iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

SAFESCAN: PROACTIVE MOBILE DEFENSE AGAINST QR CODE PHISHING (QUISHING)

Mrs. K.S.S. Soujanya Kumari (Assistant Professor)
Department of Computer Science and Systems Engineering,
Andhra University College of Engineering (A), Visakhapatnam, A.P, India.

Mullu Kavitha (MCA Student)
Regd. No.: 324207360098
Department of Information Technology and Computer Applications,
Andhra University College of Engineering (A), Andhra University, Visakhapatnam – 530003,
A.P, India.

Corresponding Author: Mullu Kavita**, Email**: kavithamullu304@gmail.com

ABSTRACT
QR code usage has increased rapidly in digital payments, online shopping, ticket booking, and business applications. However, attackers exploit this popularity by embedding malicious URLs inside QR codes, leading users to phishing websites that can steal usernames, passwords, banking credentials, and personal data. Since the destination URL is hidden until the QR code is scanned, users cannot easily verify whether it is safe before accessing it. SafeScan – Proactive Mobile Defense Against QR Code Phishing (Quishing) provides a security layer that analyzes a QR destination before direct navigation. The proposed system uses React Native with Expo for the mobile application and FastAPI with Python for the backend. After scanning a QR code, the system extracts the embedded URL, expands shortened links, validates the destination, performs heuristic-based security analysis, calculates a risk score, and classifies the URL as Low, Medium, or High risk. To improve user awareness, the backend can generate a webpage preview using Playwright and headless Chromium in a controlled environment. The mobile application presents the decoded or expanded URL, risk score, verdict, preview, and recommended action without automatically opening the destination. Functional testing in the project demonstrates the complete workflow from QR acquisition to safety reporting and controlled handling of invalid inputs and connectivity failures. SafeScan therefore provides an interpretable, mobile-oriented baseline for reducing exposure to QR-based phishing and supporting safer navigation decisions.

Keywords:
QR Code Security, Quishing, Phishing Detection, React Native, FastAPI, Playwright, URL Analysis, Mobile Security, Risk Score.

INTRODUCTION
QR codes have become a convenient mechanism for initiating digital actions in payment workflows, quick logins, event check-ins, product verification, and promotional campaigns. The convenience also creates a security concern because the actual destination is opaque until the QR code is processed. Attackers can replace or distribute QR codes that redirect users to fake login pages, fraudulent services, unwanted downloads, or other malicious destinations.

SafeScan is motivated by the need for an accessible, mobile-centric security layer that supports informed decision-making before a user opens a potentially unsafe link. Instead of treating QR scanning as a direct navigation action, the system treats the extracted content as untrusted input and sends it to a backend analysis pipeline. The pipeline validates and normalizes the URL, follows redirects, evaluates observable phishing indicators, and returns a structured safety report

The proposed approach is designed around a safety-first interaction model. The user explicitly initiates a scan, the application extracts the QR content, and the backend performs security checks before navigation. This reduces dependence on users manually inspecting hidden URLs and provides a clear decision point. The project also supports scanning from an uploaded image, which is useful for QR codes received through screenshots or messaging applications.

 © Author(s) | CC BY 4.0 International License.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

PROBLEM STATEMENT

QR codes can embed URLs that lead to phishing or malicious websites, while common scanning behavior may expose users because of limited transparency and insufficient pre-navigation checks. Existing QR scanner applications may automatically open the embedded link and generally do not provide a risk score, structured security explanation, redirect analysis, or visual preview. The core problem is therefore to design a practical mobile platform that safely extracts and evaluates QR-embedded URLs, expands redirections, estimates risk using observable characteristics, and presents the result before allowing the user to visit the destination.

EXISTING SYSTEM

Traditional QR scanner applications primarily emphasize scanning speed and convenience. They decode the QR content and may redirect the user to the embedded website without a dedicated security verification stage. This can leave users responsible for identifying suspicious domains, misleading login pages, or fraudulent content after the connection has already been initiated.

RELATED WORK

Prior work has explored QR-code security, malicious URL detection, phishing classification, and mobile security. Machine-learning approaches commonly use lexical URL features, domain characteristics, special characters, and other signals to classify phishing websites. Reputation-based services can identify known malicious destinations, while conventional scanners provide the basic decoding function. The SafeScan project combines these ideas at the application level by placing URL analysis, redirect verification, risk scoring, and visual preview generation between QR acquisition and user navigation.

The project documentation identifies limitations in existing approaches: machine-learning methods require suitable training data and retraining; reputation services may not cover newly emerging sites; and ordinary scanners can open URLs before adequate verification. SafeScan therefore adopts a layered approach intended to improve transparency and reduce the likelihood of accidental navigation.

OBJECTIVES

The primary objective of SafeScan is to provide proactive protection against QR-code phishing by analyzing an embedded URL before direct navigation. The major objectives are:

- To provide a mobile QR-scanning interface using the device camera.
- To support QR decoding from uploaded images.
- To validate and normalize extracted HTTP/HTTPS URLs.
- To expand shortened URLs and redirect chains to reveal the final destination.
- To analyze suspicious URL characteristics using heuristic security indicators.
- To calculate a bounded risk score from 0 to 100 and produce a Low, Medium, or High verdict.
- To generate an optional visual webpage preview using a controlled headless browser.
- To display a structured safety report with the URL, score, verdict, preview, and recommended action.
- To avoid automatic navigation so that the final decision remains under user control.
- To handle invalid inputs, decoding failures, network failures, and backend unavailability gracefully.

METHODOLOGY

SafeScan follows a client-server architecture consisting of a React Native/Expo mobile client and a Python FastAPI backend. The mobile client performs QR acquisition through the camera or gallery. The decoded content is transmitted to the backend through REST requests, where it is treated as untrusted input. The backend validates the URL, normalizes it when necessary, and follows redirects to determine the final destination.

QR ACQUISITION AND URL PROCESSING

The QR Code Scanner Module activates the device camera and decodes the QR payload without immediately opening the embedded URL. For uploaded images, the mobile client supplies Base64 image data and the backend uses OpenCV's QRCodeDetector to extract the QR content. Only URL-like inputs proceed to the analysis pipeline.

RISK DETECTION AND SCORING

The URL Risk Detection Module evaluates security-related characteristics including HTTPS usage, domain and URL structure, suspicious keywords, unusual characters, URL length, excessive subdomains, IP-based hosts, and checks against malicious or blacklisted URLs where configured. The resulting risk score is bounded from 0 to 100 and mapped to Low, Medium, or High risk. The project's documented conceptual ranges are Low 0–30, Medium 31–70, and High 71–100.

VISUAL PREVIEW GENERATION

 © Author(s) | CC BY 4.0 International License.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

To provide contextual evidence without requiring the user to open the destination directly, SafeScan uses Playwright with headless Chromium to render the expanded URL in a controlled environment and capture a screenshot. The screenshot is encoded as Base64 and returned to the mobile client. The Safety Report then displays the visual preview together with the URL and risk verdict.

SAFETY REPORT AND USER DECISION

The report screen presents the expanded URL, numerical risk score, qualitative verdict, and visual preview when available. Depending on the risk level, the interface can recommend proceeding safely, proceeding with caution, or exiting. Importantly, the system does not automatically open the scanned link, leaving the final navigation decision to the user.

NON-FUNCTIONAL REQUIREMENTS

Usability requires a clear purpose, readable results, and minimal interaction steps. Performance requires reasonable response time with appropriate timeouts because redirect expansion and browser rendering can be latency-intensive. Reliability requires controlled handling of invalid URLs, empty inputs, QR decoding failures, and network errors. Security and privacy requirements include treating scanned content as untrusted input and avoiding persistent storage of scanned URLs or screenshots by default. The design also supports future containerization, horizontal scaling, caching, rate limiting, and background execution of expensive screenshot tasks.

TECHNOLOGY STACK

The backend uses FastAPI with Uvicorn, httpx for redirect-aware HTTP processing, Playwright for browser automation, and OpenCV for QR decoding from image bytes. The mobile application uses React Native with Expo, including Expo Camera and Barcode Scanner for live scanning and Expo Image Picker for gallery inputs. The project report specifies React 19.2.0, React Native 0.83.2, Expo ~55.0.4, expo-camera ~55.0.9, expo-barcode-scanner ^13.0.1, and expo-image-picker ~55.0.11.

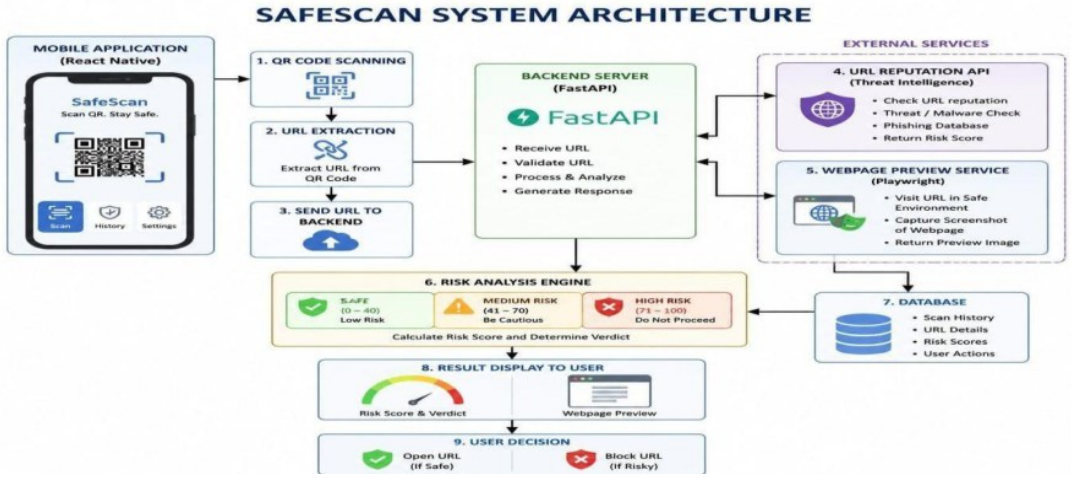

Figure 1: SafeScan System Architecture

SYSTEM DESIGN

The system design separates scanning, analysis, visualization, and reporting responsibilities. This modular organization simplifies maintenance and allows individual components to be enhanced without changing the complete application.

QR CODE SCANNER MODULE

The QR Code Scanner Module is the entry point of SafeScan. It requests camera permission, provides a scanning frame, detects and decodes QR codes, and forwards the extracted string to the backend. The module also supports uploaded images through a Base64 workflow. The URL is intentionally not opened immediately; instead, it is passed to the Risk Detection Engine.

URL RISK DETECTION MODULE

The URL Risk Detection Module determines whether a scanned destination appears safe, suspicious, or malicious. It checks HTTPS security, domain characteristics, URL structure, suspicious keywords, blacklist or reputation information where configured, and other phishing indicators. The output is a 0–100 risk score and a categorical risk level.

VISUAL PREVIEW MODULE

 © Author(s) | CC BY 4.0 International License.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

The Visual Preview Module renders the destination through Playwright and headless Chromium and captures a screenshot for inspection. The purpose is to provide an additional contextual signal, especially when a phishing page imitates the design of a legitimate service.

SECURITY REPORT MODULE

The Security Report Module consolidates the expanded URL, score, verdict, preview, and recommended action. It provides a single decision point where users can exit safely or proceed based on the available security information.

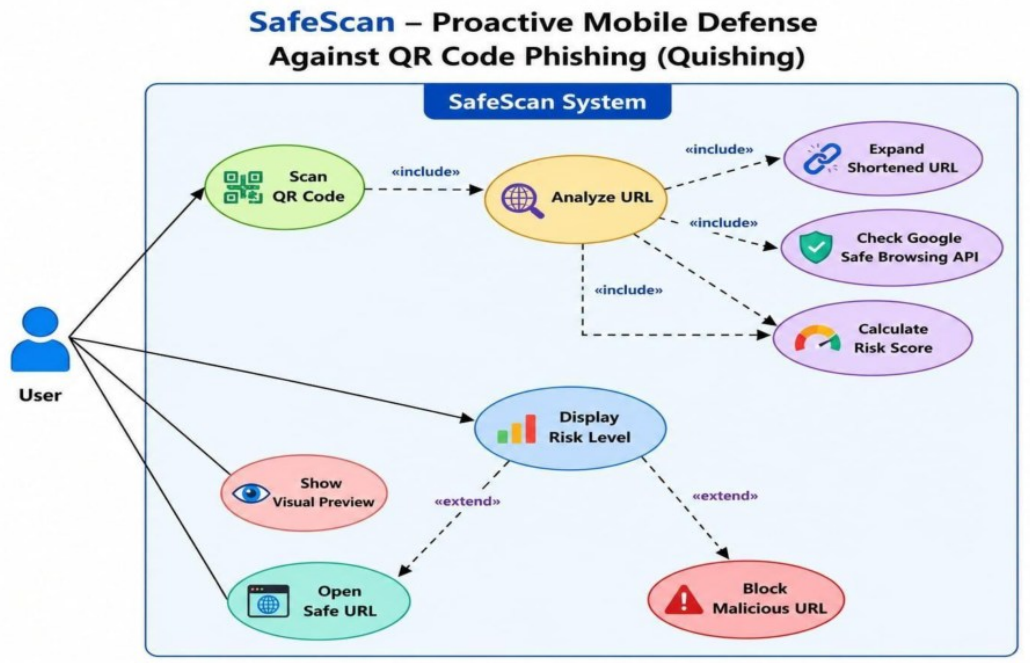

Figure 2: SafeScan Use Case Diagram

IMPLEMENTATION

BACKEND PROCESSING

The FastAPI backend exposes REST endpoints for URL analysis and image-based QR decoding. The URL analysis endpoint validates the input, normalizes it, and forwards it to the main analysis pipeline. The image endpoint decodes Base64 input, extracts QR content using OpenCV, validates the resulting URL, and invokes the same analysis pipeline. This shared processing path keeps results consistent across camera and gallery workflows.

REDIRECT EXPANSION

Shortened URLs and redirect chains can conceal the final destination. SafeScan uses an asynchronous HTTP client to follow redirects and identify the final URL before applying the analysis logic. This makes the security report more informative than a report based only on the initial shortened address.

HEURISTIC SECURITY ANALYSIS

The heuristic layer evaluates measurable URL features. Examples documented in the project include protocol type, suspicious keywords, URL length, query-string characteristics, IP-based addresses, excessive subdomains, unusual characters, and malicious or blacklisted URL checks where available. The resulting score is constrained to the 0–100 range to provide an interpretable security indicator.

ERROR HANDLING

The system is designed to handle invalid URLs, empty inputs, invalid Base64 images, QR decoding failures, backend connectivity failures, and network errors without crashing the mobile application. The mobile interface communicates actionable error information so that users can correct the input or verify backend availability.

DEPLOYMENT CONTEXT

The project documentation describes a development and demonstration setup in which the FastAPI backend runs on a computer and the Expo application runs on a mobile device, typically over the same Wi-Fi network. The documented requirements include Python 3.x, Node.js with Expo CLI, Chromium for Playwright, OpenCV, and an Android device with a camera.

 © Author(s) | CC BY 4.0 International License.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

RESULTS AND DISCUSSION

The results of SafeScan are presented as functional outputs observed during testing in a typical mobile scanning scenario. On launching the application, the user is presented with a landing section and a scanner module that requests camera permissions where necessary. The scanning frame enables the user to align a QR code and initiate scanning explicitly, reinforcing the safety-first design.

After a QR code containing a URL is scanned, the application transitions to an auditing stage. The backend expands the URL, evaluates heuristic signals to compute the risk score, and attempts to capture a screenshot of the destination. After completion, the Safety Report displays the decoded or expanded URL, numerical risk score, qualitative verdict, and visual preview when screenshot generation succeeds.

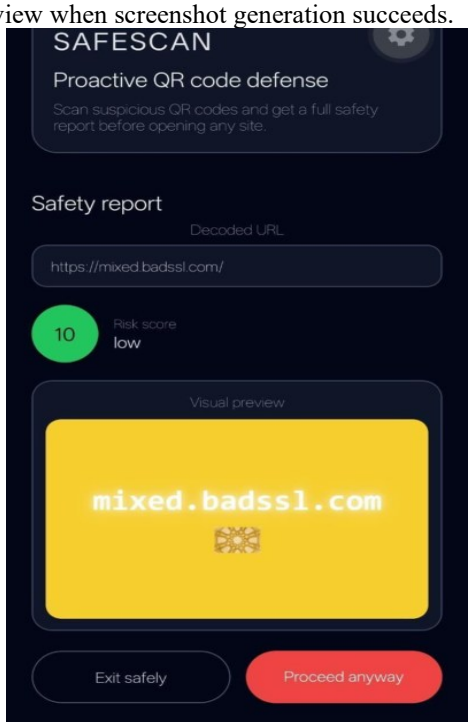

Figure 3: Low-Risk Safety Report (Risk Score: 10)

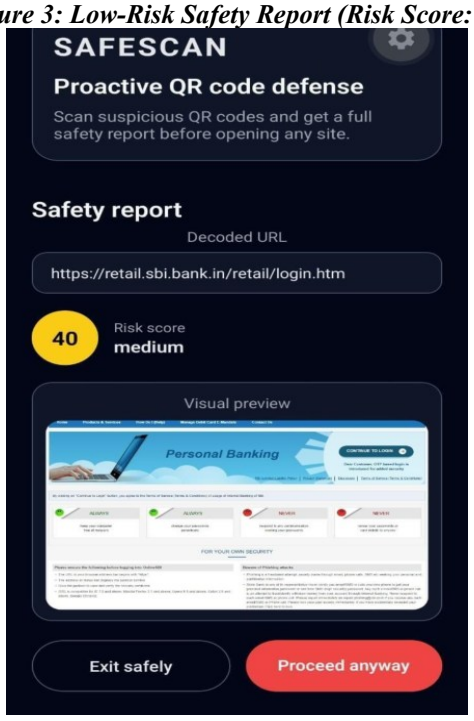

Figure 4: Medium-Risk Safety Report (Risk Score: 40)

 © Author(s) | CC BY 4.0 International License.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

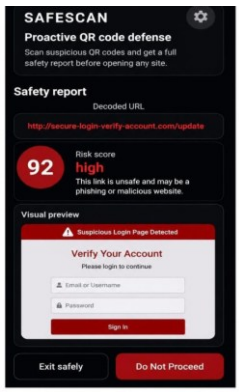

Figure 5: High-Risk Safety Report (Risk Score: 92)

The supplied project outputs illustrate three risk examples: a score of 10 for a low-risk result, 40 for a medium-risk result, and 92 for a high-risk result. The high-risk screen includes a suspicious login-style visual preview and a clear recommendation not to proceed. These examples demonstrate the intended presentation of numerical scoring, qualitative classification, and visual context.

RESULTS AND DISCUSSION

FUNCTIONAL BEHAVIOUR

When screenshot capture succeeds, the Safety Report includes the visual preview returned by the backend. This provides immediate contextual information and can help users recognize pages that imitate legitimate designs. When the backend is unreachable, the application displays an informative connectivity message. If a QR image cannot be decoded or does not contain a valid URL, the system returns a controlled error and the interface communicates the failure without crashing.

OBSERVED SAFETY PROPERTIES

The observed results demonstrate a complete workflow from QR acquisition to risk reporting while maintaining user-controlled navigation. The design prevents the application from treating scanning as permission to immediately visit the destination. Instead, every URL is passed through validation and analysis before the user is offered a navigation decision.

FUNCTIONAL TEST SUMMARY

Test Case	Expected Result	Status
QR Camera Scan	QR URL extracted and sent for analysis	Pass
Uploaded QR Image	Image decoded and URL analysed	Pass
URL Validation	Invalid URL rejected with controlled error	Pass
Redirect Expansion	Final destination identified	Pass
Risk Scoring	0–100 score and verdict generated	Pass
Visual Preview	Screenshot displayed when capture succeeds	Pass
Safety Report	URL, score, verdict and action displayed	Pass
Backend Failure	Connectivity error shown without crash	Pass
Invalid QR	Controlled decoding error displayed	Pass

Table 1: Functional Test Summary of SafeScan

The project does not report a statistical accuracy benchmark or a large labelled test-set evaluation. Accordingly, the present paper reports functional behaviour and documented example outputs rather than claiming a measured phishing-detection accuracy percentage.

ACKNOWLEDGEMENT

 © Author(s) | CC BY 4.0 International License.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

I would like to express my sincere appreciation to Mrs. K.S.S. Soujanya Kumari, Assistant Professor, Department of Computer Science and Systems Engineering, Andhra University College of Engineering (A), for her tremendous support, guidance, and encouragement throughout the development of this project. I would also like to thank Prof. D. Lalitha Bhaskari, Head of the Department of Information Technology and Computer Applications, Andhra University College of Engineering (A), for her valuable guidelines and encouragement. I express my hearty thanks to Prof. K. Venkata Subbaiah, Principal, Andhra University College of Engineering, for his support toward the successful completion of the project. I also thank the teaching and non-teaching staff of the Department of Information Technology and Computer Applications for their constant support. Finally, I express my gratitude to my parents and friends for their blessings and encouragement.

CONCLUSION

SafeScan demonstrates a practical approach for reducing QR-based phishing risk by inserting an analysis and reporting layer between QR scanning and web navigation. The project integrates a cross-platform mobile application with a Python backend to implement QR acquisition, redirect expansion, heuristic risk scoring, and optional visual preview generation. The resulting Safety Report gives users a structured and interpretable view of the destination before navigation and discourages automatic access to potentially unsafe URLs.

From an engineering perspective, the project demonstrates the use of asynchronous backend workflows for network and browser automation and Expo-based mobile development for rapid device testing. The design emphasizes explainability and usability so that non-technical users can understand the result. The current heuristic approach is a baseline defence and should not be considered a replacement for comprehensive enterprise threat intelligence.

FUTURE SCOPE

The project identifies several directions for future enhancement: integration with stronger reputation and safe-browsing services; machine-learning-based phishing classification; DNS, WHOIS, TLS and domain-similarity analysis; stricter browser sandboxing and resource controls; caching and client-level rate limiting; multi-QR and batch processing; production deployment with HTTPS, authentication and structured logging; and more explainable scoring details in the user interface.

REFERENCES

1) Reinheimer, B., Schaub, F., Mare, S., & Egelman, S. (2015). QR Code Security: A Study of Mobile QR Code Scanner Applications. Proceedings of the 10th International Conference on Availability, Reliability and Security (ARES), IEEE.
2) Rafsanjani, A. S., Kamaruddin, N. B., Rusli, H. M., & Dabbagh, M. (2023). QSecR: Secure QR Code Scanner According to a Novel Malicious URL Detection Framework. IEEE Access, Vol. 11, pp. 92523–92539.
3) Sharevski, F., Devine, A., Pieroni, E., & Jachim, P. (2022). Gone Quishing: A Field Study of Phishing with Malicious QR Codes. arXiv Preprint, Cornell University.
4) Akram, M. H., Sood, K., Hassan, M. U., & Subba, B. (2025). Exemplifying Emerging Phishing: QR-Based Browser-in-The-Browser Attack. IEEE Networking Letters.
5) Jain, A. K., & Gupta, B. B. (2018). Towards Detection of Phishing Websites Using Machine Learning Techniques. Telecommunication Systems, Springer Nature.
6) Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing Detection Based on Machine Learning and URL Features. Journal of Information Security and Applications, Elsevier.
7) Gupta, B. B., Arachchilage, N. A. G., & Psannis, K. E. (2018). Defending Against Phishing Attacks: Taxonomy of Methods, Current Issues, and Future Directions. Telecommunication Systems, Springer.
8) Wang, Y. (2022). A Survey of Phishing Detection: From an Intelligent Countermeasures Perspective. Proceedings of the International Conference on Telecommunications, Optics and Computer Science (TOCS), IEEE.
9) Google LLC. (2024). Google Safe Browsing API Documentation. Google Developers.
10) Microsoft Corporation. (2024). Playwright for Python Documentation. Microsoft Learn.
11) FastAPI Developers. (2024). FastAPI Official Documentation. FastAPI Technologies.
12) OWASP Foundation. (2024). OWASP Phishing Prevention Cheat Sheet. OWASP Project Documentation.
13) OpenCV Team. (2024). OpenCV Documentation: Computer Vision Library. OpenCV Foundation.
14) Meta Platforms Inc. (2024). React Native Official Documentation. Meta Open Source.
15) Expo Developers. (2024). Expo SDK Documentation. Expo Documentation.

 © Author(s) | CC BY 4.0 International License.