**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**

21.02.2023

Contents

[1. Introduction 3](#_Toc125221662)

[1.1. About the Project 3](#_Toc125221663)

[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/. 3](#_Toc125221664)

[1.2. Project Functionalities and Screenshots 3](#_Toc125221665)

[The main features of the Spotify project include: 3](#_Toc125221666)

[ Music streaming: Users can listen to a wide variety of music from various genres and artists. 3](#_Toc125221667)

[ Playlists: Users can create, edit, and share playlists with other users. 3](#_Toc125221668)

[ Search: Users can search for music based on artist, album, or genre. 3](#_Toc125221669)

[ 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. 3](#_Toc125221670)

[ Podcasts: Users can listen to podcasts and access a variety of podcast shows. 3](#_Toc125221671)

[ Social media integration: Users can connect their Spotify account with their social media accounts to share music and playlists with friends. 3](#_Toc125221672)

[ Offline listening: Users can download music to listen offline. 3](#_Toc125221673)

[2. Test Plan 3](#_Toc125221674)

[2.1. Scope 3](#_Toc125221675)

[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 for users with disabilities and compatible with different screen readers and keyboard navigation. 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. 3](#_Toc125221676)

[2.2. Testing Environment and Tools 3](#_Toc125221677)

[3. Test Execution 4](#_Toc125221678)

[3.1. Sing Up 4](#_Toc125221679)

[3.2. Login In **Error! Bookmark not defined.**](#_Toc125221680)

[9. Conclusion 7](#_Toc125221681)

[9.1. Testing Summary 7](#_Toc125221682)

[9.2. Final Thoughts 8](#_Toc125221683)

# 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.

# 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 for users with disabilities and compatible with different screen readers and keyboard navigation. 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.

## 2.2. Testing Environment and Tools

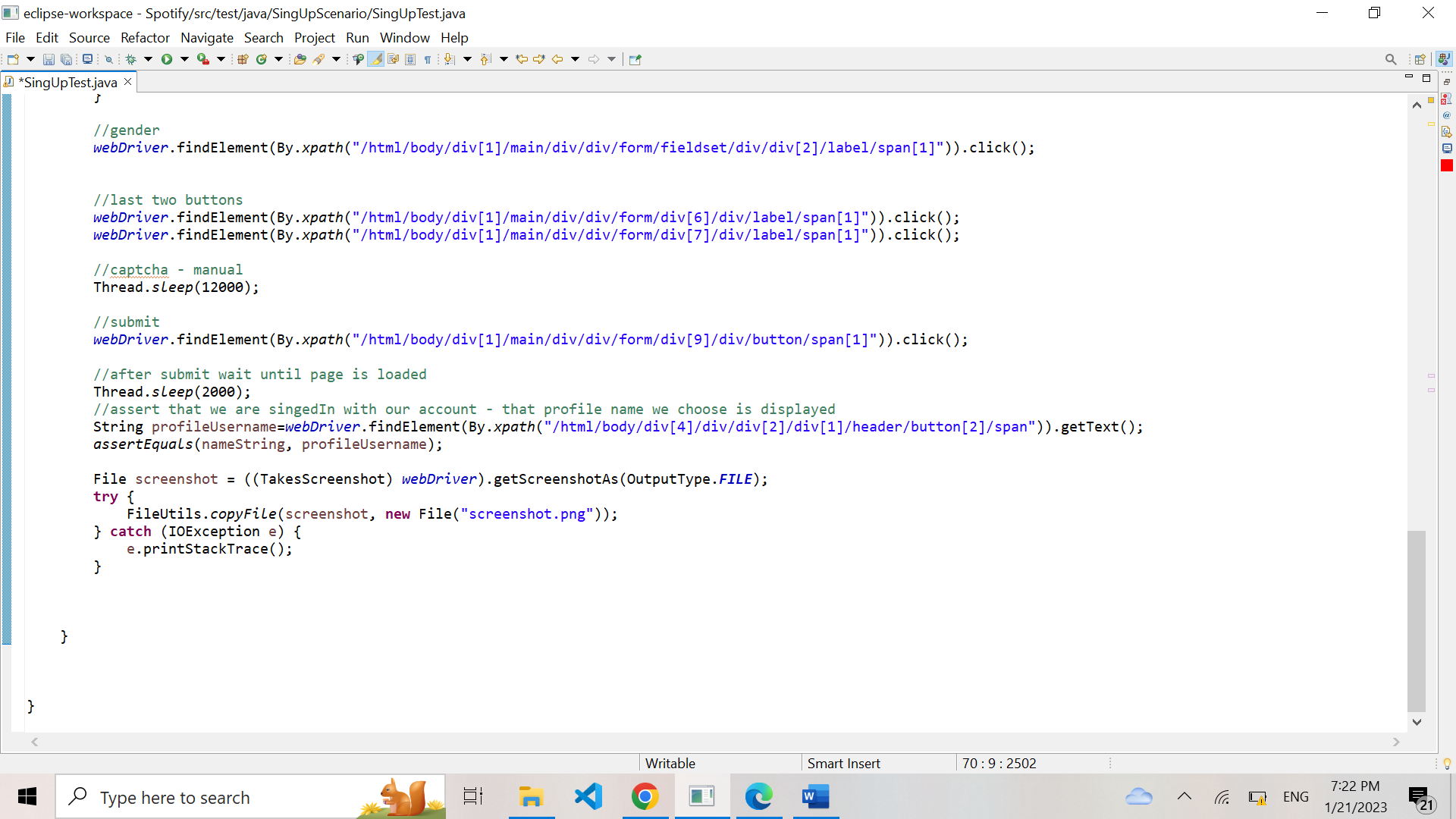
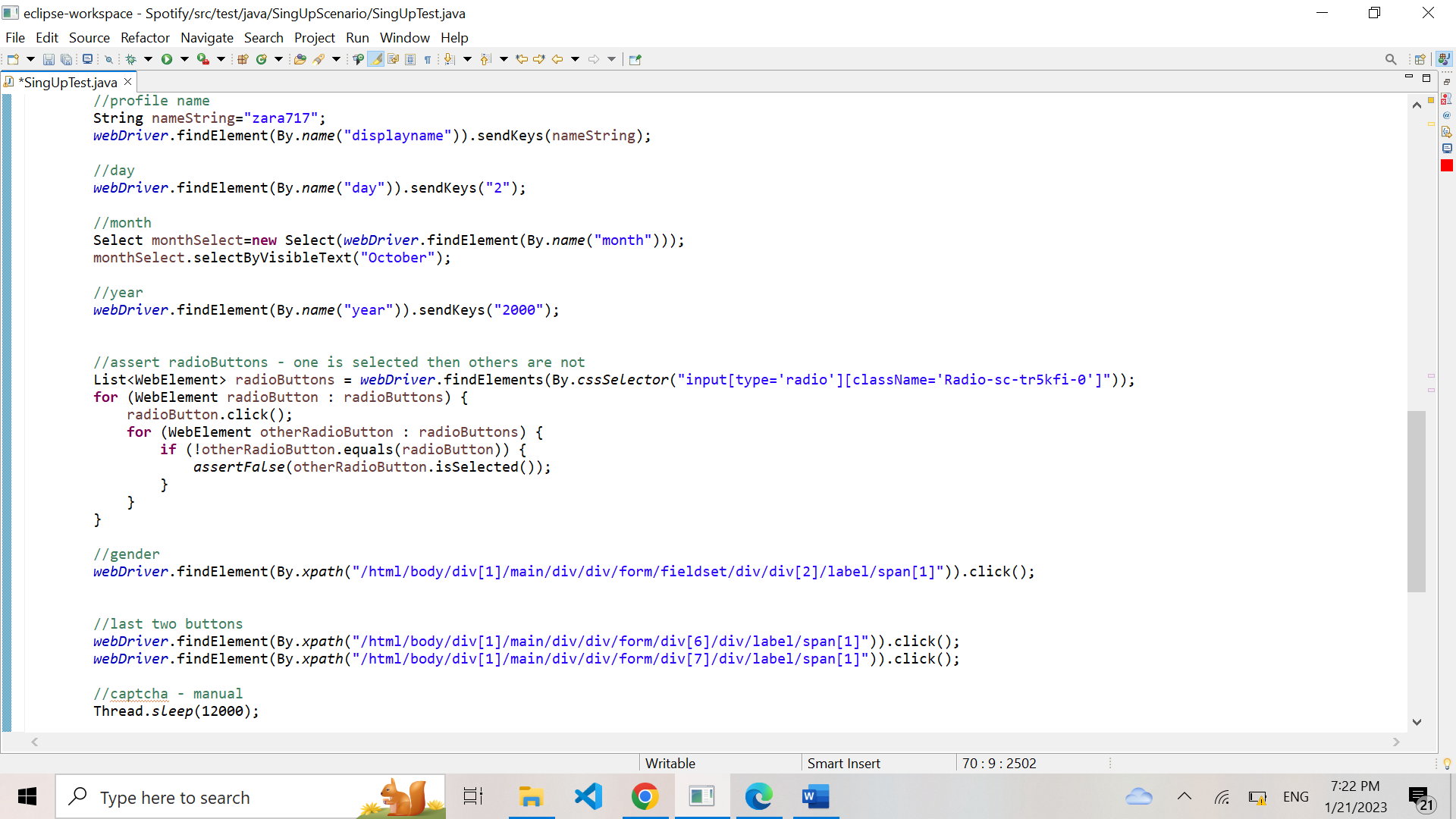
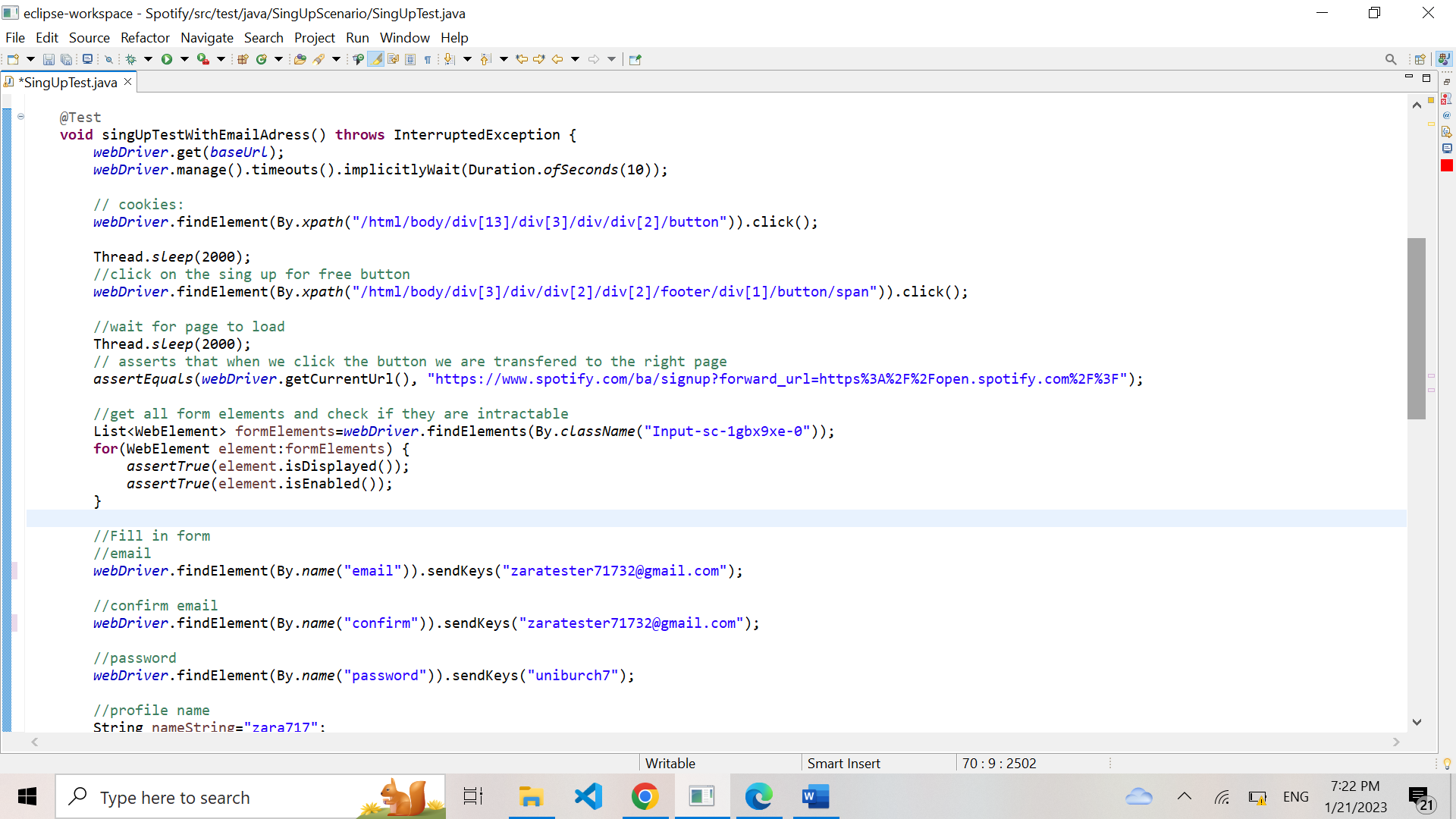
In this project, we will be using Selenium WebDriver, a widely-used open-source tool for automating web browsers, as our main testing tool. We will also be using JUnit, a popular open-source testing framework for Java, to write and run our test cases. Additionally, we will be using the selenium-a11y 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.

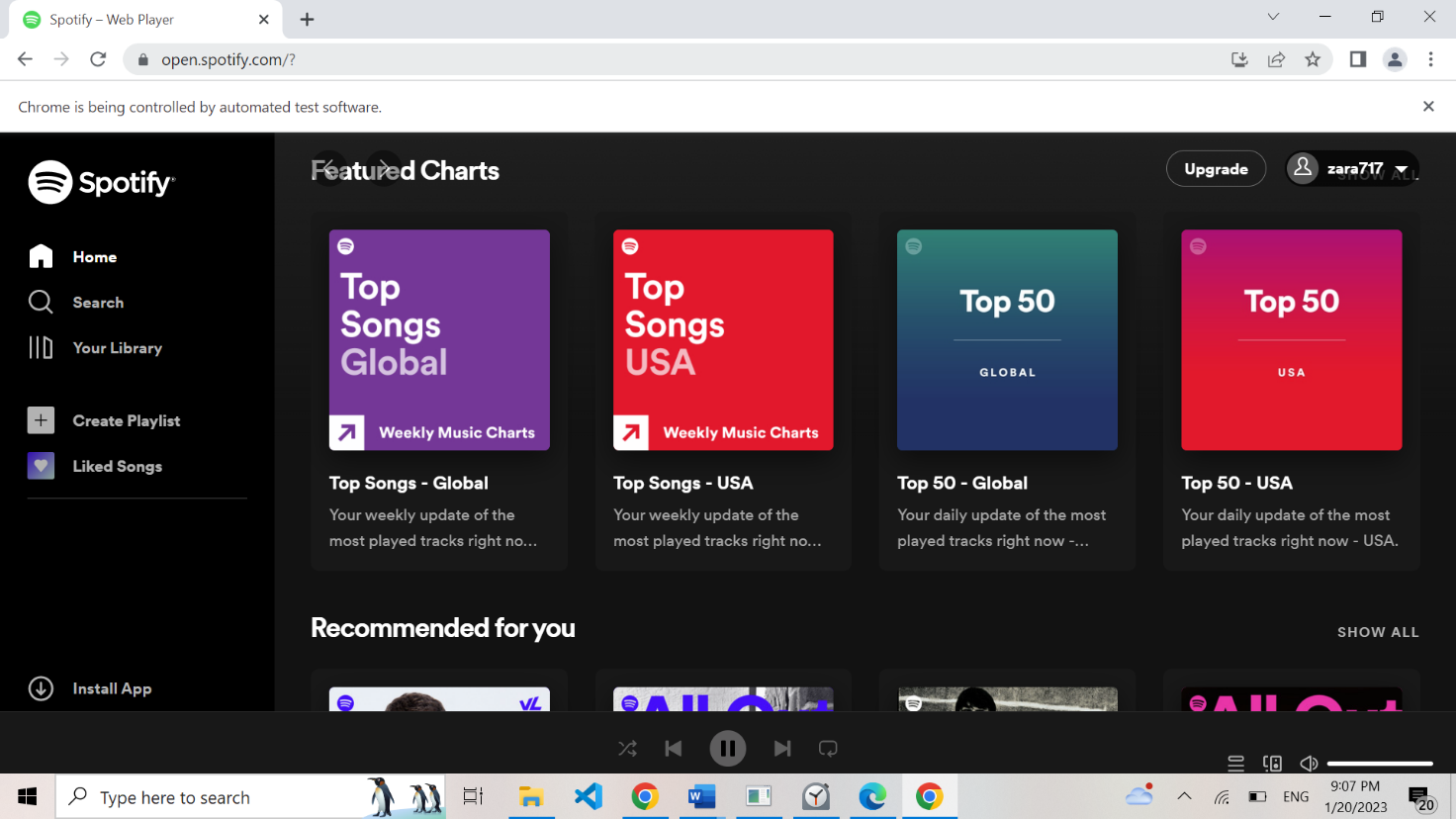
# 3. Test Execution

## 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:** | | | | |

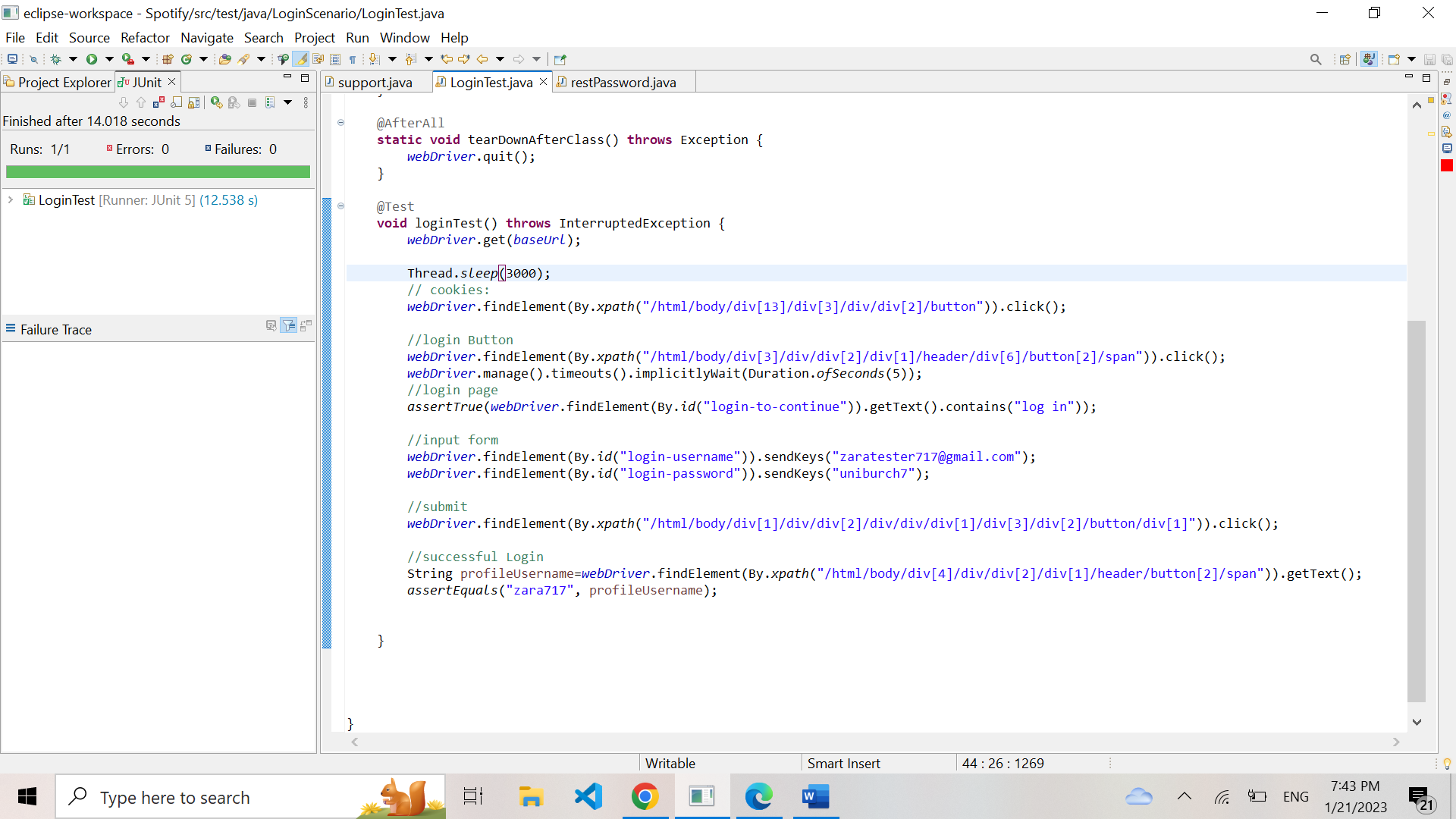




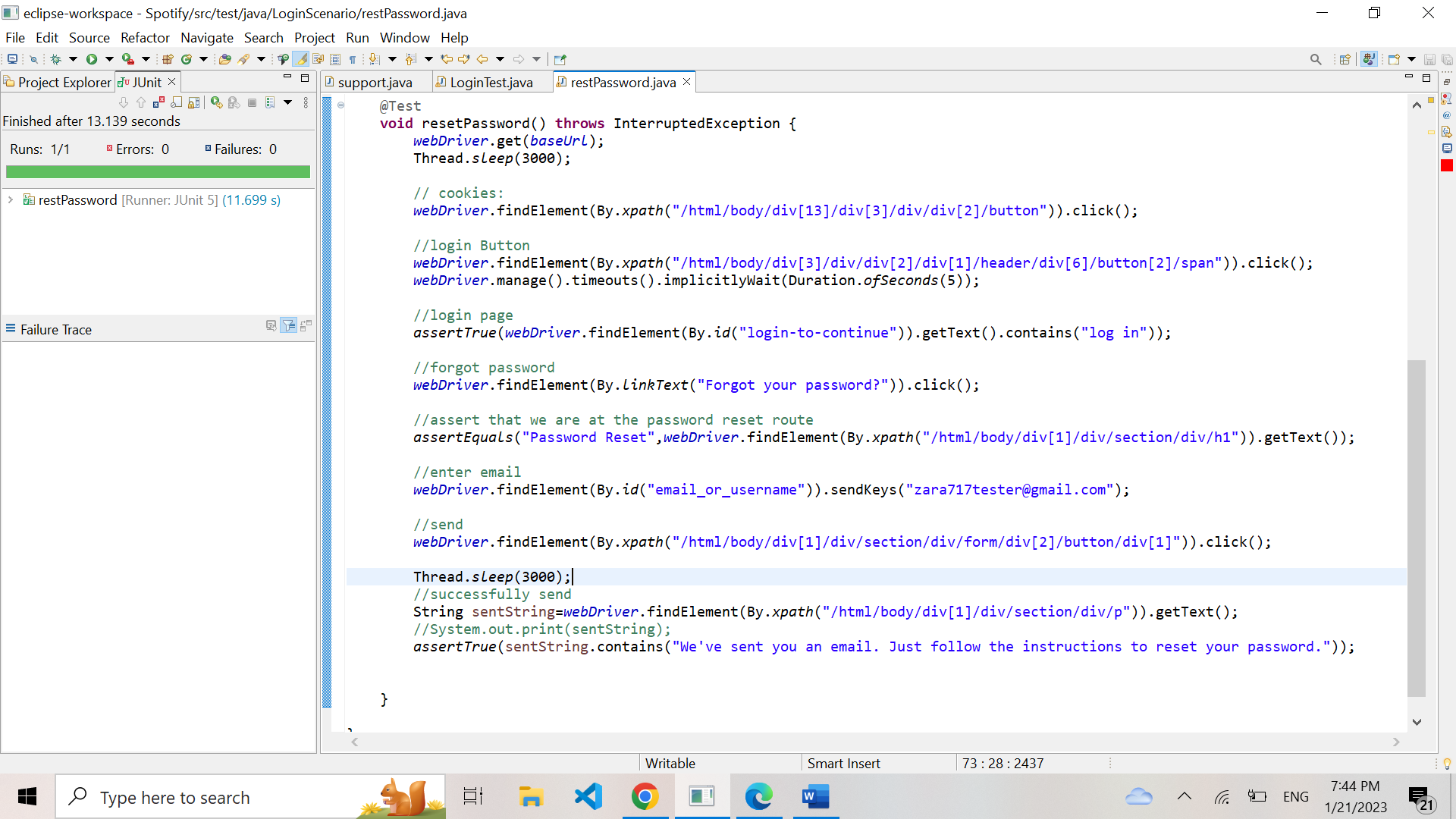
## 3.2. Login Functionality

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 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:** | | | | |

## 3.4. Search 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:** | | | | |

## 3.5. Localization

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:** | | | | |

## 3.6. Accessability

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:** | | | | |

## 3.7. Radio 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:** | | | | |

## 3.8. Profile Managment

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:** | | | | |

## 3.9. Podcast 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:** | | | | |

## 3.8. 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:** | | | | |

# 9. Conclusion

## 9.1. Testing Summary

Provide a summary of all your executed tests. Something like this would be alright:

|  |  |  |  |
| --- | --- | --- | --- |
| **Testing Tool** | **Total Tests** | **Passed Tests** | **Failed Tests** |
| Framework or tool(s) used for testing. If you wrote tests in multiple different tools, create a row with the number of tests for each of them. | total number of tests | # Of passing tests | # Of failing tests |

If there are failing tests, provide a list of their names. Ideally, you can also create bookmarks in the document and make links to those failing test cases.

## 9.2. 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?