Assessed Coursework 1 — Developing a Simple Web Browser

1 Overview

The aim of this coursework is to develop a simple web browser in C#.

This is an **individual project**, and you will have to submit your own, original solution for this coursework specification, consisting of a report, the source code and an executable.

The **learning objective** of this coursework is for students to develop proficiency in advanced programming concepts, stemming from both object-oriented and functional programming paradigms, and to apply these programming skills to a concrete application of moderate size. Design choices regarding languages, tools, and libraries chosen for the implementation need to be justified in the accompanying report.

This coursework will develop personal abilities in autonomous problem analysis, the usage of a modern object-oriented language for system programming, and deepen the understanding of integrating components on a Windows system.

2 Lab Environment

Lab environment: For this coursework you should use Visual Studio C# 2019, which will be the reference platform for testing. You can obtain a full version through Microsoft's Imagine Programme. To do this, you need to contact `ithelp@hw.ac.uk` to obtain a student account on their web site. For the GUI component under Visual Studio, the Windows Forms libraries are recommended.

Alternatively, you can use the Mono infrastructure for .NET, which runs under Linux (as well as Windows and MacOS), and use the MonoDevelop IDE or the Visual Studio Code IDE for developing your application. You need to be aware that GUI development will differ in this environment, and that the libraries available for this platform are likely to be a subset of the libraries for Visual Studio. For the GUI component under Mono, the Gtk# libraries are recommended.

If you develop the software in a different environment, you must make sure that the code is compatible with the above software. Beware of incompatibilities due to different versions of the IDE or compiler that you use. Make sure that you can run the final application on a Windows lab machine.

For technical HOWTOs about accessing software of relevance for this course, see this technical HOWTOs page for the course or the resources section on the Vision page.

The submission of the final software must be in form of both *complete sources* as well as a *stand-alone executable* that can be installed and executed on any Windows 10 (with .NET Core and Visual Studio as IDE) or on an Ubunutu 20.04 (with .NET Core and MonoDevelop as IDE) based system. If external tools are used, they need to be included in this executable. Any dependencies on tools or external software must be clearly marked in the report.

3 Developing a Simple Web Browser

The web browser must provide the following functionality:

- **Sending HTTP request messages** for URLs typed by the user.
- **Receiving HTTP response messages** and display the contents of the messages on the interface. Note that you are only required to display the HTML code returned to the web browser from the web server

(HTML parsing and graphical display are **not** required). In addition to the 200 (OK) HTTP response status code, the following HTTP response status error codes and the display of their corresponding error messages should be supported:

- 400 Bad Request
- 403 Forbidden
- 404 Not Found

Hint: To test these error codes, look up web pages like these: `https://savanttools.com/test-http-status-codes` or `httpstat.us/404`

- **Display** the HTTP response status code and (if applicable) the title of the web page at the top of the browser's main window. **Reload the current page** by sending another HTTP request for the current web page, and display the contents of the page together with the title and the HTTP response status code as specified above.

- **Home page**: The user should be able to *create* and *edit* a Home page URL. The Home page URL should be *loaded* on the browser's start up, and it should be initialised with your university home page.

- **Favourites**: The user should be able to *add* a URL for a web page requested to a list of favourite web pages. The user should also be able to *associate a name* with each favourite URL. Support for favourite items *modification and deletion* is required. The user should be able to request a favourite web page by *clicking* its name on the Favourites list. On the browser's start up, the Favourites list should be *loaded* to the browser.

- **History**: The browser should maintain a *list of URLs*, corresponding to the web pages requested by the user. The user should be able to *navigate* to previous and next pages, and jump to a page by *clicking* on the links in the History list. On the browser's start up, the History list should be *loaded* to the browser.

- **Bulk Download:** The application should provide a bulk download facility as follows. The user should be able specify a file name, default setting `bulk.txt`, containing URLs (exactly one per line). When initiating bulk download, the application should retrieve each of the pages listed in the file, and display the results as a list of lines as follows:

  ```
  <code> <bytes> <URL>
  ```

  where `<code>` should be the response status code from the page retrieval (as in "Receiving HTTP response messages" above), `<bytes>` should be the number of bytes retrieved from this URL, and `<URL>` should be the URL as specified in the bulk download file. Only these results should be shown in the textarea normally used for displaying the HTML code. One line should be shown for every entry in the bulk download file. **No HTML** contents should be displayed for this functionality.

- **Graphical User Interface:** A simple GUI should be provided to perform the operations discussed above. The GUI should be implemented using either the Windows Forms (Windows) or the Gtk# (Linux) libraries. The GUI for the web browser should support the following:

  - Using the GUI, the user should be able to perform the operations discussed above.
  - Make use of menus (with appropriate shortcut keys) as well as buttons to increase accessibility.

Library Usage

You should identify the suitable library functions out of the set of libraries installed with the .NET Core platform. The report must clearly motivate the usage of the functions that you have chosen. **You must not use the C# WebBrowser class in this implementation, but perform the required HTTP-level communication directly from within your code.** The code must clearly identify the HTTP-level client-server communication, and must explicitly manage Home page, Favourite, and History Lists. *Optionally,* you may add functionality to render a web page, but there must be an option to disable this functionality and to show only the raw HTML that has been retrieved.

4 Submission

You must submit the **complete project files, containing the source code, a stand-alone executable, and the report** (in `.pdf` format) as one `.zip` file no later than **3:30PM (Edinburgh) / 11:59PM (Dubai) on Tuesday 26th of October 2021**. Additionally, a **screencast or video** of running the application, with an explanation as voice-over audio needs to be submitted to Canvas. **This is mandatory, and without the screencast or video the submission is incomplete and may be marked as 0 points.** The standard penalty of -30% of the maximum available mark applies to late submissions. No submissions will be accepted after 5 working days beyond the submission deadline. Submission must be through *Canvas,* submitting all of the above files in one `.zip` file. **This coursework is worth 50% of the module's mark.**

You are marked for your application, the structure, code and comments used, your testing, your report and the screencast/video of demonstrating the running of your application. The marking scheme for this project is attached.

5 Report Format

The report should have between 8–12 pages and use the following format (if you need space for additional screenshots, put them into an appendix, not counting against the page limit, but don't rely on the screenshots in your discussion):

1. **Introduction:** State the purpose of the report, your remit and any assumptions you have made during the development process.
2. **Requirements' checklist:** Here you should clearly show which requirements you have delivered and which you haven't.
3. **Design Considerations:** Here you should clearly state basic design decisions you have made in developing your code, covering class design, choice of data structures, GUI design, usage of advanced language constructs, other performance-relevant choices etc.
4. **User Guide:** Use screen shots of the running application along with text descriptions to help you describe how to operate the application.
5. **Developer Guide:** Describe your application design and main areas of code in order to help another developer understand your work and how they might develop it. You may find it useful to supplement the text with code fragments.
6. **Testing:** Show the results for testing all cases and prove that the outputs are as expected. Preferably, use unit testing to test core functionality of the implementation. If certain conditions cause erroneous results or the application to crash then report these honestly.

7. **Reflections on programming language and implementation:** Based on your experience in implementing this application, reflect which language features and technologies have been most helpful, identify limitations of your application and suggest ways how to overcome this limitations. Also reflect on the usability of the (kind of) language (either system or scripting language) for this application domain, and on its wider applicability.

8. **Conclusions:** Reflect on what you are most proud of in the application and what you'd have liked to have done differently.

9. A final section should contain the main references used in this report and in the implementation.

Marking Scheme

Criteria	**Marks**
Meeting system requirements	50
Report Quality Content and communication (Design Considerations, Reflections, User guide & Developers guide)	20
The Application Code quality, secure coding, sufficient comments, sensible variable/object names, good code/class/method design, use of advanced lang. features.	15
Initiative, innovation, **professional conduct**, creativity and efforts above & beyond the call of duty	10
Screencast/Video of running the application with explanation as voice-over audio.	5
Total marks	100

Plagiarism:
This project is assessed as an **individual project**. You must work on your own and not share work with other students on this course. You can discuss general technical issues related to this work with other students, however, you must not share concrete pieces of code or text. Readings, web sources and any other material that you use from sources other than lecture material must be appropriately acknowledged and referenced. Plagiarism in any part of your report will result in referral to the disciplinary committee, which may lead to you losing all marks for this coursework and may have further implications on your degree. For details see this link