Webcessible

The internet for all

Marc Frame | Anthony Aoun | Nicolas Gnyra | Cedric Brisebois
December 13th 2018
GNG 2101 | Dr. Hanan Anis

Abstract

Webcessible was an attempt to build a simplified web browser that would make browsing the web easier on people with physical disabilities. Webcessible was setup in such a way to allow for a variety of inputs to suit users with varying handicaps.

Table of Contents

List of figures

List of tables

Main Body

Need Specification

This project involved solving a problem at a larger scale. We planned to build something that would solve a problem many patients at the hospital experienced, but all in many different ways. Thus, the proper development of our need specifications was crucial for the long term success of our solution.

Our problem statement was to build software capable of simplifying web pages to accommodate patients who have difficulty using the Internet with standard input methods and have to use custom tools to interact with computers, whether it be eye tracking, joysticks, or anything that isn't the standard keyboard-and-mouse combination.

We had chosen to make a Chrome web extension as our project. It was easier than making an entire new browser or a website (an endeavour which, after further thinking, would take an overwhelming amount of time and effort). We have also chosen to have our website configuration files stored remotely on something like an AWS S3 bucket rather than having them stored locally, while still having an option to have some locally in case the user would like something completely custom rather than something that was built for general use.

Client needs on a scale of 1 to 5, 1 being least important and 5 being most important:

#	Need	Importance
1	Multiplatform (accommodate for different OSes)	1
2	Easy to read/comprehend code (lots of comments & config template)	5
3	Accommodate for different config files for specific web pages	4
4	Configuration files are easy to implement for the user/helper	4
5	Interface is easy to use	5
6	Able to change complexity of web page	3
7	Does not hinder a person's use to navigate the Internet	5
8	Able to accommodate for different types of input (buttons, joysticks, sliders, etc.)	3
9	Easy to install	3
10	Runs reliably	4

Table 1: Need finding

Benchmarking

We found very little other solutions that addressed this problem. Most browsers already have accessibility features, but they are mostly aimed at people with sight problems rather than people with limited motor functions. Many browsers also have a "reading mode" that strips web pages of a lot of the unneeded bulk, but these sometimes remove important information that our client will want to keep, and these special modes are far from being optimized for ease-of-access. There are also services such as loband (http://www.loband.org/) or Rocket Readability for Google Chrome which are advertised as tools to simplify web pages, but they are given in a one-size-fits-all formula that often doesn't work with more complex websites such as Gmail and YouTube, most of which are the websites with which our clients have the most problems.

Metrics List

Metric #	Need #	Metric	Imp.	Units
1	7	Additional time taken to load web pages	3	s
2	1	Operating systems supported	1	list
3	2	Code quality	4	%

4	3	Number of websites configured	3	#
5	8	Input methods supported	4	list
6	6	Flexibility of configuration	4	subj
7	5	Complexity of user interface	4	subj
8	4	Complexity of configuration	5	subj
9	10	Number of bugs	4	#
10	9	Ease of installation/setup	3	subj
11	2, 4	Documentation coverage	4	%
12	10	Unit test coverage	2	%

Table 2: list of metrics

Marginal/Ideal Values

#	Metric	Units	Marginal Value	Ideal Value
1	Additional time taken to load web pages	s	< 5	< 1
2	Operating systems supported	list	Linux	Windows, Mac, Linux
3	Code quality	%	> 80	100
4	Number of websites configured	#	3	> 10
5	Input methods supported	list	Eye tracking	Eye tracking, joysticks, buttons, sliders
6	Flexibility of configuration	subj	medium	very high
7	Complexity of user interface	subj	medium	low
8	Complexity of configuration	subj	medium	low
9	Number of bugs	#	< 10 (nonbreaking)	0
10	Ease of installation/setup	subj	medium	very easy
11	Documentation coverage	%	75	100
12	Unit test coverage	%	50	100

Table 3: metric ideal value list

Specifications

#	Metric	Units	Value
1	Additional time taken to load web pages	s	< 3
2	Operating systems supported	list	Linux, Windows
3	Code quality	%	90
4	Number of websites configured	#	> 5 (nonbreaking)
5	Input methods supported	list	Eye tracking, joysticks
6	Flexibility of configuration	subj	high
7	Complexity of user interface	subj	low
8	Complexity of configuration	subj	low
9	Number of bugs	#	< 5
10	Ease of installation/setup	subj	easy
11	Documentation coverage	%	100
12	Unit test coverage	%	100

Table 4: specifications

Conceptual Designs

After grasping the problem at hand, developed specific design criteria which led us to select the idea of building a web browser extension. The design criteria we came up with were:

- Not too complex to program
- Doesn’t cost too much to have up and running continuously
- Easy to update/add support for new websites
- Minimal additional time taken to load web pages
- Lots of supported input methods
- Easy to install & set up
- Flexible configuration
- Easy to support in the future (handing project to other developers shouldn’t be an issue)

Conceptually, our initial designs were:

Facebook

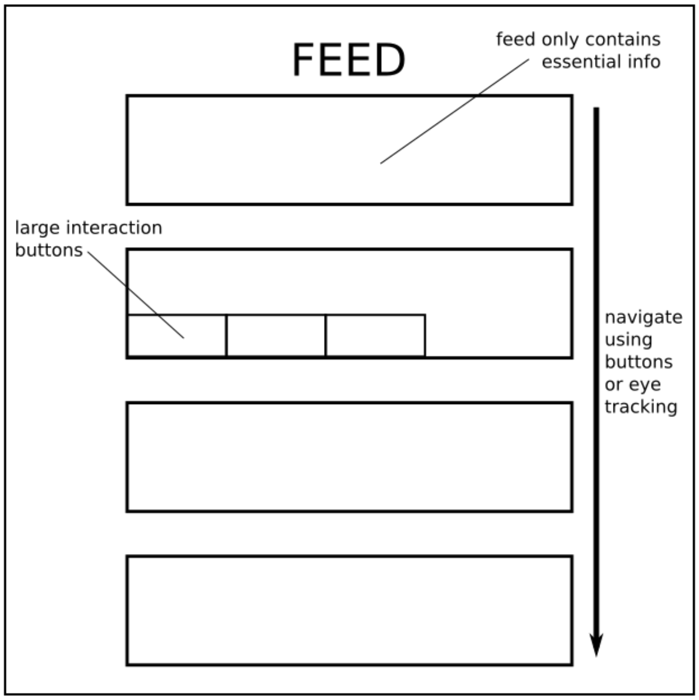

FEED
feed only contains essential info
large interaction buttons
navigate using buttons or eye tracking

Figure 1 : mockup for a feed

Gmail

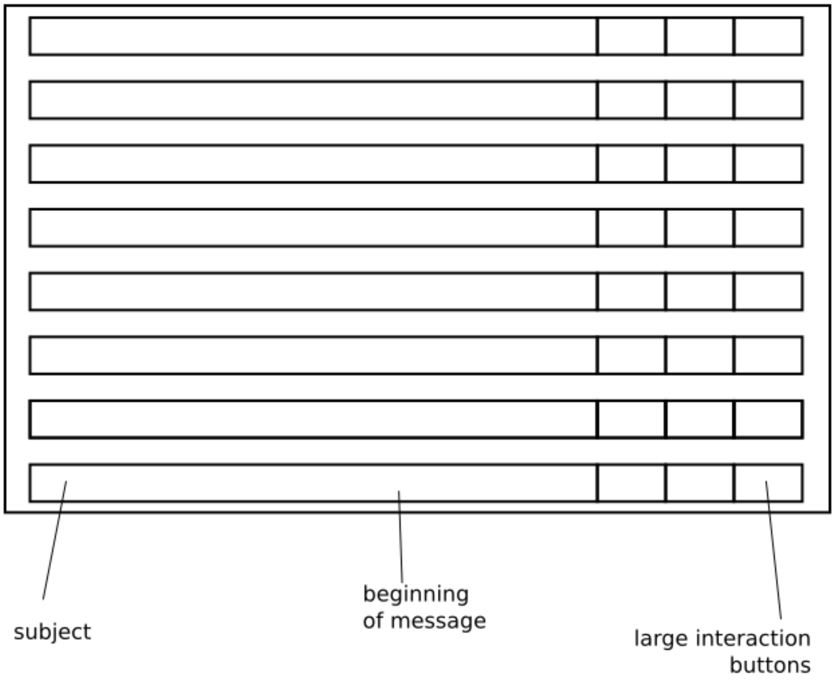


Figure 2 : mockup for gmail

After several weeks of work, we were capable of turning these designs into real life. We build a functional version of Google and Wikipedia.

Project Planning

Our project planning was done nearly exclusively through Trello, a platform on which it was possible for us to create a kanban board to organize our tasks. We decided to organize our project in this manner after discussing different ways of dealing with tasks and our timeline

Feasibility Study

Following the TELOS factors, we deemed that our project was feasible. Team members all had at least two years of software development experience, so while some of us did not have skills in JavaScript, the skills acquired through software development were easily transferable. As our project is purely software, there were virtually no costs associated with its development. The product operates on publicly available data and is fully open-source, so there were no legal problems to be worried about. Since most of the parts of the project did not directly rely on each other, it was deemed feasible to develop them in parallel and integrate them near the end. We expected development to follow a flexible but well-defined schedule so everything would be done on time.

Prototyping

Our approach to prototyping was very iterative due to the use of the Agile development model. This allowed us to always have a functional prototype available to show to our client. We unfortunately did not keep a history of the progress of our prototype as features were added, so no screenshots could be provided for this section.

Testing

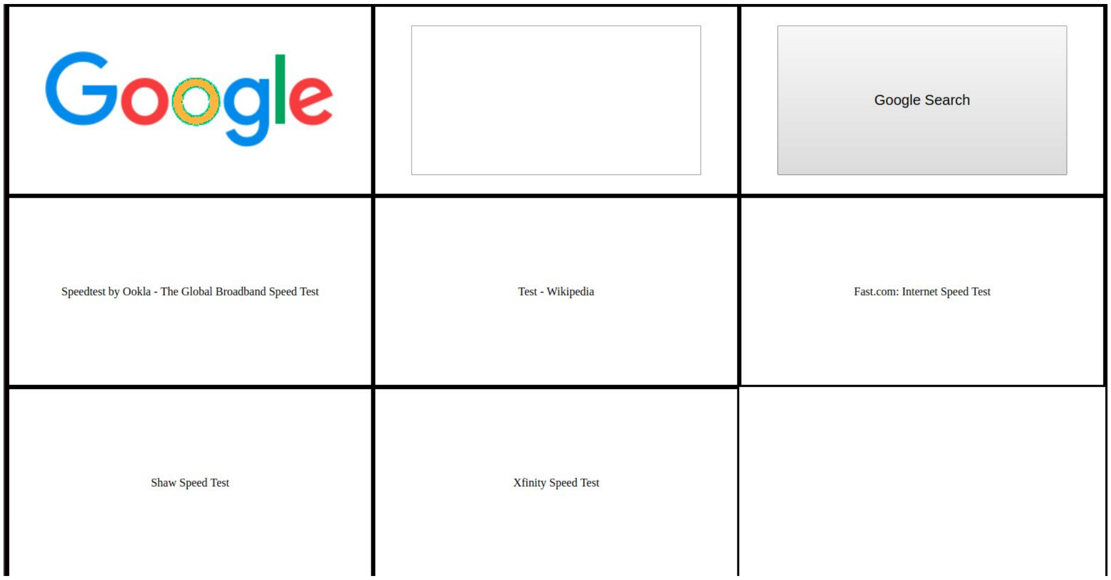
Testing was mostly done on our end. We unfortunately did not have access to the Sensact hardware, and therefore had to use a standard keyboard and mouse to interact with our product. However, we were told by our client that the Sensact interacted with the computer by simulating certain keystrokes and mouse movement, so our testing was mostly based upon using a few keys to interact with our product.

Final Product

Our final product is a robust framework that allows people with little knowledge of programming to simplify various websites’ user interface through configuration files. While some of the ideas we had along the way were unfortunately not implemented in our final product, they could easily be built upon what we have created over the last 3 months to improve the product.

The following images are taken directly from the working final product.

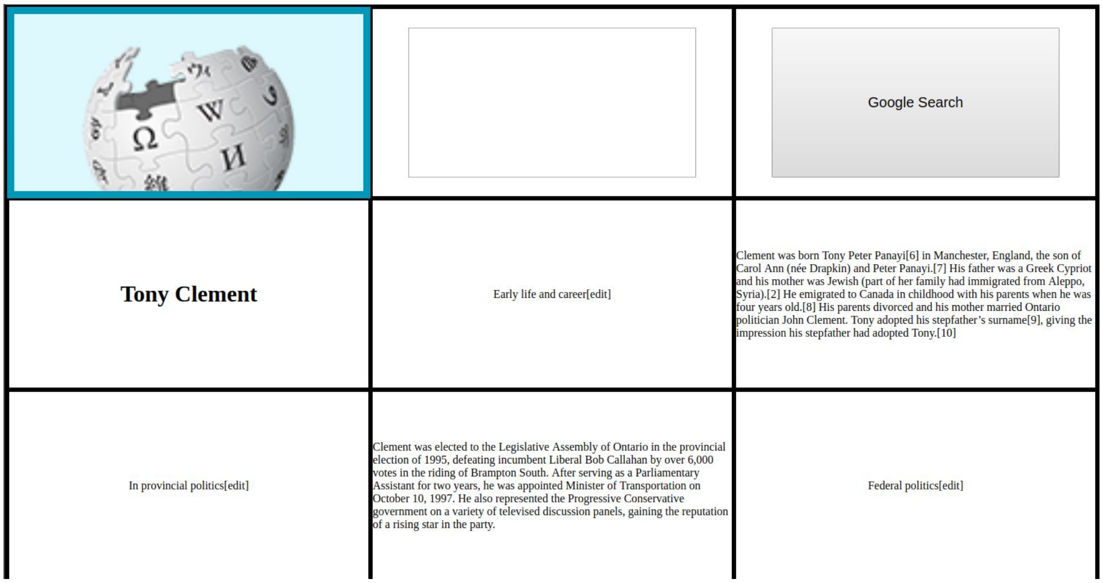
Google



Google		Google Search
Speedtest by Ookla - The Global Broadband Speed Test	Test - Wikipedia	Fast.com: Internet Speed Test
Shaw Speed Test	Xfinity Speed Test	

Figure 3 : example of final product loading google

Wikipedia



		Google Search
Tony Clement	Early life and career[edit]	Clement was born Tony Peter Panayi[6] in Manchester, England, the son of Carol Ann (née Drapkin) and Peter Panayi.[7] His father was a Greek Cypriot and his mother was Jewish (part of her family had immigrated from Aleppo, Syria).[2] He emigrated to Canada in childhood with his parents when he was four years old.[8] His parents divorced and his mother married Ontario politician John Clement. Tony adopted his stepfather's surname[9], giving the impression his stepfather had adopted Tony.[10]
In provincial politics[edit]	Clement was elected to the Legislative Assembly of Ontario in the provincial election of 1995, defeating incumbent Liberal Bob Callahan by over 6,000 votes in the riding of Brampton South. After serving as a Parliamentary Assistant for two years, he was appointed Minister of Transportation on October 10, 1997. He also represented the Progressive Conservative government on a variety of televised discussion panels, gaining the reputation of a rising star in the party.	Federal politics[edit]

Figure 4 : example of final product loading wikipedia

Business Model

Our business model was built in order to resemble to the most recent successful companies out there today. In other words, we want to take advantage of the current trend of freelancers coming to market. To date, we came up with a simple business model that can provide a constant stream of configuration files while we focus on the platform from which they are built. Our business model consists of a monthly subscription held by our users and a community of freelancers who build the desired configuration files demanded by our users. In short, freelancers build configuration files using our extension/platform in order to supply the various configuration files needed to supply the demand of the users. In terms of payment, freelancers receive a basic pay depending on the popularity of the config file for which they built. The amount of downloads a file will get will determine how much the freelancer will get paid. In that case, we can assure that the payment is correlated to the amount of users. The following graph is a simple representation of the business model.

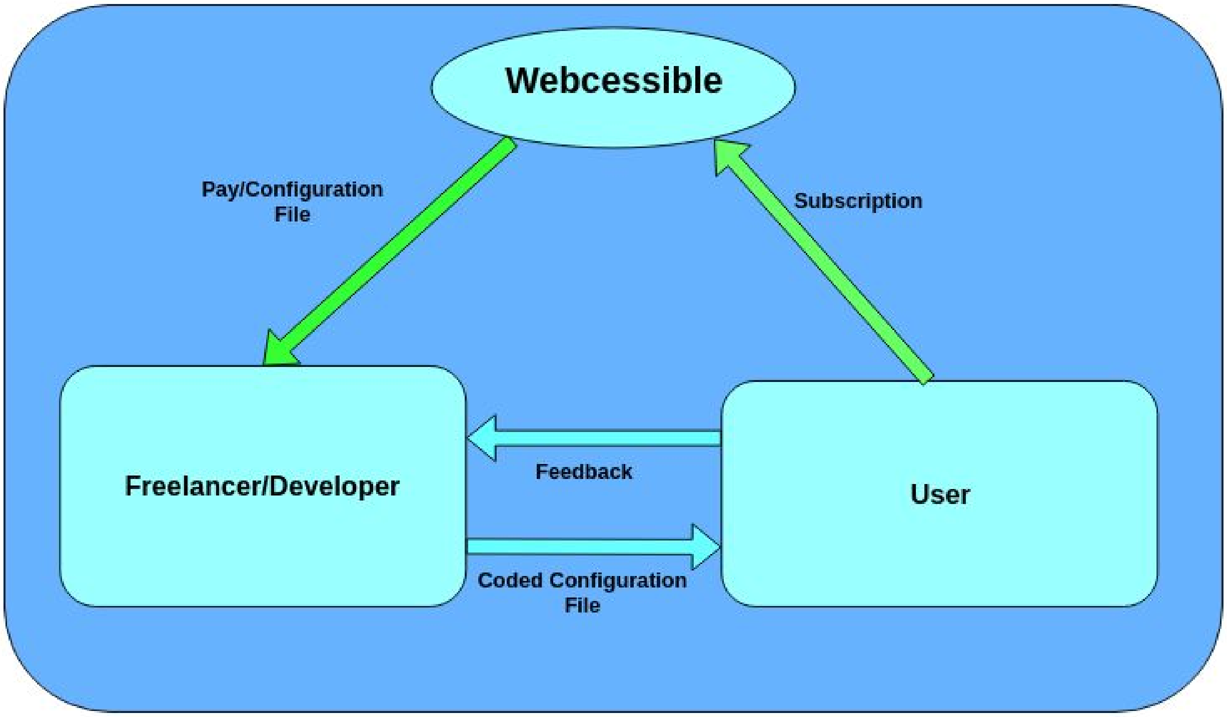


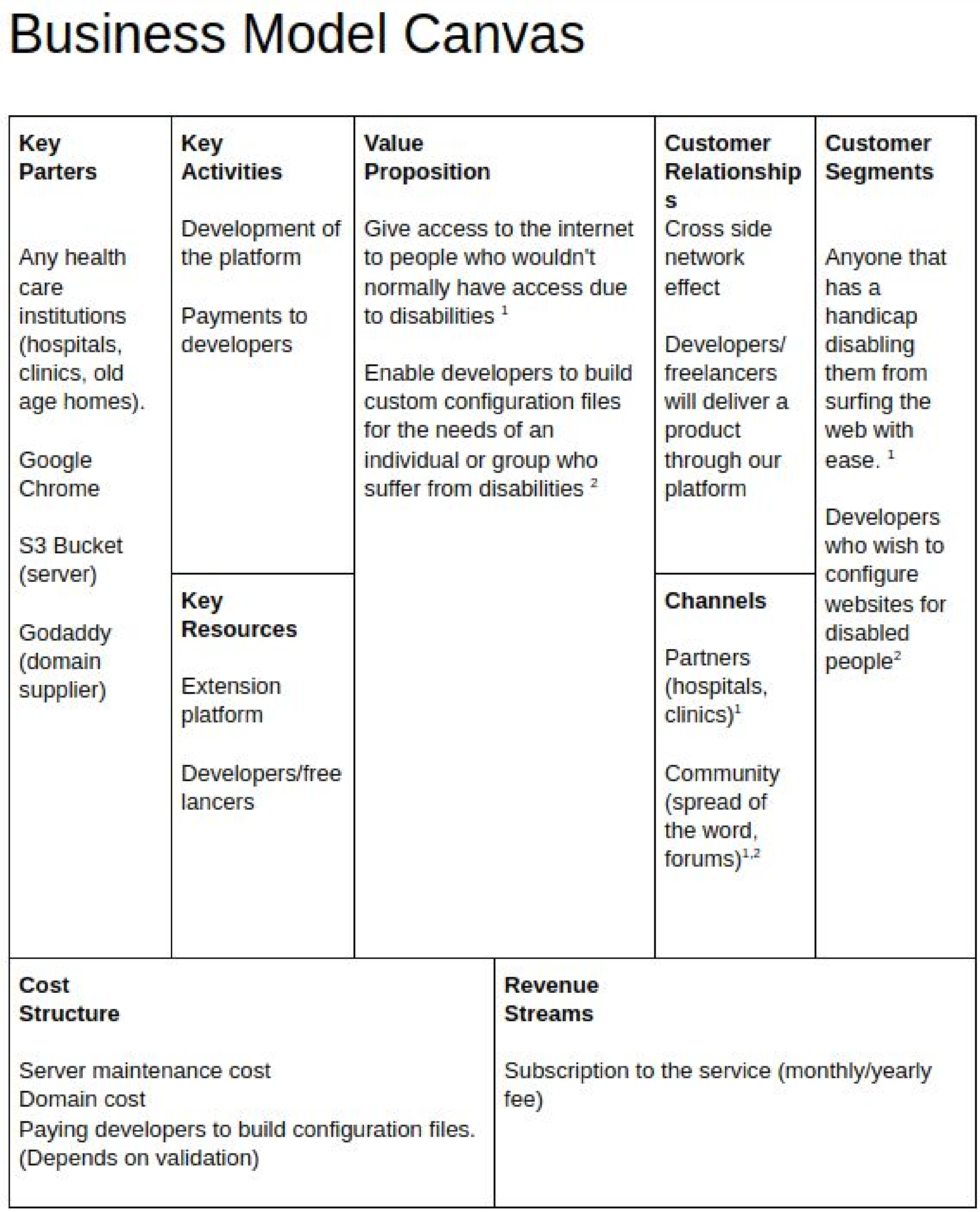
Figure 5 : monetization channels for webcessible

Our business model canvas follows. This canvas is a more detailed visualization of our business model, outlining who we can partner with, who we sell to ect.

Economic Analysis

During the building of our extension, we conducted an economic analysis in order to determine our initial costs, our BOM and out income. We also conducted a 3 year plan. Since this is a software and a BOM would not be as conventional, we built a formula in terms of the amount of users in order to determine the cost by user. The formula follows:

$$F(n) = (([\text{Cost of Development}]) + ([\text{Server Hosting}] + ([\text{Additional Cost per User}] * n))) / n$$



Business Model Canvas

Key Parters	Key Activities	Value Proposition	Customer Relationships	Customer Segments
Any health care institutions (hospitals, clinics, old age homes). Google Chrome S3 Bucket (server) Godaddy (domain supplier)	Development of the platform Payments to developers	Give access to the internet to people who wouldn't normally have access due to disabilities [1] Enable developers to build custom configuration files for the needs of an individual or group who suffer from disabilities [2]	Cross side network effect Developers/ freelancers will deliver a product through our platform	Anyone that has a handicap disabling them from surfing the web with ease. [1] Developers who wish to configure websites for disabled people[2]
	Key Resources Extension platform Developers/freelancers		**Channels** Partners (hospitals, clinics)[1] Community (spread of the word, forums)[1,2]	

Cost Structure	Revenue Streams
Server maintenance cost Domain cost Paying developers to build configuration files. (Depends on validation)	Subscription to the service (monthly/yearly fee)

Figure 6 : Business Model Canvas

In terms of cost that is unrelated to a BOM, we determined that 4000$/month would be spent on paying freelancers who would build configuration files. 10$/month would be spend in order to maintain our servers meant to hold the configuration files and the website. 10$/year would be spend in order to pay for our domain name. Finally, 3600$/month would be spend in order to hire developers who would maintain our platform. Maintenance would include bug fixes, updates and so on. We also included a 3 year forecast in terms of income and expenses. The following table is a brief overview of a 3 year plan.

year:quarter	# Users	Capital
1:1	10	**Sales Revenue:** $0 **Operating Cost:** $10 * 4 months **Operating Income:** $-40
2:1-4	2500	**Sales Revenue:** ($20 * 12 months * 2500 users) = $600,000 **Operating Cost:** $100 * 12 months + $3600 * 2 * 12 months + $3000 * 12 months + $50000 = 173,600 **Operating Income:** $426,400
3:1-4	10000	**Sales Revenue:** ($20 * 12 months * 10000 users) = $2,400,000 **Operating Cost:** $100 * 12 months + $3600 * 2 * 12 months + $8000 * 4 * 12 + $3000 * 12 months + $12000 * 12 months + $50000 = $701,600 **Operating Income:** $1,698,400

Table 5: 3 year plan

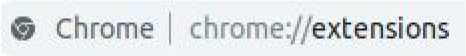
Please refer to our deliverable H for a more detailed 3 year plan. During our economic analysis, we also calculated our NPV which is 2,329,516$.

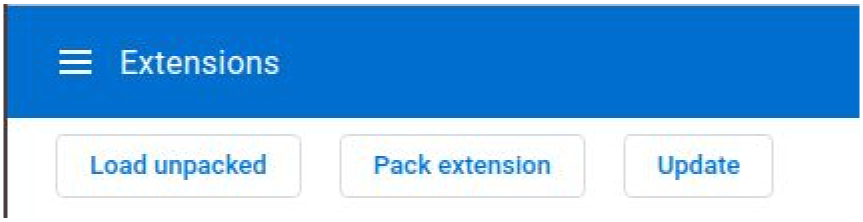
Conclusion

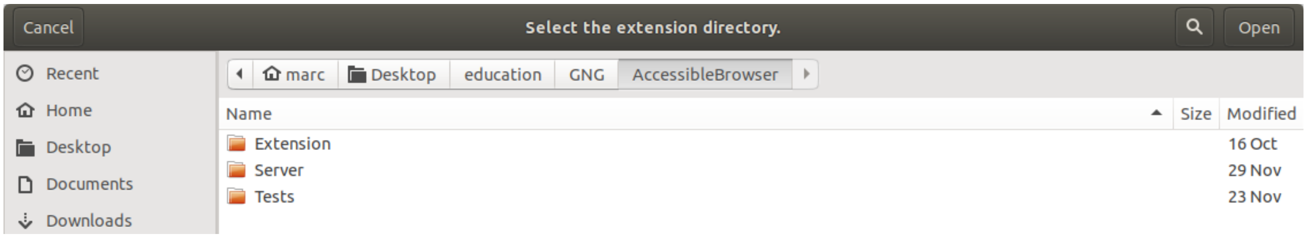
The project was a success in terms of testing the feasibility and utility of a simplified browser. Though the project in the current state would be hard pressed to help users it allowed us to determine what we should have done. If we were to redo this project entirely, we would not have used a chrome extension, despite the portability, the pain of attempting to bend within the expectations of a chrome extension with what we were attempting to do was very inconvenient and caused many delays during development. We would have chosen a framework like Electron, which offers the same portability of chrome extensions while also giving more freedom with what and how we can program.

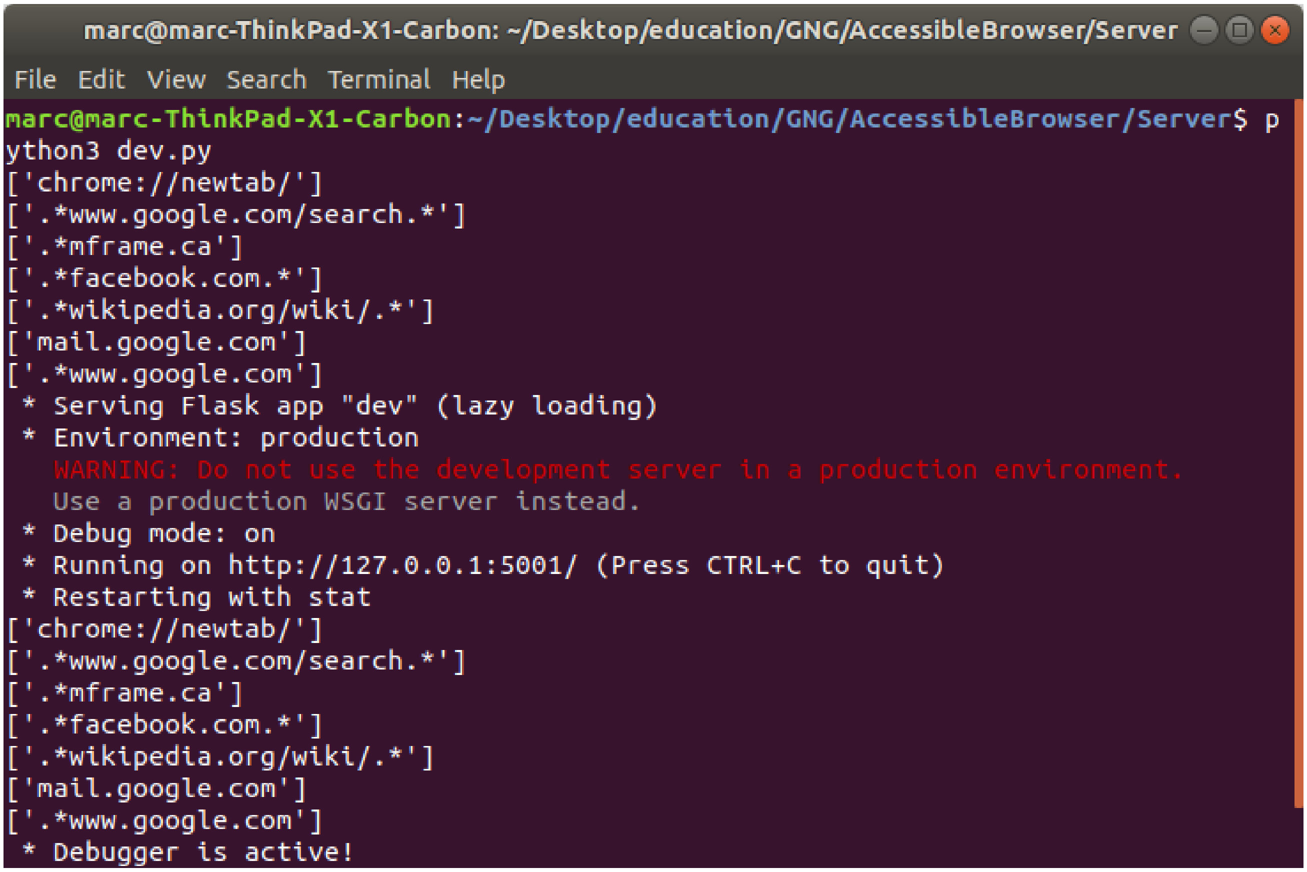
User Manual

Installation of development environment

   chrome://extensions
Go to this URL
Figure 7.1 : where to load

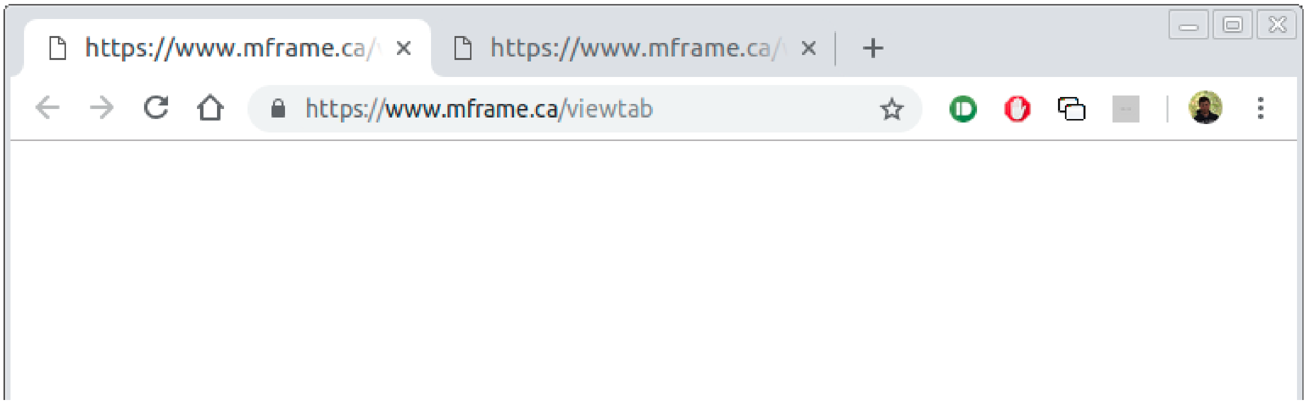

Load unpacked extension

 Cancel Select the extension directory. Open Recent Home Desktop Documents Downloads marc Desktop education GNG AccessibleBrowser Name Size Modified Extension 16 Oct Server 29 Nov Tests 23 Nov
Select Extension and click open

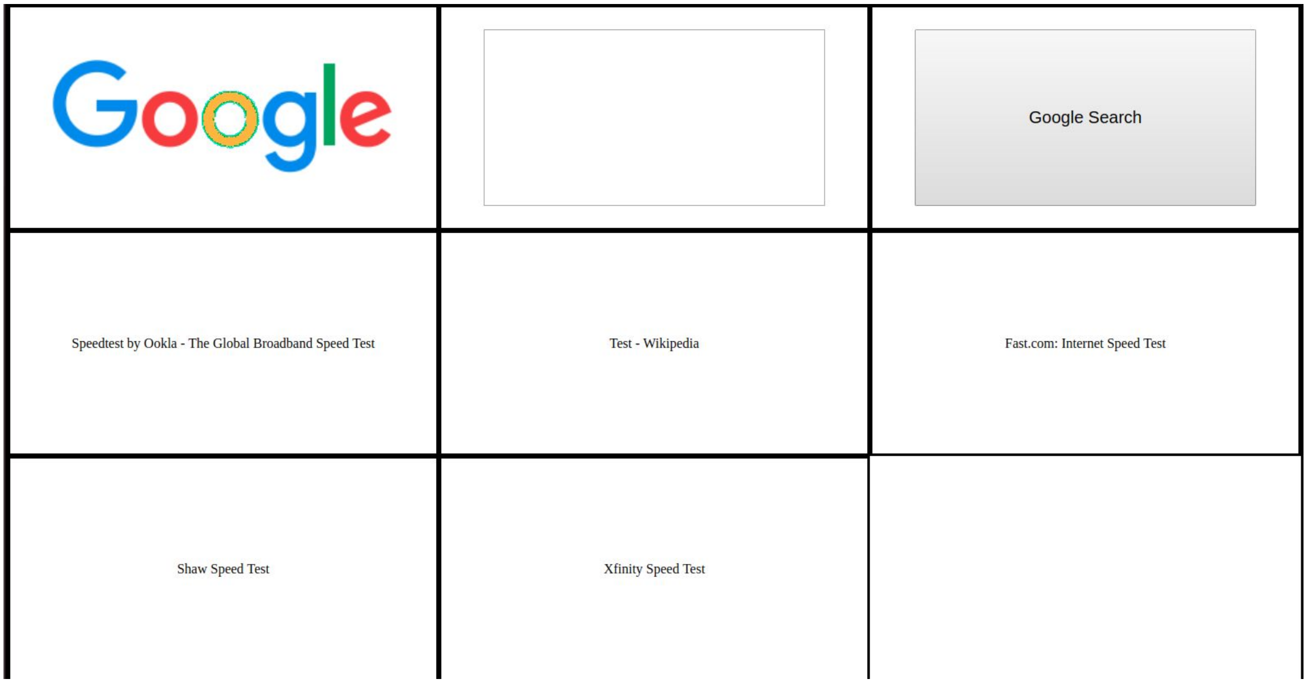

marc@marc-ThinkPad-X1-Carbon: ~/Desktop/education/GNG/AccessibleBrowser/Server
File
Edit
View
Search
Terminal
Help

```
marc@marc-ThinkPad-X1-Carbon:~/Desktop/education/GNG/AccessibleBrowser/Server$ p
ython3 dev.py
['chrome://newtab/']
['.*www.google.com/search.*']
['.*mframe.ca']
['.*facebook.com.*']
['.*wikipedia.org/wiki/.*']
['mail.google.com']
['.*www.google.com']
 * Serving Flask app "dev" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5001/ (Press CTRL+C to quit)
 * Restarting with stat
['chrome://newtab/']
['.*www.google.com/search.*']
['.*mframe.ca']
['.*facebook.com.*']
['.*wikipedia.org/wiki/.*']
['mail.google.com']
['.*www.google.com']
 * Debugger is active!
```

Run the flask development server with dev.py, this is seperate from the server that is used for the website



Worktab is used for adjusting the URL



Google		Google Search
Speedtest by Ookla - The Global Broadband Speed Test	Test - Wikipedia	Fast.com: Internet Speed Test
Shaw Speed Test	Xfinity Speed Test	

This is what shows up within the view tab when google is typed into the worktab. From here the app is completely usable for the currently developed websites. The demod example is being able to google something, view the results and then read the wikipedia article related to it

* note: to initialize the and make usable for someone with any handicap preventing regular use of a web browser, the worktab must first be operated, this was to be changed by the addition of a landing page but was never completed

Development of a config file

<table>
<tr><th>Key word</th><th></th><th>description</th></tr>
<tr><td>objectId</td><td></td><td>Automatically generated based off of the index in the objects list 0,1,2...</td></tr>
<tr><td>name</td><td>x</td><td>Name of cell</td></tr>
<tr><td>name</td><td>x</td><td>Name of object (MUST BE UNIQUE)</td></tr>
<tr><td>type</td><td></td><td>Type of object,<table><tr><td>grid</td><td>Only takes up one grid</td></tr><tr><td>multigrid</td><td>When it spans a finite number of scrolls</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>infinitescroll</td><td>Multigrid that is derived from an infinite scrolling website (facebook feed, twitter feed)</td></tr></table></td></tr>
<tr><td>gridtype</td><td></td><td>Defines the type<table><tr><td>img</td><td>The grid will be interacted with as an image</td></tr><tr><td>link</td><td>The grid will be interacted with as a link</td></tr><tr><td>...</td><td></td></tr></table></td></tr>
<tr><td>objects</td><td></td><td>type<table><tr><td>img</td><td><img> </img></td></tr><tr><td>input:text</td><td>Input tag with type text</td></tr><tr><td>...</td><td></td></tr></table>kwargs.type<table><tr><td>value</td><td>Value is given
“value” : “The Text to Appear”
“Value” : “https://example.com/image”</td></tr><tr><td>worktab</td><td>Derived from the worktab
Type: cssselector
Desc: selector utilized by jquery
Ex.
“value” : {“type” : “cssselector“, selector” : “p.ParagrahClass”, “attr” : “text”}
“value” : {“type” : “cssselector“, “selector” : “img#logo”, “attr” : “src”}
“value” : {“type” : “cssselector“, “selector” : “a.homelink”, “attr” : “href”}</td></tr></table></td></tr>
<tr><td>col</td><td>x</td><td>Which column to be placed in 0,1,2…
Will default to the first available cell</td></tr>
<tr><td>row</td><td>x</td><td>Which row to be placed in 0,1,2…
Will default to the first available cell</td></tr>
<tr><td>selectable</td><td></td><td>Default true;
False can’t be selected used for images</td></tr>
<tr><td>clicked</td><td>x</td><td>type<table><tr><td>form</td><td>Can be used for input of forms to apis</td></tr><tr><td>link</td><td>Changes the webpage to something specified by value</td></tr></table></td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>
<table>
<tr><td>...</td><td></td></tr>
</table>
Funccconfig
Value
<table>
<tr>
<td>type</td>
<td>Denotes the type
<table>
<tr><td>value</td><td>Stirng value</td></tr>
<tr><td>cssselector</td><td>Selects the value from css

This has some sibling fields that must accompany it including
<table>
<tr><td>selector</td><td>What CSSselector to use</td></tr>
<tr><td>parent</td><td>Whether to select n parents up.</td></tr>
<tr><td>attr</td><td>href, val...</td></tr>
</table>
</td></tr>
</table>
</td>
</tr>
<tr><td></td><td></td></tr>
</table>
</td>
</tr>
</table>