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

AITCS

Vol. 5 No. 2 (2024) 212-231
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

PhishShield: URL Phishing Detection Tool using Machine Learning Algorithm

Dayang Izzah Azizah Abang Patdeli[1], Isredza Rahmi A Hamid[1]*

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

*Corresponding Author: rahmi@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2024.05.02.012

Article Info

Received: 28 July 2024
Accepted: 16 October 2024
Available online: 15 December 2024

Keywords

Phishing, Machine Learning Algorithm, Uniform Resource Locator (URL), Random Forest, Security

Abstract

Phishing attacks pose a significant threat in the realm of cybersecurity, leveraging social engineering to steal customer identification and financial credentials. Existing phishing detection tools relies on blacklists and heuristics-based methods. However, blacklists are ineffective against zero-day phishing attacks, while heuristics has many false positives. To address challenges such as rapid Uniform Resource Locator (URL) growth, false positives, and the inability to adapt to new phishing attack, a proactive and adaptable solution is needed. The proposed tool, PhishShield is designed through Object-Oriented Analysis and Design (OOAD) and implemented in Python programming language. PhishShield consists of nine modules accessible to both admin and user roles, each based on their respective privileges. Machine learning for URL prediction involves dataset collection from GitHub and Kaggle, also feature extraction from address bar-based feature and the Random Forest algorithm with up to 83% accuracy. While the machine learning algorithm is powerful, it still produces false positives and false negatives. To mitigate this, a scoring mechanism is employed. Out of five phishing URLs tested, 40% were flagged as phishing, 40% as suspicious, and 20% as legitimate. The Random Forest algorithm identified 80% of these as malicious. Regarding the five legitimate URLs tested, 80% were correctly classified as legitimate, while 20% were marked as suspicious. The Random Forest algorithm flagged 80% as non-malicious. In summary, PhishShield's advanced detection capabilities, role-based access control, adaptability, and continuous learning offer a significant improvement over traditional phishing detection tools, making it a powerful solution in combating phishing threats.

1. Introduction

Phishing is a form of cyberattack where attackers use deceptive tactics to trick individuals into divulging sensitive information such as usernames, passwords, or financial details. Current phishing detection tools with blacklists and heuristics have drawbacks. Blacklists miss zero-day attacks, and heuristics may produce false positives, flagging safe URLs as threats [1]. The proposed tool, PhishShield, used machine learning algorithm to detect phishing URL.

The existing detection phishing tool [2] faces three major problems which are miss zero-day attacks, generates a high number of false positives when attempting to identify phishing URLs, and has limited data

availability to detect new threats. To overcome this problem, we proposed PhishShield, a tool -based for detecting phishing URLs using machine learning approach, specifically Random Forest algorithm and address bar-based feature for feature extraction. This tool is developed using Python programming language and will be designed using Object-Oriented Analysis and Design (OOAD). There are 9 modules for admin and user; Registration, Login, Dashboard, URL Scanner, URL Archive List, Profile, Change Password, Forgot Password, and User List. Security features include measures such as CAPTCHA, email verification, data encryption and URL validation. The objectives of this project are to design a tool-based tool called PhishShield to detect phishing URLs, to develop the PhishShield tool using machine learning approach, and to test the proposed PhishShield tool-based in terms of user testing and tool functionality. The benefits are aimed at end-users, ensuring a secure online experience, accurate URL predictions, reduced false positives, and adaptability to new threats. Moreover, the proposed tool is user-friendly with versatile functionalities.

The rest of the paper is organized as follows: Section 2 discusses a literature review on types of phishing attacks, components of URLs, and existing tools. Section 3 describes the Object-Oriented Analysis and Design (OOAD) methodology of proposed tool. Section 4 explains the analysis and design approach. Section 5 explains the result, discussion, and implementation of the proposed tool. Finally, the last section concludes the current work and highlights the future contribution.

2. Literature Review

This section discusses about types of phishing attacks, components of URLs, and existing tools.

2.1 Phishing

Phishing is a deceptive cyber-attack technique where attackers use fraudulent emails, messages, or websites to trick individuals into divulging sensitive information. They disguise themselves as trustworthy entities, exploiting human psychology to create a false sense of urgency or importance. This method often involves casting a wide net to reach many potential victims [3]. There are various types of phishing such as spear phishing, email phishing and web phishing.

Spear phishing is a targeted form where attackers tailor their deceptive messages to a specific individual or organization, aiming to gain unauthorized access to sensitive information or systems [4]. Email phishing is a pervasive cyber-attack method where malicious actors send deceptive emails to trick recipients into clicking on malicious links or providing sensitive information [5]. Web phishing involves creating deceptive websites that imitate legitimate ones to trick users into divulging sensitive information [6]. The significance of web phishing detection lies in safeguarding sensitive information, mitigating financial and reputational risks, and staying ahead of evolving cyber threats.

2.2 Component of Uniform Resource Locator (URL)

Uniform Resource Locator (URL) is a type of Uniform Resource Identifier (URI) that allows access to information from remote computers like web servers and cloud storage. URL consists of network communication protocol, subdomain, domain name, and URL's extension. Phishing attacks using URLs often involve tricks like misspelling words, special characters for redirection, shortened URLs, using deceptive keywords, and embedding malicious files in links [7]. URL components include the scheme, which indicates the protocol used, the subdomain, the domain, the top-level domain (TLD), port number, and path. URLs may also have components like a query string separator, a query string containing key-value pairs, and a fragment as shown in Figure 1. Understanding these components is crucial for navigating the web and creating URLs that accurately identify and access desired information on the internet.

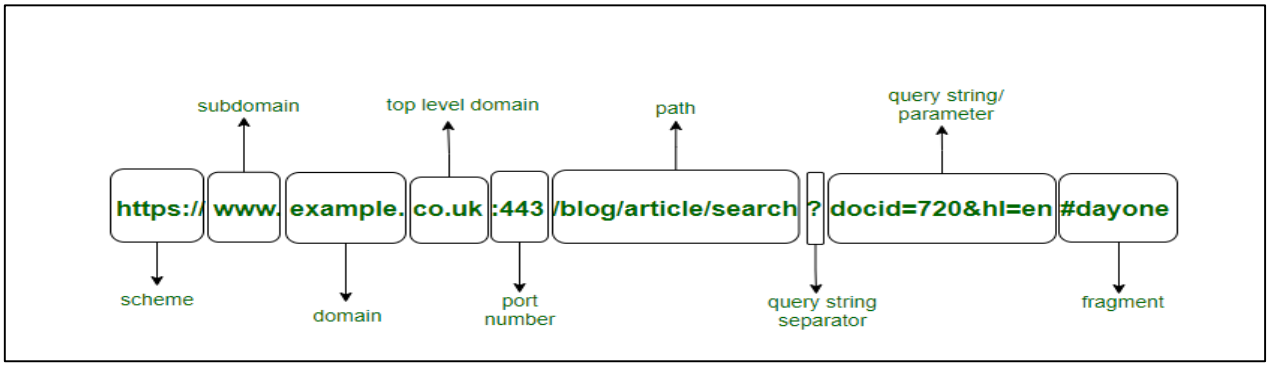


Fig. 1 *Component of URL*[8]

2.3 URL Phishing Features

The extracted features are categorized into three; Address Bar based Features, Domain based Features, and HTML and JavaScript based Features. Table 1 shows features extraction work by Phishing Attack Detection Using Machine Learning [9].

Table 1 *Categorizes in Features Extraction*

Address Bar Based	Domain Based	HTML & JavaScript Based
- IP Address in URL	- DNS Record	- IFrame Redirection
- "@" Symbol in URL	- Website Traffic	- Status Bar Customization
- Length of URL	- Age of Domain	- Disabling Right Click
- Depth of URL	- End Period of Domain	- Favicon
- Redirection "//" in URL	- Abnormal URL	- Using Popup Window
- "http/https" in Domain name	- Website Forwarding	- Server Form Handler
- Using URL Shortening Services “TinyURL”		- Anchor Tags
- Prefix or Suffix "-" in Domain		- Request Tags
- Subdomain of URL		- Link In Script Tags
- ‘https’ in the scheme		- Information Email
- Non-standard Port		
- IP Address in URL		

2.4 Algorithm and Model Evolution

Machine learning and Artificial Intelligence (AI) relies on algorithms and models for data analysis and prediction. Phishing detection tool is best done using machine learning supervised classification algorithms trained on a categorical input URL. There are three machine learning: Decision Tree Algorithm, Random Forest Algorithm and XGBoost.

Decision Tree partitions data into subsets based on key attributes, creating a tree-like structure for classification and regression. They are easy to understand and quick to train. However, decision trees can be prone to overfitting, capturing noise in the data and potentially leading to less generalizable models. Techniques like pruning are often employed to address these limitations and enhance predictive accuracy [10].

Random Forest is an ensemble learning method that builds multiple decision trees with randomized data samples and features, mitigating overfitting and enhancing generalization. Its advantages include robust performance, handling missing data well, and providing feature importance ranking. However, it may exhibit increased computational complexity with a large number of trees, and its interpretability is somewhat limited compared to a single decision tree [11].

XGBoost employing a boosting technique created a sequence of decision trees to address the errors of previous ones. With a gradient boosting architecture combining tree and linear model components, XGBoost optimizes accuracy. XGBoost offers fine-tuning settings and scalability for large datasets through parallel processing capabilities [12].

2.5 Existing Phishing Detection Tools

Many phishing detection tools have been developed on phishing email detection based on hyperlink such as [2], [13], [14], [15] and [16]. Work by [2] detected URL phishing attacks using Machine Learning and Natural Language Processing (NLP). They used URL-based features such as length and token characteristics, tokenization techniques and Internet Protocol (IP) address checking. Support Vector Machine (SVM) algorithm is used to classify URLs into safe or phishing categories, separating them using a hyperplane. The dataset used for this tool is not stated. Figure 2(a) shows the functionality of the tools is just to predict phishing URL.

Work by [14] optimized fake website detection using Logistic Regression (LR) for URL analysis. They utilized a custom tokenizer function for tokenizing URLs. The process involves loading data into a list, vectorizing URLs using Term Frequency-Inverse Document Frequency (TF-IDF) scores and performing LR for classification. The LR model is applied to binary outcomes, distinguishing between legitimate and phishing URLs. The dataset used for this tool are URL Reputation and ICML-09. Figure 2(d) shows the functionalities of the tool as follows: a “Homepage” for phishing awareness, a “Detector” page for predicting phishing URLs, a “Submit” page for submitting phishing URLs, and a “Contact” page displaying the admin’s contact information.

Worked by [13] detected phishing websites using machine learning by collecting unstructured URL dataset from sources like PhishTank, Kaggle, and Alexa. Preprocessing generates nine features, including URL length, HTTP presence, suspicious characters, prefixes/suffixes, dots, slashes, phishing terms, subdomain length, and IP address presence. The organized dataset is fed into three classifiers which are Decision Tree and Random Forest, which analyze URLs to accurately identify phishing or legitimate websites. Random Forest has the highest

accuracy for this tool. Figure 2(c) shows the functionalities of the tool as follows: a “Profile” page for user profile, “Check Website” page for predicting phishing URL, a “Feedback” page for user feedback and Logout.

Worked by [15], a website security service that scans websites for hacking signs, such as GoogleBot cloaking, spammy links, and redirection. Various criteria are used to identify potential viruses and spam links. IsItHacked also offers blacklist checks via PhishTank and Google Safe Browsing, providing users with a comprehensive view of their site's security status. Figure 2(b) shows the functionalities of the tool as follows: a first page for predicting phishing URL and several information about this tool, a “Sign up” page, a “Log in” page, a “Hacking News” page for "Hacking News" page for updating the latest cybersecurity threat news, and “Blog” page is for sharing cybersecurity awareness.

Worked by [16], a real-time URL and website scanner that uses an automated headless browser to capture live screenshots of URLs, extracting key information like natural language content, DOM, and WHOIS. This data is then analyzed using deep learning models to categorize scans into 'Phish', 'Scam', 'Suspicious', and 'Clean'. The tool distinguishes between legitimate brand pages, gift card scams, fake online stores, and potentially harmful URL patterns. The dataset used for this tool is not stated. Figure 2(e) shows the functionalities of the tool as follows: a “Domain Monitoring” page for viewing all typosquat and lookalike variants of the domain, a “Live URL Scan” page for predicting phishing URL, a “Email Plugin” page for enabling to receive real-time notification when the email address is phishing, a “APIs” page provides free API key, a “Resources” page provides blog and glossary, a “Log in” page and “Sign Up” page.

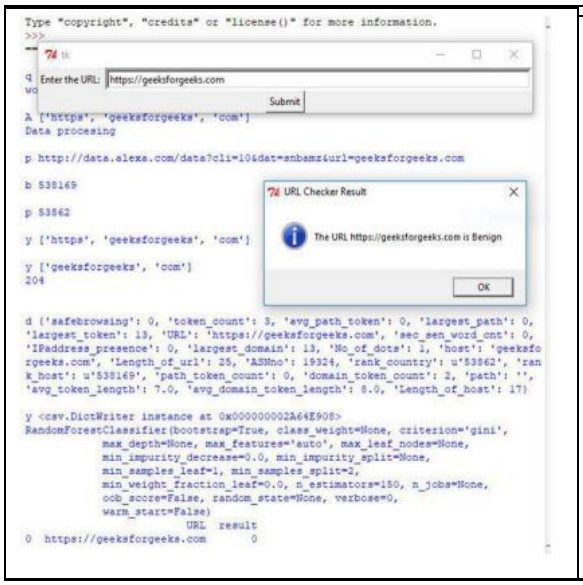


Fig. 2(a) *Interface of URL Checker* [2]

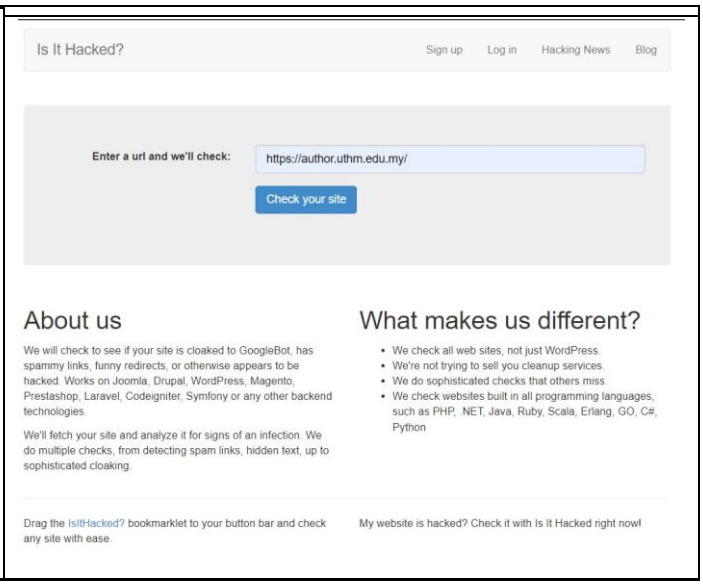


Fig. 2(b) *Interface of IsItHacked* [15]

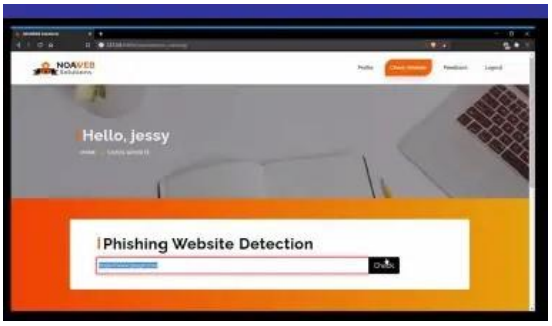


Fig. 2(c) *Interface of NoaWeb Solution*[13]

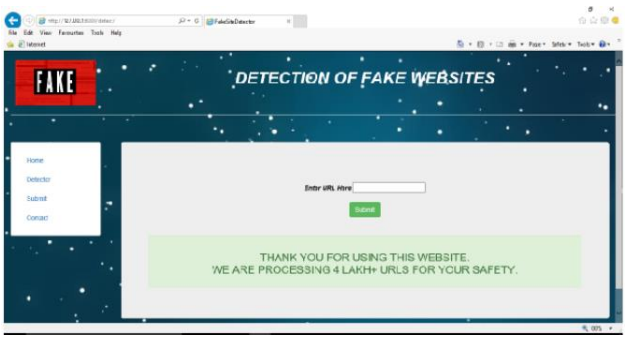


Fig. 2(d) *Interface of Detector* [14]

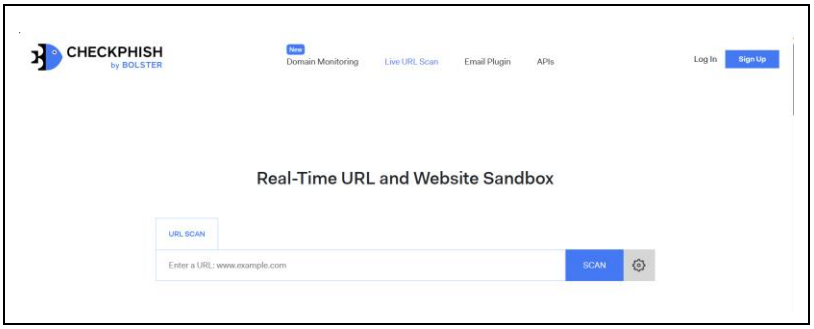


Fig. 2(e) *Interface of CheckPhish* [16]

2.6 Comparison with Existing Works

Table 2 provides a comparison of existing phishing detection tools based on URL such as URL Checker [2], Detector [14], NoaWeb Solutions [13], IsItHacked [15], CheckPhish [16]. Detector and IsItHacked share a similar dataset using the PhishTank dataset while NoaWeb Solutions and PhishShield use Kaggle dataset. However, URL Checker and IsItHacked did not specify the dataset used, Detector relies on datasets from URL Reputation and ICML-09. Notably, URL Checker employs machine learning, Natural Language Processing (NLP), and Support Vector Machine (SVM); Detector uses Logistic Regression (LR); NoaWeb Solutions uses Decision Tree (DT) and Random Forest (RF); CheckPhish uses Deep Learning (DL) Models. IsItHacked does not specify the algorithms employed. None of the tools provide a list of archive URL module whereas PhishShield includes the archive function. Sign-in or log-in functionality is present only in NoaWeb Solutions, IsItHacked, and CheckPhish and PhishShield. PhishShield developed many more functionalities such as changing password, personalizing profile, used RF algorithm employs multiple DT and offers customizable parameters for scalability and enhanced prediction accuracy.

Table 2 *Comparison Table*

Tools	URL Checker [2]	Detector [14]	NoaWeb Solutions [13]	IsItHacked [15]	CheckPhish [16]	Proposed Tool
Dataset	Not stated	URL Reputation, ICML-09	PhishTank, Kaggle, Alexa	PhishTank, Google Safe Browsing	Not stated	Kaggle [17], Github [18]
Technique used	Address Bar-based Features	Address Bar-based Features	Address Bar based Features, Abnormal Based Features, HTML and JavaScript Based Features, Domain Based Features	Blacklist	Not stated	Address Bar-based Features
Machine Learning Algorithm	NLP, SVM	Logistic Regression	Decision Tree and Random Forest	Not stated	Deep Learning Models	Random Forest
Archive URL	No	No	No	No	No	Yes
Change Password	No	No	No	No	No	Yes
Forgot Password	No	No	No	No	Yes	Yes
Profile	No	No	No	No	No	Yes
Login/Sign up	No	No	Yes	Yes	Yes	Yes

3. Methodology

Object-Oriented Analysis and Design (OOAD), shown in Figure 3, serves as a methodological foundation for developing Phishing Detection tool based on URLs, offering practical advantages. OOAD follows a systematic and modular approach that encourages code organization into reusable components called objects, promoting

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 212-231

codebase clarity and maintainability. The method's five phases: Requirement phase involve gathering requirements related to functionality, user interactions, and performance expectations. Key features, such as URL analysis and phishing pattern detection, are identified. The design phase creates a blueprint, defining architecture, classes, and components, considering data flow and machine learning algorithms. Implementation phase develops software components based on design, implementing machine learning algorithms and scoring mechanism for effective phishing URL prediction. The verification phase ensures system testing, while maintenance phase evaluates system performance for informed improvements.

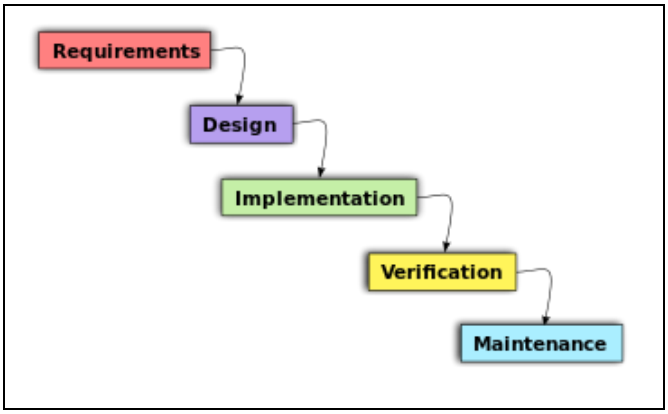


Fig. 3 *Object Oriented Analysis and Design (OOAD)*[19]

3.1 Hardware and Software Requirement

To achieve the stated objectives, both hardware (physical components) and software (system and application software) are essential tools for development and collectively contribute to goal achievement. Table 3 shows the hardware and software requirements.

Table 3 *Hardware and Software Requirements*

Category	Requirement
Hardware	
Laptop	Inspiron 5502
Processor	11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz 1.38 GHz
RAM	16.0 GB
System Type	64-bit operating system, x64-based processor
Software	
Operating System	Windows 11
IDE/Development	Anaconda Navigator, Jupyter Notebook
Web Browser	Chrome
Coding Editor	Visual Studio Code
API Testing	Postman

3.2 Analysis and Design

This section explains the design of the proposed tool which consists of system analysis and system design.

3.3 System Architecture Design

In Figure 4, the system architecture integrates a web application with the Random Forest algorithm and scoring mechanism, facilitating various functionalities through interconnected modules that handle requests, responses, verification, determination, and storage. For example, the URL Scanner module sends requests to the backend, where the Random Forest algorithm and scoring mechanism utilized for predicting the URL and the backend responds with a report of URL, which is then stored in the database. Additionally, modules such as the Login, Registration, URL Scanner, Change Password, Forgot Password, Archive URL, Dashboard and Profile interact with the database to request and retrieve data. The Login module incorporates CAPTCHA verification to ensure the user is not a robot. The registration module involves several steps, including CAPTCHA, MD5 password encryption, validation, data encryption, and user existence checks, before storing the user's information in the database. The Archive URL module requests data prediction from the database for URL reports. Security measures, including CAPTCHA, validation, and data encryption, collectively safeguard the system, ensuring a secure

operational environment. In conclusion, the integration of the Random Forest algorithm and scoring system with robust security measures and efficient module interactions ensures a reliable and secure system architecture for handling user data and URL predictions.

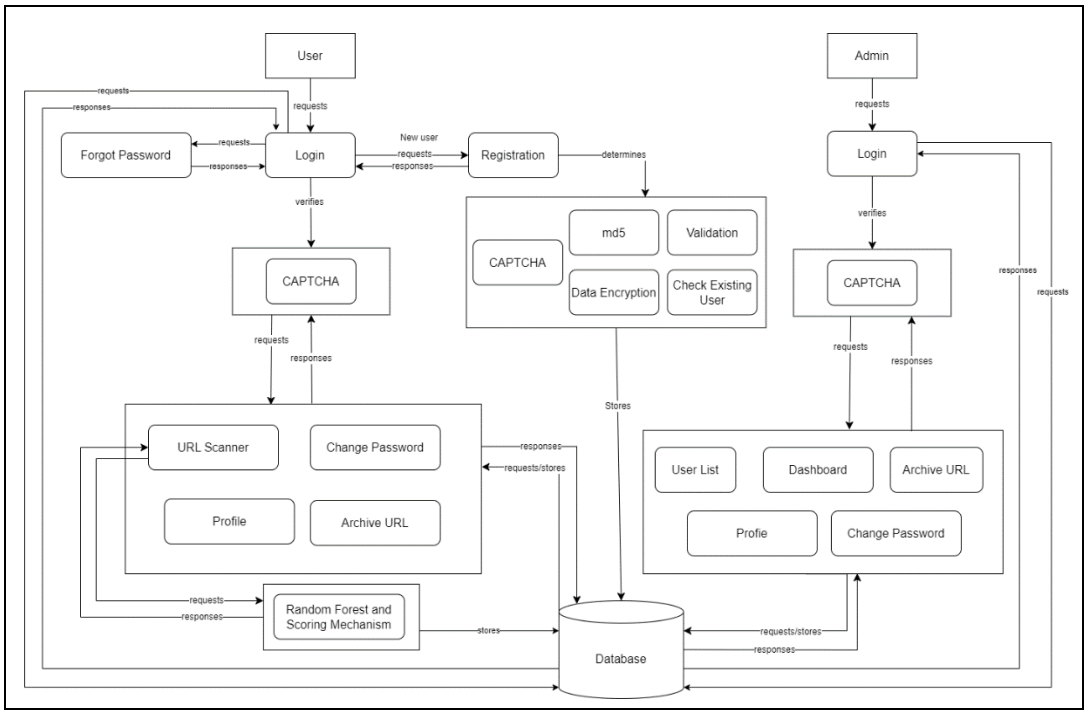


Fig. 4 *System Architecture Design for Proposed Tool*

3.4 Functional Requirement Analysis

The functional requirements for the proposed tool are outlined in Table 4. These modules are designed to be accessible by both administrators and users, promoting a user-friendly interface. While certain modules are available to both admin and user roles, exclusive administrative privileges are granted for specific modules. For example, users can engage with modules such as predicting URLs, viewing archives of URLs and their predicted URLs history. In contrast, administrators have additional privileges, enabling them to manage user accounts, manage archive URLs and monitor user list and archive URLs list. Additionally, the modules that accessible for both are changing password and managing user profiles. This functional breakdown reflects a thorough requirement analysis, ensuring that the proposed tool aligns with user needs and administrative controls.

Table 4 *Functional Requirement Analysis*

Module	Functionalities
Registration - User	- A user registers and save the record of the user. - The tool alerts for invalid inputs and confirmation.
Login - User - Administrator	- The tool contains a session for a user and an administrator login into the system. - The tool alerts for invalid inputs.
URL Scanner - User	- A user inputs the URL and predicts if the URL is phishing or legitimate using Random Forest algorithm and scoring mechanism. - A user can view the report after prediction. - The system alerts for invalid inputs.
URL Archive - User - Administrator	- A user can view archive URLs. - An administrator can manage archive URLs. - The system alerts for confirmation of deletion or changes.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 212-231

Table 4: (cont)

Module	Functionalities
Profile - User - Administrator	- A user can manage their information profile picture and username. - A user can view their predicted URLs history. - The system alerts for invalid inputs and confirmation.
Change Password - User - Administrator	- A user can change password by input their old password, new password and confirm new password. - The system alerts for invalid inputs and confirmation.
Dashboard - Administrator	- An admin can monitor and track the status or progress of user list and archive URLs list on a dashboard using numerical counters.
Forgot Password - User	- If a user forgets their password, they can access this module and input their registered email address, and subsequently receives a one-time password (OTP) via email. Then, the user can reset their password. - The system alerts for invalid inputs.
User List - Administrator	- An administrator can manage registered users such as profile picture, username and email address and action to manage user. - The system alerts for invalid inputs and confirmation.

3.5 Non-Functional Requirement Analysis

Table 5 shows the non-functional requirements of the proposed tool, with a specific focus on evaluating criteria such as the performance of the user and administrator interface, operational efficiency, and security measures. Performance metrics for the user interface encompass responsiveness, speed, and accessibility, aiming to optimize user interactions. Operational requirements address the tool's overall efficiency, reliability, and scalability, ensuring the tool can function under varying conditions. In terms of security, the non-functional requirements security measures such as data and password encryption, reset password, strong password, check existing user and CAPTCHA and resilience against potential threats to safeguard the tool's integrity.

Table 5 *Non-Functional Requirement Analysis*

Requirement	Description
Performance of user and administrator interface and system flow	- The system able to establish and manage user-administrator sessions effectively. - The system able to navigate users and administrators to their intended pages. - The system able to execute accurate actions in response to user and administrator inputs.
Operational	- The system able to function in the presence with or without internet connection.
Security	- The system capable of authorizing users and administrators through username and password, authenticating with CAPTCHA, and maintaining user and admin account information. - The system enforces password policies, requiring a minimum of 8 characters, including at least one uppercase letter, one lowercase letter, and a special symbol, along with password hashing. - The system checks whether a given email and username already exist in the system. - The system implements prepare statement mechanisms. - In cases where users forget their passwords, the system can send a token via email, allowing them to reset their password.

3.6 Use-Case Diagram

A use case diagram in Unified Modeling Language (UML) depicts a system's behavior from an external viewpoint, highlighting the interactions between actors and the system through various use cases and their relationships. For instance, as illustrated in Figure 5(a), administrators are provided with functionalities such as logging in, managing user accounts (including editing and deleting), managing archive URLs (including editing and deleting), managing profiles, changing passwords, and logging out. Similarly, as shown in Figure 5(b), users can interact with the system by registering, logging in, inputting URLs, viewing their predicted URL history, accessing archived URLs, managing their profiles, changing passwords, resetting passwords, and logging out.

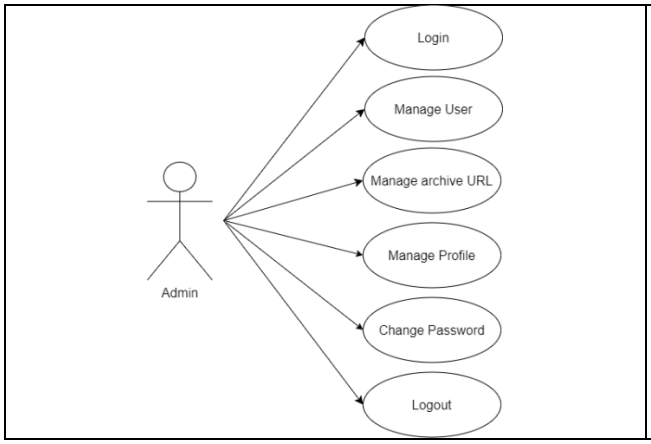


Fig. 5(a) *Use Case Diagram for Admin*

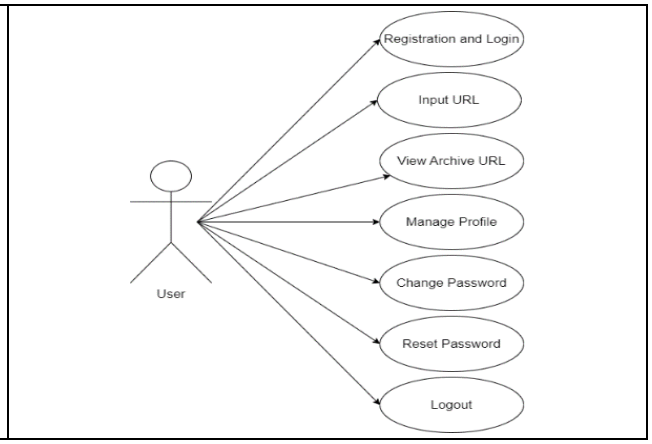


Fig. 5(b) *Use Case Diagram for User*

3.7 Activity Diagram

An activity diagram, a type of Unified Modeling Language (UML) diagram, visually represents the workflow or dynamic aspects of a system or process. Figure 6(a) illustrates the administrator's activity diagram, beginning with login authentication and leading to the dashboard. From the dashboard, administrators can access functionalities such as managing user accounts, profiles, archive URLs, and changing passwords. On the User List page, administrators can edit or delete users; on the Archive List page, they can edit or delete archives; on the Profile page, they can edit; and on the Change Password page, they can change passwords. Then, the session concludes with logout.

Figure 6(b) depicts a user activity diagram, where registered users log in, and new users complete the registration process. For forgotten password, users initiate by giving their email to receive a one-time password (OTP) via their email for password reset. Upon successful authentication, users proceed to the URL Scanner page, where they can access Archive URLs, Profile, and Change Password pages. In the URL Scanner, user inputs URLs to predict whether they are phishing or legitimate using the Random Forest algorithm and scoring mechanism. The tool then generates a report for the URL. Users can view archived URLs, update their profile, and reset their password, with the session ending upon logout.

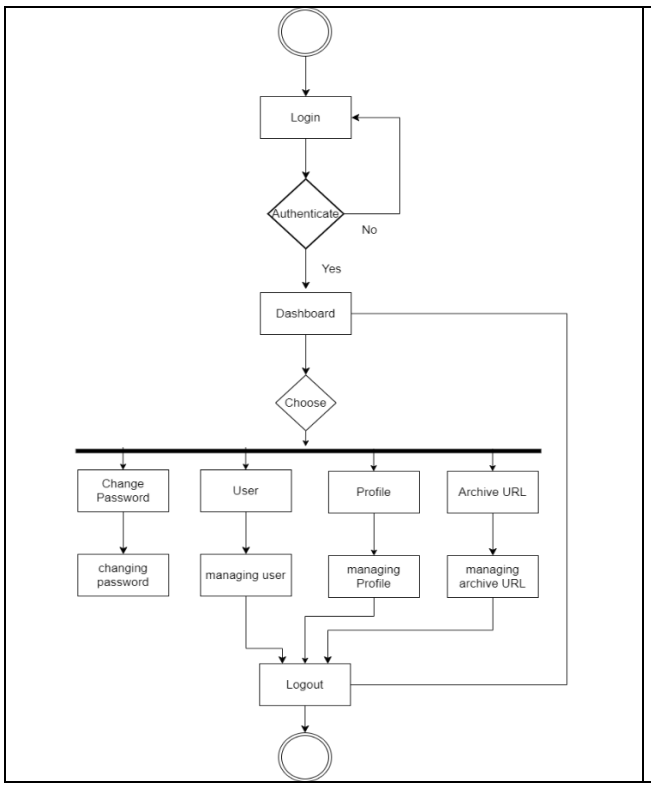


Fig. 6(a) *Activity Diagram for Admin*

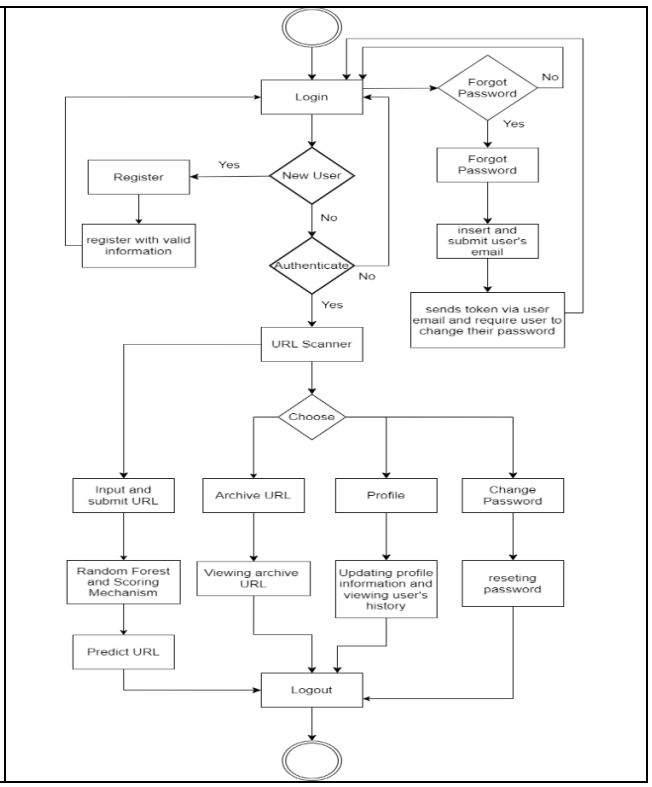


Fig. 6(b) *Activity Diagram for User*

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 212-231

3.8 Entity Relationship Diagram (ERD)

The Entity-Relationship Diagram (ERD) visually outlines relationships among database entities, as shown in Figure 8, which includes three tables. The "USER" table contains a primary key named USER_ID and a foreign key named PREDICTION_ID, with variable-length strings capturing the username, email, password, and image, and an integer value for status, indicating account activation. Additionally, 'code' is used for sending OTPs for verification, while 'statustext' validates whether the user is verified. A user can access multiple data prediction (1 → N).

The "ADMIN" table includes a primary key for ADMIN_ID and foreign keys USER_ID and PREDICTION_ID, with variable-length strings for username, email, and password. An admin can manage multiple users and multiple data prediction (1 → N).

The "DATA PREDICTION" table has a primary key PREDICTION_ID and foreign keys USER_ID and ADMIN_ID, along with seven variable-length strings capturing the URL, username, email, age, IP address. The 'result' is a message according to the scoring mechanism. The 'remarks' is a security advice message; 'modelprediction' is a result according to the Radom Forest model prediction; 'googlesafepassed' capturing Google's Safe Browsing status; 'istemporarydomain' capturing the URL uses a temporary domain status; 'isblacklistedinipset' capturing the IP address blacklisted status. It also includes integers for prediction score; 'domainlength' capturing the length of the domain; 'getdepth' capturing the URL depth (count of '/'); 'digitcount' capturing the number of digits in the URL, and verification status. In conclusion, the ERD provides a comprehensive visualization of the database structure, detailing the relationships and attributes of each table, thereby facilitating a clearer understanding of the data flow and interactions within the system.

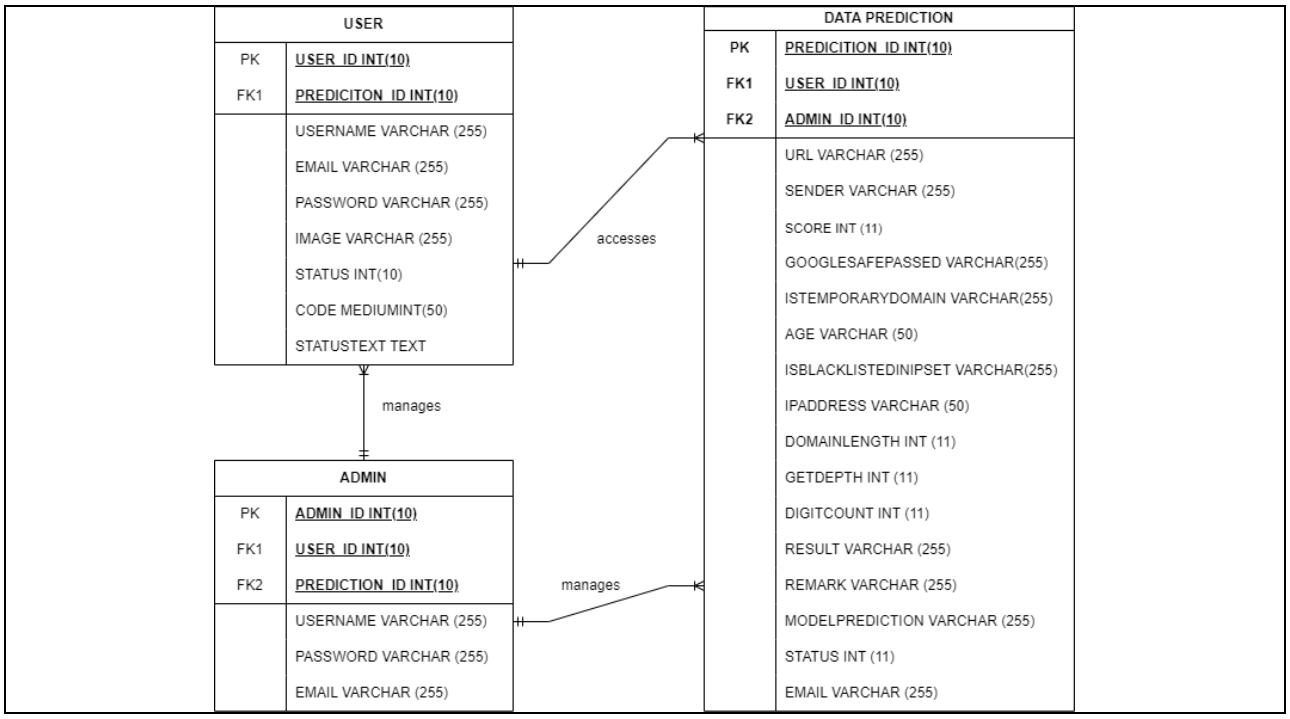


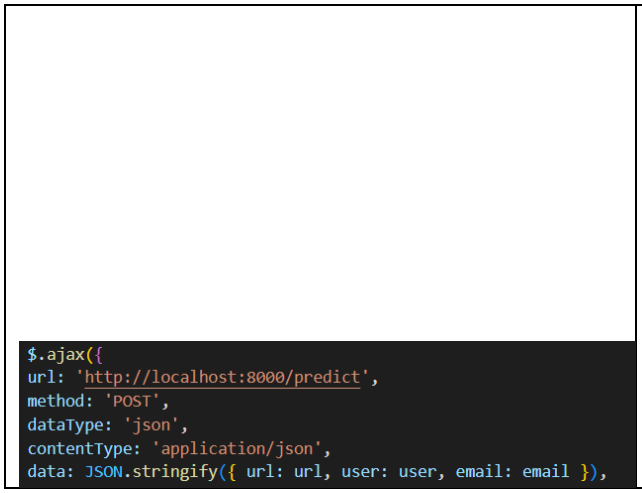
Fig. 8 *Entity Relationship Diagram of The Proposed Tool*

4. Result and Discussion

This section examines the result, discussion, and implementation of the proposed tool. This includes implementation of the security, machine learning, and scoring mechanism properties and result from the security testing, user acceptance, and Application Programming Interface (API) testing.

4.1 Implementation of Security Properties

The proposed tool leverages an Application Programming Interface (API) for seamless communication between the client and server. By utilizing an Asynchronous JavaScript and XML (AJAX) method with the HTTP POST method, this function securely transmits URL, user, and email data to the backend. Choosing the POST method over GET enhances security by preventing sensitive information from being exposed in the URL. In Figure 9(a) shows how the jQuery AJAX method communicates with the API function. In Figure 9(b) shows how the API function responses with the jQuery AJAX.



```
$.ajax({
url: 'http://localhost:8000/predict',
method: 'POST',
dataType: 'json',
contentType: 'application/json',
data: JSON.stringify({ url: url, user: user, email: email }),
```

Fig. 9(a) *Code Snippet of AJAX Function*

```
@app.post("/predict")
def predict(data: UrlData):
    try:
        url = unshorten_url(data.url)

        if not url:
            msg = "The URL cannot be unshorten"
            print(msg)
            return {"message": msg}
        else:
            print("Unshortened URL:", url)

        if not domainStatus(url):
            msg = "The URL is not valid or does not exist"
            print(msg)
            return {"message": msg}
        else:
            print("A Valid URL:", url)

        # Predict using ML model
        prediction = get_prediction(url, data.user, data.email, forest)
        print("Predicted Probability:", prediction)
```

Fig. 9(b) *Code Snippet of API*

4.2 Implementation of Machine Learning Properties

Feature extraction for URLs involves transforming the raw URL strings into a set of numerical or categorical features that can be used as input for machine learning algorithm [20]. In table 3 shows there are 22 address bar-based features extracted in the proposed tool.

Table 3 *Feature Extraction for The Proposed Tool*

Features	Description
Invalid URL	Checks if the URL is valid or not.
Using IP	Determines if the URL uses an IP address.
Long URL	Evaluates whether the URL is excessively long.
Shortening Services	Detects if the URL is generated by a URL shortening service.
Extension	Detects if the URL is using a suspicious extension.
Symbol	Detects if the URL is using a suspicious symbol.
Depth	Evaluates whether the URL has an excessive number of subdirectories.
Query Parameters	Detects if the URL is using suspicious query parameters.
Domain Entropy	Evaluates whether the URL contains high entropy.
Redirection	Detects if the URL is using double slashes after the scheme.
Prefix Suffix	Detects if the URL is using a prefix or suffix in the domain.
SubDomains	Evaluates whether the URL contains an excessive number of dots.
HTTPS	Detects if the URL starts with “https”.
TLD	Detects if the URL is using a suspicious Top-Level Domain (TLD).
Non-standard Port	Detects if the URL is using a non-standard port.
HTTPS Domain URL	Detects if the URL is using “https” in the domain.
Suspicious Domain	Detects if the URL is using a suspicious domain.
Temporary Domain	Detects if the URL is using a temporary domain.
English Letters	Detects if the URL is using non-English letters.
Dash URL	Evaluates whether the URL contains an excessive number of dashes.
Digit URL	Evaluates whether the URL contains an excessive number of digits.

Based on the performance metrics provided in Figure 10, the Decision Tree and Random Forest algorithms are among the top performers, both achieving an accuracy of 0.828. However, they have different f1_scores, recall, and precision values. The Light GBM model stands out with the accuracy of 0.827. The Gradient Boosting Classifier, XGBoost Classifier, and Multi-layer Perceptron all share an accuracy of 0.826. The model with the lowest accuracy is Logistic Regression, at 0.773. Given these results, the Random Forest algorithm is recommended due to its robustness and superior ability to handle overfitting compared to a single Decision Tree.

	ML Model	Accuracy	f1_score	Recall	Precision
0	Decision Tree	0.828	0.823	0.797	0.849
1	Random Forest	0.828	0.822	0.795	0.850
2	Light GBM	0.827	0.822	0.799	0.845
3	Gradient Boosting Classifier	0.826	0.821	0.799	0.843
4	XGBoost Classifier	0.826	0.821	0.797	0.845
5	Multi-layer Perceptron	0.826	0.818	0.781	0.856
6	Logistic Regression	0.773	0.766	0.744	0.789

Fig. 10 *Performance Metrics of Models*

Once training a machine learning model, the saved model is essential for deploying the proposed tools. Figure 11(a) illustrates the code snippet for saving a trained random forest algorithm using the `pickle` library, ensuring the model persistence and accessibility by serializing it to a file. In production environments, real-time predictions are vital, and Figure 11(b) outlines the steps for loading the saved random forest algorithm. Once loaded, the model is ready to make predictions based on input data. Figure 11(c) demonstrates how the loaded model can predict whether a given input array corresponds to a legitimate (0) or phishing (1) instance, which is performing decision-making.

```python
import pickle
# Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(n_estimators=10)

# fit the model
forest.fit(X_train,y_train)

with open('forest.pkl', 'wb') as file:
    pickle.dump(forest, file)
```

Fig. 11(a) *Code Snippet of Saving the Random Forest Algorithm*

```python
# load the the model
print("Loading the model...")
with open('forest.pkl', 'rb') as file:
    forest = pickle.load(file)
```

Fig. 11(b) *Code Snippet of Loading the Random Forest Algorithm*

```python
def getInputArray(url):
    result = []
    result.append(InvalidUrl(url))
    result.append(UsingIp(url))
    result.append(longUrl(url))
    result.append(shorteningService(url))
    result.append(Extension(url))
    result.append(symbol(url))
    result.append(getDepth(url))
    result.append(queryParameters(url))
    result.append(domainEntropy(url))
    result.append(redirecting(url))
    result.append(prefixSuffix(url))
    result.append(SubDomains(url))
    result.append(Hppts(url))
    result.append(tld(url))
    result.append(NonStdPort(url))
    result.append(pathTokens(url))
    result.append(HTTPSDomainURL(url))
    result.append(suspiciousDomain(url))
    result.append(temporaryDomain(url))
    result.append(englishLetters(url))
    result.append(dashUrl(url))
    result.append(digitUrl(url))

    return result

def isURLMalicious(url, forest):
    input = getInputArray(url)
    # make predictions on the new data
    prediction = int(forest.predict([input])[0])  # Convert prediction to int
    return prediction
```

Fig. 11(c) *Code Snippet of Random Forest Prediction*

4.3 Implementation of Scoring Mechanism Properties

While machine learning algorithm are powerful, they can be limited by the available data, necessitating additional measures in data-constrained scenarios. To mitigate false positives and false negatives, the proposed tool introduces a scoring mechanism worked by [18] . Table 4 shows how the scoring mechanism works. Figure 12 shows the code snippet of `dnsblacklist(domain)` function to identify blacklisted domains by checking them against DNS-based blacklists.

Table 4 *Scoring Mechanism*

Statement	score > =70	score<70 and score >=50	score<50
Result	Legitimate	Suspicious	Phishing

```
def dnsblacklist(domain):
    try:
        domain_checker = pydnsbl.DNSBLDomainChecker()
        result = domain_checker.check(domain)

        # Return True if the domain is blacklisted, otherwise False
        return result.blacklisted
    except:
        return False
```

Fig. 12 *Code Snippet for Identifying DNS Blacklist*

Figure 13 shows the code snippet verifies whether the result of calling the `Utils.dnsblacklist(domain)` function evaluates to True. If the domain is identified as blacklisted, the code updates the `output["isdnsblacklist"] ` to True, simultaneously reducing the SCORE by 30. Once completing the processes, the proposed tools can be fully functional. PhishShield is a web application designed to detect phishing URL using machine learning algorithm with scoring mechanism. The proposed tool uses HTML, CSS, PHP, and JavaScript. The backend logic is powered by Python, which handles requests and data processing. MySQL is used for storing user profiles, admin profiles, and prediction results. This combination of technologies makes PhishShield a simple yet effective tool for identifying phishing threats. In Figure 14(a) and Figure 14(b) shows the homepage before and after prediction.

```
try:
    if Utils.dnsblacklist(domain) == True:
        output["isdnsblacklist"] = True
        output["SCORE"] -= 30
        print("Deduct 30 A DNS Blacklist!")

except:
    print("Error Occurred while finding to domain DNS Blacklist!")
```

Fig. 13 *Code Snippet for Deducting Scoring for DNS Blacklist*

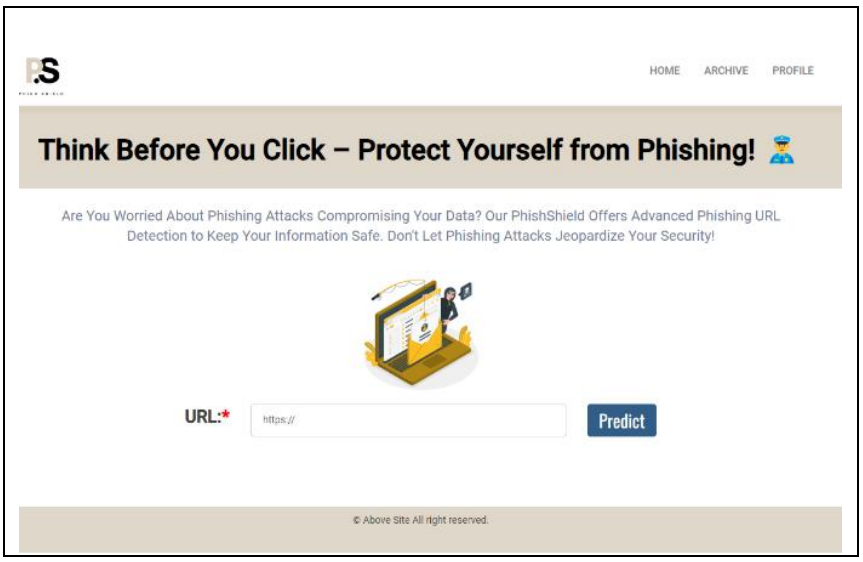


Fig. 14(a) *Homepage Before Prediction*

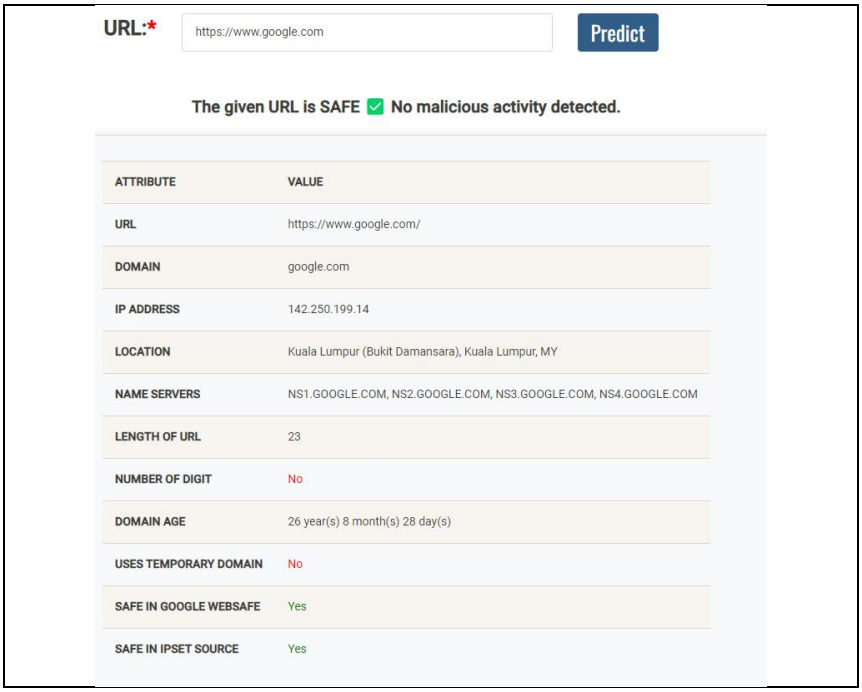


Fig. 14(b) *Homepage After Prediction*

4.4 Security Checklist Result

Table 5 shows the result of a security checklist that can be used as a starting point for evaluating the security aspects of the proposed tool. Notably, all the items on the checklist have been successfully addressed in the proposed tool.

Table 5 *Security Checklist Result*

No	Checklist	Actual Result
1	Ensure strong password policies	Pass
2	Define and enforce proper access control mechanisms	Pass
3	Limit privileges based on roles and responsibilities	Pass
4	Employ encryption algorithms for sensitive data at rest	Pass
5	Validate and sanitize all user inputs to prevent SQL injection and other injection attacks	Pass
6	Define session termination	Pass

4.5 User Acceptance Testing Result

A User Acceptance Testing (UAT) form is an essential document used during the user acceptance testing phase of a software development or implementation project. In Appendix A provides a structured approach to collect data from 14 respondents through quantitative methods, using yes or no answers, and is created with Google Forms. Table 6 shows the result of User Acceptance Testing (UAT) for admin and user side. Notably, all the items on the UAT have been successfully addressed in the proposed tool.

Table 6 *User Acceptance Testing Result*

Description	Expected Result	Actual Result
Admin login with valid credentials	Admin access dashboard without errors	Pass
Admin manages archive URLs on Archive page	Admin manages archive URLs without errors	Pass
Admin manages user accounts on User page	Admin manage user accounts without errors	Pass
Admin updates profile on Profile page	Profile updates successfully	Pass
Admin change password on Change Password Page	Admin changes password successfully	Pass
Admin terminates session	Admin terminates session without errors	Pass
User login with valid credentials	Admin access URL Scanner page without errors	Pass
User register with valid information	User registers and logs in successfully	Pass

Table 6: (cont)

Description	Expected Result	Actual Result
User password reset via Forgot Password page	User resets password successfully	Pass
User submit URL in URL Scanner page	User submits URL successfully	Pass
User views the report on URL Scanner page	The result shown successfully	Pass
User updates profile information on Profile page	Profile updates successfully	Pass
User views archive URLs on Archive page	The archive URLs shown successfully	Pass
User terminates session	Admin terminates session without errors	Pass

4.6 Application Programming Interface (API) Testing

The proposed tools used Postman software for API testing, allowing developers to send HTTP requests and view responses. A POST request sends data to the server to create or update a resource. In Figure 15 shows setting the header to `127.0.0.1:8000/predict` in Postman and pressing "Send" will send the request to the specified endpoint and display the server's response, aiding in thorough testing and debugging. A request body is mandatory for every POST request. According to the documentation, the body should be in JSON format. Figure 16 shows the required data: `url`, `email`, and `user`. The response should contain the data provided with additional information.

Fig. 15 *Header in Postman*

Fig. 16 *Request Body*

Table 7 outlines the initial values and actions for various attributes used in the proposed tool. Each attribute has a default value and specified actions, such as deducting points from the initial score (SCORE of 100), if the attribute's value changes.

Table 7 *Attribute Actions and Their Impact in the Scoring Mechanism*

Attribute	Initial Value	Action on Change
SCORE	100	Current value
URL	URL	Contains the processed URL associated with the request
user	User	Contains the username associated with the request
email	Email	Contains the email associated with the request
isGoogleSafePassed	True	If changes, 40 points are deducted from the SCORE
isDomainRegistered	True	If changes, 10 points are deducted from the SCORE
isdomainActive	True	If changes, 10 points are deducted from the SCORE
issslCertificated	True	If changes, 20 points are deducted from the SCORE
isunshorten	False	If changes, 5 points are deducted from the SCORE
isdnsblacklisted	False	If changes, 40 points are deducted from the SCORE
isredirected	False	If changes, 5 points are deducted from the SCORE
isBlackListedinIpSets	False	If changes, 30 points are deducted from the SCORE
isBlacklisted	False	If changes, 30 points are deducted from the SCORE
age	None	If changes, display the age of the domain. If less than 3 months, 40 points are deducted from the SCORE
isIpAddress	None	If changes, display the IP address. If the IP address is not found, 40 points are deducted from the SCORE

Table 7: (cont)

Attribute	Initial Value	Action on Change
location	None	If changes, display the location. If the location is not found, 10 points are deducted from the SCORE
getDomainLength	None	If changes, display the length of the domain
getDepth	None	If changes, display the number of subdirectories in URL
digitsCount	None	If changes, display the number of digits in URL
finalDomain	None	If changes, display the root domain
result	None	If changes, display a message according to the current SCORE value
remark	None	If changes, display a security advice message according to the current SCORE value
modelprediction	None	If changes, display a message. If predicted as malicious, 40 points are deducted from the SCORE

For testing purposes, we collected five phishing URLs from OpenPhish website. Figure 17(a) to Figure 17(e) show the testing result of phishing URLs. Table 8 summarizes the testing phishing URLs displaying the URL, score, result, and model prediction for each URL. Out of five phishing URLs, 40% were detected as phishing, 40% as suspicious, and 20% as legitimate. The Random Forest algorithm identified 80% of these URLs as malicious and 20% as non-malicious.


```
"predicted_probability": {
    "SCORE": 50,
    "URL": "http://4knetflix.vip/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": null,
    "isIpAddress": "195.85.19.183",
    "location": false,
    "getDomainLength": 21,
    "getDepth": 1,
    "digitsCount": 1,
    "finalDomain": "4knetflix.vip",
    "result": "The given URL might be risky",
    "remark": "It's best to use private browsing and a VPN for extra security",
    "modelprediction": "Model predicted the URL as malicious."
}
```


Fig. 17(a) *Testing Result of Phishing URL (1)*


```
"predicted_probability": {
    "SCORE": 60,
    "URL": "https://cttcommunication.com/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": "0 year(s) 6 month(s) 12 day(s)",
    "isIpAddress": "84.32.57.79",
    "location": "Frankfurt am Main, Hesse, DE",
    "getDomainLength": 29,
    "getDepth": 1,
    "digitsCount": 0,
    "finalDomain": "cttcommunication.com",
    "result": "The given URL might be risky",
    "remark": "It's best to use private browsing and a VPN for extra security",
    "modelprediction": "Model predicted the URL as malicious."
}
```


Fig. 17(b) *Testing Result of Phishing URL (2)*


```
"predicted_probability": {
    "SCORE": 10,
    "URL": "https://leayer.info",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": false,
    "issslCertificate": false,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": "0 year(s) 4 month(s) 4 day(s)",
    "isIpAddress": false,
    "location": false,
    "getDomainLength": 19,
    "getDepth": 1,
    "digitsCount": 0,
    "finalDomain": "leayer.info",
    "result": "The given URL is very dangerous",
    "remark": "Avoid visiting this site to stay safe",
    "modelprediction": "Model predicted the URL as malicious."
}
```


Fig. 17(c) *Testing Result of Phishing URL (3)*


```
"predicted_probability": {
    "SCORE": 70,
    "URL": "https://www.sojope.com/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": "0 year(s) 0 month(s) 9 day(s)",
    "isIpAddress": "47.251.10.79",
    "location": "Minkler, California, US",
    "getDomainLength": 23,
    "getDepth": 1,
    "digitsCount": 0,
    "finalDomain": "sojope.com",
    "result": "The given URL appears to be safe",
    "remark": "You can visit this site without concerns",
    "modelprediction": "Model predicted the URL as non-malicious."
}
```


Fig. 17(d) *Testing Result of Phishing URL (4)*

```
"predicted_probability": {
    "SCORE": 30,
    "URL": "http://ffeerr2.pages.dev/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": true,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": true,
    "isBlacklisted": false,
    "age": null,
    "isIpAddress": "172.66.44.84",
    "location": "Toronto, Ontario, CA",
    "getDomainLength": 25,
    "getDepth": 1,
    "digitsCount": 1,
    "finalDomain": "ffeerr2.pages.dev",
    "result": "The given URL is very dangerous",
    "remark": "Avoid visiting this site to stay safe",
    "modelprediction": "Model predicted the URL as malicious."
}
```


Fig. 17(e) *Testing Result of Phishing URL (5)*

Table 8 *Result of Testing on Phishing URLs*

Figure	URL	Score	Result	Random Forest Model Prediction
17(a)	http://4knetflix.vip/	50	The given URL might be risky	malicious
17(b)	https://cttcommunication.com/	60	The given URL might be risky	malicious
17(c)	https://leayer.info	10	The given URL is very dangerous	malicious
17(d)	https://www.sojope.com/	70	The given URL appears to be safe	non-malicious
17(e)	http://ffeerr2.pages.dev/	30	The given URL is very dangerous	malicious

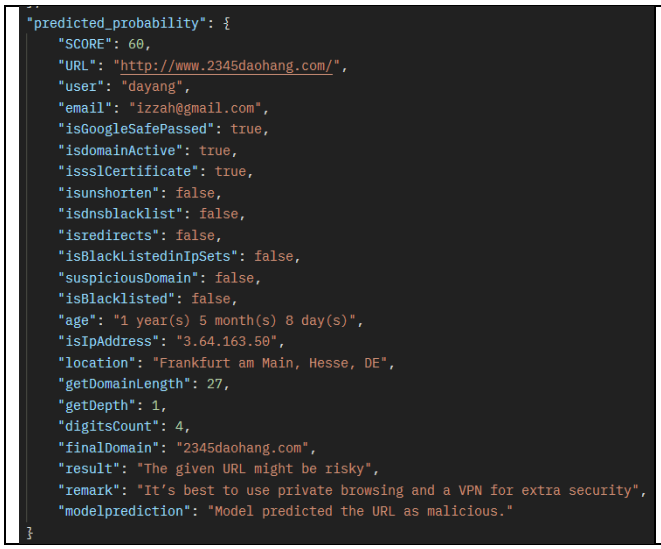
Next, we collected five legitimate URLs from Kaggle dataset. Figure 18(a) to Figure 18(e) show the testing result of legitimate URLs. In Table 9 summarizes the testing legitimate URLs displaying the URL, score, result, and model prediction for each URL. Out of 5 legitimate URLs, 80% were detected as legitimate, while 20% were identified as suspicious. The Random Forest algorithm detected 20% of these URLs as malicious and 80% as non-malicious.


```
"predicted_probability": {
    "SCORE": 100,
    "URL": "https://www.mutuo.it/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": "24 year(s) 6 month(s) 3 day(s)",
    "isIpAddress": "172.67.128.114",
    "location": "Toronto, Ontario, CA",
    "getDomainLength": 21,
    "getDepth": 1,
    "digitsCount": 0,
    "finalDomain": "mutuo.it",
    "result": "The given URL appears to be safe",
    "remark": "You can visit this site without concerns",
    "modelprediction": "Model predicted the URL as non-malicious."
}
```


Fig. 18(a) *Testing Result of Legitimate URL (1)*


```
"predicted_probability": {
    "SCORE": 100,
    "URL": "https://www.missfiga.com/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": "9 year(s) 5 month(s) 21 day(s)",
    "isIpAddress": "217.160.0.32",
    "location": "Essen, North Rhine-Westphalia, DE",
    "getDomainLength": 25,
    "getDepth": 1,
    "digitsCount": 0,
    "finalDomain": "missfiga.com",
    "result": "The given URL appears to be safe",
    "remark": "You can visit this site without concerns",
    "modelprediction": "Model predicted the URL as non-malicious."
}
```


Fig. 18(b) *Testing Result of Legitimate URL (2)*



```
"predicted_probability": {
    "SCORE": 60,
    "URL": "http://www.2345daohang.com/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": "1 year(s) 5 month(s) 8 day(s)",
    "isIpAddress": "3.64.163.50",
    "location": "Frankfurt am Main, Hesse, DE",
    "getDomainLength": 27,
    "getDepth": 1,
    "digitsCount": 4,
    "finalDomain": "2345daohang.com",
    "result": "The given URL might be risky",
    "remark": "It's best to use private browsing and a VPN for extra security",
    "modelprediction": "Model predicted the URL as malicious."
}
```


Fig. 18(c) *Testing Result of Legitimate URL (3)*

```
"predicted_probability": {
    "SCORE": 100,
    "URL": "http://www.enkiquotes.com/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": "7 year(s) 1 month(s) 0 day(s)",
    "isIpAddress": "172.67.215.222",
    "location": "Toronto, Ontario, CA",
    "getDomainLength": 26,
    "getDepth": 1,
    "digitsCount": 0,
    "finalDomain": "enkiquotes.com",
    "result": "The given URL appears to be safe",
    "remark": "You can visit this site without concerns",
    "modelprediction": "Model predicted the URL as non-malicious."
}
```


Fig. 18(d) *Testing Result of Legitimate URL (4)*

```
"predicted_probability": {
    "SCORE": 100,
    "URL": "https://author.uthm.edu.my/",
    "user": "dayang",
    "email": "izzah@gmail.com",
    "isGoogleSafePassed": true,
    "isdomainActive": true,
    "issslCertificate": true,
    "isunshorten": false,
    "isdnsblacklist": false,
    "isredirects": false,
    "isBlackListedinIpSets": false,
    "suspiciousDomain": false,
    "isBlacklisted": false,
    "age": null,
    "isIpAddress": "103.31.34.138",
    "location": "Parit Raja, Johor, MY",
    "getDomainLength": 27,
    "getDepth": 1,
    "digitsCount": 0,
    "finalDomain": "author.uthm.edu.my",
    "result": "The given URL appears to be safe",
    "remark": "You can visit this site without concerns",
    "modelprediction": "Model predicted the URL as non-malicious."
}
```


Fig. 18(e) *Testing Result of Legitimate URL (5)*

Table 9 *Result of Testing on Legitimate URLs*

Figure	URL	Score	Result	Random Forest Model Prediction
18(a)	https://www.mutuo.it/	100	The given URL appears to be safe	non-malicious
18(b)	https://www.missfiga.com/	100	The given URL appears to be safe	non-malicious
18(c)	http://www.2345daohang.com/	60	The given URL might be risky	malicious
18(d)	http://www.enkiquotes.com/	100	The given URL appears to be safe	non-malicious
18(e)	https://author.uthm.edu.my/	100	The given URL appears to be safe	non-malicious

Conclusion

In conclusion, the proposed tool, PhishShield, employs machine learning, specifically the Random Forest algorithm, to enhance the detection of phishing URLs. Utilizing the methodology of Object-Oriented Analysis and Design (OOAD), the development process ensures a modular, scalable, and maintainable system. The user-friendly interface provides distinct functionalities for both administrators and users, including URL prediction, profile management, user list management, password resetting, password changing. PhishShield not only addresses the challenges of current phishing detection methods but also introduces features like real-time detection, URL archiving, and intuitive interfaces, offering a comprehensive solution for combating evolving cybersecurity threats. However, the tool has limitations, particularly in the accuracy of machine learning in detecting phishing or legitimate URLs. To address this, future work will focus on developing a browser extension with robust machine learning accuracy, further enhancing PhishShield's capabilities. The integration of security measures, user-friendly interfaces, and advanced machine learning algorithms positions PhishShield as a valuable tool in fostering a secure and evolving environment.

Acknowledgement

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

References

[1] Mahajan, R., & Siddavatam, I. (2018). Phishing website detection using machine learning algorithms. *International Journal of Computer Applications*, 181, 45–47. https://doi.org/10.5120/ijca2018918026

[2] Anitha, R., Swathi, S., Vasuhi, R., & Thenmozhi, P. (2019). Detecting URL phishing attacks using machine learning & NLP techniques. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, 5(2), 53–56. Retrieved November 16, 2023, from https://ijsrcseit.com/paper/CSEIT19522.pdf

[3] Vayansky, I., & Kumar, S. (2018). Phishing – challenges and solutions. *Computer Fraud & Security*, 2018, 15–20. https://doi.org/10.1016/S1361-3723(18)30007-1

[4] Thomas, J. (2018). Individual cyber security: Empowering employees to resist spear phishing to prevent identity theft and ransomware attacks. *International Journal of Business and Management*, 13, 1. https://doi.org/10.5539/ijbm.v13n6p1

[5] Karim, A., Azam, S., Shanmugam, B., Kannoorpatti, K., & Alazab, M. (2019). A comprehensive survey for intelligent spam email detection. *IEEE Access*, PP, 1. https://doi.org/10.1109/ACCESS.2019.2954791

[6] Ali, W., & Abdullah, A. A. (2019). Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting. *IET Information Security*. https://doi.org/10.1049/iet-ifs.2019.0006

[7] Afandi, N. A., & Isredza Rahmi A Hamid. (2021). Covid-19 Phishing Detection Based on Hyperlink Using K-Nearest Neighbor (KNN) Algorithm. Applied Information Technology And Computer Science, 2(2), 287-301. https://publisher.uthm.edu.my/periodicals/index.php/aitcs/article/view/2317

[8] Components of a URL. (2021). Retrieved November 17, 2023, from https://www.geeksforgeeks.org/components-of-a-url/

[9] Nataraj, K. R., Yashaswini, D. K., Hema, R., Pawar, N. S., & Yashaswi, S. (2023). Phishing attack detection using machine learning. In *Proceedings of 2023 Conference*. https://doi.org/10.1007/978-981-99-2058-7_33

[10] Yang, F.-J. (2019). An extended idea about decision trees. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 349–354). https://doi.org/10.1109/CSCI49370.2019.00068

[11] Cutler, A., Cutler, D., & Stevens, J. (2011). Random forests. In *Machine Learning – ML* (Vol. 45, pp. 157–176). https://doi.org/10.1007/978-1-4419-9326-7_5

[12] Zhang, P., Jia, Y., & Shang, Y. (2022). Research and application of XGBoost in imbalanced data. *International Journal of Distributed Sensor Networks, 18*(6). https://doi.org/10.1177/15501329221106935

[13] Atharva, D., Omkar, P., Nachiket, C., & Swapna, B. (2021). Detection of phishing websites using machine learning. *International Journal of Engineering Research & Technology (IJERT)*, 10(5), 430–434. Retrieved November 16, 2023, from https://www.ijert.org/research/detection-of-phishing-websites-using-machine-learning-IJERTV10IS050235.pdf

[14] Naveen, M., & Gubbala, D. (2020). An optimized fake website detection using regression and machine learning. *Journal of Information and Computational Science*, 10, 219–229.

[15] IsItHacked. (2018). Retrieved November 16, 2023, from https://isithacked.com

[16] CheckPhish AI. (2023). Retrieved November 16, 2023, from https://checkphish.ai/

[17] Phishing and legitimate URLs. (2023). Retrieved July 2, 2024, from https://www.kaggle.com/datasets/harisudhan411/phishing-and-legitimate-urls
[18] Mhatre, D. (2024). Phishr-API. GitHub repository. Retrieved July 2, 2024, from https://github.com/deepeshdm/Phishr-API/blob/main/data/urldata.csv
[19] Object oriented analysis and design. (2018). Retrieved June 13, 2024, from https://www.codeproject.com/Articles/1137299/Object-Oriented-Analysis-and-Design
[20] Kee, X. N., & Hamid, I. R. A. (2023). Feature extraction tool for phishing dataset. *AITCS*, 4(2), 248–268. https://penerbit.uthm.edu.my/periodicals/index.php/aitcs/article/view/12219

Appendix A: User Acceptance Testing (UAT) Result for Admin and User Side Modules

Figures A.1 show the result of user acceptance testing for admin side modules.

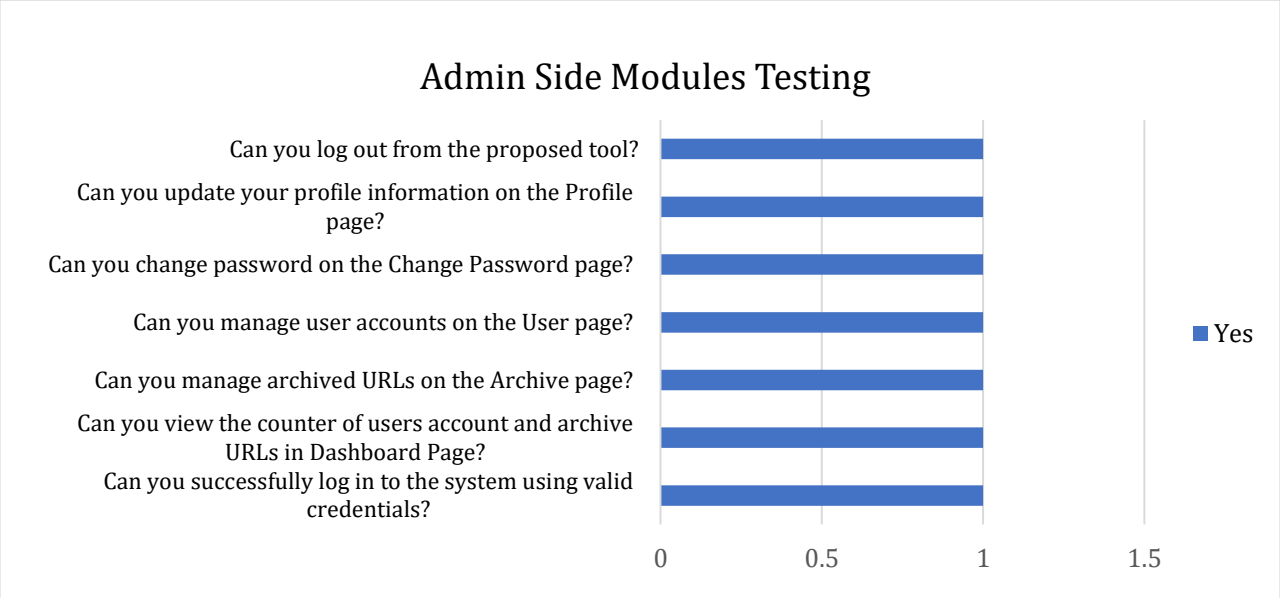


Fig. A.1 *The UAT Result of Admin Side*

Figures A.2 show the result of user acceptance testing for user side modules.

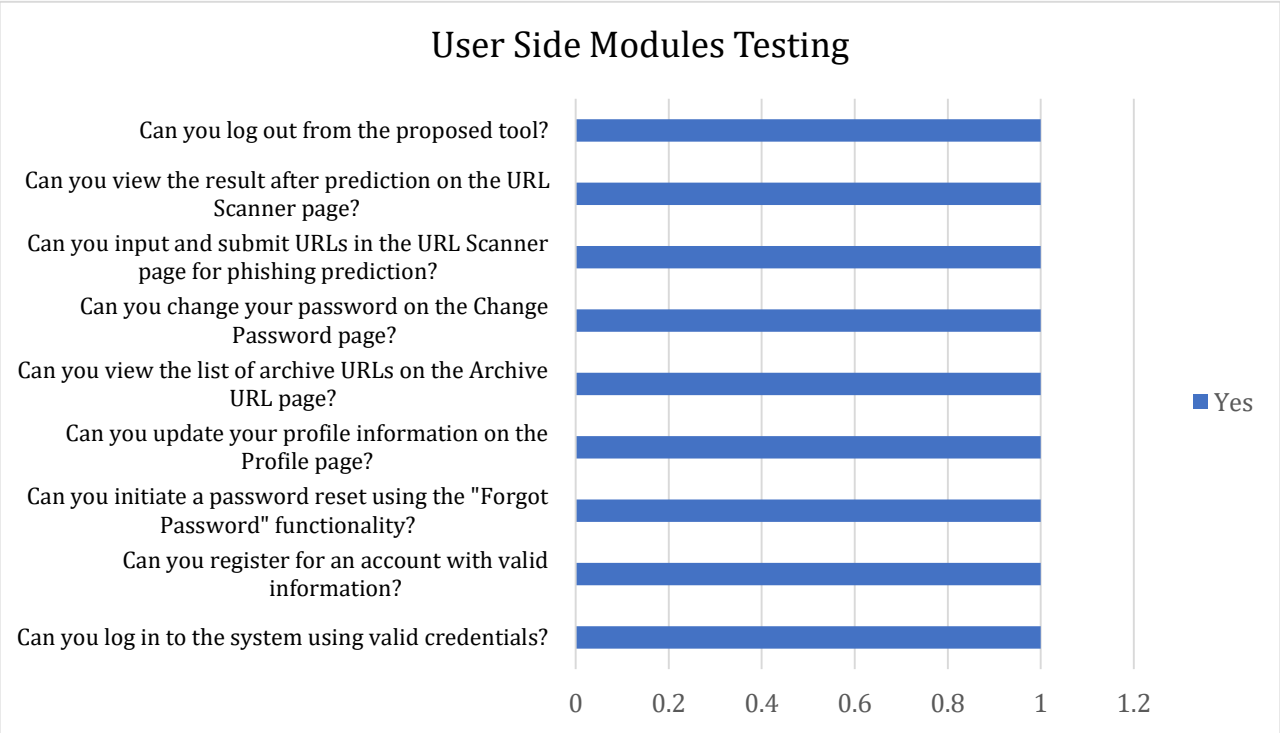


Fig. A.2 *The UAT Result of User Side*