APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 2 (2025) 833-849
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

A Phishing and Malware Link Detector Using Checkphish API

Iqmal Hazeeq Zulkarnain[1], Shamsul Kamal Ahmad Khalid[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: *shamsulk@uthm.edu.my*
DOI: https://doi.org/10.30880/aitcs.2025.06.02.045

Article Info

Received: 20 October 2025
Accepted: 19 November 2025
Available online: 30 November 2025

Keywords

URL Safety,Suspicious Link Detection Using Checkphish API,

Cybersecurity,Phishing Prevention

Abstract

The Suspicious Link Detection Extension is a web-based tool designed to enhance online security by providing real-time URL safety checks to protect users from phishing scams, malware, and other potential internet threats. Developed with Node.js and integrated with the Checkphish API, the extension evaluates URLs based on various factors, including their structure, domain reputation, and other identifying characteristics. Unlike traditional URL-checking tools that rely on outdated databases or user-reported issues, this extension offers real-time analysis of links, ensuring that users receive the most up-to-date information about potential threats. The extension's user-friendly interface is designed with non-technical users in mind, allowing anyone to easily input URLs and receive immediate feedback on whether the link is safe, suspicious, or potentially harmful. This makes it an ideal tool for individuals, educational institutions, and businesses concerned about cybersecurity, offering a simple and effective way to prevent users from visiting malicious websites. The tool aims to bridge the gap between complex cybersecurity measures and everyday internet users by providing an easy-to-use, real-time safety check, ensuring a more secure browsing experience for everyone. By making security technology more accessible, this project strives to create a safer internet environment, particularly for those with limited technical knowledge.

1. Introduction

The internet has become an important part of modern life because of how quickly digital communication has grown. It makes things like shopping, studying, social networking, and working incredibly easy. This ease of use comes with more risks, especially online threats like phishing, malware attacks, and data breaches. Putting harmful URLs in emails, websites, or social media posts is a common way for hackers to get around security. People click on these URLs because they look like real ones, which expose private information or compromises their systems. As these threats get smarter, it gets harder for people who don't know much about computers to tell the difference between safe and dangerous links.

Existing URL research tools are useful, but most people can't use them because they are too complicated or need much computing power. Many people use static databases, which can't keep up with how hackers are changing their tactics all the time. This hole shows how important it is to have a smart, easy-to-use system that can find dangerous URLs instantly and doesn't need complicated technology knowledge.

The main topic of this essay is the creation of a Suspicious Link Detection using Checkphish API that gives users an easy but useful way to check the safety of URLs. The system looks at URL structure, domain reputation,

and information to provide real-time safety analysis. The system aims to improve online safety for people and businesses by combining easy access with powerful analytical tools.

1.1 Objective

The objective of this project is to design and develop a suspicious and malicious web link detector that analyzes domain reputation, URL structures, and content characteristics using real-time scanning powered by the CheckPhish API. CheckPhish employs supervised machine learning techniques, particularly classification models such as decision trees and gradient boosting, trained on large datasets of known phishing and benign URLs to identify suspicious patterns in newly submitted links. The system is implemented as a browser extension using development tools such as JavaScript, Node.js, and WebExtension APIs to ensure seamless integration and ease of use. Additionally, the project aims to evaluate the security, reliability, and detection accuracy of the system through both functional testing and user acceptance testing involving a diverse group of participants.

2. Related Work

Phishing and malware attacks are two of the most common threats faced by internet users today. Phishing involves tricking users into revealing sensitive data by using deceptive websites or fake login pages, often delivered via suspicious links. Malware, on the other hand, refers to harmful software that can be downloaded through malicious URLs, leading to system compromise or data theft. Traditional detection tools often rely on static blocklists, which struggle to detect newly created phishing sites. To address these challenges, link detection systems increasingly utilize machine learning to analyze URL structure, domain reputation, and behavioral patterns in real time [1][2][3]. This section reviews existing URL-based phishing detection tools and highlights how CheckPhish and similar systems overcome earlier limitations.

2.1 URLCheck.io

URLCheck.io is a reliable platform designed to analyze URLs for potential threats such as phishing and malware. It focuses on providing accurate domain reputation data and URL safety assessments. While the system is user-friendly and accessible, it lacks advanced real-time analysis and machine learning capabilities, which limits its ability to predict and detect emerging threats effectively (URLCheck.io) [15].

2.2 URLVoid

URLVoid focuses on domain reputation analysis by utilizing public databases to classify URLs as safe or harmful. Its minimalist interface and straightforward design make it accessible to users with minimal technical knowledge. However, the reliance on static dataset limits its ability to address emerging threats, such as zero-day attacks or newly registered malicious domains (URLVoid) [16].

2.3 ScanURL

ScanURL is designed for basic URL evaluation by checking links against public blocklists and user-reported threats. Its simplicity makes it suitable for non-technical users who need a quick and easy solution. However, its lack of dynamic analysis and heavy dependence on static reporting restricts its ability to identify advanced or evolving cyber threats (ScanURL) [17].

2.4 Comparison with the CheckPhish

Table 1 compares four popular URL analysis tools based on their ability to perform real-time detection, evaluate domain reputation, apply machine learning techniques, and offering user accessibility.

Table 1 *Comparison of URL analysis tools*

Feature	URLCheck.io	URLVoid	ScanURL	CheckPhish
Real-time Analysis	Yes	No	No	Yes
Domain Reputation	Yes	Yes	Yes	Yes
User Accessibility	Easy	Easy	Moderate	Easy
Machine Learning	Yes (Supervised ML, classification models)	No	No	Yes (Supervised ML, likely gradient boosting or ensemble

The "Yes" under Machine Learning indicates that both URLCheck.io and CheckPhish utilize supervised machine learning techniques to improve detection accuracy. These systems are trained on large datasets of phishing and benign URLs to identify malicious patterns. While the exact models used are not fully disclosed, such systems typically rely on classification algorithms like decision trees, gradient boosting, or ensemble learning models. These approaches enable real-time and adaptive threat detection that evolves alongside emerging phishing tactics. In contrast, URLVoid and ScanURL do not incorporate machine learning, relying instead on static reputation databases and user-reported lists.

Both URLCheck.io and CheckPhish apply link detection methods, but CheckPhish stands out by integrating machine learning with real-time analysis capabilities [18]. Its API evaluates URLs by extracting features such as domain age, token count, and use of special characters, enabling more accurate phishing and malware classification. These techniques allow it to function as a dynamic link detector rather than relying solely on predefined blocklists.

3. Methodology

The System Development Life Cycle (SDLC) methodology selected for the development of the Suspicious Link Detection System is the Prototype Model. This approach involves creating simplified and preliminary versions of the system (prototypes) to gather user feedback and iteratively refine the design and functionality. Fig. 1 illustrates the phases of the Prototype Model.

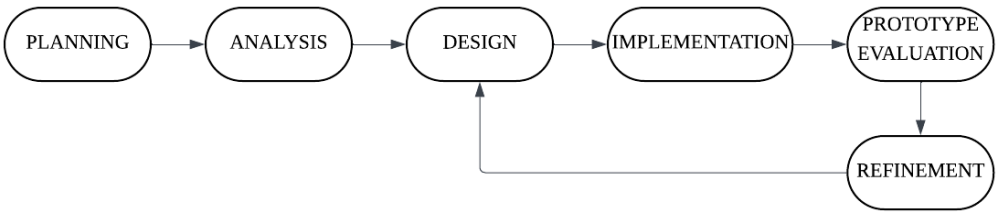


Fig.1 *Prototype Model phases*

3.1 Phase 1: Planning

This phase focused on defining the project scope, objectives, and timeline. The goals were established based on initial research and feedback gathered through an interview with a representative from Telekom Malaysia (TM), who reviewed and tested the prototype of the tool. Insights from the interview helped validate the relevance of detecting suspicious URLs based on their structure and domain reputation. A Gantt chart was developed to schedule tasks and milestones, while communication and coordination were managed using tools such as WhatsApp and Google Meet. The outcome of this phase was a detailed project plan and proposal outlining all development activities.

3.2 Phase 2: Analysis

In this phase, technical requirements and user needs were gathered through research, surveys, and interviews. Existing tools like URLCheck.io and URLVoid were evaluated to identify their limitations, such as reliance on static datasets and complex user interfaces. Key features needed were URL structure analysis, domain reputation verification, and integration with machine learning APIs. These findings justified developing a more user-friendly and effective phishing detection system.

3.3 Phase 3: Design

The design phase involved creating detailed system architecture and user interface mock-ups. Core components included URL extraction and parsing modules, domain scoring mechanisms, and asynchronous API-based phishing detection. The user interface was designed as a Firefox WebExtension popup, providing simple alerts, status badges, and notifications accessible to both technical and general users.

3.4 Phase 4: Implementation

The implementation phase transformed the design into a functional prototype using JavaScript technologies. The backend was built with Node.js and Express to serve as a proxy, managing requests to external phishing detection APIs securely. The frontend was developed as a Firefox WebExtension, employing HTML, CSS, and JavaScript to monitor active browser tabs, perform real-time URL scanning, and update UI elements dynamically.

Asynchronous calls, polling mechanisms, and browser notifications were implemented using WebExtension APIs to ensure timely and user-friendly alerts. The architecture avoided Python and traditional database systems, instead utilizing event-driven JavaScript and API integrations for data handling and threat detection.

3.5 Phase 5: Prototype Evaluation

Prototype testing was conducted with a selected sample group consisting of professionals from various organizations, including staff from Telekom Malaysia (TM), IT support personnel from SMK Temerloh Jaya, and employees from Sapura. The goal was to evaluate the system's detection accuracy, interface usability, and real-time performance in realistic environments. The testing process included survey responses and informal interviews, offering comprehensive insights into how well the system met user expectations. Feedback revealed several strengths, such as accurate detection of phishing and malware sites, as well as areas for improvement, particularly regarding response speed and user interface clarity. Users emphasized the importance of faster processing times and a more streamlined, intuitive design. These insights guided further refinement of the system to ensure it was not only technically sound but also user-friendly and practical for everyday use.

3.6 Phase 6: Refinement

Based on evaluation feedback, refinements were made to improve detection algorithms by optimizing API request handling and result polling. The user interface was simplified for greater accessibility, and technical bugs were resolved to ensure stable operation. The final product fulfilled all key functional and user experience objectives, preparing it for deployment

4. Analysis and Design

4.1 System Development Workflow

Table 2 illustrates the workflow of the prototype methodology applied to this project.

Table 2 Prototype methodology

Phase	Task	Output
Planning	1. Define project scope and goals: • Real-time URL analysis. 2. Identify core system functionalities based on the functional requirements. 3. Create project timeline	- Gantt Chart - Project Proposal
Analysis	1. Research existing solutions and identify gaps. 2. Document system requirements: • Functional (client-side URL validation, ML model integration). • Non-functional (e.g., real-time performance, scalability). 3. Specify tools and APIs: • CheckPhish.	- Literature Review - Detailed Requirements Document - System Flowchart
Design	1. Create system architecture based on requirements. 2. Develop Data Flow Diagrams (DFD) and Entity-Relationship Diagrams (ERD). 3. Design frontend: • Input form for URL submission. • Dynamic results display. 4. Plan backend: • API integrations. • Database schema for analyzed URLs.	- Architecture Diagram. - DFD and ERD.
Implementation	1. Develop the frontend using JavaScript & React.js 2. Build backend with Node.js and Express.js: • Implement URL analysis and feature extraction. • Integrate APIs for reputation checks.	- Executable Prototype. - Codebase (Frontend and Backend). - Functional Prototype.

Table 2 *(Cont)*

Phase	Task	Output
Prototype Evaluation	1. Perform end-to-end testing of core features including URL input validation, API-based threat detection, and UI behavior. 2. Evaluate CheckPhish API's response accuracy by comparing flagged results against known test URLs. 3. Conduct User Acceptance Testing (UAT) with professionals from TM, Sapura, and SK Tanjung to assess usability and clarity.	- User Reports. - Identified Improvements.
Refinement	1. Optimize system performance 2. Enhance UI/UX based on user feedback. 3. Fix bugs and finalize features: • Risk scoring display. • Actionable recommendations.	- Finalized System - Benchmark Results

4.2 System Requirements

The system requirements are divided into two categories to provide a comprehensive overview. Functional requirements (Table 3) ensure that the proposed modules effectively perform their intended tasks, while non-functional requirements (Table 4) address security, usability, scalability, and compatibility to meet user expectations.

Table 3 *Functional Requirements for Suspicious Link Detection System*

Module	Functional Requirement
URL Analysis	The system must extract key features such as URL length, special characters, and query parameters.
Security	The system must conduct reputation checks via APIs such as CheckPhish
Classification	The system must classify URLs as "Safe" or "Suspicious" using a machine learning model.
Logging	The system should log analyzed URLs and their results for future reference and system improvement.

Table 4 *Non-Functional Requirements for Suspicious Link Detection System*

Non-Functional Requirement	Description
Usability	The system should provide a simple interface with minimal technical knowledge required for operation.
Scalability	The system must handle multiple URL submissions simultaneously without performance degradation.
Compatibility	The system should operate seamlessly on major web browsers across both desktop and mobile platforms.

4.3 System Analysis

The system analysis employed a structured approach using a Context Diagram, Data Flow Diagrams Level 1, and a System Flowchart to ensure the software's design aligns with its objectives. The Context Diagram outlines overall functionality, showing interactions between users, administrators, and the system. The Data Flow Diagrams detail core processes like URL submission, analysis, and result reporting, while the System Flowchart illustrates the sequential flow of operations, including user login, URL analysis, and result. These visual tools provide a clear and concise blueprint for development, ensuring the system meets functional and user requirements.

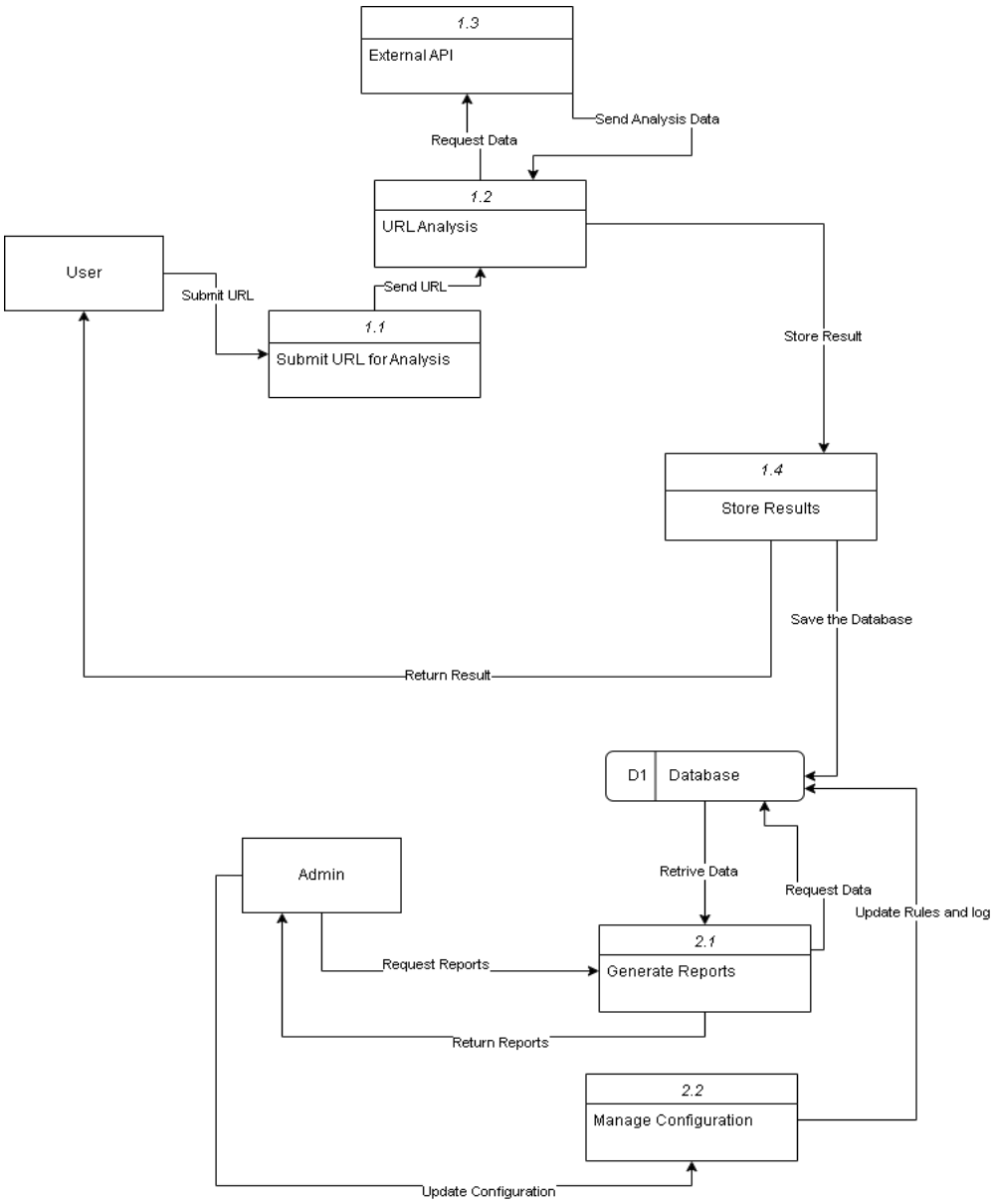


Fig.2 Data Flow Diagram (DFD) Level 1

Fig. 2 illustrates the Level 1 Data Flow Diagram (DFD) for the Suspicious Link Detection System, highlighting interactions among Users, Admin, External APIs, and the Database. Users submit URLs for evaluation and receive results based on system analysis and supplementary data from external APIs, such as domain reputations. The admin, in this case, is the system owner or developer, who manages the system's operation. As the Admin, you are responsible for configuring and managing detection protocols, reviewing logs, updating detection rules, and ensuring secure system configurations. You also oversee the integration with external APIs, using your own API account credentials to access external data sources for accurate and up-to-date threat analysis. The database securely stores analyzed URLs, user inputs, and outcomes, ensuring efficient data flow and safeguarding system integrity while maintaining privacy and accuracy.

Fig. 3 illustrates the administrator workflow within the Suspicious Link Detection System. As the authorized API account holder, the admin (API account holder) logs in to access the admin dashboard. Upon successful login, you can perform key actions such as updating detection rules, monitoring flagged links, and reviewing user activity logs. The system supports reviewing suspicious URLs, tracking user actions, and generating comprehensive reports. If login credentials are invalid, the system displays an error message denying access. Once administrative tasks are complete, the session concludes, and the admin logs off securely.

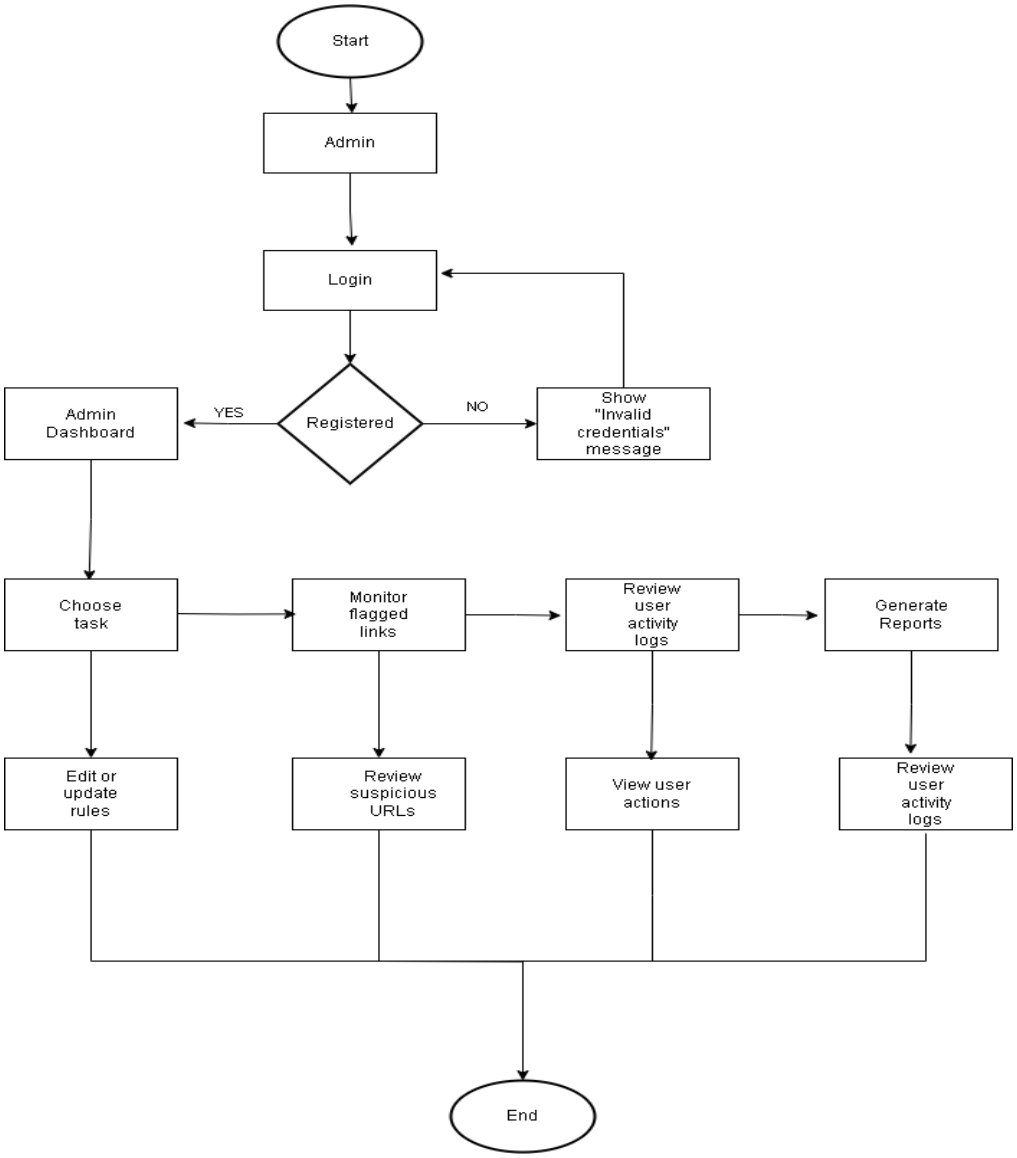


Fig.3 FlowChart for Admin

Fig.4 illustrates the workflow of the Phishing Detector extension. The process begins when a user navigates to a website. The system monitors the URL for any changes and extracts the domain. It then checks the domain against a local blacklist. If the domain is not found in the blacklist, it sends the URL to a backend proxy for further scanning. After receiving the scan results, the system evaluates whether the result is suspicious or clean. Depending on the scan results, the system will display an appropriate warning: safe, suspicious, or phishing warning. The flowchart visualizes the decision-making process that the extension follows to protect users from phishing websites.

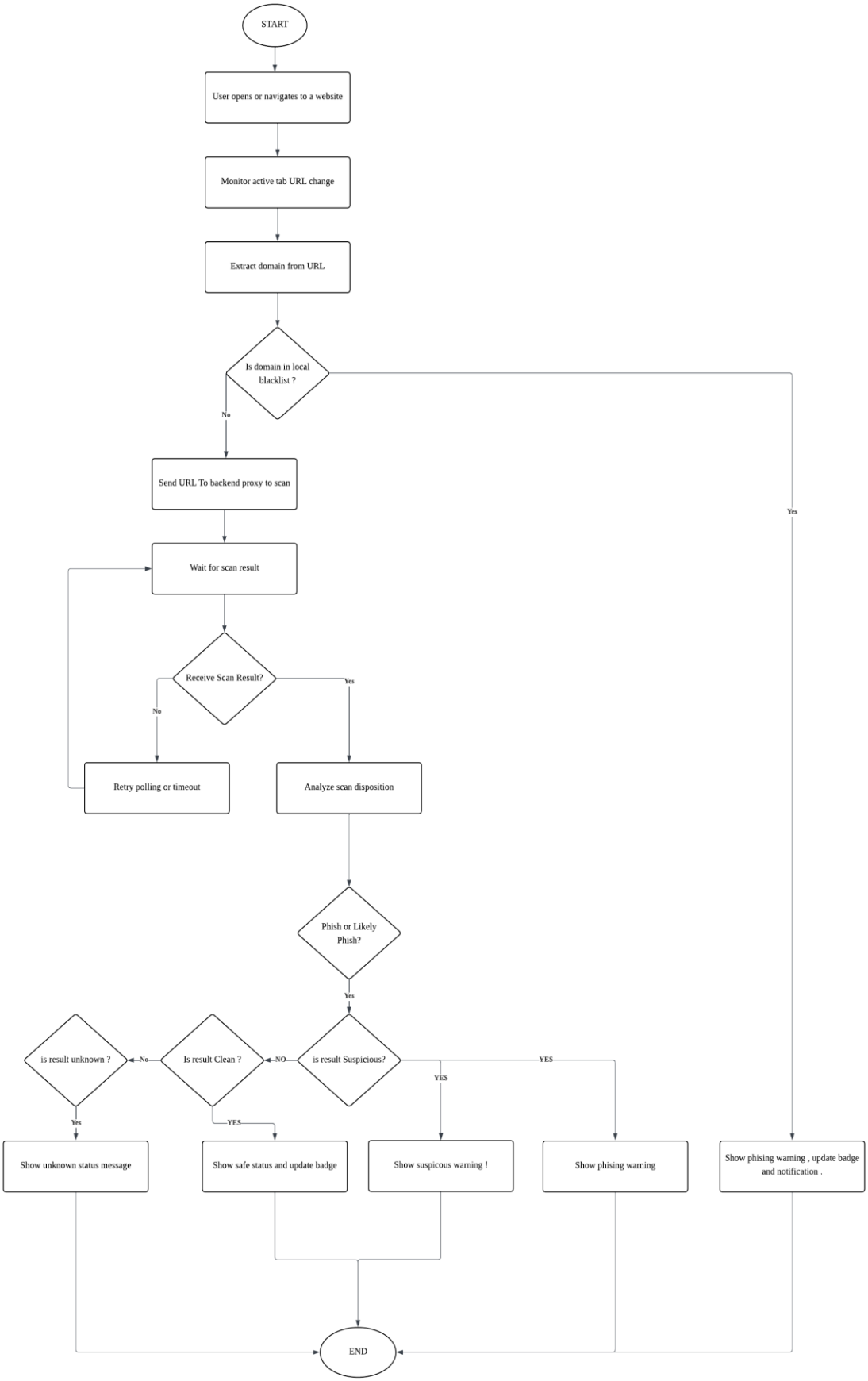


Fig.4 FlowChart for User

4.4 System Design

The system design focuses on the architecture and user interface (UI) of the Suspicious Link Detection System, ensuring a simple and user-friendly experience for all users.

4.4.1 System Architecture

System Architecture of the Suspicious Link Detection System, designed for users and administrators. Users can submit URLs for analysis and receive results indicating whether links are safe or suspicious. Administrators manage detection rules, review logs, and generate reports to ensure system performance and security.The system integrates external APIs for enhanced URL analysis, specifically the CheckPhish API [18], which provides real-time phishing detection capabilities and securely stores all data, including URLs, results, and logs, in a centralized database.

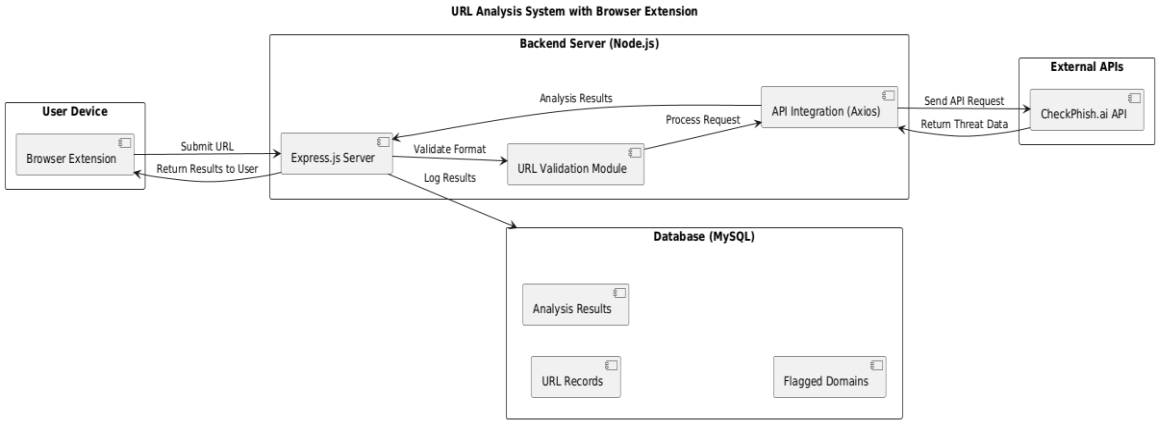


Fig.5 Suspicious Link Detection using Checkfish Architecture

Fig.5 depicts the system architecture for the Suspicious Link Detection System, which accommodates two primary roles: users and administrators. Users may submit URLs for evaluation and obtain comprehensive findings showing the safety or suspicion of the links, along with alerts for flagged URLs. Administrators oversee the system by establishing detection protocols, scrutinizing logs, and producing reports to guarantee maximum performance and security. The system engages with external APIs to retrieve supplementary reputation and threat information to improve the accuracy of URL analysis. All data, encompassing provided URLs, analytical results, and system logs, are safely housed in a centralized database, facilitating efficient operations and secure access for users and administrators.

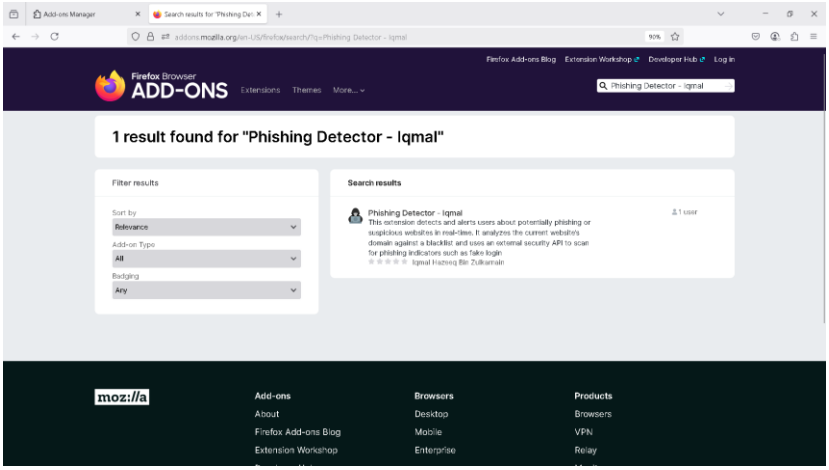


Fig.6 Firefox Extension Marketplace Listing

Fig.6 shows the Firefox Extension Marketplace listing for the Phishing Detector extension. This page presents an overview of the extension, including the developer's name and a brief description of the tool. Users can view ratings and download the extension from this page, which also provides links to other popular extensions in the marketplace.

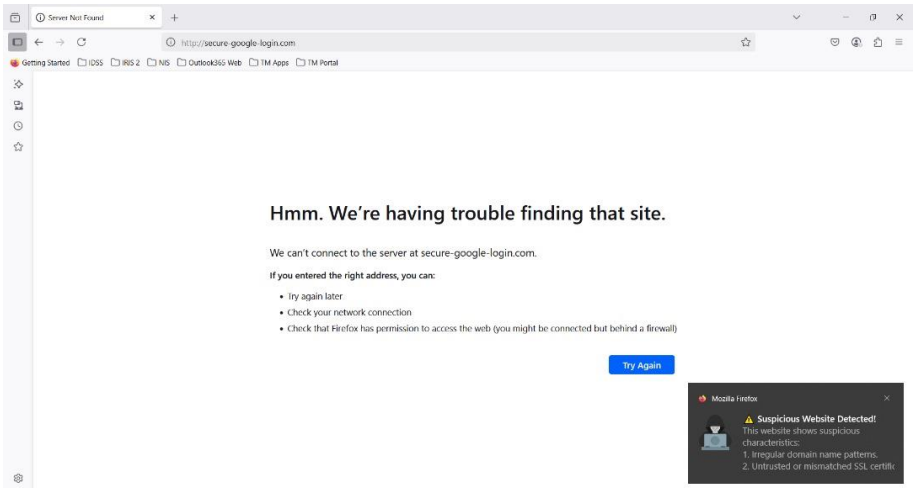


Fig.7 *Suspicious Link Page Interface Design*

Fig.7 illustrates the interface design for displaying results when a submitted link is classified as suspicious. The page features a warning message, such as “Shows signs of Suspicious Activity,” displayed with a highlighted background for emphasis.

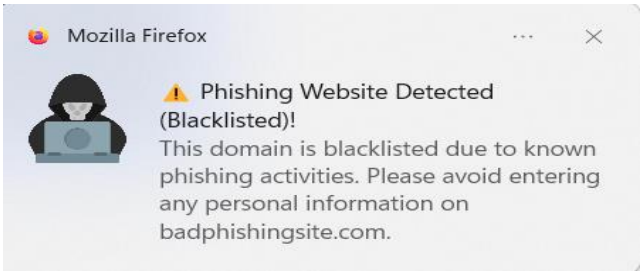


Fig.8 *Blacklist Check Module*

Fig.8 illustrates the "Blacklist Check" module, where a notification is displayed upon detecting a phishing website. The message highlights that the domain is blacklisted due to known phishing activities, advising users to avoid entering any personal information on the website.

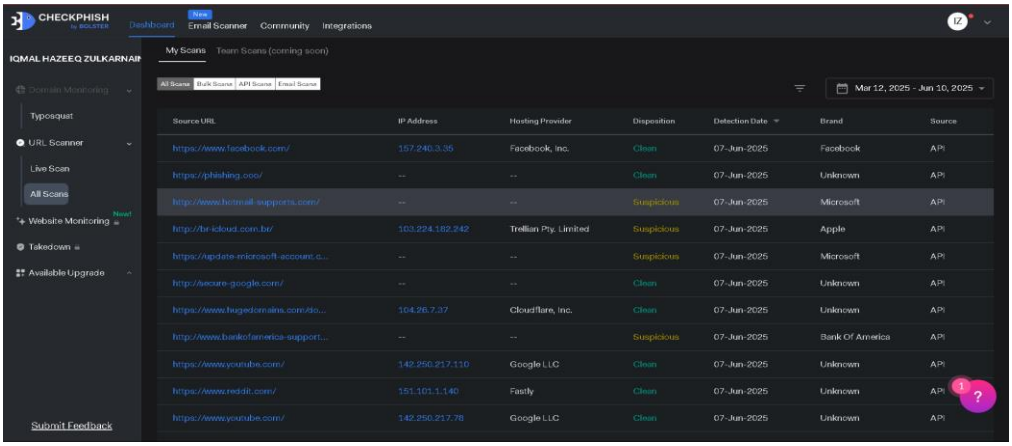


Fig.9 CheckPhish Scan Real-time History Dashboard

Fig.9 illustrates the CheckPhish Scan History Dashboard interface. In the image, when a submitted link is flagged as suspicious, it is displayed with a highlighted background to indicate that it shows signs of suspicious activity. The system includes a warning message, such as "Shows signs of Suspicious Activity," to alert users to the potential threat. The dashboard also features other details such as the source URL, IP address, hosting provider, and more, allowing users to track the results of their scan for further review.

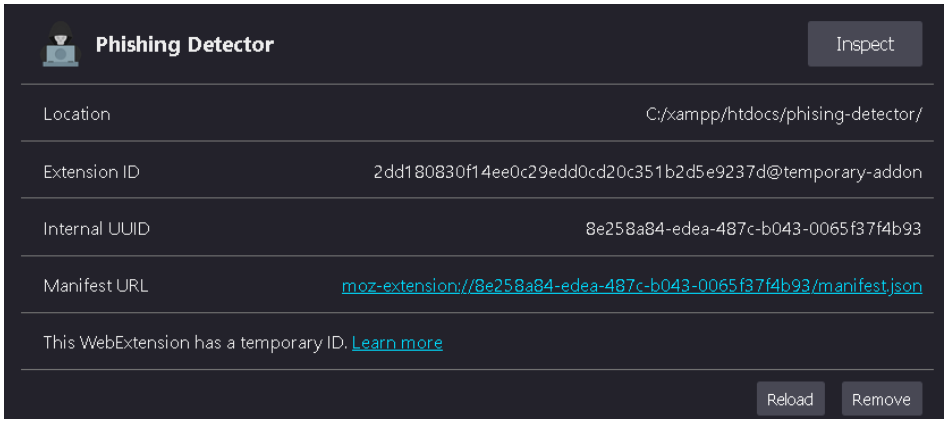


Fig 10 *Extension Settings Interface*

Fig.10 illustrates the extension settings interface for the Phishing Detector. It showcases the details of the extension's setup, such as its location, extension ID, UUID, and manifest URL. These settings allow for the inspection and management of the extension's configuration, ensuring that it functions properly. The temporary ID note indicates that the extension might not be fully integrated, highlighting that it is still in the testing phase.

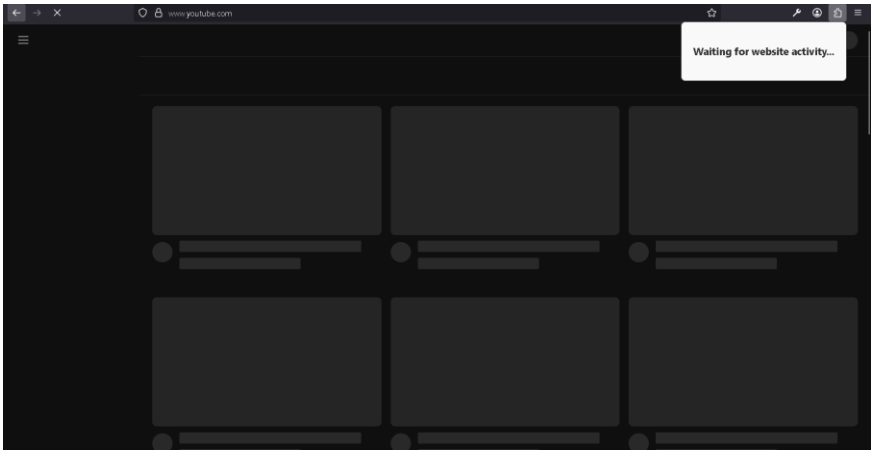


Fig.11 *Waiting for Website Activity*

Fig.11 presents the Extension Settings Interface for the Phishing Detector while the extension is running on a web page. The interface indicates a waiting status as the website activity is being monitored for potential phishing risks. This screen suggests that the extension scans websites in real time to evaluate whether the domain being accessed is blacklisted or flagged as suspicious.

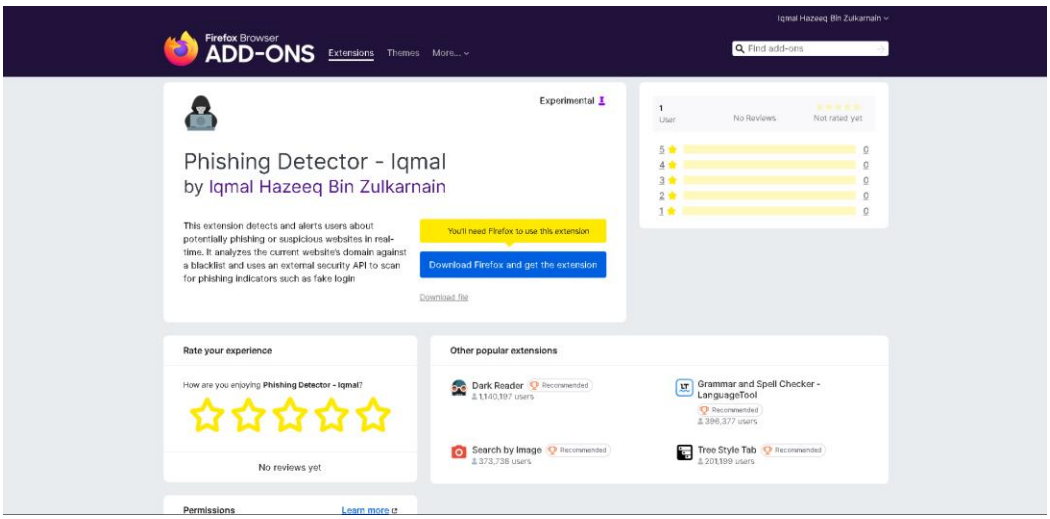


Fig 12 *Phishing Detector on Firefox Add-Ons Page*

Fig.12 shows the Phishing Detector on the Firefox Add-Ons page. This figure demonstrates the process of installing and adding the Phishing Detector extension to the Firefox browser. It highlights the page's layout, including the rating system and details about the extension's functionality. Users can view its description and decide whether to add it to their browser to help detect potential phishing websites.

```
    if (statusData.status === "PENDING") {
      console.log(`Scan still in progress: ${jobID}`);
      retries++;
      await new Promise(resolve => setTimeout(resolve, POLL_INTERVAL_MS));
      continue;
    }

    if (statusData && statusData.disposition) {
      return statusData;
    }

    console.warn("Unexpected status response:", statusData);

  } catch (error) {
    console.error("Polling error:", error.message);
    retries++;
  }

  if (retries >= MAX_RETRIES) {
    console.error(`Max retries reached for jobID: ${jobID}`);
    break;
  }

  await new Promise(resolve => setTimeout(resolve, POLL_INTERVAL_MS));
}

return statusData;
```

Fig. 13 *Polling for Scan Status*

Fig.13 shows the process of Polling for Scan Status in a JavaScript application. This figure demonstrates how the system repeatedly checks the scan status of a task identified by a specific jobID. It begins by verifying if the status is still "PENDING" and, if so, waits for a specified interval (POLL_INTERVAL_MS) before checking again. If the scan is completed (statusData.disposition exists), the process ends and returns the result. If an unexpected status or an error occurs, it logs the issue and increments a retry counter. The loop stops if the maximum number of retries (MAX_RETRIES) is reached. This mechanism ensures that the system efficiently handles asynchronous scan status updates without overwhelming the server.

```
    if (disposition === "phish" || disposition === "likely_phish") {
      title = "⚠️ Phishing Website Detected!";
      details = `This website is unsafe for the following reasons:
1. Presence of fake login forms.
2. Malicious redirects detected.
3. Suspicious IP addresses and hosting provider: ${statusData.hostingProvider}.
URL: ${url}`;
      badgeText = "!";
      badgeColor = "#d32f2f";

    } else if (disposition === "suspicious") {
      title = "⚠️ Suspicious Website Detected!";
      details = `This website shows suspicious characteristics:
1. Irregular domain name patterns.
2. Untrusted or mismatched SSL certificates.
3. Hosting provider: ${statusData.hostingProvider} (${statusData.ipAddress}), which is unusual for this domain.
URL: ${url}`;
      badgeText = "?";
      badgeColor = "#f57c00";

      chrome.runtime.sendMessage({ action: "show-confirmation", details: details });

    } else if (disposition === "clean") {
      title = "✅ Website is Safe!";
      details = `No threats detected.
Hosting Provider: ${statusData.hostingProvider}
IP Address: ${statusData.ipAddress}
URL: ${url}`;
      badgeText = "✓";
      badgeColor = "#2e7d32";
```

Fig. 14 *Classifying the Website's Disposition (Phishing Detection)*

Fig.14 shows the process of classifying a website's disposition for phishing detection. Based on the scan results, the code categorizes websites as phishing, suspicious, or clean. If marked as phishing, it displays a red badge and lists risks like fake login forms or malicious redirects. Suspicious websites receive an orange badge and are flagged for irregular patterns, such as mismatched SSL certificates or unusual hosting providers. If no threats are found, the site is marked clean with a green badge. This classification helps alert users to potential dangers and enhances online browsing safety.

```
require('dotenv').config();
const express = require('express');
const fetch = require('node-fetch');
const cors = require('cors');

const app = express();
app.use(express.json());
app.use(cors());

const API_KEY = process.env.CHECKPHISH_API_KEY;

app.post('/scan-url', async (req, res) => {
  const { url } = req.body;
  if (!url) return res.status(400).json({ error: 'No URL provided' });

  try {
    const response = await fetch('https://checkphish.ai/api/v1/url/scan', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'API-Key': API_KEY,
      },
      body: JSON.stringify({ url }),
    });

    if (!response.ok) {
      return res.status(response.status).json({ error: response.statusText });
    }

    const data = await response.json();
    res.json(data);
  } catch (error) {
    res.status(500).json({ error: 'Failed to call CheckPhish API' });
  }
```

```
if (!domain) return;

console.log(`Checking URL: ${url}`);


if (BLACKLIST_DOMAINS.includes(domain)) {
  const message = `This domain is blacklisted due to known phishing activities. Please avoid entering any personal information o
  sendMessageToPopup("phishing-detected", "⚠️ Phishing Website Detected (Blacklisted)!", message);
  updateBadge("!", "#d32f2f");
  showNotification("⚠️ Phishing Website Detected (Blacklisted)!", message);
  isProcessing = false;
  return;
}

const apiKey = "6vbg6xx1efiwt6f54k7jyrpqz18wr134wyhumin06i499f0s822cfe2kcz43jj4x";
const apiUrl = "https://developers.bolster.ai/api/neo/scan";
const statusUrl = "https://developers.bolster.ai/api/neo/scan/status";

try {
  sendMessageToPopup("loading", "Checking the website...");


  const scanResponse = await fetchWithTimeout(
    apiUrl,
    {
      method: "POST",
```

Fig .15 *API Call Flow Diagram*

Fig.15 shows the API Call Flow Diagram used for detecting phishing websites. The top section of the figure demonstrates how an Express server receives a URL from the client and sends it to the CheckPhish API using a POST request, including the required API key and request headers. The response is then parsed and returned to the client. The bottom section of the figure shows additional checks for blacklisted domains and uses another phishing detection API from Bolster. If the domain is blacklisted, a warning is displayed. Otherwise, the domain is submitted to the Bolster API for scanning, and a status URL is prepared to check the scan result. This process ensures multiple layers of phishing detection through both blacklist matching and external threat intelligence APIs.

```
if (BLACKLIST_DOMAINS.includes(domain)) {
  const message = `This domain is blacklisted due to known phishing activities. Please avoid entering any personal information on ${domain}.`;
  sendMessageToPopup("phishing-detected", "⚠️ Phishing Website Detected (Blacklisted)!", message);
  updateBadge("!", "#d32f2f");
  showNotification("⚠️ Phishing Website Detected (Blacklisted)!", message);
  return;
}
```

blacklist.json > ...

```
{
    "blacklisted_domains": [
        "badphishingsite.com",
        "malicious-example.net",
        "dangerous-site.org"

    ]
}
```

Fig. 16 *Blacklist Detection Alert*

Fig.16 shows the Blacklist Detection Alert mechanism that automatically protects users from known phishing websites. The top section contains JavaScript code that checks if the visited domain matches any entry in a predefined blacklist. If a match is found, the system displays a phishing alert, updates the extension badge, and shows a notification to the user. Additionally, the user is automatically kicked out or redirected away from the malicious website to prevent interaction. The bottom section displays the blacklist.json file, which stores the list of blacklisted domains such as badphishingsite.com, malicious-example.net, and dangerous-site.org. This setup ensures real-time phishing protection by combining blacklist matching with automated user blocking.

5. Testing

System testing ensures that all components of the Suspicious Link Detection system work as intended and meet the requirements specified in the design. This section covers functional testing and user acceptance testing (UAT) for the system. The major deliverables of this phase are shown in Appendix D.

5.1 Functional Testing

Functional testing was performed to ensure that the individual components or modules of the Phishing Detection System operated correctly in isolation. The test cases for the three main modules Phishing Detection Module, Message Notification Module, and Badge Update Module were executed, and the results are summarized in the following tables in Appendix B:

- **Table B1**: Test Case Results for Phishing Detection Module
- **Table B2**: Test Case Results for Message Notification Module
- **Table B3**: Test Case Results for Badge Update Module

5.2 User Acceptance Test

User Acceptance Testing (UAT) was conducted to validate that the Phishing Detection Extension meets the expectations of its intended users and functions effectively in real-world conditions. The testing group consisted of professionals from various organizations, including staff from Telekom Malaysia (TM), IT support personnel from SMK Temerloh Jaya, and employees from Sapura. These participants provided practical insights based on their technical roles and daily exposure to online security challenges. Their feedback focused on the extension's detection accuracy, user interface design, and the clarity of warning notifications. Overall, users reported that the extension was easy to use and successfully identified phishing threats. The detailed UAT results, covering usability, speed, and functionality, are presented in the following figures and tables. Additional feedback data can be found in Appendix C.

6. Conclusion

This research presents the development of a Suspicious Link Detection Extension using the CheckPhish API, aimed at efficiently identifying and mitigating malicious URLs. The system applies real-time detection techniques based on domain reputation and URL structure analysis to accurately flag phishing and suspicious websites. Developed using JavaScript and WebExtension APIs, the extension successfully passed all functional tests and was further validated through User Acceptance Testing (UAT) involving participants from Telekom Malaysia (TM), SK Tanjung, and Sapura. Feedback from users confirmed the system's ease of use and effective detection capabilities, although some suggested improvements in response speed and user interface clarity. These insights guided the refinement of the tool to enhance its performance. Despite its effectiveness, the system currently depends on an external API, which may pose limitations if the service becomes unavailable. Future improvements may include integrating local machine learning models and extending support to other browsers such as Chrome. Overall, the extension provides a practical and accessible solution to improve online safety, especially for non-technical users.

Acknowledgement

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

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *I. H. Zulkarnain, S. K. Ahmad Khalid;* ***data collection:*** *I. H. Zulkarnain, S. K. Ahmad Khalid;* ***analysis and interpretation of results:*** *I. H.*

Zulkarnain, S. K. Ahmad Khalid; ***draft manuscript preparation:*** *I. H. Zulkarnain, S. K. Ahmad Khalid. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] P. Kumar, M. Singh, and R. Mishra, "Phishing website detection using machine learning: A comprehensive survey," *IEEE Access*, vol. 9, pp. 60006–60030, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3073623

[2] X. Zhang, J. Zhao, and Q. Liang, "URL-based phishing detection using neural network techniques," *International Journal of Network Security*, vol. 23, no. 4, pp. 656–667, 2021. [Online]. Available: https://doi.org/10.6633/IJNS.202107_23(4).07

[3] S. Gupta, A. Goyal, and M. Varshney, "Comparison of URL-based and content-based phishing website detection techniques," *Procedia Computer Science*, vol. 132, pp. 1732–1740, 2018. [Online]. Available: https://doi.org/10.1016/j.procs.2018.05.150

[4] M. Aburrous, M. Hossain, K. Dahal, and F. Thabtah, "Intelligent phishing detection system for e-banking using fuzzy data mining," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7913–7921, 2010. [Online]. Available: https://doi.org/10.1016/j.eswa.2010.04.044

[5] P. Kumar, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, "Protecting people from phishing: The design and evaluation of an embedded training email system," *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 2007. [Online]. Available: https://dl.acm.org/doi/10.1145/1240624.1240770

[6] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," *Proc. of the Sixth Conference on Email and Anti-Spam*, 2009. [Online]. Available: http://ceas.cc/2009/papers/blacklist.pdf

[7] Y. Pan and X. Ding, "Anomaly-based web phishing page detection," *Proc. of the 22nd Annual Computer Security Applications Conference*, 2006. [Online]. Available: https://ieeexplore.ieee.org/document/4338701

[8] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine learning-based Android malware detection," 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8456704

[9] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural networks," *Neural Computing and Applications*, 2014. [Online]. Available: https://link.springer.com/article/10.1007/s00521-013-1350-x

[10] G. Varshney, M. Misra, and P. K. Atrey, "A phish detector using lightweight search features," *Computers & Security*, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167404816300882

[11] G. Xiang, J. Hong, C. P. Rose, and L. F. Cranor, "CANTINA+: A feature-rich machine learning framework for detecting phishing websites," *ACM Transactions on Information and System Security*, vol. 14, no. 2, 2011. [Online]. Available: https://dl.acm.org/doi/10.1145/2043628.2043631

[12] W. Zhang, Q. Jiang, L. Chen, and C. Li, "Two-stage ELM for phishing web pages detection using hybrid features," *World Wide Web*, vol. 20, 2017. [Online]. Available: https://link.springer.com/article/10.1007/s11280-017-0461-6

[13] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection: A recent intelligent machine learning comparison based on models content and features," *IEEE Symposium on Signal Processing and Information Technology*, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/7091564

[14] Y. Fang, X. Li, J. Zhang, and J. Hong, "Enhancing URL-based phishing detection using a deep neural network," *Journal of Internet Services and Information Security*, vol. 10, no. 3, 2020. [Online]. Available: http://jis.isis.poly.edu

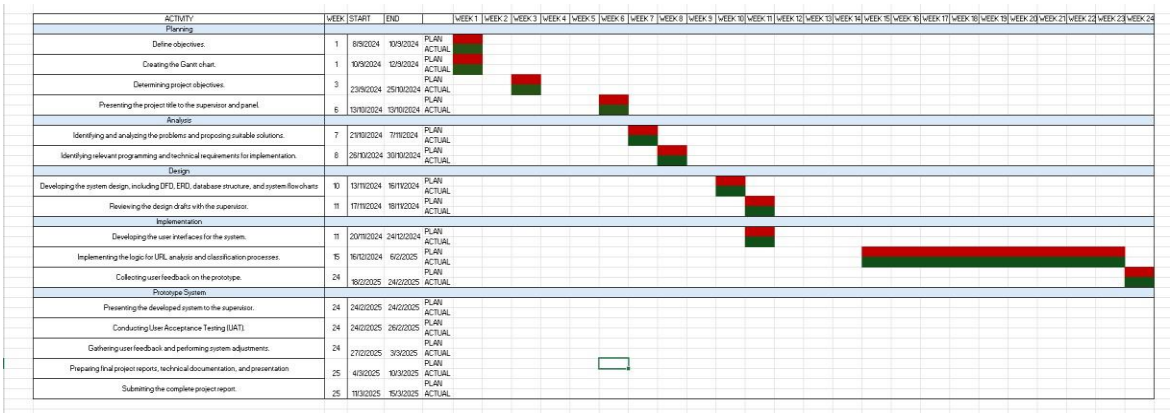
[15] URLCheck.io. (n.d.). "URL analysis for potential threats." [Online]. Available: https://www.urlcheck.io

[16] URLVoid. (n.d.). "URL reputation analysis using public databases." [Online]. Available: https://www.urlvoid.com

[17] ScanURL. (n.d.). "Basic URL evaluation and threat detection." [Online]. Available: https://scanurl.net

[18] CheckPhish. (n.d.). "Phishing website detection and analysis." [Online]. Available: https://checkphish.com

APPENDIX A: GANTT CHART



APPENDIX B: User Acceptance Testing Functionalities Testing Results

No	Test Case Description	Expected Outcome	Actual Outcome	Pass/Fail
1	Test detection of phishing website	Phishing website detected, and user receives a warning message	Phishing website detected with warning message displayed	Pass
2	Test detection of suspicious website	Suspicious website detected, and user receives a cautionary message	Suspicious website detected with caution message displayed	Pass
3	Test detection of clean website	Clean website detected with no warning message displayed	Clean website detected with no warning message	Pass
4	Test UI update when phishing site is detected	Badge colour changes to red, and confirmation message shown to the user	Badge colour changed to red, and confirmation message displayed	Pass
5	Test UI update when suspicious site is detected	Badge colour changes to orange, and confirmation message shown to user	Badge colour changed to orange, and confirmation message displayed	Pass
6	Test UI update when clean site is detected	Badge colour changes to green, and confirmation message shown to user	Badge colour changed to green, and confirmation message displayed	Pass
7	Test handling of invalid URLs	Error message displayed for invalid URL	Error message displayed for invalid URL	Pass
8	Test integration with background script (check URL status)	Background script successfully checks the URL status and updates UI	Background script checked URL status and updated UI accordingly	Pass
9	Test popup interaction when phishing site is detected	User can interact with the popup and decide to proceed or exit	User was able to interact with the popup and exit the site safely	Pass
10	Test blocking the website if the user declines the risk	Website is blocked and tab is closed when user declines the risk	Website was blocked, and tab was closed when user declined the risk	Pass

Table B1: Test Case Results for Phishing Detection Module

No	Test Case Description	Expected Outcome	Actual Outcome	Pass/ Fail
1	Test notification display for phishing website	User receives a notification indicating phishing website detected	Notification displayed correctly indicating phishing website	Pass
2	Test notification display for suspicious website	User receives a notification indicating suspicious website detected	Notification displayed correctly indicating suspicious website	Pass
3	Test notification display for clean website	User receives a notification indicating clean website detected	Notification displayed correctly indicating clean website	Pass
4	Test notification content for phishing site	Notification displays "Phishing website detected, proceed with caution"	Notification displayed with correct message for phishing site	Pass
5	Test notification content for suspicious site	Notification displays "Suspicious website detected, verify before proceeding"	Notification displayed with correct message for suspicious site	Pass
6	Test notification content for clean site	Notification displays "Website is safe"	Notification displayed with correct message for clean site	Pass

Table B2: Test Case Results for Message Notification Module

No	Test Case Description	Expected Outcome	Actual Outcome	Pass/ Fail
1	Test badge colour for phishing site	Badge colour changes to red for phishing site	Badge colour changed to red when phishing site detected	Pass
2	Test badge colour for suspicious site	Badge colour changes to orange for suspicious site	Badge colour changed to orange when suspicious site detected	Pass
3	Test badge colour for clean site	Badge colour changes to green for clean site	Badge colour changed to green when clean site detected	Pass
4	Test badge colour reset after detection	Badge colour resets after detection and confirmation	Badge colour reset correctly after detection	Pass

Table B3: Test Case Results for Badge Update Module