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

AITCS

Vol. 6 No. 2 (2025) 548-567
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

URLCHECK: A License Web-Based Suspicious Uniform Resource Locator (URL) Detector for Business Organization

Muhammad Syafiq Mohd Nazri[1], Nor Bakiah Abd Warif[1]*

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

*Corresponding Author: norbakiah@uthm.edu
DOI: https://doi.org/10.30880/aitcs.2025.06.02.030

Article Info

Received: 20 July 2025
Accepted: 19 November 2026
Available 30 November 2026

Keywords

Dataset, Features Extraction, Machine Learning, Suspicious, Token, Training Model, Uniform Resource Locator (URL)

Abstract

The rise in malicious URL attacks has posed a major threat to business organizations, resulting in data breaches, financial losses, and privacy concerns. In addition, business owners and employees often lack the technical knowledge or tools to determine whether a URL is safe or malicious, increasing their vulnerability to phishing attacks. The proposed tool, URLCHECK is designed through Object-Oriented Analysis and Design (OOAD) as a web-based system integrating Python backend with frontend technologies to combat suspicious URL threats targeting businesses. It employs a machine learning model (TF-IDF Vectorizer and Logistic Regression) trained on external sources to classify URLs as "Safe" or "Suspicious", enhanced by URL breakdown analysis. The system features tiered access: Unlicensed users receive basic classifications, while Licensed business users obtain detailed reports with mitigation plans and archived URL analysis. Administrators can update the training model to improve accuracy. Accessible for daily organizational use, URLCHECK empowers employees and general users to identify suspicious URLs. By integrating advanced machine learning techniques and a clear user interface, the tool aims to improve organizational cybersecurity and safeguard sensitive data.

1. Introduction

In the IT world, no Uniform Resource Locator (URL), website, or software is entirely safe. Individuals often underestimate the potential threats they may encounter. While URLs offer convenient access to information, they also present significant risks, especially when their sources are unknown or untrustworthy. Business organizations frequently receive URLs through emails, messages, or other digital platforms, which can conceal malicious intentions. Those 311 million records containing 77 million unique real-world URLs submitted to VirusTotal from December 2019 to January 2020. From that, 2.6 million suspicious campaigns, of which 77,810 were verified as malicious [1]. The challenge faced by all internet users, especially business organizations, is the inability to confidently assess whether a received URL is safe or potentially dangerous. With the increasing complexity of URL forms, including shortened or obscured links, business organization may not realize that the links they are clicking on are designed to steal information, install malware, or compromise their privacy. This paper designed URLCHECK which is A License Web-Based Suspicious Uniform Resource Locator (URL) Detector for Business Organization. The URLCHECK is developed as a suspicious URL detector tool by using a Python-based web application while the functionalities of the tool are tested with target user from business organization and general users.

As a web-based suspicious URL detection tool, URLCHECK is designed to classify URLs as either "Suspicious" or "Safe" through training model process by using machine learning. The training model process utilizes an externally sourced URL dataset, where URLs are pre-processed, converted into numerical values, and transformed into feature vectors using TF-IDF Vectorizer. A Logistic Regression model is trained and evaluated based on the dataset and saved as a training model for the suspicious URL prediction. The prediction process enables users to input new URLs, which undergo the same feature extraction and transformation before being analyzed by the trained model. The tool then classifies the URL as either "Suspicious" or "Safe" and displays the detail result, ensuring a robust and adaptive system for assessing URL safety in real-time.

This paper is structured as follows: Section 2 discusses the literature review related to URL detection process, dataset, features extraction techniques, machine learning algorithms and existing suspicious URL detection tools. Section 3 will explain the methodology used. Section 4 explains the analysis and design of the tool. Section 5 presents the implementation and testing while conclusions are drawn in section 6.

2. Literature Review

This Section explained about URL, URL detection process, dataset, features extraction techniques, machine learning algorithms, software for development tool and study of existing tools.

2.1 Component of Uniform Resource Locator (URL)

A Uniform Resource Locator (URL) is the address of a unique resource on the internet. This address enables browsers to retrieve published resources, including HTML pages, CSS documents, pictures, and more [2]. Theoretically, every legitimate URL leads to a distinct resource. The most frequent exception in practice is a URL that points to a resource that has moved or is no longer available. It is the responsibility of the web server owner to properly manage the resource, and its associated URL as the Web server handles both the resource that the URL represents and the URL itself.

Fig. 1 shows the components of URL. URL consists of several components in the URL: scheme, host, port, path, query, and fragment [3]. The first component, Scheme, defines the communication protocol used to access the resource on the internet. The Scheme specifies the protocol, such as HTTP or HTTPS—the latter providing encrypted communication via SSL/TLS for secure data exchange. The Host identifies the domain name or IP address of the server, often resolved via DNS. An optional Port number (e.g., :80 or :443) indicates the gateway for connection. The Path directs to a specific file or page on the server, resembling a folder structure. The Query, introduced by a '?', sends extra data to the server using key-value pairs (e.g., ?q=laptop). Lastly, the Fragment, marked by #, points to a specific section within a page to enhance navigation.

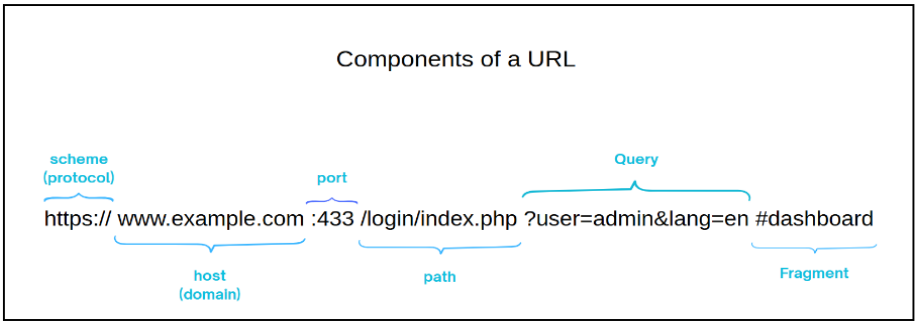


Fig. 1 *Components of a URL*

2.2 Uniform Resource Locator (URL) detection process

The Uniform Resource Locators (URLs) detection process will show detailing step-by-step workflow on how to analyze, classify, and report the safety status of a URL. The process incorporates advanced techniques such as feature extraction, machine learning, and multi-method threat detection to ensure accuracy and reliability. Fig. 2 shows one example of malicious URL detection process uses Logistic Regression (LR) to classify the target URL. This approach consists of three important modules which include data exploration (dataset), feature extraction and classification the result [3]. The process begins with data exploration, which is URL dataset from public repositories on Mendeley, PhishTank, Kaggle, and Github. The dataset will be trained and tested to make a training model. Various processes, including feature extraction and machine learning, will be trained and tested to determine the most efficient machine learning model based on the shortest testing time. The model's accuracy will be assessed, and using this performance, the process is confirmed to be selected as a training model to evaluate new input URLs whether they are malicious or safe.

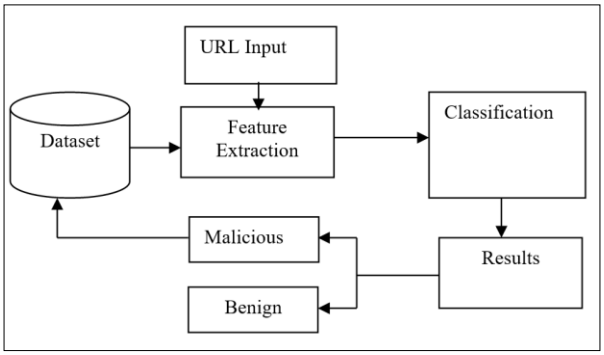


Fig. 2 *Uniform Resource Locator (URL) detection process [4]*

2.2.1 Dataset

A dataset is a structured compilation of data, organized and stored for analysis or processing. It typically contains related data gathered from a specific source or tailored for a particular project. The URL dataset was sourced from PhishTank, Kaggle, public repositories on GitHub and Mendeley Phishing URL dataset. Another dataset also can be considered if the dataset is more recent and more complex. Table 1 show the type of URL dataset.

Table 1 *Type of URL Dataset*

Dataset	Dataset Year	Description
PhishTank	Ongoing (Live)	A community-based platform that provides a live, verified feed of phishing URLs submitted by users.
Kaggle Phishing Datasets	2021	Collection of phishing URL datasets hosted on Kaggle, often used for machine learning research and classification tasks.
GitHub Public Repositories	Varies (e.g., 2020–2024)	Open-source datasets uploaded by researchers and developers containing phishing and benign URLs.
Mendeley Phishing URL Dataset	April 2024	A well-structured dataset hosted on Mendeley Data, containing features of phishing and legitimate URLs for academic research.

2.2.2 Features Extraction

Feature extraction is a pivotal process in URL classification, enabling the identification of malicious web addresses through various analytical techniques. Lexical features involve analyzing the URL's textual components, utilizing methods like CountVectorizer and TfidfVectorizer to transform text into numerical data for pattern recognition [5]. Other than that, host-based features assess the properties of the domain and hosting server, examining aspects such as domain age, IP address location, and blacklist status to evaluate the URL's legitimacy [6]. Next, domain-based features scrutinize the domain name itself, detecting manipulations like multiple subdomains, hyphens, and uncommon top-level domains that are often indicative of phishing attempts [7].

Lexical features focus on analyzing the textual structure of a URL, using libraries like CountVectorizer and TfidfVectorizer to extract meaningful patterns. The CountVectorizer converts text into a numerical format by counting the occurrence of each word in a document. While the TfidfVectorizer transforms text into numerical data by considering both the frequency of words in a specific document "Term Frequency" (TF) and how unique or rare the words are across all documents "Inverse Document Frequency" (IDF).

2.2.3 Classification using Machine Learning

Machine learning algorithms are essential in creating systems that can automatically classify URLs as safe or malicious based on learned patterns. These algorithms are trained using labelled datasets to identify features that typically appear in safe or malicious URLs. After training, the model can detect the new URLs and accurately classify them. Model evaluation is a critical step in this process to measure the performance of the algorithm. Common metrics such as precision, accuracy, recall, and F1 score help determine whether the model can generalize well to unseen data, ensuring that it is effective and reliable. By evaluating the model, areas for improvement can be identified and the system can be refined for better results [8].

Logistic Regression (LR) model is a widely used machine learning algorithm for binary classification tasks, predicting the probability that an instance belongs to one of two classes. The model works by calculating the odd of the instance belonging to the default class using a logistic function, which outputs values between 0 and 1.

2.3 Software for Development Tool

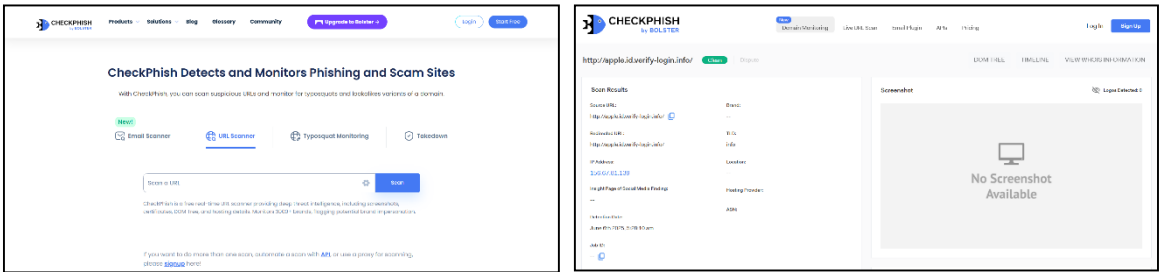
The development of web-based tools usually involves the use of customized software for both frontend and backend processes. Frontend development focuses on designing the user interface and experience using tools and languages like HTML for structuring content and CSS for styling. On the contrary, backend development is such as Python which handles dataset, algorithms and server communication. These languages enable seamless operation, ensuring the application is efficient, secure, and capable of meeting user requirements.

2.4 Study of Existing Suspicious URL Detection Tool

There are three existing tools similar to URLCHECK: CheckPhish, SiteCheck by Sucuri, and ScyScan. Each of these tools offers unique advantages that can serve as valuable references in the development of the proposed system.

2.4.1 CheckPhish

CheckPhish is a phishing detection platform that utilizes a combination of machine learning models and a large-scale phishing URL repository to analyze and classify suspicious URLs. It employs techniques such as URL lexical analysis, domain reputation checks, and heuristic feature extraction to detect phishing patterns.



(a) (b)

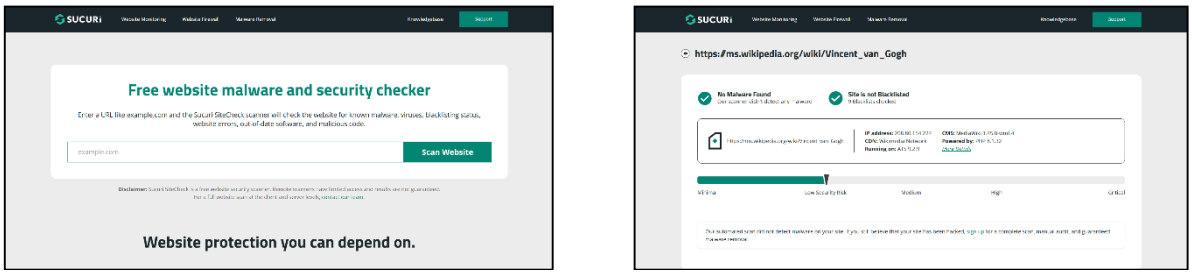
Fig. 3 *Interface of CheckPish(a) and Scanning result page with Threat Detection Summary(b)*

Fig. 3(a) illustrates the CheckPhish interface, where users can input a URL to verify its safety. The platform integrates threat intelligence feeds and real-time data sources to enhance detection accuracy, enabling rapid identification of phishing websites and protecting users from potential credential theft and fraud.

Once a URL is submitted, the scanning results are displayed as shown in Fig. 3(b). This results page indicates whether the URL is clean or malicious. In addition to the classification result, it also provides detailed information about the URL, such as its IP address, top-level domain (TLD), and a screenshot of the associated web page.

2.4.2 SiteCheck by Sucuri

SiteCheck by Sucuri is a comprehensive website security scanner that focuses on detecting malware, website vulnerabilities, and blacklisting status. It scans websites for a wide range of threats, including suspicious code injections, outdated software, and suspicious activities that could compromise a site's security.



(a) (b)

Fig. 4 *Interface of SiteCheck(a)and Scanning result page with the level of malware risk(b)*

Fig. 4(a) illustrates the SiteCheck by Sucuri interface, where users can input a URL to verify its safety. SiteCheck provides users, especially website owners and administrators, with detailed reports on security issues and actionable recommendations to help clean and protect their websites from cyberattacks.

Once a URL is submitted, the scanning results are displayed as shown in Fig. 4(b). This results page indicates in 5 scales it is minimal, low security risk, medium, high, and critical. In addition to the classification result, it also provides detailed information about the URL, such as its IP address, powered by, and content delivery network (CDN).

2.4.3 ScyScan

ScyScan leverages multi-layered URL and web content analysis, combining static and dynamic inspection methods to detect malicious behaviour. It uses sandboxing to execute suspicious code in a controlled environment and behavioural profiling to identify threats like zero-day exploits, drive-by downloads, and phishing campaigns.

(a) **(b)**

Fig. 5 *Interface of ScyScan(a)and Scanning result page with security grade(b)*

Fig. 5(a) illustrates the ScyScan interface, where users can input a URL to verify its safety. Once a URL is submitted, the scanning results are displayed as shown in Fig. 5(b). This results page indicates in grading system from A+ to T to indicate the security level of a URL after a scan. Grades A+ to A- indicate a very secure URL, B and C indicate medium risk, D and E indicate an unsafe URL with significant vulnerabilities, while F and T indicate high risk with serious threats. This display is reinforced with a colour bar from green (safe) to red (high risk). Three existing tools, CheckPish, SiteCheck, and ScyScan, were examined compared to the URLCHECK in Table 2.

Table 2 *Comparison of Existing URL Detection Tool*

Tools	CheckPhish [9]	SiteCheck by Sucuri [10]	ScyScan [11]	URLCHECK
Primary Function	Detect phishing and suspicious URLs.	Detect website blacklisting and other security issues.	Detect vulnerabilities in web applications	Detect the URL link whether Safe or Suspicious.
Technique	DOM inspection, and brand detection.	Remote site scanning and static content analysis.	Signature-based vulnerability testing and script inspection.	Address Bar-based Features and URL Breakdown Analysis
Output	Phishing detection results.	Malware detection, blacklist status, outdated software alerts.	Use a grading system from A+ to T to indicate the security level of a URL after a scan.	Detection report and Safety Status (Safe or Suspicious).

Table 2 *(Cont.)*

Tools	CheckPhish [9]	SiteCheck by Sucuri [10]	ScyScan [11]	URLCHECK
Paid Plan	**Free Plan:** - URL scanning limited to 25 URLs per day. **Premium Plan:** - More flexible and no scanning limit.	No	No	**Business Plan:** (License Users): Detailed reports with Vulnerability guidance and get access to archive URL.
Primary Function	Web-Based	Web-Based	Web-Based	Web-Based

Based on Table 2, SiteCheck by Sucuri and ScyScan are available for free and do not require users to subscribe to a paid plan for their core features. This makes them accessible to a wide audience. However, this free model also means that advanced features such as scan history or in-depth reporting are missing. On the other hand, CheckPhish and URLCHECK provide paid plans for users. CheckPish provides two main subscription options: a Free Plan, which limits users to 25 URL scans daily, and a Premium Plan, which offers unrestricted scanning capabilities and greater feature accessibility. While URLCHECK provides a Business Plan for licensed users. This plan unlocks detailed vulnerability insights, reporting, and archive features, making it suitable for enterprises that require a higher level of security assessment.

3. Methodology

This Section explained about Object-Oriented Analysis and Design (OOAD) and Tool Development Activities.

3.1 Object-Oriented Analysis and Design (OOAD)

Object-Oriented Analysis and Design (OOAD) is a systematic methodology used for developing complex systems by focusing on their structure and behaviour using object-oriented concepts. Object-oriented modelling can enhance the understanding and abstraction of classes in Unified Modelling Language (UML), underscoring OOAD's role in improving conceptual modelling fundamentals [12].

Based on Fig. 6, the Object-Oriented Analysis and Design (OOAD) process is well-suited for developing the URLCHECK tool as it emphasizes iterative development, collaboration, and adaptability to changing user needs. The methodology comprises five essential phases.

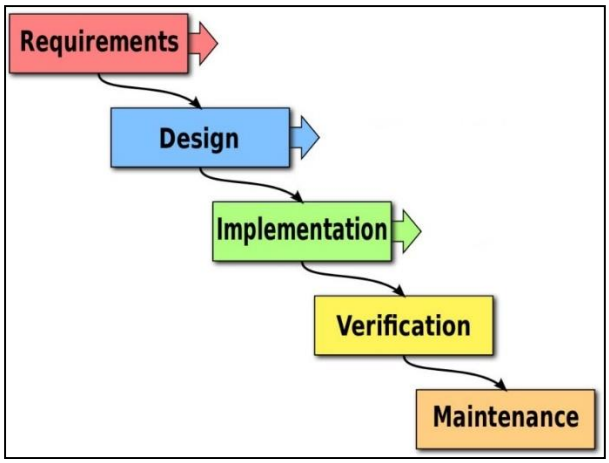


Fig. 6 *Phase in Object-Oriented Analysis and Design*

3.2 Tool Development Activities

Based on Table 3, it's the tool development activities for the URLCHECK project, structured around the key phases of Requirements, Design, Implementation, verification, and Maintenance. This workflow outlines the specific tasks and outputs associated with each phase of the development process.

Table 3 *Tool Development Activities*

Phase	Task
Requirements	1. Identify system requirements 2. Determine the dataset for training the logistic regression model and define the feature set for URL analysis.
Design	1. Design user interface. 2. Define software and hardware requirements. 3. Develop UML diagrams for system components, including use case and sequence diagrams.
Implementation	1. Design user interface. 2. Define software and hardware requirements. 3. Develop UML diagrams for system components, including use case and sequence diagrams.
Verification	1. Perform functional testing of the tool's ability to classify URLs correctly. 2. Debug and fix issues identified during testing phase.
Maintenance	1. Monitor tool performance.

4. Analysis and Design

This section explains the components of the analysis and design phase. This phase includes the functional and non-functional requirements, URLCHECK Detection Process Architecture, Use Case Diagram, Activity Diagram, Entity-Relationship Diagram (ERD) and Interface Diagram.

4.1 Functional Requirement

Functional requirements define the essential features, capabilities, and actions that the URLCEHK tool must demonstrate to achieve its intended purpose. Table 4 presents the key functional requirements crucial for the successful operation of the URLCHECK tool.

Table 4 *Functional Requirements*

No	Phase	Role	Task
1	Registration	User	• The system should allow new user to register and save the record.
2	Login	User and Admin	• The system should allow user and admin input username and password.
3	URL Scan	User	• The system should allow User to input any URL. • The system should be able to receive any URL input by user and proceed to python detection process.
4	Result Scan	User	• License users have access to view results, detailed scanning reports, and archived URLs. While Guest/Unlicensed users can only view results indicating whether a URL is safe or malicious. • The system should be able to display the analysis results from python detection process and save in user database.
5	Archive URL	User and Admin	• The system should allow user to view archived URLs. • Admin have access to deletes archived URLs. • System alerts for invalid inputs and requires confirmation for changes.
6	Profile	User	• Users can manage their profile information, including their profile picture, email address, and username. • System alerts for invalid inputs and requires confirmation for changes.
7	Forgot Password	User	• User input their email address to proceed the forgets password. • System alerts for invalid inputs and requires confirmation for changes.
8	Reset Password	User and Admin	• User input their old password, new password and confirmation password. • System alerts for invalid inputs and requires confirmation for changes.
9	Dashboard	Admin	• Overview of system activities such as Archive URLs modules, monitor Users and updating data collection.
10	Training Model	Admin	• Overview of system activities.
11	Generate Token	Admin	• Admins generate new token number for user activate their account. • User needs to enter token number when registering account.

4.2 Non-Functional Requirement

Non-functional requirements outline the key criteria used to assess the URLCHECK Tool's operations, performance, security, and usability beyond its core functionalities. These requirements are vital to ensure the tool operates effectively in various scenarios. Table 5 highlights the essential non-functional requirements that contribute to the overall quality and performance of the tool.

Table 5 *Non-Functional Requirements*

Requirement	Description
Operational	• The web-tool system can be accessed when there is internet connection.
Performance	• Users and admin should be able to access the correct assigned pages. The system can allocate the user to the correct session.
Security	• Implement encrypted method to password. • Password should at least 12 characters including at least one uppercase letter, one lowercase letter, and a special symbol. • System verifies whether a given email already exist in the system. • Implement limit Login Attempt to deter from brute-force attacks and unauthorized access. • System capable of authorizing users and through authenticating with reCAPTCHA.
Usability	• The system features a user-friendly interface designed for effortless navigation and ease of use.

4.3 URLCHECK Detection Process Architecture

The URL detection tool workflow, illustrated in Fig. 7, integrates two key processes: model training and URLCHECK tool process. The first process is training model algorithm process starts with an externally sourced dataset of URLs labeled as either "Suspicious" (1) or "Safe" (0). These URL datasets undergo preprocessing, including label conversion and feature extraction using TF-IDF vectorization, which transforms textual components into numerical vectors. The dataset is then split into training (80%) and testing (20%) sets, and a Logistic Regression model is trained. Models achieving 80% accuracy or higher are selected for Training Model. Next process, in the URLCHECK tool, new user-input URLs are processed by extracting features and applying the same TF-IDF vectorization as in training. The transformed data is passed to the trained model, which classifies the URL as either "Suspicious" (1) or "Safe" (0), providing real-time safety assessments to users. This seamless integration ensures the tool is both adaptive and accurate. The second process is the URLCHECK tool process. This process represents a URL classification system that combines machine learning prediction with rule-based checks to determine whether a URL is 'Safe' or 'Suspicious'.

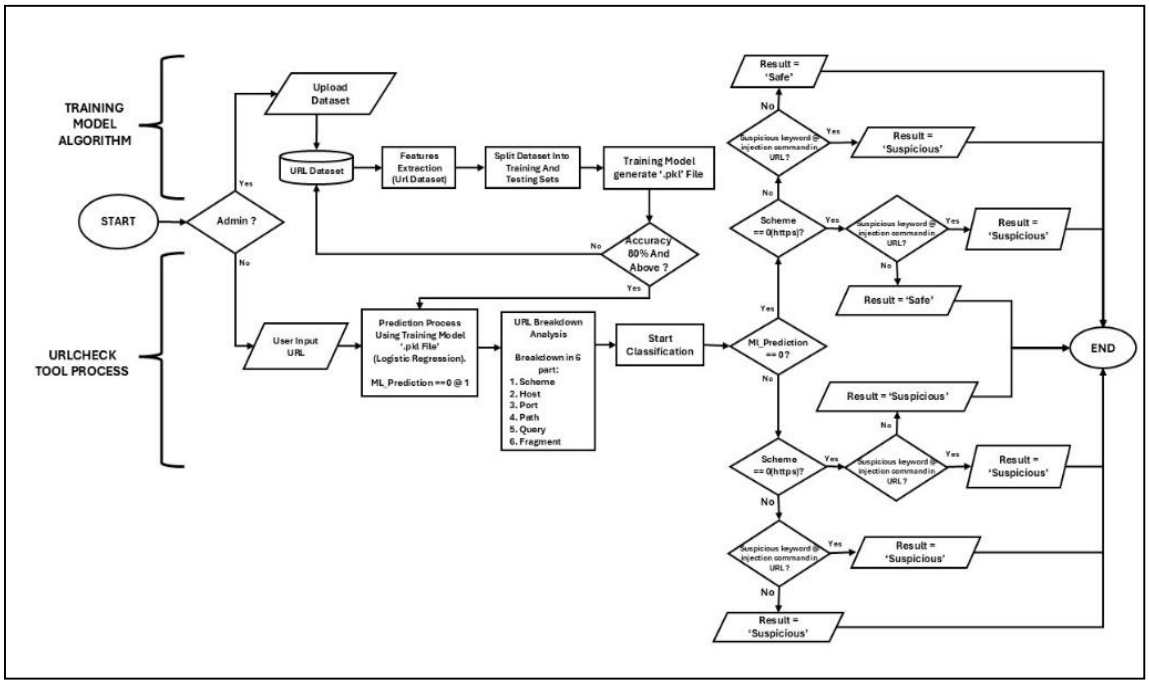
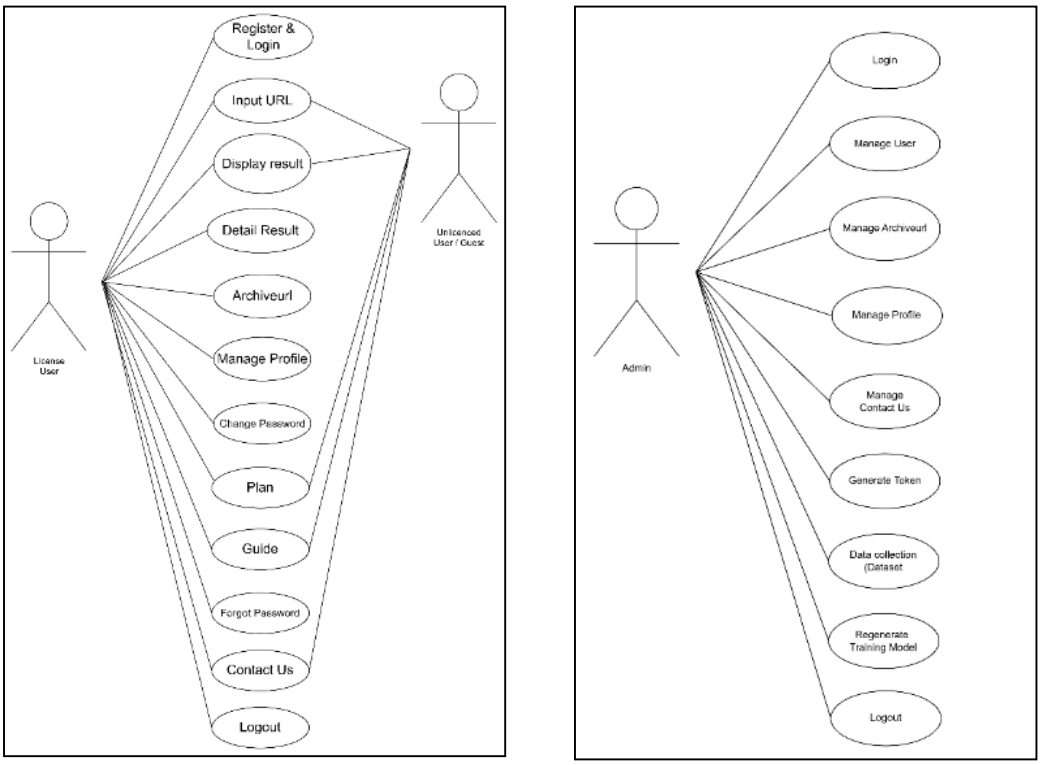


Fig. 7 *URLCHECK Detection Process Architecture*

4.4 Use case Diagram

A Use Case Diagram is a visual representation that illustrates the interactions between users (actors) and a system detailing the system's functional requirements. UML Use Cases and Class models, highlighting the importance of Use Case Diagrams in conceptual modelling and their role in bridging the gap between different modelling approaches [13]. Fig. 8(a) shows a use case diagram for user, it shows two types of users, it's licensed user and unlicensed user/guest. For Licensed Users can access all modules. Meanwhile, unlicensed users have limited access to it they cannot see the Detection Report and View URL list. Fig. 8(b) displays a use case diagram for the admin, it demonstrates available to the admin, including login, manage users, view URL list and managing profile, and logout. Admin also can update data collection (dataset) and regenerate the Training Model as a prediction in URCHECK scan process.



(a) **(b)**

Fig. 8 *User Use Case Diagram (a); Admin Use Case Diagram (b)*

4.5 Activity Diagram

The URLCHECK web-tool activity diagram in Fig. 9, illustrates the workflow for two key roles for users it's Licensed User and Guest/Unlicensed User, with each role having specific functionalities tailored to their needs. Licensed users are required to log in to access the system. To become a licensed user, the user needs to register an account that in register processes a token number must be entered to activate the account. Meanwhile, Guest\unlicensed users can access the system either with or without logging in. Guest\unlicensed users can navigate to the Paid Plan, Guide and Contact Us only. While Licensed users can navigate to Detail Result, Archive URL, Change Password and Profile. For a forgotten password, licensed users can provide their email address to receive a one-time password (OTP) sent directly to their inbox, allowing them to securely reset their password. After successfully logging in, licensed users are directed to the URL Scanner page.

The URL Scanner allows users to input URLs to predict whether they are safe or suspicious using the Logistic Regression algorithm. Based on the analysis, the tool generates a result whether they are safe or suspicious. Licensed users enjoy additional benefits, including access to archived URLs and comprehensive scan reports and the session concludes upon logout.

Fig. 10 illustrates the process flow for admin. Admin have the ability to manage user accounts, update their own profiles, and view archived URLs. Admin also has full control over the Training Model Algorithm, which includes critical functionalities such as managing the Dataset, performing Feature Extraction, and overseeing Machine Learning processes. These components work together to create and optimize the training model used for predicting URL classifications during the scanning process that will be used by users. If any updates or

modifications are required, such as updating the dataset, the admin has the authority to make these changes, ensuring the model remains accurate and up to date and the session concludes upon logout.

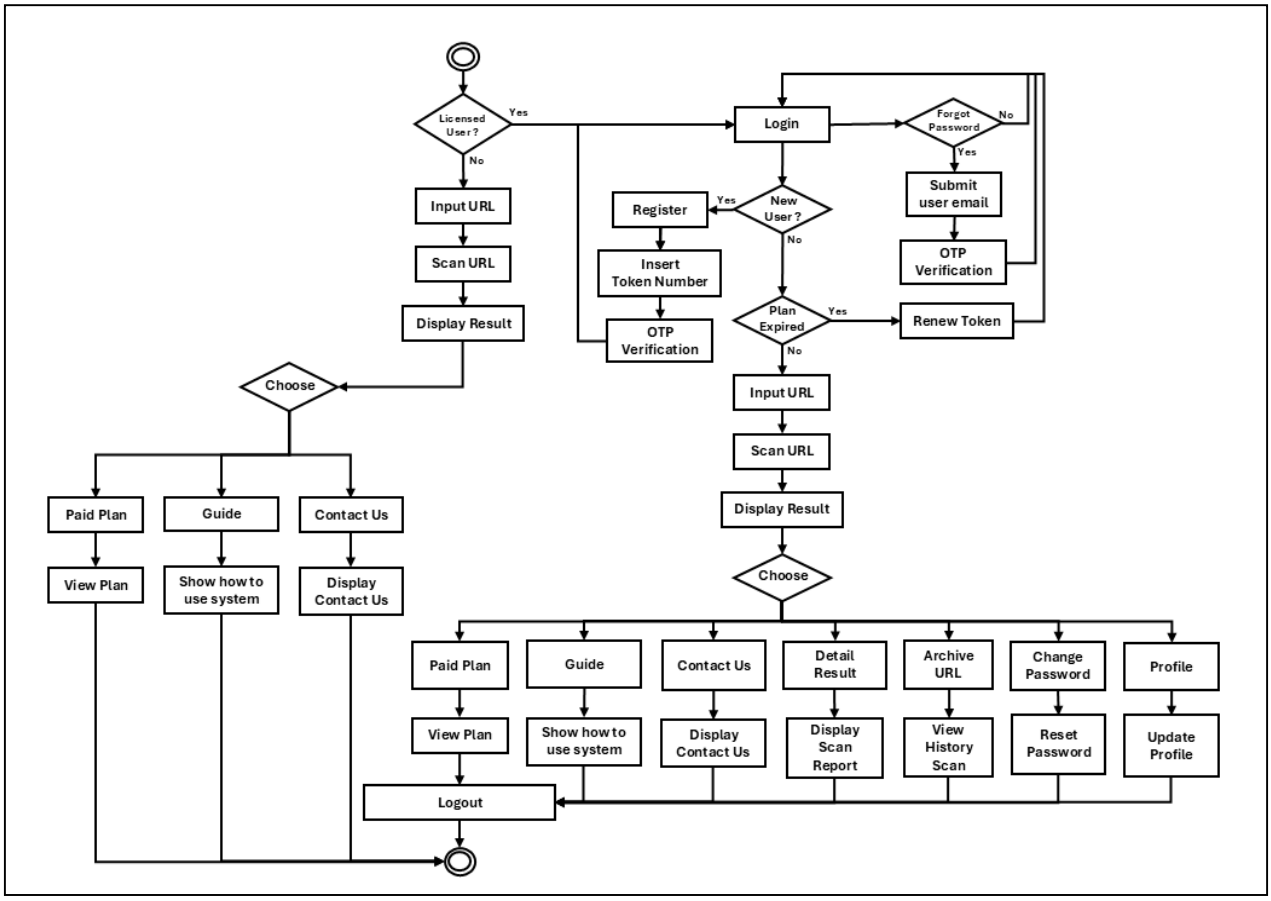


Fig. 9 User *Activity Diagram*

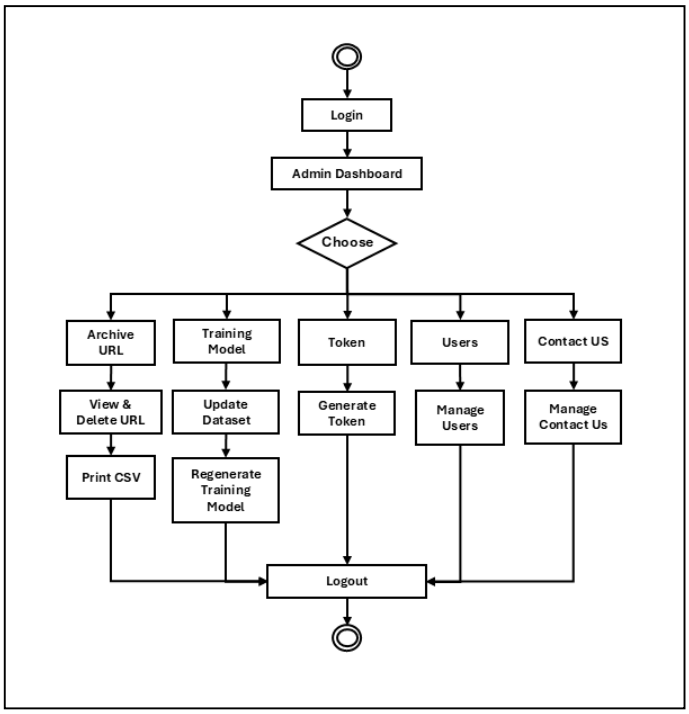


Fig. 10 Admin *Activity Diagram*

4.6 Entity Relationship Diagram (ERD)

Fig. 11 shows the Entity-Relationship Diagram (ERD) database structure for a URLCHECK system, comprising seven primary entities: "tbl_user", "tbl_admin", "tbl_scan", "tbl_urlbreakdown", "tbl_contactus", "tbl_token" and "tbl_trainingmodel". All tables are related using foreign key relationships to maintain data integrity. The tbl_user table stores user account information including usernames, emails, passwords, and profile images for system authentication. The tbl_admin table maintains administrator credentials and access controls with similar user management fields. The tbl_scan table records URL scanning activities, storing scanned URLs, IP addresses, protocols, timestamps, and classification results. The tbl_urlbreakdown table provides detailed analysis of scanned URLs, including scan parts, reasons for flagging, mitigation recommendations, and sender information. The tbl_contactus table manages user inquiries and supports requests with contact details and message content. The tbl_token table handles user session management through authentication tokens with expiration dates. Finally, the tbl_trainingmodel table stores machine learning model data including accuracy metrics, timestamps, and sender information for the URL classification system. These interconnected tables work together to provide comprehensive URL security scanning, user management, and system administration capabilities.

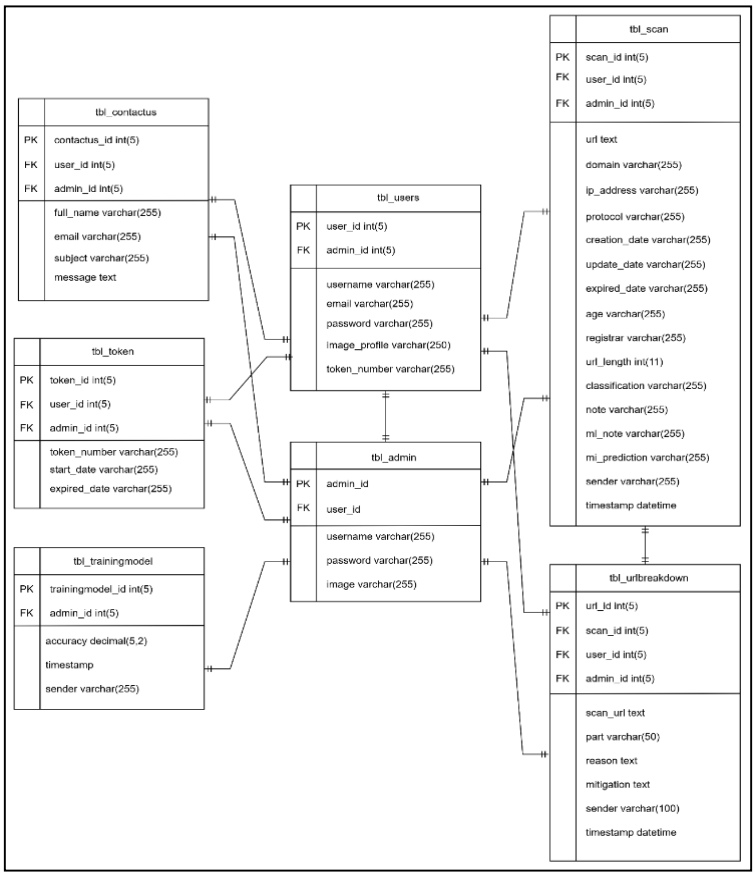


Fig. 11 *Entity-Relationship Diagram (ERD)*

4.7 Interface System and Functionality

Interface design is the process of crafting visually engaging and intuitive interfaces for software applications, websites, or any interactive platforms. It focuses on shaping the appearance, functionality, and interactivity of the interface to provide an enjoyable and seamless user experience. A successful UI design considers key factors such as ease of use, accessibility, and overall user satisfaction, ensuring that the interface is both effective and enjoyable for users to navigate.

4.7.1 Login/Sign In module

Fig. 12 depicts a user authentication interface, specifically a Sign In form. The form includes fields for Email and Password, and a prominent "Sign In" button. Below the password field, a "Forgot Password?" link provides recovery options, and at the bottom, a prompt "Don't have an account?" includes a "Sign up" link, indicating this interface serves both existing users logging in and new users needing to register.

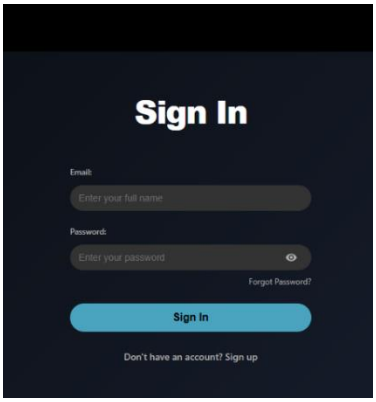
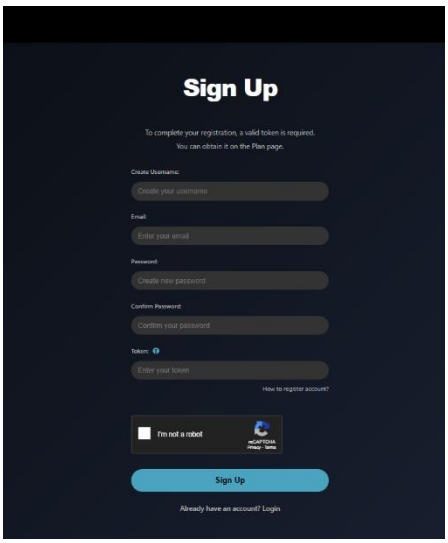


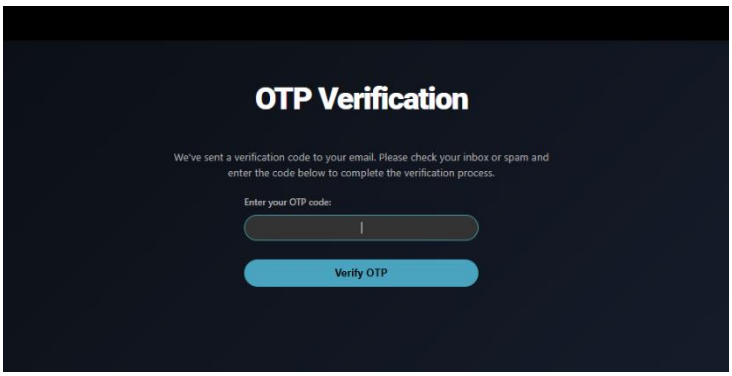
Fig. 12 *Sign In page*

4.7.2 Registration/Sign Up module

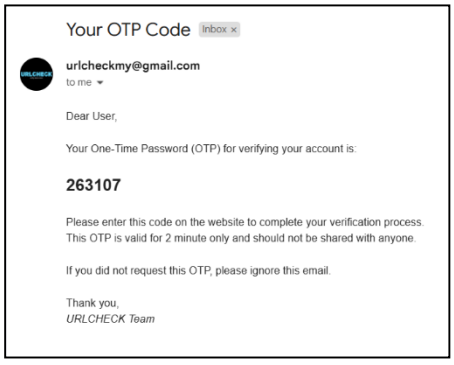
Fig. 13(a) displays a Registration/Sign Up interface where new users enter details typically including name, email, and password to create an account. Users also are required to enter a valid token number, which they must first purchase through the Plan page, in order to complete their registration and gain access to premium features. Fig. 13(b) shows an OTP verification screen that immediately follows successful sign-up. Fig. 13(c) displays the OTP (One-Time Password) code sent to the user, which remains valid for 2 minutes. In this workflow, after submitting the registration form, users receive a OTP (One-Time Password) via email, which they must enter in the verification interface to confirm their identity and complete account activation. This two-step process enhances security by ensuring the user owns the provided contact information before the account is fully registered and accessible.



(a)



(b)



(c)

Fig. 13 *Sign Up page(a); OTP Verification page (b); OTP code verification from email (c)*

4.7.3 Main Page (URL Scanner Page)

Fig. 14 presents the interface of the Main page; it's URL Scanner Page. At the top, show the navigation menus with options it's "Home", "Plan", "Profile", and "Login". Below, we see the name of URLCHECK again and some content in text would be displayed. Below that, have a search bar for users to enter a URL and a button for executing the search. This kind of layout is common on many websites and facilitates user navigation by presenting key information in a structured way.

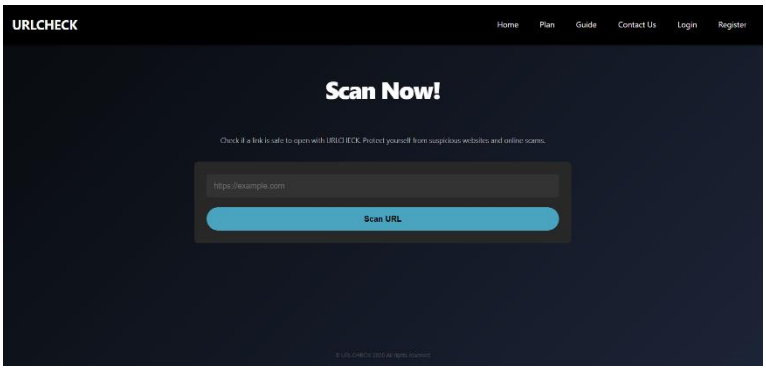
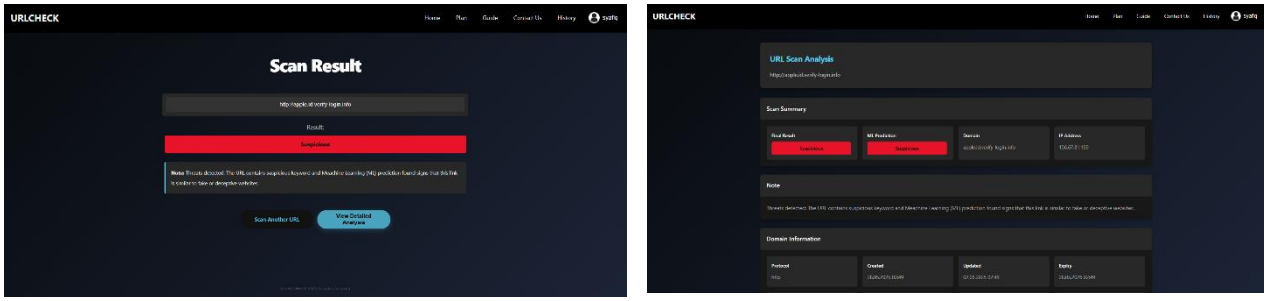


Fig. 14 *Main Page (URL Scanner Page)*

4.7.4 Result Page

Fig. 15(a) and Fig. 15(b) display the interface of the Result Page after scanning and predicting a URL. Fig. 15(a) presents a result of scanning in classification whether safe or malicious. In addition, this page will display more scanning details and reports like mitigation suggestions. This page is for Licensed users who have an advantage over guest/licensed users.

15(b) illustrates the interface for guest and licensed users. After entering a URL on the URL Scan Page, this page displays the classification result, indicating whether the URL is Safe or Malicious. Guest users have limited access compared to licensed users, who can access additional features.

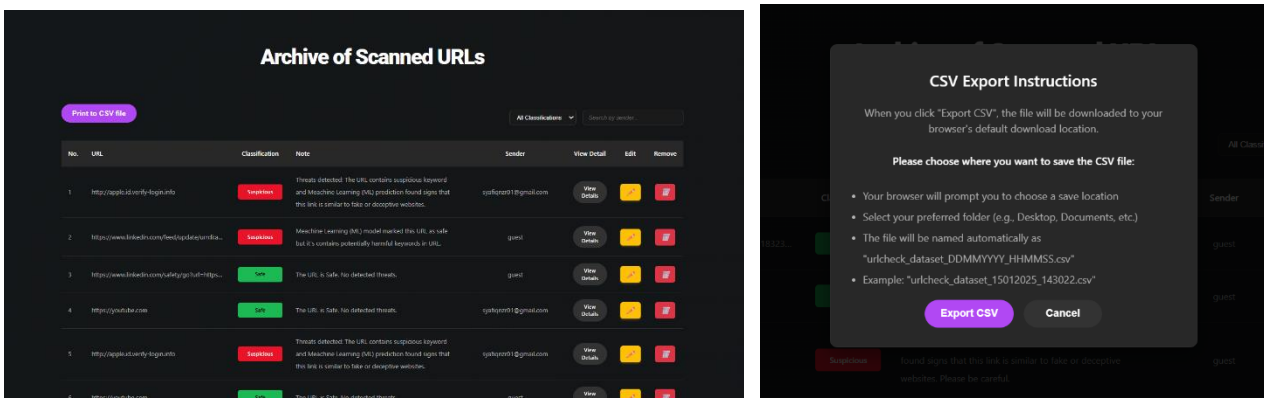


(a) **(b)**

Fig. 15 *Result Page for Licensed user (a); Result Page for Unlicensed user (b)*

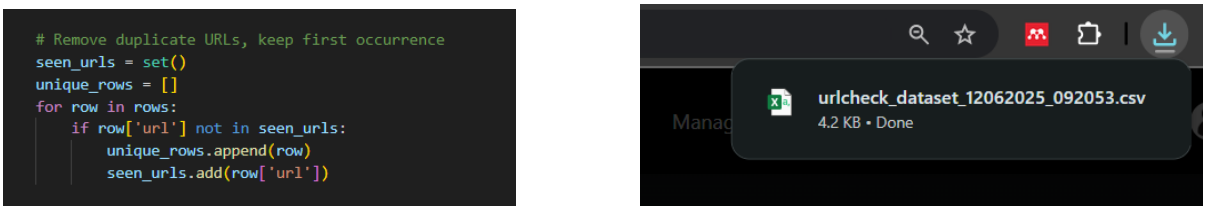
4.7.5 Admin Archive URL Page

Fig. 16(a) showcases the interface, that presenting a record of past URL security scan schedules. Each entry includes the scanned URL, the classification of "Safe" or "Suspicious", and the threat details can be viewed by clicking the "View Details" button. Fig. 16(b) shows the CSV Export Instruction popup, guiding the admin to save the scanned URL data as a CSV file. It explains the automatic filename format and the steps for exporting the file. Fig. 17(a) displays the Python code snippet responsible for eliminating duplicate URLs from the dataset. It uses a "set ()" to track seen URLs and a loop to add only the first occurrence of each unique URL to the final list. Fig. 17(b) displays the successful download of the CSV file, confirming the export process with the generated filename.



(a) **(b)**

Fig. 16 *Archive URL Page (a); CSV Export Instruction (b)*

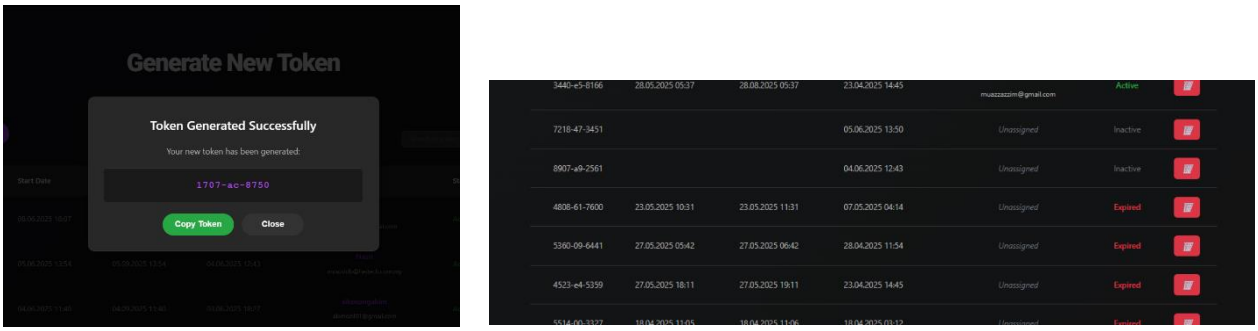


(a) **(b)**

Fig. 17 *Code of remove duplicate URL (a); The csv fie has been download (b)*

4.7.6 Admin Generate Token

Fig. 18(a) and Fig. 18(b) depicts the Admin Generate New Token, in Fig. 18(a) Admin is given access to generate the new token by clicking on the "Generate Token" button. after that a pop-up will come out and give a unique token number. Fig. 18(b) shows the status of tokens that are active, inactive and expired. Admin can also delete the token number.



(a) **(b)**

Fig. 18 *pop-up of Generate token page(a); Interface of Generate token page (b)*

4.7.7 Admin Training Model Page

Fig. 19 illustrates the interface of the Training Model Page, which is exclusively accessible to admin. Admin can print the CSV file dataset from the Admin Archive URL page as shown in Fig. 20(a) and the dataset file was successfully saved, and redundant URLs were removed from the dataset, as illustrated in Fig. 20(b). This Training model page provides essential functionality, including the ability to update and delete datasets through dedicated buttons labelled 'Update' and 'Delete'. These features allow the admin to modify or remove existing data collections to ensure the dataset remains accurate and relevant. Once the dataset has been updated, the admin can finalize the changes by clicking the 'Save Changes' button, which saves the modifications and triggers the Training Model to regenerate, incorporating the new dataset. This process ensures the model remains up-to-date and continues to deliver accurate predictions.

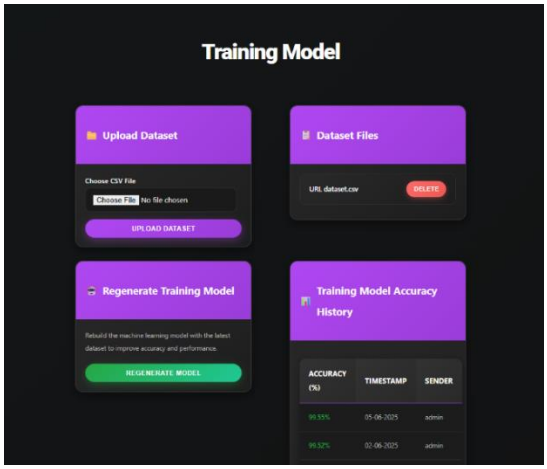
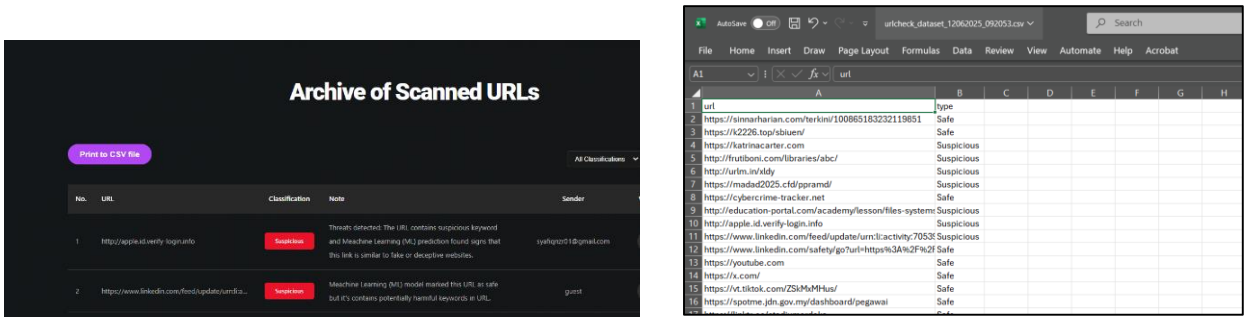


Fig. 19 *Training Model Page*



(a) **(b)**

Fig. 20 *Interface of Archive URL(a) and the content in csv file with remove redundant URL(b)*

5. Result and Discussion

This section explains about the URLCHECK Security Properties, results of Training Model Algorithm, classification, URL scanned Testing and user testing.

5.1 URLCHECK Security Properties

Fig. 21(a) shows the code for hash password by using werkzeug.security in python. It provides the generate_password_hash function, which converts a plain text password into a secure hashed string using strong algorithms like PBKDF2 with SHA-256 and a random salt, making it safe to store in databases or sessions. Fig. 21(b) shows the Hash and salt password in database. Fig. 21(c) shows the code for Validate password; this password validation script ensures strong security by enforcing latest OWASP guidelines.

```
from werkzeug.security import generate_password_hash, check_password_hash

session['registration_data'] = {
    'username': username,
    'email': email,
    'password': generate_password_hash(password),
```

(a)

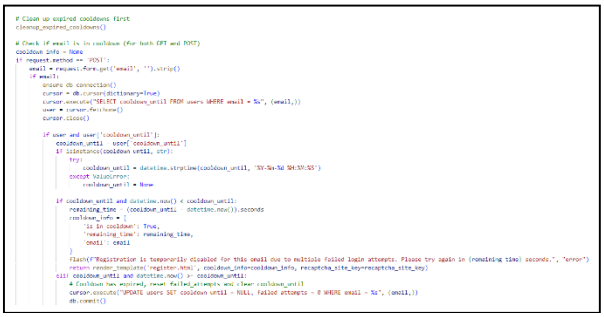
(b)

```
# Validate new password
import re
if len(new_password) < 12:
    return jsonify({'success': False, 'message': 'Password must be at least 12 characters long.'})
if not re.search(r'[A-Z]', new_password):
    return jsonify({'success': False, 'message': 'Password must contain at least one uppercase letter.'})
if not re.search(r'[a-z]', new_password):
    return jsonify({'success': False, 'message': 'Password must contain at least one lowercase letter.'})
if not re.search(r'[0-9]', new_password):
    return jsonify({'success': False, 'message': 'Password must contain at least one digit.'})
if not re.search(r'[!@#$%^&*]', new_password):
    return jsonify({'success': False, 'message': 'Password must contain at least one special character (!@#$%^&*).'})
```

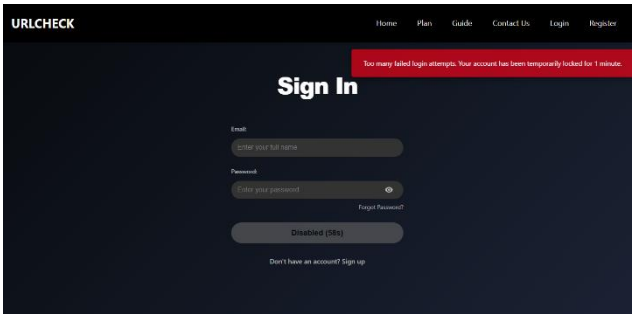
(c)

Fig. 21 *Code for hash password (a); Hash and salt password in database (b); Code for Validate password(c)*

Fig. 22(a) shows the code system includes a fixed cooldown mechanism to protect against brute-force login attempts. When a user fails to log in five consecutive times using the same email address, the system activates a cooldown period, during which login attempts are temporarily blocked for one minute. This cooldown is consistently applied after every set of five failed attempts and does not increase with repeated offenses. The system checks the cooldown status on both login attempts and page loads, allowing it to inform users in real-time if they must wait before trying again. This ensures basic security while maintaining a smooth user experience. Fig. 22(b) shows the login interface that user here cooldown login after failed login attempts 5 times.



(a)



(b)

Fig. 22 *Code for Failed Login Cooldown (a); the login cooldown in 1 minute (b)*

5.2 Training Model Algorithm Results

Fig. 23(a) illustrates the training phase of a machine learning system for URL classification, where URLs are converted into numerical features using TF-IDF vectorization, then split into training and testing sets with an 80:20 ratio (80% for training, 20% for evaluation), and used to train a Logistic Regression model to distinguish between "Safe" (label 0) and "Suspicious" (label 1) URLs. Fig. 23(b) then demonstrates the deployment phase, where the trained model and vectorizer from Fig. 5.7 are reused to classify new, unseen URLs: the raw URL is transformed using the same TF-IDF vectorizer, fed to the pre-trained model for prediction, and finally mapped to a human-readable label ("Safe" or "Suspicious"), completing an end-to-end workflow from model training to real-time inference.

```python
# Initialize the TF-IDF Vectorizer and the Logistic Regression model
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(url_list)
y = labels

# Split into training and testing dataset (80:20 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the logistic regression model
logit = LogisticRegression()
logit.fit(X_train, y_train)
```

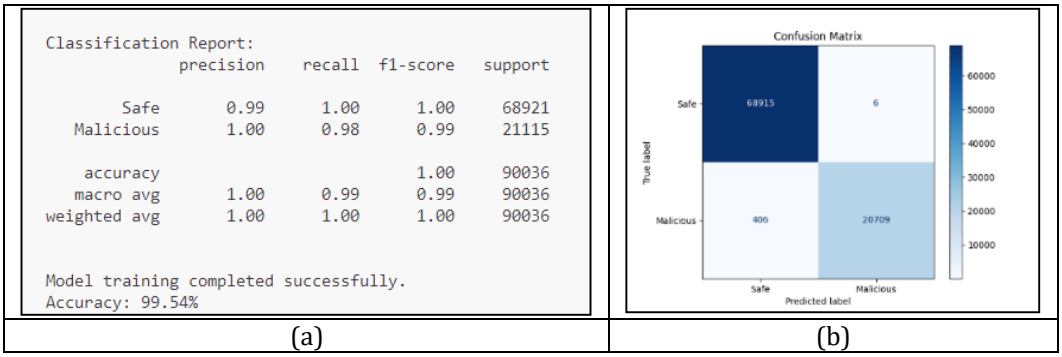
(a)

```python
# Explicit ML prediction output
url_vector = vectorizer.transform([url])
prediction = trainedmodel.predict(url_vector)[0]

if prediction == 1:
    ml_prediction = "Suspicious"

else:
    ml_prediction = "Safe"
```

(b)

Fig. 23 *Training and testing phase of a machine learning (a); Operational phase (b)*

This section presents the implementation results of a machine learning model for malware detection using the Mendeley dataset with TF-IDF vectorizer for feature extraction and logistic regression as the classification algorithm. Fig. 24(a) shows the comprehensive classification report demonstrating exceptional model performance with 99.54% overall accuracy, where the model achieved perfect precision (1.00) for both safe and malicious file classification, near-perfect recall scores (1.00 for safe files and 0.98 for malicious files), and F1-scores of 1.00 and 0.99 respectively, with the dataset containing 68,921 safe samples and 21,115 malicious samples totaling 90,036 instances. Fig. 24(b) complements these results through a confusion matrix visualization that illustrates the model's classification accuracy in a heatmap format, showing that out of the safe files, 68,915 were correctly classified as safe with only 6 false positives, while for malicious files, 406 were misclassified as safe but 20,709 were correctly identified as malicious, confirming the model's robust performance with minimal misclassification errors and validating its effectiveness for cybersecurity applications.

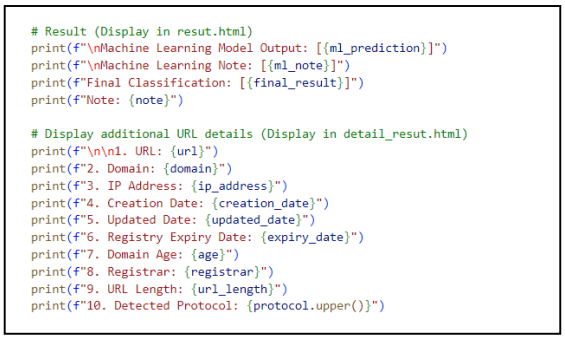


(a) (b)

Fig. 24 *Training Model Classification report (a), Training Model result in confusion matrix (b)*

5.3 Classification Result

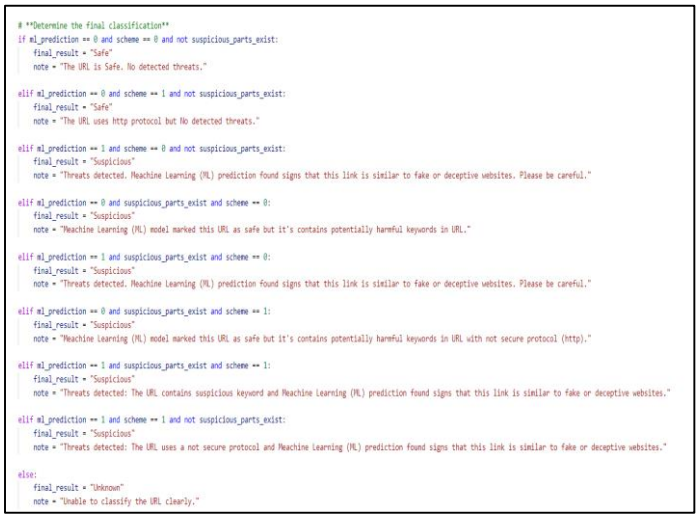
The URL classification system operates through detecting the domain whether it exists or not and detecting the protocol type. Fig. 25(a) formats the output display, presenting both a summary report (showing machine learning results, final classification, and explanatory notes in result.html) and technical URL metadata (including domain, IP, dates, and protocol details in detail_result.html). Fig. 25(b) shows the code snippet for the URL Breakdown Analysis feature, which splits a URL into six parts: scheme, host, path, port, query, and fragment. The scheme is checked for the use of "http" and marked as suspicious (1) if found. The host displays the domain type. For the remaining parts (path, port, query, and fragment), each is analyzed for suspicious keywords or injection commands; if detected, the part is also marked as suspicious (1), while Fig. 25(c) defines the classification logic, evaluating URLs using three risk factors—machine learning prediction (ml_prediction), protocol security (scheme), and suspicious keywords (suspicious_parts_exist)—to assign "Safe" status only when all indicators are benign (HTTPS + clean ML/keywords) and flag "Suspicious" for any risk (e.g., HTTP usage, ML threat, or keywords).



```
# Result (Display in resut.html)
print(f"\nMachine Learning Model Output: [{ml_prediction}]")
print(f"\nMachine Learning Note: [{ml_note}]")
print(f"Final Classification: [{final_result}]")
print(f"Note: {note}")

# Display additional URL details (Display in detail_resut.html)
print(f"\n\n1. URL: {url}")
print(f"2. Domain: {domain}")
print(f"3. IP Address: {ip_address}")
print(f"4. Creation Date: {creation_date}")
print(f"5. Updated Date: {updated_date}")
print(f"6. Registry Expiry Date: {expiry_date}")
print(f"7. Domain Age: {age}")
print(f"8. Registrar: {registrar}")
print(f"9. URL Length: {url_length}")
print(f"10. Detected Protocol: {protocol.upper()}")
```

(a)

(b)



```
# **Determine the final classification**
if ml_prediction == 0 and scheme == 0 and not suspicious_parts_exist:
    final_result = "Safe"
    note = "The URL is Safe. No detected threats."

elif ml_prediction == 0 and scheme == 1 and not suspicious_parts_exist:
    final_result = "Safe"
    note = "The URL uses http protocol but No detected threats."

elif ml_prediction == 1 and scheme == 0 and not suspicious_parts_exist:
    final_result = "Suspicious"
    note = "Threats detected. Meachine Learning (ML) prediction found signs that this link is similar to fake or deceptive websites. Please be careful."

elif ml_prediction == 0 and suspicious_parts_exist and scheme == 0:
    final_result = "Suspicious"
    note = "Meachine Learning (ML) model marked this URL as safe but it's contains potentially harmful keywords in URL."

elif ml_prediction == 1 and suspicious_parts_exist and scheme == 0:
    final_result = "Suspicious"
    note = "Threats detected. Meachine Learning (ML) prediction found signs that this link is similar to fake or deceptive websites. Please be careful."

elif ml_prediction == 0 and suspicious_parts_exist and scheme == 1:
    final_result = "Suspicious"
    note = "Meachine Learning (ML) model marked this URL as safe but it's contains potentially harmful keywords in URL with not secure protocol (http)."

elif ml_prediction == 1 and suspicious_parts_exist and scheme == 1:
    final_result = "Suspicious"
    note = "Threats detected: The URL contains suspicious keyword and Meachine Learning (ML) prediction found signs that this link is similar to fake or deceptive websites."

elif ml_prediction == 1 and scheme == 1 and not suspicious_parts_exist:
    final_result = "Suspicious"
    note = "Threats detected: The URL uses a not secure protocol and Meachine Learning (ML) prediction found signs that this link is similar to fake or deceptive websites."

else:
    final_result = "Unknown"
    note = "Unable to classify the URL clearly."
```

(c)

Fig. 25 *URL details (a); URL Breakdown Analysis(b); determine final classification(c)*

5.4 URL scanned Testing

The URL scanning test results, as illustrated and demonstrate the effectiveness of the URLCHECK system in classifying URLs as either safe or suspicious. Table 6 presents five examples of Safe URLs, where the system detected no threats and confirmed the links as secure. In contrast, Table 7 highlights five Suspicious URLs where threats were identified. These URLs contained suspicious keywords, lacked secure protocols, or showed patterns commonly associated with phishing or deceptive sites. The system's Machine Learning (ML) model detected anomalies and issued warnings based on similarities to fake or phishing web pages.

Table 6 *URL scanned testing result on Safe URLs*

No	URL	Classification	Note
1.	https://youtube.com	Safe	The URL is Safe. No detected threats.
2.	https://author.uthm.edu.my/	Safe	The URL is Safe. No detected threats.
3.	https://spotme.jdn.gov.my/dashboard/pegawai	Safe	The URL is Safe. No detected threats.
4.	https://cybercomp.uthm.edu.my/	Safe	The URL is Safe. No detected threats.
5.	https://vt.tiktok.com/ZSkMxMHus/	Safe	The URL is Safe. No detected threats.

Table 7 *URL scanned testing result on Suspicious URLs*

No	URL	Classification	Note
1.	http://apple.id.verify-login.info	Suspicious	Threats detected: The URL contains suspicious keyword and Machine Learning (ML) prediction found signs that this link is similar to fake or deceptive websites.
2.	http://education-portal.com/academy/lesson/files-systems-fat-ntfs-hfs-and-ffs.html#lesson	Suspicious	Threats detected: The URL uses a not secure protocol and ML prediction found signs that this link is similar to fake or deceptive websites.
3.	http://katrinacarter.com/old/doc	Suspicious	Threats detected. ML prediction found signs that this link is similar to fake or deceptive websites. Please be careful.
4.	http://urlm.in/xldy	Suspicious	Threats detected: The URL uses a not secure protocol and ML prediction found signs that this link is similar to fake or deceptive websites.
5.	http://frutiboni.com/libraries/abc/	Suspicious	Threats detected: The URL uses a not secure protocol and Machine Learning (ML) prediction found signs that this link is similar to fake or deceptive websites.

5.5 User Testing

The system was developed to evaluate scanned URLs and classify them as either Safe or Suspicious. To assess user satisfaction, a User Acceptance Testing (UAT) form was distributed to general users, including students, employees, business owners, and administrators. A total of 11 individuals from diverse backgrounds participated in the testing by accessing the system via the link: urlcheck.my.

Functionality Testing was conducted to ensure all features of the URLCHECK system operate correctly and meet requirements. User Functional Testing aims to verify that all key components of the system operate as intended from user perspective. This includes testing core features such as URL submission, classification result display, user registration and login, token handling, and profile-related functionality for general users. Other that hat, Admin Functional Testing aims to verify that all key components of the system operate as intended from admin perspective such as user management, data set handling, token generation, and access to full scan results. The user acceptance testing evaluation form for User Functionality and Admin Functionality system can be referred to APPENDIX. The results showed 100% agreement that the system is fully functional, proving its stability and readiness for use, especially by business organizations.

Usability Testing was conducted to assess the overall user experience of the URLCHECK system, ensuring it is intuitive, efficient, and user-friendly. The results showed that 90.9% of users rated the system a 5 on the scale for statements "The design of the system looks modern and professional" and "The system responded quickly to my actions." Additionally, 100% of users gave the highest rating for the statement "Overall, I am satisfied with the usability of this system," indicating a strong level of satisfaction with the system's design and responsiveness. The user acceptance testing evaluation form for Usability of system can be referred to APPENDIX.

6. Conclusion

In conclusion, URLCHECK has been successfully developed using the object-oriented methodology, enabling effective classification of URLs as either "Safe" or "Suspicious" through a trained machine learning model and url breakdown analysis. The Logistic Regression algorithm has the advantage of generating training models faster (4). By combining these two methods, namely the Logistic Regression algorithm together with the URL breakdown analysis mechanism, this will further increase the accuracy in classifying a URL as "Safe" or "Suspicious". For limitation, this URLCHECK absence of advanced security tools, such as Nessus, which can perform in-depth vulnerability assessments, further limits the tool's effectiveness. These developments are expected to significantly strengthen the tool's capability to accurately differentiate between legitimate and phishing URLs.

Acknowledgment

The authors express their gratitude to Universiti Tun Hussien Onn Malaysia's Faculty of Computer Science and Information Technology for its support and assistance.

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:*** *M. S. Mohd Nazri, N. B. Abd Warif;* ***data collection:*** *M. S. Mohd Nazri, N. B. Abd Warif;* ***analysis and interpretation of results*** *M. S. Mohd Nazri, N. B. Abd Warif****; draft manuscript preparation:*** *M. S. Mohd Nazri, N. B. Abd Warif. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Almashor, M., Ahmed, E., Pick, B., Abuadbba, S., Gaire, R., Wang, S., Camtepe, S., & Nepal, S. (2021). Characterizing Malicious URL Campaigns. https://doi.org/10.1145/1122445.1122456

[2] "What is a URL? - Learn web development | MDN." Accessed: Nov. 13, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL

[3] *What is a URL? - Learn web development | MDN*. (n.d.). Retrieved December 30, 2024, from https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Web_mechanics/What_is_a_URL

[4] Chiramdasu, R., Srivastava, G., Bhattacharya, S., Reddy, P. K., & Reddy Gadekallu, T. (2021). Malicious url detection using logistic regression. *2021 IEEE International Conference on Omni-Layer Intelligent Systems, COINS 2021*, 1–6. https://doi.org/10.1109/COINS51742.2021.9524269

[5] Ambata, J. S., Gaurana, J., Jacinto, D., & De Goma, J. (n.d.). Malicious URL Classification Using Extracted Features, Feature Selection Algorithm, and Machine Learning Techniques.

[6] Liu, R., Wang, Y., Xu, H., Qin, Z., Liu, Y., & Cao, Z. (2023). Malicious URL Detection via Pretrained Language Model Guided Multi-Level Feature Attention Network. https://arxiv.org/abs/2311.12372v1

[7] Hinton, G., Learning, D., Learning, D., Learning, M., Learning, M., Learning, M., Learning, M., & Learning, W. M. (2023). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow : concepts, tools, and techniques to build intelligent systems. xxv, 834 pages : https://www.oreilly.com/library/view/hands-on-machine-learning/9781098125967/

[8] Al-Fedaghi, S. (2021a). Classes in Object-Oriented Modeling (UML): Further Understanding and Abstraction. *IJCSNS International Journal of Computer Science and Network Security*, *21*(5), 139–150. https://doi.org/10.22937/IJCSNS.2021.21.5.21

[9] *Real-Time URL Checker & Fake Phishing Site Sandbox*. (n.d.). Retrieved June 12, 2025, from https://checkphish.bolster.ai/live-url-scan/

[10] Website Security Checker | Malware Scan | Sucuri SiteCheck. (n.d.). Retrieved June 12, 2025, from https://sitecheck.sucuri.net/?gad_source=1&gad_campaignid=22483687334&gbraid=0AAAAD4EI64RnECcKCv5Sd8_2h30PIvDB&gclid=Cj0KCQjw0qTCBhCmARIsAAj8C4alYuyvyYPykesdoTQyYkMY48VJFSBO4VUrakBTuRBnHnHlUd7TopsaAl_9EALw_wcB

[11] Free Web Security Scanner | Website Checker | Whois Lookup - ScyScan. (n.d.). Retrieved June 12, 2025, from https://www.scyscan.com/

[12] Al-Fedaghi, S. (2021a). Classes in Object-Oriented Modeling (UML): Further Understanding and Abstraction. IJCSNS International Journal of Computer Science and Network Security, 21(5), 139–150. https://doi.org/10.22937/IJCSNS.2021.21.5.21

[13] Al-Fedaghi, S. (2021b). TMUML: A Singular TM Model with UML Use Cases and Classes. *IJCSNS International Journal of Computer Science and Network Security*, *21*(6). https://arxiv.org/abs/2107.00757v1

Appendix