The basic functionality of this software was very easy to engineer, windows forms has a fairly low learning curve. This encouraged me to try an implement things beyond the spec such as sorting algorithms for the history and favourites. I even began to implement a filewatcher, with the intent to handle multiple browsers running in parallel reading and writing the same files, while all maintaining history and favourites in sync with each other. I strongly feel that there are so many areas to improve the software, for example the GUI lists do not reflect the sorted order of the entries in History and Favourites. If starting again I would implement a function to convert the sorted List into a ToolStripItemCollection type instead of just adding the most recently modified one to the end. Another area I feel could be really improved is error handling, C# feels quite relaxed about handling errors explicitly compared to other languages that use the functional paradigm such as Rust where whenever an error might be raised you have to handle it or explicitly tell the compiler it wont be an error (.unwrap()) [1]. Unfortunately this means its very likely there are a number of unhandled exceptions in my code because I often found myself handling only one of several exceptions (the easiest one to throw), and without going through it with a fine tooth comb I may not encounter the unhandled exceptions at all. I also feel I didnt make much good use of the existing interfaces and likely re-engineered alot of algorithms due to my lack of familiarity with the subtlelys of C#. Overall it felt like a productive project that required revisiting a wide range of programming skills such as async, inheritence, considering the effect of immutability and working with collections.