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: The software needed for this lab is installed on all machines in the Windows Lab (EM 2.45). If you want to use your own laptop, you will need to install several packages. This section describes which packages you need and how to test the installations.

For this coursework you should use Visual Studio C# (2015). You can obtain a full version through Microsoft's Dreamspark Programme. To do this, you need to contact `ithelp@hw.ac.uk` to obtain a student account on their web site.

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.

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 7 or above system. If external tools are used, they need to be included in this executable. Any dependencies on tools are 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

- **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.

- **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.

- **Multi-threading:** The web browser should be designed in such a way that the browser-server communication is separated from the GUI support, using different threads. This way the user can request more than one page at a time. This should be implemented by supporting separate Tabs in the browser, with one thread for each Tab, and with each Tab containing information about its local state.

Graphical User Interface Requirements

In addition to allowing the user to perform the operations discussed above, the GUI for the web browser should support the following:

- Using the GUI, the user should be able to perform the operations discussed in the previous section.

- 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 Visual Studio for C#. 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, History Lists and Tabs. 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.

The HyperText Transfer Protocol (HTTP) governs the communication between a web browser and web server. HTTP 1.1 is defined in the Internet RFC 2616 specification document which can be found at: RFC 2616 specification.

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 **3:30 PM on Friday 21rd/Sunday 23th of October 2016**. For Edinburgh campus the Friday, for Dubai campus the Sunday deadline applies. Submission must be through Vision, submitting all of the above files in one `.zip` file, as well as posting a hardcopy of the report with a cover sheet in the appropriate drop-box near the School Office 1.24. Additionally, an electronic submission of report, sources and application by email is recommended. **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 your demonstration of the implementation. The marking scheme for this project is attached. Following submission, there will be mandatory demos, in which you have to present your implementation, explain its functionality and implementation, and you must be prepared to answer knowledge questions about the programming language used.

5 Report Format

The report should have between 10–20 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 what are expected. If certain conditions cause erroneous results or the application to crash then report these honestly.
7. **Conclusions:** Reflect on what you are most proud of in the application and what you'd have liked to have done differently. An optional final section of references is also encouraged.

Marking Scheme

Criteria	**Marks**
Meeting system requirements	50
Report Quality Content and communication (Design Considerations, Reflections, User guide & Developers guide)	15
The Application Code quality, sufficient comments, Sensible variable/object names, Good code/class/method design.	15
Initiative, innovation, professional conduct, creativity and efforts above & beyond the call of duty	10
Demo (presentation of implementation, knowledge questions)	10
Total marks	100