Student Declaration of Authorship

Course code and name:	F21SC Industrial Programming
Type of assessment:	**Individual**
Coursework Title:	Developing A Simple Web Browser
Student Name:	Sasha Fahima Suhel
Student ID Number:	H00410394

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature *(type your name):* *Sasha Fathima Suhel*

Date: *24/10/2022*

Copy this page and insert it into your coursework file in front of your title page.
For group assessment each group member must sign a separate form and all forms must be included with the group submission.

Your work will not be marked if a signed copy of this form is not included with your submission.

Assessed Coursework 1 — Developing a Simple Web Browser

Sasha Fathima Suhel, MSc. Data Science

Heriot Watt University Dubai

H00410394

Index

Contents

Introduction

The report specifies the work I have done in building a web browser with the help of C# Language. I will be explaining through the design structures I have used along with the explanation on my code development functionality. Attempting to develop my own Web Browser without the use of the Web Browser class has helped me push my knowledge in C# and object-oriented languages even further.

The aim of my project for me was to develop proficiency in advanced programming concepts, and understand functional programming paradigms, and to apply these programming skills to a concrete application of moderate size. Using different data structures for different functions to have better efficient outcomes as desired. First it's important to understand which symbol one would encounter while inspecting an URL.

Character used in a URL	Characters that are not used in a URL
0 1 2 3 4 5 6 7 8 9 a b c d e f g h I j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z **Special characters:** $ – _ . + ! * ‘ ( ) ,	[ ] { } | \ ” % ~ # < >

Based on most common URLs I have taken into consideration to omit the use of **‘#’** through any of the codes.

Specifications

Visual Studio Community 2022 (17.3.6), Windows OS Version (10.0.22000), C# Language version (10.0), .Net SDK 6.0.402, .Net Framework 6.0.10, JetBrains Rider (2022.2.3)

Requirement Checklist

- ✓ Sending HTTP request messages
- ✓ Receiving HTTP response messages

- ✓ Display the HTTP response status code and title of the page on top of the web browser and be able to *refresh* the page with the current URL with the help of the reload button.
- ✓ *Home page* has been set to the Herriot Watt's default home page. The home page is loaded at start-up and is editable through the Settings button.
- ✓ URLs can be saved into a *Favourite List* which can be opened into a new URL to be accessed as well be modified by the user and deleted if required. Favourite's List is loaded during start up.
- ✓ *History List* is loaded at start-up that loads all the previous visited URLs. The list can be accessed as well as deleted individually or cleared completely.
- ✓ *Bulk Download* will show the HTTP status code and byte sizes of the multiple links provide in a text file that can be selected by the user.
- ✓ *Simple user-friendly GUI* has been implemented with some use of shortcut's which are shown in the settings button.
- ✓ An extra feature for *Settings Button* is created for additional features
- ✓ *Modularization* of the entire code into different classes for easy understanding and reading and to increase code reusability

Design Considerations

Class Design

Web Browser is separated into 10 classes. I attempted to implement Single Responsibility Principle where every class holds a specific responsibility to carry out functions. Modularizing the whole application into different classes will be most efficient while fixing errors that would come during first testing of the application. With this I was able to handle non-functional requirements to the class.

Classes I have implemented for Web Browser Application –
<u>Interpreter</u> – class created to help read and write to all the text files, so that each class is incremented to the next line in order. Helps in interpreting every file <u>Address</u> – this class is used for serialization, to split the URL into title name, URL and into the timestamp at which it is being utilized at, which is set as individual properties. <u>Browser</u> – class handles all the browser functions to save items into the designated list for storing

Classes I have implemented for Web Browser Application –
Tab Manager – this class is inherited from TabPage which contains the main class to handle all the HTTP functionalities for the web browser. FavouriteForm – this class open a form to edit the favourite items and allow the user to customize the details HomepageForm – this class will show a form to set a default home page so that it loads the URL during start-up which will be displayed. This default page can be changed by the user as well. FavouriteClass – handles most of the favourite functionalities HistoryClass – handles most of the history functionalities MainForm – main class which connects all the methods and classes to the GUI for front end development Program – this class has the Main method

Data Structures

The main two data structures I have implemented are Dictionary for Favourites and List for History.

Tab Manager

The "*TabManager*" class inherits the "*TabPage*" class. Every time a user sends an HTTP request, the list is accessed to store all the URLs entered from the history list. To go back and forth through the pages, each tab will pull the desired URL from the list to traverse between the links entered. This class is made so that it helps users open their URL onto a new tab by applying indexing, for each list so that it points to the current URL.

History

List is more logical to be used as a data structure for History as it saves every Item in the file in chronological order. It is important that the items are in order of access time. There is no requirement for the items to be unique either since they can be repeated. The order is determined by the timestamp. There is no sorting used as every time a file is saved onto the file it is automatically saved in the right order, so no extra time is necessarily taken during computation.

The method "*WriteToHistorcyFile*" pushes the URL to the "HistoryFile.txt" and then loads them into the History List. The URL is inserted on top of the History list each time the HTTP request is received. The time complexity for this is expected to be O(N) (Linear time complexity) since the URL is inserted into the first position of the List, which is also the same when the URL is deleted at its index.

Favourites

Unlike History, if the URL already exists in dictionary, the item should not be added again to the list. Dictionary is more optimal as it has one key pair value so if the item already exists, a Message Box will open informing the user that the item is already present in the list. If not, then the URL along with the title will be pushed to the list. The Favourite List has additional features like adding a URL from the Settings button or using the shortcut “Ctrl + D” to edit or delete a specific item from the list. One downside of this data structure is that it must load the entire Favourite list file and order it by ascending order which takes up time and have a time complexity of O(N*logN). Since dictionaries have constant time complexity, **O(1)** is what is taken to add, delete or view an URL in the list which is beneficial in this case.

GUI Design

Like many browsers that exist, my application is inspired from famous browsers that I’ve come across such as Microsoft Edge, Mozilla Firefox, Opera and more. Like every browser, there is a text field that is used to enter the URL which will load up the HTML code for the website.

It consists of many similar components like Back button, Forward button, Reload button, Home Button, and many more functionalities. The main Tab panel will pull the HTML code of the website and display the content.

The GUI is very user friendly and intuitive, certain shortcuts such as “Ctrl + D” to add to the Favourites List which is used as a reference to Google Chrome’s interface. The Web Browser will accept the URL entered in the Textbox when the “Enter” button on the keyboard is pressed. The below table describes the functionalities of every button displayed on the Browser Form for better insight.

Browser Title - This is a label that pulls the <title> from the html code into a label used on top of the web browser
Close – Closes the application
Max – Maximizes the application to full screen
Min – Minimizes the application to the taskbar
(>>) – This button will only be enabled if there is an index to go forward to from the history list
(<<) – This button will only be enabled if there is an index to go back to from the history list

Tab Head – Is the main Tab panel that displays all the html content
Reload – Refreshes the current page into the *Tabhead* panel
Favourite – This button helps to show the Favourite List (*Favlist)* which contains all the favourite items added by the user
History – This button will display the History List (*Hislist*) which contains all the previously opened URLs browsed through by the user. The items are ordered based on the time.
Bulk Download – Will load up a txt file that contains multiple URL's and show the response status code along with the byte size of the URL.
+Tab – Adds a new top into the *Tabhead* panel
-Tab – Deletes the selected tab
Home – Displays the Default onto a new tab
Settings – Will open additional Settings *(Settingsmenu)* which shows more options such as clear all history items, clear all favourite items etc
Set Default Page – Sets any URL as a default page that can be customized by the user to load at start-up
Clear History– Clears all the History items in one click
Bookmark Page – Bookmark's the page into the Favourite List
Clear Favourite – Clears all the Favourite items in one click
Context Menu Strip Favourite – When the user right clicks on an item in the Favourite list, a pop up menu for Edit and Delete will show up to prompt the user for the next course of action
Context Menu Strip History – When the user right clicks on an item in the Favourite list, a pop up menu Delete will show up to prompt the user for the next course of action

The Favourite and History button will display the list of items which are being pulled respectively from the FavouriteFile.txt and HistoryFile.txt which are being saved in '.txt' files by their timestamp in specific order.

When the URL from the list is pressed a Message Box opens up prompting the user to open the link onto a new tab or else to close the Message box.

When an item from the list is right clicked by the mouse a menu tool strip will be displayed showing the user two options whether to Edit or Delete the following item. A new Favourite Form will open if the Edit button is pressed from the menu tool strip to customize the way the title and URL is saved onto the list.

Advanced Language Constructs

Event

For Events, I’ve applied it in *“BrowserManager*” class which calls the *“TabManager”* to add URLs to the HistoryFile.txt. Since the *“BrowserManager”* class knows which URL exist both int the Favourites and History List of the “*MainForm”.* Through this the “*BrowserManager*” passes the function to “*MainForm”* class to add URLs to the Favourite and History list when required.

Delegate and Generics

The Class “*Interpreter*” is used to convey how the information would like to be stored in a file to reduce duplicity, I have implemented simple generics to show how the URLs would be stored into the Favourite and History file.

It Is important that each URL is saved into a specific format so it can be pulled for better usage. With the help of *“Address”* class we can easily create delegate that do not depend on the constricted data types, and I am able to store each line into a file with the required measurements.

LINQ

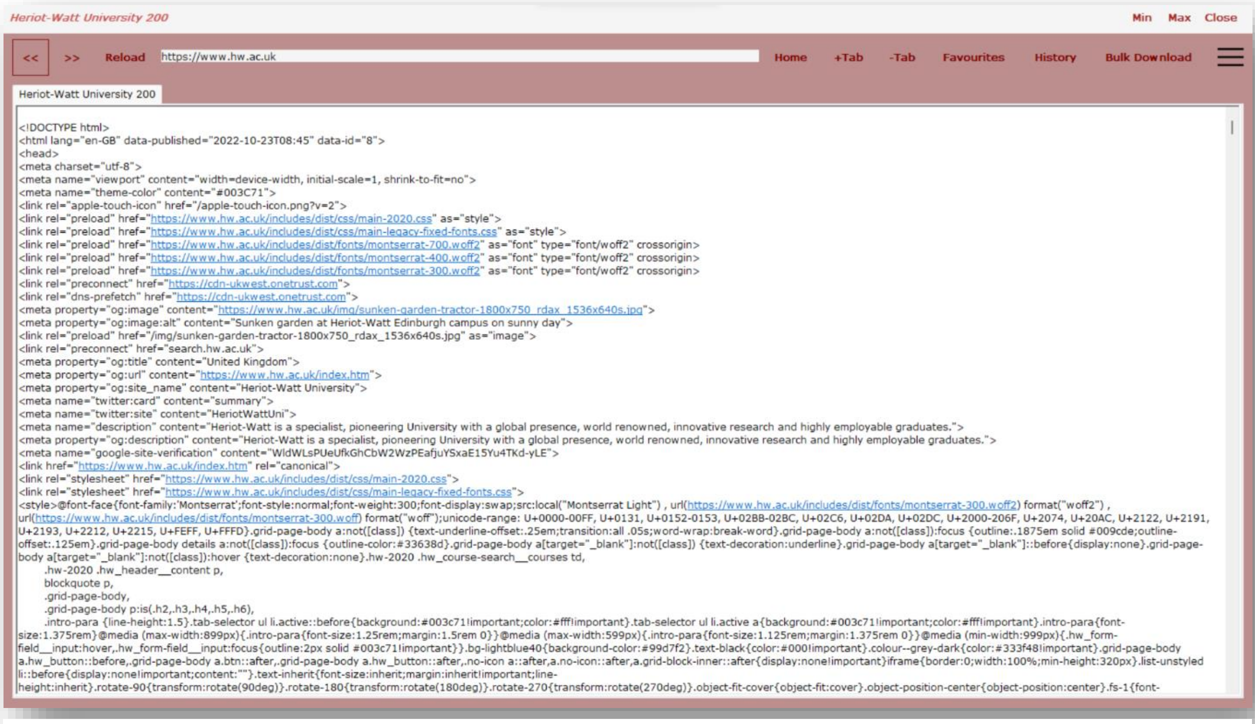
I’ve implemented LINQ to handle the items that are shown in the Favourites List. It acts as a relational data base back to the Dictionary structure of list by matching the indexes with the help of timestamp. Every user can customize the items visible in the list. To prevent changes that would affect the main file, LINQ is most optimal usage of a construct since it only changes the title and URL at display on the list which is then passed to the Favourite List as the updated title and URL. LINQ provides functions to query cached data from all kinds of data sources.

User Guide

I will be going through each functionality of my Browser Application. Each function carries out its own responsibilities. I will list and explain through each of the features that I think make my Web Browser unique and functional.

To start with just like all web browser applications, there is one text field that allows the user to give their preferred URL. The user then must press “Enter” to trigger the HTTP response method in the program. If the URL is valid and sends the HTTP response, then the appropriate HTML of the URL will be displayed on the tab panel of the application as seen in

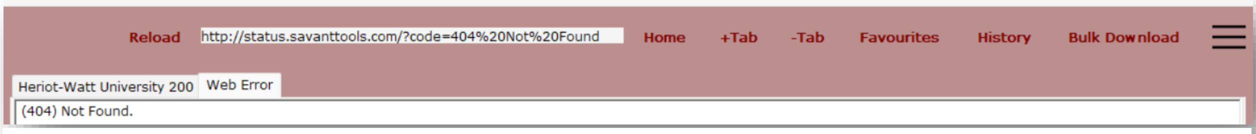
[Figure1.1]. Along with this, the title and the status code of the website is fetched and displayed on the tab bar as well as on top of the web browser as a label to indicate the user which URL they are on.



```
<!DOCTYPE html>
<html lang="en-GB" data-published="2022-10-23T08:45" data-id="8">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="theme-color" content="#003C71">
<link rel="apple-touch-icon" href="/apple-touch-icon.png?v=2">
<link rel="preload" href="https://www.hw.ac.uk/includes/dist/css/main-2020.css" as="style">
<link rel="preload" href="https://www.hw.ac.uk/includes/dist/css/main-legacy-fixed-fonts.css" as="style">
<link rel="preload" href="https://www.hw.ac.uk/includes/dist/fonts/montserrat-700.woff2" as="font" type="font/woff2" crossorigin>
<link rel="preload" href="https://www.hw.ac.uk/includes/dist/fonts/montserrat-400.woff2" as="font" type="font/woff2" crossorigin>
<link rel="preload" href="https://www.hw.ac.uk/includes/dist/fonts/montserrat-300.woff2" as="font" type="font/woff2" crossorigin>
<link rel="preconnect" href="https://cdn-ukwest.onetrust.com">
<link rel="dns-prefetch" href="https://cdn-ukwest.onetrust.com">
<meta property="og:image" content="https://www.hw.ac.uk/img/sunken-garden-tractor-1800x750_rdax_1536x640s.jpg">
<meta property="og:image:alt" content="Sunken garden at Heriot-Watt Edinburgh campus on sunny day">
<link rel="preload" href="/img/sunken-garden-tractor-1800x750_rdax_1536x640s.jpg" as="image">
<link rel="preconnect" href="search.hw.ac.uk">
<meta property="og:title" content="United Kingdom">
<meta property="og:url" content="https://www.hw.ac.uk/index.htm">
<meta property="og:site_name" content="Heriot-Watt University">
<meta name="twitter:card" content="summary">
<meta name="twitter:site" content="HeriotWattUni">
<meta name="description" content="Heriot-Watt is a specialist, pioneering University with a global presence, world renowned, innovative research and highly employable graduates.">
<meta property="og:description" content="Heriot-Watt is a specialist, pioneering University with a global presence, world renowned, innovative research and highly employable graduates.">
<meta name="google-site-verification" content="WldWLsPUeUfkGhCbW2WzPEafjuYSxaE15Yu4TKd-yLE">
<link href="https://www.hw.ac.uk/index.htm" rel="canonical">
<link rel="stylesheet" href="https://www.hw.ac.uk/includes/dist/css/main-2020.css">
<link rel="stylesheet" href="https://www.hw.ac.uk/includes/dist/css/main-legacy-fixed-fonts.css">
<style>@font-face{font-family:'Montserrat';font-style:normal;font-weight:300;font-display:swap;src:local("Montserrat Light") , url(https://www.hw.ac.uk/includes/dist/fonts/montserrat-300.woff2) format("woff2") ,
url(https://www.hw.ac.uk/includes/dist/fonts/montserrat-300.woff) format("woff");unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02BB-02BC, U+02C6, U+02DA, U+02DC, U+2000-206F, U+2074, U+20AC, U+2122, U+2191,
U+2193, U+2212, U+2215, U+FEFF, U+FFFD}.grid-page-body a:not([class]) {text-underline-offset:.25em;transition:all .05s;word-wrap:break-word}.grid-page-body a:not([class]):focus {outline:.1875em solid #009cde;outline-
offset:.125em}.grid-page-body details a:not([class]):focus {outline-color:#33638d}.grid-page-body a[target="_blank"]:not([class]) {text-decoration:underline}.grid-page-body a[target="_blank"]::before{display:none}.grid-page-
body a[target="_blank"]:not([class]):hover {text-decoration:none}.hw-2020 .hw_course-search__courses td,
      .hw-2020 .hw_header__content p,
      blockquote p,
      .grid-page-body,
      .grid-page-body p:is(.h2,.h3,.h4,.h5,.h6),
      .intro-para {line-height:1.5}.tab-selector ul li.active::before{background:#003c71!important;color:#fff!important}.tab-selector ul li.active a{background:#003c71!important;color:#fff!important}.intro-para{font-
size:1.375rem}@media (max-width:899px){.intro-para{font-size:1.25rem;margin:1.5rem 0}}@media (max-width:599px){.intro-para{font-size:1.125rem;margin:1.375rem 0}}@media (min-width:999px){.hw_form-
field__input:hover,.hw_form-field__input:focus{outline:2px solid #003c71!important}}.bg-lightblue40{background-color:#99d7f2}.text-black{color:#000!important}.colour--grey-dark{color:#333f48!important}.grid-page-body
a.hw_button::before,.grid-page-body a.btn::after,.grid-page-body a.hw_button::after,.no-icon a::after,a.no-icon::after,a.grid-block-inner::after{display:none!important}iframe{border:0;width:100%;min-height:320px}.list-unstyled
li::before{display:none!important;content:""}.text-inherit{font-size:inherit;margin:inherit!important;line-
height:inherit}.rotate-90{transform:rotate(90deg)}.rotate-180{transform:rotate(180deg)}.rotate-270{transform:rotate(270deg)}.object-fit-cover{object-fit:cover}.object-position-center{object-position:center}.fs-1{font-
```

Figure 1.1 Web browser at start-up

Exceptional cases where the URL is typed incorrectly by the user, a message will be shown on the tab panel displaying the error. If the HTTP response status of the URL is not (200 ok) then the respective error status code message will be displayed as shown in [Figure1.2].


Reload
http://status.savanttools.com/?code=404%20Not%20Found
Home
+Tab
-Tab
Favourites
History
Bulk Download
Heriot-Watt University 200
Web Error

(404) Not Found.

Figure 1.2 Web Error caused by incorrect URL

The user can traverse back and forth between URLs by clicking (>>) (<<) buttons. The back button (<<) will go back to a previous index stored in the history for that specific individual tab. Same as for the forward button (>>) which will allow the user to ahead to the

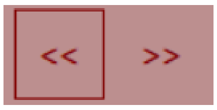


Figure 1.3

next index stored in the history list. These buttons will only be enabled if there are the indexes that can be accessed to by the *"TabPage"* class. This class checks back with the file and conveys the "MainForm" class for the buttons to be visible or not, based on the index [Figure1.3].

The home page is loaded at start-up which is customizable by the user which can be accessed through the settings button for additional features.

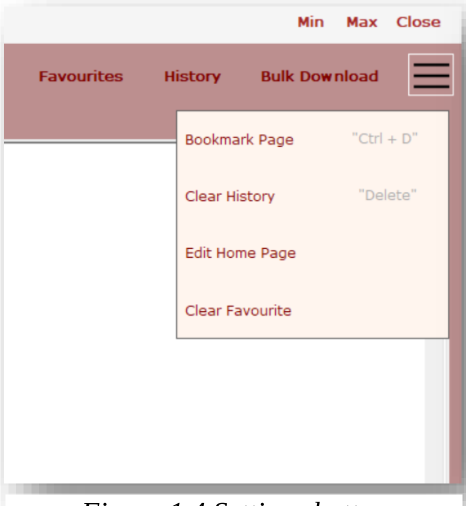
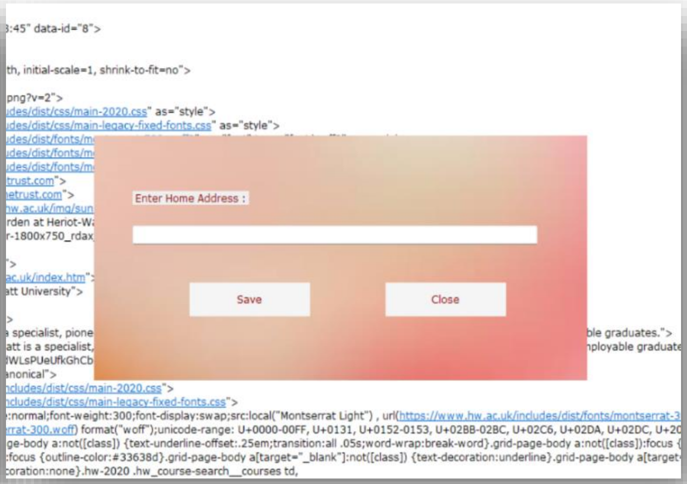


Figure 1.4 Settings button



```
3:45" data-id="8">

th, initial-scale=1, shrink-to-fit=no">

png?v=2">
ıdes/dist/css/main-2020.css" as="style">
ıdes/dist/css/main-legacy-fixed-fonts.css" as="style">
ıdes/dist/fonts/m
ıdes/dist/fonts/m
ıdes/dist/fonts/m
trust.com">
etrust.com">
hw.ac.uk/img/sun
rden at Heriot-W
r-1800x750_rdax

">
ac.uk/index.htm"
att University">

>
a specialist, pione                                                          ble graduates.">
att is a specialist,                                                         nployable graduate
WLsPUeUfkGhCb
anonical">
ncludes/dist/css/main-2020.css">
ncludes/dist/css/main-legacy-fixed-fonts.css">
:normal;font-weight:300;font-display:swap;src:local("Montserrat Light") , url(https://www.hw.ac.uk/includes/dist/fonts/montserrat-3
rrat-300.woff) format("woff");unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02BB-02BC, U+02C6, U+02DA, U+02DC, U+20
ge-body a:not([class]) {text-underline-offset:.25em;transition:all .05s;word-wrap:break-word}.grid-page-body a:not([class]):focus {
:focus {outline-color:#33638d}.grid-page-body a[target="_blank"]:not([class]) {text-decoration:underline}.grid-page-body a[target
coration:none}.hw-2020 .hw_course-search__courses td,
```

Figure 1.5 Default page form

The Default page form allows the user to add their preferred home page on the textbox and then proceed to either 'Save' or 'Cancel' the following option [Figure 1.4, Figure 1.5]. To access the home page again the user can press the home button which will show the set home page onto a new tab.

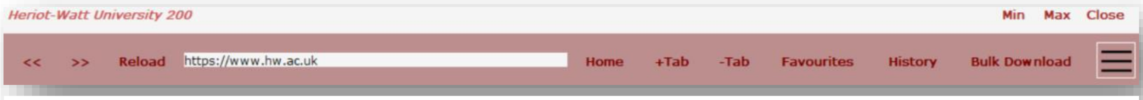
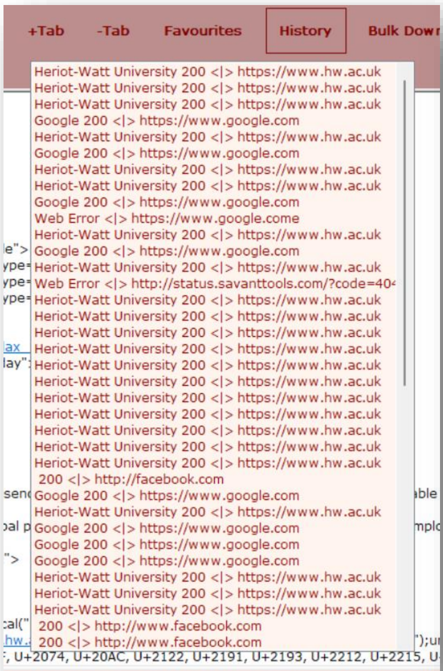


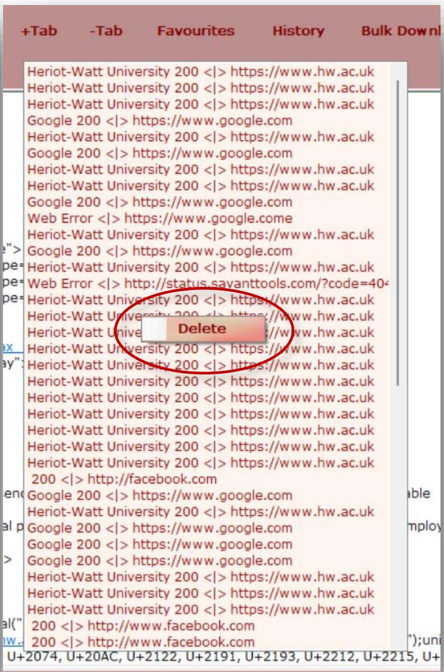
Figure 1.6 Browser functions

The (+Tab) allows the user to add an additional tab which will open a blank page. While the (-Tab) will delete the selected tab from the browser. These function similarity to Google Chrome and other browser applications [Figure 1.6]. The user also has an option to maximize, minimize or close the entire application in one go. All the URLs that were passed through any of the tabs will be displayed on the History list.



```
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Web Error <|> https://www.google.come
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Web Error <|> http://status.savanttools.com/?code=40
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
 200 <|> http://facebook.com
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Google 200 <|> https://www.google.com
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
 200 <|> http://www.facebook.com
 200 <|> http://www.facebook.com
, U+2074, U+20AC, U+2122, U+2191, U+2193, U+2212, U+2215, U
```

Figure 1.7 History List



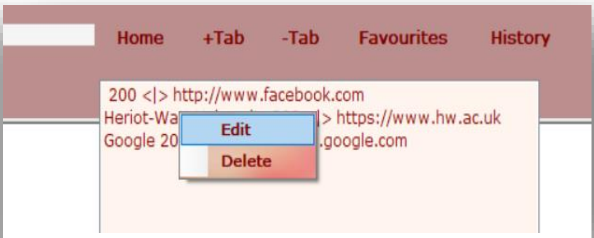
```
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Web Error <|> https://www.google.come
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Web Error <|> http://status.savanttools.com/?code=40
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt U                          www.hw.ac.uk
Heriot-Watt Unive               //www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
 200 <|> http://facebook.com
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com
Google 200 <|> https://www.google.com
Google 200 <|> https://www.google.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Heriot-Watt University 200 <|> https://www.hw.ac.uk
 200 <|> http://www.facebook.com
 200 <|> http://www.facebook.com
U+2074, U+20AC, U+2122, U+2191, U+2193, U+2212, U+2215, U+
```

Figure 1.8 Edit/Delete items

When the History button is clicked, the History List that contains all the content is displayed [Figure 1.7]. Specific history items can be deleted by right clicking the item with the mouse. But what happens is that when each individual item is deleted by the user, it's highly time consuming as many items could be present in the History List.

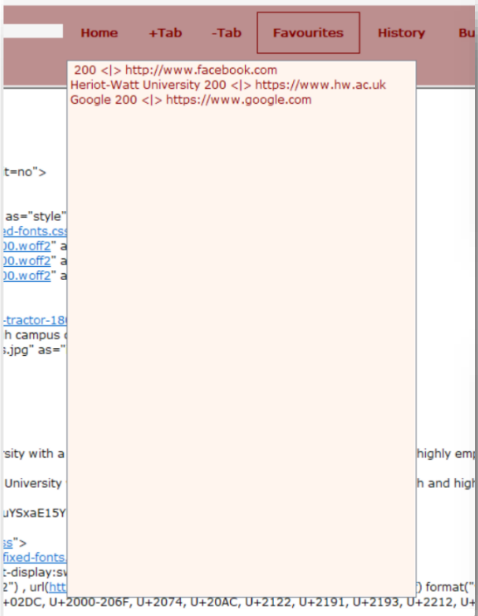
To save time from the user's end the "Clear All History" button can be clicked which is available in the settings button [Figure 1.4]. The button will empty out all the URLs from "HistoryFile.txt". As well as a shortcut has been added which is shows as a grey label beside the button that can be pressed at any time to carry out the function.

Any URL from History List can be opened into a new tab. When the user clicks on an URL, a Message box will be prompted requesting for the next course of action to whether open the link onto a new tab or to cancel the function.



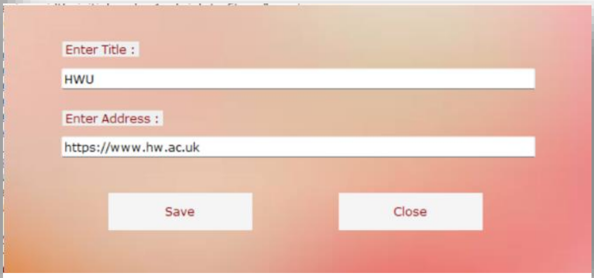
200 <|> http://www.facebook.com
Heriot-Wa... |> https://www.hw.ac.uk
Google 20... .google.com

Figure 2.0 Edit/Delete item



200 <|> http://www.facebook.com
Heriot-Watt University 200 <|> https://www.hw.ac.uk
Google 200 <|> https://www.google.com

Figure 2.1 Favourites List



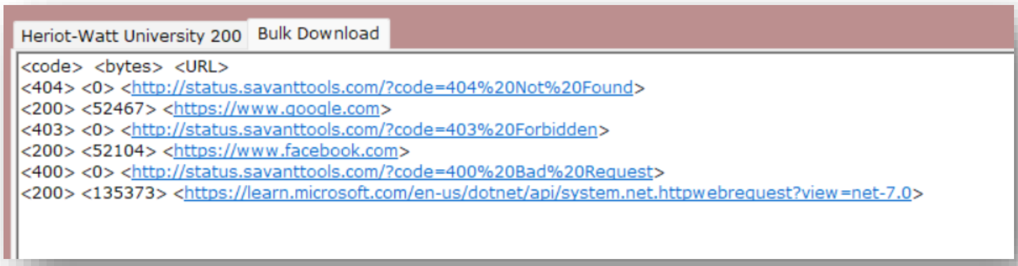
Enter Title : HWU

Enter Address : https://www.hw.ac.uk

Figure 1.9 Favourites Edit Form

When the Favourites button is clicked, the Favourites List that contains all the content is displayed [Figure 2.1]. Same as the history most of the front-end usability is the same as History button which allows the user to add URLs to the Favourites List by Clicking on the "Bookmark Page" option from the Settings button [Figure 1.9]. The user can also right-click on a selected item to either "Edit" or "Delete" the item from the list [Figure 2.0]. This method also contains the same functionality as history by using the function "Clear All Favourites" which deletes all the saved items from the list.

The last functionality of the Web Browser is the Bulk Download which will open multiple URLs onto a new tab and display the status code, byte size of the HTML of the URL along with the URL typed in the text field. This is done by an Open File Dialogue where the user can select the input file which must be a '.txt' file.



```
<code> <bytes> <URL>
<404> <0> <http://status.savanttools.com/?code=404%20Not%20Found>
<200> <52467> <https://www.google.com>
<403> <0> <http://status.savanttools.com/?code=403%20Forbidden>
<200> <52104> <https://www.facebook.com>
<400> <0> <http://status.savanttools.com/?code=400%20Bad%20Request>
<200> <135373> <https://learn.microsoft.com/en-us/dotnet/api/system.net.httpwebrequest?view=net-7.0>
```

Figure 2.2 Bulk Download

If the file inserted is not in the right format, a message box will pop up informing that it will not accept files which is not in .txt format. The file should contain URLs in a downwards list format for the function to work.

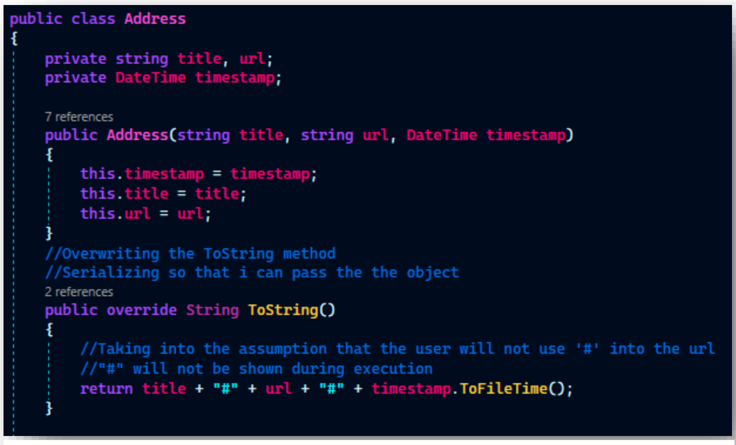
Developer Guide

The class Address contains 3 different main properties.

- The Title, URL and the Timestamp

It let me enable this class to be used in a public way of getting and setting values, while hiding implementation [Figure2.2].

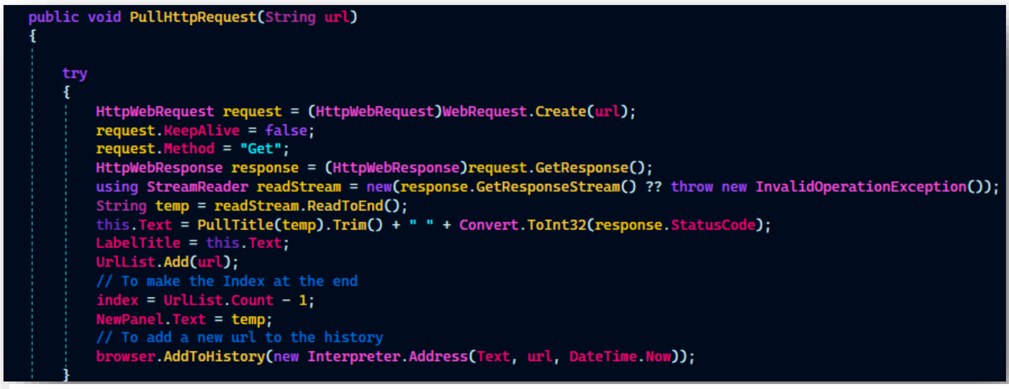
By using serialization, I was able to save the URLs in such a way that I could reuse the title, URL and timestamp separately for other functions implemented in the code.



```
public class Address
{
    private string title, url;
    private DateTime timestamp;

    7 references
    public Address(string title, string url, DateTime timestamp)
    {
        this.timestamp = timestamp;
        this.title = title;
        this.url = url;
    }
    //Overwriting the ToString method
    //Serializing so that i can pass the the object
    2 references
    public override String ToString()
    {
        //Taking into the assumption that the user will not use '#' into the url
        //"#" will not be shown during execution
        return title + "#" + url + "#" + timestamp.ToFileTime();
    }
```

Figure 2.2 Address Class

The "*PullHttpRequest"* method which is in the Tab Manager class is the main class that fetches the HTTP requests. It is necessary to use try catch statements as exceptions/errors could take place whether it's from the user or browser's side. After the URL is entered into the text field this method is triggered which passes the HTTP request for the specified URL. If an incorrect URL is entered the code will proceed to the exceptions to prevent loss of control. "Invalid URL" will be shown in the tab title if an URL with syntax error is typed in by the user [Figure 2.3].



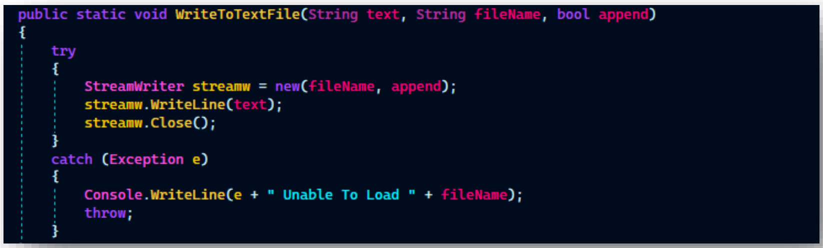
```
public void PullHttpRequest(String url)
{

    try
    {
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
        request.KeepAlive = false;
        request.Method = "Get";
        HttpWebResponse response = (HttpWebResponse)request.GetResponse();
        using StreamReader readStream = new(response.GetResponseStream() ?? throw new InvalidOperationException());
        String temp = readStream.ReadToEnd();
        this.Text = PullTitle(temp).Trim() + " " + Convert.ToInt32(response.StatusCode);
        LabelTitle = this.Text;
        UrlList.Add(url);
        // To make the Index at the end
        index = UrlList.Count - 1;
        NewPanel.Text = temp;
        // To add a new url to the history
        browser.AddToHistory(new Interpreter.Address(Text, url, DateTime.Now));
    }
```

Figure 2.3 HTTP response method

This method also pulls the title into the Label on top of the browser and trim the excess characters that are not required to be displayed on the label and the tab title.

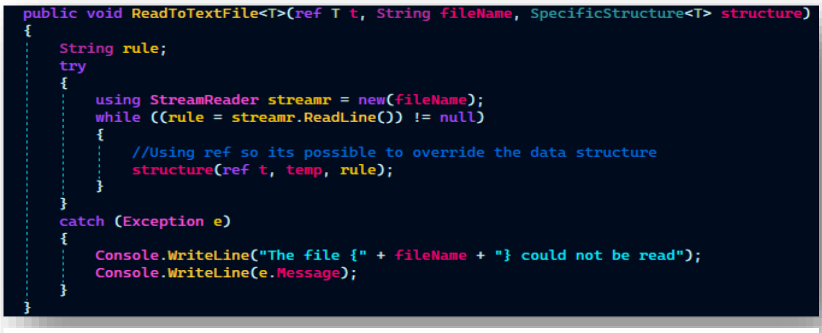
"Interpreter" class handles pushing and pulling the URLs from the HistoryFile and FavouriteFile in the required format.

The methods [Figure 2.4, 2.5] contain StreamWriter and StreamReader that help read individual lines from the text files. Exceptions are created in case the user gives their own file to read from with broken/ incorrect URLs.



```
public static void WriteToTextFile(String text, String fileName, bool append)
{
    try
    {
        StreamWriter streamw = new(fileName, append);
        streamw.WriteLine(text);
        streamw.Close();
    }
    catch (Exception e)
    {
        Console.WriteLine(e + " Unable To Load " + fileName);
        throw;
    }
```

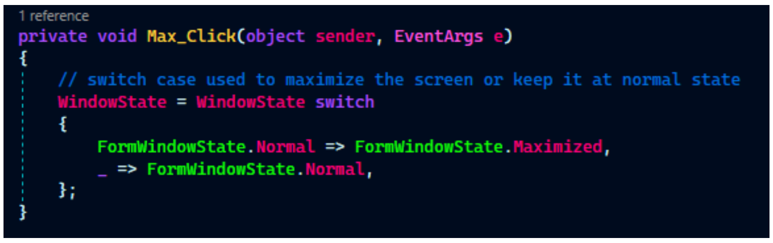
Figure 2.4 Writes the URLs to the respective file



```
public void ReadToTextFile<T>(ref T t, String fileName, SpecificStructure<T> structure)
{
    String rule;
    try
    {
        using StreamReader streamr = new(fileName);
        while ((rule = streamr.ReadLine()) != null)
        {
            //Using ref so its possible to override the data structure
            structure(ref t, temp, rule);
        }
    }
    catch (Exception e)
    {
        Console.WriteLine("The file {" + fileName + "} could not be read");
        Console.WriteLine(e.Message);
    }
}
```

Figure 2.5 Writes the URLs to the respective file

Simple usage of the Discard is utilized wherever a variable is being deleted. The discards are a way to intentionally ignore local variables which are irrelevant for the purposes of the code being produced. Multiple methods contain discard in attempt to improve runtime speed and computation [Figure 2.6].



```
1 reference
private void Max_Click(object sender, EventArgs e)
{
    // switch case used to maximize the screen or keep it at normal state
    WindowState = WindowState switch
    {
        FormWindowState.Normal => FormWindowState.Maximized,
        _ => FormWindowState.Normal,
    };
}
```

Figure 2.6 Usage of discard " _ "

Switch statements are significantly faster than an if-else ladder if there are many nested if-else's involved. This is due to the creation of a jump table for switch during compilation.

Testing

Bugs:

- The History List had glitched when I tried to open a link from the list because it accessed multiple methods at the same time. I have used a Message box to help overcome this issue.
- The list for Favourite, History and Settings button kept overlapping each other when opened. The issue got fixed by toggling the visibility of each of the buttons when one is visible.
- Whenever a new tab was opened, the title of the application should display "Blank page," but it conflicted with the previous tab's title. For this I removed the Label function when a new Blank Tab is added.
- The speed at which the Bulk download fetches the URL from a text file is significantly slower because of the nested if else statements.
- Multiple try-catch constructs have been implemented as errors were caused at various scenarios.

Apart from the above-mentioned bugs, everything works perfectly.

Reflections on programming language and implementation

Reflection is particularly important and proven to be an asset for programming languages, in this case for OOP, by increasing its adaptability and extensibility of programs. Many famous and well-established Web Applications already exist, but here I attempt to recreate my own Web Browser which shows the HTML code of the URLs. Concepts like serialization and LINQ was very insightful as the usage of this at an industrial level appears very beneficial and through this application. I was able to have a deep understanding in such data structures. It is also important to be able to compute the time and space complexities of the data structures utilized to be able to create an efficient application.

I’ve learnt over the development of this application that OOP languages such as C# contains both data and functions, which I smore integrated and much faster than other process-oriented programming languages especially in this instance to create a web application. Through different unit testing’s I was able to have a better execution time than my initial testing which was comparatively slow.

Conclusions

I have learned a lot while developing this application. Working with C# has helped me discover new OOPS concepts. I first thought that it would be like Java, but after working on the application, I can conclude that C# has its own unique features. My foundations in C# have grown stronger. If I were to make any further improvements in my application, I would improve the efficiency and reduce the execution time of the code. Also, I would modularize the entire code into MVVM (Model — View — ViewModel) architecture to separate the code logic and user interface controls.

References

- https://www.winsocketdotnetworkprogramming.com/httpgetrequestdotnetworkprogramming10.html
- https://www.gnu.org/software/dotgnu/pnetlib-doc/System/Net/HttpWebRequest.html#HttpWebRequest.AddRange%28System.String%2C%20int%29%20Method
- https://www.c-sharpcorner.com/UploadFile/72d20e/concept-of-linq-with-C-Sharp/
- https://link.springer.com/content/pdf/bbm:978-1-4302-0709-2/1.pdf
- https://en.wikibooks.org/wiki/C_Sharp_Programming/Delegates_and_Events#:~:text=A%20delegate%20is%20a%20way,occurred%2C%20if%20you%20wish%20so