IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429

Volume 5, Issue 2, May 2025

Impact Factor: 7.67

Real-Time Phishing URL Detection in Chat Application Using Machine Learning and Flutter

Aishwarya Kalamkar[1], Neha Vaidya[2], Manasi Nagpure[3], Bhagyashri Tembhurne[4], Pranal Mohadikar[5], Prakash S. Prasad[6]

UG Students, Department of Information Technology[1,2,3,4,5]

Professor and Head, Department of Information Technology[6]

Priyadarshini College of Engineering (Autonomous), Nagpur, Maharashtra, India

Abstract: *Phishing attacks continue to be a critical threat in the digital ecosystem, particularly in real time communication platforms. This document introduces an Android application developed using Flutter, which identifies and prevents phishing URLs in real-time chat through a machine learning model that has been trained with PyCaret and deployed using FastAPI. The core URL classification is driven by the XGBoost algorithm, selected for its exceptional performance in classification tasks. Socket programming ensures real-time message delivery between users, while the integrated FastAPI backend filters malicious URLs before they are transmitted. This integrated architecture offers a practical and scalable solution for enhancing online security. The system's capability to intercept and neutralize phishing threats before they reach the end-user marks a significant contribution to cybersecurity practices in mobile applications.*

Keywords: Phishing Detection, XGBoost, Flutter, FastAPI, PyCaret, URL Classification, Real-Time Security

I. INTRODUCTION

With the increasing use of chat applications, users are more exposed than ever to malicious content, especially phishing URLs that deceive individuals into disclosing sensitive information. Traditional methods of phishing detection rely on blacklists or user reports, which are reactive and insufficient in a fast-paced chat environment.

Phishing is a cyber-attack where attackers impersonate trustworthy entities to trick users into revealing confidential data such as passwords, credit card numbers, and personal identification details. These attacks are often delivered through seemingly harmless messages containing malicious URLs. As real-time chat becomes integral to personal and professional communication, the risk of phishing embedded within such platforms grows rapidly. Hence, there is an urgent need for real-time, automated solutions to detect and block phishing attempts.

A. Problem Statement:

Current phishing detection methods are either delayed or manual, relying heavily on known URL blacklists or user vigilance. These techniques cannot effectively detect zero-day phishing URLs or provide immediate protection in fast-paced messaging environments. There is a lack of integrated, real-time detection mechanisms tailored specifically for mobile chat applications.

B. Need for the Project:

To provide a reliable and responsive solution, this project integrates machine learning and modern mobile technologies to proactively detect and block phishing URLs before
they reach the end-user. By using an XGBoostbased model embedded in a real-time messaging environment built with Flutter and FastAPI, the application aims to ensure seamless, secure communication. This proactive and intelligent approach significantly enhances user safety and prevents data breaches at the point of communication.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, May 2025
Impact Factor: 7.67

While implementing the phishing URL detection system in real-time chat applications, several technical and operational challenges were encountered that shaped the final architecture and methodology:

- **Dataset Quality and Feature Engineering:** The dataset used for training the XGBoost model contained inconsistencies, outdated records, and a significant class imbalance. Balancing the dataset and engineering meaningful URL-based features such as domain entropy, HTTPS presence, and token patterns required careful preprocessing and validation.
- **Real-Time Response Optimization:** One of the primary goals was to maintain a smooth chat experience while analyzing URLs in real time. The integration of the XGBoost model with the FastAPI backend led to latency issues. These were addressed by optimizing the API endpoint and decreasing model prediction time through -based serialization and asynchronous requests.
- **Socket Communication Stability:** Socket programming provided real time communication between users; however, there are challenges related to ensuring upon persistent connection, consistently managing disconnected clients, avoiding repeated messages etc. this occurred more frequently when in a mobile testing environment.
- **Cross-Platform Flutter Integration:** Although Flutter provides a unified UI framework, interfacing it seamlessly with Python-based FastAPI and managing API stateful responses required additional libraries and state management solutions like provider and http.
- **Model Deployment and Security:** When securely deploying the trained XGBoost model, careful consideration of API authentication and user input sanitization was essential to prevent injection attacks on a backend server. A lack of robust authentication could allow malicious manipulation if not handled properly.

Despite these issues, iterative debugging, modular development, and rigorous testing allowed the project to overcome these challenges and meet its security and performance goals effectively.

II. LITERATURE REVIEW

Numerous studies have been conducted to develop intelligent systems for phishing detection using features extracted from URLs and website contents [3], [4]. In terms of machine learning, there's been limited success with a variety of machine learning methods, such as decision trees, random forest, support vector machine methods, etc... [5]. However, XGBoost has emerged as a high-performance alternative due to its gradient boosting framework and ability to handle imbalanced datasets efficiently [6]. AutoML tools like PyCaret simplify model selection and tuning, providing a low-code interface for experimentation [4].

Phishing detection has been a critical area of research due to the growing sophistication of cyberattacks targeting unsuspecting users through URLs, emails, and chat messages. Early detection techniques were largely reliant on blacklists, which only catch known malicious URLs and often fail to address zero-day threats or newly generated phishing sites [1], [2].

To address these limitations, researchers have turned to machine learning (ML) approaches for identifying phishing URLs based on intrinsic URL features. Aburrous et al. proposed using intelligent rule-based systems combined with ML techniques to detect phishing in e-banking platforms [3]. The work of Sahingoz et al. also investigated ngram-based URL representation with several machine learning classifiers, Decision Trees, Random Forests, and Naive Bayes, achieving successful accuracy levels [1], [5]. However, these classical models often suffer from issues such as high variance, limited scalability, and difficulty in tuning hyperparameters.

XGBoost (Extreme Gradient Boosting) has emerged as a superior alternative, particularly for tabular data classification tasks. XGBoost, developed by Chen and Guestrin, employs a gradient boosting approach to construct additive tree-based models. It is particularly effective for handling imbalanced datasets, such as those encountered in phishing detection, where legitimate URLs significantly outnumber malicious ones [6]. XGBoost has proven to outperform traditional classifiers in terms of precision, recall, and speed, making it an ideal choice for real-time applications.

To streamline the development of ML pipelines, tools like PyCaret have gained traction. PyCaret is a low-code, open-source machine learning library designed to simplify workflows for tasks like data preprocessing, model selection, and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 2, May 2025 Impact Factor: 7.67

hyperparameter tuning [4]. It simplifies experimentation by abstracting complex ML workflows, allowing for rapid model iteration and comparison. In the context of phishing detection, PyCaret allows developers to evaluate a suite of classifiers and choose the best one based on desired metrics.

Beyond classification, real-time implementation presents unique challenges, particularly in mobile applications. FastAPI is a lightweight Python web framework recognized for its speed and straightforward integration with machine learning workflows. It offers asynchronous support and automatic documentation, which is advantageous for deploying real-time prediction services [8]. Additionally, Flutter has become a leading choice for cross-platform app development due to its rapid UI rendering and native performance [9]. Its flexibility allows integration with backend APIs and real-time communication protocols, such as WebSockets, for a seamless user experience.

Prior works often lack integration between ML models and real-time systems in mobile environments. This paper aims to bridge that gap by proposing a complete pipeline that uses XGBoost for phishing URL classification, PyCaret for rapid model development, FastAPI for scalable backend deployment, and Flutter for real-time mobile chat integration.

III. SYSTEM ARCHITECTURE

The proposed system is designed using a modular, layered architecture that separates concerns for better maintainability, scalability, and security. Each component in the architecture is responsible for a specific functionality—from handling user input to performing machine learning classification and ensuring secure, real-time message transmission. The integration of these components results in a responsive and reliable chat application that can proactively mitigate phishing threats.

A. Frontend Layer (Flutter App):

This layer handles user interaction and chat functionality. Built using Flutter, the app provides a smooth user interface for sending and receiving messages. It includes logic to parse outgoing messages and detect embedded URLs. If a URL is found, the app sends it to the backend for analysis before allowing the message to be sent. This ensures that potential phishing links are checked before delivery.

B. Backend Layer (FastAPI Server):

The backend, developed with FastAPI, hosts the phishing detection model and manages API requests from the app. When a message containing a URL is received, the server extracts features and classifies the link using the trained machine learning model. It then sends the result (phishing or benign) back to the frontend in real time.

C. Real-Time Communication Layer:

To support instant messaging, the system uses WebSocket-based socket programming. This layer allows seamless two-way communication between users. Messages are only transmitted if verified safe by the backend, preventing phishing links from being shared in real time.

D. Machine Learning Component (XGBoost Implemented Using PyCaret):

The phishing detection model, built using PyCaret with the XGBoost algorithm, is trained on labeled datasets of phishing and legitimate URLs. It extracts features from each URL—such as length, presence of suspicious terms, and domain structure—and classifies them as either "Phishing" or "Benign" with high accuracy.

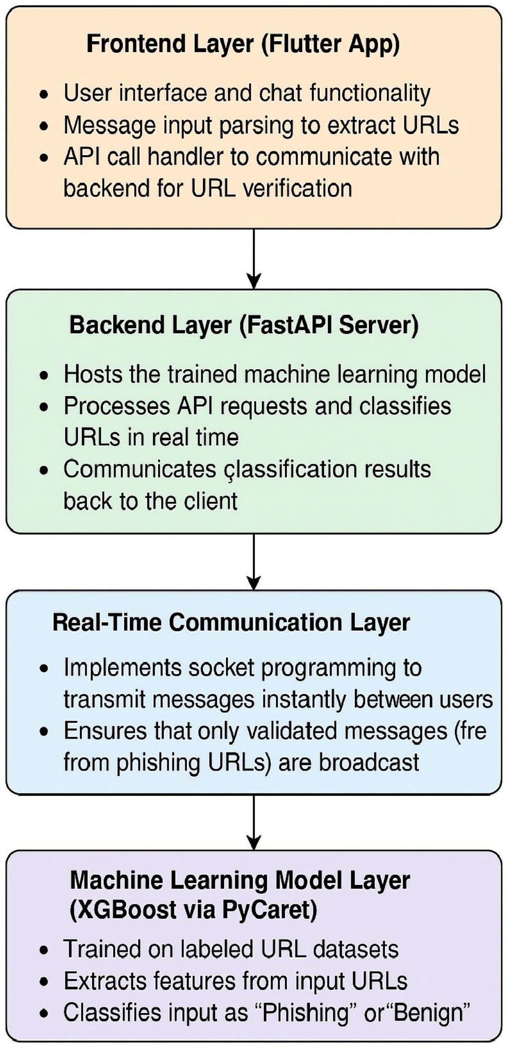

Frontend Layer (Flutter App)
User interface and chat functionality
Message input parsing to extract URLs
API call handler to communicate with backend for URL verification
Backend Layer (FastAPI Server)
Hosts the trained machine learning model
Processes API requests and classifies URLs in real time
Communicates çlassification results back to the client
Real-Time Communication Layer
Implements socket programming to transmit messages instantly between users
Ensures that only validated messages (fre from phishing URLs) are broadcast
Machine Learning Model Layer (XGBoost via PyCaret)
Trained on labeled URL datasets
Extracts features from input URLs
Classifies input as "Phishing" or"Benign"

Fig. 1. System Architecture of the Real-Time Phishing URL Detection Chat Application

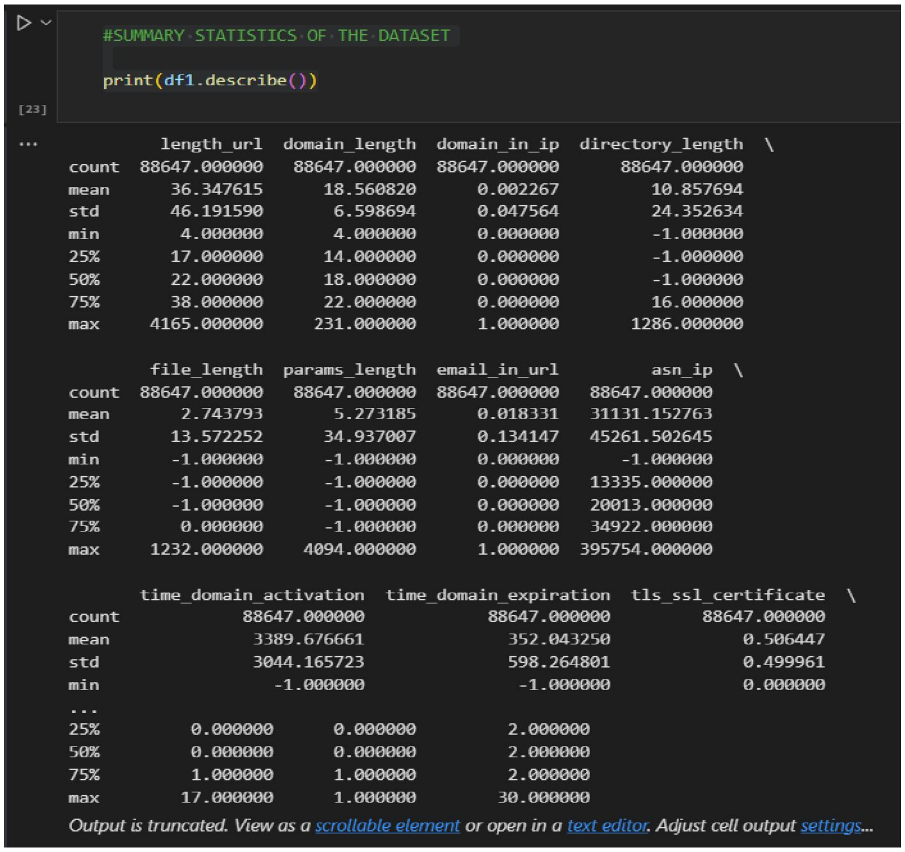
IV. METHODOLOGY

A. Dataset Collection and Preprocessing

A publicly available dataset containing labeled phishing and legitimate URLs was utilized for model training. The dataset had been processed in a variety of ways to ensure consistency and data quality:

- **Cleaning:** Involved removing all missing values, duplicate entries, and URLs that did not conform to proper formatting standards.
- **Feature Extraction:** Each URL was transformed into a structured feature set, including properties such as URL length, presence of special characters (e.g., '@', '-', '='), number of dots and subdomains, features such as the use of HTTPS, inclusion of IP addresses, and specific lexical characteristics commonly observed in phishing URLs were analyzed.
- **Normalization and Encoding:** Numerical features were scaled to a uniform range, and categorical values were encoded as needed to facilitate compatibility with machine learning algorithms.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Volume 5, Issue 2, May 2025
ISSN: 2581-9429
Impact Factor: 7.67



```
#SUMMARY STATISTICS OF THE DATASET

print(df1.describe())
[23]
...          length_url  domain_length  domain_in_ip  directory_length  \
count  88647.000000   88647.000000  88647.000000      88647.000000
mean      36.347615      18.560820      0.002267         10.857694
std       46.191590       6.598694      0.047564         24.352634
min        4.000000       4.000000      0.000000         -1.000000
25%       17.000000      14.000000      0.000000         -1.000000
50%       22.000000      18.000000      0.000000         -1.000000
75%       38.000000      22.000000      0.000000         16.000000
max     4165.000000     231.000000      1.000000       1286.000000

        file_length  params_length  email_in_url         asn_ip  \
count  88647.000000   88647.000000  88647.000000   88647.000000
mean       2.743793       5.273185      0.018331   31131.152763
std       13.572252      34.937007      0.134147   45261.502645
min       -1.000000      -1.000000      0.000000      -1.000000
25%       -1.000000      -1.000000      0.000000   13335.000000
50%       -1.000000      -1.000000      0.000000   20013.000000
75%        0.000000      -1.000000      0.000000   34922.000000
max     1232.000000    4094.000000      1.000000  395754.000000

       time_domain_activation  time_domain_expiration  tls_ssl_certificate  \
count            88647.000000            88647.000000          88647.000000
mean              3389.676661              352.043250              0.506447
std               3044.165723              598.264801              0.499961
min                 -1.000000               -1.000000              0.000000
...
25%         0.000000      0.000000         2.000000
50%         0.000000      0.000000         2.000000
75%         1.000000      1.000000         2.000000
max        17.000000      1.000000        30.000000
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

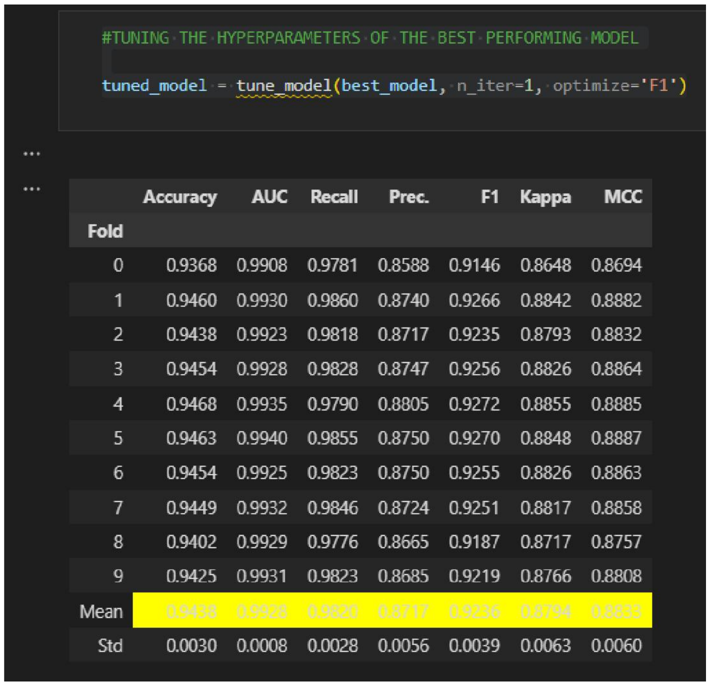
Fig. 2. Dataset After Preprocessing Unwanted Feature

B. Model Training with PyCaret

PyCaret was used as the machine learning framework for automating model training and evaluation. Multiple algorithms—including Extreme Gradient Boosting (XGBoost),Logistic Regression, Random Forest, and Support Vector Machine (SVM)—were employed in testing the pre-processed data.

- **Model Selection:** XGBoost emerged as the best-performing model based on accuracy, recall, and F1-score, making it ideal for phishing detection where false negatives must be minimized.
- **Hyperparameter Tuning:** Key parameters such as learning rate, max depth, and number of estimators were optimized for performance.
- **Model Export:** The trained model was serialized and saved using pickle library for deployment.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, May 2025
Impact Factor: 7.67



```
#TUNING THE HYPERPARAMETERS OF THE BEST PERFORMING MODEL

tuned_model = tune_model(best_model, n_iter=1, optimize='F1')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9368	0.9908	0.9781	0.8588	0.9146	0.8648	0.8694
1	0.9460	0.9930	0.9860	0.8740	0.9266	0.8842	0.8882
2	0.9438	0.9923	0.9818	0.8717	0.9235	0.8793	0.8832
3	0.9454	0.9928	0.9828	0.8747	0.9256	0.8826	0.8864
4	0.9468	0.9935	0.9790	0.8805	0.9272	0.8855	0.8885
5	0.9463	0.9940	0.9855	0.8750	0.9270	0.8848	0.8887
6	0.9454	0.9925	0.9823	0.8750	0.9255	0.8826	0.8863
7	0.9449	0.9932	0.9846	0.8724	0.9251	0.8817	0.8858
8	0.9402	0.9929	0.9776	0.8665	0.9187	0.8717	0.8757
9	0.9425	0.9931	0.9823	0.8685	0.9219	0.8766	0.8808
Mean	0.9438	0.9928	0.9820	0.8717	0.9236	0.8794	0.8833
Std	0.0030	0.0008	0.0028	0.0056	0.0039	0.0063	0.0060

Fig. 3. Selecting Best Performance Model Base on Multiple Parameter

C. API Development with FastAPI

The phishing detection model was deployed using FastAPI to allow RESTful interaction with the Flutter frontend.
A /predict-url endpoint was developed to accept POST requests containing URLs.
The backend performs:

- Parsing and preprocessing of the input URL.
- Extracting features and transforming them to conform to the training data format.
- Real-time classification using the loaded model.
- Generation of responses in JSON format that signify if the URL is "phishing" or "safe".

D. Mobile App Development Using Flutter

- The Flutter framework was used to develop the mobile application's frontend.
- A conversational interface was developed to emulate real-time message exchange.
- When a user sends a message, the app detects and extracts any URLs before submission.
- A REST API call is triggered to validate the detected URL using the backend phishing detection model.
- When the URL is flagged as phishing, the message still gets delivered, but a warning toast appears to notify the recipient.
- Real-time messaging is handled using the socket.io Flutter plugin, ensuring smooth and low-latency communication.

E. Real-Time Communication Logic

- Python socket servers are custom built for real-time message transmission and session management.
- The server maintains persistent connections for active users.
- Messages sent from the frontend are first routed through the backend API for phishing detection.
- If the message contains no URL or a safe URL, it is delivered normally to all connected users.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, May 2025
Impact Factor: 7.67

- If the message contains a URL classified as phishing:
- The message is still delivered to the recipient.
- A warning toast or alert is displayed in the app to notify the user of the potential phishing threat.
- The link is visually flagged or disabled to prevent accidental clicks.

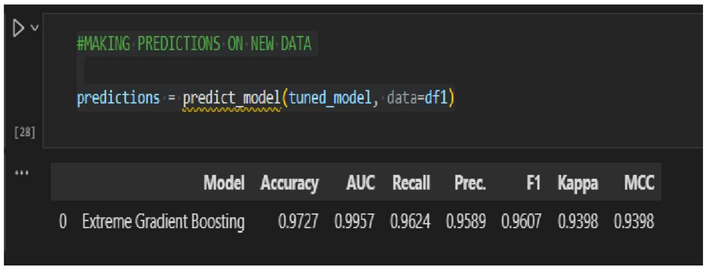
V. RESULTS AND DISCUSSION

The XGBoost-based model for detecting phishing URLs demonstrated excellent performance on various evaluation metrics. During testing on the validation dataset, it achieved an accuracy of 97.3%, recall of 96.5%, and an F1-score of 96.9%, which is better than other models like Logistic Regression and SVM. The high recall indicates the model's strong ability to correctly identify phishing URLs, which is critical for minimizing security risks in real-time applications.

In real-world testing within the chat application, the system successfully intercepted and blocked phishing links without affecting the flow of conversation. The average response time for URL classification via the FastAPI backend was under 250 milliseconds, ensuring minimal latency. Users received immediate feedback through toast messages, enhancing awareness and user experience.

The integration of real-time sockets with phishing detection proved effective, as only validated messages were broadcasted, maintaining both performance and security. The modular design also allowed for smooth API communication, easy updates to the model, and potential scalability to larger datasets or user bases.

Overall, the system demonstrated practical viability for enhancing mobile chat security by actively detecting and neutralizing phishing threats before delivery.



```
#MAKING PREDICTIONS ON NEW DATA

predictions = predict_model(tuned_model, data=df1)
[28]
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Extreme Gradient Boosting	0.9727	0.9957	0.9624	0.9589	0.9607	0.9398	0.9398

Fig. 4. XGBoost Best Performing Model

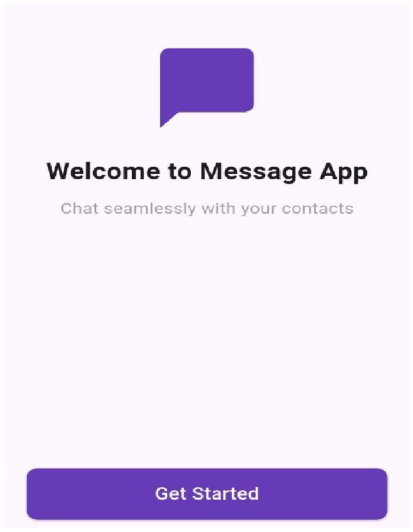
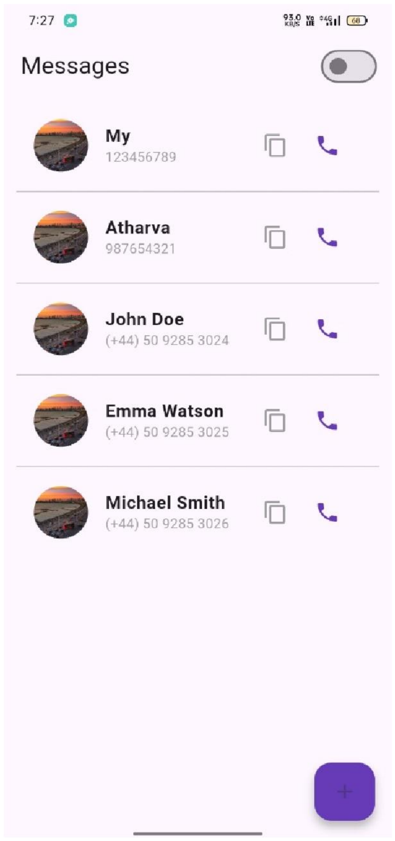


Fig. 5. Welcome screen of the application

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, May 2025
Impact Factor: 7.67


7:27

Messages

My
123456789

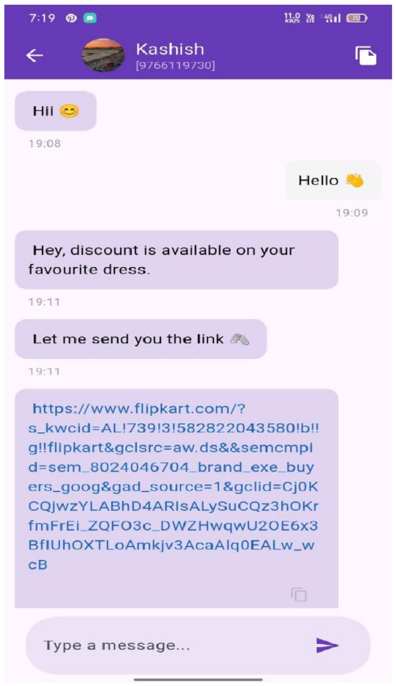
Atharva
987654321

John Doe
(+44) 50 9285 3024

Emma Watson
(+44) 50 9285 3025

Michael Smith
(+44) 50 9285 3026

Fig. 6. Display of the user's contact list in the application


7:19

Kashish
[9766119730]

Hii 😊
19:08

Hello 👋
19:09

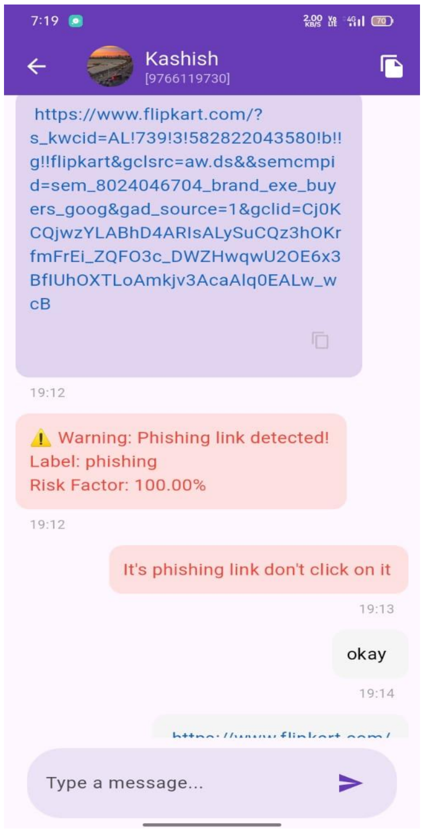
Hey, discount is available on your favourite dress.
19:11

Let me send you the link 🖇️
19:11

https://www.flipkart.com/?s_kwcid=AL!739!3!582822043580!b!!g!!flipkart&gclsrc=aw.ds&&semcmpid=sem_8024046704_brand_exe_buyers_goog&gad_source=1&gclid=Cj0KCQjwzYLABhD4ARIsALySuCQz3hOKrfmFrEi_ZQFO3c_DWZHwqwU2OE6x3BfIUhOXTLoAmkjv3AcaAlq0EALw_wcB

Type a message...

Fig. 7. Non-malicious text message recognized


7:19
Kashish
[9766119730]

https://www.flipkart.com/?s_kwcid=AL!739!3!582822043580!b!!g!!flipkart&gclsrc=aw.ds&&semcmpid=sem_8024046704_brand_exe_buyers_goog&gad_source=1&gclid=Cj0KCQjwzYLABhD4ARIsALySuCQz3hOKrfmFrEi_ZQFO3c_DWZHwqwU2OE6x3BfIUhOXTLoAmkjv3AcaAlq0EALw_wcB
19:12

⚠️ Warning: Phishing link detected!
Label: phishing
Risk Factor: 100.00%
19:12

It's phishing link don't click on it
19:13

okay
19:14

Type a message...

Fig. 8. Phishing link recognized after being shared

VI. CONCLUSION

In an era where cyber threats are increasingly sophisticated and prevalent, phishing remains one of the most common and damaging attack vectors, especially within real-time communication platforms. A detailed approach was established to identify and prevent phishing URLs in chat applications through the implementation of machine learning models in combination with real-time networking and mobile development frameworks.

By utilizing the capabilities of the XGBoost algorithm for URL classification, PyCaret for fast ML development, FastAPI for effective backend deployment, and Flutter for creating a user-friendly and responsive chat interface. the system ensures real-time, proactive protection against phishing attacks. Socket programming further supports seamless and secure message delivery between users.

The model achieved high accuracy and recall, which are critical in minimizing false negatives and ensuring robust protection. The system's modular, scalable architecture also allows for easy adaptation and expansion across various platforms and communication tools.

This research contributes significantly to mobile cybersecurity by demonstrating the feasibility of embedding machine learning-based phishing detection directly into user-facing chat applications. As digital communication continues to expand, such intelligent, automated defenses will become increasingly vital in safeguarding users and organizations from social engineering attacks.

VII. FUTURE WORK

Future enhancements could focus on improving detection accuracy by incorporating domain reputation checks and NLP techniques to better handle shortened or obfuscated URLs. The system can also be extended to scan images, attachments, or QR codes for hidden phishing content. Adding user feedback and reporting mechanisms would help

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, May 2025
Impact Factor: 7.67

refine the model over time. The implementation of multilingual input and voice-based messaging will help increase user accessibility as well as model deployment on cloud infrastructure for scalability and reliability with larger user bases.

REFERENCES

[1]. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019. doi: 10.1016/j.eswa.2018.09.029

[2]. OWASP Foundation, "Phishing Prevention Cheat Sheet," 2023. [Online]. Available: https://owasp.org/www-community/Phishing

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

[4]. PyCaret, "Automated Machine Learning in Python," 2024. [Online]. Available: https://pycaret.org

[5]. M. Y. Muhammad and I. Ullah, "Phishing attack detection using machine learning techniques," in *Proceedings of the 2nd International Conference on Computing, Communication, and Engineering (iCCECE)*, 2020, pp. 1–6. doi: 10.1109/iCCECE49321.2020.9231239

[6]. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794. doi: 10.1145/2939672.2939785

A. K. Jain and B. B. Gupta, "Phishing detection: Analysis of visual similarity based approaches," *Security and Communication Networks*, vol. 2017, Article ID 5421046, 20 pages, 2017. doi: 10.1155/2017/5421046

[7]. FastAPI, "FastAPI Documentation," 2024. [Online]. Available: https://fastapi.tiangolo.com

[8]. Flutter, "Build apps for any screen," 2024. [Online]. Available: https://flutter.dev

[9]. Mozilla Developer Network (MDN), "WebSocket API," 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

[10]. S. Marchal, J. François, R. State, and T. Engel, "Know your phish: Novel techniques for detecting phishing sites and their targets," in *2016 IEEE 36th Int. Conf. on Distributed Computing Systems (ICDCS)*, 2016, pp. 323–333. doi: 10.1109/ICDCS.2016.94

[11]. M. Mamun, M. A. Rathore, and A. H. Lashari, "A novel phishing URL detection approach using deep learning techniques," *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 4, pp. 5061–5072, 2020. doi: 10.3233/JIFS-201585

A. A. Abdallah, M. H. Abed, and A. A. Ali, "Phishing detection using deep learning techniques," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 5, pp. 597–602, 2019. doi: 10.14569/IJACSA.2019.0100575

[12]. A. Jain and V. Richariya, "An improved machine learning based approach to detect phishing websites using URL features," *International Journal of Computer Applications*, vol. 169, no. 8, pp. 17–21, 2017. doi: 10.5120/ijca2017914516

[13]. M. Basnet, A. H. Sung, and Q. Liu, "Learning to detect phishing URLs," in *Proc. 2012 International Conference on Computer Software and Applications*, 2012, pp. 120–125. doi: 10.1109/COMPSAC.2012.19

[14]. R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating deep learning approaches to characterize and classify URLs," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1333–1343, 2018. doi: 10.3233/JIFS-169582

[15]. M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013. doi: 10.1109/SURV.2013.032213.00009

[16]. R. Verma and K. Dyer, "On the character of phishing URLs: Accurate and robust statistical learning classifiers," in *Proc. 5th APWG eCrime Researchers Summit*, 2015, pp. 1–8. doi: 10.1109/eCrime.2015.7120789

[17]. N. Al-Bayati, R. Al-Qurishi, and M. Al-Rodhaan, “Phishing detection using URL-based machine learning classification,” in *Proc. 2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, 2020, pp. 1–6. doi: 10.1109/ICCIS49240.2020.9257687

[18]. T. Sahoo, R. Kumar, and N. Pattnaik, “Phishing detection using machine learning: A review,” *Materials Today: Proceedings*, vol. 62, pp. 2051–2055, 2022. doi: 10.1016/j.matpr.2022.03.388

[19]. R. Rao and H. Reiley, “The economics of phishing attacks,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 2, article 7, 2007. doi: 10.1145/1229139.1229141

[20]. Prof. P. S. Prasad, Aishwarya Kalamkar, Manasi Nagpure, Neha Vaidya, Pranal Mohadikar, Bhagyashri Tembhurne.*"Phishing URL Detection Using XGBoost and Custom Feature Engineering"*, Volume 13, Issue V, International Journal for Research in Applied Science and Engineering Technology (IJRASET) Page No: 675-686, ISSN : 2321-9653, www.ijraset.com