C# Web Browser

Industrial Programming

24/10/2021

Created and devloped by
Musa Mohammadi

Table of Contents

Introduction 3
Purpose: 3
Assumptions: 3
Requirements Checklist 4
Design Considerations 5
GUI design considerations 5
Multiple windows forms 5
Icons 5
Colours 5
Other remarks 5
Code design considerations: 6
Class design 6
Data structures 6
HTTP 6
User Guide: 7
Search for a page 7
Change Home Page 7
Add/Edit/Access/Remove Favourites 8
Accessing your history: 9
Initiating bulk downloads: 9
Keyboard Shortcuts 10
Developer Guide: 11
Favourites 11
History 11
Main Browser 12
Settings 12
HTTP 12
Testing 13
Conclusion 14
Things I am happy with 14
Things I would have liked to have done differently 14
Appendix 15

Introduction

Purpose:

The purpose of this report is to reflect on my implementation of a web browser in C#. The reflection will cover these main points:

- Requirement's checklist
- Design Considerations
- Testing
- Conclusions

This report will also contain documentation relating to my program, these are:

- User Guide
- Developer Guide

The report is intended to be read by Hans-Wolfgang Loidl.

Assumptions:

This section will cover assumptions I made with regards to the development process of my web browser.

Some assumptions I made are as follows:

1. I assumed the use of external txt files to save history, home page and favourites was allowed
2. I assumed the use of multiple windows forms was allowed
3. I assumed my implementation of history is allowed. My history adds previously visited pages as a new occurrence to the history. But when using the previous page button, instead of going back to the previous page, the program navigates the user to the second most recent visited page in the history.
4. I assumed my use of the WebResponse class was allowed
5. I assumed putting the status code in the form title was acceptable

Requirements Checklist

This section covers which requirements I have implemented in my web browser and any comments with regards to the implementation process/functionality of the feature.

Section/Feature	Description	Implemented(Y/N)	Comments
HTTP Reponses			
Sending HTTP request messages	URLs typed by the user	Y	
Receiving HTTP response messages	Display the contents of the messages on the interface	Y	
Support relevant status codes	– 200 Okay – 400 Bad Request – 403 Forbidden – 404 Not Found	Y	
Display			
Display the status code and title	Display title at top of browser main window	Y	
Display contents of the page	Display the HTML code from the web server	Y	
Reload the current page	Send another HTTP request for the current web page	Y	No dedicated refresh button but users can press the search button again to re-load the search
Home Page			
Create and edit home page URL		Y	Modification is done in settings form
Home page is initialized and loaded upon startup	Home is Initialized with my university home page	Y	
Favourites			
Add a URL to the list of favourites		Y	Addition is completed via favourites tab in settings form. Favourites are saved in an external txt file
Associate a name with each favourite URL		Y	Naming is completed via favourites tab in settings form.
Modify/delete favourite		Y	Modification is done in favourites tab in settings form
Request favourite by clicking on its name in favourites list		Y	This action is completed in the favourites form
History			
Navigate to next and previous pages in history		Y	Navigation can be done in history form or via previous and next page buttons on main form
Maintain a list of URLs		Y	Modification done in settings form in history tab
Bulk Download			
User can specify filename to initiate bulk download	Default file is “bulk.txt”	Y	

Retrieve pages and display in relevant format	One line should be shown for every entry in the bulk download file	Y	
Graphical User Interface			
GUI allows users to perform operations stated above		Y	
Make use of menus (with appropriate shortcut keys) as well as buttons to increase accessibility.		Y	No use of menus. However, use of various GUI tools such as list view box, tab control, text boxes etc.

Design Considerations

This section covers the design considerations I made with regards to both the code aspect of the program and the graphical user interface (GUI).

GUI design considerations

My main goal with the GUI was to provide a simple, yet elegant GUI for the user to use that would make identifying individual components of my browser easy to identify and use.

Multiple windows forms

My browser utilizes 4 main forms. The forms being the main browser form, a favourites form, a history form and a settings form. The rationale behind splitting up different sections in forms was to provide a clean user experience and prevent forms from being cluttered with lots of information/sections. It also aids in my overall goal of my GUI to be easy to use by having dedicated forms for specific functions.

Icons

The main form makes use of large icons. The decision to use simple, large icons was to help to make certain sections easy to identify, thus contributing to my overall goal of a simple look.

Colours

My overall colour scheme is a dark mode inspired colour palette. Dark mode is a personal preference for me. I have found the contrast between my dark grey background and my grey/white interaction buttons provides a nice contrast that makes certain sections of the forms easy to follow.

Other remarks

I used multiple display components for my GUI, some of these are:

- ListView boxes – useful for putting data in set columns
- Text boxes – good for large bodies of text
- Combo boxes – used for lists of information

Code design considerations:

Class design

My initial decision was to separate the code into 5 separate classes to each represent the major functions of the browser. I decided to create 5 classes, with 4 forms. The class without a form is the HTTP class that handles the HTTP requests and responses.

The classes themselves contain functions that are used multiple times to achieve an overall efficient running program. Some classes could be optimized better to have more functions, some classes have a lot of functionality contained within the buttons which is not ideal.

Data structures

To keep a record of the user's history, favourites and homepage, I decided to save the data in external txt files. The reason being was from a performance standpoint, txt files do not rely on heavy libraries to operate so it is computationally well balanced. It also makes the browser accessible on various machines and operating since txt files are standalone.

The data in the homepage and history txt files are saved as URLs with line breaks. To deal with favourites having a name and a URL associated with each instance, I decided to save the names for each URL in the even indexes of the txt file, and the URLs in the odd indexes of the txt file. The format can be seen below in figure 1.1.

```
News
https://www.bbc.co.uk/news
Amazon
https://www.amazon.co.uk/
Youtube
https://www.youtube.com/?hl=en-https://www.youtube.com/
Wikipedia
https://en.wikipedia.org/wiki/Main_Page
wikiHow
https://www.wikihow.com/Main-Page
```

Figure 1.1

The reason behind saving the data in this way was to make it at least computationally taxing as possible. I considered other possibilities such as using two txt files, one for names, one for URLs but I decided against it as I thought reading in from two separate files may be confusing.

The data from the txt files are read into an array. The reason being was that I was comfortable with using arrays. As development proceeded, arrays became harder to work with since they have limited functionality compared to something like a linked list or Dictionary. I did use a list on a few occurrences since arrays were not suffice.

HTTP

To send HTTP responses and receive HTTP messages, I utilized the WebResponse class from System.Net. This library worked well when getting and receiving as the functions associated with the library were clear and easy to understand. For the request I used the following piece of code in figure 1.2 to generate a response.

```
Uri addr = new UriBuilder(url).Uri;
HttpWebResponse response = HttpWebRequest.Create(addr).GetResponse() as HttpWebResponse;
```

Figure 1.2

From this response, I could get the response stream which is used to read the body of the response from the server.

```
Stream receiveStream = response.GetResponseStream();
```

Figure 1.3

User Guide:

Search for a page

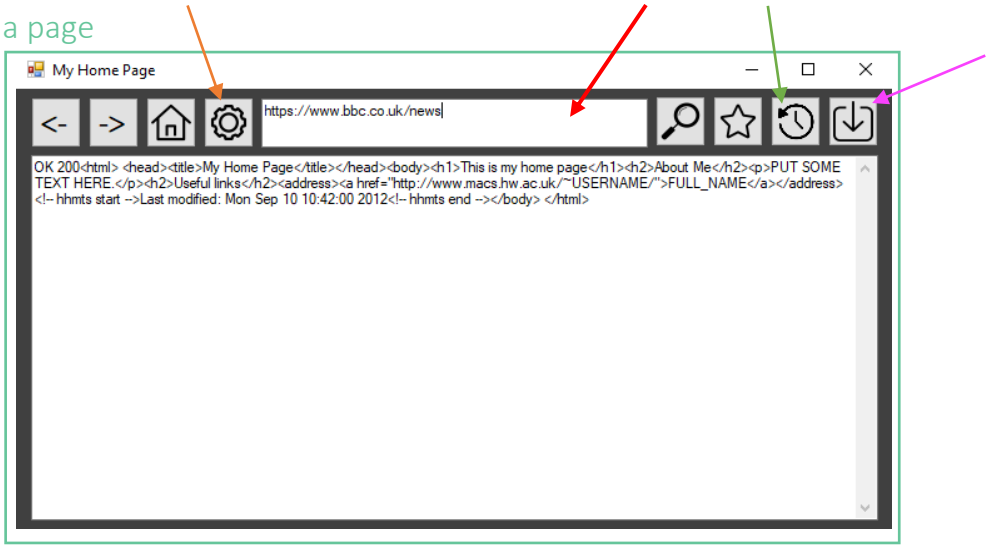

Figure 2.1

To retrieve a HTTP web response, the user can enter a URL into the input box as shown in figure 2.1, the input box is at the top of the form and can be identified by the red arrow. Once the user has entered in their specified URL, they can initiate the search by pressing the magnifying glass icon. Once loaded, the status code of the web response will be displayed as well as the corresponding HTML code for the page. A sample output for the URL in figure 2.1 can be found in the appendix at figure 5.1.

Change Home Page

Homepage allows users to set a specified URL to the home button which they can access quickly and easily.

To change the user's homepage, a user must first open the settings form by clicking on the cog wheel highlighted by the orange arrow in figure 2.1. Upon opening the settings form, the user is greeted with the form shown in figure 2.2 below.

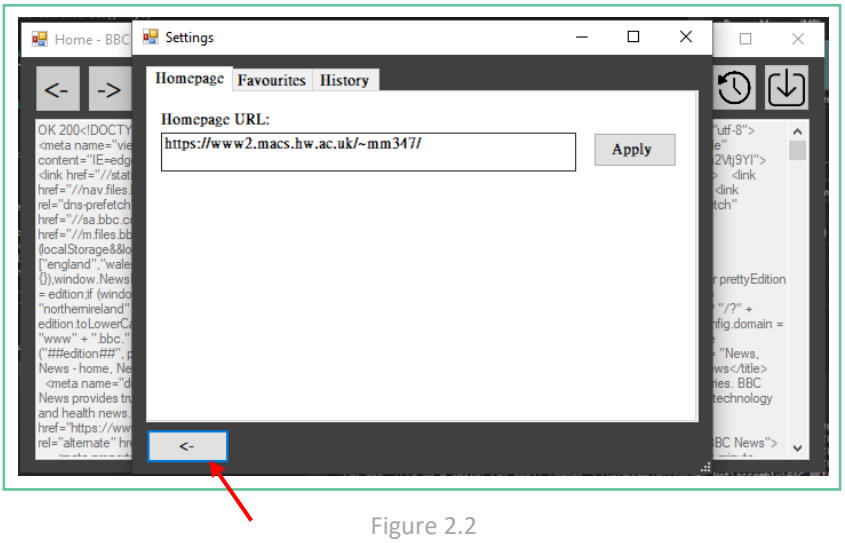

Figure 2.2

To change the homepage URL, simply replace the URL in figure 2.2 in the textbox with your own URL and press the "Apply" button on the right. Once completed, you should get a message box appear alerting you if your homepage was successfully changed or not. To test out the new homepage, press the back button at the bottom of the screen highlighted in figure 2.2 by the red arrow. This will take you back to your main form. Press the icon with the house logo, your homepage should be loaded.

Add/Edit/Access/Remove Favourites

Favourites allows users to bind names to specific URLS and maintain/access them in the browser.

To create a favourite URL, first navigate to the settings form (details on how can be found at the start of the "Change Home Page" section). Once in settings, click on the "Favourites" tab as shown in figure 2.3.

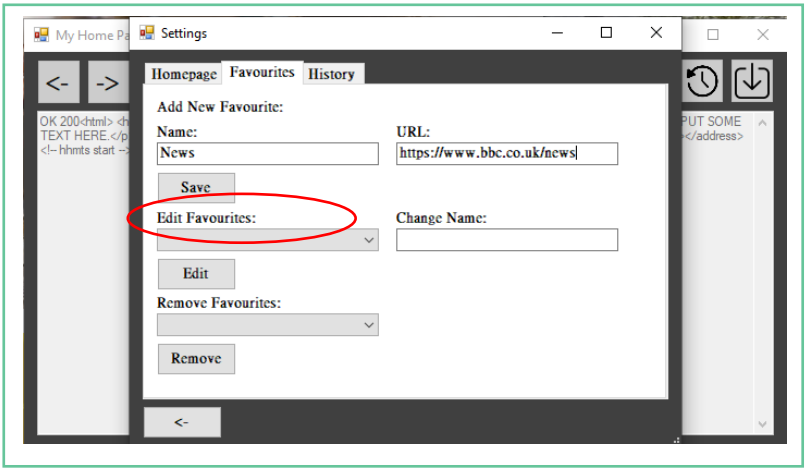

Figure 2.3

Once in favourites, to add a new favourite simply assign a name and a URL in the above text fields under the labels "Name:" and "URL:". Next, press the save button to save your new favourite. A message box will appear stating the status of your favourite being added.

To edit a favourite, select your favourite from the drop-down list highlighted in red in figure 2.3. Enter in a new name in the input box under the "Change Name:" label and press the "Edit" button to confirm. A message alerting you of the status of your changed favourite will appear.

To remove a favourite, select the favourite from the drop-down list as shown in figure 2.4 under "Remove Favourites" and press the remove button. A message box will appear stating if the removal was successful or not.

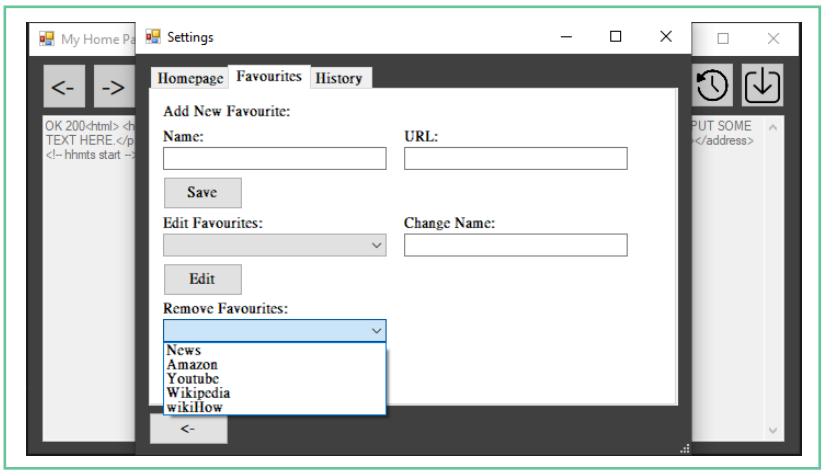

Figure 2.4

To access a favourite, navigate back to the home screen and click on the icon in the top row containing the star. A new form will appear containing a list of favourites. Click the name of the favourite you wish to load then the "Load Page" button in the bottom right-hand corner. An example of the favourites form can be found in the appendix at figure 5.2.

Accessing your history:

To access a URL that a user previously searched, users have two options; they can either user the previous or next page buttons on the main form (these are the buttons with the arrows). Or optionally, they can access their history and freely choose which URL in their history they wish to access. To access the history list, simply press the history icon highlighted in the figure 2.1 by the green arrow.

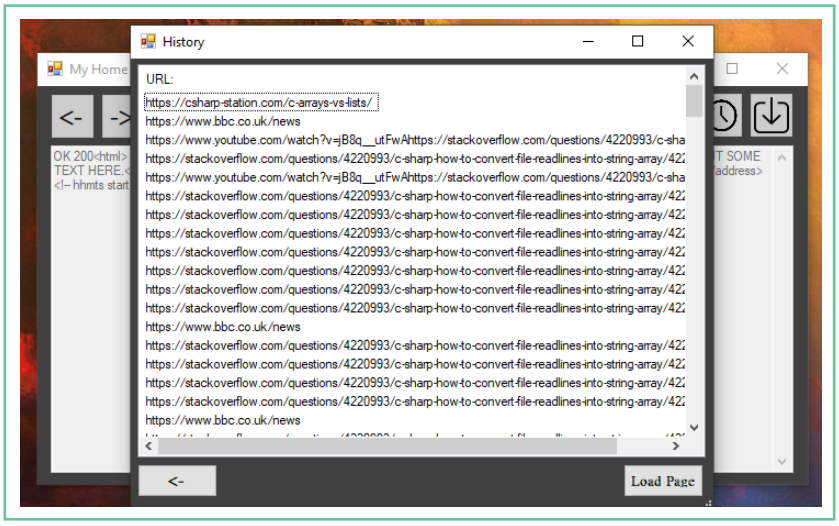

Figure 3.1

After pressing the icon, users will be presented with the form as shown in figure 3.1. Users can then press any link they choose then load the page by pressing the “Load Page” button. Pressing this will close the current history window and display the main form with the users chosen history link being displayed.

URLs from your history can be removed by navigating to the “Settings” form, then the “History” tab.

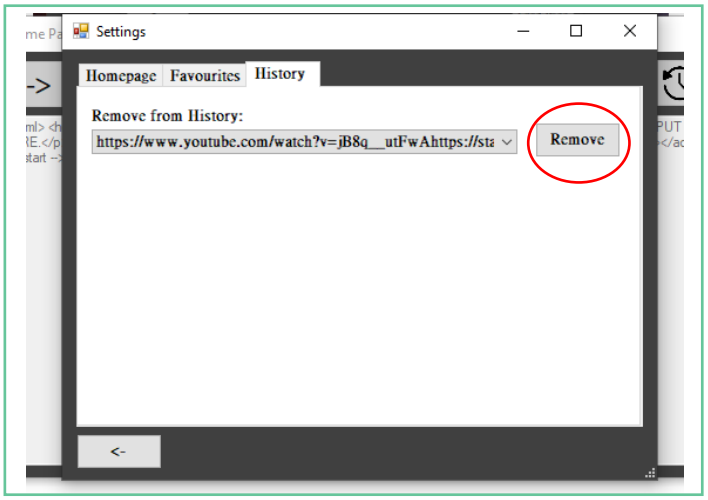

Figure 3.2

Users can then select a URL from the dropdown list and press the “Remove” button as shown in figure 3.2. A message box will appear stating whether it was successful or not and the URL will be removed from the dropdown list.

Initiating bulk downloads:

Bulk download allows the user to perform a HTTP request on URLs in a text file. Bulk download retrieves the status code, number of bytes retrieved from the URL and the URL.

To use the bulk download feature, first the user must press the bulk download icon on the main form. This is the icon in the top right-hand corner, it can be seen in figure 2.1 with a pink arrow being pointed to it.

Once pressed, a pop-up box will appear, asking if the user wishes to use their own file or the default file.

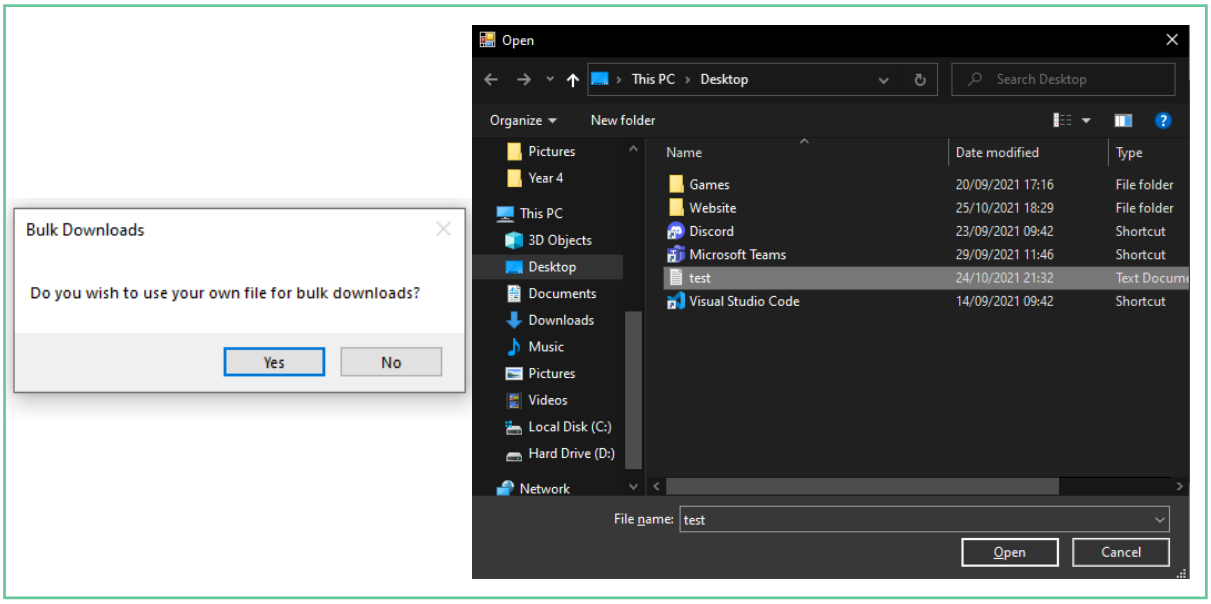

Figure 3.3

Pressing yes will open a window that lets the user find and choose a file to use as shown in figure 3.3. Once the user has pressed "Open", the file picker window will close, and the main form will appear with the output of the bulk downloads function as shown in figure 3.4.

Pressing "No" will load the default bulk.txt file.

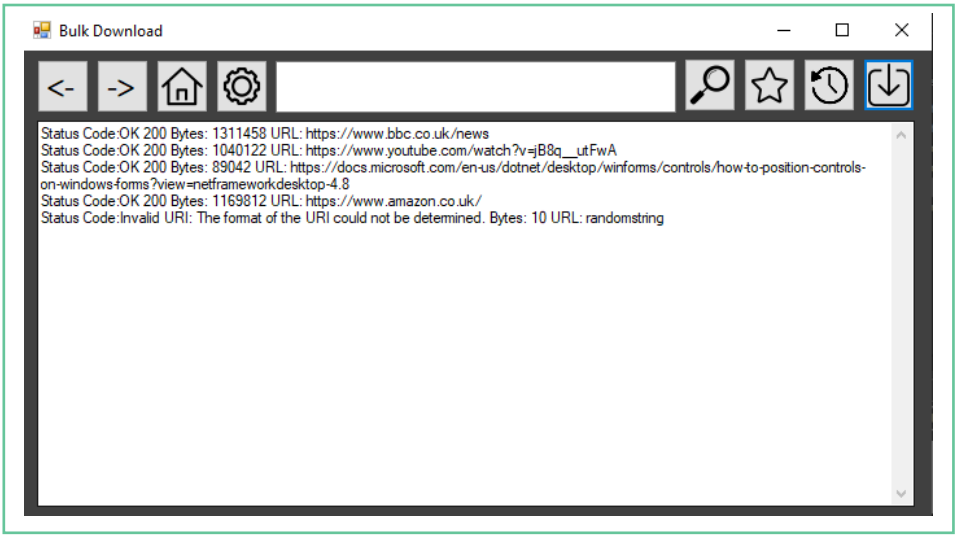

Figure 3.4

Keyboard Shortcuts

Keyboard shortcuts allows the users to navigate to different parts of the program using simple key bindings. The following shortcuts are supported:

- CTRL+H – Opens the homepage
- CTRL+F – Opens the favourites form
- CTRL+S – Opens the settings form

Developer Guide:

The web browser is split up into 5 main classes.

Each class is responsible for a main point of functionality of the program.

Splitting up the class allows for easy modification of functionality that will not have a huge detrimental affect on the rest of the program.

It also allows for simple use of functions from other classes.

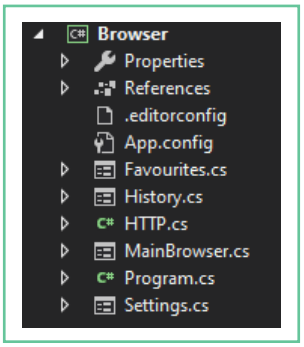

Figure 4.1

Favourites

Favourites is responsible for displaying the favourites in a ListView box in the favourites form. It retrieves the list of favourites from a external txt file. This class communicates closely with the MainBrowser class as it sends the URL of the favourite the user wants to load to the MainBrowser class. It achieves this by using a getter and setter method for a variable containing the URL. The “setURL” function is called when a favourite is selected by a user on the favourites page and a request to load that page has been registered.

This class can be easily modified to include extra details to display on the favourites form such as index numbers for favourites.

```
3 references
public string getURL()
{
    return url;
}

1 reference
public void setURL(string _url)
{
    url = _url;
}
```

Figure 4.2

History

History follows a very similair layout to favourites. It too handles the retrival and displaying of the external history txt file into a ListView box in the history form. This class also communicates in the same manner as favourites with the MainBrowser class using the same setter and getter method format. History also contains the functions for the navigation buttons. The navigation buttons take the length of the history list -1 and keep a running counter of its current position. The global index is then incremented or decremented based on whether a next or previous page is required. The index is reset whenever something new is added to the history. This functionality for page navigation could easily be replaced with something like a linked list that would make keep track of the current url easier.

```
// Return the URL of the previous page in the history list relative to the current page in the history list
1 reference
public string getPreviousPage()
{
    string[] temp = getHistory();
    if (index-1 < 0)
    {
      return temp[index];
    } else
    {
        index = index -1;
        return temp[index];
    }
}

// Return the URL of the next page relative to the current page in the history list
1 reference
public string getNextPage()
{
    string[] temp = getHistory();
    if (index+1 >= temp.Length)
    {
        return temp[index];
    } else
    {
        index++;
        return temp[index];
    }
}
```

Figure 4.3

Main Browser

Main Browser is responsible for performaing the navigation to and from all the other forms of the program. It also contains the fuctionality for bulk downloads as well as the functionality for displaying the HTTP response from sending a HTTP request. The code regarding bulk downloads is 3 functions and these should and could be moved to an external class to make management of it much easier. Main browser also contains a lot of functions contained within buttons themselves, these could be moved to independent functions to allow for better management.

Settings

Settings is responsible for editing the users favourites, history and homepage. This class performs majority of the read/write operations to the external txt files containing the homepage, history list and the favourites. The txt file for homepage just contains the users URL. The txt file for history list contains a list of URLs the user has previously visited seperated by a new line. This makes reading the file into an array much easier. The txt file for the users favourites contains the users specified name and the URL associated with the name, each seperarted with a new line. The format for the favourites file can be seen here. Indexing at an even number provides the name of the favourite whilst indexing at a odd number provides the URL. This sytem could be changed in the future.

```
News
https://www.bbc.co.uk/news
Amazon
https://www.amazon.co.uk/
Youtube
https://www.youtube.com/?hl=en-https://www.youtube.com/
Wikipedia
https://en.wikipedia.org/wiki/Main_Page
wikiHow
https://www.wikihow.com/Main-Page
```

Figure 4.4

HTTP

This class handles the sending of HTTP requests and retrieval of the HTTP response. The requirements for this project listed the program need only handle a 4 status codes and display them. However, the code is designed in such a way to further add support for other status codes.

The default case returns the description of the status code in case any URLs are entered that provide a status code that is not fully supported with a message.

This class can be further modified to utilise other functions of the HttpWebResponse class to add additional functioanlity to the web browser

```
try
{
    HttpWebResponse response = HttpWebRequest.Cr

    switch (response.StatusCode)
    {
        case HttpStatusCode.OK:
            return "OK 200";
        case HttpStatusCode.BadRequest:
            return "400 Bad Request";
        case HttpStatusCode.Forbidden:
            return "403 Forbidden";
        case HttpStatusCode.NotFound:
            return "404 Not Found";
        default:
            return response.StatusDescription;
    }

}
```

Figure 4.5

Testing

This section covers the testing I performed on the web browser.

Section/Feature	Data	Expected Output	Actual Output
HTTP Reponses			
Load URL	https://www.bbc.co.uk/news	Status code “200 OK” and HTML code retrieved	Status code “200 OK” and HTML code is outputted
Load URL	http://status.savanttools.com/?code=404	Status code “404 Not Found” is displayed	Status code “404 Not Found” is outputted
Display			
The title of the web page is at the top	https://www.bbc.co.uk/news	Title should read “Home – BBC News”	Form title reads “Home – BBC News”
Home Page			
Edit and load home page	https://www.amazon.co.uk/	Pressing home button should display HTML code and status code associated with the URL	HTML and status code is displayed
Favourites			
Create a new favourite	Name: “Reddit” URL: “https://www.reddit.com/”	Favourite is available in favourites form and can be loaded	Favourite is available in favourites form and can be loaded
Change the name of a favourite	Previous Name: “Reddit” New Name: “The Reddit”	Name change is visible favourites form. Loading the favourite with the new name still leads to the same URL as before.	Name change is visible favourites form. Loading the favourite with the new name still leads to the same URL as before.
Delete favourite	Name: “The Reddit”	Favourite is removed from favourites form and anywhere else in the application.	Favourite is removed from favourites form and anywhere else in the application.
History			
Navigate to a previous page in the history	Select a URL from the history form and load it	Page is loaded	Page is loaded
Bulk Download			
Initiate bulk download on the default txt file		Output should be the same as in the appendix figure 5.3	Output matches screenshot in appendix figure 5.3
Graphical User Interface			
Navigation between all form’s functions correctly		Forms open and close correctly	Forms open and close correctly
Keyboard shortcuts work	CTRL+H – Homepage CTRL+F – Favourites CTRL+S – Settings	Forms open correctly upon keyboard shortcut presses	Forms open correctly upon keyboard shortcut presses

Conclusion

This section is used to reflect on my overall experience and implementation of this project.

Things I am happy with

I am glad the functionality of the program is complete, and it meets requirements. It has been an enjoyable project to work on. I am very proud of my GUI, I think I have created a very easy to use, clean user interface that allows users to easily navigate between forms. I am happy with how the HTML code from the URL is displayed and my use of ListView boxes to display the history and favourites list in an organized manner.

Things I would have liked to have done differently

I think my classes could be better organized, there is code in classes that should be in other classes. The use of arrays to hold the data from the text files was not the best, I think a list would have been better. I would have liked to create a separate class for the bulk downloads as it currently lives in MainBrowser.

Appendix

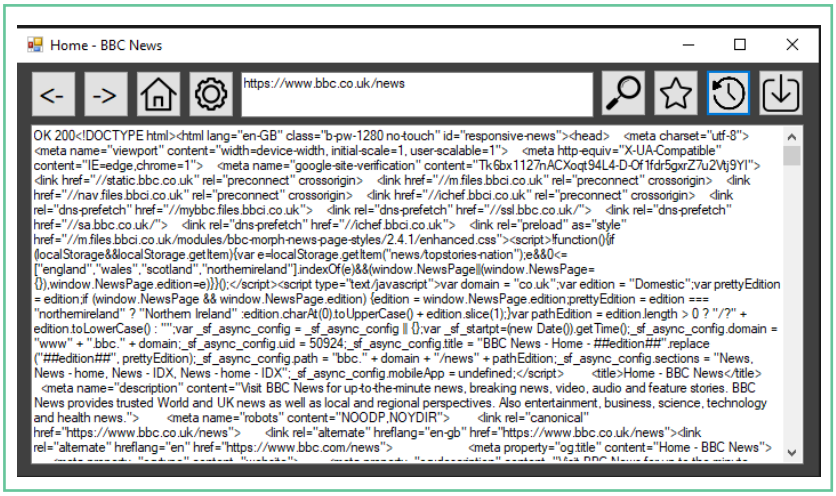

Figure 5.1

Figure 5.2

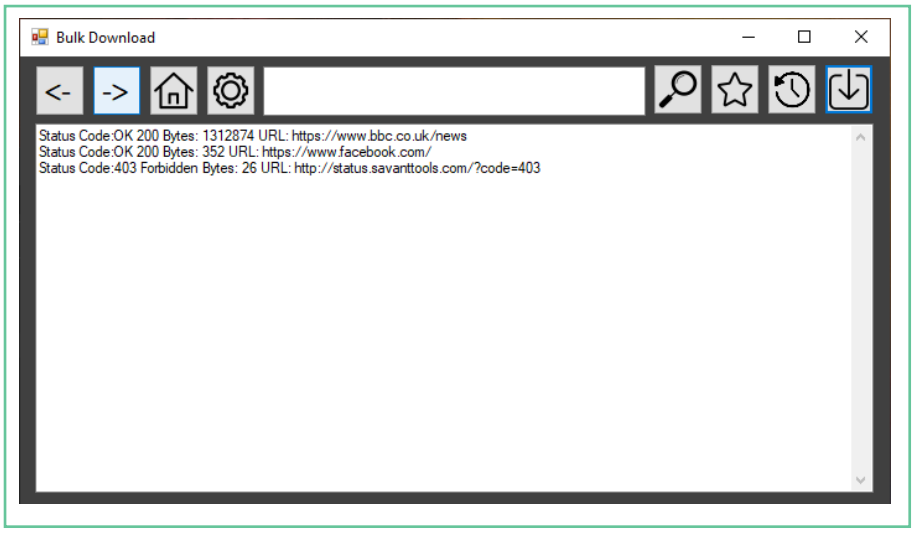

Figure 5.3