T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

WEB FILTER EXTENSION

SENA NUR ULUKAYA

SUPERVISOR
DR. GÖKHAN KAYA

GEBZE
2022

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

WEB FILTER EXTENSION

SENA NUR ULUKAYA

SUPERVISOR
DR. GÖKHAN KAYA

2022
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 31/08/2021 by the following jury.

JURY

Member (Supervisor)	:	Dr. Gökhan Kaya
Member	:	Dr. Yakup Genç

ABSTRACT

The main contribution of this project is to provide a solution for unwanted content on the internet by providing a user-friendly and efficient filtering tool in the form of a Chrome extension. The extension hides the items that contain specific keywords according to user input and supports different web pages with different structures.

Keywords: chrome extension, web filtering, content filtering.

SUMMARY

This project presents a Chrome extension that aims to improve the browsing experience by filtering unwanted content on webpages. The extension is designed to filter list structures on webpages according to user input, enabling users to more easily find the information they are looking for. These keywords are saved to extension's storage, allowing the filter choice to be remembered on different webpages.

The extension's user interface is simple and allows the user to enter keywords to filter, as well as options to clear all filters or remove individual filters.

One of the unique features of this extension is its ability to recognize custom attributes on webpages where the list structures do not have repeated class names or specific IDs. This is achieved through a "learning mode" where the user can select a list element and save the attributes associated with that element. The extension then uses these attributes to recognize the structure on that domain in the future, making the filtering process more efficient and supporting as many of web pages as possible.

In addition, the extension uses a dynamic filtering process that continues to filter even when new content is loaded on the page, ensuring that the browsing experience is not interrupted.

Overall, this project presents a comprehensive solution to the problem of unwanted content on the internet by providing a user-friendly and efficient filtering tool in the form of a Chrome extension.

Keywords: chrome extension, web filtering, content filtering.

ACKNOWLEDGEMENT

I would like to express my thanks to all of my instructors who have provided me with knowledge and guidance throughout my academic journey. Their lectures and teachings have been instrumental in helping me to develop the skills and understanding necessary to complete this project. I would like to thanks to my supervisor, Dr. Gokhan Kaya, for all the help and support they gave me throughout my project. They helped me with their knowledge and experience, and had time to answer my questions. I would also like to express my gratitude to my family and friends for their love and support. Their encouragement and understanding have been a constant source of motivation and inspiration to me.

Sena Nur Ulukaya

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation		**Explanation**
API	:	Application Programming Interface
DOM	:	Document Object Model
SPA	:	Single Page Application
UI	:	User Interface
CSP	:	Content Security Policy

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The main focus of this project is the development of a Chrome extension that allows users to filter out unwanted content on web pages.

The extension is designed to identify and hide elements on a web page that contain a specific keyword or phrase entered by the user. Additionally, the extension has a special feature, which is a learning mode. This mode enables the user to select and save custom attributes of a list element to recognize the structure on the web page for future filtering.

This project addresses the need to filter unwanted content on dynamic and custom-structured web pages. With the growing trend of dynamic and custom-structured content on web pages, current filtering methods become less effective. This research aims to provide a solution that addresses these challenges by allowing users to filter out unwanted content on any web page, regardless of its structure.

The project covers the development and implementation of the Chrome extension, as well as an analysis of the most frequently visited web pages to understand the common practices and unique attributes used on these websites. This research does not cover browser extension development for other web browsers, neither it analyzes other web browsers.

The limitations of this research include the fact that the extension is only compatible with Google Chrome, it's not able to filter out all types of unwanted content such as malware or phishing. Additionally, the extension's performance may differ on web pages with a large amount of content or complex structures.

1.1. Motivation

It can be difficult to separate the relevant from the irrelevant with the increasing amount of information available on the internet. Users often find themselves scrolling through pages of unwanted content, wasting time and energy trying to find what they're looking for. This project is motivated by the need to improve the efficiency and effectiveness of the internet browsing experience. By providing users with a tool that allows them to filter unwanted content, this project aims to save users time and energy, and ultimately improve their overall browsing experience.

1.2. Background Research

1.2.1. Chrome Extension Architecture

There are a variety of web browsers available, each with their own extension development ecosystem. However, for this research, the focus is on Google Chrome extensions. This choice was made for several reasons: Firstly, Chrome is one of the most widely used web browsers, with a large user base and a strong market share. This means that the research has the potential to reach a large audience and have a significant impact. Secondly, Chrome provides a robust and well-documented extension development API, which allows for a wide range of functionality and flexibility in the development of extensions. Additionally, Chrome's popularity among developers and its strong developer community makes it a favorable choice to develop and test the extension. Lastly, Chrome is compatible with a wide range of operating systems and devices, which makes it a versatile platform for the extension.

Google Chrome extensions are small software programs that can customize and improve the functionality of the Google Chrome web browser. These extensions are built using HTML, CSS, and JavaScript, and they can interact with the browser through an API (Application Programming Interface) provided by the browser.

The architecture of a Chrome extension is composed of three main components: content scripts, background scripts, and pop-ups. Content scripts are the scripts that run on the web page and can interact with the DOM to manipulate the web page's content. Background scripts are the scripts that run in the background of the browser and can interact with the browser's API. Pop-ups are the user interface that can be accessed through the browser's extension toolbar or a pop-up window. Each component has its own role in the extension's functionality and they work together to provide an efficient and user-friendly experience.

1.2.2. DOM

The Document Object Model (DOM) is a crucial aspect of web development, as it represents the structure of a web page and allows for the manipulation of its elements. The DOM is a tree-like representation of a web page, where each element on the page is represented as a node in the tree. These nodes can be accessed and manipulated through JavaScript, which allows for the dynamic modification of the content and layout of a web page. In the context of Chrome extension development, the DOM is an essential tool for interacting with web pages. Content scripts, which are scripts that run on the web page, can access and manipulate the elements of a web page. This allows Chrome extensions to add or remove content, change styles, or respond to user input on a web page.

1.2.3. Analyzing Different Web Pages

In order to understand the most common practices used in website development, a thorough analysis of the most frequently visited web pages was conducted. The purpose of this analysis was to understand the common structures and attributes used on these websites, in order to develop a solution to filter unwanted content on these web pages.

The analysis revealed that the majority of websites have dynamic content, with similar list structures. Many of these websites use common class names or IDs to identify their list elements, but some websites have their own unique attributes on their list elements. This understanding is essential for developing a solution to filter unwanted content on these dynamic and custom-structured web pages.

1.2.4. Single Page Applications

Single Page Applications (SPAs) have become increasingly popular in recent years due to their ability to provide a smooth and seamless user experience. SPAs are web pages that load a single HTML page and dynamically update the content as the user interacts with the page, without requiring a full page reload. This can make it difficult for the extension to identify and filter the list structures as the lists are dynamically generated.

To overcome this, the extension can be supported with a Mutation Observer. Mutation Observers are a JavaScript API that allows developers to monitor changes in the Document Object Model (DOM) in real-time. They are an efficient and performant way to detect and respond to changes made to the DOM, such as the addition or removal of elements, changes to the text content, or changes to the attributes of elements. Mutation Observers work by creating an instance of the MutationObserver object and passing in a callback function that is executed when changes to the DOM are detected. The callback function is passed a MutationRecord object, which contains information about the type of change that occurred, the element that was affected, and any previous and/or new values of the element.

1.2.5. Similar Projects

1.2.5.1. Adblock

[1] It's a popular ad-blocker browser extensions that allow users to block unwanted content such as ads, trackers, and other forms of unwanted content on web pages. They use filter lists that users can subscribe to in order to block unwanted content. These filter lists are created and maintained by a community of users, and they allow users to customize which types of content they want to block.

AdBlock uses a set of rules, called ”filter lists,” to identify and block ads on web pages. These filter lists are created and maintained by a community of users, and typically include a combination of CSS selectors and JavaScript code to target specific elements on a web page that are associated with ads. AdBlock also uses machine learning algorithms to detect and block ads on web pages. The algorithm is trained to recognize patterns in web pages that are commonly associated with ads, such as certain CSS styles, image dimensions and text. AdBlock also uses browser-specific APIs that allow it to block ads directly at the browser level, bypassing the need to rely on filter lists or other external sources.

1.2.5.2. Stylus

[2] This is a browser extension that allows users to customize the appearance of web pages by hiding certain elements, such as ads or headers. It allows users to create and share custom styles, or ”userstyles” as they are called, that can be applied to web pages to hide specific elements based on their CSS class or ID.

2. METHOD

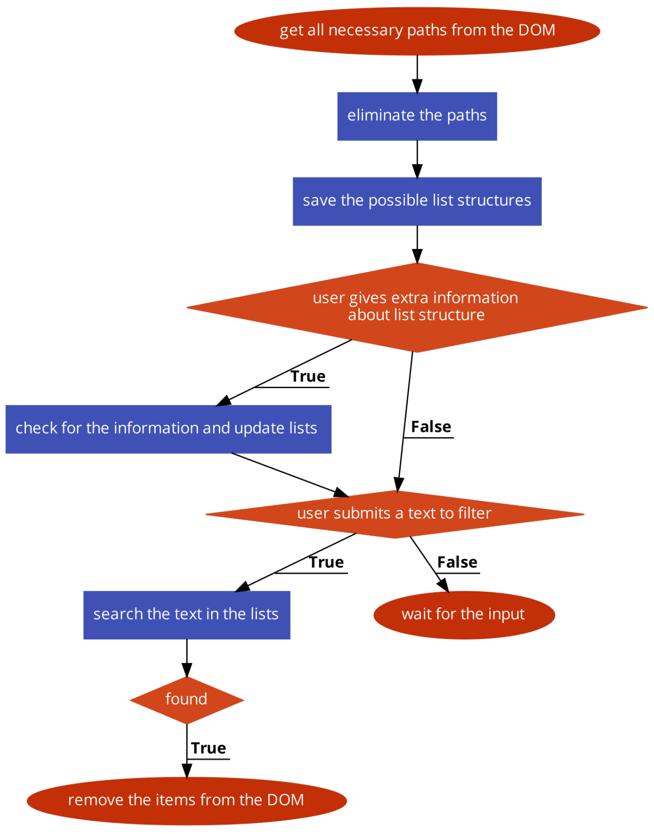

get all necessary paths from the DOM
eliminate the paths
save the possible list structures
user gives extra information about list structure
True
check for the information and update lists
False
user submits a text to filter
True
False
search the text in the lists
wait for the input
found
True
remove the items from the DOM

Figure 2.1: General Flow of the Extension Algorithm

This chapter of describes the general flow and implementation of the chrome extension developed for filtering list structures on web pages. The chapter begins by providing an overview of the user interface and the various features of the extension. It then goes into detail on the process of identifying list structures on web pages, including the algorithm and techniques used. The chapter also covers the implementation of the filtering feature, including how elements are removed and added to the page based on user input. Lastly, the chapter describes the implementation of the learning mode, which allows the extension to recognize custom attributes used by some websites.

2.1. User Interface and Features

The design of the extension is simple and user-friendly. Users can enter the word or sentence they want to filter, and these words will be saved to the storage as well as

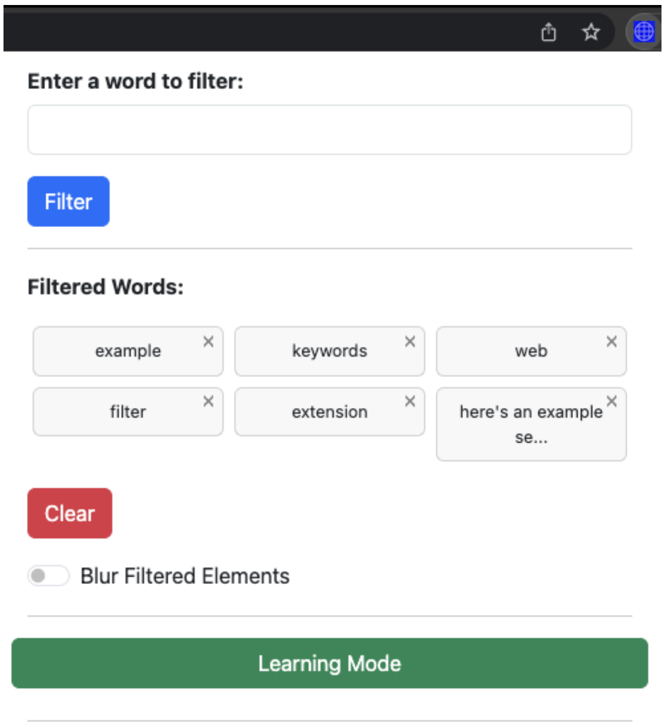


Figure 2.2: User Interface of the Algorithm

displayed in the ”Filtered Words” section of the extension. If the sentence is long, it will display partially by default. But users can hover over the element and display the whole sentence if they want. Users can remove these words one by one by clicking the little cross sign on the top-right, or clear them all at once with the ”Clear” button.

There’s an option to ”Blur Filtered Elements”. If this option is selected, the unwanted content will be blurred out on the page. Users can see that there are contents that are hidden but cannot display them.

The ”Learning Mode” button activates the learning mode which will be discussed in detail in this chapter. Overall, the general flow and design of the extension is easy to understand, and simple to use.

2.2. General Flow

The extension begins by setting up an Mutation Observer that listens for changes in the web page’s DOM and this allows the extension to detect new lists as they are loaded. When a change is detected, the extension checks the storage to find if any options are saved before in the current domain. If any are found, it uses them as the starting point for filtering the page. If not, it starts by traversing the DOM.

Next, the extension uses a traversal algorithm to identify all unique paths to the list elements found in the previous step. After identifying the unique list structures, the extension applies a process of elimination to remove any paths that correspond to elements that are not lists.

Finally, once the lists have been identified and filtered, the extension provides the user with a simple interface to remove or blur elements from the page. The extension also allows the user to add back any previously removed elements.

2.3. Identifying List Structures

2.3.1. Traversing the DOM

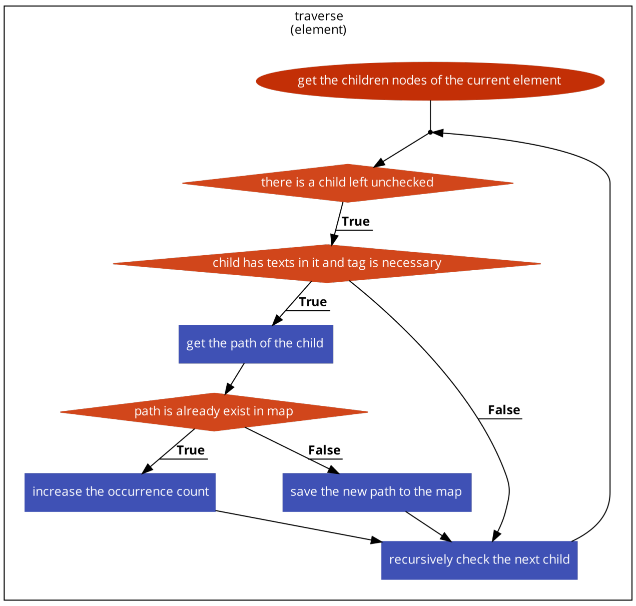

traverse
(element)
get the children nodes of the current element
there is a child left unchecked
True
child has texts in it and tag is necessary
True
get the path of the child
False
path is already exist in map
True
False
increase the occurrence count
save the new path to the map
recursively check the next child

Figure 2.3: Traverse Algorithm Flow

The first step in the algorithm is to traverse the DOM of the web page. The function starts by getting all the same-tagged children of the input element, and then loops through each of them. Within the loop, the function checks if the tag of the current child element is in a predefined list of skipped tags, and if so, it continues to the next child element. If the tag is not in the list of skipped tags, the function then checks if the child element has a 'role' attribute and if so, it checks if that role is in a predefined list of skipped roles. If the role is in the list of skipped roles, it continues to the next child element. If the current child element passes both checks, the function then checks if the child element has a text content.

These checks are added to improve the performance by not wasting time on elements like nav, header, footer, etc. that are not the list structure that is wanted. Also, the extension is interested in filtering the text content so, an element without text are not considered.

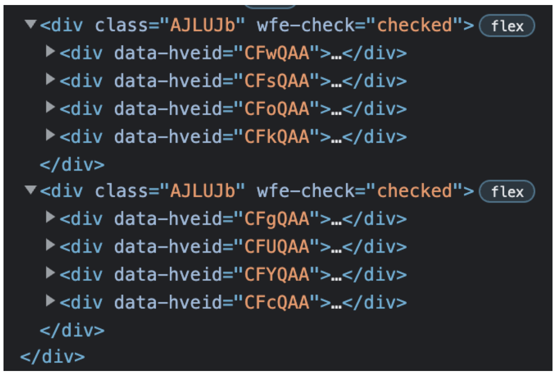
Then the path of that element is added to a map of all possible paths found on the web page. Map elements' key values are the path and values are the occurrence. The goal is to find repeated elements, therefore occurrence value is stored.

After this, the ”traverse” function is called again with the current child element as the input to recursively traverse its children as well.

The paths are stored as string. The function starts by checking if the input element has a parent node and if so, it appends the tag name and class name of the element to the parent's path. If the element does not have a class name, it checks if it has an id, and if so, it appends the tag name and id of the element to the parent's path. If the element has neither a class name nor an id, it appends only the tag name of the element to the parent's path ad identifiers of the elements.

2.3.2. Eliminating the Saved Paths

Any paths from the map that found less than 4, is removed. The reason to set a ”4” as a boundary is this:



```
▼<div class="AJLUJb" wfe-check="checked"> flex
  ▶<div data-hveid="CFwQAA">…</div>
  ▶<div data-hveid="CFsQAA">…</div>
  ▶<div data-hveid="CFoQAA">…</div>
  ▶<div data-hveid="CFkQAA">…</div>
  </div>
▼<div class="AJLUJb" wfe-check="checked"> flex
  ▶<div data-hveid="CFgQAA">…</div>
  ▶<div data-hveid="CFUQAA">…</div>
  ▶<div data-hveid="CFYQAA">…</div>
  ▶<div data-hveid="CFcQAA">…</div>
  </div>
</div>
```

Figure 2.4: Structure of the Google Search Related Searchs Sections

As seen in Figure 2.5, the search list items of ”Google Search's Related Searches” are within two identical elements, which can be detected as a list structure by the algorithm. To address this issue, a boundary is added to adjust the minimum number of items required for it to be considered a list. This is due to the algorithm's reliance on identifying specific list structures and filtering elements based on their text content. In the case of ”Related Searches,” there is a list within a list, which the algorithm will detect the outer structure as a valid list structure if there's no boundary. Also, adding a an option for user to set the boundary by their needs can be nice additional feature.

2.3.3. Saving the Possible Lists

This function is responsible for identifying and saving potential list elements. The function iterates over the paths Map, which contains all the paths of the elements on the web page that have been identified as potential list elements. Then, with querying the path, all the elements with that path is retrieved and added to the list with their siblings.

Also by a helper function, it checks if the parent element of a given path has already been identified as a list. If the parent is marked, it means that it has been identified as a list and the current path being checked is a child element of that list. This function is used to prevent the same list from being identified multiple times and to ensure that only unique lists are saved. Elements are marked by a unique attribute.

After iterating over all the paths in the paths Map, the "possible list elements Set" contains all the list elements that were identified by the algorithm.

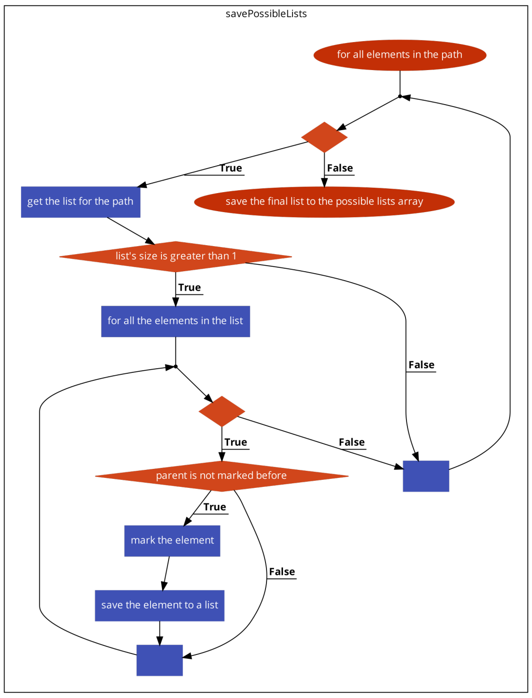

savePossibleLists
for all elements in the path
True
False
get the list for the path
save the final list to the possible lists array
list's size is greater than 1
True
for all the elements in the list
False
True
False
parent is not marked before
True
mark the element
False
save the element to a list

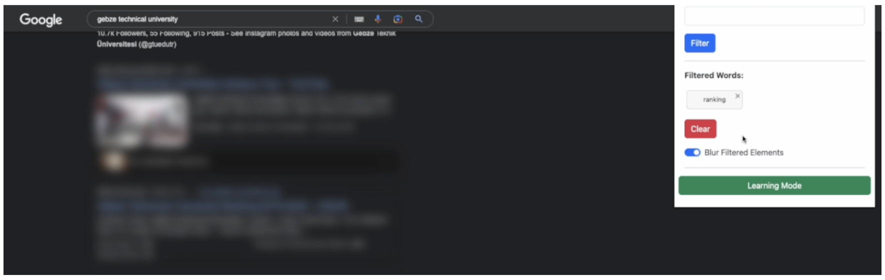
Figure 2.5: Saving Lists Algorithm Flow

2.4. Hiding/Displaying Elements

For hiding elements, different approaches are tested. Initially, started with actually removing elements from the DOM and keeping the removing element in a storage.

But this method was using unnecessary extension storage. Other issue is, removing an element from the DOM is not a good approach in terms of maintaining the structure and integrity of the web page. Some web pages have certain security mechanisms in place that prevent the removal of elements from their DOM, making this approach not feasible for all web pages. Additionally, removing elements from the DOM can cause issues with the layout and functionality of the web page, as the removal of an element may affect the styling and behavior of other elements on the page. As a result, it was decided to implement two alternative approaches, blurring and hiding.

Blurring elements involves applying a blur effect on the element, making it unreadable but still visible on the page. Hiding elements, on the other hand, involves setting the ”display” property of the element to ”none”, making it invisible on the page. Both of these methods are more flexible and efficient compared to removing elements from the DOM and they provide similar functionality of hiding unwanted content from the user.



10.7K Followers, 55 Following, 915 Posts - See Instagram photos and videos from Gebze Teknik Üniversitesi (@gtuedutr)

Figure 2.6: Blurred Elements on Google

2.4.1. Hiding Elements

This part is responsible for hiding elements from the web page that contain a certain text. The function first retrieves the value of the ”blur” option from local storage. If the option is set to true, the function blurs the element. If the option is set to false, the function hides the element by setting its ”display” CSS property to ”none”. Additionally, the function adds an unique attribute ”wfe-check” with a value of ”hidden” to the element to mark it as removed to make it easier to add back. The function also checks if all the child elements of the element’s parent are hidden. If so, the parent element is also hidden and an attribute ”wfe-check” with a value of ”hiddenParent” is added to it.

In some cases, when all the children of the same parent element are hidden, the style of the parent element can be affected. This can cause issues in the layout of the website and make the user experience less satisfactory. To address this issue, the extension has been designed to hide the parent element as well when all of its children are hidden. This ensures that the style of the website is preserved, and the layout remains intact.

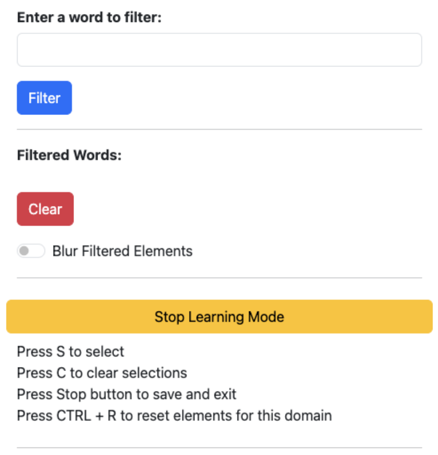
Also after user refreshes the page, the hidden element should remain. Therefore another function is responsible for hiding elements that contain words that have been previously filtered by the user. The function retrieves the list of filtered words from local storage and calls the function for each word in the list until one of them is exist in the element. In this case there's no need to control until the end of the list.

2.4.2. Displaying Elements Again

This part is responsible for adding back the previously hidden elements by comparing them to filtered words. When there's no filtered word exist in the element, it can be displayed again. The function first retrieves the value of the ”blur” option from local storage. If the option is set to true, the function un-blurs the element. If the option is set to false, the function makes the element visible by changing its ”display” CSS property. Additionally, the function removes the ”wfe-check” attribute from the element. The function also checks if the parent element of the current element has ”wfe-check” attribute with a value ”hiddenParent” if so it removes the attribute and makes the parent element visible.

2.5. Learning Mode

The ”learning mode” feature allows users to manually identify elements on a web page that they wish to filter, instead of relying on the algorithm to automatically identify elements. The mode is activated by the user through a button in the extension’s UI, and once activated, the user can hover over elements and select them by pressing a key ”S”. User can also clear their choices by pressing a key ”C”. Additionally, they can reset the storage with ”CTRL + R”.



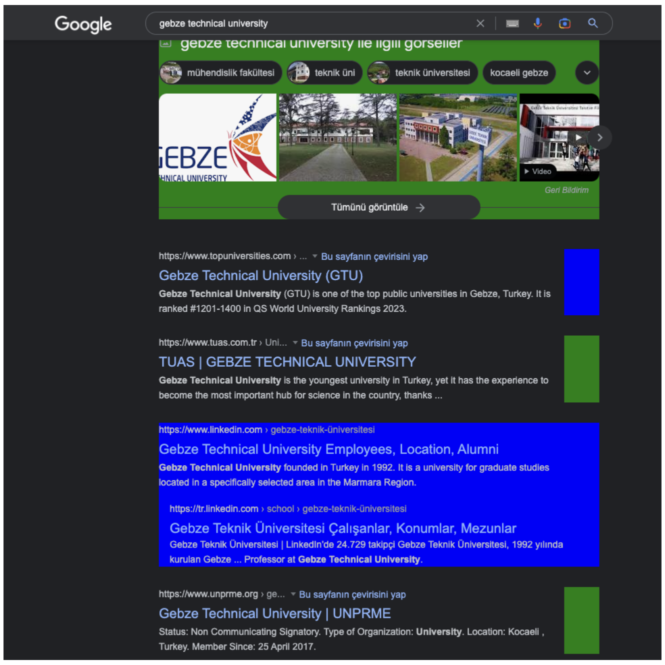
Press S to select
Press C to clear selections
Press Stop button to save and exit
Press CTRL + R to reset elements for this domain

Figure 2.7: Learning Mode is Activated

When an element is selected in this way, the script first finds the parent element of the selected element that contains multiple siblings with the same tag name. This is done to ensure that the parent element, which is likely to contain a list of items, is selected instead of just a single item within the list. The parent element and its child elements that have the same tag name as the selected element are then highlighted and given an attribute ”wfe-check” with a value of ”learned” to mark them as selected by the user.

Once the user is finished selecting elements, they can deactivate the learning mode by clicking on the stop button in the extension’s UI. When the mode is deactivated, the script saves the selected elements’ attributes in the extension’s storage with the current domain, remembers and uses them in the future to filter elements on that domain.

This method of manually selecting elements is important for the extension because it allows the user to customize their experience by only hiding or blurring elements that they find distracting or unwanted. Without this feature, the extension would have to rely on a predefined set of elements to hide or blur, which may not be suitable for all users or web page. Also, it extends extension’s range and modularity. This feature allows the user to manually identify elements on the web page that the base algorithm



https://www.topuniversities.com › ... Bu sayfanın çevirisini yap
Gebze Technical University (GTU)
Gebze Technical University (GTU) is one of the top public universities in Gebze, Turkey. It is ranked #1201-1400 in QS World University Rankings 2023.

https://www.tuas.com.tr › Uni... Bu sayfanın çevirisini yap
TUAS | GEBZE TECHNICAL UNIVERSITY
Gebze Technical University is the youngest university in Turkey, yet it has the experience to become the most important hub for science in the country, thanks ...

https://www.linkedin.com › gebze-teknik-üniversitesi
Gebze Technical University Employees, Location, Alumni
Gebze Technical University founded in Turkey in 1992. It is a university for graduate studies located in a specifically selected area in the Marmara Region.

https://tr.linkedin.com › school › gebze-teknik-üniversitesi
Gebze Teknik Üniversitesi Çalışanlar, Konumlar, Mezunlar
Gebze Teknik Üniversitesi | LinkedIn'de 24.729 takipçi Gebze Teknik Üniversitesi, 1992 yılında kurulan Gebze ... Professor at **Gebze Technical University**.

https://www.unprme.org › ge... Bu sayfanın çevirisini yap
Gebze Technical University | UNPRME
Status: Non Communicating Signatory. Type of Organization: **University**. Location: Kocaeli , Turkey. Member Since: 25 April 2017.

Figure 2.8: Highlighted List Elements on Google

may have missed. For example, some web pages may not use specific class names or ids, making it difficult for the algorithm to identify certain elements as lists. The learning mode feature allows the user to manually select these elements and add them to the extension's list of learned elements.

Additionally, this feature also provides the ability to identify unique attributes of the learned elements. This is important as it allows the extension to be more modular and flexible. By identifying unique attributes of elements, the extension is able to filter out specific elements. This means that the extension can be used on a wider range of web pages and can more effectively filter out unwanted content.

Furthermore, the use of this feature increases the accuracy of the extension and the overall user experience. The extension will be able to filter out more unwanted content and provide a more personalized experience for the user. This feature also allows the user to have more control over the extension and the content they see on the web pages they visit.

2.6. Extension Architecture

The interaction between the background, popup, content script allows the extension to function correctly by enabling the user to control the extension, manipulate the web page's DOM, and persist the data. It also allows the extension to be modular, the different functionality can be separated into different files and can be easily maintained.

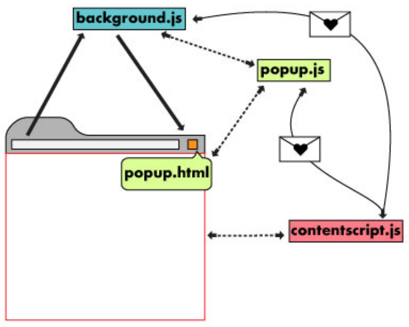


Figure 2.9: Chrome Extension Interactions[3]

2.6.1. Content Script

Content script contains all the functionality discussed above. Also, when a message is received from the popup script it will execute the appropriate action such as hiding elements from the DOM, adding back elements that were hidden, and enabling the learning mode. The content script also listens for changes in the storage, for example, when the user changes the blur setting, it will update the web page accordingly.

2.6.2. Background Script

The background script in the extension is responsible for controlling the extension's behavior and communication with other parts of the extension such as the content script. One of its primary functions is to enable the extension according to the user's input and execute the content script according to.

2.6.3. Popup

Popup script is responsible for handling user interactions with the extension's popup menu. When a user clicks on the extension icon, the popup menu is displayed and the popup file is executed. The popup menu allows the user to filter out certain words or phrases from the web page, clear the filtered words, and enable the learning mode. When the user performs one of these actions, popup script sends a message to the content script to execute the corresponding action.

For the UI of the popup, HTML with bootstrap is used.

2.6.4. Storage

The storage is used to persist data across different web pages and browser sessions. The extension uses the storage to save the filtered words, the learned elements, and the user's settings. The content script interacts with the storage to retrieve the filtered words and user's settings, and to save the learned elements.

3. RESULTS

In this chapter, the results of the extension's performance will be discussed and evaluated against a set of criteria established for the project's success. The criteria that have been considered include:

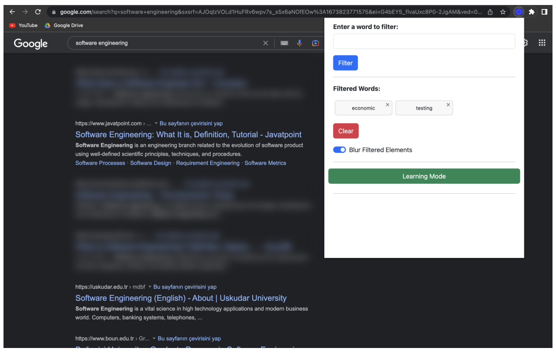
- Extension functionality on a wide range of websites with varying list structures, with a goal of achieving compatibility on 90% of these websites.
- Modularity of the extension, allowing for easy customization and modification of its features.
- Fast response time, not exceeding more than 80% of the normal response time of the website.
- Offline functionality, allowing the extension to work without the need for servers.

3.1. Results of Different Websites

In this section, the results from different types of web pages will be discussed. To provide a comprehensive analysis, the extension was initially tested on some of the most popular websites worldwide. [4].

3.1.1. Google Search

The extension performed well on Google Search, effectively identifying lists without the need for the learning mode and successfully hiding and retrieving them.



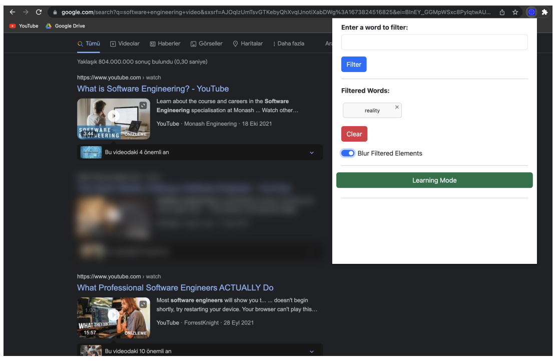
https://www.javatpoint.com › ... Bu sayfanın çevirisini yap
Software Engineering: What It is, Definition, Tutorial - Javatpoint
Software Engineering is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures.
Software Processes · Software Design · Requirement Engineering · Software Metrics

https://uskudar.edu.tr › mdbf Bu sayfanın çevirisini yap
Software Engineering (English) - About | Uskudar University
Software Engineering is a vital science in high technology applications and modern business world. Computers, banking systems, telephones, ...

https://www.boun.edu.tr › Gr... Bu sayfanın çevirisini yap

Figure 3.1: Filtered Results on Google Search

Additionally, it was able to identify various types of lists, such as search results, image results, and video results on the main page.


google.com/search?q=software+engineering+video&sxsrf=AJOqlzUmTsvGTKebyQhXvqlJnotiXabDWg%3A1673824516825&ei=BlnEY_GGMpWSxc8PylqtwAU...

Yaklaşık 804.000.000 sonuç bulundu (0,30 saniye)

https://www.youtube.com › watch
What is Software Engineering? - YouTube
Learn about the course and careers in the **Software Engineering** specialisation at Monash ... Watch other...
YouTube · Monash Engineering · 18 Eki 2021
Bu videodaki 4 önemli an

https://www.youtube.com › watch
What Professional Software Engineers ACTUALLY Do
Most **software engineers** will show you t... ... doesn't begin shortly, try restarting your device. Your browser can't play this...
YouTube · ForrestKnight · 28 Eyl 2021
Bu videodaki 10 önemli an

Enter a word to filter:
Filtered Words:
reality
Blur Filtered Elements

Figure 3.2: Filtered Video Results on Google Search

The extension performed well on various tabs of the Google Search including Search, Images, Videos, Shopping, Books, The News and Finance. The base algorithm was able to effectively identify and hide lists on these tabs.

Overall extension demonstrates a high degree of effectiveness and flexibility in identifying and hiding lists on the Google Search page.

3.1.2. Twitter

The extension is able to continuously detect and filter tweets as the user scrolls through them with only the base algorithm.

3.1.3. Facebook

Facebook posts are detected and filtered with base algorithm successfully.

However, it should be noted that Facebook has implemented strict security measures to prevent third-party extensions from interacting with their website. One of these measures is the use of a Content Security Policy (CSP) which blocks attempts to frame the website with another origin.

It can potentially cause issues when working with larger amounts of filtered words and list elements. However, in the tests conducted, this issue did not present any problems.

3.1.4. Instagram

Instagram, as an image-based and mobile-based platform, is not typically accessed through a web browser by users. However, users have the option to use the extension on the Instagram website if they wish. The extension is able to automatically detect posts and comments, making it easy for users to filter their feed according to specific keywords or phrases. This allows them to filter their feed based on the captions of the posts they are viewing.

3.1.5. Youtube

It can be observed that the extension is unable to detect lists on YouTube, as the website uses the same class name throughout the page. This makes it difficult for the extension to differentiate between different types of lists on the page, even if the attribute is saved. To accurately detect lists on YouTube, the extension would need to take into account additional factors, such as the context of the list or the layout of the website. Overall, YouTube's use of consistent class names presents a unique challenge for the extension's list detection capabilities.

Table 3.1: Results of Different Websites

Website	Base Algorithm	Learning Mode	Filtered Lists - Comments
Google Search	+	+	Detailed comments on 3.1.1
Twitter	+	+	Detailed comments on 3.1.2
Facebook	+	+	Detailed comments on 3.1.3
Instagram	+	+	Detailed comments on 3.1.4
Youtube	-	-	Detailed comments on 3.1.5
Yandex	+	+	Search results
Yahoo	+	+	Search results
Yahoo News	+	+	News
Whatsapp	+	+	Chat messages
Wikipedia	+	+	Search and page contents
Amazon	+	+	Products
Live	+	+	Emails
Yahoo JP	+	+	News
Netflix	+	+	Movies and series
Tiktok	+	+	Videos
Reddit	-	+	Posts
Linkedin	-	+	Posts
Mail.Ru	+	+	News, articles
Naver	+	+	News, articles
Bing	+	+	Search results
Discord	+	+	Chat messages
Twitch	-	+	Videos
Pinterest	-	+	Photos
Globo	+	+	News
eBay	+	+	Products
Quora	+	+	Posts
The NY Times	+	+	News
CNN	+	+	News

Table 3.2: Results of Different Websites

Website	Base Algorithm	Learning Mode	Filtered Lists - Comments
Stack Overflow	+	+	Search and QA
Medium	+	+	Articles
Github	+	+	Search and feed
Tumblr	+	+	Posts
Hepsiburada	-	+	Products
Hürriyet	+	+	News
Mynet	+	+	News
Habertürk	+	+	News
Memurlar.net	+	+	News
Ekşi Sözlük	+	+	Entries
N11	+	+	Products
Onedio	+	+	News, articles
Akakçe	+	+	Products
Nefis Yemek Tarifleri	+	+	Recipes
GTU	+	+	Search and page contents

Based on the results presented in the tables, it can be observed that the extension has a success rate of 97.67% when considering both the base algorithm and the learning mode. This indicates that out of the 43 websites that were tested, 42 were successfully filtered by the extension.

Furthermore, it can be noted that the extension has a success rate of 86.04% when only considering the base algorithm. This implies that out of the 43 websites tested, 37 were successfully filtered by the extension using only the base algorithm.

One of the websites, YouTube, tested was not able to be detected and filtered by the extension because of the design structure of the website.

In conclusion, the results suggest that the extension is able to successfully filter the majority of websites tested, with the exception of a few that have specific limitations or blocking mechanisms in place. The ability to use the learning mode as a fallback allows for more flexibility in filtering different types of websites, further increasing the overall success rate.

3.2. Response Time

In order to ensure that the extension did not significantly slow down the browsing experience for the user, the response time of the extension was measured while detecting the list elements and filtering the page with different-sized keyword lists. The results of these measurements show that the extension was able to achieve fast response times, with the majority of websites tested not exceeding more than 80% of their normal response time, which is around 1 second, in average.

Table 3.3: Response Time Tests

# of Filtered Words	Worst Case Response Time
1	5.2 ms
10	10.3 ms
100	0.28 s
300	0.40 s
500	0.65 s
700	0.85 s
1000	1.05 s
5000	3.87 s

To measure the response time, ”now” function from Performance API is used, which can provide insight into the extension’s performance and responsiveness. In terms of response time, the extension is able to identify and filter lists quickly. The results showed that the extension had a minimal impact on the response time of the website, with the majority of websites tested.

The worst-case scenario for this system is when none of the keywords, except for the last one, not appear in any of the list elements. In this case, the extension must check each list element until the end of the keywords list. The results of the response time tests show that, even under this worst-case scenario, the response time remains under one second, even when filtering thousands of words. However, it is important to note that this worst-case scenario is unlikely to occur in most real-world situations.

The ability to maintain fast response times is an important aspect of the extension, as it ensures that the browsing experience for the user is not negatively impacted. The extension’s ability to accomplish this is a testament to its efficient design and implementation, allowing users to efficiently filter unwanted elements on a web page without sacrificing speed.

It is worth noting that some websites, particularly those with constant DOM mutations, may experience slightly longer load times while using the extension. However, in most cases, the change in load time is not noticeable.

3.3. Modularity

The extension's modularity is a key factor in its design and implementation. The goal of the extension is to be able to filter out unwanted lists on web pages, regardless of the structure or layout of the website. To accomplish this, the extension is designed to be modular, meaning that it can be easily adapted to different types of websites, with different list structures.

The extension achieves modularity by utilizing a combination of base algorithm and learning mode. For websites that have unique structures or use different attributes for their lists, the extension is able to adapt by utilizing the learning mode. By using the learning mode, the extension is able to learn the specific attributes of these unique lists, and filter them out accordingly.

Overall, the results of this project suggest that the extension is able to successfully filter out unwanted lists on web pages, with a high success rate, when considering both base algorithm and learning mode. This success is largely due to the extension's modular design, which allows it to adapt to different types of websites and user preferences.

3.4. Offline

In order to ensure that the extension can function offline, it has been designed to rely solely on the use of the storage API. This allows the extension to store and retrieve information locally, without the need for any external servers.

The storage API is used to store information such as filtered words, learned elements, and settings. This information is then used by the extension to filter elements on the page. This approach allows the extension to provide a seamless and uninterrupted user experience, regardless of the user's internet connection.

Additionally, the extension's offline functionality also helps to protect user privacy, as all data is stored locally and is not transmitted to any external servers. This ensures that user data remains secure and confidential, and is not subject to any potential privacy breaches on external servers.

Overall, the offline functionality of the extension is a key feature that allows it to be versatile and adaptable to different environments and situations, making it a reliable and efficient tool for filtering web content.

4. CONCLUSIONS

In conclusion, the extension developed in this project aims to provide a solution for users to hide or blur unwanted elements on web pages with various techniques and algorithms to identify lists and filtering them. It offers a ”learning mode” feature, which empowers users to manually identify and save lists of elements on a web page that they would like to hide. It uses mutation observers to monitor changes in the web page's DOM and automatically hide or blur newly added elements that match the filtered words. This approach makes the extension useful for a wide range of web pages, including the most popular sites worldwide.

The extension makes use of the Chrome extension API to interact with the browser and the web page, and uses Chrome's storage to store the filtered words and learned elements. The extension's modular design allows for easy modification and the addition of new features in the future.

Furthermore, the extension provides a fast response time and offline functionality, enabling the extension to work without the need for servers. Its use of mutation observers and local storage allows for efficient and seamless operation.

Overall, the extension provides a useful tool for users to customize their browsing experience and enhance their focus and productivity while browsing the web. With the extension, users can easily hide or blur unwanted elements on web pages, and the learning mode feature allows for even more flexibility and customization.

BIBLIOGRAPHY

[1] "Adblock." (2023), [Online]. Available: https://getadblock.com/.

[2] "Stylus." (2023), [Online]. Available: https://chrome.google.com/webstore/detail/stylus/clngdbkpkpeebahjckkjfobafhncgmne.

[3] "Chrome developers , architecture overview." (2023), [Online]. Available: https://developer.chrome.com/docs/extensions/mv3/architecture-overview/.

[4] "List of most visited websites." (2023), [Online]. Available: https://en.wikipedia.org/wiki/List_of_most_visited_websites (visited on 01/10/2023).