Applied Information Technology And Computer Science Vol. 4 No. 2 (2023) 46-65

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Facebook Spam Filtering Tool using Keyword-Based Technique

Luqman Khir Azman[1], Nurul Azma Abdullah[1]*

[1]Faculty of Computer Science & Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2023.04.02.004
Received 02 November 2023; Accepted 06 November 2023; Available online 30 November 2023

Abstract: Facebook has gained tremendous popularity worldwide, revolutionizing the way people consume news and eliminating the need for physical newspapers. However, this widespread usage has also exposed users to certain vulnerabilities, particularly the pervasive issue of spam. The prevalence of spam on Facebook not only disrupts user experience but also raises concerns about potential device damage. To address this problem, we propose the development of the "Facebook Spam Filtering Tool" - a browser extension designed specifically for Google Chrome. This tool aims to identify and filter spam within the Facebook domain, providing users with a safer and more enjoyable browsing experience. The extension will employ various techniques for spam detection, including content analysis of messages and identification of behavioral patterns commonly associated with spam. By implementing this tool, we can effectively reduce unwanted and annoying communication while safeguarding users against malicious activities.

Keywords: Spam, Malicious Link, Chrome Extension

1. Introduction

Spam is often sent for commercial purposes, promoting products, services, or scams. It can also contain malicious content, such as phishing attempts, malware, or fraudulent schemes. The primary characteristic of spam is that it is sent without the consent or desire of the recipients and is often disruptive, annoying, or potentially harmful [1]. Multiple communication methods, including email, and social media, can be used to send spam messages. According to statistics, a significant portion of all communications sent through social networks are spam [2]. For instance, according to a survey by Proofpoint, social spam increased by 355% in the first half of 2018 [3]. Five new spam accounts were eliminated for every seven new social media accounts (Redmiles et al., 2018). When it comes to social media, Facebook, as one of the largest and most popular platforms, faces its fair share of spam-related challenges. Facebook's massive user base and extensive sharing capabilities make it an attractive target for spammers. To protect its users and maintain a trustworthy environment, Facebook has implemented several measures to combat spam effectively.

*Corresponding author: azma@uthm.edu.my

One of the primary challenges associated with conventional spam tools is the ever-evolving tactics employed by spammers to circumvent spam filters. This persistent endeavor results in an upsurge of spam messages and comments across the Facebook platform, significantly impacting user experience by causing annoyance and consuming valuable time. Moreover, businesses face potential financial implications as employees dedicate substantial amounts of time to managing spam activity instead of engaging in productive work.

The project aims to develop an object-oriented JavaScript extension titled "Facebook Spam Filtering Tool using Keyword-Based Technique." This tool will automatically filter spam content on Facebook, either based on user-defined keywords or a predefined dictionary, enhancing the user experience by reducing manual reporting efforts. The extension will detect and filter spam keywords in messages, feeds, or pictures, providing a user-friendly interface for configuration while ensuring privacy and security. Thorough testing, documentation, release, and ongoing maintenance will be integral to its success, with a feedback-driven approach for continuous improvement.

By introducing this extension, we aim to alleviate the burden on human users, who would otherwise be required to invest significant time and effort in combating spam. This tool streamlines the spam detection process, allowing users to enjoy a cleaner and more secure Facebook environment. It empowers individuals by relieving them of the tedious task of dealing with spam, thereby enabling them to focus their energy on more meaningful and productive endeavors.

A proposed solution had been made to solve the spam issue in Facebook which is the extension will block the spam based on the keywords that had been collected or keywords that have been stored in the database. The Facebook Spam Filtering Tool is a project aimed at simplifying the process of filtering spam on Facebook. Instead of manually reporting spam by clicking "Report as Spam" on each message, feed, or picture, the tool will automatically filter spam keywords. It will also analyze user behavior to identify potentially spammy activity and provide alerts or recommendations to help users avoid scams.

The tool will be developed as an extension for Google Chrome and Microsoft Edge browsers only, utilizing Google API within the Chromium browser architecture. The target users are ordinary people who use Facebook on their PCs. The development will be done using Microsoft Visual Code and Figma for designing the user interface and developing the code and database. The tool's purpose is to enhance user safety on Facebook and prevent them from falling victim to spam or fraudulent activity.

2. Related Work

This section discusses the related word that had been carried out for the tool and the current existing tool. There are many sections in this chapter that will discuss briefly about the certain topics that are related to the project.

2.1 Facebook

There are several reasons why Facebook has become so popular today. One reason is that it provides a platform for people to connect with friends and family, share content, and engage in online communities. It also offers a range of features and tools, such as the ability to create and join groups, events, and pages, that make it easy for people to share and discover new things. Additionally, Facebook has successfully expanded beyond its original social networking function, branching out into areas such as advertising, e-commerce, and virtual reality. This has helped to diversify its revenue streams and make it a dominant player in the tech industry [5].

Facebook has implemented security features to combat spam and ensure user safety. One such feature is CAPTCHA, which distinguishes between humans and bots during account creation or actions susceptible to abuse. By completing visual or audio-based tests, users prove their humanity and prevent automated programs from creating fake accounts or engaging in spam activities [6]. Additionally,

Facebook offers a verification process for profiles and pages to establish their authenticity. Verified accounts are marked with a blue tick badge, reducing impersonation and abuse risks. Users can submit a request with supporting identification or official documents, and Facebook reviews and approves verification on a case-by-case basis [7]. These measures enable users to identify trustworthy profiles and minimize interactions with impersonators or fraudulent entities.

To detect spam in statuses and messages, Facebook maintains a team of human reviewers who manually assess reported content [7]. These reviewers are trained to identify spam, including fake accounts, spammy posts, and suspicious activities. They take appropriate action, such as issuing warnings, disabling accounts, or removing violating content. Human review is particularly valuable when subjective judgment is required, as automated systems may struggle to accurately assess contextual nuances. Facebook continuously trains and updates its team to stay informed about emerging spam techniques and effectively address spam-related issues [8]. Additionally, Facebook encourages users to report spam and suspicious accounts through various reporting options. User reports provide vital information for the review process, allowing Facebook to identify patterns, trends, and new spam tactics [8]. The collective reporting helps refine automated systems and algorithms, improving spam detection and prevention.

2.1 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a machine learning technique that can be used to filter out spam [9]. It analyzes patterns in data to identify spam messages or email which will also be applied in this project to filter out the spam messages or status in Facebook by recognizing and extracting text form images [10]. OCR can be an effective method for filtering spam, as it can adapt and improve over time as it is exposed to more data, allowing it to identify spam more accurately as it evolves [11].

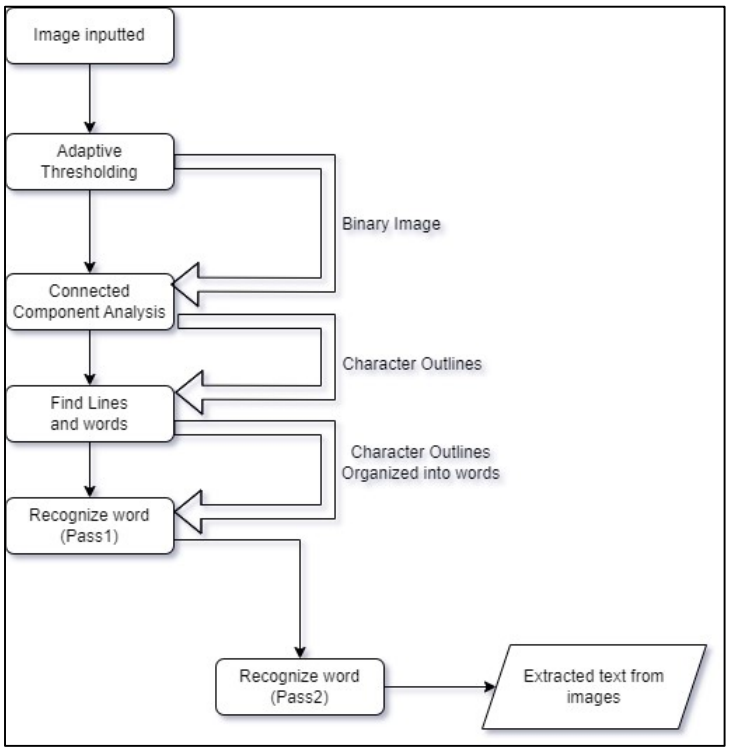

Figure 1: The Process of Tessaract OCR [12]

2.2 Region-Based Convolutional Neural Network (RCNN)

Region Convolutional Neural Network (RCNN) is a type of neural network architecture 10 using a convolutional neural network, and then using those features to detect objects within the image. There

are two stages of process in the RCNN algorithm which are region proposal and classification and bounding box regression. In region proposal stage, the RCNN algorithm will generates a set of region proposals, or candidate regions in the input image that may contain and object of interest [13]. These proposals are generated using a method called selective search, which works by grouping together pixels in the image that are similar in color and texture.

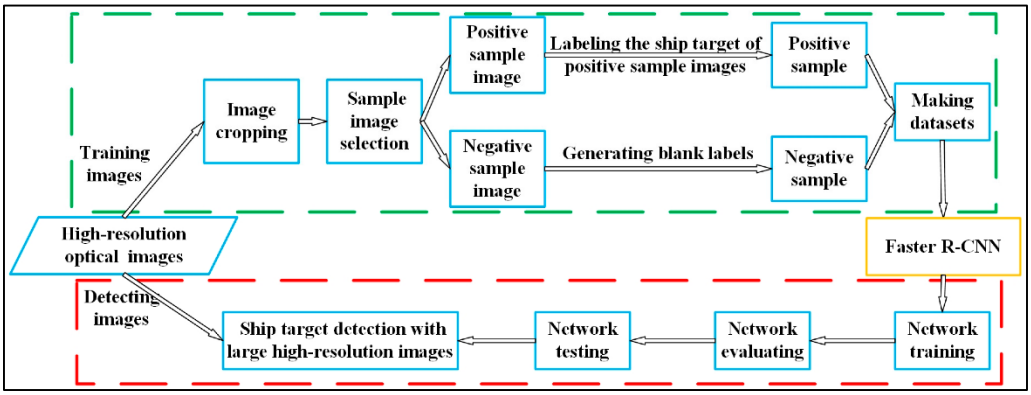

Figure 2: The Process of RCNN Algorithm [14]

The second stage which is classification and bounding box regression, will takes each of the region proposals and classifies them as either containing an object or not, and adjusts the bounding box around the object to be more precise [13]. This is done using a separate fully connected neural network, which takes the features extracted from the region proposal as input and outputs a class label and bounding box coordinates.

2.3 Existing Spam Detection Tools

In the digital world, spam is a pervasive issue that demands effective solutions. Three key spam detection tools are Malena for identifying concealed explicit images, the Rosetta system for recognizing text in photos, and SpamAssassin for filtering unwanted emails, each play a vital role in maintaining online safety and content quality.

Malena is a tool designed to identify adversarial promotional porn pictures (APPI) that utilize distortions to conceal sexual content and make them harder to detect [15]. The tool utilizes two distinct features to detect such images. Firstly, it detects advertising elements like text or QR codes using a scene text detection program. Secondly, it identifies explicit content that is intentionally less obscured to maintain some level of sexual attraction, using ROI-based detection methods [16]. Malena employs a Region-Based Convolutional Neural Network (RCNN) to locate regions of interest (ROIs) within an image and examines each region using four commonly used detectors: Google Cloud Vision API, Baidu AipImageCensor API, Yahoo Open NSFW model, and Clarifai NSFW API. This comprehensive approach allows Malena to accurately identify APPI images [15].

The Rosetta system is an Optical Character Recognition (OCR) system used to identify text in uploaded photographs on Facebook [17]. It consists of text detection and text recognition, with the Fast-RCNN model used for word detection and the Connectionist Temporal Classification (CTC) model used to classify text for each observed box [17]. The system was created to identify spam photos, but its vulnerability to altered text content in images has not been assessed. There is researcher evaluated the robustness of OCR systems using gradient-based attacks by introducing noise or blur to images or inserting hostile text into photos [17]. These experiments also showed that automatic spellchecking is unreliable and that other issues such as OOV (Out-Of-Vocabulary) terms and complexity must be addressed.

SpamAssassin is an open-source software project designed to detect and filter spam emails using a combination of techniques [18]. It analyzes the content, headers, and other components of an email to assign a spam score, comparing it against a threshold to flag emails as spam [18]. Rule-based filtering involves predefined rules that identify patterns and characteristics commonly found in spam emails [19]. Collaborative filtering integrates with systems like Razor and Pyzor, querying distributed databases to check if an email has been reported as spam by other users [19]. DNS-based filtering checks the sending IP address against blacklists of known spam sources. Bayesian filtering utilizes statistical models and user feedback to calculate the probability of an email being spam. While primarily focusing on text-based analysis, SpamAssassin can also consider non-text components like headers, HTML formatting, and metadata during the filtering process [18].

2.4 Comparison Proposed Tool with Existing Tool

Table 1 summarizes various systems and their techniques/approaches, algorithms, and the items they detect. Malena focuses on malicious explicit content detection using APPI detection, scene text detection, ROI-based detection, and animated picture processing with algorithms like RCNN and image filtering. Rosetta is a large-scale system for text detection and recognition, employing robustness testing, OCR system, and text recognition techniques using Fast-RCNN, CTC, and gradient-based attacks. Apache Spam Assassins utilizes rule-based, collaborative, DNS-based, and Bayesian filtering techniques along with OCR and Naïve Bayes algorithm for email spam detection. The proposed system employs OCR and text recognition algorithms for text filtering purposes.

Table 1: Comparison Table of Existing System with Facebook Spam Filtering Tool

Features / System	Technique/Approach	Algorithm	Item Detected
Malena: a Malicious Explicit Content Analyzer [15]	- APPI Detection - Scene Text Detection - ROI-based Detection - Animated Picture Processing	- Region-Based Convolutional Neural Network (RCNN). - Libmagic - Python Imaging Library	- Image Filtering
Rosetta: Large Scale System for Text Detection and Recognition in Images [17]	- Robustness Testing - OCR System - Text Recognition	- The Fast Convolutional Recurrent Neural Network (Fast-RCNN) - Connectionist Temporal Classification (CTC) - Gradient-based Attacks	- Image Filtering - Text Filtering
Apache SpamAssassin: An E-mail Spam Filtering [18]	- Rule-Based Filtering - Collaborative Filtering - DNS-Based Filtering - Bayesian Filtering	- Optical Character Recognition (OCR) - Naïve Bayes Algorithm	- Text Filtering
Facebook Spam Filtering Tool	- OCR System - Text Recognition	- Optical Character Recognition (OCR)	- Text Filtering

The Malena, Rosetta, and Apache SpamAssassin tools share similarities in terms of utilizing image and text filtering techniques. They employ image filtering to process and analyze images and text filtering to detect and analyze text content. However, they differ in their specific focuses and applications. Malena is designed to detect adversarial promotional porn pictures (APPI), using techniques like scene text detection and ROI-based detection. Rosetta focuses on large-scale text detection and recognition in images, employing OCR and text recognition methods. Apache SpamAssassin specializes in email spam filtering, employing rule-based filtering, collaborative filtering, DNS-based filtering, and Bayesian filtering techniques. These tools also differ in their neural network architectures and additional techniques used for content analysis and filtering.

The main strategies include enhancing contextual understanding and developing an automated spam filtering tool for Facebook that operates independently of OCR. This tool will incorporate multilingual

and multifont support, as well as filtering and verification techniques. By implementing these strategies, the resulting Facebook Spam Filtering Tool will offer powerful, accurate, and versatile functionalities, improving spam detection and effectively addressing spam on the Facebook platform.

3. Methodology

The methodology that will be utilised to create the Facebook Spam Filtering Tool is a prototype model based on the research that has been done. The best model to employ in the creation of this project is a prototype. The prototype concept also offers the ability to lower the rising and complex development costs of online systems.

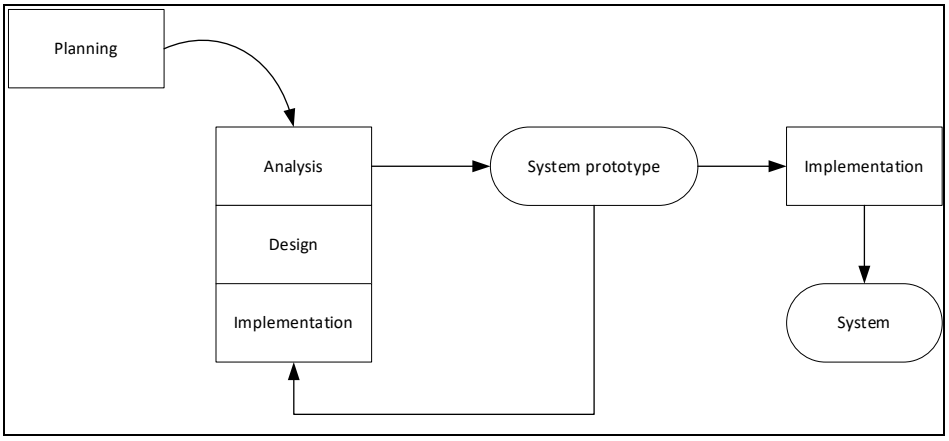

Figure 3: Prototype Model Phases

Figure 3 presents six phases in the Prototype Model that are Planning Phase, Analysis Phase, Design Phase, Implementation Phase, Prototype Phase and Testing Phase. The Prototype model is selected due to its ability to effectively reduce the number of iterations required in the System Development Life Cycle (SDLC). This results in time savings and increases the probability of user satisfaction.

3.1 Tools Development Workflow

There are a total of six phases that are involved in this prototyping model. Table 2 shows the system development activities and the deliverables for each phase.

Table 2: System Development Activities

Phase	Task	Output
Planning	Work scheduling, problem identification, scope, and objective.	Gantt chart and proposal.
Analysis	Collect and analyze information.	System requirements, software hardware requirements, Unified Modelling Language (UML), interface design and tool architecture.
Design	Design user architecture with the suitable programming language.	Architecture, system flow and process of the Facebook Spam Filtering Tool.
Implementation	Carry out the testing system and fix the errors.	System program code using HTML, JavaScript and JavaScript Object Notation (JSON).
Prototype	Identify the problems that exist in the system and repair the existing system, Repetition of the planning phase until the implementation phase and Problem identification and repair the developed system.	System Prototype.
Testing	Testing and tracking if there is an error on the system by the user.	The system has been fully tested and is ready for use.

The prototyping model is an iterative approach in software development where a functional prototype of the system is created before full development. It starts by gathering initial requirements and quickly designing and constructing a basic prototype. Stakeholders evaluate the prototype and provide feedback for iterative improvements. Once stakeholders are satisfied, the final system is developed using the refined prototype. This model allows for early user involvement, a clearer understanding of requirements, and risk mitigation. However, if not carefully managed, it may compromise scalability and performance. In summary, the prototyping model is a flexible and collaborative approach that enables faster feedback cycles and customized solutions.

Table 3: The Functional Requirement

Functional Requirements
The extension tool should provide user friendly interface for user to interact with the tool.
The extension tool should be allowed user to toggle the button whether the user want to enable or disable the extension tool.
The extension tool should be allowed the user to add, edit or remove the keyword that contain in the 'chrome.storage' through the extension tool.

Table 4: The Non-Functional Requirement

Requirement	Description
Operational	This extension tool is able to use on Google Chrome browser perfectly.
Performance	Reasonable response time should be expected when operating the extension tool.
Security	The blacklist of keywords or any sensitive data is stored locally or synchronized across devices using browser storage (such as chrome.storage.sync), ensure that the data is encrypted and securely stored.
Usability	The general flow of the extension tool is easily understandable.
Integrity	Storage of data will be done properly by the system so that it will not corrupt or non-readable.

Table 3 and Table 4 show the functional and non-function requirements of the extension tool project, respectively. JavaScript is an ideal programming language for implementing an OCR algorithm to filter spam in a Facebook Spam Filtering Tool due to its large and active community of developers and users, the availability of libraries and resources for tasks such as image processing and text analysis, and its general-purpose nature and built-in features that make it easy to read, understand, and maintain. Its dynamic typing and automatic memory management make it a good choice for developing complex systems like a spam detection tool. Overall, JavaScript's combination of features and resources makes it well-suited for implementing an OCR algorithm to filter spam effectively and efficiently.

3.2 Hardware and Software Requirement

For software requirement, JavaScript Programming language is the best programming language for the development of the Facebook Spam Filtering Tool. JavaScript's programming style known as object-oriented programming (OOP) revolves around the use of objects and classes. This approach aims to incorporate real-world concepts such as inheritance, polymorphism, and encapsulation into programming. The core concept behind OOP is to combine data and the functions that operate on it in such a way that no other part of the code can access it.

Table 5: Hardware Requirement

Hardware	Specification
CPU	Intel Core i7-10870H 2.20GHz
Memory	32GB DDR4 2933MHz SODIMM
Operating System	Windows 11 Pro 22H2
Browser Specification	Any latest Chromium based architecture browser.

The hardware requirements for a Facebook Spam Filtering Tool can vary based on several factors. These factors include the size and complexity of the words being blocked. Table 5 shows the hardware requirement utilized in this project for the development of the extension tool.

4. Tools Analysis and Design

This section explains the analysis and design of the Facebook Spam Filtering Tools that have been proposed in this project.

4.1 Tool Architecture

The Facebook Spam Filtering Tools architecture in Figure 4 describes the way to use a browser extension tool to filter out certain keywords or symbols on a webpage. This tool requires manual installation by the user on specific browsers, Google Chrome, or Microsoft Edge. The installation process involves dragging the .crx file to the browser or opening it directly. After installation, the browser needs to be refreshed or restarted. The tool reads a dictionary of spam keywords saved in JavaScript, which can be customized by the user. The user inputs the desired spam keywords or symbols and saves them. The tool saves the keywords to 'chrome.storage' and reads them for spam filtering. When a webpage is fully loaded, the tool executes a spam block feature that removes any text matching the defined spam keywords from the user's view. OCR scanning is used to improve text detection accuracy. The user can choose to cancel or proceed with blocking spam text on Facebook pages when prompted.

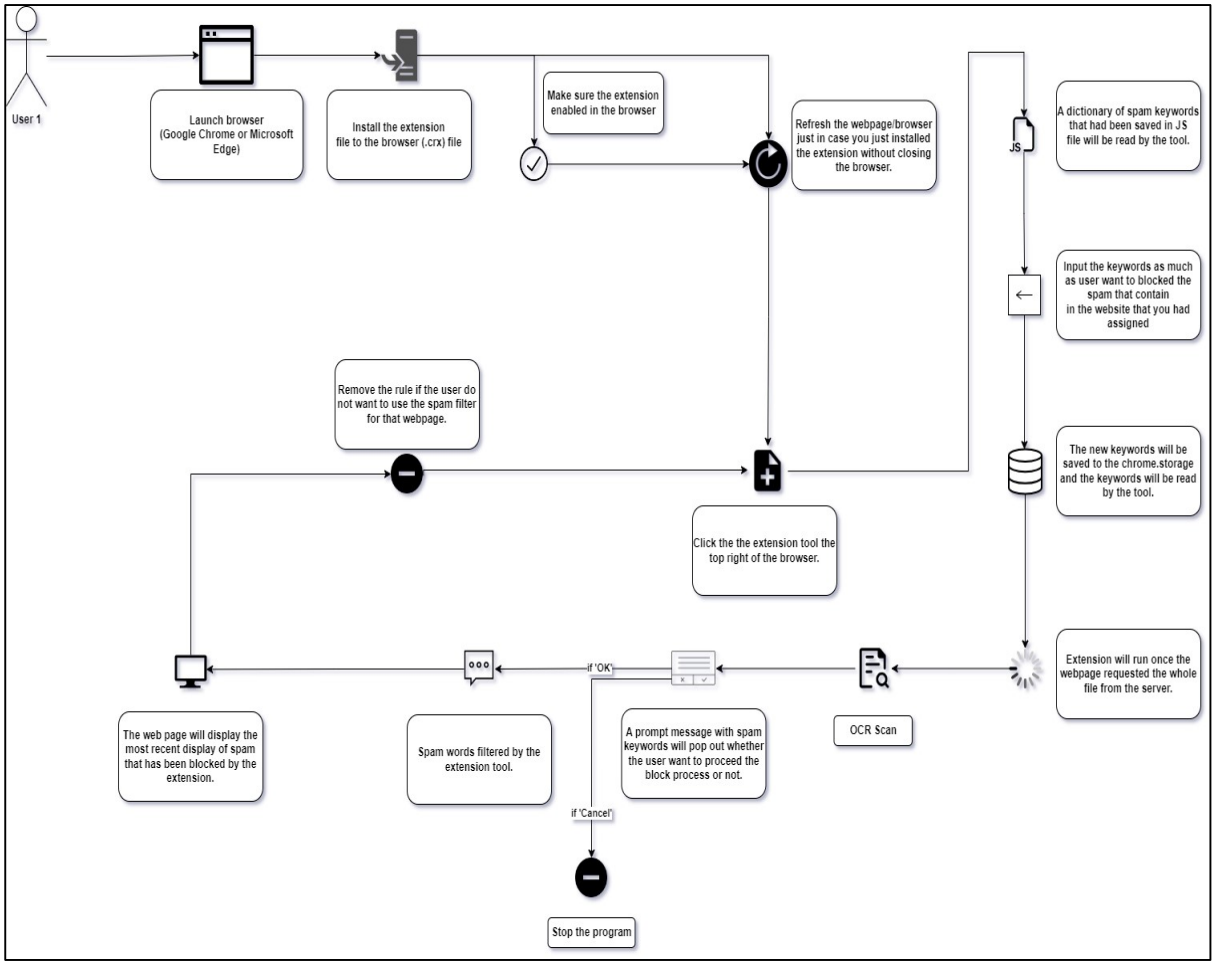

Figure 4: The Architecture of Facebook Spam Filtering Tool

4.2 Use Case Diagram

The use case diagram in Figure 5 there are four modules such as input keywords, view keywords, edit or remove keywords, and prompt enable or disable the tool contained in the proposed system which can be accessed by user. Besides, users can also input the new spam keywords that are not available in the spam keywords dictionary so that the spam detection is able to fulfil the user's defined keyword criteria. At the same time, the user also can edit or remove the keywords that had been defined by the user and choose whether the user wants to enable or disable from the prompt message of the proposed extension tool.

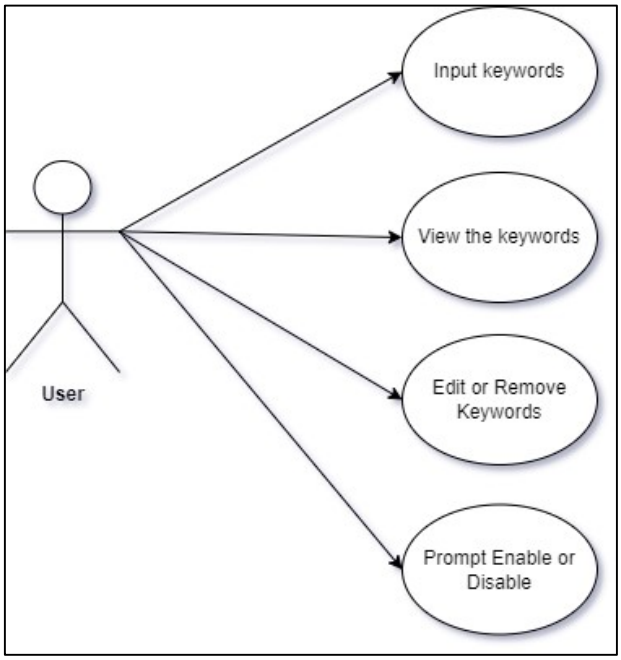

Figure 5: The Use Case Diagram of Facebook Spam Filtering Tool

4.3 Sequence Diagram

The sequence diagram in Figure 6 depicts the sequence diagram illustrating the interactions in the use case described in the previous use-case diagram. Upon running the program, the user is presented with the option to input their own spam keywords. However, if they prefer not to define their own keywords, they can utilize the extension tool, which already includes a pre-set dictionary of spam keywords established during its development. If any of the spam keywords match the text on a Facebook page, a prompt message will appear, notifying the user of the specific spam keywords that should be blocked. The user then has the choice to proceed with the blocking or cancel the action. It is important to note that this extension tool operates automatically and once set up in the Chrome browser, will continuously run whenever it detects the Facebook domain.

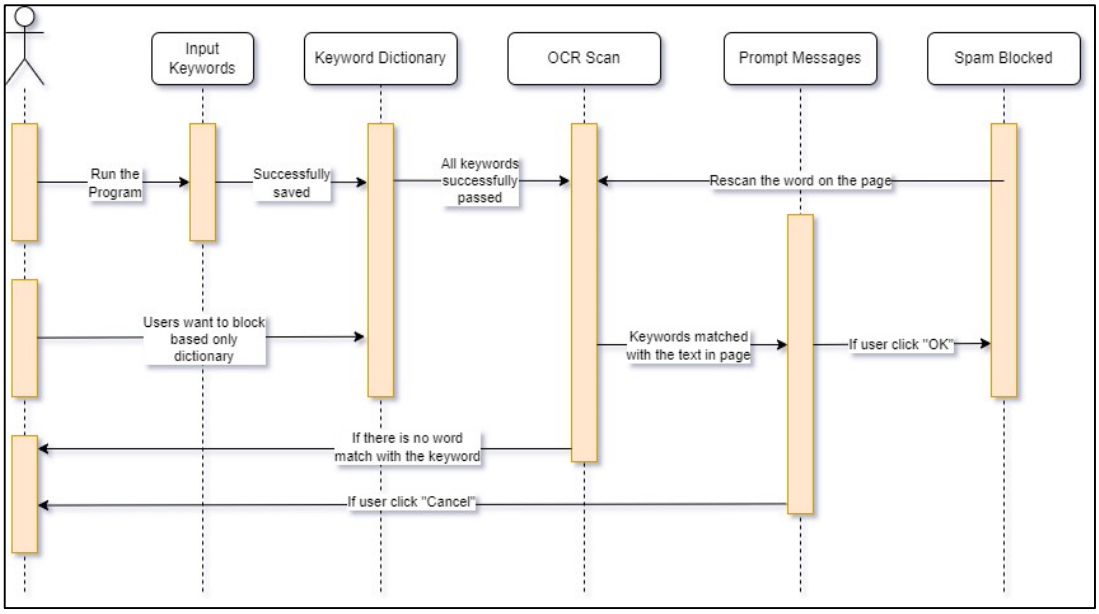

Figure 6: The Sequence Diagram of Facebook Spam Filtering Tool

4.4 Class Diagram

Figure 7 shows there are five different classes which are ContentScript, chrome.storage , spamkeywords.js. The attributes that are used in the class diagrams are xpathPatterns, filteredKeywords, isFiltering, and isPromptShown. The Set() is used to get the inputs from the user while Get() is used to display the output to the user.

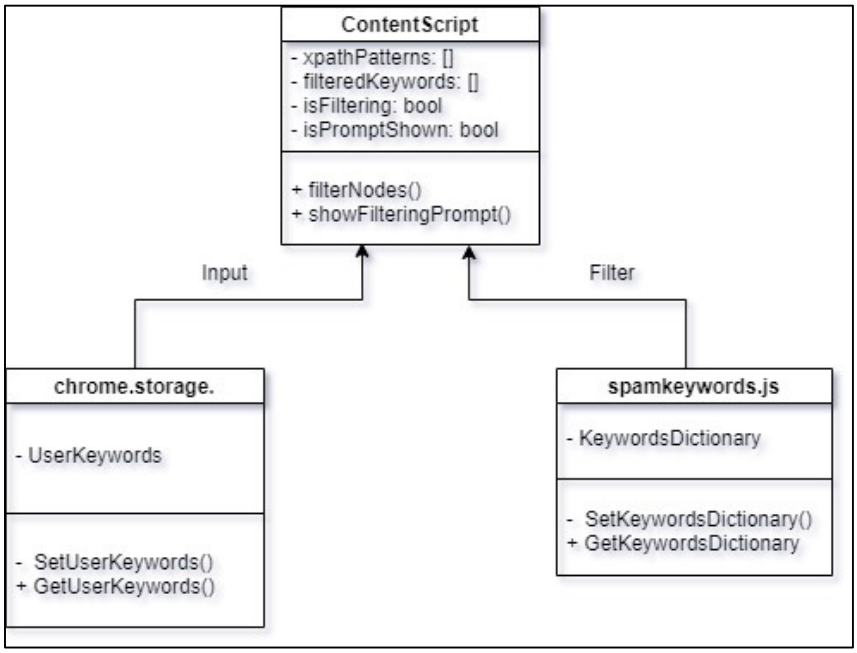

Figure 7: The Class Diagram of Facebook Spam Filtering Tool

4.5 Activity Diagram

Figure 8 presents an activity diagram, which functions as a flowchart, illustrating the interactions of the use case described in the previous use-case diagram. Upon running the program, the initial step involves the user having the option to input their own spam keywords. However, if they choose not to define these keywords, the extension tool will proceed using a predefined dictionary of spam keywords that was set up during the tool's development.

If there are any spam keywords that match the text on the Facebook page, a prompt message will appear, informing the user about the spam keywords that need to be blocked. At this point, the user can decide whether to proceed with the blocking action or cancel it. It's important to note that this extension tool operates automatically. Once set up in the Chrome browser, it will continuously run and detect Facebook domains, triggering the blocking mechanism whenever necessary.

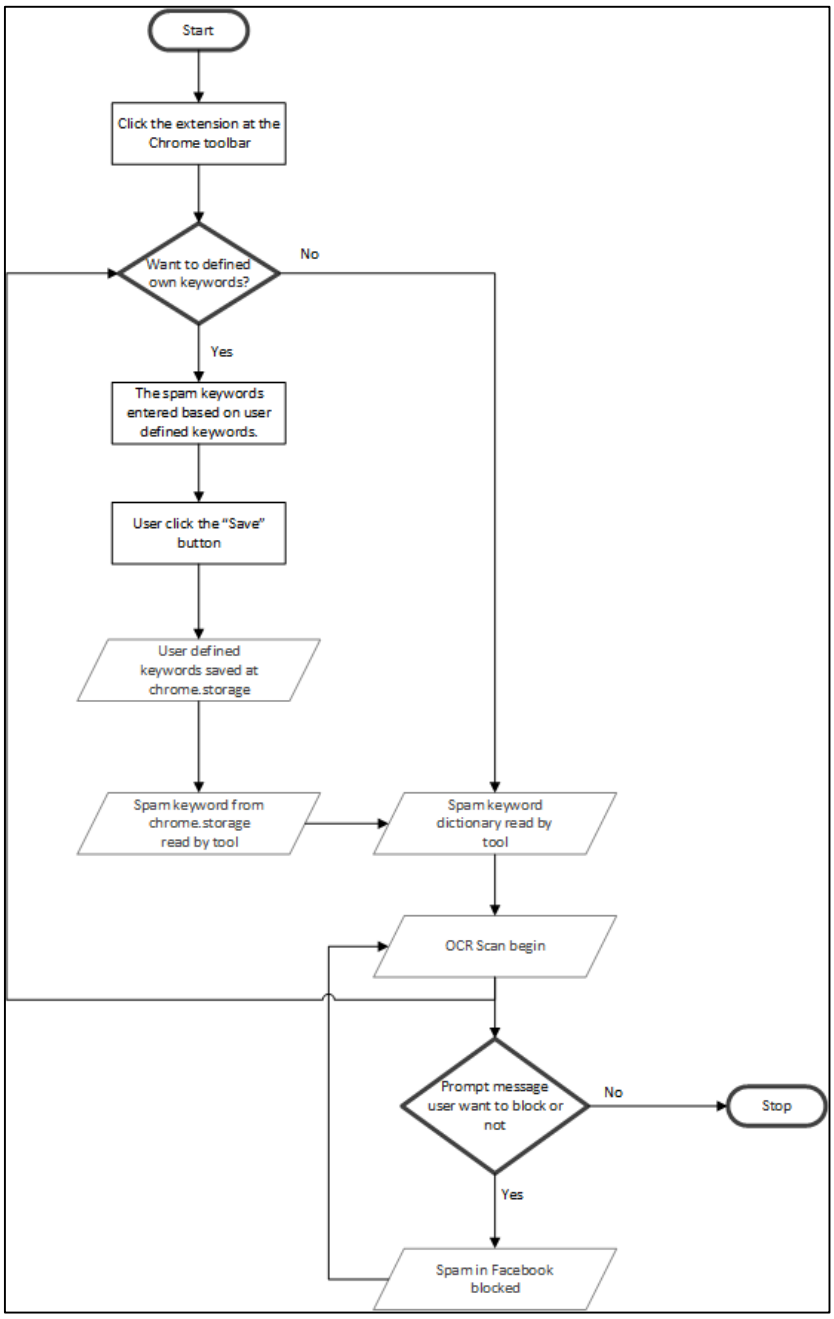

Figure 8: The Activity Diagram of Facebook Spam Filtering Tool

4.6 Design Interface

Figure 9 shows the design of the extension tool. Once the user clicks the extension tool that had been setup, the tool will display the text box which enables user to write user defined spam keywords. This feature needs to be included because there are some users who would like to block keywords based on their preference. So, by implementing this feature user not just able to block spam keywords that already defined in the dictionary of spam keywords that had been setup by the developer, they can even define their own keywords to block the spam.

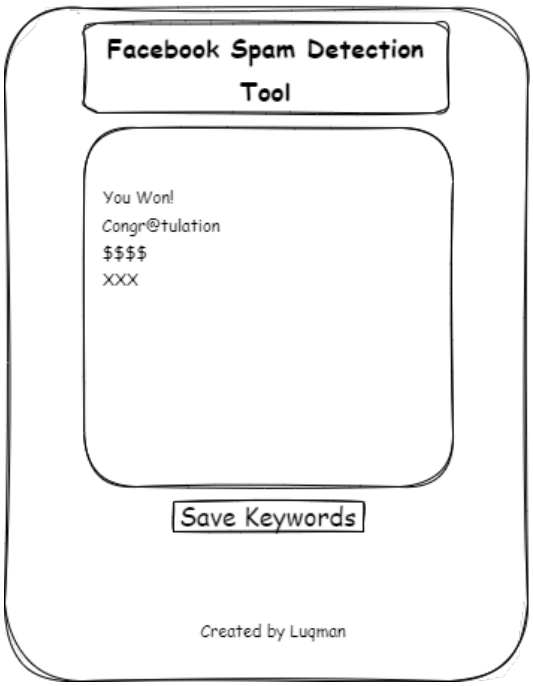

Figure 9: The Design Interface of Facebook Spam Detection

5. Result and Discussion

In this section, the screen capture of the tools and code segment and the result of the proposed tools (Facebook Spam Filtering Tool) will be shown. The implemented main function and the result of the proposed tools (Facebook Spam Filtering Tool) will be discussed in this section.

5.1 Tool Implementation and Result

The feedback on the suggested extension tool may be obtained through prototyping, which can also determine whether the tool is prepared to meet the information needs of its users. Therefore, the implementation phase is the final stage of implementation, during which the Facebook Spam Filtering Tool will be tested on end users to see if it can work as intended and fulfil user requirements.

Figure 10 shows the provided ‘manifest.json’ script represents the configuration file for a Chrome extension called "Facebook Spam Detection." The extension utilizes Manifest V3 and aims to automatically block spam on Facebook based on user-defined keywords. It includes details such as the extension's name, version, and description. The content security policy allows scripts from the extension itself, inline scripts, and scripts from both secure and insecure sources. The browser action section defines the extension's default icon, popup page, and tooltip title displayed when hovering over the icon. Overall, this script sets up the necessary information and functionalities for the "Facebook Spam Detection" extension in Chrome.

```
{
  "manifest_version": 3,
  "name": "Facebook Spam Detection",
  "version": "2.1",
  "description": "This tool will automatically block the spam that appeared in Facebook based on the keyword that had been assigned by the user",
  "content_security_policy": {
    "extension_pages": "script-src 'self'; object-src 'self'; script-src-elem 'self' 'unsafe-inline' https://* http://*;"
},
  "action" :
  {
    "default_icon" : "images/icon-550x550.png",
    "default_popup": "options.html",
    "default_title" : "You are inside your Facebook Spam Detection Filter"
  },
  "permissions": [
    "tabs",
    "storage"
  ],
  "content_scripts": [
    {
      "matches": [
        "*://*.facebook.com/*"
      ],
      "exclude_globs": [
        "*://*github.com*"
      ],
      "js": ["js/jquery-2.1.4.min.js", "js/filter.js", "js/spamkeywords.js"],
      "run_at": "document_idle",
      "all_frames": false
    }
  ],
  "icons" : {
    "16": "images/icon-550x550.png",
    "19": "images/icon-550x550.png",
    "48": "images/icon-550x550.png",
    "128": "images/icon-550x550.png",
    "130": "images/icon-550x550.png",
    "550": "images/icon-550x550.png"
  },
  "background": {
    "service_worker": [false, "background.html"]
  }

}
```

Figure 10: The Source Code of 'manifest.json'

```
for (var i = 0; i < keywords.length; i++) {

    var word = keywords[i];
    xpathPatterns.push(
      ["//body//*[not(self::script or self::style)]/text()[contains(translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz'), '" + word + "')]", word],
      ["//body//a[contains(translate(@href, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz'),'" + word + "')]", word],
      ["//body//img[contains(translate(@src, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz'),'" + word + "')]", word],
      ["//body//img[contains(translate(@alt, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz'),'" + word + "')]", word]
    );
  }
```

Figure 11: The Partial Coding of Building XPath Patterns

Figure 11 shows the loop iterates over each keyword in the 'keywords' array and performs the following actions for each keyword:

i) Constructs XPath patterns to search for occurrences of the keyword in different elements of the HTML document.
ii) Adds the constructed XPath patterns, along with the corresponding keyword, to the 'xpathPatterns' array.

The constructed XPath patterns target the following elements:

i) Text nodes withing '<body>' element that contain the keyword.
ii) '<a>' elements with 'href' attributes that contain the keyword.
iii) '<img>' elements with the 'alt' attributes that contain the keyword.

```
for (var i = 0; i < xpathPatterns.length; i++) {
  var xpathResult = document.evaluate(
    xpathPatterns[i][0],
    document,
    null,
    XPathResult.UNORDERED_NODE_ITERATOR_TYPE,
    null
  );
  var thisNode = xpathResult.iterateNext();
```

Figure 12: The Partial Coding of Iterating through XPath Patterns

Figure 12 shows the code employs a loop to iterate through an array called ‘xpathPatterns’, which presumably contains XPath patterns and their corresponding keywords for filtering. Within this loop, the code evaluates each XPath pattern against the document using ‘document.evaluate()’. The result is an iterator (‘xpathResult’) that holds the matching nodes.

```
for (var i = 0; i < filteredNodes.length; i++) {
  var node = filteredNodes[i].node;
  var keyword = filteredNodes[i].keyword;

  if (filteredKeywords.includes(keyword)) {
    continue; // Skip if keyword was previously cancelled
  }

  if (confirmFilterKeyword(keyword)) {
    var p = node.parentNode;
    if (p !== null) {
      p.removeChild(node);
    }
  } else {
    filteredKeywords.push(keyword); // Add keyword to filtered list
  }
}
```

Figure 13: The Partial Coding of Handling Filtered Nodes

Figure 13 shows another loop is introduced to process the filtered nodes stored in the filteredNodes array. For each filtered node, the associated keyword is retrieved. The code checks if the keyword has been previously canceled by checking against the filteredKeywords array. If it has, the code skips further processing for that node.

The code then prompts the user for confirmation to remove the node by invoking the confirmFilterKeyword(keyword) function. If the user confirms the removal, the node is removed from its parent by accessing the parent node (p) and using p.removeChild(node). In case the user does not confirm the removal, the keyword is added to the filteredKeywords array to indicate that it has been processed.

Figure 14 shows the interface of the extension tool after installation. Users will see its symbol at the top-right corner. If it's not visible, they can access it by clicking the puzzle symbol and pinning it to the toolbar. The tool automatically detects and filters spam keywords on the Facebook page being browsed. Users can input additional spam keywords in the extension by clicking it and entering the desired keywords in the text box. These keywords, along with the ones in the tool's dictionary, will be used for blocking purposes. To save the additional keywords, users need to click the 'Save' button, storing them in the 'chrome.storage' system.

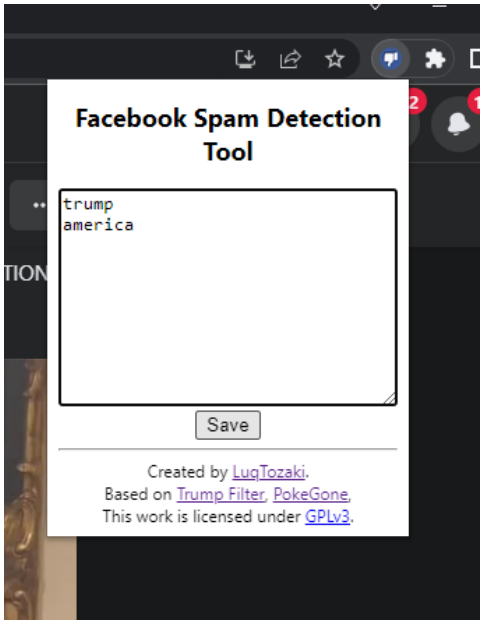

Figure 14: The Interface of the Extension Tool

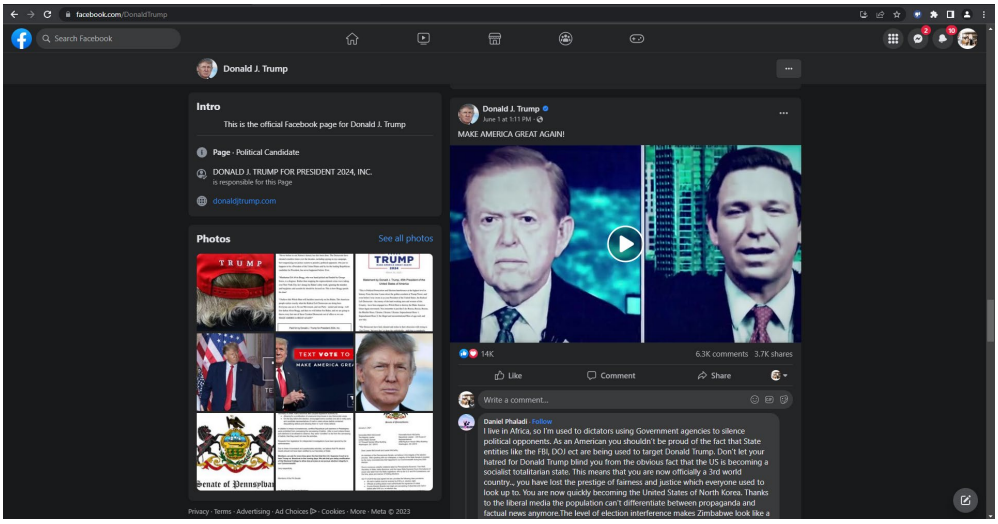

Figure 15: Screenshot of Facebook Page Before used the Extension Tool

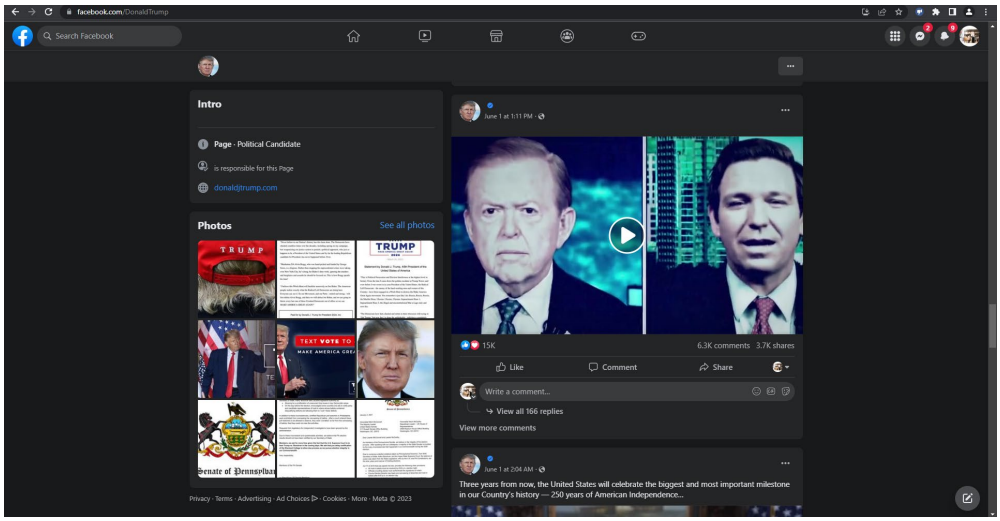

Figure 16: Screenshot of Facebook Page After used the Extension Tool

The initial screenshot which in Figure 15 captured prior to the application of the filter serves as a reference point to establish the presence of spam keywords within the Facebook environment. This snapshot showcases a typical Facebook browsing session where spam content, including posts or comments containing the keywords 'trump' and 'america', can be observed.

The developed extension, specifically designed to detect and filter spam keywords on Facebook, was deployed to the Chrome browser. The extension's functionality includes scanning the content of posts, comments, and other relevant elements to identify instances of spam keywords. After the successful installation of the extension, the same Facebook browsing session was revisited. A second screenshot was captured, representing the Facebook interface after the spam filter had been activated. This image demonstrates the impact of the filter, revealing the absence of posts or comments containing the targeted spam keywords ('trump' and 'america').

5.2 Testing Phase

In this section, the test case plan will be carried out to determine the result of Facebook Spam Filtering Tool. The test case plan shows the list of test plan and the result of the testing. The expected result of the test plan is also included in the table to as standard test to the Facebook Spam Filtering Tool tools. The test case plan is show as in table 5.

Table 6: List of Test Cases

No.	Test Cases	Description
TEST_100		
1.	TEST_100_001	The interface of the extension tool appears when user click the ‘Thumb Down’ symbol at the toolbar.
2.	TEST_100_002	Able to click embedded link on ‘LuqTozaki’ text which navigate to the GitHub.
3.	TEST_100_003	Able to click embedded link on ‘Trump Filter’ text which navigate to the GitHub of the reference.
4.	TEST_100_004	Able to click embedded link on ‘PokeGone text which navigate to the GitHub of the reference.
5.	TEST_100_005	A message “Words saved” pop up when user click the ‘Save’ button.
6.	TEST_100_006	Able to resize the text box using the resize feature at the bottom-right corner of the box.
7.	TEST_100_007	Able to click ‘OK’ when user want to block the spam and click ‘Cancel’ button if the user do not want to block the spam.
TEST_200		
1.	TEST_200_001	Able to input the spam keyword that define by the user itself.
2.	TEST_200_002	Able to change the spam keyword that define by the user.
3.	TEST_200_003	The spam keyword that had been assigned by the user successfully save and loaded even when the browser close.
TEST_300		
1.	TEST_300_001	Prompt message pop up when detect every spam keyword in that page.
2.	TEST_300_002	Able to block the spam keyword based on dictionary of spam keywords that had been defined by the developer.
3.	TEST_300_003	Able to block the spam keyword based on dictionary of spam keywords that had been defined by the user.
4.	TEST_300_004	The spam keywords will only block at the page with the domain “facebook.com”.

Table 6 shows the functional test for the user interface is conducted. This tool will work when the click and navigate every button available in the extension tool. There will be three tests run in this extension tool which TEST_100 for User Interface Test, TEST_200 for the Ability to Save Spam Keywords and TEST_300 for the Ability to Block Spam Keywords.

In these three categories of testing, all of them meet the requirements stated in the previous chapter. This indicates that the developed system is now well-functioning and to be proposed for public use.

Table 7: List of Test Result

No.	Test Cases	Description	Expected Result	Actual Result
TEST_100				
1	TEST_100_001	The interface of the extension tool appears when user click the 'Thumb Down' symbol at the toolbar.	The interface successfully launches in the browser	Pass
2	TEST_100_002	Able to click embedded link on 'LuqTozaki' text which navigate to the GitHub.	The link will navigate by opening it in new tab.	Pass
3	TEST_100_003	Able to click embedded link on 'Trump Filter' text which navigate to the GitHub of the reference.	The link will navigate by opening it in new tab.	Pass
4	TEST_100_004	Able to click embedded link on 'PokeGone text which navigate to the GitHub of the reference.	The link will navigate by opening it in new tab.	Pass
5	TEST_100_005	A message "Words saved" pop up when user click the 'Save' button.	The message successfully appears.	Pass
6	TEST_100_006	Able to resize the text box using the resize feature at the bottom-right corner of the box.	The box successfully resizes based on user preference.	Pass
7	TEST_100_007	Able to click 'OK' when user want to block the spam and click 'Cancel' button if the user do not want to block the spam.	The button is well-functioning and able to process based on button that is chosen by the user.	Fail
TEST_200				
1	TEST_200_001	Able to input the spam keyword that define by the user itself.	Successfully enter the keywords in the text box.	Pass
2	TEST_200_002	Able to change the spam keyword that define by the user.	Successfully backspace or remove the keywords in the text box.	Pass
3	TEST_200_003	The spam keyword that had been assigned by the user successfully save and loaded even when the browser close.	Successfully saved the keywords that had been defined by the user.	Pass
TEST_300				
1	TEST_300_001	Prompt message pop up when detect every spam keyword in that page.	The prompt message successfully pop-out.	Fail
2	TEST_300_002	Able to block the spam keyword based on dictionary of spam keywords that had been defined by the developer.	The spam keyword successfully blocked.	Pass
3	TEST_300_003	Able to block the spam keyword based on dictionary of spam keywords that had been defined by the user.	The spam keyword successfully blocked.	Pass
4	TEST_300_004	The spam keywords will only block at the page with the domain "facebook.com".	The spam keyword successfully blocked only in the domain.	Pass

Table 7 presents a comprehensive overview of the test cases conducted for the extension tool. These test cases aimed to evaluate the functionality and performance of the tool, ensuring that it meets the desired requirements. Under the category TEST_100, various aspects of the extension tool were assessed. The interface of the tool successfully launched upon clicking the 'Thumb Down' symbol, and embedded links within the text directed users to the relevant GitHub pages in new tabs. The 'Save' button triggered the expected message "Words saved," while the text box could be resized using the

intuitive resizing feature. The 'OK' and 'Cancel' buttons functioned correctly, allowing users to block or not block spam based on their choice.

Moving on to TEST_200, the focus shifted to user-defined spam keywords. Test cases confirmed that users were able to input their own defined keywords, change them as needed, and observe that the keywords were consistently saved and loaded even after closing the browser. In TEST_300, the examination centered on the ability of the tool to detect and block spam keywords. The extension effectively triggered prompt messages when spam keywords were detected on a page. It successfully blocked spam keywords based on both developer-defined and user-defined dictionaries. Additionally, the extension correctly limited the blocking of spam keywords to pages with the domain "facebook.com," ensuring its effectiveness on the intended platform.

In these three categories of testing, all of them meet the requirements stated in the previous chapter. This indicates that the developed system is now well-functioning and to be proposed for the public user.

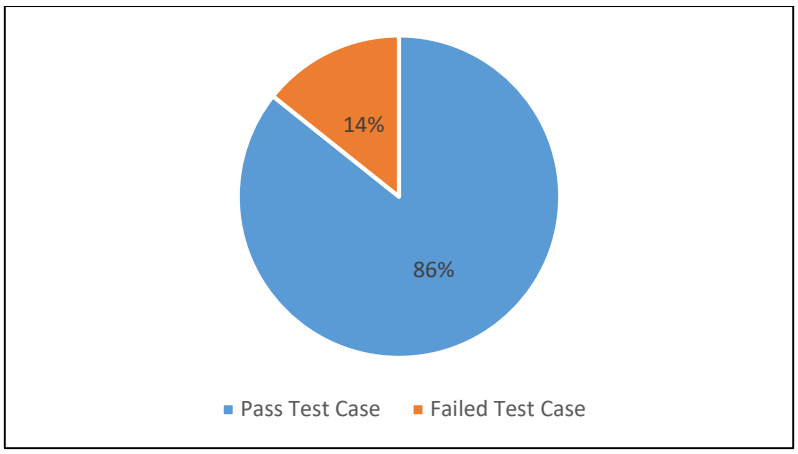

Figure 17: Summary of Overall Test Result

Figure 17 shows the summary of overall test result in the form of pie chart. There is a little problem that happening regarding the prompt message which make them to vote Failed. It is because they have a lot of extensions that had been setup in the Chrome browser, so it is possible that there may be conflicts between your extension and other installed extensions on the user's browser. Try disabling other extensions temporarily to see if the issue persists. If the problem disappears when other extensions are disabled, there may be a conflict that needs to be resolved.

6. Conclusion and Future Works

The Facebook Spam Filtering Tool has the potential to have a significant impact on society by helping to protect people from spam and fraudulent activity on the Facebook platform. This impact can be measured through various metrics, including the reduction in the number of reported spam incidents, increased user satisfaction, and a decline in instances of fraudulent or harmful content. By making it easier for users to identify and avoid spam, the tool can help to increase trust in the platform, measured through user feedback and surveys, and make it a safer and more enjoyable place for people to connect and share, reflected in improved user engagement and a reduction in spam-related incidents.

The Facebook Spam Filtering Tool is a valuable tool that can help to protect users from spam on the Facebook platform. By providing customizable spam filters and a range of other features, it offers a comprehensive solution for tackling the spam issue on the platform.

The use of the OCR algorithm in the tool allows it to recognize and extract text from images and scanned documents, making it effective at detecting spam messages that are embedded in non-textual formats. Additionally, the use of JavaScript as the programming language allows for the efficient and

accurate processing of large amounts of data, making the tool well-suited for handling the high volume of messages that are typically present in a spam detection system.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, p. e01802, Jun. 2019, doi: 10.1016/J.HELIYON.2019.E01802.

[2] R. Kaur, S. Singh, and H. Kumar, "Rise of spam and compromised accounts in online social networks: A state-of-the-art review of different combating approaches," *Journal of Network and Computer Applications*, vol. 112, pp. 53–88, Jun. 2018, doi: 10.1016/J.JNCA.2018.03.015.

[3] A. Barushka and P. Hajek, "Spam filtering in social networks using regularized deep neural networks with ensemble learning," *IFIP Adv Inf Commun Technol*, vol. 519, pp. 38–48, 2018, doi: 10.1007/978-3-319-92007-8_4.

[4] E. M. Redmiles, N. Chachra, and B. Waismeyer, "Examining the Demand for Spam: Who Clicks? - Meta Research," *Meta Research*, Apr. 2018, Accessed: Jan. 09, 2023. [Online]. Available: https://research.facebook.com/publications/examining-the-demand-for-spam-who-clicks/

[5] J. Boyd, "The History of Facebook: From BASIC to global giant | Brandwatch," Brandwatch. Accessed: Jan. 09, 2023. [Online]. Available: https://www.brandwatch.com/blog/history-of-facebook/

[6] Google, "Introducing reCAPTCHA v3: the new way to stop bots | Google Search Central Blog | Google for Developers." Accessed: Jun. 14, 2023. [Online]. Available: https://developers.google.com/search/blog/2018/10/introducing-recaptcha-v3-new-way-to

[7] K. Curran, S. Morrison, and S. M. Cauley, "Google+ vs Facebook: The Comparison," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 10, no. 2, pp. 379–388, Jun. 2012, doi: 10.12928/TELKOMNIKA.V10I2.814.

[8] N. N. Gomes de Andrade, D. Pawson, D. Muriello, L. Donahue, and J. Guadagno, "Ethics and Artificial Intelligence: Suicide Prevention on Facebook," *Philos Technol*, vol. 31, no. 4, pp. 669–684, Dec. 2018, doi: 10.1007/S13347-018-0336-0/METRICS.

[9] N. H. Imam, V. G. Vassilakis, and D. Kolovos, "OCR post-correction for detecting adversarial text images," *Journal of Information Security and Applications*, vol. 66, p. 103170, May 2022, doi: 10.1016/J.JISA.2022.103170.

[10] S. Impedovo, L. Ottaviano, and S. Occhinegro, "Optical Character Recognition - A Survey," *Intern J Pattern Recognit Artif Intell*, vol. 05, no. 01n02, pp. 1–24, Nov. 2011, doi: 10.1142/S0218001491000041.

[11] S. Karthikeyan, A. G. S. De Herrera, F. Doctor, and A. Mirza, "An OCR Post-Correction Approach Using Deep Learning for Processing Medical Reports," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 5, pp. 2574–2581, May 2022, doi: 10.1109/TCSVT.2021.3087641.

[12] I. M. D. R. Mudiarta *et al.*, "Balinese character recognition on mobile application based on tesseract open source OCR engine," *J Phys Conf Ser*, vol. 1516, no. 1, p. 012017, Apr. 2020, doi: 10.1088/1742-6596/1516/1/012017.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 580–587, Nov. 2013, doi: 10.48550/arxiv.1311.2524.

[14] L. Gao, Y. He, X. Sun, X. Jia, and B. Zhang, "Incorporating Negative Sample Training for Ship Detection Based on Deep Learning," *Sensors 2019, Vol. 19, Page 684*, vol. 19, no. 3, p. 684, Feb. 2019, doi: 10.3390/S19030684.

[15] K. Yuan *et al.*, "Stealthy porn: Understanding real-world adversarial images for illicit online promotion," *Proc IEEE Symp Secur Priv*, vol. 2019-May, pp. 952–966, May 2019, doi: 10.1109/SP.2019.00032.

[16] A. P. B. Lopes, S. E. F. De Avila, A. N. A. Peixoto, R. S. Oliveira, M. D. M. Coelho, and A. D. A. Araújo, "Nude detection in video using bag-of-visual-features," *Proceedings of SIBGRAPI 2009 - 22nd Brazilian Symposium on Computer Graphics and Image Processing*, pp. 224–231, 2009, doi: 10.1109/SIBGRAPI.2009.32.

[17] F. Borisyuk, A. Gordo, and V. Sivakumar, "Rosetta: Large scale system for text detection and recognition in images," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 71–79, Oct. 2019, doi: 10.1145/3219819.3219861.

[18] Alistair. McDonald, *SpamAssassin : A Practical Guide to Configuration, Customization, and Integration*, First. Birmingham: Packt Pub, 2004.

[19] S. Gibson, B. Issac, L. Zhang, and S. M. Jacob, "Detecting spam email with machine learning optimized with bio-inspired metaheuristic algorithms," *IEEE Access*, vol. 8, pp. 187914–187932, 2020, doi: 10.1109/ACCESS.2020.3030751.