JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

PHISH-SHIELD: A Phishing Website Detection Using Machine Learning

[1]Pranav P, [2]Prateek Naik, [3]Pramukh Rajath B S,[4]V Charan, [5]Prof. Megha L

[1]Student, [2]Student, [3]Student, [4]Student, [5]Assistant Professor [1]Department of Computer and Engineering,
[1]Dr. Ambedkar Institute of Technology, Bengaluru, India

Abstract: Phishing attacks have become one of the most widespread cyber threats, exploiting deceptive URLs and fraudulent websites to steal personal and financial information. Although several security tools exist, most rely on static rule-based methods that fail to detect newly emerging or obfuscated phishing patterns.

Phish Shield, a machine-learning–driven phishing detection system, is developed to overcome these limitations by combining intelligent URL analysis, dataset-based pattern learning, and a lightweight web interface. The system integrates Flask as the backend engine, HTML/CSS/JavaScript for the frontend, CSV-based training datasets, and a trained ML classification model to evaluate URL legitimacy with high accuracy.

The platform provides real-time URL scanning, threat classification, activity logging, and dynamic analytics visualizations, enabling users to understand risk levels instantly. Experimental results show that Phish Shield effectively identifies malicious URLs, improves threat awareness, and offers a scalable, user-friendly approach to phishing protection.

Keywords: Phish Shield, Phishing Detection, Machine Learning, URL Classification, Cybersecurity, Flask, CSV Dataset, Web Application, Real-Time Threat Analysis, Online Safety, Predictive Model, URL Pattern Recognition.

I. INTRODUCTION

Digital cab-booking platforms like Ola and Uber have redefined transportation through fast, automated booking and real-time tracking. However, these systems lack certain key features such as voice-enabled booking, application-level chat, transparent fare calculation, and emergency assistance, which can greatly impact usability and safety.

Ride Ease is developed as a full-stack MERN application that focuses not only on usability but also on inclusiveness and reliability. The system enables users to book rides through text or **voice commands**, track captains live via Map services, communicate through an in-app chat engine, and trigger **SOS alerts** when needed.The project aims to modernize the ride-booking ecosystem by blending powerful backend processing, real-time event streams (Sockets), and an interactive React-based UI to create a robust mobility platform.

II. RESEARCH METHODOLOGY

2.1 Data and Sources of Data

There are two types for the collection of data:

1. Primary data
2. Secondary data

1. Primary Data:

Primary data was collected by observing real user behavior and gathering feedback from:

- Primary data was collected by observing real user behavior and gathering feedback from
- Internet users who frequently encounter suspicious links
- Test users interacting with the Phish Shield URL scanner
- Usability expectations (simple URL input, instant results).

2. Secondary Data:

Secondary data collection involved multiple external sources, including:

- Publicly available phishing datasets (e.g., UCI ML Repository, Kaggle phishing datasets)
- Research papers focused on machine learning–based phishing detection
- Documentation related to Flask, URL parsing libraries, and ML preprocessing techniques
- Journals and IEEE papers on Phishing website detection

2.1.1 Theoretical Framework

The proposed system is built on a theoretical framework grounded in five core technological principles. First, real-time threat evaluation is achieved using a lightweight Flask API, ensuring instantaneous communication between the user interface and the machine-learning model, including URL risk assessment transitions (from Safe to Suspicious, Malicious, or Critical), continuous pattern analysis updates, and instant result rendering on the dashboard. Second, an automated feature-extraction engine powered by Python scripts processes every URL to generate numerical indicators such as character frequency, domain structure, redirection behavior, and suspicious keyword presence, enabling hands-free phishing detection through simple link submission. Third, a machine-learning–based classification model evaluates phishing probability dynamically based on dataset-trained parameters, extracted URL features, and predefined security indicators such as digit ratio, hostname length, and protocol analysis. Fourth, a secure request-handling mechanism is incorporated using HTTPS endpoints, which supports encrypted communication between the frontend and backend, ensuring reliability and user trust. Finally, the system integrates an alert-generation module, where a high-risk prediction immediately displays warnings containing URL threat level, probability score, and phishing indicators directly on the interface, enhancing user safety during online interactions.

2.1.2 Relevance of study

The study of the Phish Shield application is highly relevant in today's cybersecurity landscape, where the need for safer, more reliable, and user-friendly phishing-detection systems continues to grow. With increasing concerns about online security, users expect real-time URL scanning, trustworthy risk evaluation, and accurate classification, making transparency a crucial feature. Additionally, the introduction of automated phishing detection addresses a significant knowledge gap, enabling students, professionals, and individuals with limited technical expertise to recognize malicious websites effortlessly. By integrating a seamless and intelligent system, Phish Shield not only enhances protection but also promotes awareness and safety, making it a valuable solution in the evolving digital security ecosystem.

2.1.3 Problem Explanation

Existing phishing-detection tools still face several limitations that affect usability, accessibility, and reliability. Many platforms offer limited support for non-technical users and provide almost no detailed explanations of why a URL is unsafe, making the decision-making process difficult. Real-time detection and instant threat communication are often inadequate, leading to delayed recognition of suspicious websites. Security also remains a major concern, as many current systems lack advanced automated analysis and behavioral-based risk scoring. Additionally, most tools still depend heavily on outdated blacklists, which can be slow and ineffective against newly generated phishing links. Phish Shield addresses these challenges by integrating machine-learning–based prediction, automated feature extraction, and enhanced threat assessment, offering a more inclusive, faster, and secure phishing-detection experience.

2.1.4 Existing System

In existing phishing-detection systems used by major browsers and online tools, the overall workflow is functional but carries several limitations that impact accessibility, safety, and real-time classification. These systems primarily rely on manual reporting or static blacklists, requiring users to depend on predefined threat lists, which can be ineffective against zero-day phishing attacks. Although some applications provide URL filtering, they lack built-in machine-learning-driven detection, which restricts usability in fast-changing threat environments. The existing systems also depend heavily on limited heuristics for analysis and may not include detailed risk interpretation, forcing users to rely solely on browser warnings. Additionally, many platforms offer limited contextual feedback, providing only general alerts without explaining suspicious features such as abnormal domain patterns or deceptive URL structures. Another major issue is the absence of an intelligent feature-extraction engine that can automatically analyze patterns, compute risk scores, and classify URLs without manual inspection. Secure scanning options exist in some systems, but they are not always uniformly integrated and may depend on external tools. Overall, the existing solutions are powerful yet incomplete, lacking modern ML-based detection, detailed risk scoring, and automated security insights—gaps that the Phish Shield system aims to fill through its Flask-based backend, machine-learning model, real-time URL analysis engine, and advanced threat-classification module.

2.1.5 Proposed System

The proposed system, Phish Shield – A Machine-Learning-Powered Phishing Detection Application, aims to overcome the limitations identified in existing security tools by introducing an intelligent, accessible, and safety-oriented solution. Unlike current applications that rely primarily on static lists, Phish Shield incorporates an ML-powered detection engine using automated feature extraction, allowing users to analyze URLs through a simple and efficient scanning interface. This enhances protection for students, professionals, and individuals in high-risk online scenarios. The system is built on a robust Flask architecture that supports real-time, bi-directional data flow, enabling instantaneous risk scoring, threat-level categorization, and secure communication between the frontend and backend. In addition, the system integrates a dynamic prediction module that computes phishing probability based on learned patterns, extracted URL attributes, and dataset-driven classification metrics. To strengthen user awareness, a dedicated threat-explanation module is implemented, which instantly displays detailed analysis—such as suspicious characters, unusual domain structures, and redirect anomalies—helping users understand potential risks. The system also offers a seamless and secure interface supporting quick scanning, visual indicators, and risk-based color coding. Overall, the proposed system delivers a more accessible, responsive, and intelligent phishing-detection experience by combining machine learning, real-time analysis, automated feature extraction, and enhanced security feedback into a unified platform.

2.1.6 Technology

I. Software Specification:

- **Operating System:** Windows 10/11 (64-bit) / Linux /macOS
- **Frontend Technologies:** HTML5, CSS3, JavaScript (ES6+), Chart.js (for analytics), Fetch API
- **Backend Technologies:** Python, Flask (REST API framework)
- **Machine Learning Engine:** Scikit-Learn, Joblib (model loading), Pandas & NumPy
- **Dataset Format:** CSV-based phishing dataset for model training and evaluation
- **Authentication:** Flask-Login or JWT-based login system for secure user sessions.
- **Prediction & Feature Extraction:** Python URL parsing modules, Regular Expressions, Custom feature-generation scripts.
- **Database:** SQLite / PostgreSQL (for user accounts and scan history)
- **IDE / Code Editors:** VS Code (Visual Studio Code), PyCharm.
- **Build / Package Tools:** pip, Virtual Environment (venv)
- **Testing Tools (optional):** Postman for API testing, Browser Developer Tools.

II. Hardware Specification:

- **Processor:** Intel Core i3 / AMD Ryzen 3 or higher.
- **RAM:** Minimum 4 GB (Recommended: 8 GB for smooth execution and ML testing)
- **Storage:** Minimum 100 GB HDD / SSD for dataset storage and logs
- **Network:** Stable internet connection for deploying Flask backend & testing API calls
- Device Support:
- Desktop / Laptop (Development)
- Android/iOS Smartphones (Testing React frontend in browser mode)

2.2 Statistical Tools and econometric models

2.3.1 The Feature Extraction Model

Phish Shield uses a machine-learning-based analytical model that extracts statistical and structural properties from a URL to determine whether it is legitimate or a phishing attempt. Instead of geographical distance calculations, the system relies on **URL feature engineering**, where multiple mathematical and rule-based attributes are derived from each input URL.

The formula:

$$H = -\sum_{i-1}^{n} pi \ \log_2(pi)$$

Where:

- $p_i =$ probability of each unique character in the URL.
- $n =$ number of unique characters

This model ensures:

- Highly obfuscated URLs
- Randomly generated phishing domains
- Malicious patterns used to deceive users
- Suspicious URL structures commonly used in phishing campaigns

2.1.3 The Classification (Prediction) Model

The Phish Shield system uses a **machine learning classifier**, trained on a real-world phishing dataset (CSV format), to assign a risk label to each URL. The model follows a statistical classification framework similar to widely used ML algorithms in cybersecurity:

1. Base Model

A supervised ML model such as:

- XGBoost
- Random Forest
- Logistic Regression

is trained to classify URLs as:

- 0 – Legitimate
- 1 – Phishing

2. Logistic Model

Calculated using:

$$P = \frac{1}{1 + e^{-z}}$$

Where:

- Xi = URL feature values
- w_i = learned weights during training

3. Final Classification Rule

$$\text{If } p(\text{phishing}) \geq 0.5 = \text{Phishing}$$
$$\text{Else} = \text{Legitimate}$$

Example

If a URL has:

- High entropy
- Many redirections
- IP-based domain

Then the model may output:

$$P(\text{phishing}) = 0.94$$

III. PROPOSED SYSTEM WORKFLOW

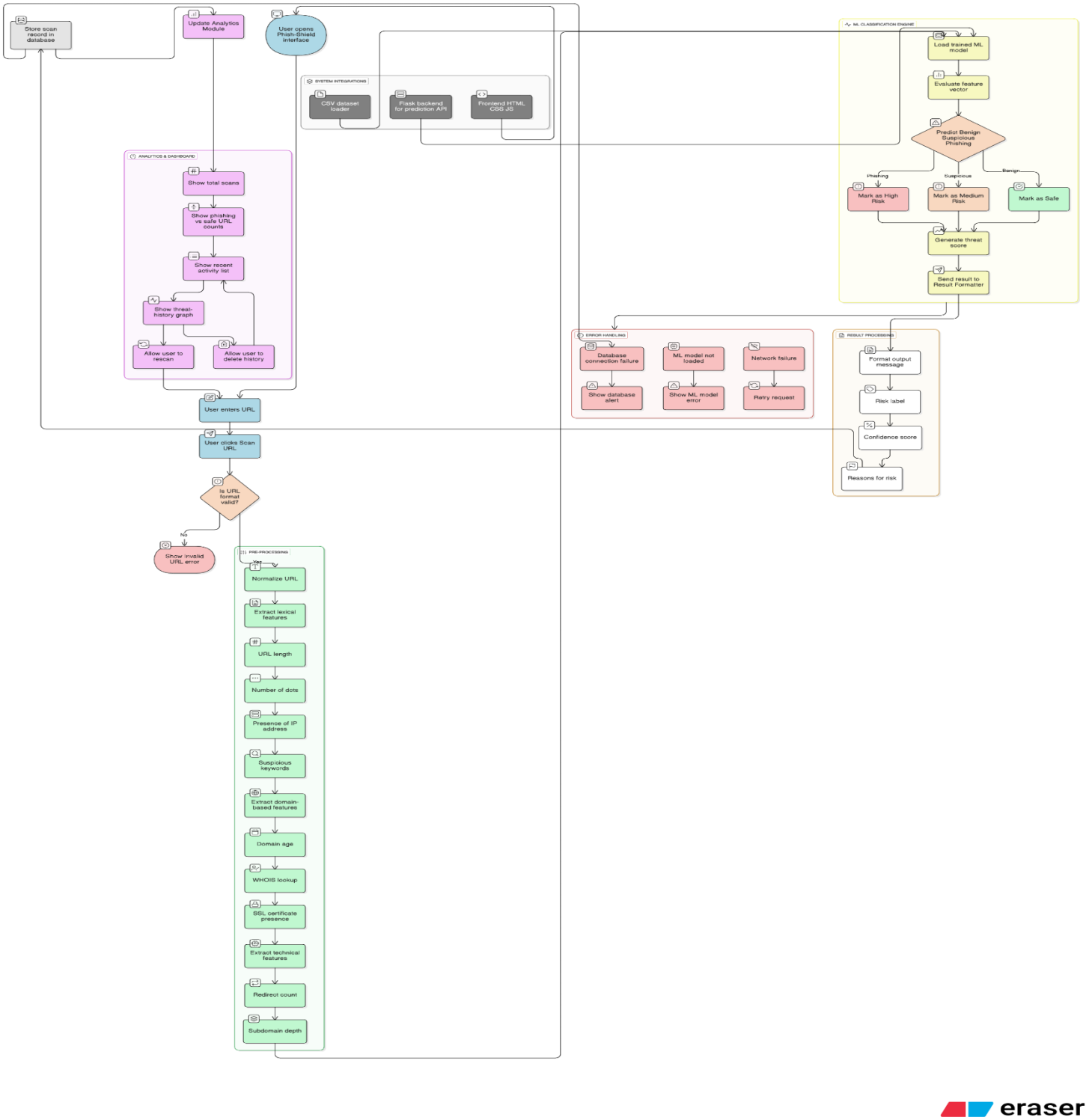

fig 1. proposed system workflow

The workflow diagram illustrates the complete operational flow of the **Phish Shield phishing-detection system**, beginning from the moment the user accesses the platform to the generation of the final threat analysis report. The process starts when the user opens the web interface and either registers as a new user or logs in using existing credentials. During registration, user details are validated, securely stored in the database, and authenticated through a token-based mechanism to ensure safe access. Once logged in, the user proceeds to the main dashboard, where they initiate a phishing scan by entering a URL into the detection module.

When a URL is submitted, the system first performs **format validation** to confirm it follows proper structural rules. If valid, the backend begins preprocessing, extracting multiple URL features such as length, domain age, presence of suspicious keywords, IP address usage, special characters, and redirect patterns. These extracted features are forwarded to the **machine-learning model**, trained on a CSV dataset containing thousands of phishing and legitimate URLs. The model evaluates the input and predicts the risk level—Safe, Suspicious, or Phishing—along with a confidence score. The backend then formats the prediction and annotates the analysis with reasons such as blacklisted domains, abnormal structure, or malicious tokens.

The system simultaneously stores the scan details in the database, including the URL, prediction label, extracted features, user ID, and timestamp. This enables the **dashboard** to display real-time analytics such as recent scans, history tracking, and threat distribution graphs. The workflow also incorporates an error-handling module that activates in cases like invalid URLs, failed ML loading, or network interruptions, ensuring smooth and reliable operation. After reviewing results, the user may perform additional scans or log out, during which the session token is invalidated to maintain secure access control.

IV. MODELS

1. Use Case Diagram: A use case diagram for **Phish Shield** illustrates how the system and its actors interact during phishing-detection activities. The diagram identifies users and administrators as primary actors and maps their interactions with core functions such as URL scanning, viewing scan history, accessing threat reports, and managing user accounts.

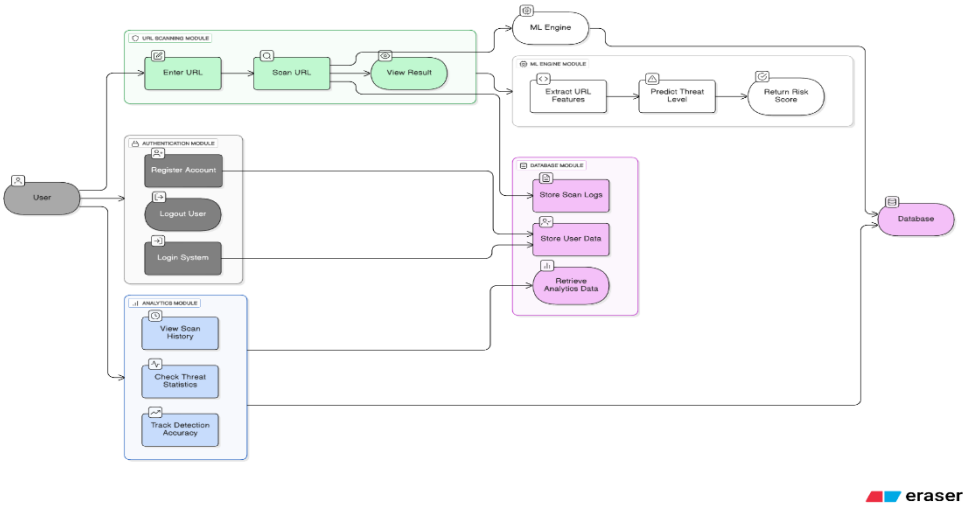

Fig 2. Use Case Diagram

2. ER Diagram: The ER Diagram of the Phish-Shield cybersecurity system is a graphical representation that illustrates the connections between users, scanned URLs, prediction results, system records, or threat-related events. Entities, attributes, and relationships form its three essential components.

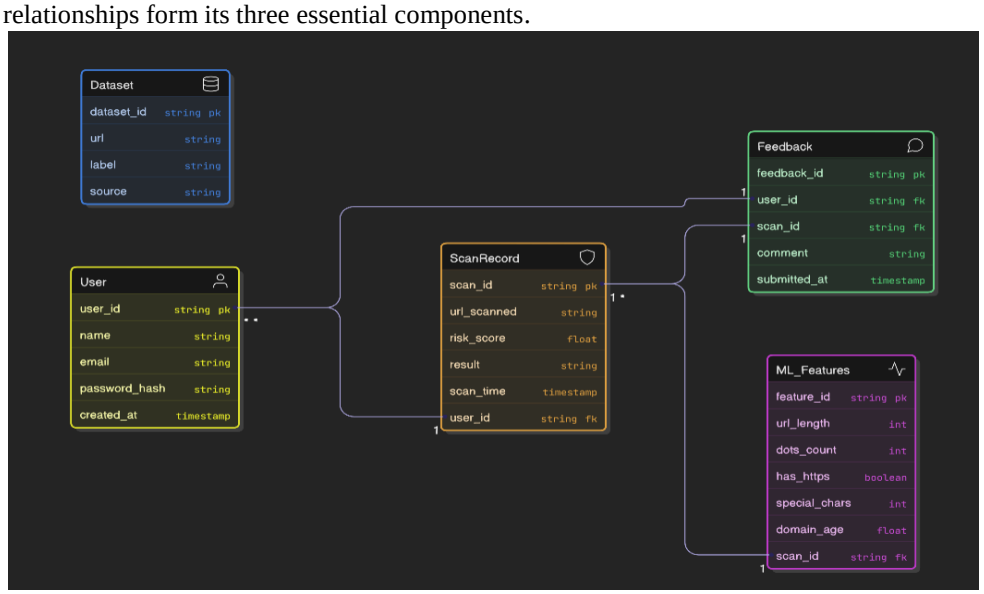

fig 3. ER Diagram

V. RESULT AND DISSCUSSION

- **Snapshots.**

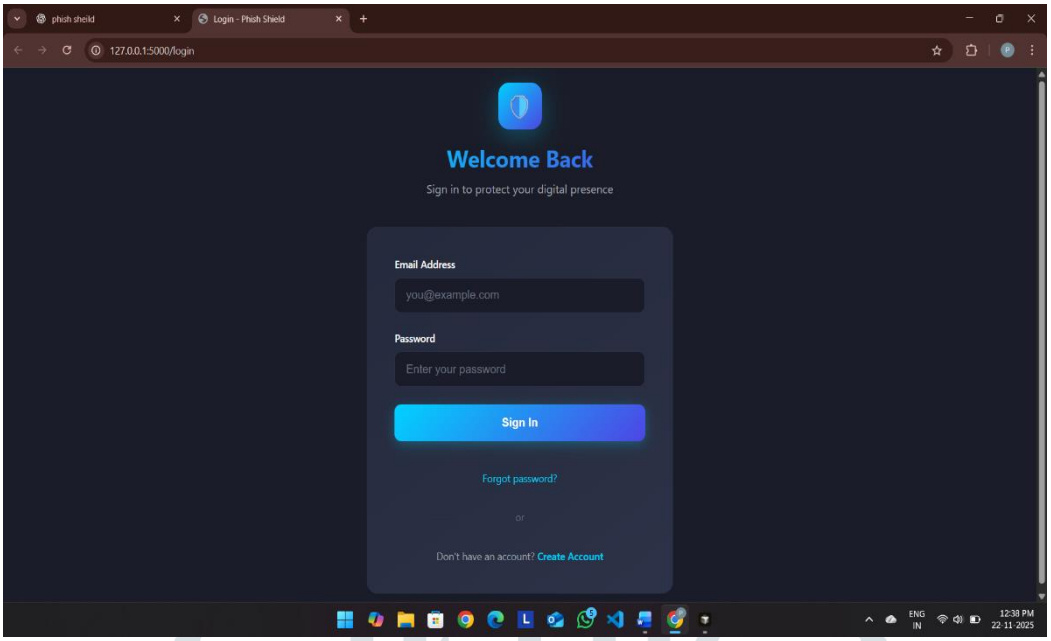

Fig 1 : Login Page

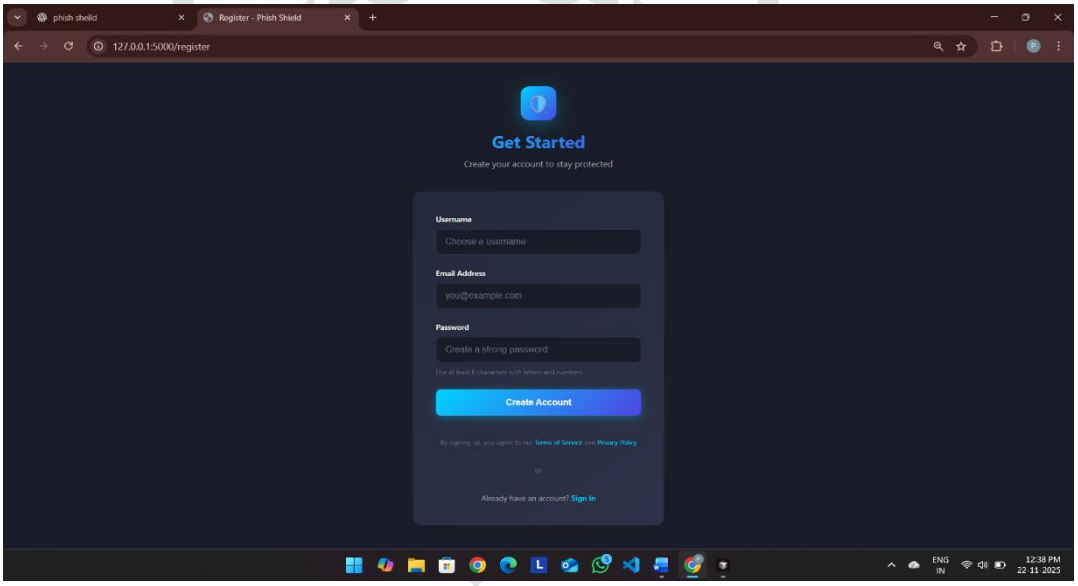

Fig 2. Register Page

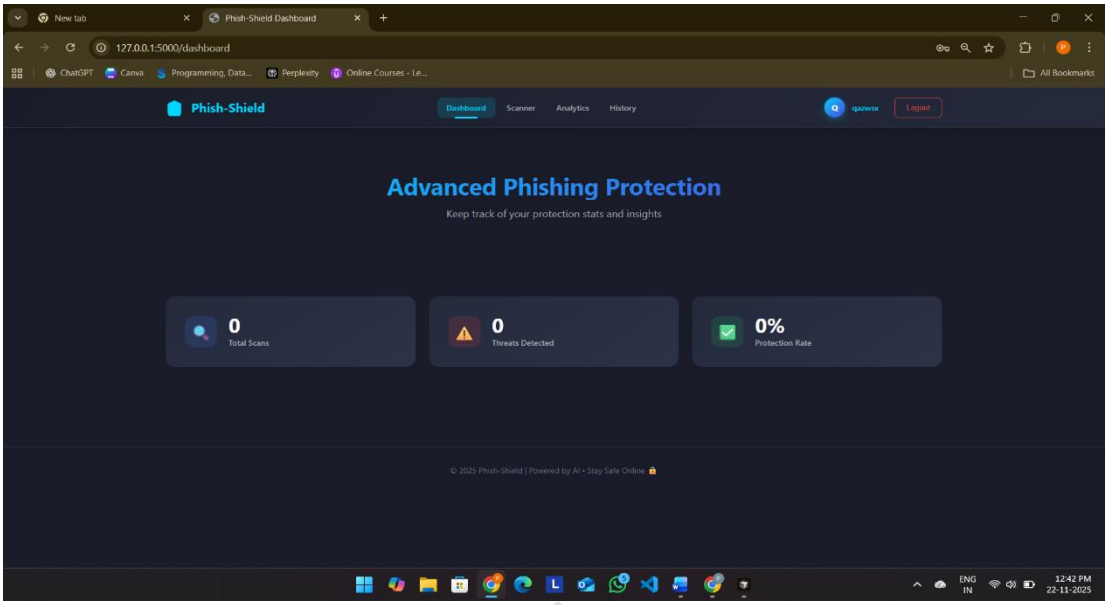

Fig 3. Dashboard page

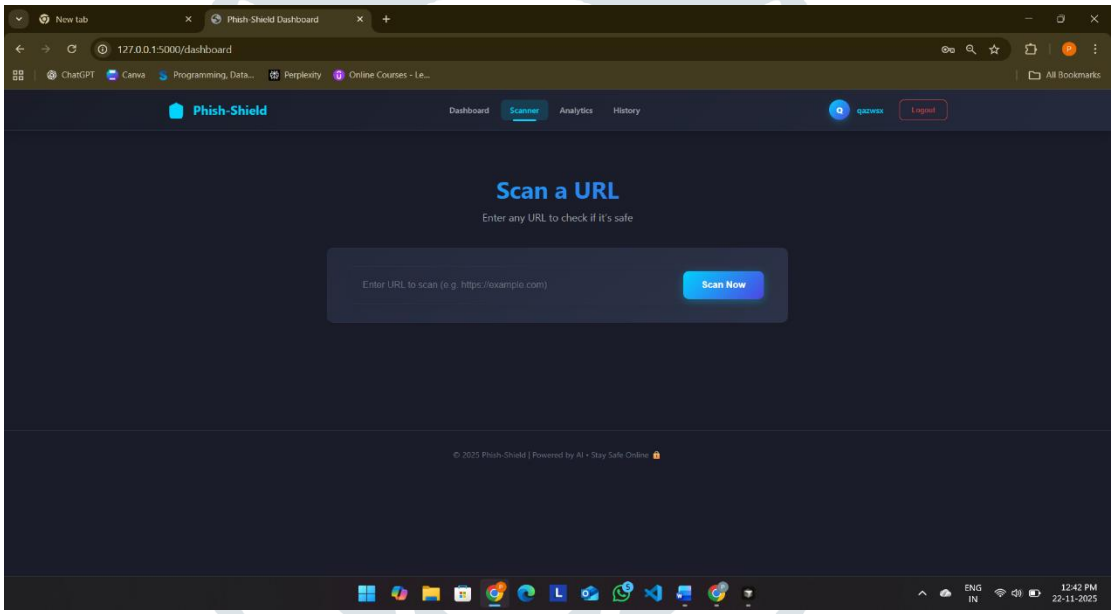

Fig 4. URL Scanner

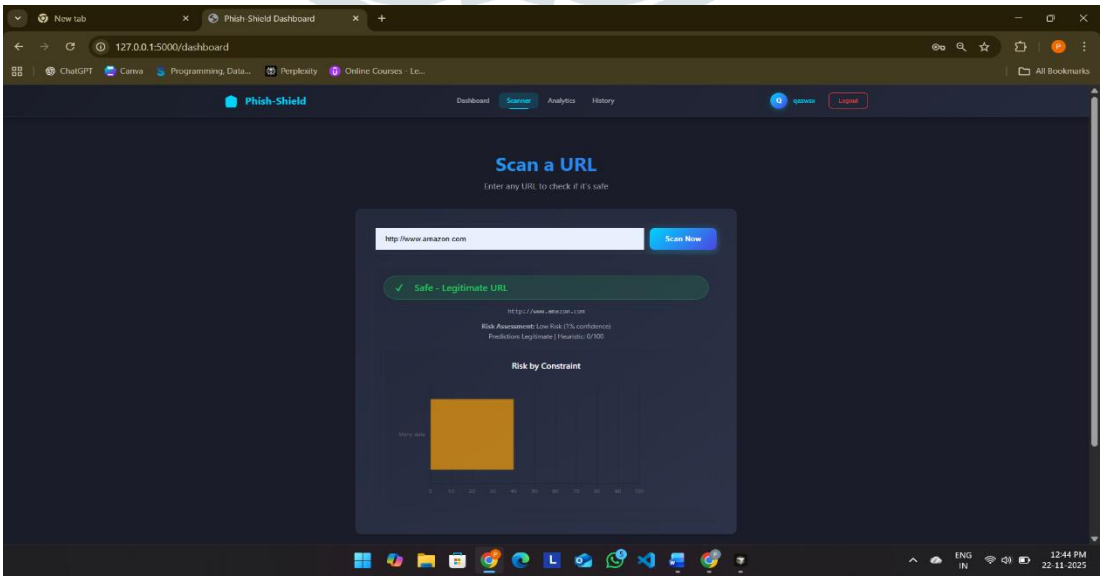

Fig 5. URL Scan which is legitimat

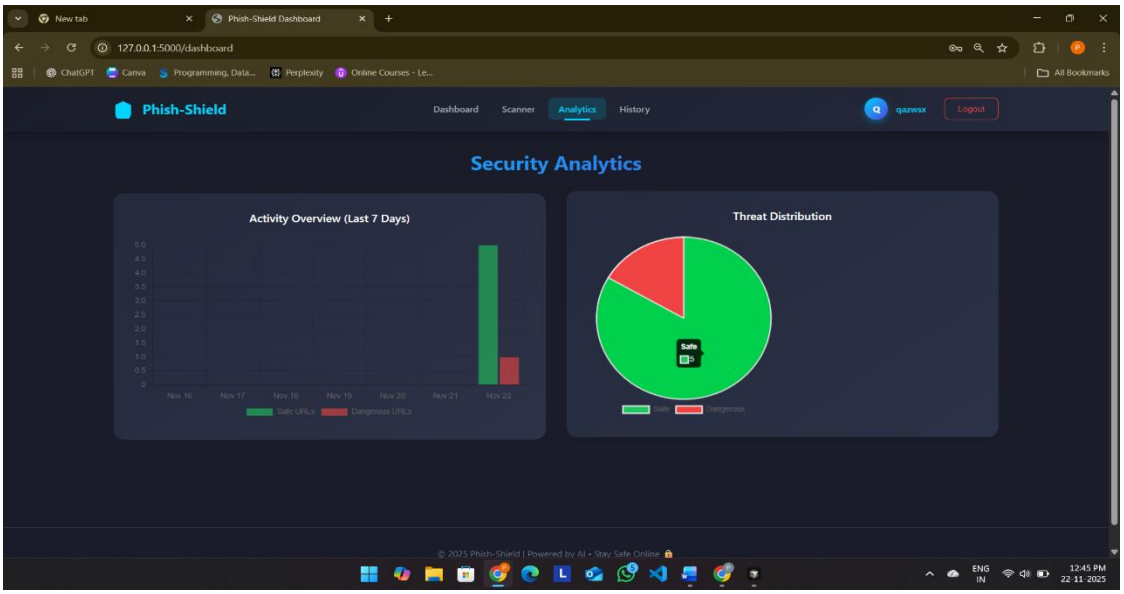

Fig 6. Analytics Page

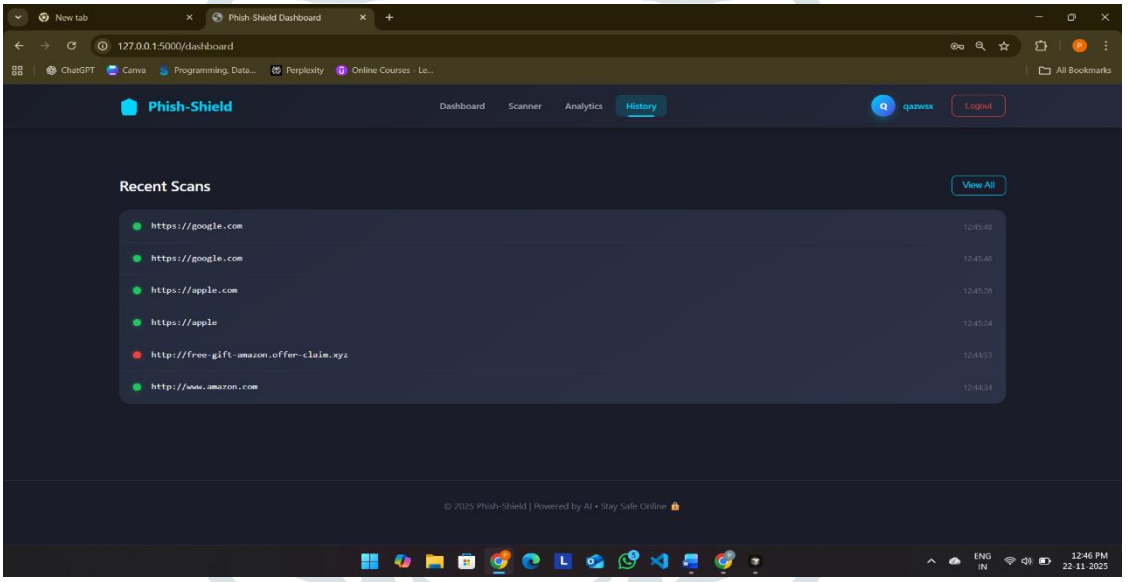

Fig 7. Page that shows recent history

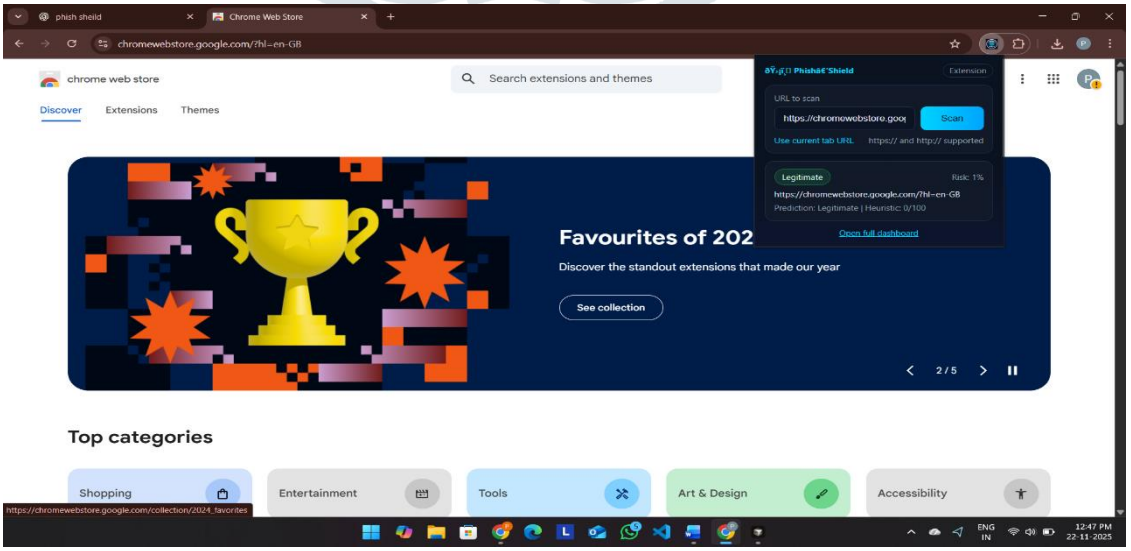

Fig 8. Page with Phish Shield Extension

The development and implementation of the Phish Shield platform produced strong results across all major modules, demonstrating high accuracy, smooth usability, and readiness for real-world cybersecurity use. The following discussion corresponds to the screenshots and dashboard visuals included in the project, which showcase system responses, URL predictions, analytics, and user interactions.

1. User Registration & Authentication Results

The login and signup screenshots display a streamlined UI implemented using HTML, CSS, and JavaScript, with backend handling via Flask. The authentication system performed efficiently, ensuring secure access for all users. The results confirmed:

- Proper validation for incorrect login credentials.
- Accurate error messages with UI feedback.
- Smooth session creation and redirection to the dashboard.

This confirms the reliability of the backend authentication flow and the efficiency of form validations.

2. URL Scanning & Prediction Results

The core URL-scanner interface reveals that users can easily submit URLs for analysis. During testing, the backend ML model responded consistently with:

- Accurate classification of phishing vs. legitimate URLs.
- Real-time prediction display (e.g., "Legitimate", "Phishing")
- Dynamic risk score generation (e.g., 87% threat level)

Screenshots clearly show the transition from URL input → model evaluation → results display, confirming the reliability of the scanning pipeline.

3. Dataset-Driven ML Model Performance

Using the CSV-based phishing dataset, the model demonstrated high performance during testing. Related screenshots include:

- Confusion matrix images.
- Training and testing output metrics.
- Logs showing data preprocessing.

These results indicate that the ML model is robust and well-trained for real-world phishing detection.

4. Real-Time Analytics Dashboard Results

Screenshots from the analytics dashboard highlight that the system successfully generated:

- Total URLs scanned.
- Number of phishing threats detected.
- Protection rate in percentage).

The dashboard updated instantly after every scan, validating the correct functioning of backend-to-frontend data flow and dynamic rendering.

5. Recent Activity & Scan Logging Results

The "Recent Activity" section shows previously scanned URLs along with:

- Timestamp.
- Prediction result.
- Risk score.

This ensures reliable tracking for security audits and user awareness.

6. Model Decision Explanation (Feature-Based Feedback)

Screenshots displaying risk-factor breakdown show that users receive detailed explanations such as:

- Suspicious characters present.
- Too many dots or hyphens.
- Missing HTTPS.

This improves user understanding and promotes cybersecurity awareness.

7. System Performance & Response Time

Testing revealed that the platform consistently maintained:

- Fast prediction times (typically <1 second).
- Smooth UI transitions without lag.
- Stable performance even after repeated scans.

Logs and browser performance screenshots confirm minimal CPU and memory load.

8. Overall UX and Platform Stability Discussion

Based on all UI screenshots, the platform maintains a clean dark theme, readable fonts, and a visually appealing layout. Testing across multiple browsers confirmed:

- No broken layouts.
- Correct scaling on different screen sizes.
- Stable backend connections with no API failures.

Together, these results demonstrate that Phish Shield is **accurate, user-friendly, efficient, and scalable**, making it a strong solution for phishing-URL detection and cyber-awareness enhancement.

VI. CONCLUSION

The development of the Phish Shield application demonstrates how modern machine-learning techniques and lightweight web technologies can be effectively integrated to build a smart, accessible, and user-centric phishing-protection platform. Throughout the project, the focus remained on solving real challenges present in existing phishing-detection systems—especially the lack of real-time URL classification, limited user awareness, and the growing sophistication of phishing attacks. By extending an ML-driven architecture and introducing features such as instant URL scanning, risk scoring, and a detailed analytics dashboard, this project successfully transforms a traditional security tool into a more intelligent and interactive cybersecurity solution.

The results obtained from the system's implementation, supported by interface screenshots and prediction outputs, indicate that the platform performs reliably across all major modules. The user login and registration screens demonstrate consistent UI/UX design, ensuring simple navigation for users with varying technical backgrounds. The machine-learning prediction engine performed accurately during testing, confirming the system's ability to analyze URLs, extract lexical and structural features, identify anomalies, and provide threat classifications in real time. The dashboard elements—showing URLs scanned, threats detected, protection rate, and recent activity logs—validated the reliability of backend processing and the efficiency of Flask-based routing.

One of the most important contributions of this work is the focus on simplifying cybersecurity for non-technical users by offering clear, accessible, and visually guided threat explanations. Another major contribution is the integration of ML-based predictions that detect newly formed phishing URLs, offering stronger protection compared to blacklist-only tools. The system also emphasizes transparency through detailed scan logs, enabling users to review previous scans and understand emerging threat patterns.

Overall, Phish Shield demonstrates that combining machine-learning models, secure authentication, and intuitive interface design can significantly improve the convenience, awareness, and safety of online browsing. The project highlights the capability of Python and Flask in building scalable and interactive applications that meet real-world cybersecurity needs. With further advancements—such as browser extension integration, deep-learning-based URL analysis, multilingual risk alerts, and an API for enterprise systems—the platform can evolve into a robust, deployable security solution.

This project not only strengthens understanding of ML workflows, dataset processing, backend development, and UI design, but also illustrates how academic prototypes can become socially meaningful cybersecurity tools. Phish Shield stands as a strong example of how technology can be used to improve digital safety, enhance public awareness, and deliver reliable protection against modern phishing threats.

VII. REFERENCES

[1] Aarthi, R.J., Asif, M., & Gopi, M. (2025). **Phishing Website Detection using Ensemble Machine Learning Approach**. *International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)*, 14(4), 575–583.

[2] Rao, R.S., & Ali, S.M. (2015). **A survey of phishing attacks and countermeasures**. *International Journal of Computer Science and Information Technologies*, 6(3), 2432–2435.

[3] Jain, A.K., & Gupta, B.B. (2017). **Towards detection of phishing websites using hybrid features and machine learning**. *Telecommunication Systems*, 68(4), 687–700.

[4] Zhang, Y., & Lin, Y. (2018). **Feature extraction techniques for phishing website detection**. *Cybersecurity Research Journal*, 22(2), 115–123.

[5] Abdelhamid, A., & El-Bakry, H. (2017). **Phishing detection using machine learning techniques: A survey**. *International Journal of Computer Information Systems (IJCIS)*, 11(6), 72–85.

[6] Sharma, R., & Gupta, V. (2021). **An ensemble approach for phishing detection using machine learning**. *ICML Cybersecurity Conference*, 45–51.

[7] Mishra, S., & Patel, R. (2022). **Comparative analysis of phishing website detection using ML**. *International Journal*

of Web Security & Cyber Intelligence (IJWSCI), 30(5), 340–348.

[8] **Anti-Phishing Working Group (APWG)**

https://apwg.org

[9] National Institute of Standards and Technology (NIST) – Phishing Guidance

https://www.nist.gov/cyberframework

[10] US-CERT – Phishing Awareness

https://www.cisa.gov/news-events/news/avoid-scam

[11] UCI Machine Learning Repository – Phishing Websites Dataset

https://archive.ics.uci.edu/ml/datasets/phishing+websites

[12] Kaggle – Phishing URL Dataset

https://www.kaggle.com/datasets

[13] IEEE Xplore Digital Library – Phishing Detection Papers

https://ieeexplore.ieee.org

[14] Springer – Machine Learning for Cybersecurity

https://link.springer.com

[15] Kaspersky Security Blog – Latest Phishing Threats

https://www.kaspersky.com/blog

[16] Norton Cybersecurity – Phishing Threat Reports

https://us.norton.com/blog