Programming III Project Testing Documents

Media player



**Umakant Awasthi**

South MetroTafe

Contents

[INTRODUCTION 2](#_Toc27034161)

[PROJECT PERSPECTIVE 2](#_Toc27034162)

[MATERIALS 2](#_Toc27034163)

[TESTING PROCEDURE 2](#_Toc27034164)

[Test Case - Outline 4](#_Toc27034165)

[Test Case # 4](#_Toc27034166)

[Action 4](#_Toc27034167)

[Test Result 4](#_Toc27034168)

[Page 4](#_Toc27034169)

[TEST CASE RESULTS 8](#_Toc27034170)

[Test Case # 8](#_Toc27034171)

[Action 8](#_Toc27034172)

[Input & expected output 8](#_Toc27034173)

[Result 8](#_Toc27034174)

[REFERENCES 21](#_Toc27034175)

# INTRODUCTION

The purpose of this project is to create a media player application that allows user to login and access the fundamental functions for a media player, but also has features to save user file to the database, then display saved file to the Playlist so the next time when the user successfully logged in to the media application they gain direct access to the existing media file they saved.

# PROJECT PERSPECTIVE

The system will be composed into two different sides. Server-side and Client-side. The server-side component will be using MySQL database which stores user information from Register and their current playlist file path to the system.

The Client-side component will be a graphical interface which has all the core functions and features for a media player. The application will guide the user to add all the media files from the local path and supports the user with a quick search function that allows the user to find the media they wanted within a second.

# MATERIALS

1. - Must contain dynamic data structures
2. - Must contain hashing techniques
3. - Must contain sorting algorithm
4. - Must contain searching technique
5. - Must contain 3rd party library
6. - Must have a GUI - Must adhere to coding standards

# TESTING PROCEDURE

***GUI*** *Design*

1. **Graphical User interface** design which provides information for the user to Register.
2. Graphical User interface design which allows the user to log in using either Email or Username and password for logging in.
3. **Graphical User interface** design which allows the user to access the fundamental functions and features for a media player when they successfully logged in. *Function & Features*
4. The user creates an account in the Register page.
5. User information such as password is passed through the various methods and **hashing techniques**. A salt value is generated which added to the input of hash functions to create unique hashes.
6. User information (username, hashed password and the salted value) are stored to the database using **3rd party library --- MySQL**.
7. The user enters her/his username and password. Login Page
8. The system compares user name from the user inputs and MySQL User information.
9. If the methods find the user information it will then read this users database information and then the methods will then hashes the user input password with the salt number that’s been read and compared to the stored password
10. If the password matches. The current form will be closed and the media player will be shown.
11. The user is able to add media to the **dynamic data structure/Playlist.**
12. The user could access basic features such as to play the current song, previous song, next song, and pause and start the song.
13. If the playlist is unsorted, the user is able to use a **bubble sort method** to sort playlists alphabetically.
14. To find a song, the user could user **linear Search method** to search a song by its name.

# Test Case - Outline

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case # | Action | Test Result | Page |
| Test Case 1 | GUI - Register | Pass | 8 |
| Test Case 2 | GUI - Login | Pass | 8 |
| Test Case 3 | GUI - Media Player | Pass | 8 |
| Test Case 4 | In the register section user enters empty Username & password | Pass | 9 |
| Test Case 5 | In the register user enters an empty username with Valid password | Pass | 9 |
| Test Case 6 | In the register section user enters valid username, the password field is empty | Pass | 9 |
| Test Case 7 | In the register section user enters Wrong Passwords | Pass | 9 |
| Test Case 8 | Enters correct password and username | Pass | 10 |
| Test Case 9 | Information is correctly stored in the database using hashing techniques | Pass | 10 |
| Test Case 10 | In the login section user enters empty username and password | Pass | 10 |
| Test Case 11 | In the login section user enters valid username with empty password | Pass | 10 |
| Test Case 12 | In the login section user enters wrong username and wrong password | Pass | 11 |
| Test Case 13 | In the login section user enters correct username and wrong password | Pass | 11 |
| Test Case 14 | In the login section user enters correct username and password | Pass | 11 |
| Test Case 15 | In the Media player page users’ username is displayed at the top of the screen. | Pass | 12 |
| Test Case 16 | Support various audio and video formats | Pass | 12 |
| Test Case 17 | Media file is added to the playlist | Pass | 13 |
| Test Case 18 | Play Song where there is no song selected in the playlist | Pass | 13 |
| Test Case 19 | Start Button image changed when music is playing | Pass | 13 |
| Test Case 20 | Double click playlist to play the song | Pass | 14 |
| Test Case 21 | Selecting the song and click the start button while another song is playing | Pass | 14 |
| Test Case 22 | Play Next Song | Pass | 14 |
| Test Case 23 | Play Previous Song | Pass | 15 |
| Test Case 24 | Stop song | Pass | 15 |
| Test Case 25 | Fast Forward & Rewind | Pass | 15 |
| Test Case 26 | Hide playlist  &  Show playlist | Pass | 16 |
| Test Case 27 | Search playlist With empty result | Pass | 16 |
| Test Case 28 | Search playlist With Wrong file name | Pass | 17 |
| Test Case 29 | Search playlist With Correct Name | Pass | 17 |
| Test Case 30 | Sorting | Pass | 18 |
| Test Case 31 | Removes duplicate items in the list | Pass | 18 |
| Test Case 32 | Auto Restart song when current song is end | Pass | 19 |
| Test Case 33 | Display song information and duration | Pass | 19 |
| Test Case 34 | Play first song when index of the song is at last | Pass | 20 |
| Test Case 35 | Full Screen | Pass | 20 |
| Test Case 36 | Playlist is saved when form close | Pass | 21 |

# 

# TEST CASE RESULTS

# 

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case # | Action | Input & expected output | Result |
| Test Case 1 | GUI - Register | Ahead to coding standards |  |
| Test Case 2 | GUI - Login | Ahead to coding standards |  |
| Test Case 3 | GUI - Media Player | Ahead to coding standards |  |
| Test Case 4 | In the register section user enters empty Username & password | *Inputs*  \*empty  *Expected output*  Display Error message |  |
| Test Case 5 | In the register user enters an empty username with Valid password | *Inputs*  Username=\*empty  Password=”user”  Confirm password=”user”  *Expected output*  Display Error message |  |
| Test Case 6 | In the register section user enters valid username, the password field is empty | *Inputs*  Username=Users  Password=\*empty  Confirm password=\*empty  *Expected output*  Display Error message |  |
| Test Case 7 | In the register section user enters Wrong Passwords | *Inputs*  Username=”Users”  Password=”users”  Confirm password=”user”  *Expected output*  Display Error message |  |
| Test Case 8 | Enters correct password and username | *Inputs*  Username=”Users”  Password=”user”  Confirm password=”user”  *Expected output*  Username is successfully created. |  |
| Test Case 9 | Information is correctly stored in the database using hashing techniques | *Inputs*  Username=”Users”  Password=”user”  *Expected output*  Username is successfully stored in database using hashing techniques. |  |
| Test Case 10 | In the login section user enters empty username and password | *Inputs*  Username=\*Empty  Password=\*Empty  *Expected output*  Display Error message |  |
| Test Case 11 | In the login section user enters valid username with empty password | *Inputs*  Username=”Users”  Password=\*Empty  *Expected output*  Display Error message |  |
| Test Case 12 | In the login section user enters wrong username and wrong password | *Inputs*  Username=”admin”  Password=”admin”  *Expected output*  Display Error message |  |
| Test Case 13 | In the login section user enters correct username and wrong password | *Inputs*  Username=”Users”  Password=”admin”  *Expected output*  Display Error message |  |
| Test Case 14 | In the login section user enters correct username and password | *Inputs*  Username=”Users”  Password=”user”  *Expected output*  User is successfully logged in to the Second form : Media Player. |  |
| Test Case 15 | In the Media player page users username is displayed at the top of the screen. | *Expected output*  Users username is displayed at the top of the screen. |  |
| Test Case 16 | Support various audio and video formats | *Expected output*  Mp3,Mp4,Avi,AllFiles |  |
| Test Case 17 | Media file is added to the playlist | *Expected output*  Successfully added |  |
| Test Case 18 | Play Song where there is no song selected in the playlist | *Input*  *Click start button*    *Expected output*  Successfully play the first song in the playlist  Playing Song is selected |  |
| Test Case 19 | Start Button image changed when music is playing | *Input*  *Click start button*    *Expected output*  *Image changed to pause icon* |  |
| Test Case 20 | Double click playlist to play the song | *Input*  *Double Click*    *Expected output*  Successfully playing the selected song. |  |
| Test Case 21 | Selecting the song and click the start button while another song is playing | *Input*  *Click Start button*    *Expected Output*  Successfully playing the selected song while another song is playing. |  |
| Test Case 22 | Play Next Song | *Input*  *Current song :*    *Expected Output*  We Are One |  |
| Test Case 23 | Play Previous Song | *Input*  *Current song :*    *Expected Output*  *Playing Hellcat.mp3* |  |
| Test Case 24 | Stop song | *Input*  *Current song duration :*    *Expected Output*  *Stopped* |  |
| Test Case 25 | Fast Forward & Rewind | *Input*  *Current song duration :*  *00:01*    *Expected Output*  *Fast forward*  *Input*    *Expected Output*  *backwards* |  |
| Test Case 26 | Hide playlist  &  Show playlist | *Input*    *Expected Output*  *Hide playlist*  *Then click again to show playlist* |  |
| Test Case 27 | Search playlist With empty result | *Input*  *\*empty*  *Expected Output*  *Error message displays* |  |
| Test Case 28 | Search playlist With Wrong file name | *Input*  *123123*  *Expected Output*  *Error message displays item not found* |  |
| Test Case 29 | Search playlist With Correct Name | *Input*  *“Hellcat”*  *Expected Output*  *Hellcat is found and song is selected* |  |
| Test case 30 | Sorting | *Input*    *Expected Output*  *Playlist been sorted*  *Alphabetically* |  |
| Test case 31 | Removes duplicate items in the list | *Input*      *Expected Output*  *Removes duplicate item* |  |
| Test case 32 | Auto Restart song when current song is end | *Expected Output*  Restart playing current song |  |
| Test Case 33 | Display song information and duration | *Expected Output*  Display playing song information when its been played |  |
| Test Case 34 | Play first song when index of the song is at last | *Input*    *Expected Output*  Play first song |  |
| Test case 35 | Full Screen | *Input*    *Expected Output*  *Full screen* |  |
| Test case 36 | Playlist is saved when form close | *Expected Output*  *Display playlist when you login again* |  |

# REFERENCES

1. Blackboard - [https://blackboard.southmetrotafe.wa.edu.au](https://blackboard.southmetrotafe.wa.edu.au/webapps/login/)
2. GitHub - <https://github.com/MarcusMarcusMarcusss/PROGRAMMING_III_PROJECT>