Ethan Blatti

Robert Dudasik

Nicholas Biegel

Simple Web Browser: Final Report

The purpose of this report is to provide documentation as well as insight on our semester project: a simple web browser in C++. During the development process, many things occurred which we had not planned for. Originally, we decided that we were going to split up time and work on this project day by day- uploading our code to Github. Although using Github allowed us to have a fully convenient way to share code amongst ourselves, it wasn't always the best way to do so. In some instances, one of us would finish a section of the program which the other was working on at the same time, this created a conflict of interest and design. To combat this, we were able to chat with each other live and derive a solution integrating key aspects of each other's implementations. Another assumption we made was there was going to be no interruption of our deadlines. However, we all know that complications arise within everyday life whether it be other classwork, job changes, or anything else that could hinder deadlines. We will start off the report with a checklist of the requirements.

Due to those unexpected changes in our schedules causing deadlines having to be changed, not all features were implemented in the web browser.Some requirements were easily implemented using Chromium. Chromium is a base which we used in our program because it is open-source and well documented. Using Chromium allowed us easy implementation of certain features such as HTTP response messages. Chromium also automatically handles things such as

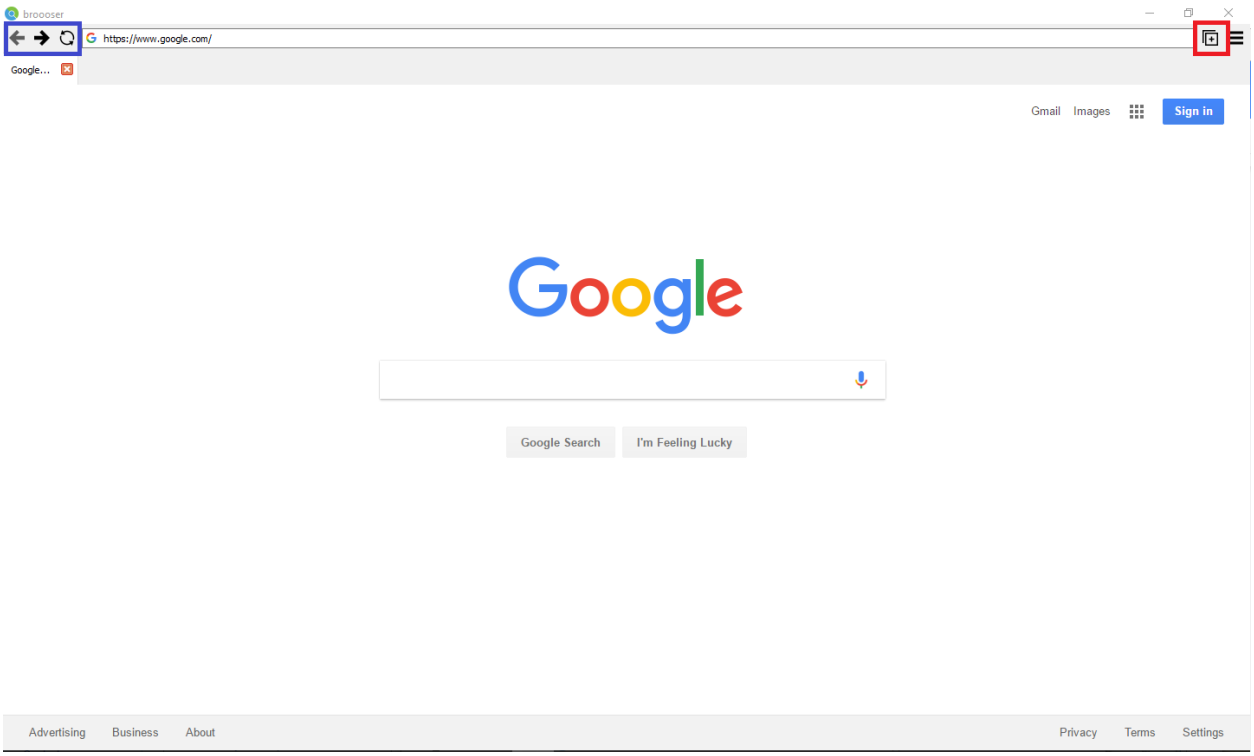
400-bad request, 403-Forbidden, and 404-Not Found. We took an alternate route and created a simplistic dialog displaying these rather than using the stock chromium ones. To allow easy navigation through history, favorites, etc we created a JSON file to store all of these. Using the JSON file allowed us to implement Homepage, Favorites, and History pretty easily. Another one which was a breeze because of our use of libraries was multithreading. This was taken care of on the Chromium side rather than us having to create separate threads within the C++ code. Since we were more pressed on time than we had anticipated, there were a couple features which couldn't be squeezed in. Those features are blocking pages and bookmarks. Faced with these time concerns, we focused on the core of the web browser rather than the "nit-picky" behind the scenes features and implementations.

We wanted our web browser to have a simple, yet intuitive GUI. When designing our web browser, we were inspired by other web browsers that exist today. Therefore, we included features that are found in the most popular web browsers and we designed our GUI around those same browsers. Similar to the browser Google Chrome, we included our back, forward, and refresh buttons to the left of the address bar so that they could be accessed easily by the user. We also included a menu at the upper right-hand corner of the web browser. This menu initially includes options to create a new tab, create a new window, bookmark a website, block a website, set a website as the homepage, and exit the browser. Located below the address bar are the tabs for the different web pages that a user may have open. However, it was important for us to include a feature that will remember the tabs that a user has open at a given time. For this reason, we implemented the "session" feature. The session feature allows users to store the tabs that they have open at a given time and open them again later. For example, if a user wants to research a particular topic, they can open several tabs germane to that topic. They can then create a session

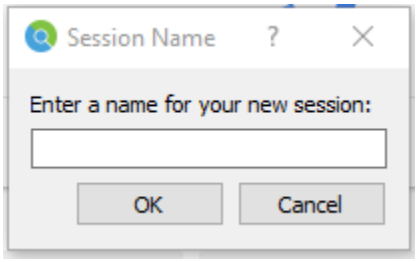
by clicking the button at the right of the address bar. The user can then give the session a name and save it. Now, if the user closes the browser, they can easily reopen the tabs that they had open by going to the menu and clicking “Sessions”. The user can then resume their research from here.

We also made careful design choices when writing the code. We used QT as the backbone for our web browser. QT is an application framework that allows developers to write an application that will work on many different platforms without having to make significant changes to the underlying code. We also used Chromium , an open-source web browser, as the base. We were able to take the best elements from Chromium and, using QT, improved upon them and made a basic web browser. The design of our web browser is based heavily around the BrowserWindow class. This class holds the the navigation widgets and the tab/view widget. Each of these areas are then controlled by functions that set up each, respective area. We also designed a JsonManager class so that data from certain widgets is saved into a JSON object. The JsonManager class allows for seamless session management, homepage, blocking, and bookmarking of sites.

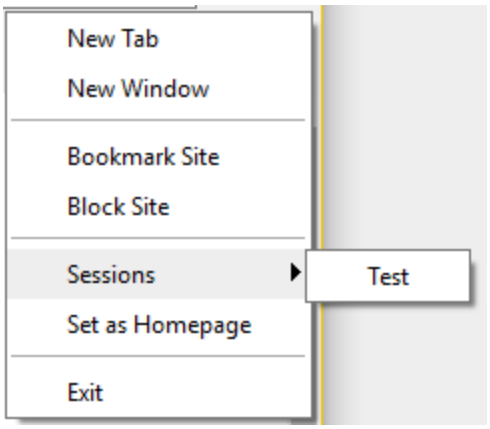
Once the application is launched, it is fairly intuitive to use. Users can go to different web pages by entering the desired web page’s URL into the address bar and hitting enter. The user can also navigate through the browser history by using the back and forward buttons located in the upper left-hand corner. The user can also use the refresh button in that area to refresh the page. This is shown below in the blue box.



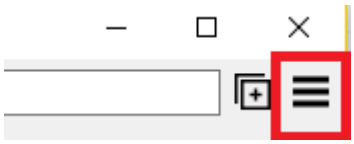
Also included in the above screenshot is the “New Session” button. This is outlined in the red box. Clicking this button saves the tabs that are currently open and a dialogue box pops up. This dialogue box then prompts the user to name their session.



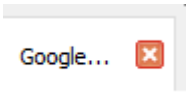
Once the user enters in a name for their session it is then saved under the “Sessions” option in the menu. The session can then be reopened by clicking on its name.



The other menu options are visible in the above screenshot. These options allow users to create new windows, tabs, bookmark sites, block sites, set and delete their homepage, and exit the browser. All of these options can be accessed by clicking on the menu icon located in the upper right-hand corner, as depicted below.



Finally, users can close tabs by clicking on the red “X” on the tab that the user wishes to close. An example of this is shown below.



As previously stated, our web browser was developed from a combination of QT and Chromium. The BrowserWindow class is the core of our application. In this class, the navigation widgets and the tab/view widget are managed. Two areas are created. One area is used for navigation, where the user enters in the URL and uses the functions built into the browser. The

second area manages the tabs and displays the content at the web page. Each respective area is then controlled by a function that "sets up" the area. Saving data was also important in our design. There are many different widgets that require the use of saved data. It was determined that the most effective way to save data was through the use of JSON objects. The saving of these JSON objects is managed by the JsonManager class. This class allows the data from different widgets to be saved and accessed later. This is helpful since it permits seamless session management, homepage, blocking, and bookmarking of sites.

When it came to testing our browser, we were the most concerned with the basic functionality. We thoroughly tested the browser's ability to reach different web pages and display their content appropriately. We did this by typing different URLs into the address bar. We found that valid URLs would appropriately return the corresponding web page. However, if a URL was invalid, the browser displays the built-in page for a 404 error. Ideally, with more time, we would like to turn invalid requests into a Google search and then display the results of that search instead of the 404 error. We also tested the session feature thoroughly. We opened up various combinations of tabs and gave each combination a different name which contained combinations of uppercase and lowercase letters, numbers and special characters. In each case that was tested, the session was named appropriately and functioned properly. Other functions that rely on the use of menu items and buttons in the GUI, such as the setting of a homepage and creation of new tabs, were tested on various web pages. These functions all worked properly for each case that was tested.

While implementing this application, we were exposed to many new things. Sure, we were exposed to new ideas in C++, but we were also exposed to completly new languages and APIs. We started this project using MFC or Microsoft Foundation Class- a library which wraps

portions of the Windows API with C++ classes; allowing functions which allow them to use a default framework. Using MFC was very confusing at first and although Microsoft had laid out documentation, the documentation often seemed too cluttered-especially when referencing to other libraries or header files. To us, the MFC library seemed to “daisy-chain” too much and seemed impossible to follow variables or references to see where they originated. Because of all this, we decided to put MFC to the side and research some more languages. We decided on using QT, described briefly above. While using QT, we had to change our mindsets on how to go about the program, as syntax and usage differs between the two. We also stepped away from the MFC library because it favored Windows platforms. Using QT, we easily overcome this complication because QtT is cross-platform. As a result of this, we believe that this allowed a much greater usability of our application not only on a system level, but wider as in mobile devices.

What stands out most in the application-our most proud feature- is the usage of what we call sessions. “Sessions” allow a user to save a list of tabs, or what they’re currently viewing in a browser instance for later. Users can use this to their benefit without the need of plugins or special add ons to the browser itself. To do this, we created a JSON file which will store information about the current session. Let's say a user is interested in a particular programming language to learn-perhaps python. We all know that you can find excellent resources which are very hard to dig and find. Often, we try to search for these resources and have a hard time finding them. To reference them, we poke through history for what seems years until finally giving up because we cannot remember that keyword we searched for. Our sessions allows a user to save all those excellent resources, and simply re-open them at a later date. The user will have all tabs which were open at the time of the save, whether they open the saved session a day, week, or even months later. Unfortunately, this feature only saves and opens session from a local

machine. One thing that we would love to work on and polish is this implementation on more of a global level. If our project was to take off and become the next great Chrome, Firefox, or IE, we would love to see those sessions saved to a “cloud profile.” With this information stored on the web, it would allow the user to log into the browser on any machine and be able to access their saved sessions from anywhere! The bookmark and block do not work. We were not able to ring out errors in these features due to spending time implementing the others. Other than that, we implemented almost everything which we wanted to and what was required of us. A final thought, if we were in a setting where we could all collaborate on the project together for extended periods of time, we feel that we could have come up with better ideas by allowing more time for thoughts to accumulate. Also, speaking through a chat does not always get a point across as it does not allow adequate explanation- as typing ideas is always challenging and tedious. Provided bountiful collaboration time and stricter deadlines, we truly believe we could make what is now a “simple web browser” something as lavash as the most popular browsers today.