Industrial Programming, F21SC
Heriot Watt University

Assessed Coursework 1

Written by

Marcin Kopacz (H00176255)

1. Introduction

In this report I want to show evidence of my hard work. I was focusing a lot on creating quality project but in some cases I am still not happy. During development I made few assumptions:

1. User cannot open more than 25 browser tabs.
2. List of Favourites and History is limited by Form
3. History stored is per tab.
4. Some functionality is not instant.
5. Layout responsiveness and design as secondary issue.
6. I am not collecting history on home page

I used MS Visual Studio 2013 – Trial Version. It is very good IDE but sometimes slow and a bit overload with features. I also used git as well for version control.

I try to be minimalistic in coding style some assumptions in data structures might not be not very oblivious. Objectives and requirements where fairly different.

Objective: learn and use advanced language features.

Requirements : Build a web browser with few expensive but similar features

I decided that requirements are more important. I tried to be DRY as much as possible but in some cases that was not easy to achieve.

2. Requirements Checklist

My realisation fully meet requirements. I developed required features in this order:

1. Sending HTTP requests messages
2. Receiving HTTP response messages
3. Multi-threading
4. Home page and homepage setting options
5. History
6. Favourites

I was working in this project in incremental way. I tried to build working features then refactor and test and moving to next.

3. Design Considerations.

I was using MVC pattern. I was trying to separate logic-presentation-view layers.

Because of WinForms is very specific and not best designed to begin with. Presentation layer is by default build on top of Form class with inherited a lot of control and logic. It very hard to move control structures and event handlers to different place. I was trying to move event Listeners to controllers but It was close to impossible. Apparently suggested solution is using partial classes to build maintainable views. Java have better and more robust solution.

I used few classes using singleton pattern. I am in doubt if this was necessary, surely this speeded my development since object creation was simplified. I could also find reasonable explanation. But in terms maintenance my code is bit worse because of that.

In terms of responsiveness and GUI, HTML, CSS is way better and more flexible. This explain why MS is investing into JavaScript related technologies like Sencha, Ext.js. I decided that I will not use fancy elements with complicated event binding. I used basic text field, button and label. Also because WinForms are specific only to windows platform, learning them will not bring any value for me personally.

I wanted to make this project as close as possible to real life applications. It is possible to dynamically change number of browser tabs. Also response data is stored in some kind of cache making sure I will reach decent performance.

I tried to implemented asyc and await construct but without success. C# is lacking in terms of documentation with was another issue. I did not used a lot advance C# feature because for most time standard libs where more than enough.

Overall I think that C# in very elegant languages, It is possible to write beautiful code with swearing too much. I liked to code in C#. It is really fast, tooling is good and nothing to complain.

I used visualisation tool in Visual studio to create interesting class diagrams. First diagram will show high overview of application. It will nicely show MVC pattern.

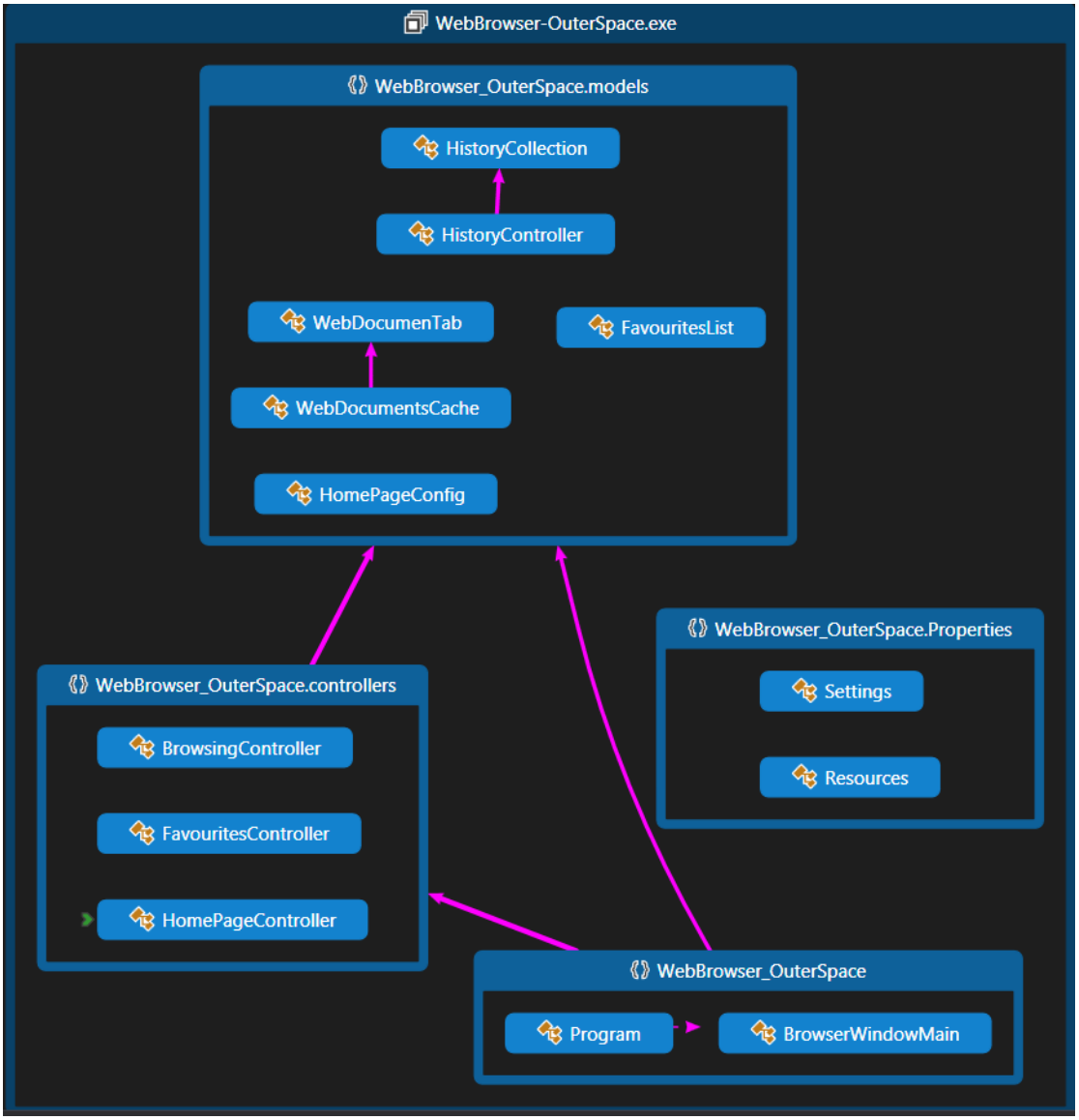


Figure 1. High overview.

BrowserWindowMain is View part MVC. BrowserWindowMain is big class made of partial. I didn't like this approach but again it was hard to change anything. There is MVC.NET framework that focus on this patter much more.

Program class is start-ups class that initialise controllers and load config data to application. It also boots up a BrowserWindowMain.

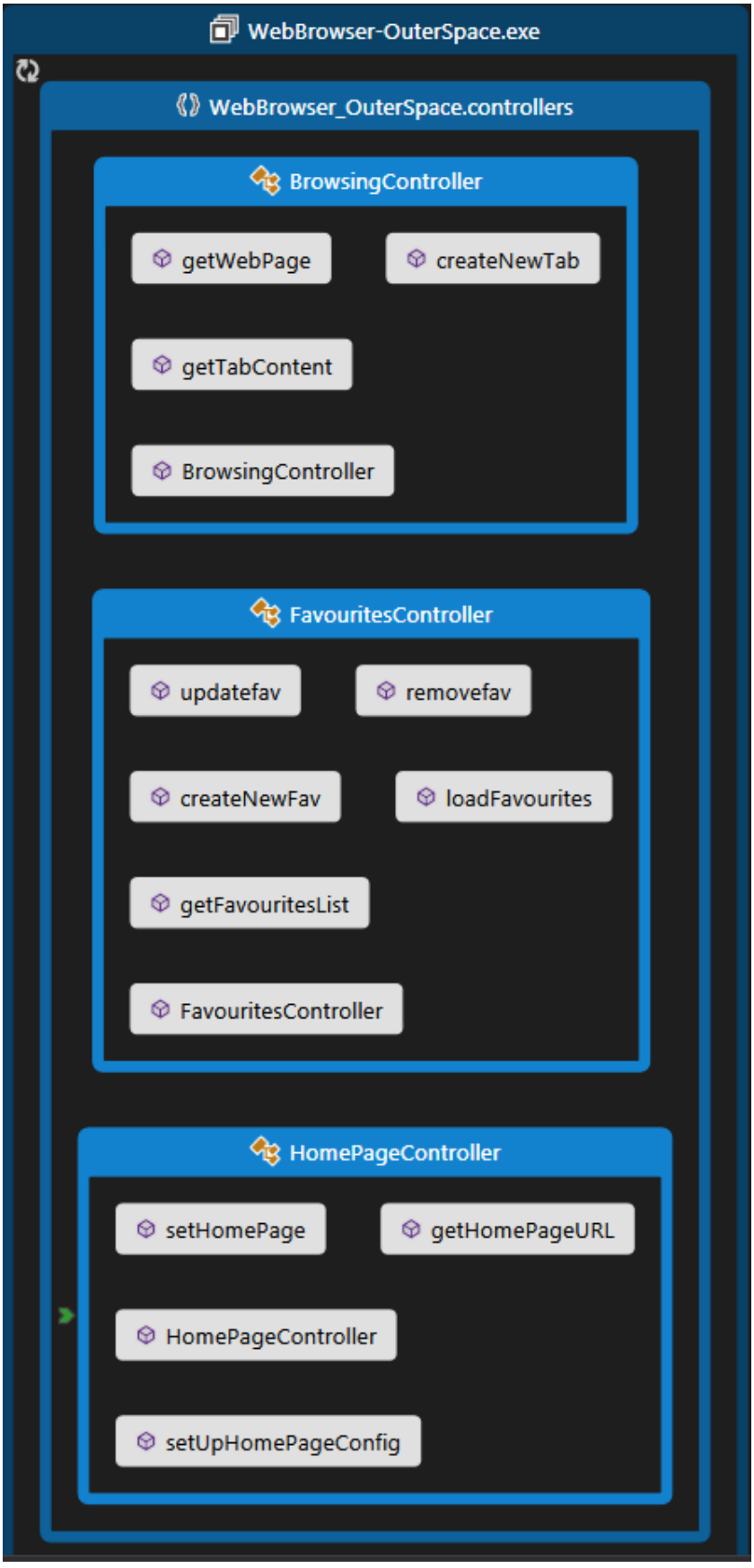


Figure 2. Controller Classes.

I used different controllers for different functionality. In theory they should be handling control and user interactions. In my case they were more like interfaces between model and view. At some point I was considering leaving MVC patter but even in this situation this solution bring a bit more clarity.

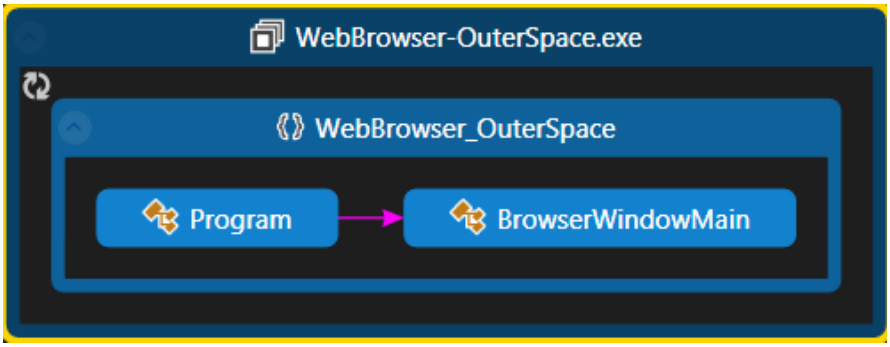


Figure 3. View Classes.

As you can see View is basically one class. I didn't like my Views part of my application, but It was hard to abstract logic.

Next there will be more detailed diagrams of each part of application showing methods and interconnection between them.

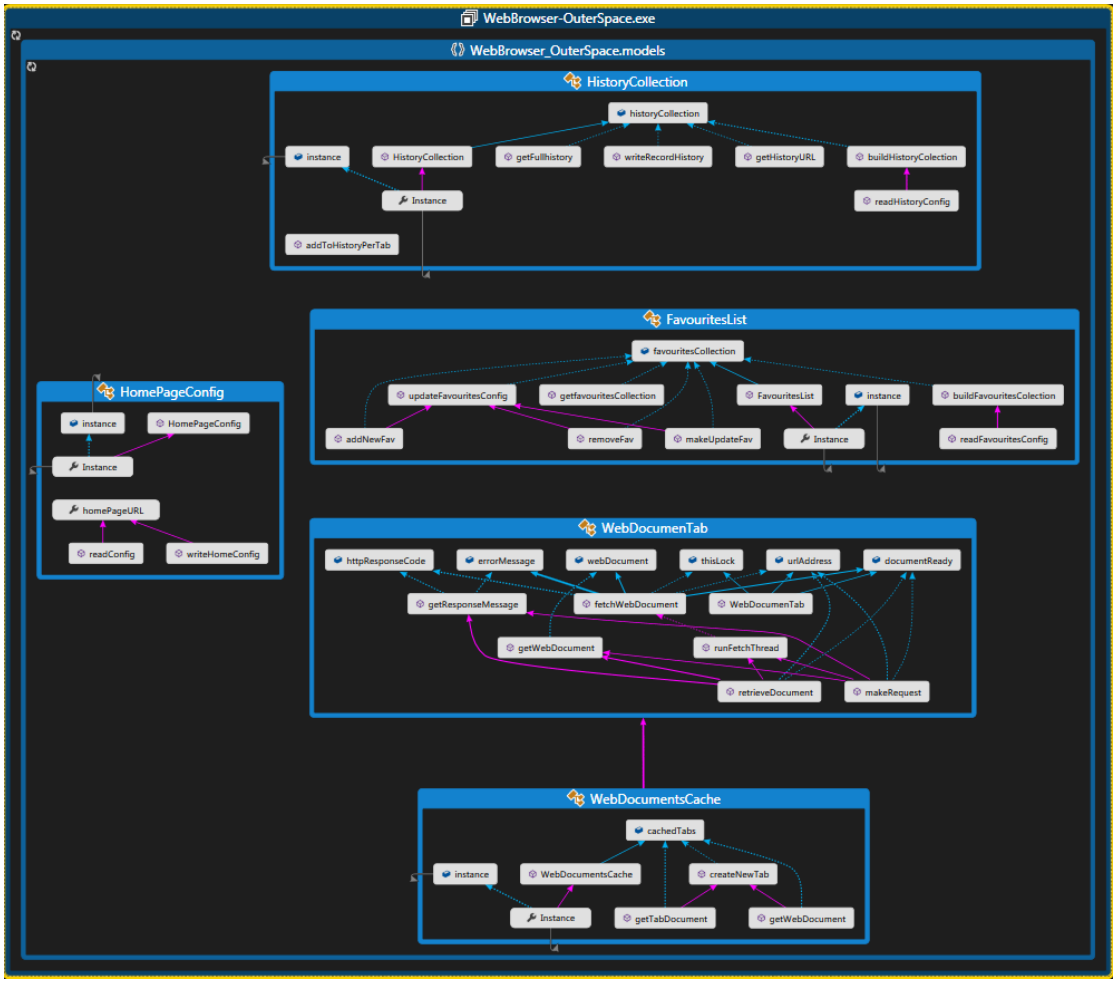


Figure 3. Model Classes with methods overview.

I think I made quite good design of models classes. They are well separated beside cases of composition. There are clear interfaces and if I would like to, it would be possible to create some tests easily. The biggest problems I had, was a threading model. Ideally for this kind of application, each thread per tab should wait for new requests without need to create Threads each time . Actor system will be just fine for this problem. My early implementation using while loop was somehow working but communication between threads was not good enough.

I decided on using thread on method and simply lock on shared resources.

Method runFetchThread() create thread on fetchWebDocument() method with is performing a web requests. fetchWebDocument() should be split into two methods but then Exception handling and threaded model will be even harder.

For data structures I used mostly ArrayList and Arrays. Again I wanted to finish on time and these structures as easy to manipulate and efficient in storing ordered data. Basically from browser tabs to favourite's I was able to easily adapt them. ArrayList are very flexible in C#, maybe they are less performing as List<T> but auto resizing and foreach methods are great thing.

As you can see on next diagram these methods handlers shouldn't be in View. Maybe I missed something a there is simple way to move into controllers.

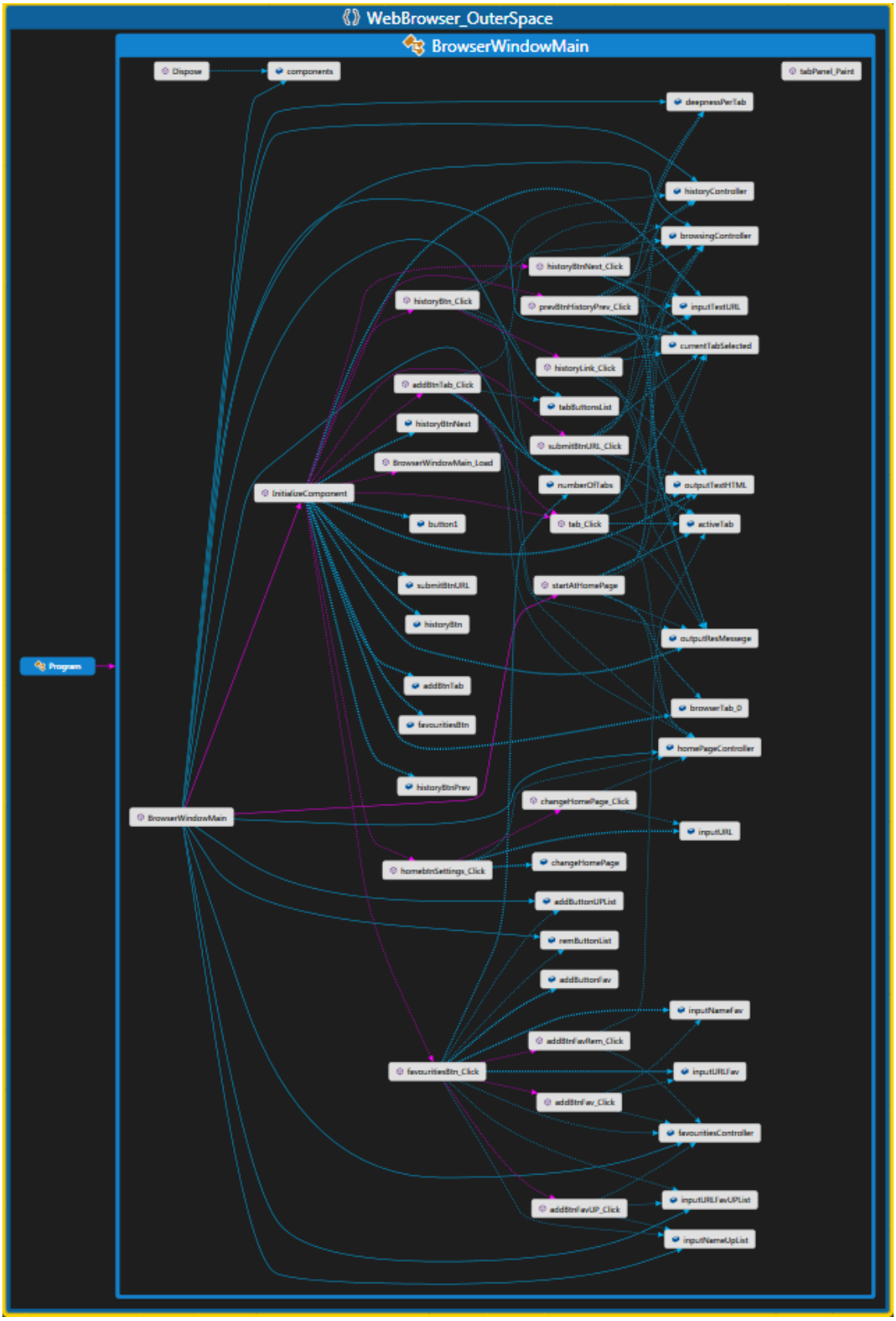
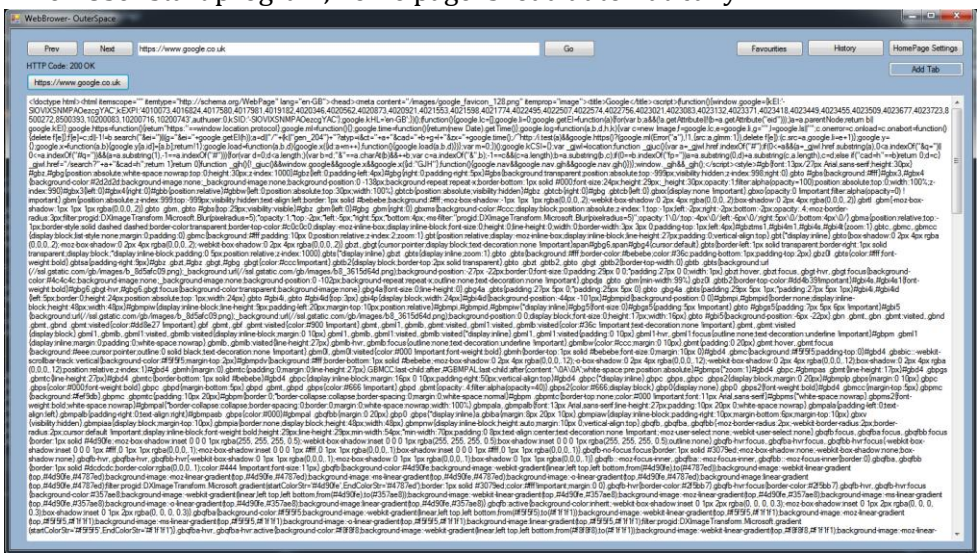


Figure 3. View Classes with methods overview.

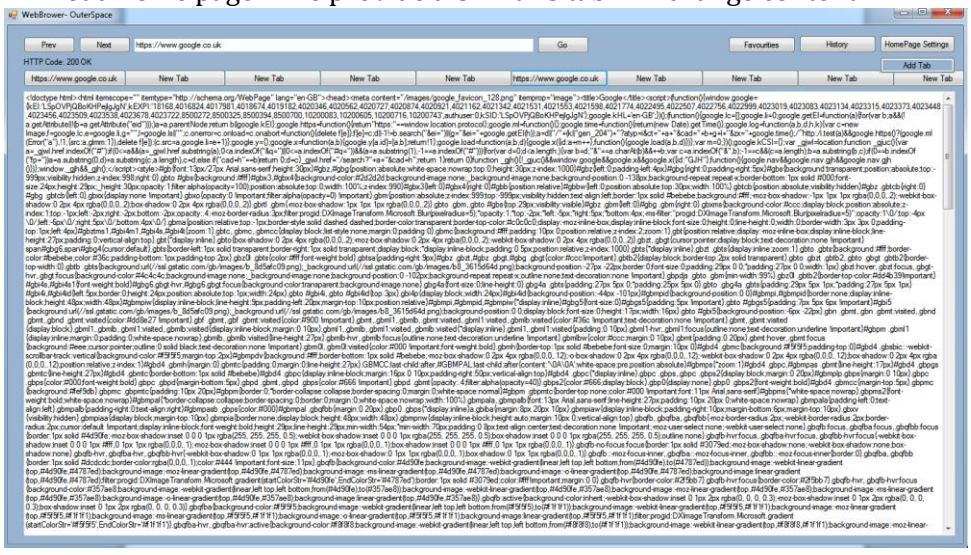
4. User Guide.

I was trying to explain as good as possible but I will provide some screenshots. for this part as well.

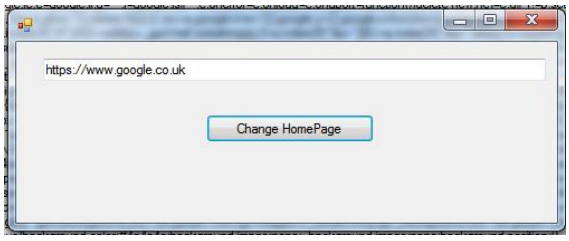
1. When User start program, home page is load automatically



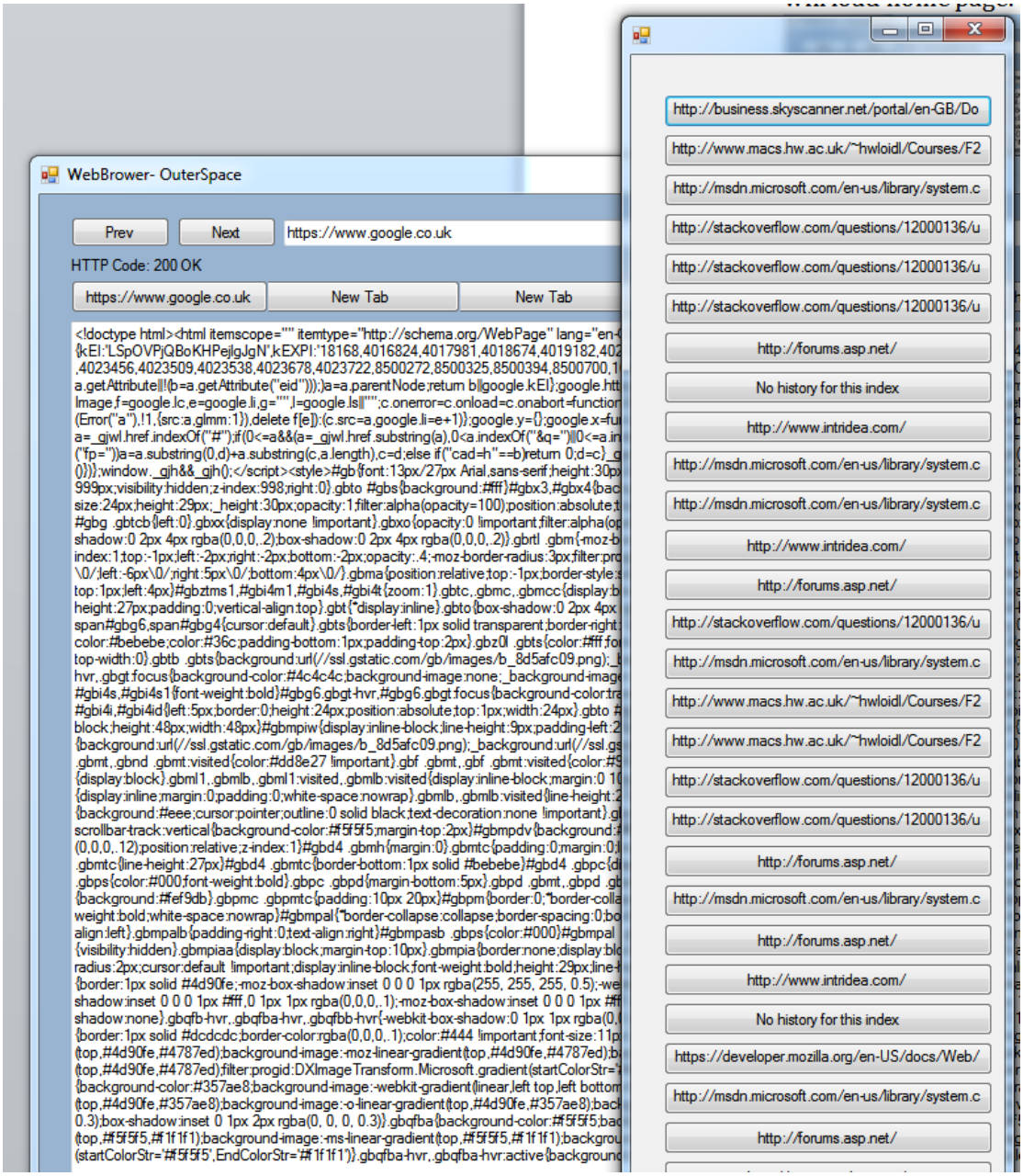
2. User can create up 25 open browser tabs. When he click on newly opened tab. He will load home page. If he provide a URL this tab will change content.



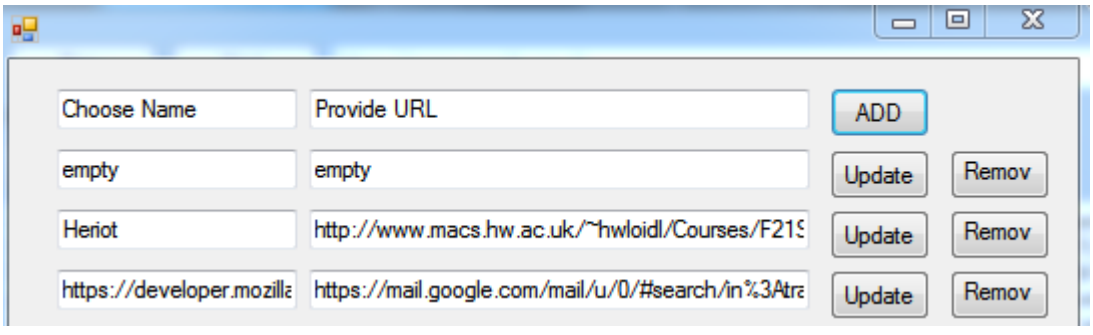
3. User Can change home page using HomePage Settings button. Change will work for new tabs.



4. User can see browsing history using history button, and he can travel back for each tab. Using Next and
Prev.



5. User can use favourites button to see editable List of favourite website.



5. Testing.

I tested application during development. In requirements we didn't have specific method. I was using more or less BBD. I was testing each feature extensively.

I sure that all features are well tested. But I have small issues with some functionality of favourite List. In some cases it is not working as intended. It might be caused by multiple dynamically created controls.

For threaded model I decided to use join even if performance is much lower. In some cases even with lock but without join. Application loose some requests. I could add wait, but this could make it unstable.

6. Conclusion.

I would like to improve threading model to make more performing and more robust. If I would have a bit more time I could spend some on GUI. Dynamically generating controls is doable but not very recommended.

I am proud of architecture and over all well done methods. But I would like to improve and use async model with would be great in this application. I should use different events for some cases like writing config files on form close etc.

I learned a lot about programming in C#.