November 22

Controlled Assessment

2017

Computer Science – Unit 3

Report

Scope of the Problem

Purpose of the Project

When teachers are preparing resources to use with their pupils they should make sure that the material is suitable for the class. One factor teachers should take into account is the reading age of the text that they have produced. If the reading age is too high then the resources could be unsuitable. The purpose of the project is for teachers who are preparing resources for their pupils to calculate the reading age of the child needed to understand the text. Teachers also need to be able search for the text using the keywords and calculated reading age of the resource. The program will use the Automated Readability Index (ARI) to calculate the reading age of the text.

Objectives of the Project

- Search for the text using keywords and the reading age.
- Allows the user to login to the program.
- Has an easy to use interface.
- Validate the inputted information.
- Must be reliable, well structured, and robust.

Data Required

- Title of the text.
- Intended reading age.
- Keywords.
- Text content.
- Username (For login).
- Password (For login).
- The option of what the user would like to search with either:
    - Keywords
    - Calculated reading age

Outputs Required

- Title of the resource.
- Intended reading age of the resource.
- Keywords of the resource.
- Text content of the resource.
- Calculated reading age of the resource.

Processing

- Validate the login details by comparing it with usernames and passwords saved in a text file (login.txt) to ensure it's a valid user account.
- Enter and store the title, text, keywords and intended reading in the resources array.
- Calculate the numbers of characters, words and sentences in the text using:
    - Count the characters using the length of the text.
    - Count the words by counting the spaces in the text.
    - Count the sentences by counting the full stops in the text.
- Use the ARI to calculate the reading age of the text using the following calculation:

    - Reading age of text = $4.71 \times \left(\frac{Characters}{Words}\right) + 0.5 \times \left(\frac{Words}{Sentences}\right) - 21.43$
    - Round the reading age up to a whole number (always round up).
- Store the calculated reading age alongside the other text detail in the resources text file (resources.txt).
- Output the intended reading age and the calculated reading age of the text.
- Allow other teachers to search for the text using keywords and calculated reading age.

Design

Inputs and Outputs

Input	Output
Title, Intended Reading Age, Keywords, & Text	Calculated Reading Age
Keywords & Calculated Reading Age	Text
Username & Password	If right they will be welcomed, else refused

File Used

The title, text, keywords, intended reading age and calculated reading age are all separated by a ";," which ensures that none of the strings get merged together and also makes it easier when pulling it out of the resource file to assign it to its string for future reference. The program pulls the data out of the text file when the program is started. It separates all the strings/integers using the ";," characters and adds it all to an array for future reference. When the user wants to change something within the data the program updates the file and adds it to the array.

Data	Type
Title	String
Text	String
Keywords	String
Intended Reading Age	Integer
Calculated Reading Age	Integer

Data Structures

Authentication			
Variable	Type	Format	Validation
Username	String		Letters only
Password	String		Letters + numbers only
Found	String	Boolean	True / False
login	Array	String (title, text, keywords, intended reading age, calculated reading age)	
login.txt	File	String (Username, password) Separated by ";,"	

Menu			
Variable	Type	Format	Validation
choice	Integer		Numbers only

Searching			
Variable	Type	Format	Validation
numbers	Array	String (number1, number2, number3…)	
keywords	String		
calculated	String		
choice	Integer		Only numbers

Calculate			
Variable	Type	Format	Validation
words	String		
sentences	Integer		
ari	Integer		

Adding Resource			
Variable	Type	Format	Validation
title	String		
text	String		
keywords	String		
age	Integer		Numbers only
calculated	Integer		Numbers only
resources	Array	String (title, text, keywords, intended reading age, calculated reading age)	
resources.txt	File	String (title, text, keywords, intended reading age, calculated reading age) Separated by ";,"	

Validation Rules

Data	Rules
Intended Reading Age	Must be integer

Keywords	Must be string
Username	Must be the same as pre-existing credentials
Password	Must be the same as pre-existing credentials
Title	Must be string
Keywords	Must be string
Text	Must be string
Calculated Reading Age (Search Function)	Must be integer

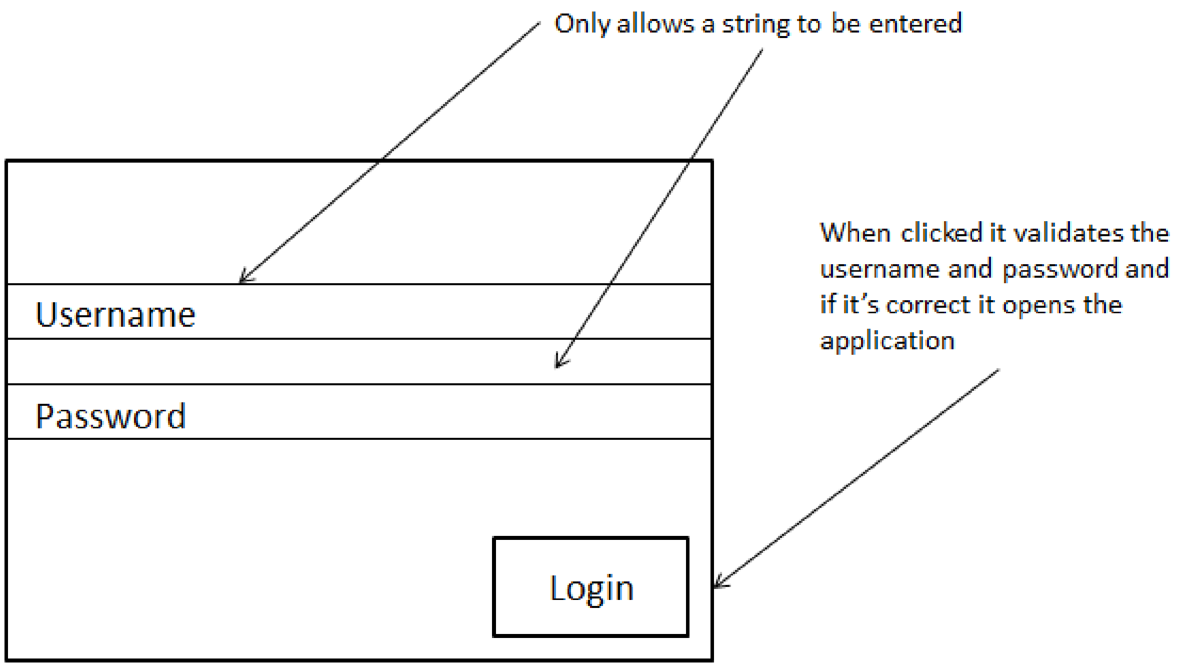
Input Formats

Input	Format	Aid
Intended Reading Age	Add to the data file	Provide information on what to enter.
Keywords	Add to the data file	Provide information on what to enter.
Username	Validate then proceed to welcome	Tell them to enter the pre-existing password for the program.
Password	Validate then proceed to welcome	Tell them to enter the pre-existing password for the program.
Title	Add to the data file	Tell them to input the title of the resource that they want to add to the program.
Keywords	Add to the data file	Tell them to input the keywords of the resource that they want to add to the program.
Text	Add to the data file	Tell them to input the text of the resource that they want to add to the program.
Calculated Reading Age (Search Function)	Validate then send to search function	Tell them to enter the calculated reading age they want to search for.
Keywords (Search Function)	Validate then send to search function	Tell them to enter the calculated reading age they want to search for.

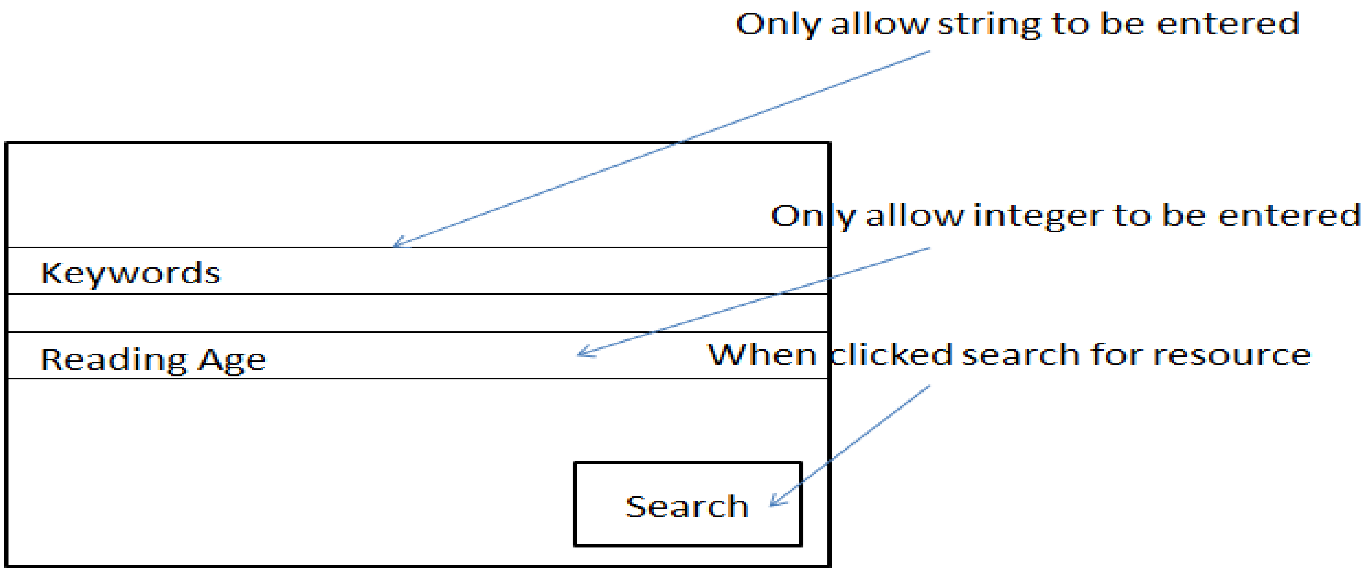
Output Formats

Output	Format
Invalid (Input)	“Invalid input, please try again.”
Invalid (Selection)	“Invalid selection, please try again.”
Invalid (Authentication)	“Invalid credentials, please try again.”
Intended Reading Age (Not Integer)	“Please input a valid reading age.”
Calculated Reading Age (Not integer (Search function))	“Please input a valid reading age.”

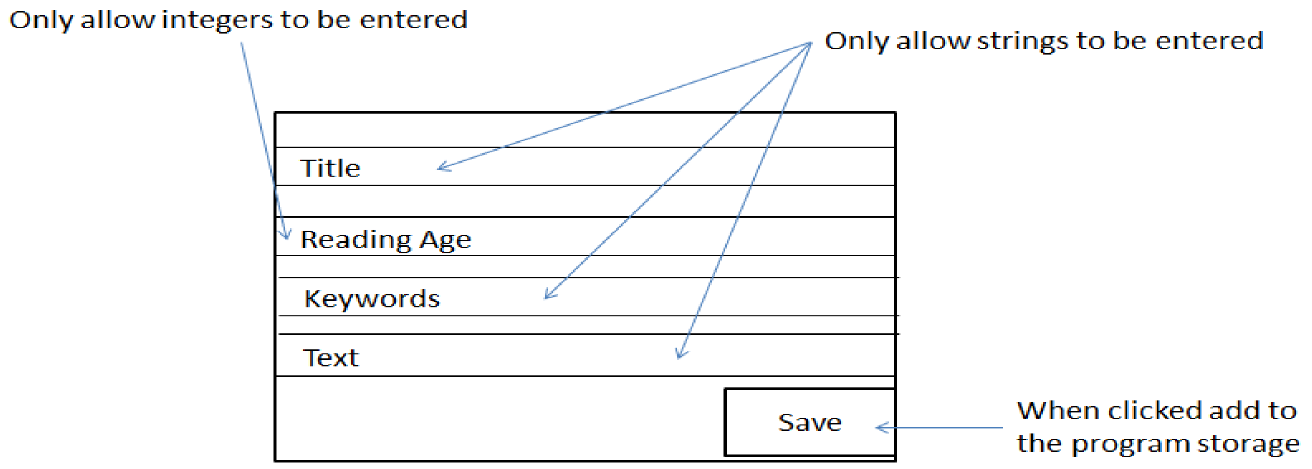
Authentication Design


Only allows a string to be entered
When clicked it validates the username and password and if it's correct it opens the application
Username
Password
Login

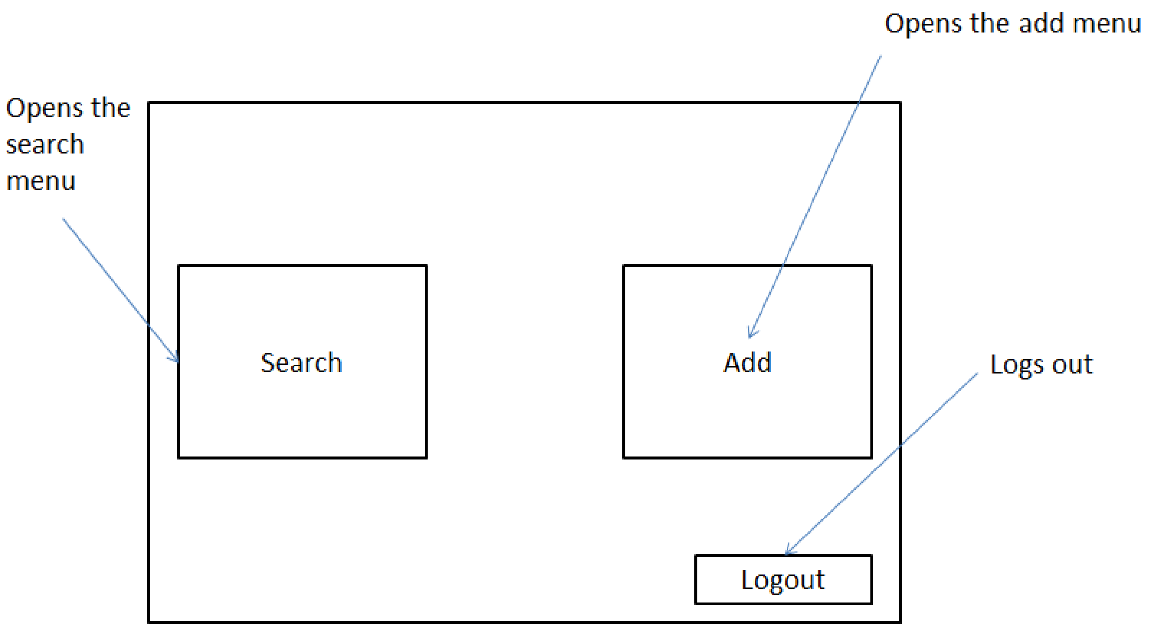
Search Design


Only allow string to be entered
Only allow integer to be entered
When clicked search for resource
Keywords
Reading Age
Search

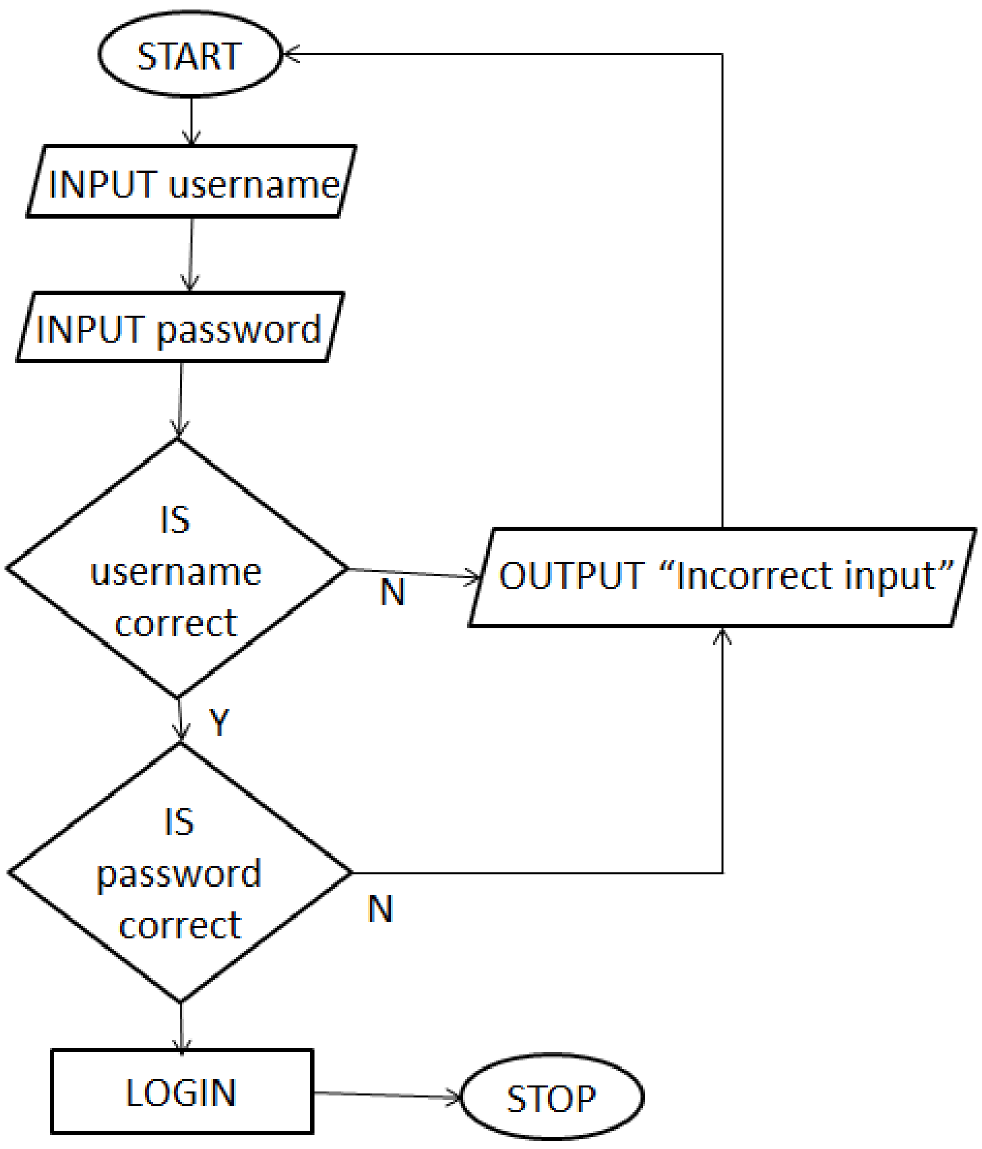
Entry Design


Only allow integers to be entered
Only allow strings to be entered
Title
Reading Age
Keywords
Text
Save
When clicked add to the program storage

Menu Design

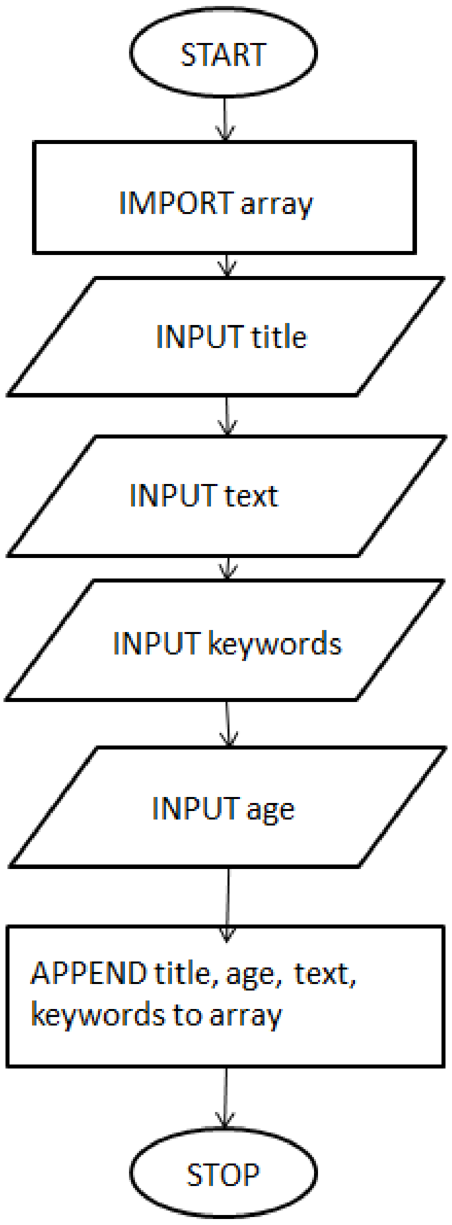


Authentication Flowchart



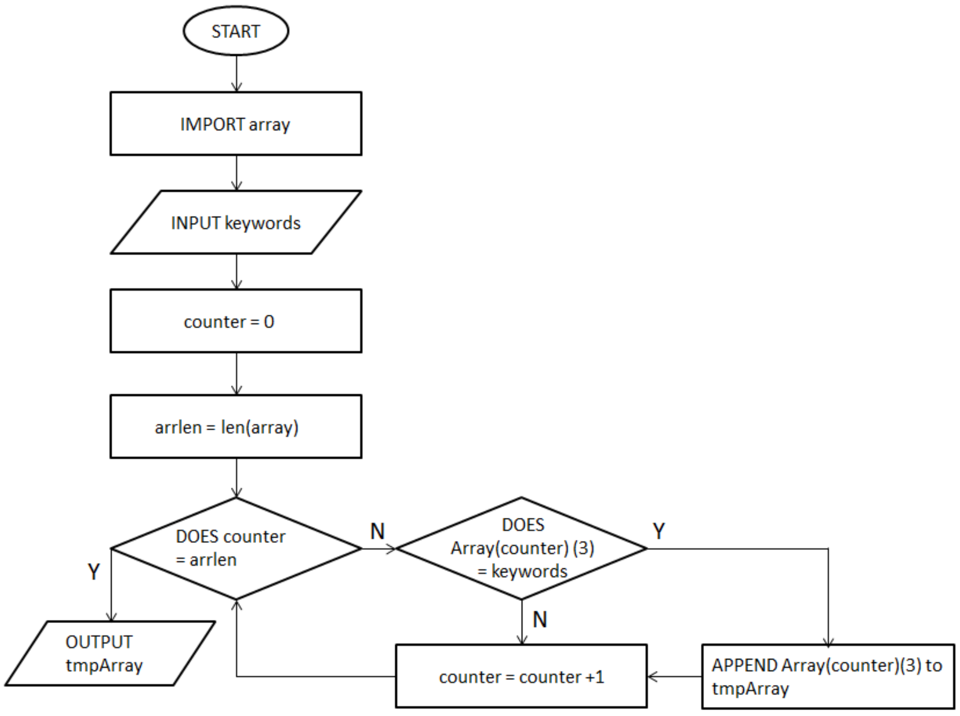
The user enters their username and password. The program then checks if the username is correct. If it’s incorrect it outputs “Incorrect input” and then goes back to the inputs. If it’s correct then the program checks if the password is correct. If it is it logs in the user and if it isn’t it outputs “Incorrect input” and sends them back to the inputs.

Entry Flowchart

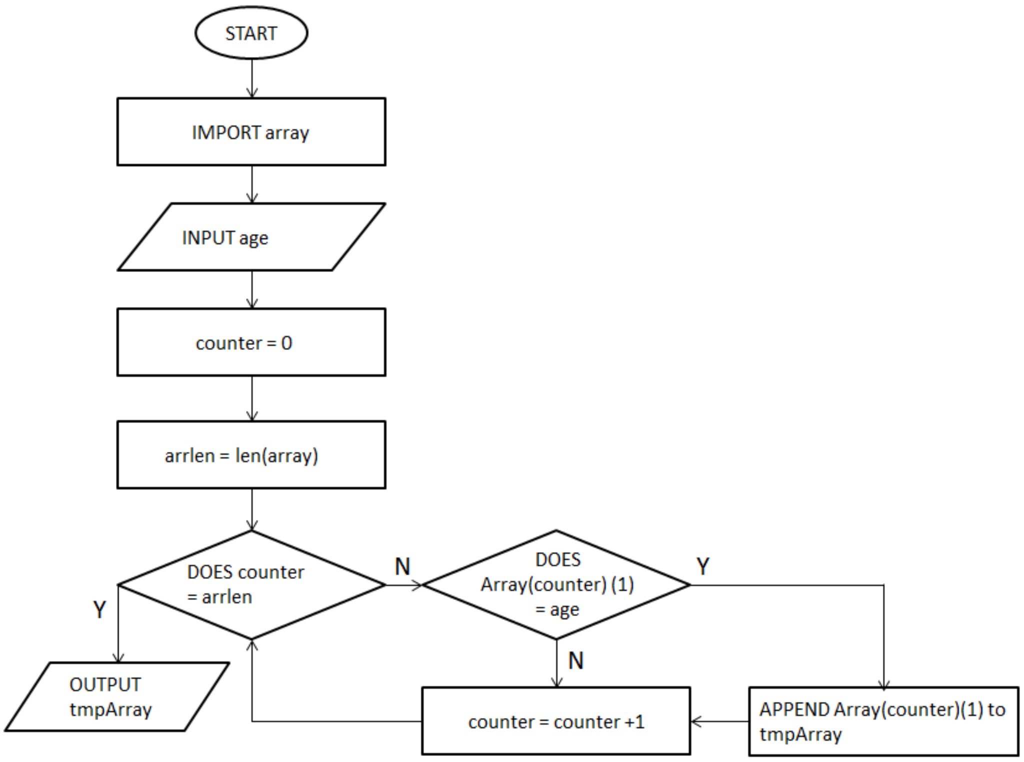


The program first imports the array which contains all the information to do with the resources. It then displays four inputs. One for the title, one for the text, one for the keywords and then one for the age. It then appends all four of the inputs to the array.

Search Flowchart

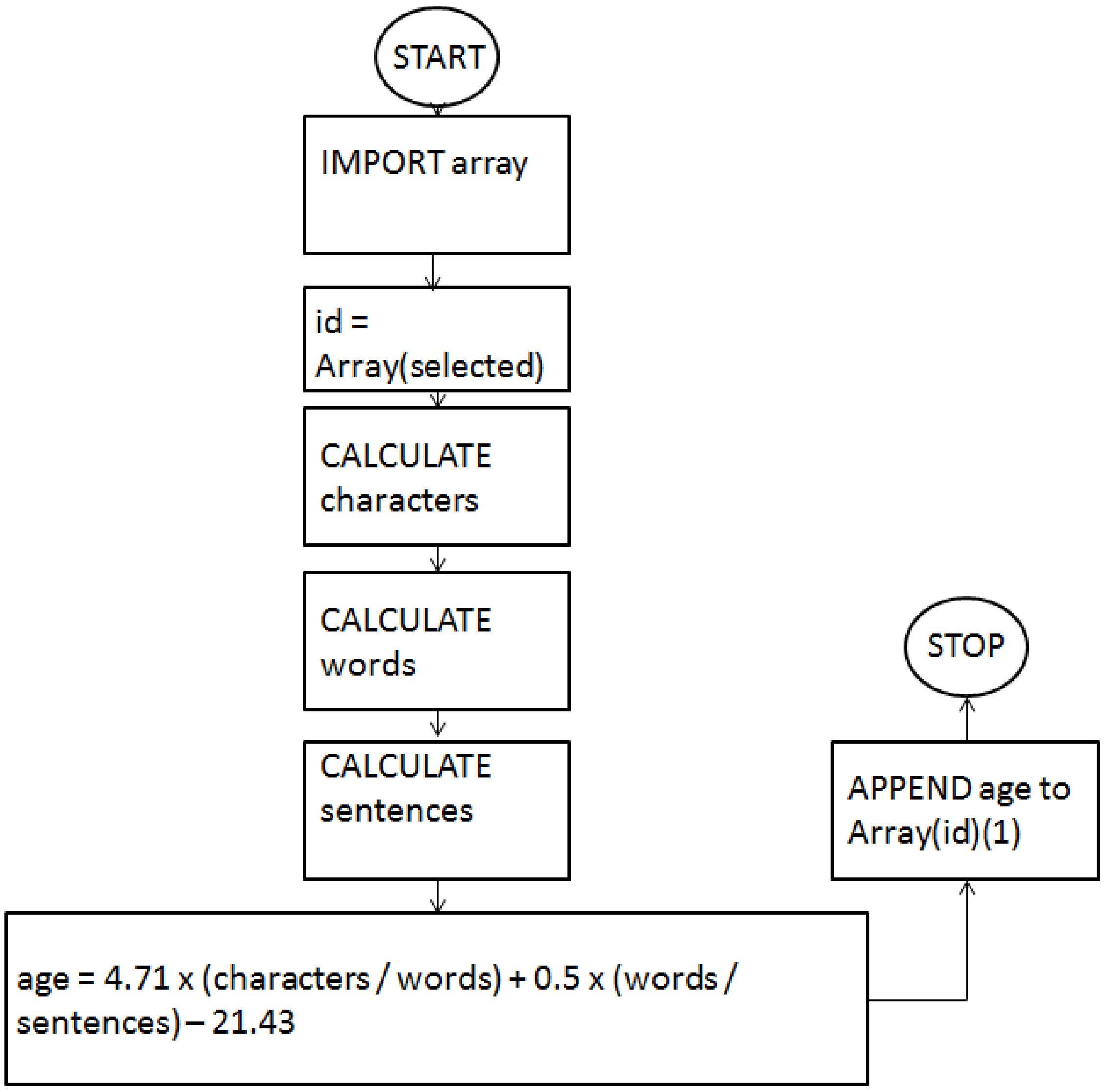


First the program imports the array and then displays an input for keywords. It then defines a counter variable which loops through the array, checking if it is the same as the length of the array. If the counter is equal to the array length then it outputs the array “tmpArray”. If it doesn’t then it checks if the array entry (equal to the counter) is equal to the input. If it is, it appends that part of the array into the tmpArray. It then adds one to the counter integer and continues the loop, checking if the counter is equal to the array length.



First the program imports the array and then displays an input for the age. It then defines a counter variable which loops through the array, checking if it is the same as the length of the array. If the counter is equal to the array length then it outputs the array “tmpArray”. If it doesn’t then it checks if the array entry (equal to the counter) is equal to the input. If it is, it appends that part of the array into the tmpArray. It then adds one to the counter integer and continues the loop, checking if the counter is equal to the array length.

Calculate Flowchart


START
IMPORT array
id = Array(selected)
CALCULATE characters
CALCULATE words
CALCULATE sentences
age = 4.71 x (characters / words) + 0.5 x (words / sentences) – 21.43
APPEND age to Array(id)(1)
STOP

The program first imports the resource array and then gets the selected resource's information from the array and gets its id saving it as a variable for later use. It then calculates the number of characters and saves it as a variable, calculates the number of words and saves it as a variable and then calculates the number of sentences and saves it as a variable. It then calculates the calculated reading age using the ARI and then appends it to the array.

Authentication Pseudocode

```
IMPORT array
INPUT username
INPUT password
found = false
FOR i to LENGTH(array) step 1
    IF array(i)(0) = username
        IF array(i)(1) = password
            found = true
            BREAK
        ELSE
            CONTINUE
    ELSE
        CONTINUE
IF found = true
    LOGIN
ELSE
    OUTPUT “Incorrect input”
```

It will import the array and then get two inputs, the username & the password. It will then loop through the array for the amount of array entries. It will then check if the user’s username is correct to the one selected in the array, if it isn’t it goes onto the next entry, but if it is correct it checks the password. If it’s correct it sets found to true and then exits the array. Otherwise it will go onto the next entry. Once it’s exited the loop it checks if found is true, if it is it logs in the user otherwise it will output “Incorrect input”

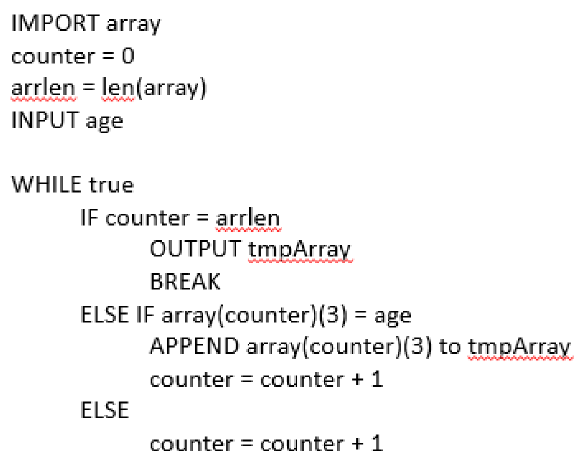
Entry Pseudocode

```
IMPORT array

INPUT title
INPUT text
INPUT keywords
INPUT age

APPEND title, text, keywords, age TO array
```

It will import the resources array and then put four inputs – title, text, keywords & age. Once they've entered that it appends it to the array.

Search Pseudocode

```
IMPORT array
counter = 0
arrlen = LENGTH(array)
INPUT keywords

WHILE true
        IF counter = arrlen
                OUTPUT tmpArray
                BREAK
        ELSE IF array(counter)(3) = keywords
                APPEND array(counter)(3) to tmpArray
                counter = counter + 1
        ELSE
                counter = counter + 1
```



```
IMPORT array
counter = 0
arrlen = len(array)
INPUT age

WHILE true
        IF counter = arrlen
                OUTPUT tmpArray
                BREAK
        ELSE IF array(counter)(3) = age
                APPEND array(counter)(3) to tmpArray
                counter = counter + 1
        ELSE
                counter = counter + 1
```

It starts by importing the array and defining three variables – counter and arrlen. counter counts the amount of iterations we've done so far and arrlen is the length of the array. It will then begin iteration and start by checking if we've completed the iterations by comparing the counter and arrlen variables. If they're the same then we'll output our temporary array and quit the iteration. Otherwise it will check if the array entry (numbered counter) is equal to the input. If it is it appends it to the array and adds one to the counter. Otherwise it will just add 1 to the counter.

Calculate Pseudocode

```
IMPORT array

INPUT selArray
INPUT text

id = array(selected)
characters = characters(text)
words = words(text)
sentences = sentences(text)

age = $4.71 \times (characters / words) + 0.5 \times (words / sentences) - 21.43$
APPEND age to array(id)(1)
```

It imports the array and then gets two inputs – selArray for the selected array and text for the text to be calculated. It then defines the variable id which gets the selected array. It then calculates the amount of characters, words, and sentences in the text and saves them as a variable. It then calculates the ARI and saves it as a variable for the text and then appends that to the resource array for later use.

Review

Testing

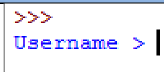
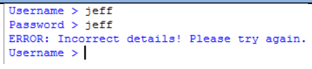
Regardless of the development methodology used, it is necessary to test my code before submitting. This is to ensure that the final solution of my code meets the requirements of the success criteria. Every time I build a part of the program I will test it to ensure that it works and fits the success criteria of being a reliable, well-structured and robust program as well as working as functioned. I will test my program by inputting random strings into inputs to test validation rules such as testing the logging in feature with username & password. I will also test inputs where an integer may be required and I will input a random string instead to test that it correctly checks the input to make sure it's an integer. This is important so it can make sure that my program hits the success criteria requirements as well as ensuring the program is reliable, well-structured and robust. If the result of the test is that it doesn't work as expected then I will go back and fix the issue and then retest the issue, ensuring that it is robust & reliable and meets the requirements of the success criteria.

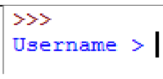
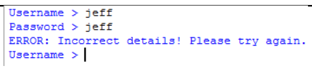
I will test each aspect of the program starting with authentication, search, adding, menu, and calculate methods and record the results of each test in the testing table below. I will test all scenarios to ensure that the program that the program hasn't got any problems with it, and if the result is that there is a problem then I will attempt to improve it. If I'm unable to fix the problem I will note it in my review/evaluation and say that it would be an improvement needed for the future.

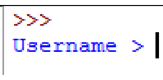
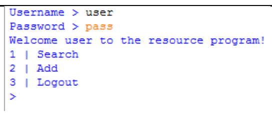
I will ensure my program hits my success criteria by testing whether you can:

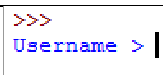
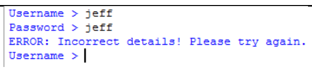
- Add a resource with the title, intended reading age, and text content.
- Login with a username and password.
- Search for the text using keywords and calculated reading age.
- Ensure the program can't be run if it can't access the login or resources file.
- Use the ARI to calculate the reading age of the text.

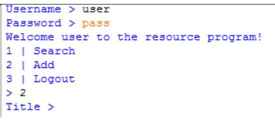
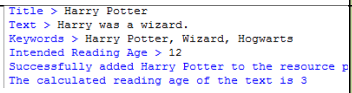
Number	Description	Before	After	Result	Explained

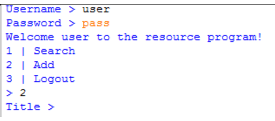
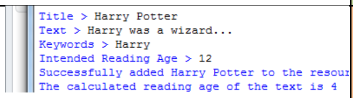
1.	Logging in with wrong username but correct password.	
		`>>>`
		`Username >`
			
			`Username > jeff`
			`Password > jeff`
			`ERROR: Incorrect details! Please try again.`
			`Username >`
				🟩 Works as expected.	This works because the program compared the username and password with all of the username and passwords in the text file and determined it isn't correct and so the program didn't let them login.

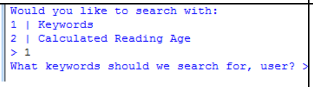
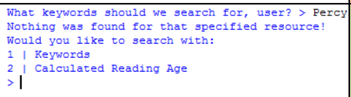
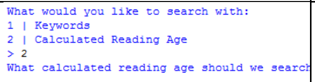
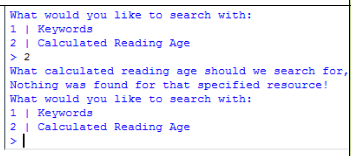
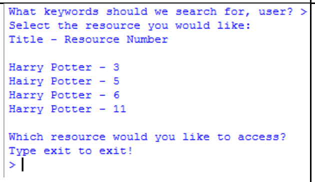
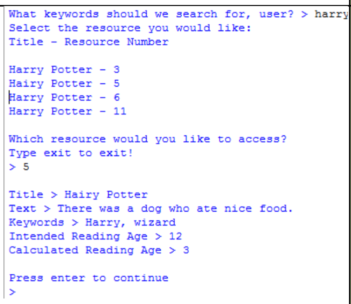
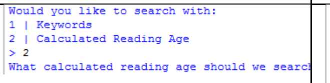
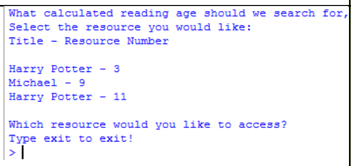
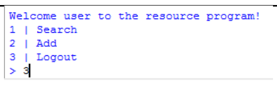
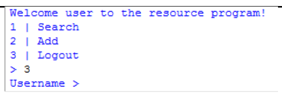
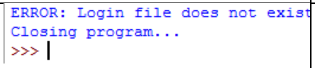
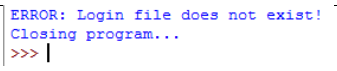
2.	Logging in with wrong password but correct username.	
		`>>>`
		`Username >`
			
			`Username > jeff`
			`Password > jeff`
			`ERROR: Incorrect details! Please try again.`
			`Username >`
				🟩 Works as expected.	This works because the program compared the username and password with all of the username and passwords in the text file and determined it isn't correct and so the program didn't let them login.

3.	Logging in with correct details.	
		`>>>`
		`Username >`
			
			`Username > user`
			`Password > pass`
			`Welcome user to the resource program!`
			`1 | Search`
			`2 | Add`
			`3 | Logout`
			`>`
				🟩 Works as expected.	This works because the program compared the inputs with all of the username and passwords in the text file / login array and checked if they were the correct username to password in order to log them in.

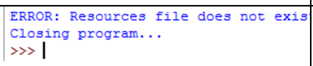
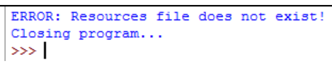
4.	Logging in with both wrong username and password.	
		`>>>`
		`Username >`
			
			`Username > jeff`
			`Password > jeff`
			`ERROR: Incorrect details! Please try again.`
			`Username >`
				🟩 Works as expected.	This works because the program compared the username and password with all of the username and passwords in the text file and determined it isn't correct and so the program didn't let them login.

5.	Adding a resource to the program.	
		`Username > user`
		`Password > pass`
		`Welcome user to the resource program!`
		`1 | Search`
		`2 | Add`
		`3 | Logout`
		`> 2`
		`Title >`
			
			`Title > Harry Potter`
			`Text > Harry was a wizard.`
			`Keywords > Harry Potter, Wizard, Hogwarts`
			`Intended Reading Age > 12`
			`Successfully added Harry Potter to the resource p`
			`The calculated reading age of the text is 3`
				🟩 Works as expected.	This works as the user is able to add a resource to the program which adds it to the text file and array without error.

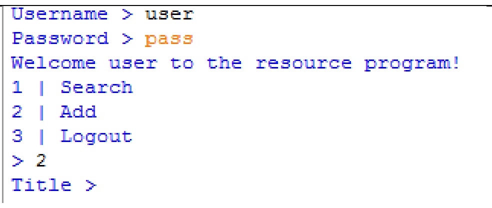
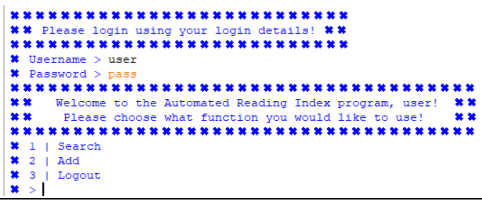
6.	Adding a resource to the program with an ellipse (…).	
		`Username > user`
		`Password > pass`
		`Welcome user to the resource program!`
		`1 | Search`
		`2 | Add`
		`3 | Logout`
		`> 2`
		`Title >`
			
			`Title > Harry Potter`
			`Text > Harry was a wizard...`
			`Keywords > Harry`
			`Intended Reading Age > 12`
			`Successfully added Harry Potter to the resour`
			`The calculated reading age of the text is 4`
				🟧 Interprets the ellipse as three sentences.	This doesn't work as intended as the program believes that there is three sentences when there is only the one.

					This could be an improvement made at a later date.
7.	Searching for a non-existent resource using the keywords.	 Would you like to search with: 1 | Keywords 2 | Calculated Reading Age > 1 What keywords should we search for, user? >	 What keywords should we search for, user? > Percy Nothing was found for that specified resource! Would you like to search with: 1 | Keywords 2 | Calculated Reading Age >	Works as expected.	It works as it checks all of the resources in the array which was pulled from the text file and sees if it contains the keywords that the user is searching for.
8.	Searching for a non-existent resource using the calculated reading age.	 What would you like to search with: 1 | Keywords 2 | Calculated Reading Age > 2 What calculated reading age should we search	 What would you like to search with: 1 | Keywords 2 | Calculated Reading Age > 2 What calculated reading age should we search for, Nothing was found for that specified resource! What would you like to search with: 1 | Keywords 2 | Calculated Reading Age >	Works as expected.	It works as it checks all of the resources in the array which was pulled from the text file and sees if it contains the calculated reading age that the user is searching for.
9.	Searching for a text using keywords.	 What keywords should we search for, user? > Select the resource you would like: Title - Resource Number Harry Potter - 3 Hairy Potter - 5 Harry Potter - 6 Harry Potter - 11 Which resource would you like to access? Type exit to exit! >	 What keywords should we search for, user? > harry Select the resource you would like: Title - Resource Number Harry Potter - 3 Hairy Potter - 5 Harry Potter - 6 Harry Potter - 11 Which resource would you like to access? Type exit to exit! > 5 Title > Hairy Potter Text > There was a dog who ate nice food. Keywords > Harry, wizard Intended Reading Age > 12 Calculated Reading Age > 3 Press enter to continue >	Works as expected.	It works as the user wanted to search for any resource with the keywords containing harry and if it does it lists them all with the option for the user to pick one from.
10.	Searching for a text using the calculated reading age.	 Would you like to search with: 1 | Keywords 2 | Calculated Reading Age > 2 What calculated reading age should we search	 What calculated reading age should we search for, Select the resource you would like: Title - Resource Number Harry Potter - 3 Michael - 9 Harry Potter - 11 Which resource would you like to access? Type exit to exit! >	Works as expected.	It works as the user wanted to search for any resource with the calculated reading age and if it does it lists them all with the option for the user to pick one from.
11.	Logging out of the program.	 Welcome user to the resource program! 1 | Search 2 | Add 3 | Logout > 3	 Welcome user to the resource program! 1 | Search 2 | Add 3 | Logout > 3 Username >	Works as expected.	It allows the user to logout and return to the menu without error.
12.	Missing login credentials file.	 ERROR: Login file does not exist Closing program... >>>	 ERROR: Login file does not exist! Closing program... >>>	Works as expected.	Checks if there's an IOError when attempting to find the login file, this could be caused by it missing or another reason. It works as it

					doesn’t let the user access the program as it could cause errors or let an unwanted user to access the resources.

13.	Missing resources file.	 `ERROR: Resources file does not exis`
		`Closing program...`
		`>>>`
			 `ERROR: Resources file does not exist!`
			`Closing program...`
			`>>>`
				Works as expected.	Checks if there’s an IOError when attempting to find the resources file, this could be caused by it missing or another reason. It works as it doesn’t let the user access the program as it could cause errors or not correctly save a resource which a teacher wanted to add or wouldn’t allow them to search for a resource either.

During testing, I noticed that program was hard to use the and the menus were hard to distinguish from each other making the program not user friendly, so I spent half an hour changing the layout of the program so that it is much easier to navigate.

BEFORE	AFTER
	

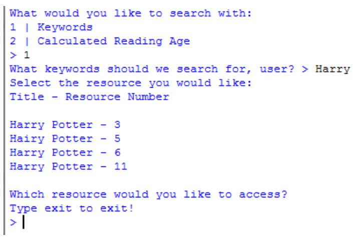
```
Username > user
Password > pass
Welcome user to the resource program!
1 | Search
2 | Add
3 | Logout
> 2
Title >
```

```
✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖
✖✖ Please login using your login details! ✖✖
✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖
✖ Username > user
✖ Password > pass
✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖
✖✖   Welcome to the Automated Reading Index program, user!  ✖✖
✖✖    Please choose what function you would like to use!    ✖✖
✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖✖
✖ 1 | Search
✖ 2 | Add
✖ 3 | Logout
✖ >
```

Success Criteria

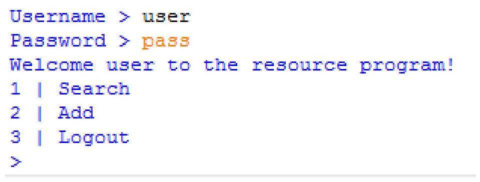
- Search for the text using keywords and the reading age.
- Allows the user to login to the program.
- Validate the inputted information.
- Must be reliable, well structured, and robust.

- The ability to search for a resource/text using keywords and the reading age is an essential part of the program and was a needed function for the program to work correctly. The user is asked first what they would like to search with – either keywords or reading age and then they enter either the keywords or the reading age. I think I have met this as I am able to search for text on the program. Here’s an example:



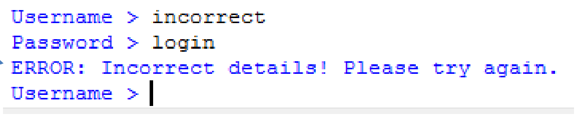
```
What would you like to search with:
1 | Keywords
2 | Calculated Reading Age
> 1
What keywords should we search for, user? > Harry
Select the resource you would like:
Title - Resource Number

Harry Potter - 3
Hairy Potter - 5
Harry Potter - 6
Harry Potter - 11

Which resource would you like to access?
Type exit to exit!
> |
```

- The program has got a function which allows the user to login to the program ensuring that nobody accesses it without a username and password. Once they've done what they've needed to do on the program they can then logout and will require another login for the functions to be accessed again. I've met this as the user is able to login:



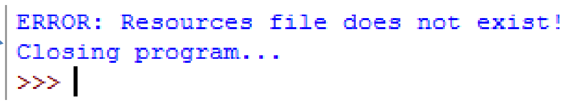
```
Username > user
Password > pass
Welcome user to the resource program!
1 | Search
2 | Add
3 | Logout
>
```

- I've added the validation of the inputted information in both the authentication and search methods to ensure that for the authentication method only people with access can enter and then with the search methods it makes sure that they enter in the correct information to retrieve the desired resource. I've met this as on for example the authentication method it checks if the information is correct before allowing them to use the program.



```
Username > incorrect
Password > login
ERROR: Incorrect details! Please try again.
Username > |
```

- It's a reliable, well-structured, and robust program as it's got its own methods to check if both of the files (login and resources) exist. If they don't the program will close down to ensure that nobody accesses the program without a login and to ensure that no errors happen. If the login or resources file is missing it shows the following message:



```
ERROR: Resources file does not exist!
Closing program...
>>> |
```

Good Features

- The program is able to check if it can access the resource and login files and if it can't it'll close the program down so no resources are lost.
- The appearance is aesthetically pleasing and user friendly, especially after I improved it using feedback from a user.
- It's able to validate data such as the search and login function.

Improvements Needed

- Ensure that ellipses are interpreted as three sentences. This can cause an issue when calculating the ARI as it uses the amount of sentences to increase/decrease the calculated reading age which, if it has an ellipses in, can cause an invalid result.
- Make it so that when you copy the text into the resource it interprets the new lines in it as separate inputs which usually causes an error with the program so it doesn't run properly.
- Being able to edit other inputs once you've pressed enter.

Future Features

- The ability to add and delete logins for multiple users so that you can control the users which can access the resources, instead of having to modify the text file with the login details.
- Store the login / resource data within a database as you can password protect a database but you can’t protect a text file.
- The ability to delete a resource that maybe isn’t needed anymore isn’t used or is incorrect.
- The ability to edit/change a resource to modify it so that it’s correct or to add an addition to it.
- Having admin accounts to be able to control logins as well as being able to add or delete the accounts for people to be able to add users.
- The ability to control who has access to each resource and the admin account could give new users access to the resource as well as the admin account being able to access and modify the private or one user only resource.
- When adding a resource the program updates the file but it would be more effective to update the resource array and then update the file only once when the program closes so it isn’t constantly updating the resource file which in some cases could cause programs, slowing down the program/computer.