SSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 13, Issue 2, pp: (80-91), Month: April - June 2025, Available at: www.researchpublish.com

SocialScreen: A Chromium Plugin That Filters Social Media Posts And Web Text Content Using Fuzzy Matching

Roumel John S. Nidoy[1], Jonh Zaldy Catalan[2] , Anthony Ross D. Arayata[3], Elmer C. Matel[4], Elizabeth S. Nsubuga[5], Jerian R. Peren[6]

Lyceum of the Philippines University, Cavite, Philippines

DOI: https://doi.org/10.5281/zenodo.15773604

Published Date: 30-June-2025

Abstract: **The overwhelming volume of unfiltered and potentially harmful content on social media and websites poses significant challenges to digital well-being, especially for users seeking a more controlled and distraction-free online experience. In response to this issue, this study presents the design, development, and evaluation of SocialScreen: a Chromium browser extension that filters social media posts and web text content using a Fuzzy Matching algorithm. The primary objective of the project is to provide users with an effective tool for managing online content by allowing both preset and user-defined datasets for filtering. The extension was developed using Visual Studio Code, JavaScript, Node.js, Webpack, HTML, and CSS, and was guided by the Agile methodology throughout the software development lifecycle. Functional and performance testing were conducted to ensure the reliability, efficiency, and responsiveness of the system. Evaluation sessions and structured surveys gathered feedback from both end-users and IT professionals, focusing on usability and software quality, based on ISO/IEC 25010 standards.**

Results indicate that SocialScreen demonstrated strong performance in maintainability, with a modular design and ease of updates, supported by comprehensive documentation. However, some limitations were identified in the area of portability, particularly with respect to accessibility compliance and full compatibility across all Chromium-based browsers. The implementation of the Fuzzy Matching algorithm proved to be a valuable enhancement, enabling flexible and accurate filtering of textual content. The Agile methodology facilitated iterative improvements, ensuring a user-centric development approach. Overall, evaluation findings confirmed the tool's functional effectiveness and acceptable quality performance.

Keywords: **Chromium Plugin, Fuzzy Matching Algorithm, Content Filtering, Social Media, JavaScript.**

I. INTRODUCTION

Social media refers to websites and online tools that facilitate user interaction, enabling individuals to share information, opinions, and interests [1][2]. Popular platforms such as Facebook, Twitter, and Instagram have become integral to daily life, serving as sources of entertainment, communication, and information. According to the Digital 2022 report by Hootsuite and We Are Social, 82.4% of the Philippine population were active social media users at the beginning of 2022—one of the highest usage rates globally. While the internet offers a wealth of useful content, it is also saturated with harmful, misleading, and inappropriate material. This becomes a concern, especially for users—particularly millennials (ages 23–38)—who spend considerable time online and often seek a safer, more positive, and stress-free browsing experience. Studies have shown that many users prefer to avoid controversial, explicit, or misleading content, desiring a more personalized and controlled digital environment [3]. Despite the utility of social media for promoting media literacy and social connection, its content moderation systems often fall short. Platforms have built-in keyword filters, but these are typically limited to exact matches, making them ineffective against variations or misspelled versions of problematic content. Furthermore, these features are often buried in settings, rendering them inaccessible to users who lack advanced digital literacy. This lack of intuitive, customizable, and intelligent filtering tools exposes users—including children and adolescents—to inappropriate material, such as graphic content or misinformation [4][5]. As noted by Siddiqui (2021), researcher Abi Perry highlighted the dual nature of targeted content: while helpful educational advertisements may appear, so too might content related to self-harm or other age-inappropriate material. To address this growing concern, there is a need for a more advanced, user-friendly solution that empowers individuals to filter digital content more effectively. One promising approach is the use of browser extensions—small software modules that enhance browser functionality. Extensions built for Chromium-based browsers can be programmed using JavaScript, HTML, and CSS, and can offer customized features such as content filtering. By incorporating fuzzy matching algorithms, these tools can extend beyond exact keyword matches to detect variations and related terms, providing a more robust filtering capability [6]. This study responds to the need for an intelligent, customizable, and accessible content filtering tool by developing and evaluating SocialScreen, a Chromium browser extension that filters social media posts and web text content using fuzzy matching. The project aims to empower users to manage their digital environment more effectively, addressing limitations in current platform-based filtering systems, and contributing to the broader goal of improving online safety and digital well-being.

A. Objectives of the Study

The study aimed to develop a Chromium plugin—**SocialScreen**—designed to filter social media posts and web content using fuzzy matching techniques. The specific objectives of the study were:

1. to identify and address Chromium-specific limitations that could impact the implementation of SocialScreen's features;
2. to conduct functionality and performance testing on SocialScreen and gather expert feedback from IT professionals; and
3. to assess the plugin's acceptability based on the ISO/IEC 25010 standard for software quality.

II. METHODOLOGY

A. System Architecture

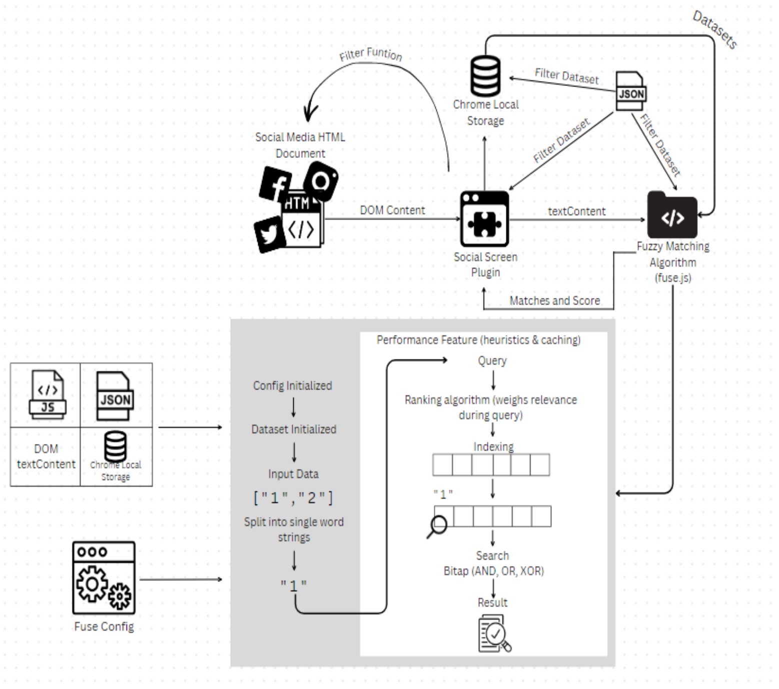


Fig. 1 System Architecture of SocialScreen

The system architecture shown in Fig.1 follows SocialScreen's approach to content filtering. First, it initializes when the page finishes loading. Then, it prepares the datasets by importing preset filters from JSON files and organizing them into arrays. The selected options are stored in an array object. Next, the system retrieves the text content of the HTML document and filters the content based on matches with the filter datasets based on the selected options, the datasets are then stored in an array object. It also obtains the website URL to determine specific configurations. The main function will then initialize a Fuse.js object and take the text content of the HTML document as an argument along with the options object which is a configuration for Fuse.js. The main function is responsible for the filtering process and includes specialized functions for different websites. It will match the text content of the HTML document against the selected filter array. The matched text is then hidden or modified using styling techniques based on the website's requirements, the parent container of the matching text is targeted in this process which cleanly removes it from the web page. The system continuously monitors the HTML document for changes and updates the displayed content accordingly. Users can also add custom filters, which are stored in an array and can be managed through a popup window. The custom filters, along with the selected filters, are saved in the Chrome local storage for persistent storage and retrieval.

In summary, this system architecture provides a practical approach to content filtering. It handles the initialization, dataset preparation, text content retrieval, filtering process, website-specific configurations, and dynamic updates. It also allows users to customize their filtering experience with custom filters and ensures their preferences are saved for future use.

B. Fundamental Algorithm Used: Fuzzy Matching

Fuzzy string matching involves identifying strings that partially match a given string, rather than requiring an exact match. This technique is particularly useful in scenarios where users misspell words or input partial words, such as in search engines. The algorithm used in fuzzy string matching goes beyond comparing two strings for exact equivalence [7].

The Bitap algorithm, also known as the Bitap fuzzy string searching algorithm or the Shift-Or method, is one of the most widely used fuzzy-string matching algorithms [8][9][10]. The Bitap method effectively searches for a specified pattern inside a text while accounting for flaws or mismatches in the pattern using a bitwise comparison technique. It uses a bitwise mask that changes dynamically depending on whether there are matching characters present. The algorithm can handle changes like insertions, deletions, replacements, and transpositions by using bitwise operations, making it a flexible method for approximating string matching.

Fuzzy Matching Algorithm Implementation

1. *Initialize when the DOM Content finished loading.*
2. *Initialize datasets*
   a. Import the preset filters from the JSON files
   b. Push the datasets to their respective array
   c. Get the selected options that will be stored in an array object
   d. Initialize an empty array to store the datasets for algorithm use.
3. *Get the text content of the HTML document.*
4. *Push the datasets based on the selected options array.*
5. *Get the website URL.*
6. *Run the main function for filtering. This contains other functions specialized for each website.*
7. *Run the function based on the website URL.*
   a. Initialize the options object that contains the configuration for Fuse.
   b. Run the function for the specified website.
   c. Initialize a Set object to be used as a container.
   d. Get the text content of the document from the document query selector.
   e. Push the text content to the Set object.
   f. Initialize a Fuse instance

- Convert the Set object to an array and pass it to the Fuse instance as an argument. This initializes the fuzzy search on the array object.
- Pass the options object to the Fuse instance as an argument to modify the configuration to be used in fuzzy search.

g. Split the array that contains the dataset of filters.

h. Match the filters against the text content of the document.

i. The fuzzy matching will score the matches.

j. Select a text that is within the score threshold set in the configuration.

k. Find the parent element of matched text using the query selector.

l. Apply the style to hid the content. Social media websites hide the entire posts.

8. *Observe the changes in the HTML document.*

9. *Run the filter function when there are changes.*

10. *Run the filter function when the user adds another filter option.*

11. *If the field is filled when the user clicks the add button, push the contents of the text filled in the custom filter array.*

12. *If the user clicks the show custom filter button, get the string objects inside the custom filter array object*

- *Add the string objects to a list HTML document in the popup.*

13. *If the user clicks the "x" button next to the words in the list, remove the word from the custom filter array object*

14. *Save the custom filter array object in the chrome local storage.*

15. *Save the selected filter array object in the Chrome local storage.*

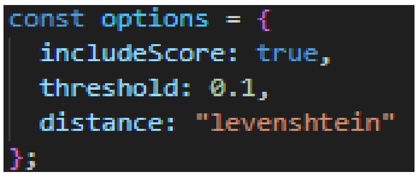
C. Fuzzy Matching Algorithm Simulation

Text Content: "This is a sample text with various words and variations."

Text Content Array: ["This", "is", "a", "sample", "text", "with", "various", "words", "and", "variations"]

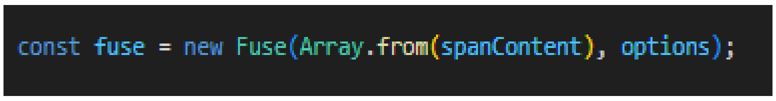
Filter Keywords: ["sample", "variations"]

Configuration: threshold = 0.1, distance = levenshtein



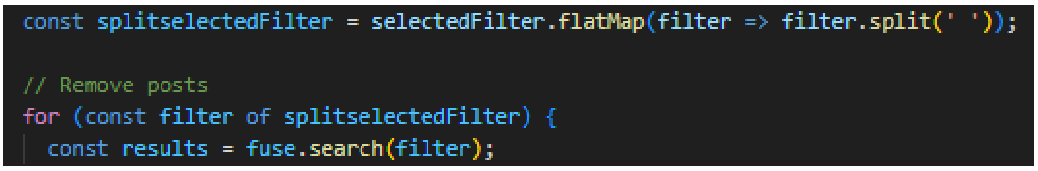
```
const options = {
  includeScore: true,
  threshold: 0.1,
  distance: "levenshtein"
};
```

Fig. 2 Fuse.js Options Initialization



```
const fuse = new Fuse(Array.from(spanContent), options);
```

Fig. 3 Fuse.js Object Initialization



```
const splitselectedFilter = selectedFilter.flatMap(filter => filter.split(' '));

// Remove posts
for (const filter of splitselectedFilter) {
  const results = fuse.search(filter);
```

Fig. 4 Filter Dataset Splitting and Iteration

Step 1: As shown in Figure 2, options are initialized with Fuse.js configuration values that are optimal for SocialScreen. In Figure 3, a new fuse object is initialized and takes the web page text content and options as arguments.

Step 2: Fuse.js performs fuzzy matching for each word in the web page text content against the filter keywords dataset. The lower the score the higher the similarity. As shown in Figure 4, "splitselectedFilter" and "selectedFilter" are objects that contain the filter keywords dataset. Fuse.js iterates through the "splitselectedFilter to find matches.

The score is computed using the following formula:

$$score = \frac{(number\ of\ matches)}{(pattern\ length)} - (penalty * (number\ of\ mismatches))$$

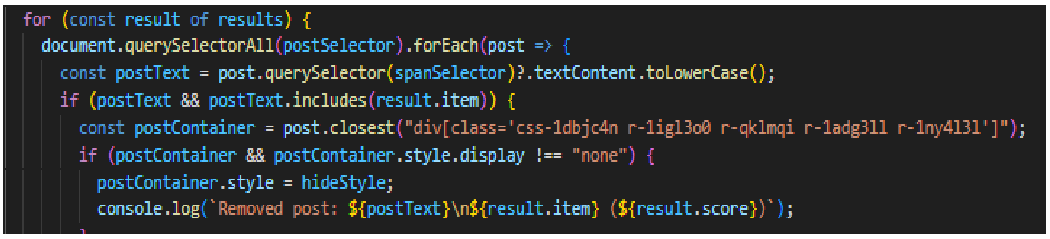
where the "Number of matches" is the number of characters in the string pattern that match text. "Pattern length" is the number of characters in the string pattern that do not match the text. "Number of mismatches" is the number of characters in the string pattern that do not match the text. Lastly, the "penalty" is a configured penalty value, this ranges from 0.5 to 1 and reduces the score based on the number of mismatches.

TABLE 1: SCORING SYSTEM FOR FUSE.JS

Text Content	**Similarity Score ("sample")**	**Similarity Score ("variation")**
This	1.00	1.00
is	1.00	1.00
a	1.00	1.00
sample	0.00	0.78
text	0.89	1.00
with	0.88	1.00
various	0.89	1.00
words	1.00	0.89
and	1.00	1.00
variations	0.89	1.00

TABLE 1 presents a comparative analysis of how Fuse.js, a JavaScript-based fuzzy search library, evaluates the similarity of individual words in a given text against two different search keywords: "sample" and "variation." The Text Content column lists the words from a sentence or phrase, while the corresponding Similarity Score columns indicate how closely each word matches the keywords, based on a scoring range from 0.00 to 1.00, with 1.00 representing a perfect or exact match. When compared to the keyword "sample," the words "This," "is," "a," "words," and "and" all received a similarity score of 1.00, which might suggest that Fuse.js considered them as exact matches or very high-probability matches, possibly due to default threshold settings or tokenization behavior. Interestingly, the word "sample" itself received a score of 0.00, which may seem counterintuitive. This could be a result of how Fuse.js handles exact keyword matches in fuzzy mode or could point to the influence of specific configurations in the scoring algorithm. Other words such as "text," "with," "various," and "variations" received scores in the range of 0.88 to 0.89, indicating moderate similarity to "sample." In contrast, when the keyword "variation" was used as the basis for scoring, all words received significantly high scores, mostly 1.00, with the exception of "sample" (0.78) and "words" (0.89). The word "variations," being the plural form of "variation," appropriately scored 1.00, confirming a strong similarity. This pattern suggests that the keyword "variation" is more flexibly matched to the set of words than "sample," potentially because of its broader linguistic root and form variations.

Overall, the table illustrates how Fuse.js assigns similarity scores based on fuzzy logic and highlights how keyword choice affects the resulting scores. It also reflects the sensitivity of the Fuse.js algorithm to word structure, spelling variations, and possibly default settings like threshold, distance, and token matching.



```
for (const result of results) {
  document.querySelectorAll(postSelector).forEach(post => {
    const postText = post.querySelector(spanSelector)?.textContent.toLowerCase();
    if (postText && postText.includes(result.item)) {
      const postContainer = post.closest("div[class='css-1dbjc4n r-1igl3o0 r-qklmqi r-1adg3ll r-1ny4l3l']");
      if (postContainer && postContainer.style.display !== "none") {
        postContainer.style = hideStyle;
        console.log(`Removed post: ${postText}\n${result.item} (${result.score})`);
```

Fig. 5 Content Filtering

Step 3: As shown in Figure 5, the web extension will search the document for the content that contains words that scored within the score threshold and apply style attribute changes to the parent attribute to hide them.

D. Mathematical Models/Formula Used

Levenshtein Distance

Levenshtein distance, also known as edit distance, is a measure of the similarity between two strings, which is the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into the other. It was first proposed by Vladimir Levenshtein in 1965 (Binary Codes Capable of Correcting Deletions, Insertions and Reversals, nd.). Levenshtein distance is applied in the Fuse.js configuration used in SocialScreen to improve the accuracy for matching.

As shown in equation 1 is the formula for Levenshtein distance algorithm which is used to compare and match words especially in NLP tasks such as machine translation, text summarization, and information retrieval. Formula for the Levenshtein distance:

$$lev(s,t) = \min(lev(s,t-1)+1, lev\ (s-1,t)+1, lev\ (s-1,t-1)+[s\ != t])$$

The Levenshtein distance is a measure used to assess the dissimilarity between two strings, represented by s and t. It involves different computations: lev(s,t) determines the distance between s and t, lev(s, t-1) measures the distance between s and t with the exclusion of the last character in t, and lev(s-1, t) evaluates the distance between s with the omission of its last character and t. Furthermore, [s != t] serves as an indicator that yields 1 when the last characters of s and t are dissimilar, and 0 when they are identical.

E. Simulation of Calculating the Levensthein Distance

The distance matrix will be filled to find the distance between two strings "words" and "this" as the two input words.

TABLE 2. CALCULATION OF LEVENSHTEIN DISTANCE

		w	o	r	d	s
	0	1	2	3	4	5
t	1	1				
e	2					
x	3					
t	4					

As shown in TABLE 2, the first distance to be calculated is between two prefixes of the two words, which are t and w. Because the two prefixes are different, the distance between them is calculated as the minimum of the 3 existing values (0, 1, and 1) + 1. So the distance is min(0,1,1)+1=0+1=1.

TABLE 3. CALCULATION OF LEVENSHTEIN DISTANCE

		w	o	r	d	s
	0	1	2	3	4	5
t	1	1	2	3	4	5
e	2					
x	3					
t	4					

As shown in TABLE 3, by calculating the distance between the first prefix of the word, t and the second prefix for the second word, he. the distance is 2, and for the next prefixes we just add one to get a 3, 4, and 5 distance. Here is the distance matrix after being updated with these distances.

TABLE 4. CALCULATION OF LEVENSHTEIN DISTANCE

		w	o	r	d	s
	0	1	2	3	4	5
t	1	1	2	3	4	5
e	2	2	2			
x	3					
t	4					

As shown in TABLE 4, calculating the distance between e and h, then the 2x2 matrix prefix that will be used is highlighted according to the next figure. Because the prefixes e and w are different, then the resultant distance equals min(1,1,2)+1=1+1=2.

TABLE 5. CALCULATION OF LEVENSHTEIN DISTANCE

		w	o	r	d	s
	0	1	2	3	4	5
t	1	1	2	3	4	5
e	2	2	1	2	3	4
x	3					
t	4					

TABLE 5 shows the matrix being updated by the newly calculated distance. The process will continue until all the cells are filled for the row corresponding to “e”.

TABLE 6. CALCULATION OF LEVENSHTEIN DISTANCE

		w	o	r	d	s
	0	1	2	3	4	5
t	1	1	2	3	4	5
e	2	2	1	2	3	4
x	3					
t	4					

TABLE 6 shows the matrix being updated by the newly calculated distance. The process will continue until all the cells are filled for the row corresponding to “e”.

TABLE 7. CALCULATION OF LEVENSHTEIN DISTANCE

		w	o	r	d	s
	0	1	2	3	4	5
t	1	1	2	3	4	5
e	2	2	2	3	4	5
x	3	3	3	3	4	5
t	4	4	4	4	4	**5**

As shown in TABLE 7, continuing the process from figures 7-10. This is the complete distance matrix that is shown below. The distance between the words "words" and "text", and with a value of 5, which means 5 edits are required to transform text into words.

F. Cosine Similarity

Using Cosine similarity is used to measure the similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. The formula for the Cosine Similarity which will be used in fuzzyset.js is:

$$\cos(A,B) = \frac{A.B}{||A||.||B||}$$

where, A and B refer to the two vectors under comparison, while ||A|| and ||B|| denote the magnitudes or lengths of these vectors. Cosine similarity ranges from -1 to 1. A value of 1 means that the vectors are identical, a value of 0 means that the vectors are orthogonal (at a 90-degree angle) and a value of -1 means that the vectors are completely dissimilar. It is also commonly used in natural language processing, information retrieval, and machine learning to compare text or other data in a vector space. It is particularly useful in high-dimensional spaces, and it is insensitive to the magnitude of the vectors being compared (Alake, 2023).

III. RESULTS

A. Functional Test Results

The functionality test results are derived from the test cases that were implemented. The researchers have defined the scenarios and criteria for evaluating the system. In this section, the researchers will discuss the outcomes and observations from the test execution, specifically focusing on any issues or deviations that were encountered. The purpose of this section is to give a summary of the functionality test results and their significance in terms of the system's performance and adherence to requirements.

TABLE 8: FUNCTIONALITY TESTING RESULT

Test Case ID	Name of Module Function	Test Results
SS-1	Pop up	✓
SS-2	Checkbox (Custom, Custom Filter, and Profanity filter)	✓
SS-2.1	Input Custom Words	✓
SS-2.2	Add Custom words	✓
SS-2.3	Apply Filters	✓
SS-2.4	Show Custom Filters	×
SS-2.5	Remove Custom Filters	✓
Percentage of Functionality		**85.71%**

TABLE 8 presents the result of Functional Testing. Overall evaluation of the functionality test cases is that it has received an 85.71% percentage of functionality, based on the test cases that were executed by the IT Expert. The number SS-2.4 which has failed the test cases has contributed to the lower percentage of functionality, which indicates that there might be some enhancement to improve the functionality of the "Show Custom Filters".

B. Performance Test Results

The performance test results are based on the executed test cases. The researchers have defined the scenarios and criteria for evaluating the system's performance. In this section, the researchers will discuss the outcomes and observations from the test execution, specifically highlighting any issues or variations encountered. The objective is to provide a concise overview of the performance test results and their implications for the system's efficiency and compliance with performance requirements.

TABLE 9: PERFORMANCE TESTING RESULT

Test Case ID	Name of Module Function	Test Results
SS-1	Using the extension on a profile with custom filters	✓
SS-2	Applying Multiple Filters simultaneously	×
SS-3	Enabling and disabling Filters	✓
SS-4	Testing and the performance impact of adding custom words	✓
SS-5	Long periods of continuous usage	✓
Percentage of Functionality		**80%**

TABLE 9 presents the result of Performance Testing. Overall evaluation of the performance test cases is that it has received an 80% percentage of performance, based on the test cases that were executed by the IT Expert. The number SS-3 which has failed the test cases has contributed to the lower percentage of performance, which indicates that there might be some enhancement to improve the functionality of the “Enabling and Disabling filters”.

C. Summary of Respondents Evaluation by ISO IEC 25010 Criteria

Most respondents expressed positive views regarding the system’s features and intended users. According to the table, Functional Suitability achieved the highest weighted mean of 4.47 and 0.66 standard deviations, indicating its "Highly Acceptable" rating. This reflects the system's ability to provide functions that exceed the stated and implied needs of the system. On the other hand, Portability received the lowest weighted mean of 4.18 and 0.83 standard deviations, classified as "Moderately Acceptable."

TABLE 10: SUMMARY OF END USERS EVALUATION BY ISO IEC 25010 CRITERIA

Criteria	Mean	Standard Deviation	Interpretation
Functional Suitability	4.47	0.66	Highly Acceptable
Usability	4.25	0.68	Highly Acceptable
Portability	4.18	0.83	Moderately Acceptable
Reliability	4.43	0.85	Highly Acceptable
Security	4.41	0.80	Highly Acceptable
Maintainability	4.44	0.69	Highly Acceptable
Performance Efficiency	4.35	0.62	Highly Acceptable
Interoperability	4.30	0.79	Highly Acceptable
Grand Mean	**4.35**	**-**	**Highly Acceptable**
Grand Standard Deviation	**-**	**0.59**	**Highly Acceptable**
Functionality Verbal Equivalent		**Excellent**	

TABLE 10 presents a summarized evaluation based on ISO/IEC 25010 criteria, conducted by 60 end users who provided feedback on the system. The table provides the average mean, standard deviation, and the interpretation of the results.

Although the system is deemed satisfactory, it still may require additional considerations or efforts for seamless adaptation to different environments, with a grand mean of 4.35 and a standard deviation of 0.59, the overall evaluation of the developed research study is interpreted as "Highly Acceptable". This suggests that the research study achieved a satisfactory level of performance across the evaluated criteria, as outlined by the ISO IEC 25010 standard.

TABLE 11: SUMMARY OF IT EXPERTS EVALUATION BY ISO IEC 25010 CRITERIA

Criteria	Mean	Standard Deviation	Interpretation
Functional Suitability	4.14	0.89	Moderately Acceptable
Usability	3.64	1.15	Moderately Acceptable
Portability	3.85	1.31	Moderately Acceptable
Reliability	4.00	1.15	Moderately Acceptable
Security	4.14	1.01	Moderately Acceptable
Maintainability	4.14	1.12	Moderately Acceptable
Performance Efficiency	4.05	1.12	Moderately Acceptable
Interoperability	4.00	1.30	Moderately Acceptable
Grand Mean	**3.99**	**-**	**Moderately Acceptable**
Grand Standard Deviation	**-**	**1.04**	**Moderately Acceptable**
Functionality Verbal Equivalent		**Very Good**	

TABLE 11 presents a summary of the evaluation conducted by IT experts based on ISO IEC 25010 Criteria. The table provides the average mean and interpretation of the results obtained from 7 respondents who assessed the system and provided their feedback. Most respondents expressed positive opinions regarding the system's features and intended uses. The table indicates that the criteria Functional Suitability, Security, and Maintainability achieved the highest weighted mean of 4.17 and having a standard deviation of 0.89, 1.01, and 1.12 respectively which all signify a "Moderately Acceptable" level. A high score on Functional Suitability indicates that the web extension's functions meet the specifications required to execute its intended use. This implies that SocialScreen effectively filters social media posts and web text content with the use of fuzzy matching. The high score on Security reflects that the web extension has appropriate measures to protect user data and ensure secure browsing. SocialScreen's process minimizes user data leaks by only getting texts from elements that contain what the users see in the web page. The Maintainability score indicates that the system has been developed with a focus on ensuring that future maintenance tasks can be performed efficiently and effectively. The evaluation conducted by the IT experts provides valuable insights into the strengths and areas of improvement for SocialScreen. These results can serve as an outline for further enhancements and advancements in both SocialScreen and advancements in web extension development, contributing to the field of content filtering.

IV. CONCLUSION

The study successfully achieved its objectives in developing SocialScreen: A Chromium Plugin that Filters Social Media Posts and Web Text Content Using Fuzzy Matching. By implementing a fuzzy matching algorithm within the web extension, the system was able to accurately and efficiently filter social media posts and general web content. This algorithm enabled matching of web page text against a filter dataset, effectively accounting for word variations and misspellings—enhancing the reliability of the content filtering process.SocialScreen also provided users with a high level of customization through its custom filter option, allowing users to add or remove keywords based on their preferences. This feature enabled a personalized browsing experience tailored to individual user needs.The fuzzy matching algorithm was rigorously tested and fine-tuned to ensure both strictness and accuracy, thereby improving the consistency and dependability of SocialScreen's filtering capabilities. Security was also a priority; input sanitization and DOM manipulation measures were integrated to prevent cross-site scripting (XSS), reinforcing the plugin's safety as a web extension. Following a comprehensive review and inspection process by Google's development team, SocialScreen was successfully published in the Chrome Web Store, further validating its security and usability. The extension was developed using Manifest V3, aligning with Chrome's standards for modern Chromium extensions. This ensured compatibility across a broad range of Chromium-based browsers. Throughout the functionality and performance testing phases, no critical system errors were observed, confirming the extension's stable and user-friendly performance. The study, conducted at Lyceum of the Philippines University – Cavite Campus from September 2022 to June 2023, involved evaluation sessions with social media end-users and IT experts to gather feedback and insights. In terms of evaluation, SocialScreen received the highest scores in the Maintainability criterion—4.25 from end-users and 4.17 from IT experts. This indicates that the extension is robust, modular, and easy to support or improve in the future.

However, it scored lower in the Portability criterion—3.97 from end-users and 3.75 from IT experts. These scores suggest a need for improvement in terms of accessibility and compatibility across certain Chromium browsers. Fortunately, due to the modular structure of the extension, these enhancements can be implemented efficiently in future updates.

Overall, the study not only met its development objectives but also contributed meaningfully to improving user experience on social media platforms. Future improvements may focus on expanding system capabilities, ensuring compliance with evolving accessibility standards, and adapting to ongoing changes in Chromium browsers and social media ecosystems.

In conclusion, SocialScreen demonstrated its effectiveness in addressing limitations in social media content filtering. Through the successful integration of a fuzzy matching algorithm, it provided accurate and consistent filtering performance, significantly enhancing the user browsing experience. The study highlighted the potential of fuzzy matching in software development and showcased the value of browser plugins as tools for improving digital environments.

V. RECOMMENDATION

Based on the findings and conclusions of the study, the following recommendations are proposed to enhance the functionality and effectiveness of **SocialScreen**:

1. Adopt a Community-Based Development Approach. It is recommended to adopt a user-centered and community-driven development model. By continuously gathering feedback from users, releasing regular updates, and incorporating user-suggested improvements, the system can evolve to meet the dynamic needs of its user base. This collaborative approach will help maintain the relevance and usability of the extension while fostering a more satisfying and engaging user experience.

2. Introduce Confirmation Dialogs for Critical Actions. To prevent unintended actions and enhance user contr.ol, the system should implement pop-up confirmation dialogs for critical or irreversible functions. These dialogs will prompt users to confirm their intent before executing such actions, thereby improving overall user experience and system reliability.

3. Expand Customization Features. Enhancing customization options will allow users to tailor the extension to their specific needs. Suggested improvements include advanced filter settings, options to import and export custom keyword lists, and controls for adjusting the sensitivity or strictness of the filter. Providing these capabilities will accommodate a wider range of user preferences and improve user satisfaction.

4. Explore New Datasets and Integration with Databases. Future researchers are encouraged to use this project as a foundation for further development by incorporating new and more diverse datasets. Exploring the use of database systems for managing filter datasets may also yield performance improvements and scalability benefits. Additionally, expanding the scope of the dataset to include multilingual content, varying cultural contexts, and more diverse user preferences can significantly enhance the system's accuracy and adaptability.

REFERENCES

[1] Gupta, C., Jain, A., & Joshi, N. (2018). Fuzzy Logic in Natural Language Processing – A Closer View. Procedia Computer Science, 132, 1375–1384. https://doi.org/10.1016/j.procs.2018.05.052

[2] Hartwig, K., & Reuter, C. (2019). AIS Electronic Library (AISeL) - Wirtschaftsinformatik 2019 Proceedings: TrustyTweet: An Indicator-based Browser-Plugin to Assist Users in Dealing with Fake News on Twitter. https://aisel.aisnet.org/wi2019/specialtrack01/papers/5/

[3] Hegde, A., & Bavalatti, A. (2023). What is an API (Application Programming Interface)? ERP Information. https://www.erp-information.com/application-programming-interface.html?expand_article=1

[4] Kabir, M. N., Tayan, O., Alginahi, Y., Hasan, M. M., & Rahman, M. A. (2019, February 1). On the development of a web extension for text authentication on Google Chrome. IEEE Conference Publication | IEEE Xplore. https:// ieeexplore.ieee.org/abstract/document/8679250

[5] Keizer, G. (2020, November 9). Google's Chromium browser explained. Computerworld.https://www. computerworld.com/article/3261009/googles-chromium-browser-explained.html

[6] Meyer, J. (2020, January 2). History of Twitter: Jack Dorsey and The Social Media Giant. TheStreet. https://www. thestreet.com/technology/history-of-twitter-facts-what-s-happening-in-2019-14995056

[7] Krosel, A., Eads, A., & Garcia, R. (2023). What Is a Software Development Methodology? (With 11 Types). Indeed.com Australia. https://au.indeed.com/career-advice/career-development/software-development-methodology.

[8] Dutta, M. (2023). Fuzzy String Matching – A Hands-on Guide. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/fuzzy-string-matching-a-hands-on-guide/

[9] Brave Software. (2023). What are browser extensions, and are they safe? Brave Browser. https://brave.com/learn/what-are-web-browser-extensions/#:~:text=Security%20and%20privacy%20risks%20with%20browser%20extensions,Many%20browser%20extensions&text=Installing%20an%20extension%20introduces%20new,flaws%20that%20hackers%20can%20exploit.

[10] Aho, A. V., & Corasick, M. J. (1975). Efficient string matching. Communications of the ACM, 18(6), 333–340. https://doi.org/10.1145/360825.360855

[11] Dollarhide, M. (2023). Social Media: Definition, Effects, and List of Top Apps. Investopedia. https://www.investopedia.com/terms/s/social-media.asp

[12] Britton, J. (2021). What Is ISO 25010? Perforce Software. https://www.perforce.com/blog/qac/what-is-iso-25010

[13] Delfino, D., & Antonelli, W. (2022). A beginner’s guide to Instagram, the wildly popular photo-sharing app with over a billion users. Business Insider. https://www.businessinsider.com/guides/tech/what-is-instagram-how-to-use-guide

[14] Higgins, M. (2022). Is Google’s Manifest V3 the end of ad blockers? NordVPN. https://nordvpn.com/blog/manifest-v3-ad-blockers/

[15] Krol, A. (2023, February 27). Google Chrome Payment Plugins [Top Popular Browser Extensions]. webbylab. https://webbylab.com/blog/google-chrome-plugins-a-guide-to-popular-browser-extensions/

[16] Persson, M. (2020). JavaScript DOM Manipulation Performance : Comparing Vanilla JavaScript and Leading JavaScript Front-end Frameworks. DIVA. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1436661&dswid=-2465

[17] Siddiqui, A. (2021, August 5). Children Exposed to Inappropriate Content Immediately After Creating Social Media Accounts, Study Reveals. Digital Information World. https://www.digitalinformationworld.com/2021/08/children-exposed-to-inappropriate.html

[18] Padmaja, K., & Hegde, N. (2019, March 1). Twitter sentiment analysis using adaptive neuro-fuzzy inference system with genetic algorithm. IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8819770

[19] Sidekick. (2023). What is a browser plugin? Sidekick. https://www.meetsidekick.com/what-is-a-browser-plugin/

[20] Tripathi, N. (2022, December 26). Likert Scale: Whats, Whys, Hows & Everything to know in 2023. Ramakant Baunthiyal. https://www.surveysensum.com/blog/everything-you-need-to-know-about-the-likert-scale/#:~:text=A%205%2Dpoint%20Likert%20scale%20is%20a%20psychometric%20response%20method,%3B%20(5)%20Strongly%20Agree.

[21] Wallis, J. (2022, July 5). What Actually ARE Website "Plugins & Extensions" And What Do They Do? - WEBO Digital. WEBO Digital. https://webo.digital/blog/what-actually-are-website-plugins-extensions-and-what-do-they-do/