Web Browser

Cameron Bell
cjb15@hw.ac.uk

Introduction

The purpose of this report is to showcase the web browser application created for this project. Several assumptions we're made during development such as this app being intended for more technical users rather than a general audience. Such the program mainly used a generic clean design rather that something more stylised. Also errors aren't explained super thoroughly such as what a 404 page not found means or what makes a valid URL. It was also assumed that pages loaded from bulk files shouldn't be added to the users history as the page content isn't displayed to the user.

Requirement's Checklist

No.	Requirement	Achieved?
1	Sending HTTP request messages for specified URLs.	Yes
2	Receiving HTTP responses and displaying the HTML code of the requested page.	Yes
3	Receiving HTTP responses and displaying the response status code (a.k.a. 404 Not Found) as well and the title of the loaded page.	Yes
4	User can set a home page which they can change, this page loads when the application is opened.	Yes
5	The user can save URLs as favourites, give these pages associated names, load pages by selecting them as well as edit and delete them.	Yes
6	The browser should log each search query, allow the user to view and load previous searches.	Yes
7	The user can quickly access prevision pages via a previous and next pages button.	No
8	The user can link to a file of URLs, this will then download each page displaying the response code, its number of bytes and the URL used.	Yes
9	The program has a GUI that allows the user to access all functions of the program, allowing them to use some shortcut keys along side the on screen buttons.	Yes

The only requirement not fully met is the back button one, this is simply due to misreading the requirements and missing that part and not having sufficient time to add it in latter.

Design Considerations

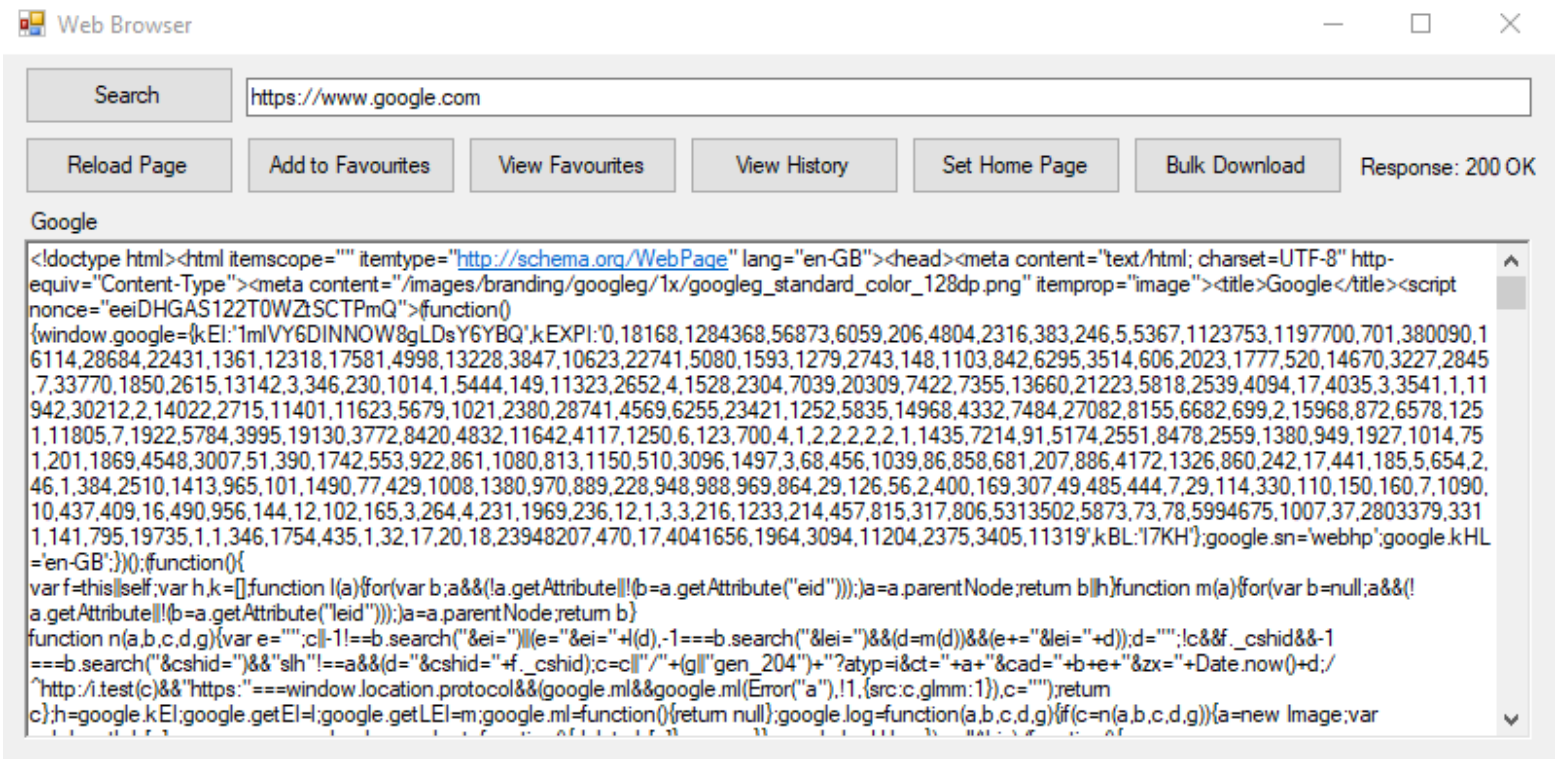
My approach when developing this app was to have only the bare essential code in the form .cs files, if code wasn't related to changing the appearance of one of the forms then it would be put in one of the relevant other non form .cs files, then if data needed to be shown or collected from the user these classes would create a new form. This helped to keep the form files clean and reduce duplicate code as most of the code related to processing data would be in functions that could be called in multiple places. This did create the issue that the main form couldn't be updated while data was being processed in a different class file, such as when bulk downloading it would be convoluted to constantly exit the class file back to display how the downloading precess was going. One solution could have been to move the code for bulk downloading back to the main form or into into a new form that only contains a loading bar, this is something that I would have experimented with more but haven't due to time constraints.

As mentioned before the GUI was designed to be clean, simple and mainly uses the generic windows forms aesthetic as it's was expected that this would mainly be used by technical users and not by the general public. If the window sizes are adjusted the size of textboxes will also be resized, allowing the user to use whatever window size they want.

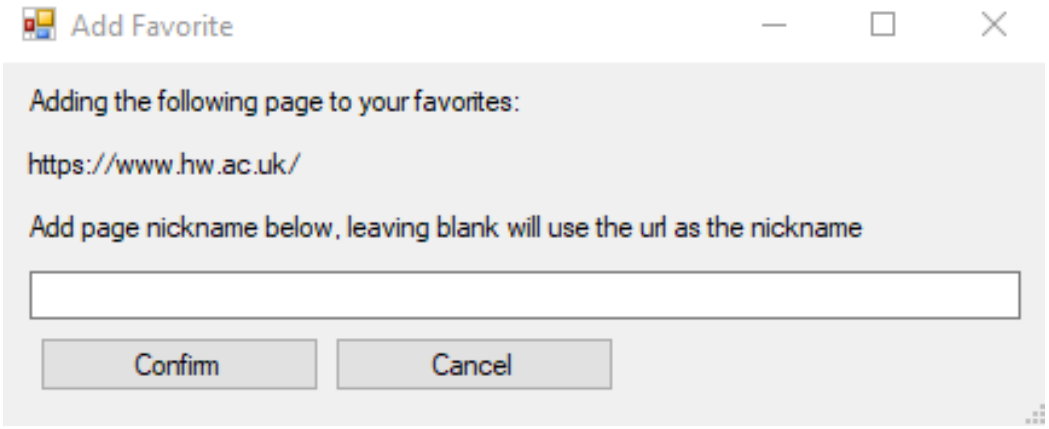
For storing data for both the favourites and history an object was created to hold the variables for 1 entry, such as favEntry which contains the strings favName and favUrl, each favourite entry is then stored in a list. This list is then serialized into JSON format and saved as a string using the user settings feature that .NET which stores the variable in Settings.settings. When the program is opened the history and favourite JSON files are read and the JSON contents deserialized back into lists of objects.

The home page URL is stored using the user settings feature aswell.

User Guide

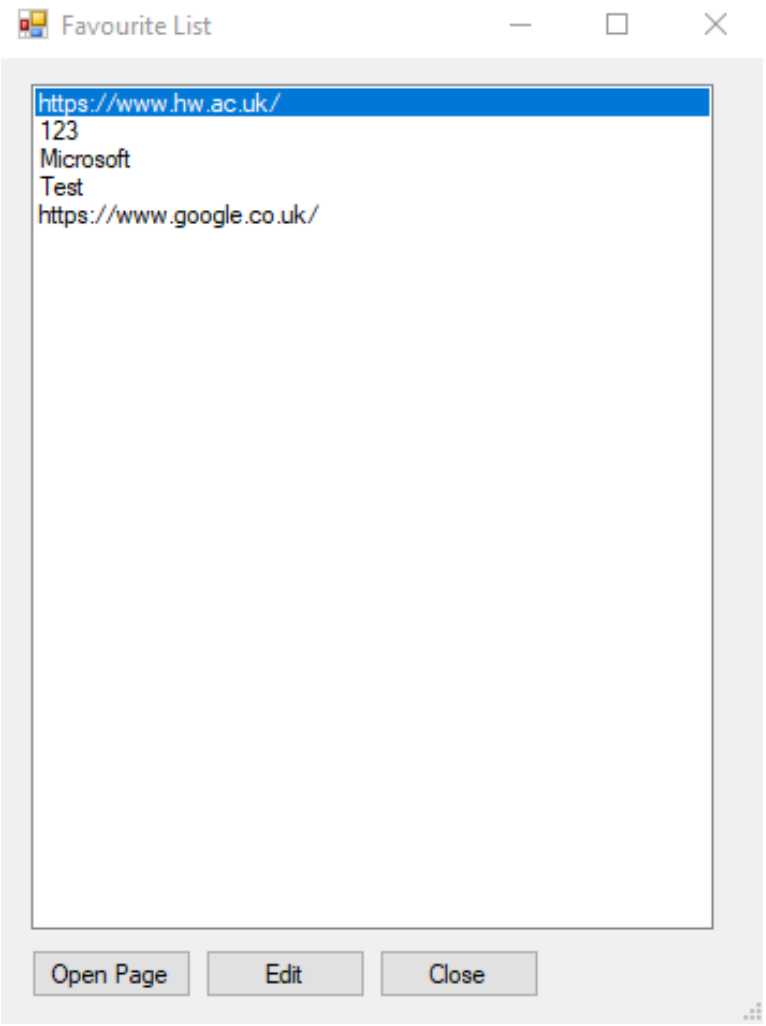


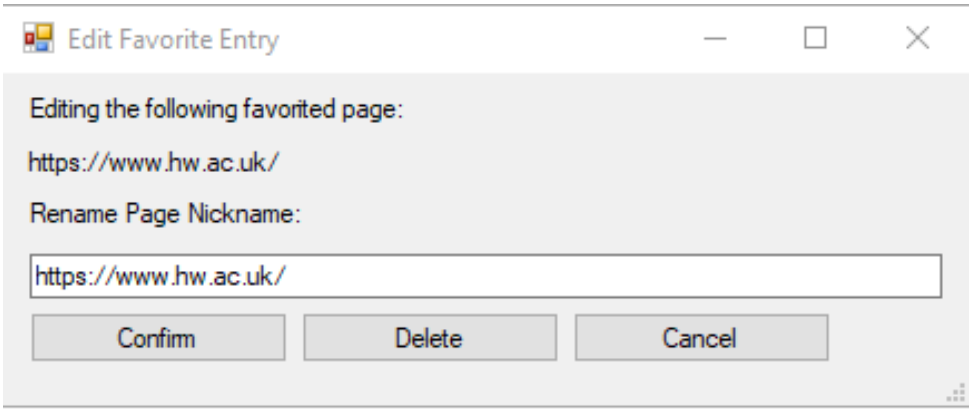
This is the main browser page.
At the top of the page is the search box, here the user would enter the URL of the page they wish to view.
The “Search” button to the left of the box will search for the page entered into the search box, the enter button can also be pressed while editing search box to search.
The “Reload Page” button under the search button will re-search for the current loaded regardless of what is in the search box.
The “Add to Favourites” will display a menu that allows the user to add the currently loaded page to their favourites list, if the current page is already in their favourites list then this button will be disabled.
The “View Favourites” button will display the list of pages that the user has in their favourites list.
The “View History” button will display the list of all pages the user has visited previously.
The “Set Home Page” will set the current page as the home page, which will be loaded when the program is first booted up. This button will be disabled If the current page is the hoe page.
The “Bulk Download” button opens a window allowing the user to enter a file to be bulk downloaded.
The text to the left of the bulk download button will display the response code of the webpage if it was successfully reached or display the error if one occurs.
The text below the buttons is the title of the loaded webpage if one was found.
The text box below everything else if the html box, this contains the html code returned from the searched web page and the results from the bulk download.



This is the add favourite dialogue window.
The user can type a nickname for the favourite entry, if left blank then the URL itself will be used as the nickname.
If the user presses the “Confirm” button or presses the enter key then the nickname will be entered and the window will close.
If the “Cancel” button or the escape key is pressed then the window will be closed without the entry being added to the favourite list.

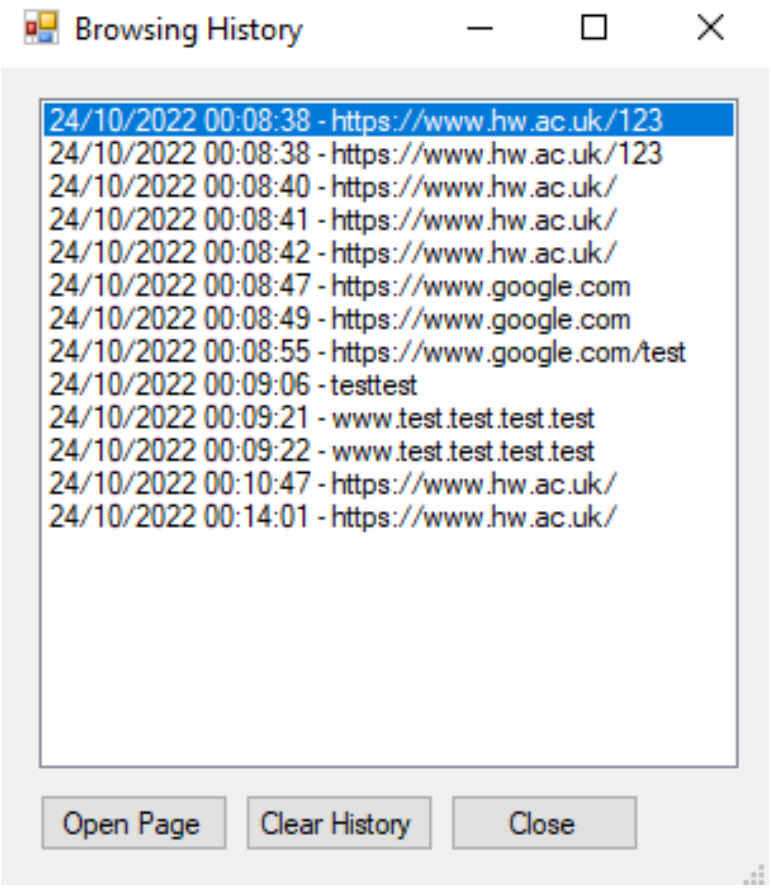
This is the favourites list.
It lists each webpage that the user has favourited, displaying the nickname the user gave them or the URL itself if one wasn’t given.
An entry can be selected by clicking on it or by using the up and down arrow keys.
An entry can be opened back on the browser window by clicking the “Open Page” button or by pressing Enter.
By pressing the “Edit” button a window to edit the selected entry will appear.
By pressing the “Close” button or the escape key the window will close and focus will return to the browser window.

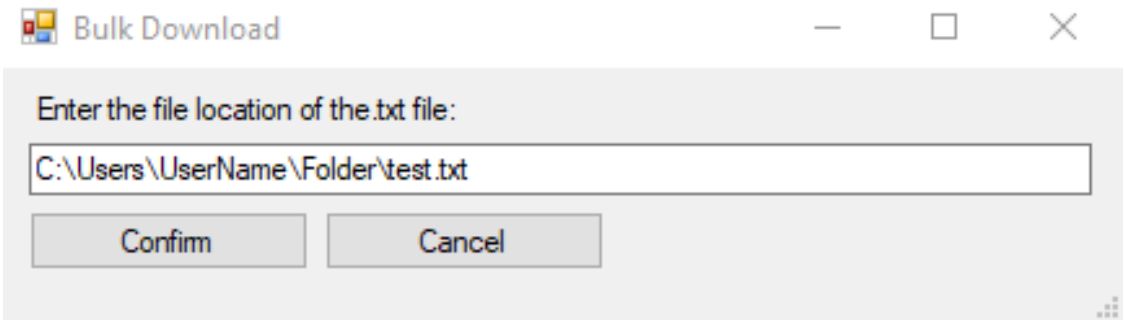




This is the favourite editing dialogue box.
The user can edit the entry's nickname by editing the text in the textbox.
The user can confirm the changes to the nickname by pressing the "Confirm" button or by pressing the enter key.
The user can delete the entry by pressing the "Delete" button, this will remove it from the favourite list.
The user can press the "Cancel" button or press the escape key to close the window without changing the nickname.

This is the browsing history window.
It lists each webpage that the user has visited, displaying the time the user gave searched for it along side the URL itself.
An entry can be selected by clicking on it or by using the up and down arrow keys.
An entry can be opened back on the browser window by clicking the "Open Page" button or by pressing Enter.
By pressing the "Clear History" button the browsing history will be wiped leaving the list blank.
By pressing the "Close" button or the escape key the window will close and focus will return to the browser window.





This is the bulk download dialogue box.
The user can enter the file location of the file to be loaded, it should be typed in the format shown above.
The file should be a text file (.txt) with a list of URLs each on the next line than the last.
The user can confirm the selection by pressing the “Confirm” button or by pressing the enter key, this will display the results on the browser window.
The user can cancel the bulk download by pressing the “Cancel” button or pressing the escape key.

Developer Guide

The application consists of a main browser window form, here only code related to chaining said main window is kept here. All the code related to editing and processing the favourites list is kept within the UpdateFavList class, same for the history list with UpdateHistoryList class, all the code related to sending and receiving the website data in the WebLookUp class and all code relsted to bulk downloading in the BulkDownload class. When the program is first booted a UpdateFavList, UpdateHistoryList and WebLookUp objects are all created where as bulk download create a new object when required. These objects will then call dialogue windows when ever they need user input.

Both the favourites list and history list are lists of objects in their retrospective update list class, they are loaded from user settings strings which is deserialised (using System.Text.Json) and loaded into a list. These lists consist of either favEntry or historyEntry objects. With favEntry consisting of 2 strings, the entries URL and Nickname, and historyEntry also consisting of 2 string, the entries URL and message consisting of the url + date and time accessed). When ever the lists are updated relevant list is reserialised and overwrites the .json file.

The System.Net.Http HttpClient() is used to request and receive the web pages, returning the HTML code and response code. It's functionality is also used confirm the URL is valid.

Windows forms is used for the GUI of the program.

A more thorough breakdown of the code can be found in the code itself by viewing its internal commentary.

Tetsing

Action Taken: Entered “https://www.hw.ac.uk/” into the URL search box and pressed search button
Outcome expected: Displays the HTML code from the Heriot Watt main page
Actual outcome: Displays the HTML code from the Heriot Watt main page
Success

Action Taken: Entered “https://www.hw.ac.uk/123” into the URL search box and pressed search button
Outcome expected: Displays the HTML code from the Heriot Watt page not found page and displays the 404 error not found
Actual outcome: Displays the HTML code from the Heriot Watt page not found page and displays the 404 error not found
Success

Action Taken: Entered “https://www.website.dosnt.exist/” into the URL search box and pressed search button
Outcome expected: No html code is displayed and the page not found error is displayed
Actual outcome: No html code is displayed and the page not found error is displayed
Success

Action Taken: Entered “NotValidUrlFormat” into the URL search box and pressed search button
Outcome expected: No html code is displayed and the invalid URL error is displayed
Actual outcome: No html code is displayed and the invalid URL error is displayed
Success

Action Taken: Entered “https://www.hw.ac.uk/” into the URL search box and pressed the enter key while still editing the URL search box
Outcome expected: Displays the HTML code from the Heriot Watt main page
Actual outcome: Displays the HTML code from the Heriot Watt main page
Success

Action Taken: Entered nothing into the URL search box, it is blank
Outcome expected: The search button and search with enter key is disabled
Actual outcome: The search button and search with enter key is disabled
Success

Action Taken: With the “https://www.hw.ac.uk/” page loaded and “NotValidUrlFormat” entered into the URL text box.
Outcome expected: It should re-load “https://www.hw.ac.uk/” and not load “NotValidUrlFormat” which would cause an error
Actual outcome: It re-loads “https://www.hw.ac.uk/”
Success

Action Taken: With the “https://www.hw.ac.uk/” page loaded press the add to favourites button
Outcome expected: Opens the add favourites dialogue box asking the user to add “https://www.hw.ac.uk/” to their favourites
Actual outcome: Opens the add favourites dialogue box asking the user to add “https://www.hw.ac.uk/” to their favourites
Success

Action Taken: The user confirms the option to add “https://www.hw.ac.uk/” to their favourites, no nickname is entered
Outcome expected: Close the window, now the add to favourites button is disabled
Actual outcome: Close the window, now the add to favourites button is disabled
Success

Action Taken: The user presses the view favourites button after adding “https://www.hw.ac.uk/” without any nickname.
Outcome expected: Open the favourites list displaying only “https://www.hw.ac.uk/”
Actual outcome: Open the favourites list displaying only “https://www.hw.ac.uk/”
Success

Action Taken: The user presses the open page button on the favourites list with “https://www.hw.ac.uk/” selected
Outcome expected: Close the favourites list and load “https://www.hw.ac.uk/”, displaying it’s HTML code
Actual outcome: Close the favourites list and load “https://www.hw.ac.uk/”, displaying it’s HTML code
Success

Action Taken: The user presses the edit button on the favourites list screen with “https://www.hw.ac.uk/” selected
Outcome expected: Close the favourites list and open the edit favourite screen to edit entry “https://www.hw.ac.uk/”
Actual outcome: Close the favourites list and open the edit favourite screen to edit entry “https://www.hw.ac.uk/”
Success

Action Taken: The user presses the edit button on the favourites list screen with “https://www.hw.ac.uk/” selected
Outcome expected: Close the favourites list and open the edit favourite screen to edit entry “https://www.hw.ac.uk/”
Actual outcome: Close the favourites list and open the edit favourite screen to edit entry “https://www.hw.ac.uk/”
Success

Action Taken: The user presses the delete button on the edit favourite screen to edit “https://www.hw.ac.uk/”
Outcome expected: Close the edit favourite screen and remove “https://www.hw.ac.uk/” from the favourite list
Actual outcome: Close the edit favourite screen and remove “https://www.hw.ac.uk/” from the favourite list
Success

Action Taken: The user presses View History button on the main browser window
Outcome expected: Open the browsing history displaying all the previously viewed pages and the date/time accessed
Actual outcome: Open the browsing history displaying all the previously viewed pages and the date/time accessed
Success

Action Taken: The user presses clear history button on the browsing history menu
Outcome expected: All entries from the history list are wiped leaving it blank
Actual outcome: All entries from the history list are wiped leaving it blank
Success

Action Taken: The set home page is pressed while "https://www.hw.ac.uk/", the current home page, is loaded
Outcome expected: Nothing the button is disabled
Actual outcome: Nothing the button is disabled
Success

Action Taken: The set home page is pressed while "https://www.hw.ac.uk/123", which is not the current home page, is loaded
Outcome expected: "https://www.hw.ac.uk/123" is set as the new home page and loads when the program first loaded.
Actual outcome: "https://www.hw.ac.uk/123" is set as the new home page and loads when the program first loaded
Success

Action Taken: The bulk download button is pressed
Outcome expected: The bulk download dialogue box appears asking the user for a file location
Actual outcome: The bulk download dialogue box appears asking the user for a file location.
Success

Action Taken: The bulk download dialogue box is given a .txt file that contains:
"https://www.hw.ac.uk/
https://www.hw.ac.uk/
https://www.hw.ac.uk/
https://www.hw.ac.uk/
https://www.hw.ac.uk/"
Outcome expected: The bulk download dialogue box closes and the html text box displays:
"Response: 200 OK - [size of HW website] bytes - https://www.hw.ac.uk/
Response: 200 OK - [size of HW website] bytes - https://www.hw.ac.uk/
Response: 200 OK - [size of HW website] bytes - https://www.hw.ac.uk/
Response: 200 OK - [size of HW website] bytes - https://www.hw.ac.uk/
Response: 200 OK - [size of HW website] bytes - https://www.hw.ac.uk/"
Actual outcome: The bulk download dialogue box closes and the html text box displays:
"Response: 200 OK - 171833 bytes - https://www.hw.ac.uk/
Response: 200 OK - 171833 bytes - https://www.hw.ac.uk/
Response: 200 OK - 171833 bytes - https://www.hw.ac.uk/
Response: 200 OK - 171833 bytes - https://www.hw.ac.uk/
Response: 200 OK - 171833 bytes - https://www.hw.ac.uk/"
Success (Seemingly as size of webpage in bytes hasn't been confirmed)

Conclusion

The program created meets most of the requirements with the exception of the previous page arrow most browsers have. There were many features considered that were such as the loading bar screen to indicate the loading time of the bulk downloading to let the user know it hasn't frozen, including a proper file explorer for when the user is selecting a file for the bulk downloader rather than requiring the exact location be typed/coped in or including a setting menu to allow manual changing of the home page as well as other options such a dark mode. In retrospect I could also have tried storing the These could have been implemented but were not due to time constraints.

I am proud of how I handled storing and loading of the favourites list and history list as well as the well as the actual URL searching itself. The program is also never encountered any major crashes during testing.