | **Actual Result:**  User is successfully logged in and redirected to their home page. | **Status:**  PASS |
| **Notes:** | | | | |



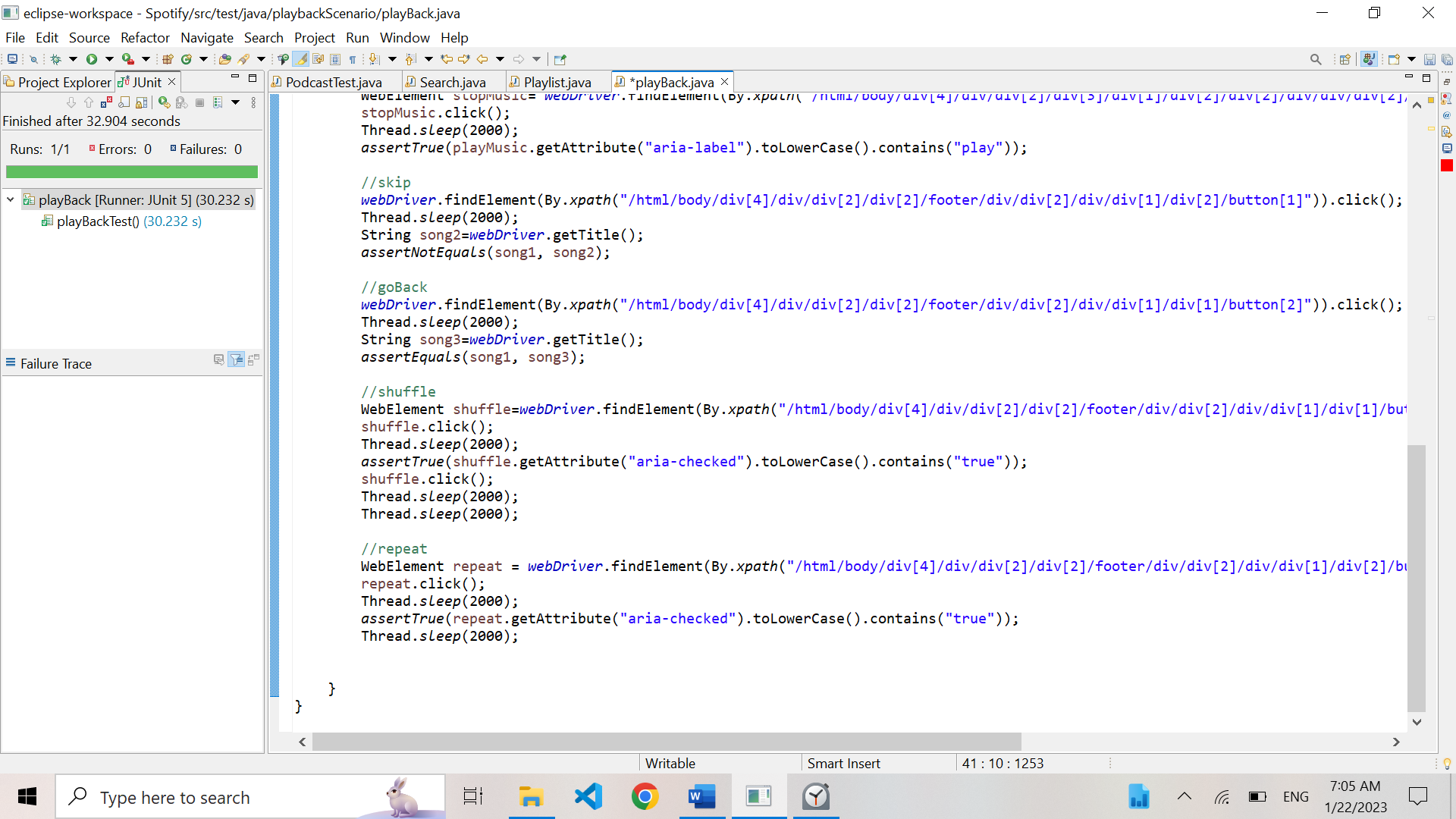
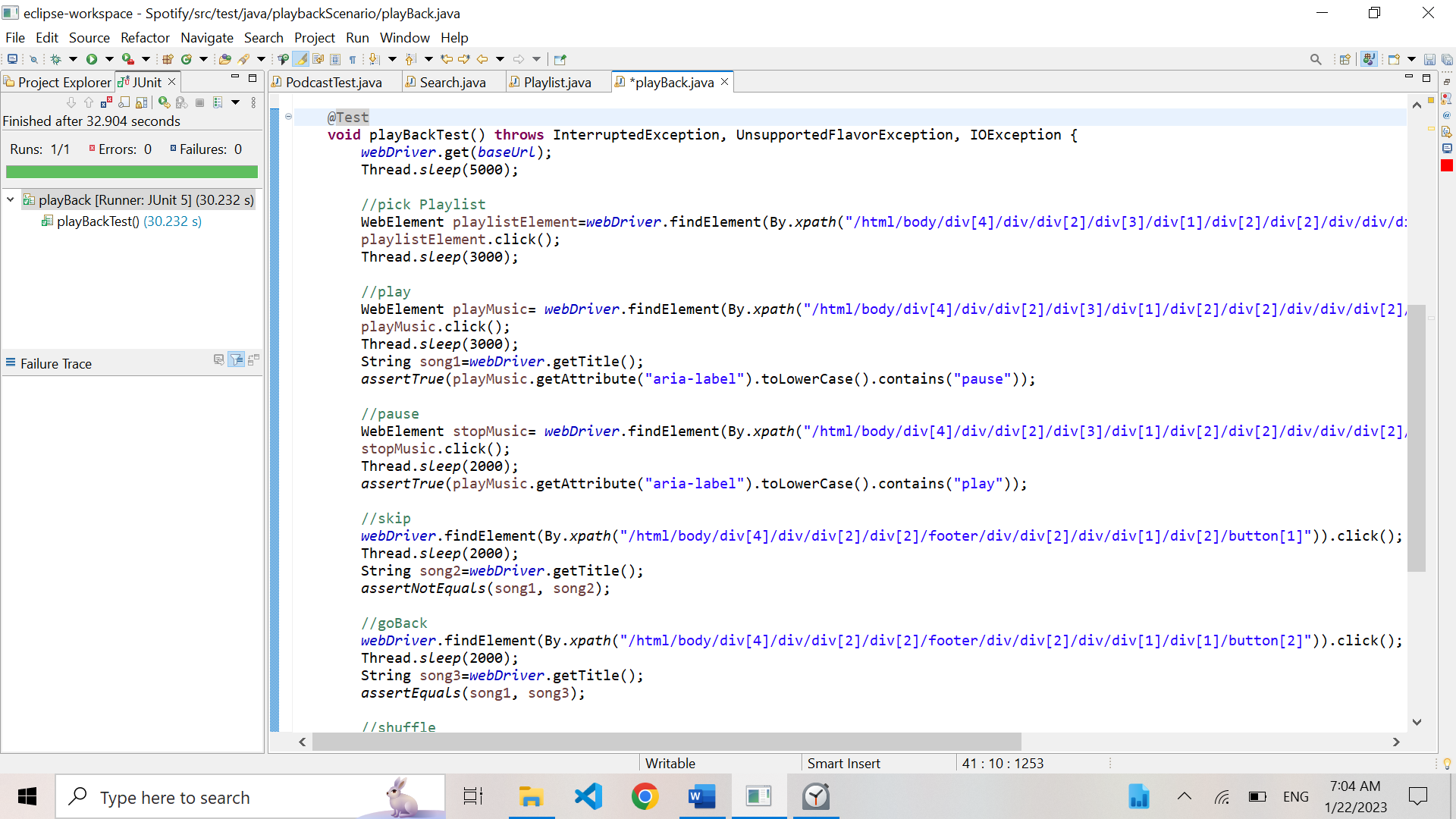
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Forgotten Password | | | | |
| **Description:** Verify that users can reset their password if they have forgotten it. | | | | |
| **Pre-condition(s): The user has an exciting account with Spotify** | | | | |
| **Test Steps:**  1.Open the Spotify website and click on the “Login” button.  2. Click on the "forgot password" link.  3.Enter a valid email and click the "send” button.  4.Confirm that an email has been successfully send | **Test Data:**  Email=” zaratester717@gmail.com”, | **Expected Result:**  User is able to reset their password | **Actual Result:**  User is able to reset their password | **Status:**  PASS |
| **Notes:** | | | | |



## 3.3. Playback Functionality

User want to play a song

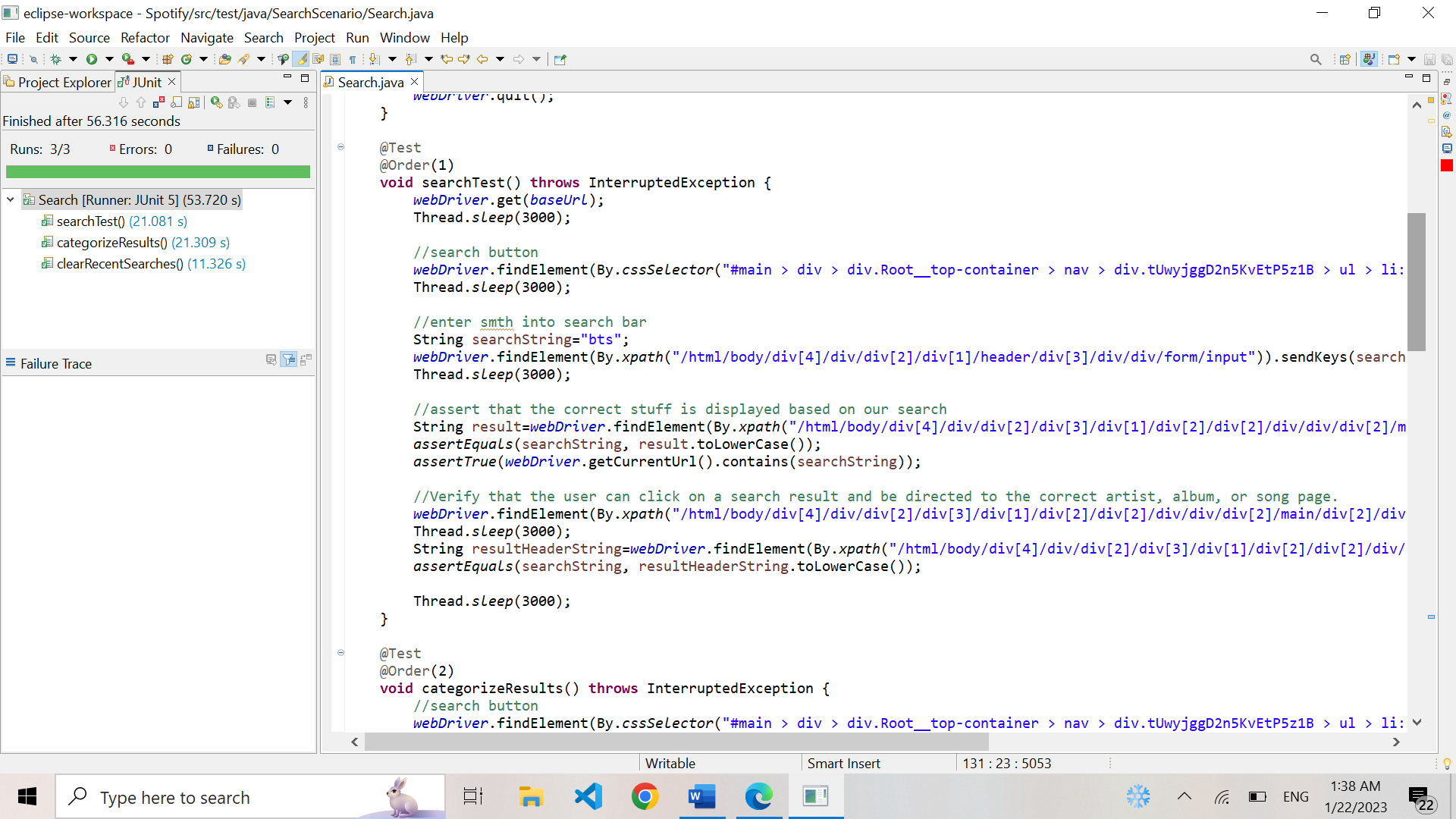
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Playback | | | | |
| **Description:**  Test the functionality of playing a song, pausing a song, skipping a song, shuffling songs, and repeating a song | | | | |
| **Pre-condition(s): A user has selected a playlist** | | | | |
| **Test Steps:**  1.Click “play”  2.Click “pause”  3.Click “skip”  4.Click “go Back”  5. Click “Shuffle”  5.Click “Repeat” | **Test Data:** | **Expected Result:**  The song plays, pauses, skips, shuffles and repeats. | **Actual Result:**  The song plays, pauses, skips, shuffles and repeats. | **Status:**  PASS |
| **Notes:** | | | | |



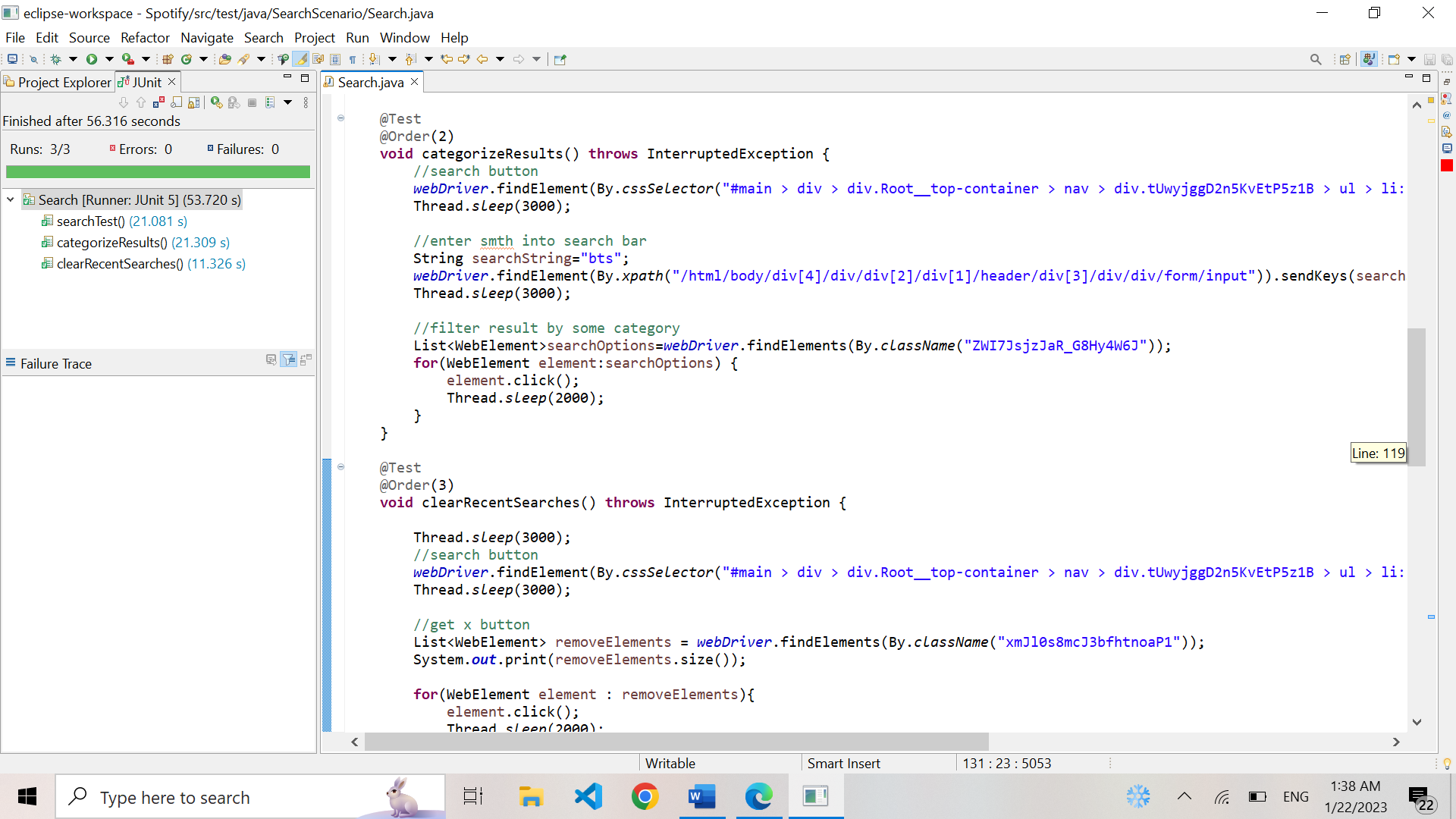
## 3.4. Search and Browse Functionality

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.

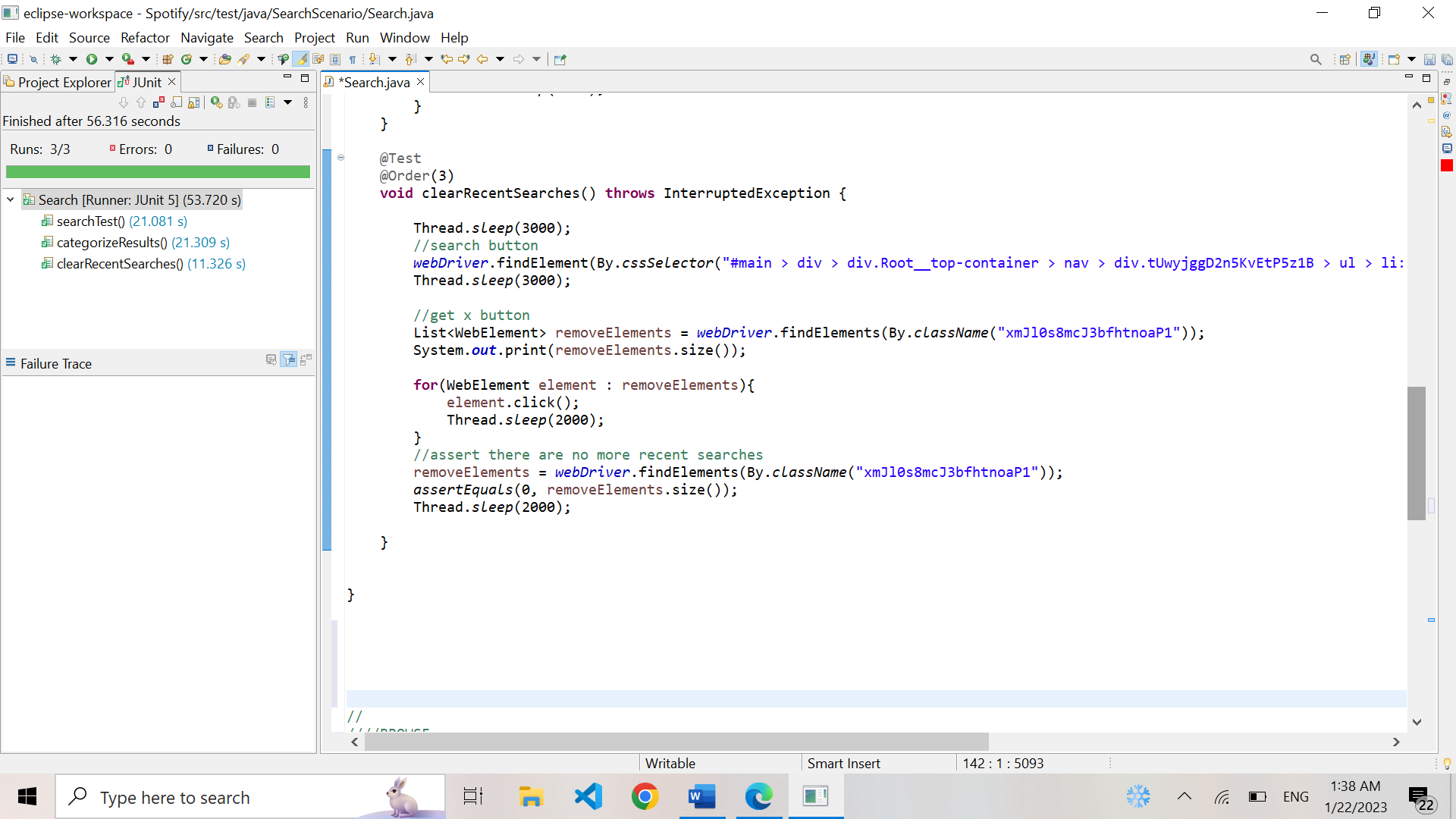
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Search Functionality | | | | |
| **Description:**  Test the search functionality and verify the correct results are displayed | | | | |
| **Pre-condition(s): A user is logged in to their account** | | | | |
| **Test Steps:**  1. Click the “search” button  2. Enter data into search bar  3. Verify that the correct results are displayed  4. Verify that the results are clickable and lead to correct path | **Test Data:**  Search data = “BTS” | **Expected Result:**  The search result should display correct result based on entered data. | **Actual Result:**  The search result should display correct result based on entered data. | **Status:**  PASS |
| **Notes:** | | | | |



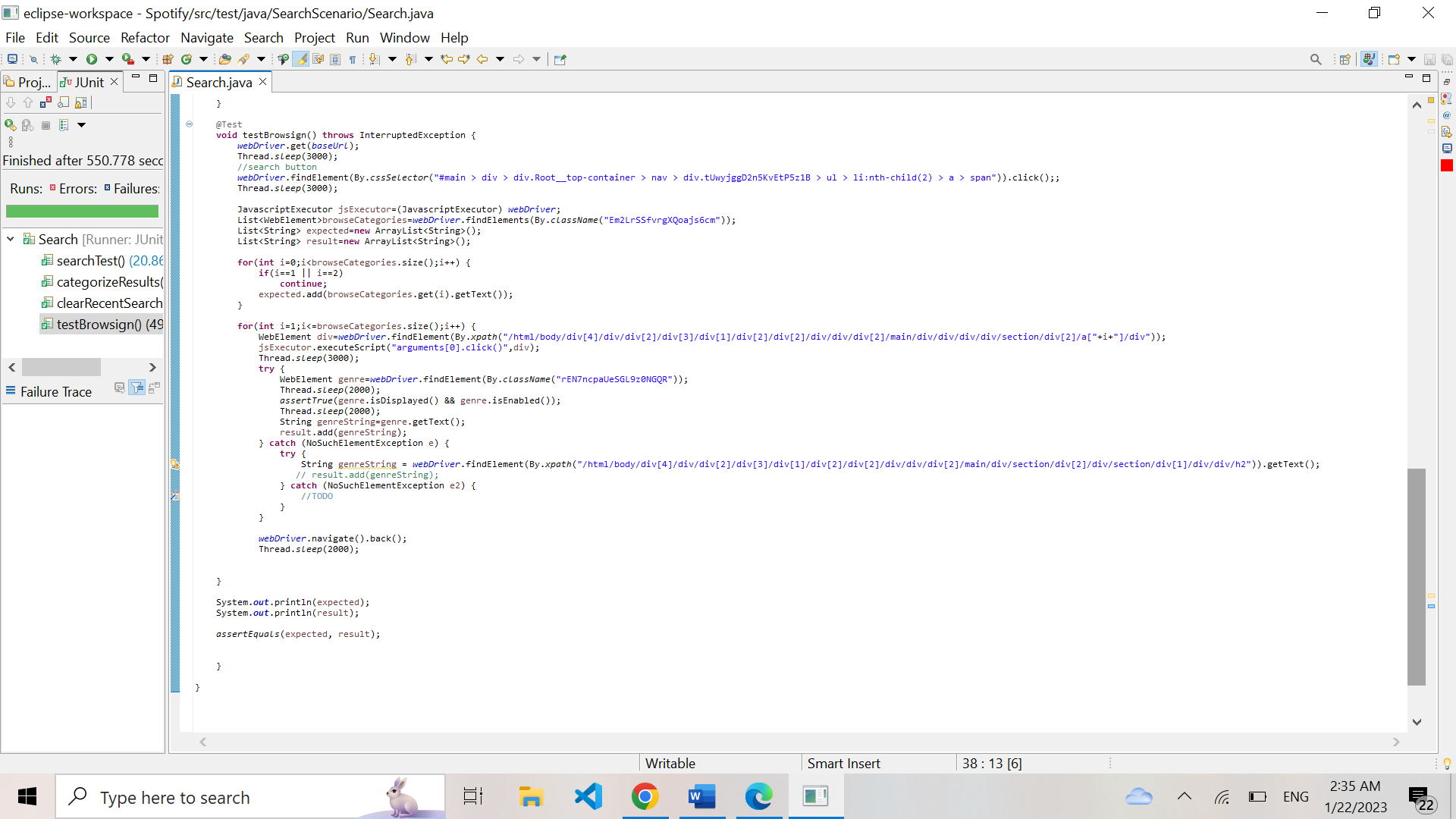
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Search by Categorizes | | | | |
| **Description:**  Test the functionality of searching by category (e.g., album, song.) | | | | |
| **Pre-condition(s): A user is logged in to their account** | | | | |
| **Test Steps:**  1. Click the “search” button.  2. Enter data into search bar  3. Test the feature to filter result by category | **Test Data:** | **Expected Result:**  The searched data should be correctly filtered and displayed by selected category | **Actual Result:**  The searched data is correctly filtered and displayed by selected category | **Status:**  PASS |
| **Notes:** | | | | |



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



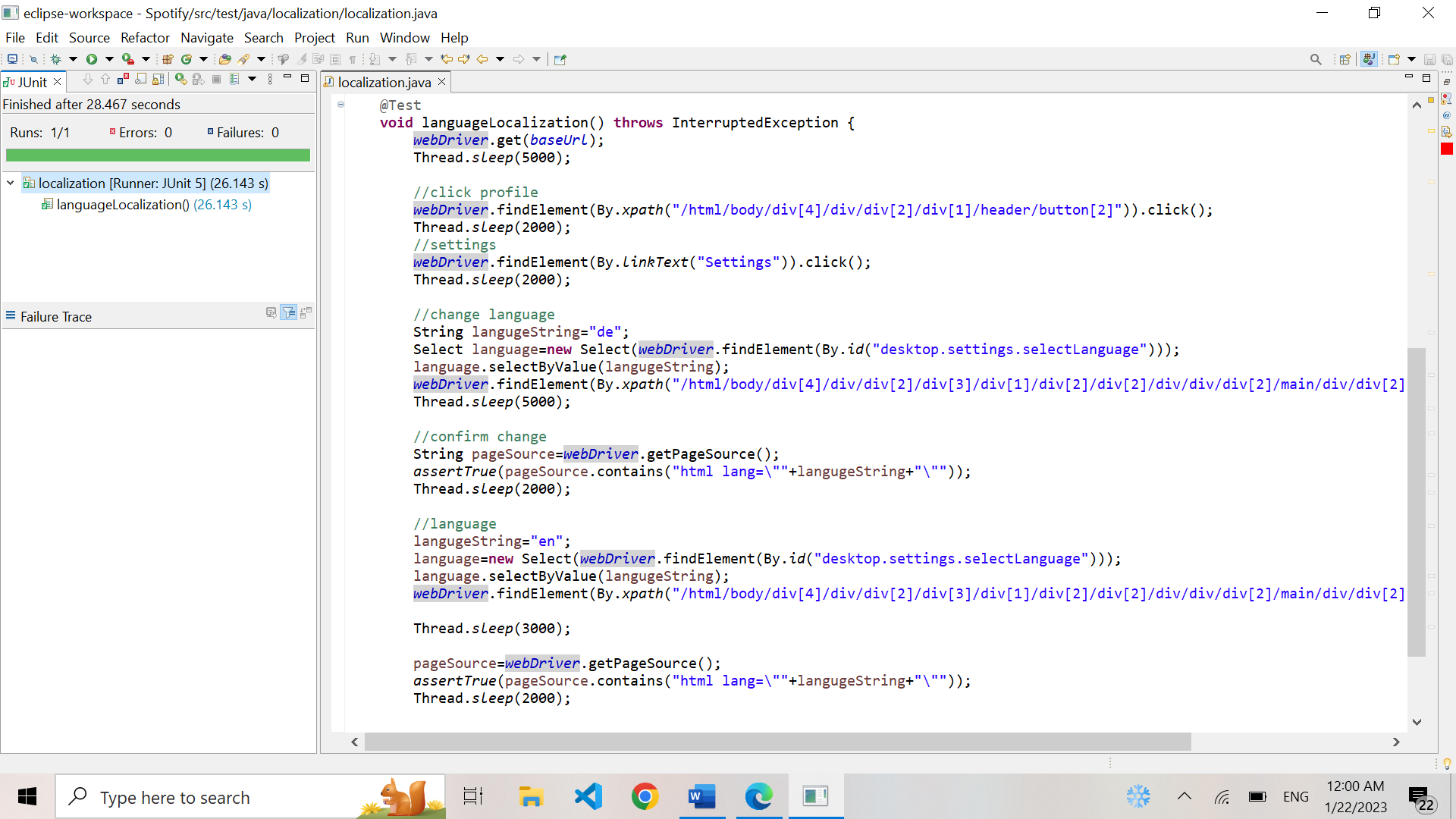
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Browsing Functionality | | | | |
| **Description:**  Test that user can browse through the music library by some genre/option | | | | |
| **Pre-condition(s): A user is logged in to their account** | | | | |
| **Test Steps:**  1. Click the search link  2. Verify that each prelisted option for “browsing” is clickable and leads to correct path | **Test Data:**  , | **Expected Result:**  The genres are all valid, clickable and lead to a correct path | **Actual Result:**  The genres are all valid, clickable and lead to correct a path | **Status:**  PASS |
| **Notes:** | | | | |



## 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

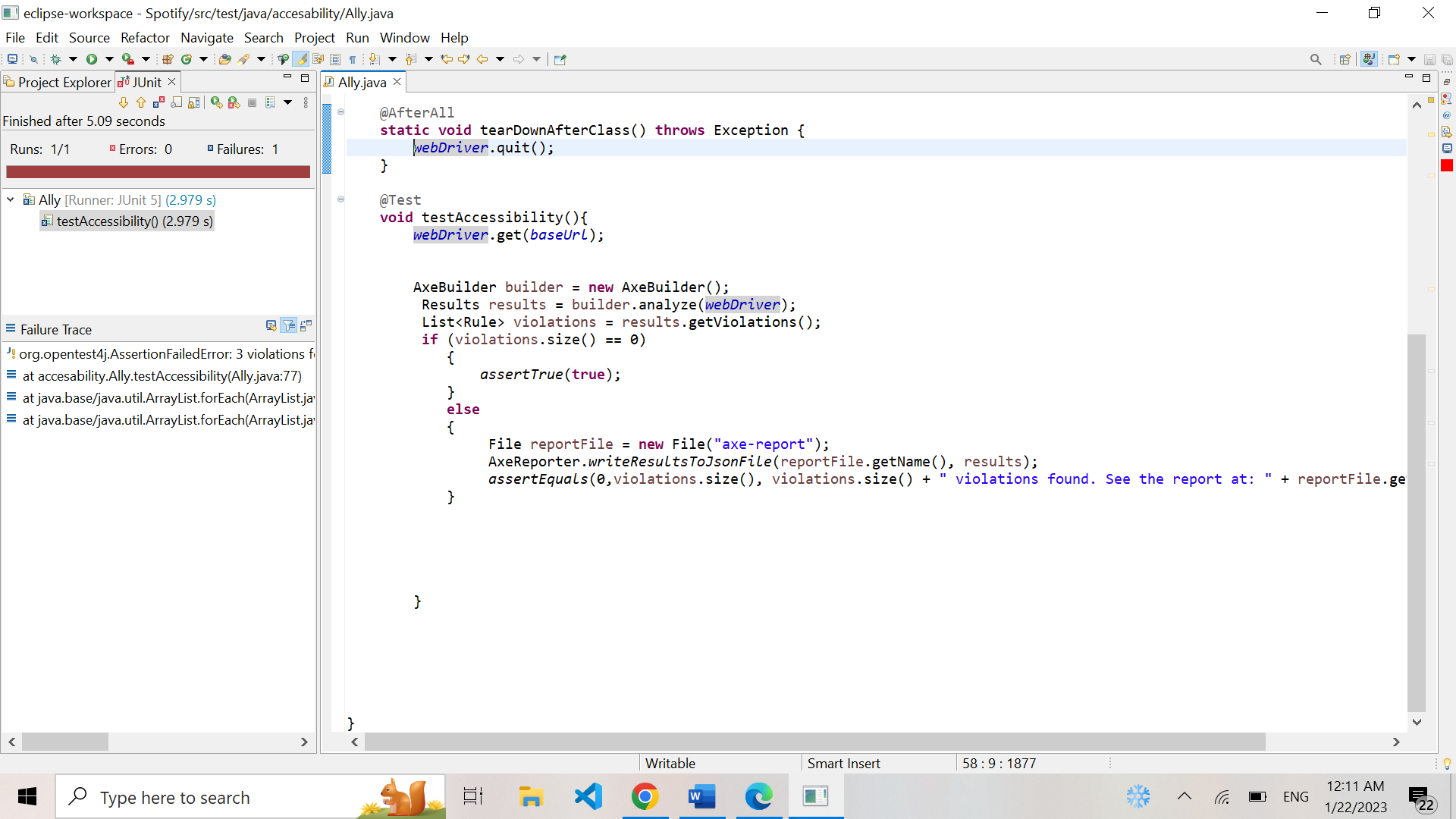
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Localization for Different Languages | | | | |
| **Description:**  Test that the website is properly localized for different languages | | | | |
| **Pre-condition(s): The website should be available for different languages** | | | | |
| **Test Steps:**  1.Open website  2.Go to settings  3.Change the language preference to a chosen language  4. Verify that the website is properly localized for the chosen language.  5. Repeat step 3 | **Test Data:**  Language preference= “de” and “en” | **Expected Result:**  The website should be properly localized for the chosen language. | **Actual Result:**  The website is properly localized for the chosen language. | **Status:**  PASS |
| **Notes:** | | | | |



## 3.6. Accessibility

This test scenario aims to check if the website is accessible for users with disabilities and compatible with different screen readers, as well as keyboard navigation.

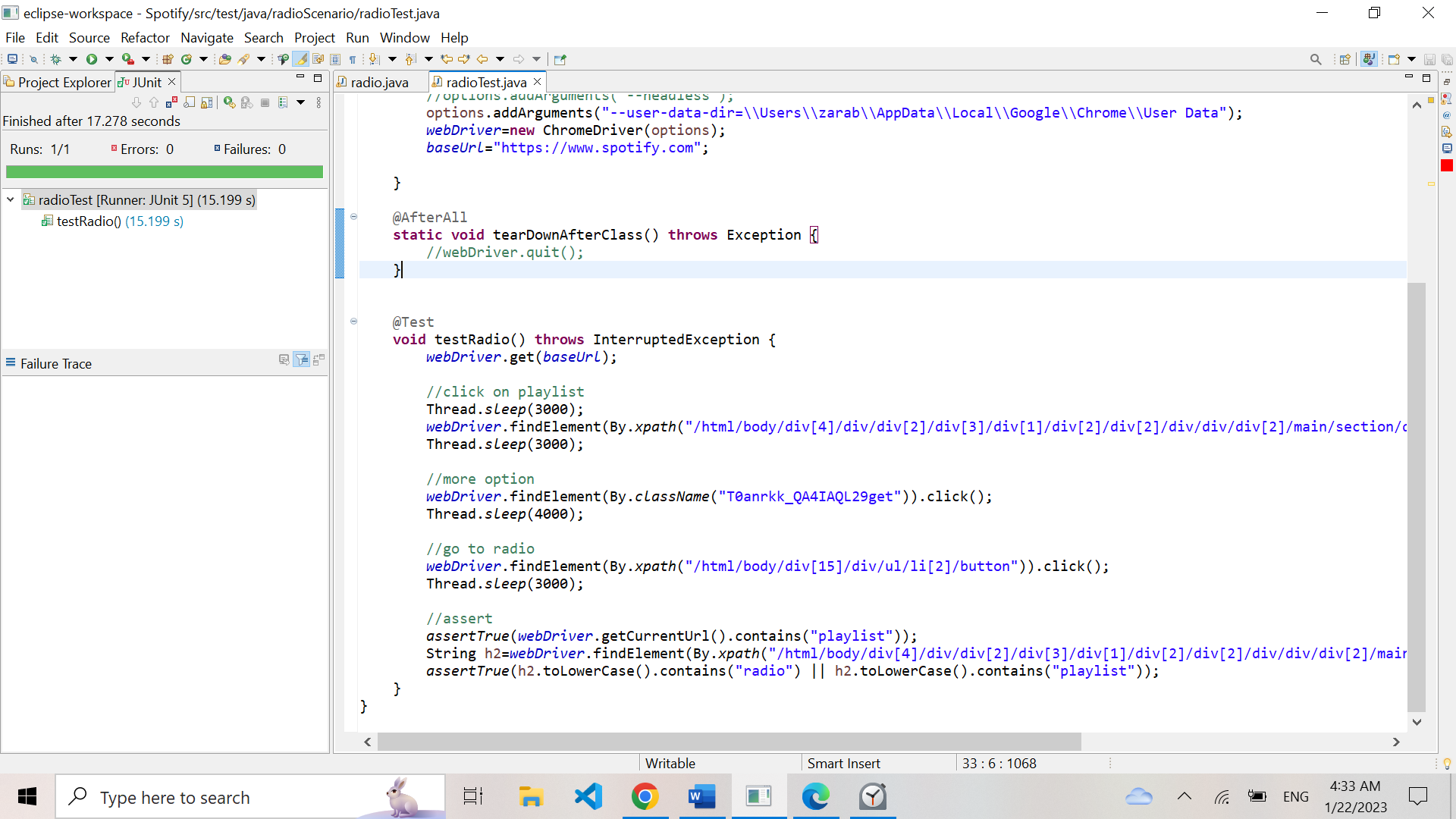
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Website’s Accessibility | | | | |
| **Description:**  Verify that the website is accessible for users with disabilities / compatible with different screen readers / compatible with keyboard navigation | | | | |
| **Pre-condition(s): None** | | | | |
| **Test Steps:**  1. Navigate to the website  2.Create an instance of Axe Builder class  3.Use analyze () method to analyze the website  4.Retrieve the list of violations | **Test Data:** | **Expected Result:**  The website should have no accessibility issues and be fully navigable and usable for users with disabilities, as well as compatible with different screen readers and keyboard navigation. | **Actual Result:**  The has accessibility issues. | **Status:**  FAIL |
| **Notes:** | | | | |



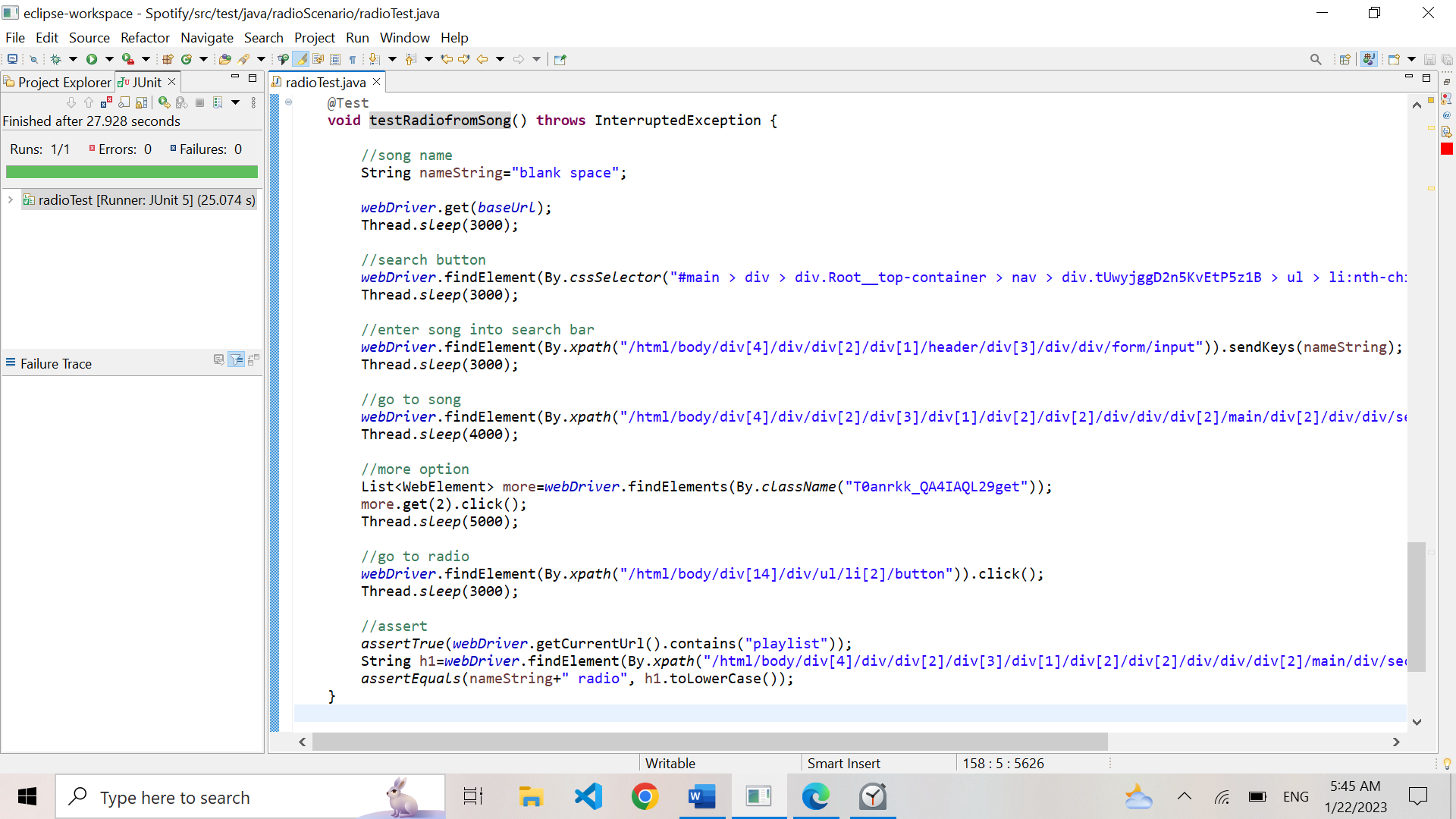
## 3.7. Radio Functionality

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.

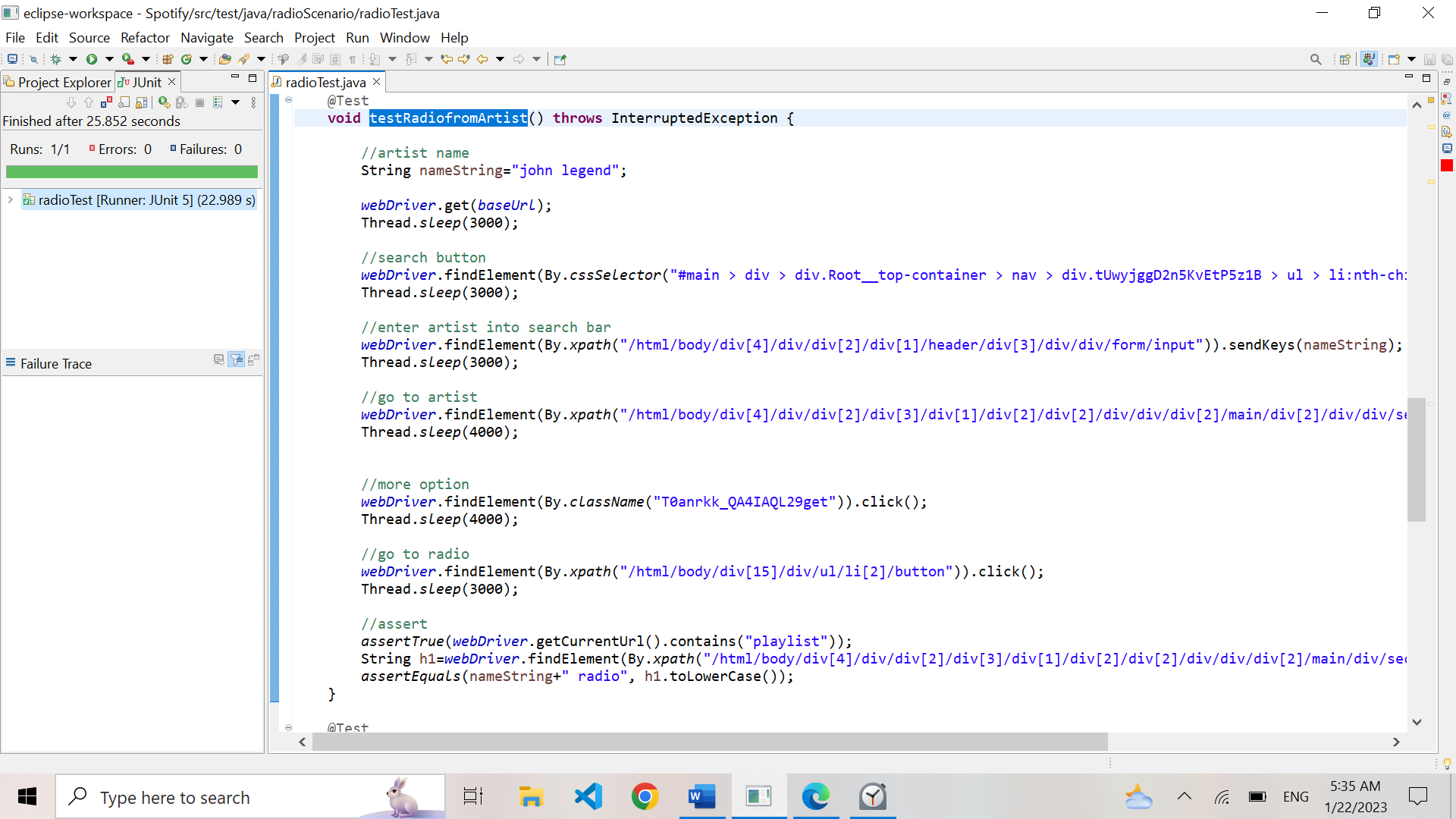
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Radio Creation from a Playlist | | | | |
| **Description:**  Test the functionality of creating a radio playlist from a certain playlist | | | | |
| **Pre-condition(s): A user is logged in and has a playlist selected** | | | | |
| **Test Steps:**  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 | **Test Data:** | **Expected Result:**  The user is able create and listen to a playlist based on a specific playlist. | **Actual Result:**  The user is able create and listen to a playlist based on a specific playlist. | **Status:**  PASS |
| **Notes:** | | | | |



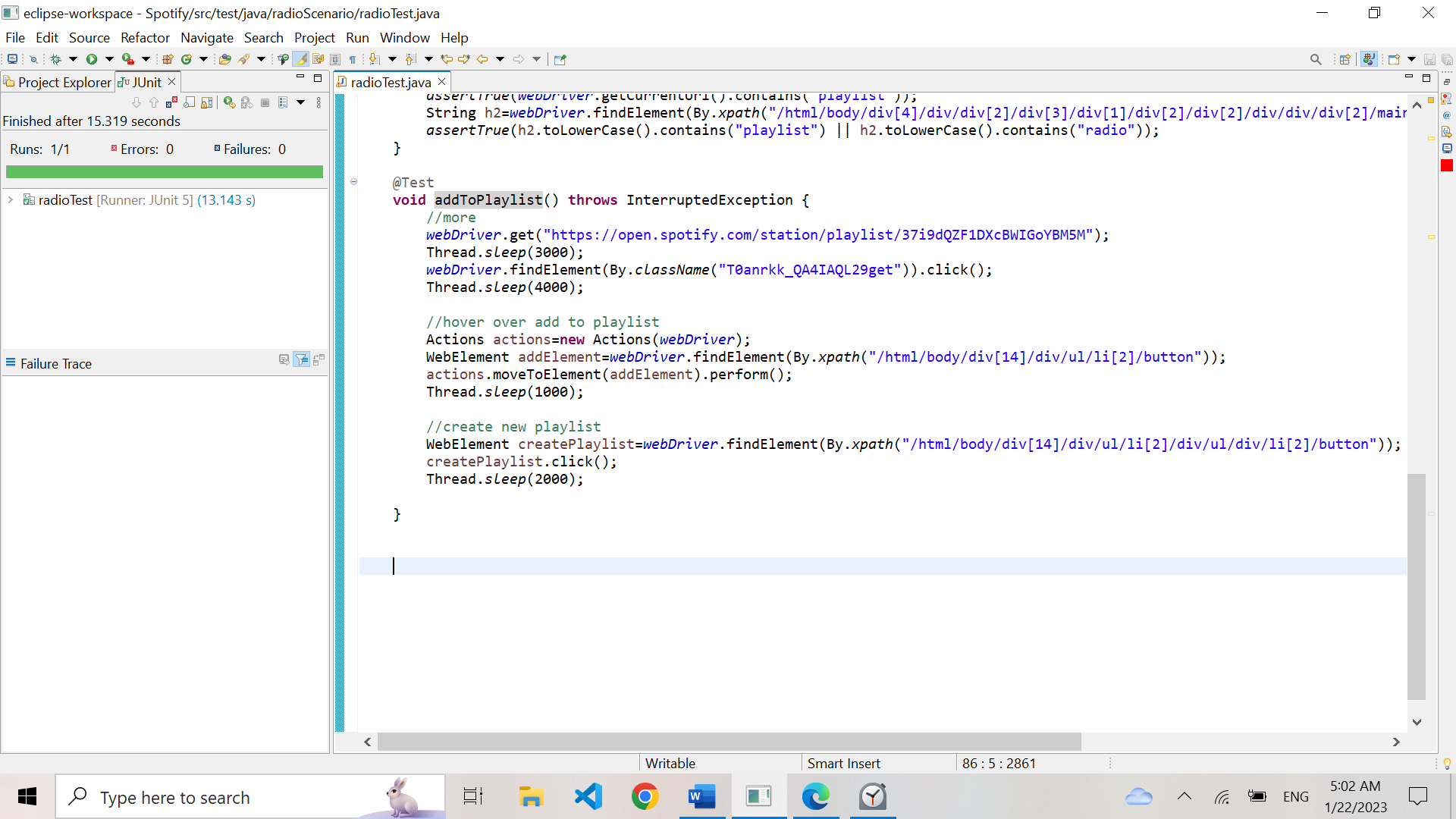
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Radio Creation from a Song | | | | |
| **Description:**  Test the functionality of creating a radio playlist from a certain song | | | | |
| **Pre-condition(s): A user is logged in and has a song selected** | | | | |
| **Test Steps:**  1. Select song  2. Click the “more” button  3.Click the “go to radio” button.  4.Verify that you are redirected to that song’s radio playlist | **Test Data:**  Name of Song = 1989 | **Expected Result:**  The user is able create and listen to a playlist based on a specific song. | **Actual Result:**  The user is able create and listen to a playlist based on a specific song. | **Status:**  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:**  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 | **Test Data:**  Name of Artist = John Legend | **Expected Result:**  The user is able create and listen to a playlist based on a specific artist. | **Actual Result:**  The user is able create and listen to a playlist based on a specific artist. | **Status:**  PASS |
| **Notes:** | | | | |



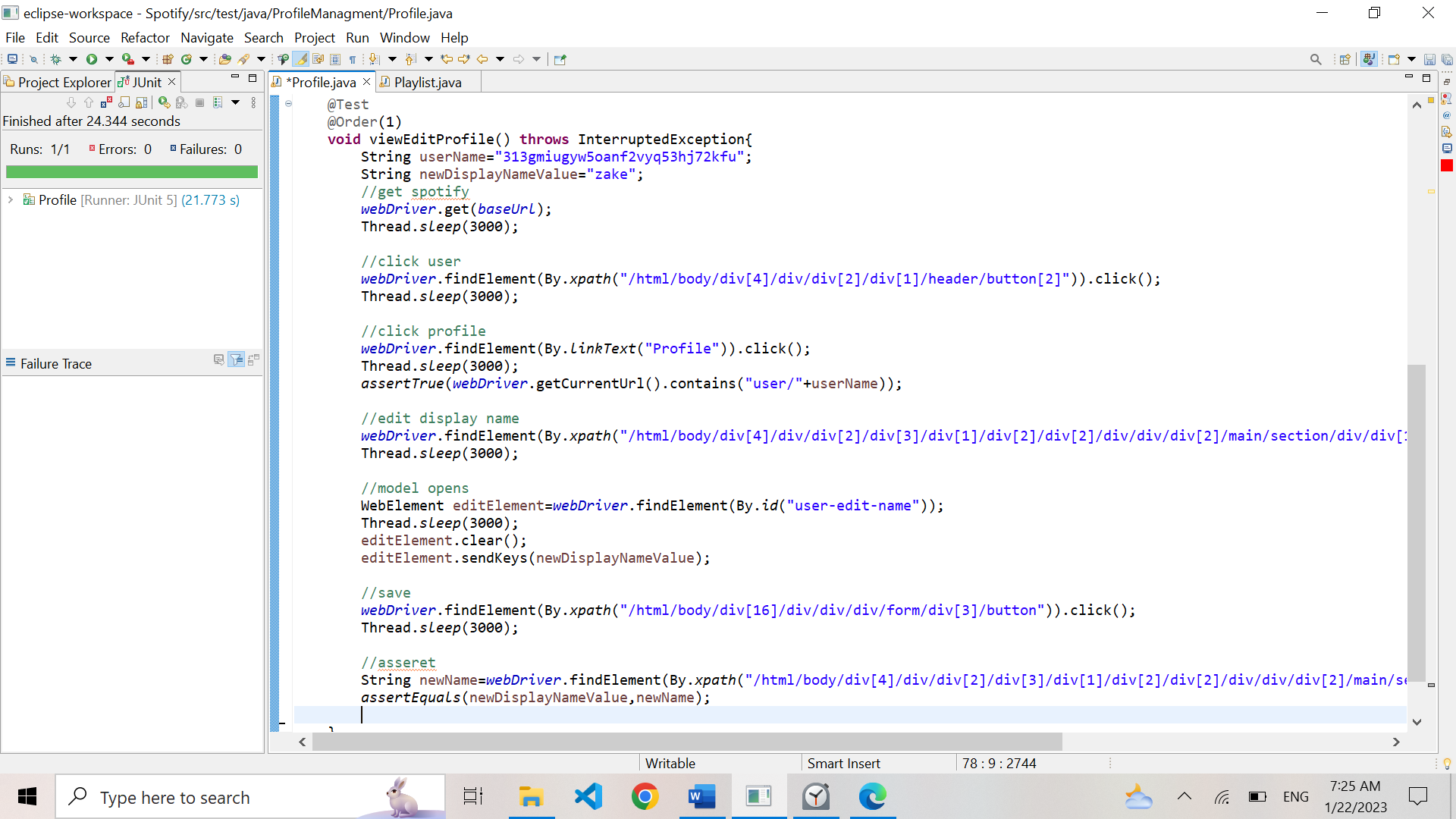
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Saving Radio to Personal Playlist | | | | |
| **Description:**  Test the functionality of saving a radio playlist to user’s personal playlist collection | | | | |
| **Pre-condition(s): A user is logged in and has a radio playlist created.** | | | | |
| **Test Steps:**  1. Go to radio playlist  2. Click “more” button  3. Click “add to playlist” button  4.Click “create new playlist” button  5. Wait for the confirmation message pop-up. | **Test Data:** | **Expected Result:**  The user is able save a radio playlist to their personal playlist collection | **Actual Result:**  The user is able save a radio playlist to their personal playlist collection | **Status:**  PASS |
| **Notes:** | | | | |



## 3.8. Profile Management

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Profile Update | | | | |
| **Description:**  Test that user can view and edit their profile | | | | |
| **Pre-condition(s): A user has an account and is logged in** | | | | |
| **Test Steps:**  1. Click on the “user” tab  2. Click on the “profile” link  3.Verify that the link has redirected us to the correct profile  4. Click on display name  5. Edit it and save it  6. Verify that the display name has been correctly modified | **Test Data:**  Username = “313gmiugyw5oanf2vyq53hj72kfu”  New display name = “zake” | **Expected Result:**  The user can view their profile and successfully edit it. | **Actual Result:**  The user can view their profile and successfully edit it. | **Status:**  PASS |
| **Notes:** | | | | |

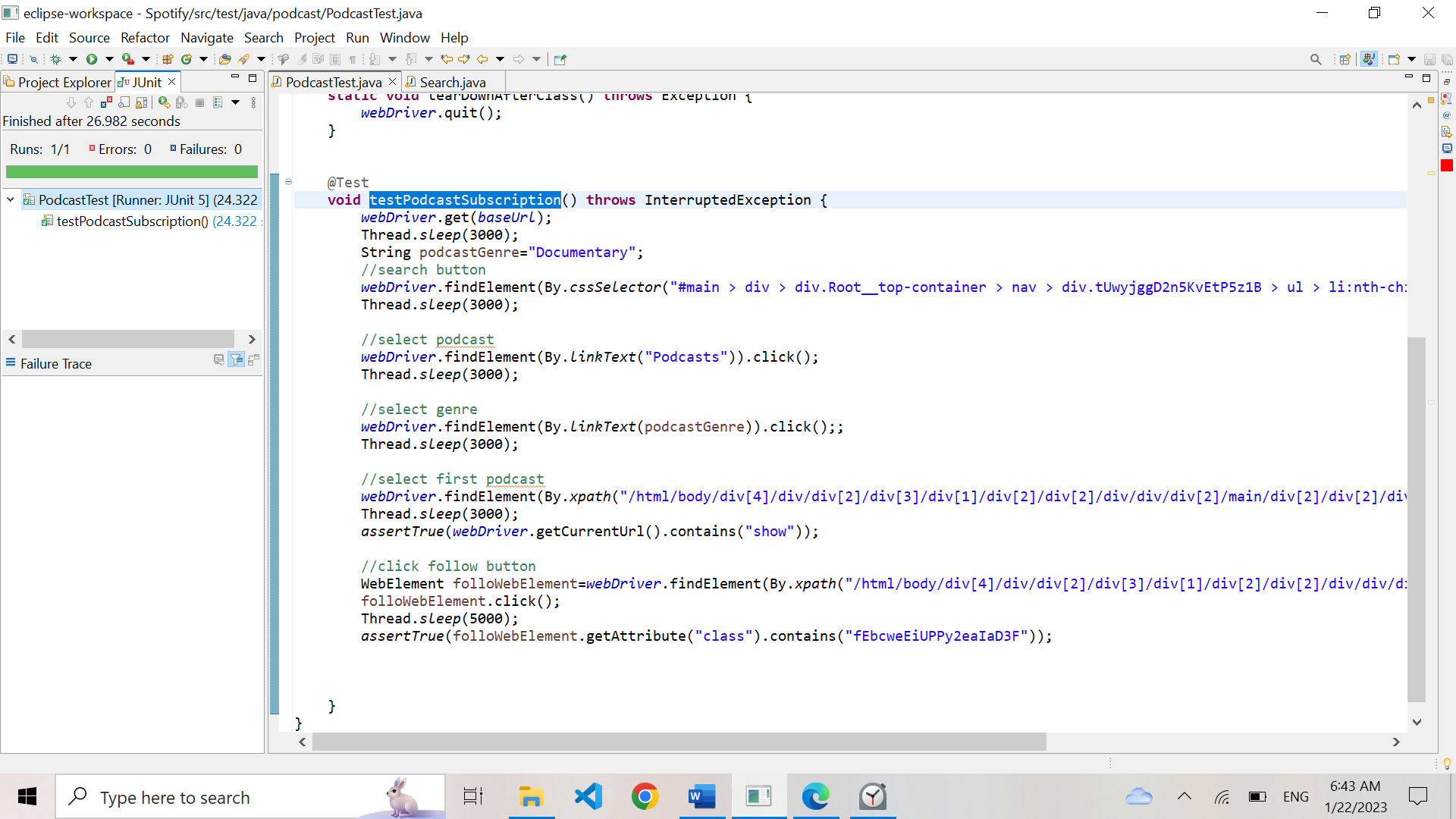


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Account Settings Update | | | | |
| **Description:**  Verify that a user can update their account settings | | | | |
| **Pre-condition(s): A user has an account and is logged in** | | | | |
| **Test Steps:**  1. Click on the “user” tab  2. Click on the “settings” link  3.Verify that the link has redirected us to the correct page | **Test Data:**  Username = “313gmiugyw5oanf2vyq53hj72kfu”  New display name = “zake” | **Expected Result:**  The account settings are updated and saved successfully | **Actual Result:**  The account settings are updated and saved successfully. | **Status:**  PASS |
| **Notes:** | | | | |

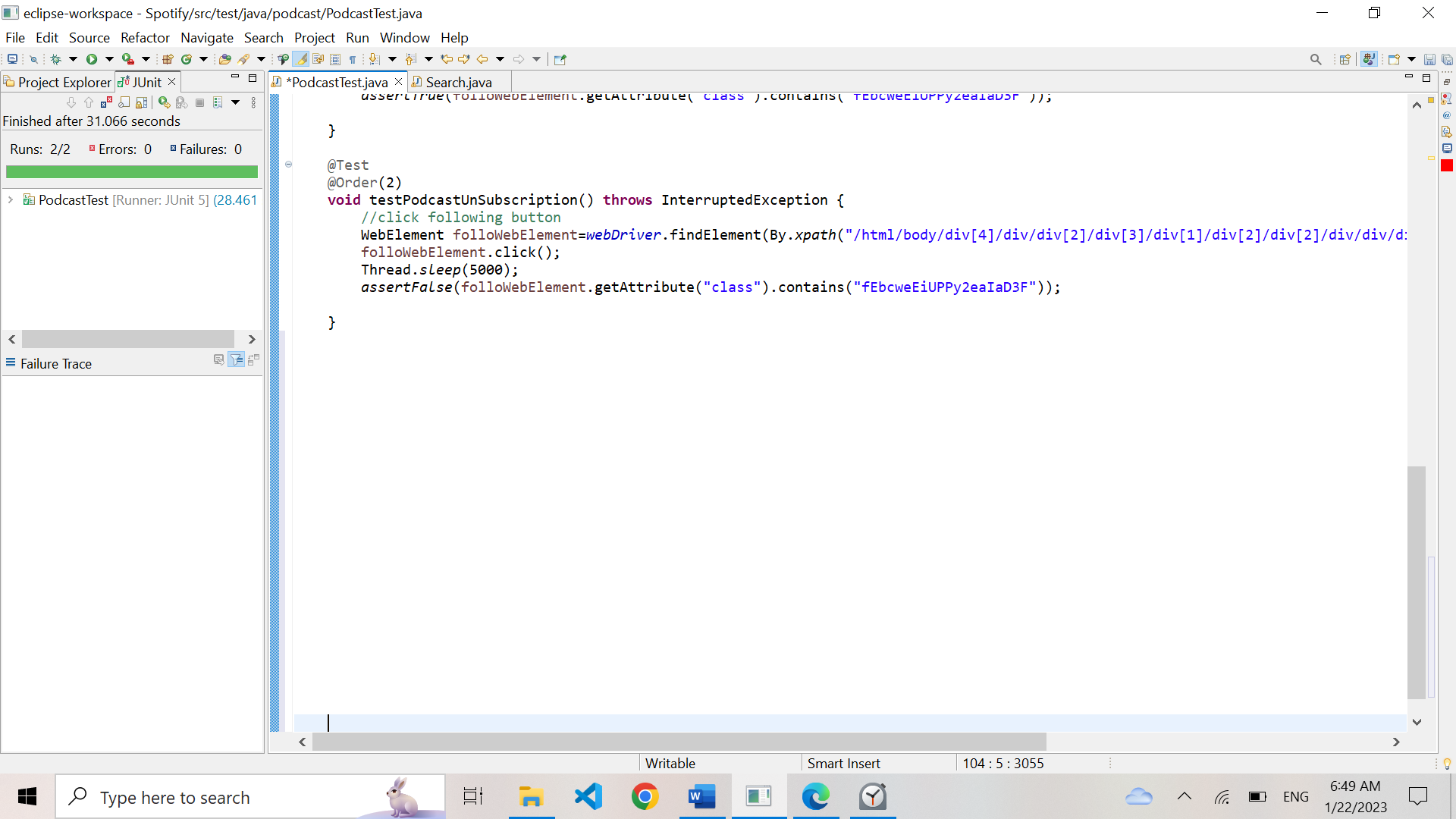
## 3.9. Podcast Functionality

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:**  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. | **Test Data:**  Genre = “Documentary” | **Expected Result:**  The user is successfully subscribed to the podcast. | **Actual Result:**  The user is successfully subscribed to the podcast. | **Status:**  PASS |
| **Notes:** | | | | |

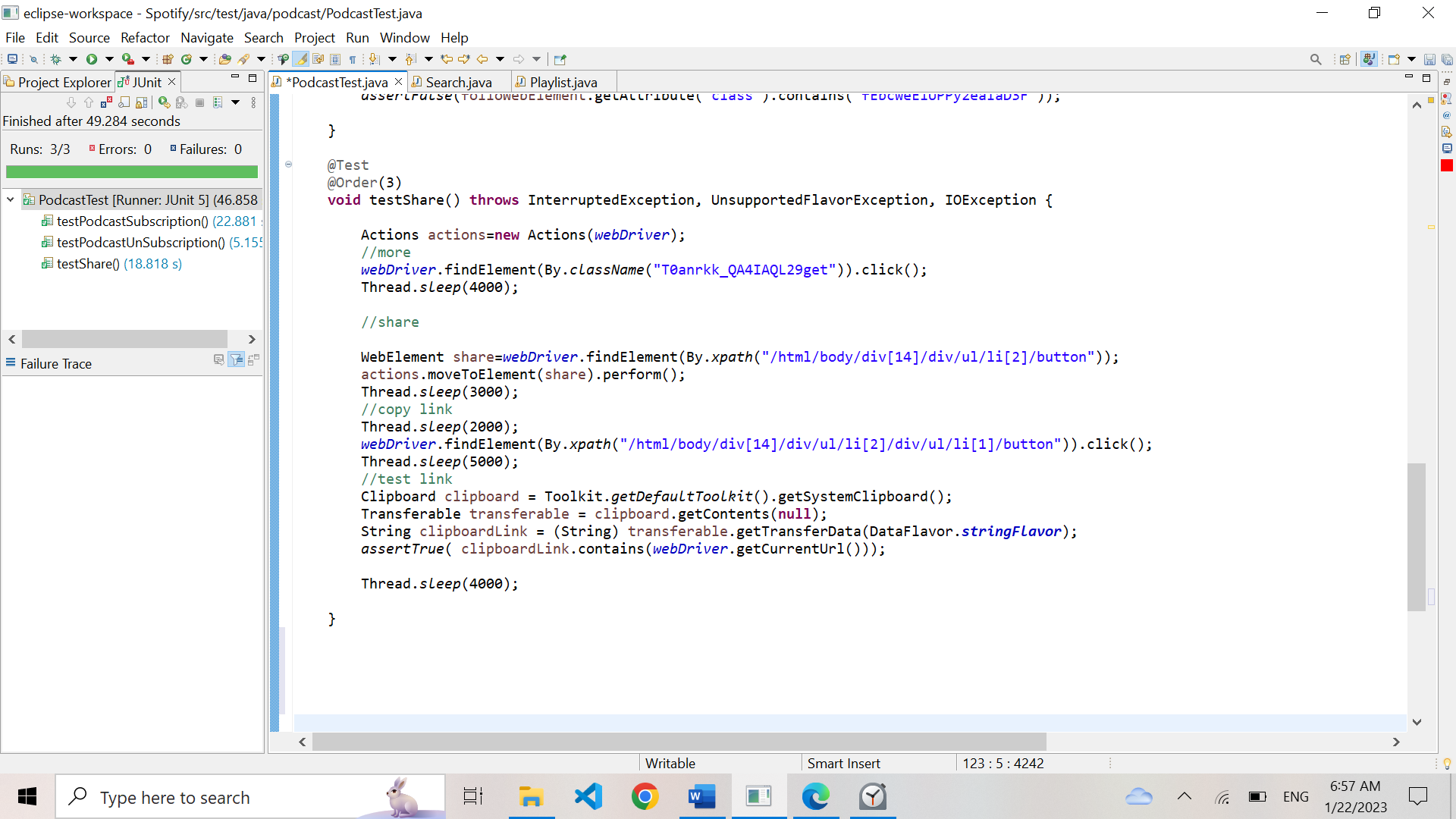


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Podcast Un-Subscription | | | | |
| **Description:**  Verify that a user can subscribe to a podcast | | | | |
| **Pre-condition(s): A user is logged in and is subscribed to at least one podcast** | | | | |
| **Test Steps:**  1. Select podcast.  4. Wait for the success pop-up notification and assert that user has successfully unsubscribed by checking class name. | **Test Data:** | **Expected Result:**  The user is successfully un-subscribed to the podcast. | **Actual Result:**  The user is successfully un-subscribed to the podcast. | **Status:**  PASS |
| **Notes:** | | | | |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Podcast Playback | | | | |
| **Description:**  Test the functionality of playing an episode, pausing an episode, | | | | |
| **Pre-condition(s): A user has selected a podcast** | | | | |
| **Test Steps:**  1.Click “play”  2.Click “pause”  3.Click “skip forward”  4.Click “skip back” | **Test Data:** | **Expected Result:**  The episode plays, pauses, can be skipped forward or back | **Actual Result:**  The episode plays, pauses, can be skipped forward or back | **Status:**  PASS |
| **Notes:** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Test Podcast Shareability | | | | |
| **Description:**  **Verify that user can generate a link to share podcast** | | | | |
| **Pre-condition(s):** | | | | |
| **Test Steps:**  1.Navigate to the podcast you wish to share  2. Click on the share button.  3. Select the option to share the playlist by copying link  3. Assert that the copied link leads to the episode in question | **Test Data:** | **Expected Result:**  The link to podcast has been successfully copied. | **Actual Result:**  The link to podcast has been successfully copied. | **Status:**  PASS |
| **Notes:** | | | | |



## 3.10. Library Functionality

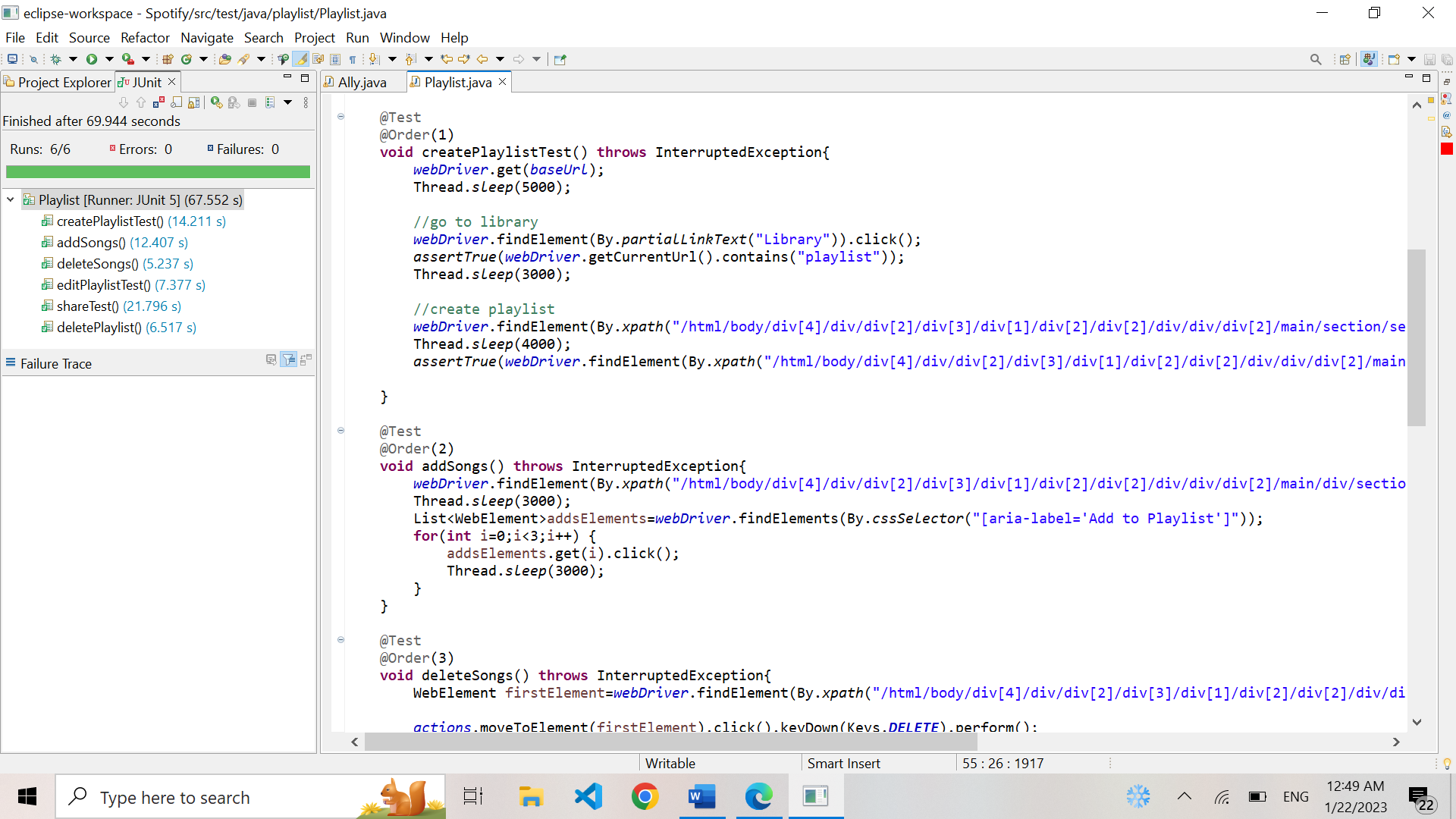
User want to play a song

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Playback | | | | |
| **Description:**  Test the functionality of playing a song, pausing a song, skipping a song, shuffling songs, and repeating a song | | | | |
| **Pre-condition(s): A user is logged in and has a song/album/playlist selected** | | | | |
| **Test Steps:**  1.Click “play”  2.Click “pause”  3.Click “skip”  4.Click “Go Back”  5. Click “Shuffle”  5.Click “Repeat” | **Test Data:** | **Expected Result:**  The song plays, pauses, skips, shuffles and repeats. | **Actual Result:**  The song plays, pauses, skips, shuffles and repeats. | **Status:**  PASS |
| **Notes:** | | | | |

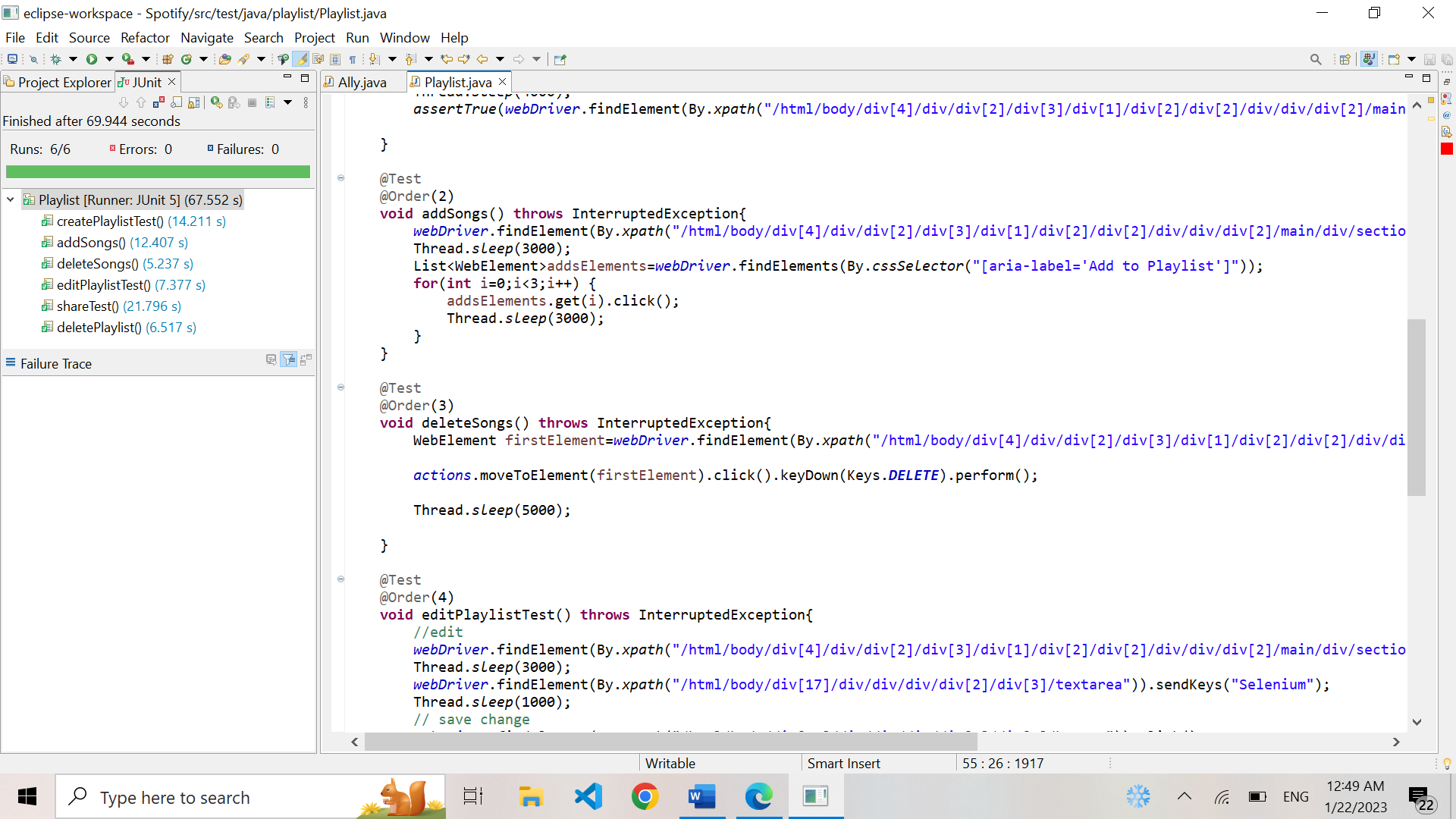
## 3.11. Playlist Functionality

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

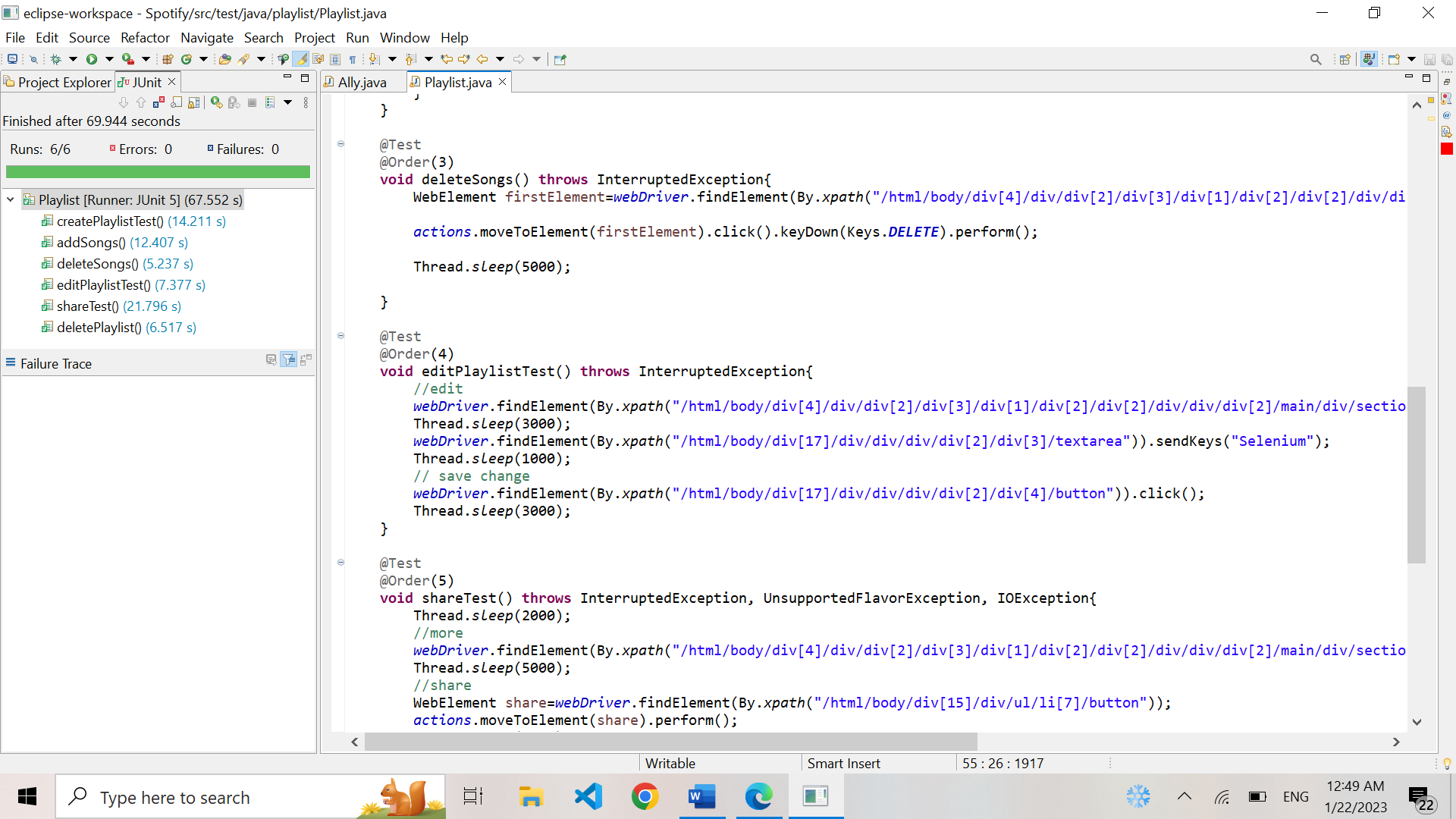
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Playlist Creation | | | | |
| **Description:**  Verify Playlist creation | | | | |
| **Pre-condition(s): A user is logged in to their account** | | | | |
| **Test Steps:**  1.Navigate to the “Library” tab  2. Click on the “New Playlist” button  3. Assert that the library has been created | **Test Data:** | **Expected Result:**  A new playlist is created and added to the user's library | **Actual Result:**  A new playlist is created and added to the user's library | **Status:**  PASS |
| **Notes:** | | | | |



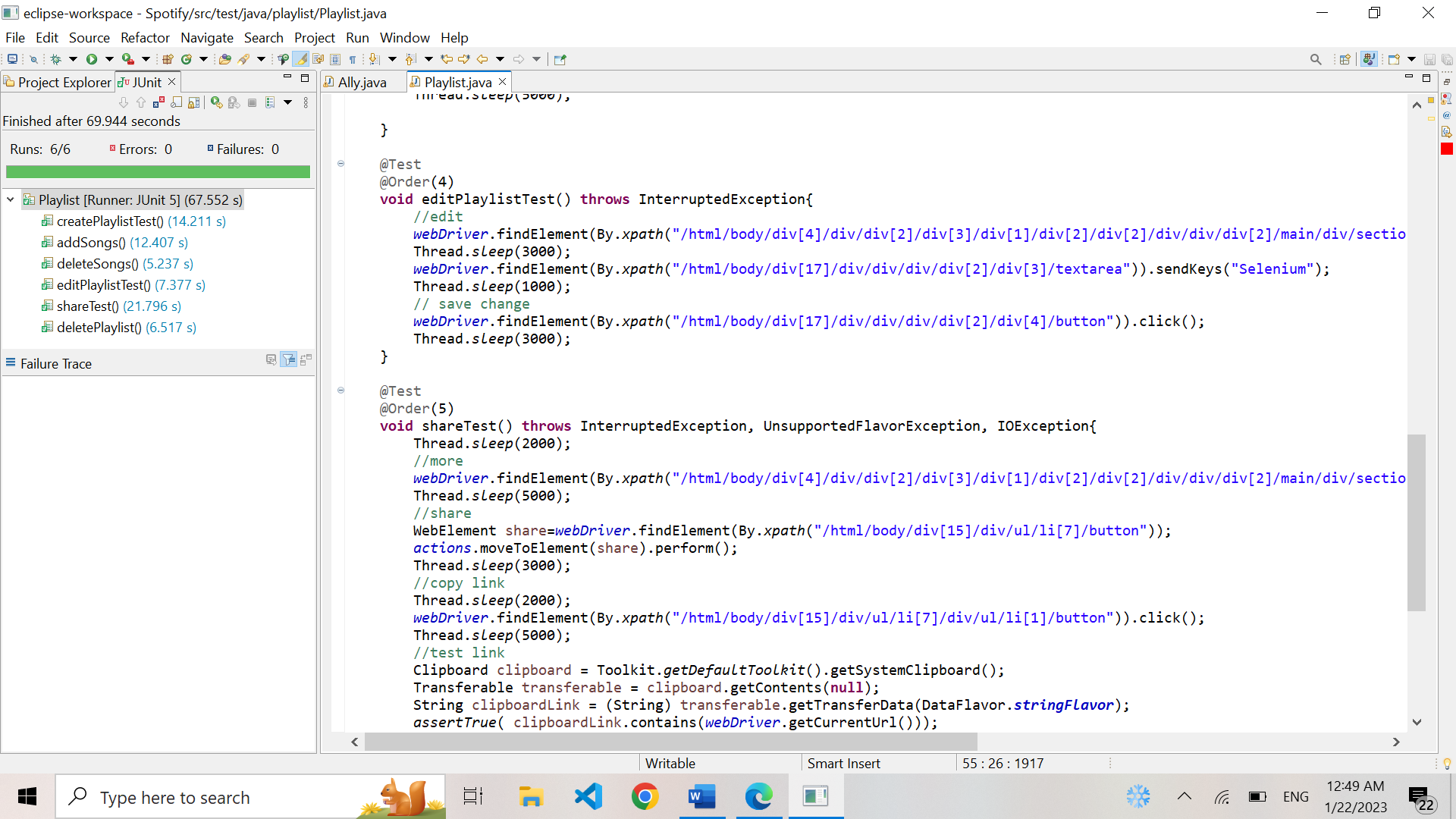
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Song Addition to a Playlist | | | | |
| **Description:**  Test the ability for a user to add songs to an existing playlist | | | | |
| **Pre-condition(s): A user is logged in to their account, and has at least one existing playlist** | | | | |
| **Test Steps:**  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 | **Test Data:**  Search song/artist= “bts” | **Expected Result:**  The selected songs are added to the selected playlist | **Actual Result:**  The selected songs are added to the selected playlist | **Status:**  PASS |
| **Notes:** | | | | |



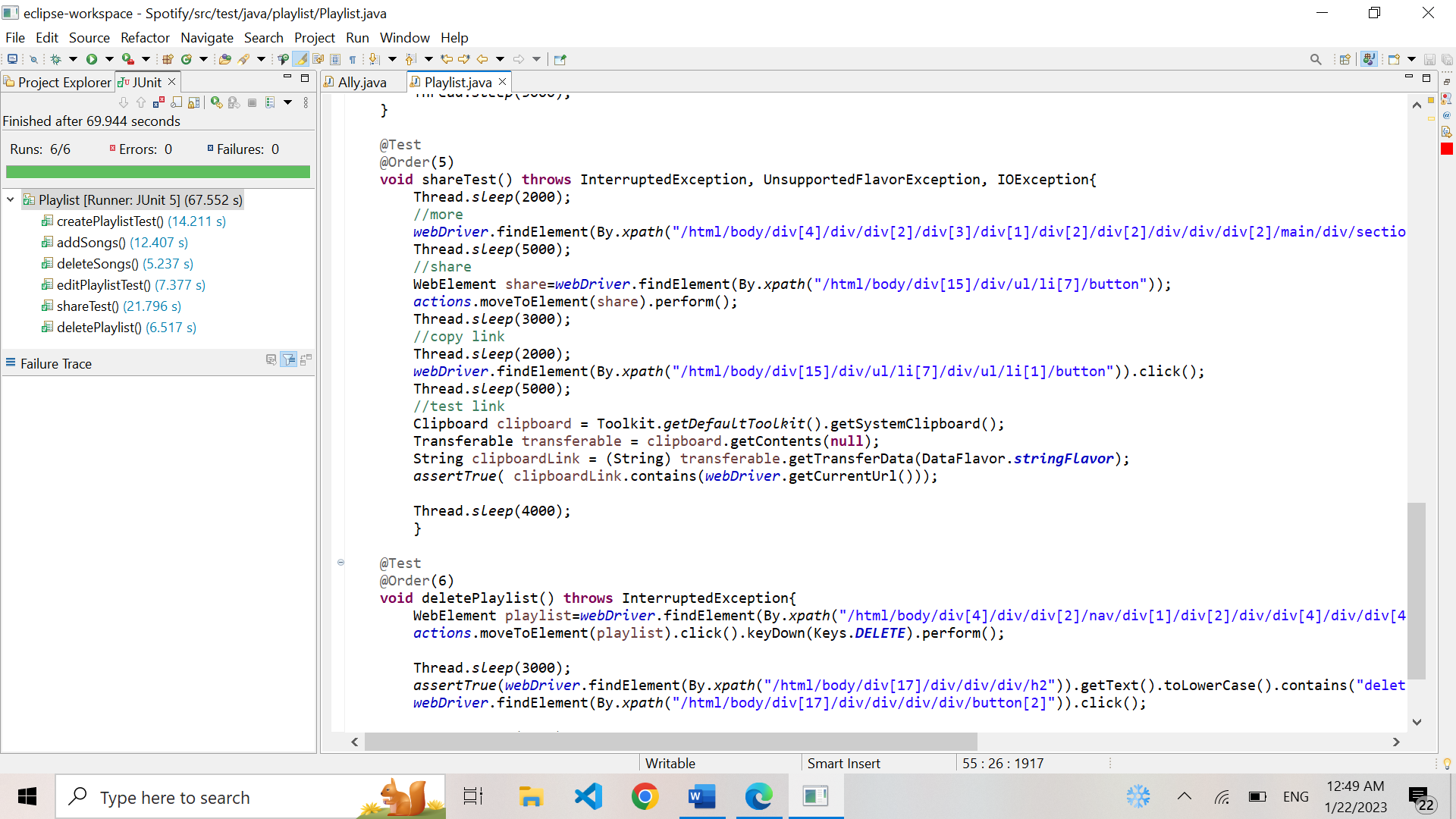
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Verify Song Deletion from Playlist | | | | |
| **Description:**  Test the ability for a user to delete songs from an existing playlist | | | | |
| **Pre-condition(s): A user is logged in to their account** | | | | |
| **Test Steps:**  1.Navigate to the playlist from where you want to delete the song.  2. Search for song you want to delete and select it  3. Press delete key | **Test Data:** | **Expected Result:**  The selected song is deleted from the selected playlist | **Actual Result:**  The selected song is deleted form the select playlist | **Status:**  PASS |
| **Notes:** | | | | |



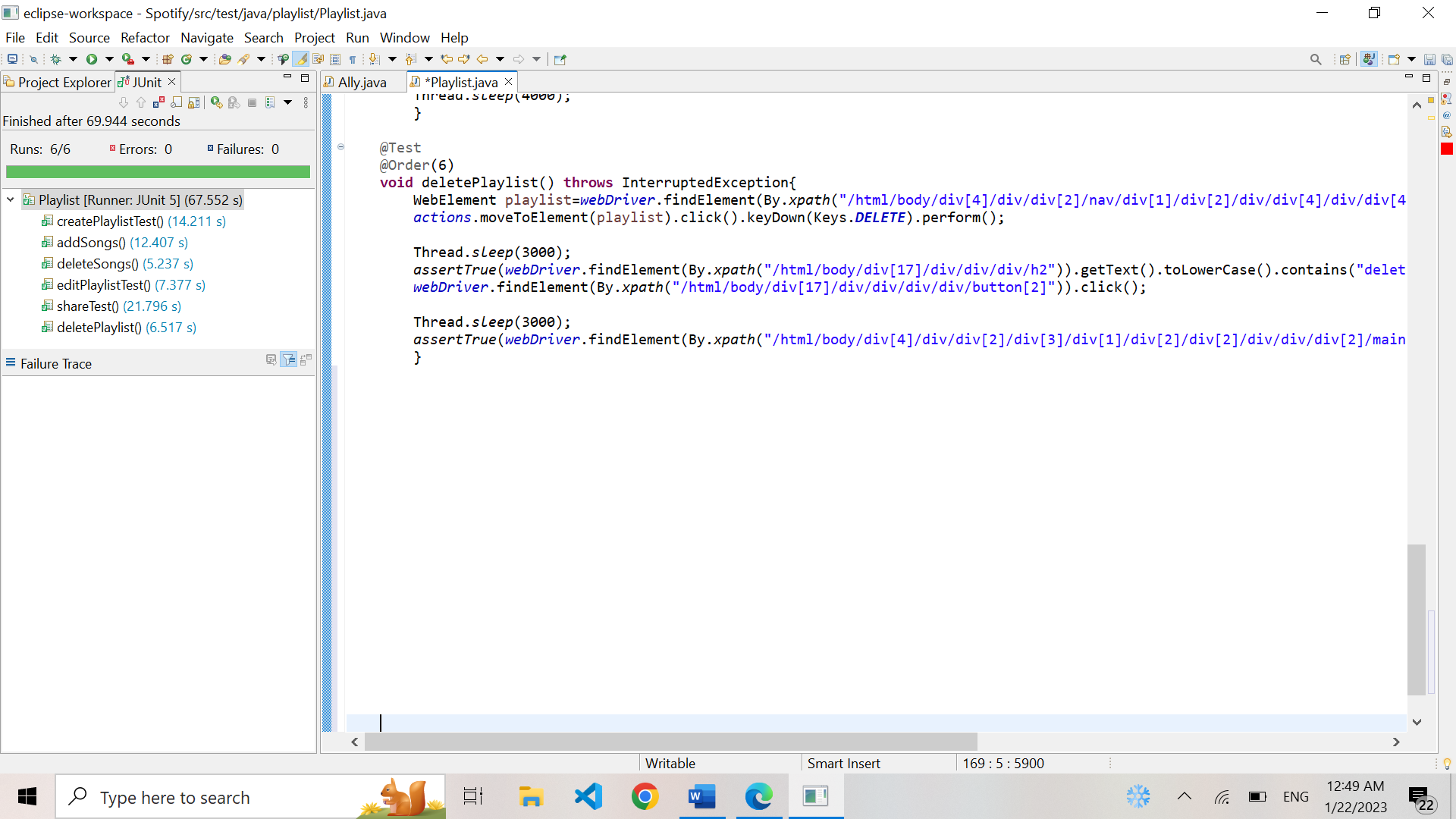
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name:** Test Playlist Editing | | | | |
| **Description:**  Verify that you can edit playlist | | | | |
| **Pre-condition(s): A user is logged in to their account and has at least one playlist** | | | | |
| **Test Steps:**  1. Navigate to your playlist  2. Click on the edit button  3. Once a popup appears, add text  4.Save it | **Test Data:**  Text= “Selenium” | **Expected Result:**  The playlist has been edited | **Actual Result:**  The playlist has been edited | **Status:**  PASS |
| **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:**  1.Navigate to the playlist you wish to share  2. Click on the share button located on the playlist.  3. Select the option to share the playlist by copying link  3. Assert that the copied link leads to the playlist in question | **Test Data:**  , | **Expected Result:**  The link to playlist has been successfully copied. | **Actual Result:**  The link to playlist has been successfully copied. | **Status:**  PASS |
| **Notes:** | | | | |



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



# 4. Conclusion

## 4.1. Testing Summary

|  |  |  |  |
| --- | --- | --- | --- |
| **Testing Tool** | **Total Tests** | **Passed Tests** | **Failed Tests** |
| Junit,  Selenium Web Driver,  aXe |  | # Of passing tests | 1– testingAccessibility |

## 4.2. Final Thoughts

Provide some closing statements or your final thoughts about the project you tested. Did you find that it was implemented well? Did you notice any obvious mistakes or flaws in it? Are there some recommendations or observations you would note for the site?