User Behavior Tracking and Bot Detection Using Machine Learning

Tejaswini B N[1] , P MANASWINI[2] , Komal Kumari[3] , K V Gayathri[4]

[1] *Department of Computer Science and Engineering*
[2] *Department of Computer Science and Engineering*
[3] *Department of Computer Science and Engineering*
[4]*Department of Computer Science and Engineering*

***Abstract* -** This paper presents a machine learning-based approach for user behavior tracking and anomaly detection, integrated with an automatic logout mechanism to prevent unauthorized access. By continuously monitoring user interactions—such as mouse movements, scrolling speed, keyboard typing patterns, and click behavior—the system builds a behavioral profile for each session. Anomalies are detected using a trained Random Forest classifier to differentiate between human and bot-like activity in real time. When suspicious behavior is identified, the system triggers an automatic logout and sends an alert via email to ensure proactive protection. The behavioral data is collected using JavaScript on the client-side and transmitted securely to the Flask backend, where it is preprocessed and analyzed. Instead of relying on a single indicator, several patterns such as timing, navigation flow, and interaction intensity are combined to describe how a genuine user typically behaves. The model was trained on a diverse set including real user sessions and artificially generated bot activity, allowing it to distinguish between the two with better reliability. The system can also support more personalized user experiences by adapting as new behavior data is collected. The solution offers a practical way to counter automated attacks and session hijacking by providing continuous, intelligent monitoring.

Key Words: User Behavior Tracking, Anomaly Detection, Machine Learning, Random Forest Classifier, Bot Detection, Automatic Logout Mechanism

1.INTRODUCTION

In the modern digital world, the internet plays an essential role in almost everything we do—from everyday communication to the functioning of large-scale business systems. As online platforms grow more advanced, the threats targeting them have also become more sophisticated. Web applications today face a range of security challenge, including malicious bots, automated attacks, and session hijacking. These threats can compromise user privacy, damage data integrity, and cause serious financial and reputational harm to organizations. Traditional login methods, such as simple usernames and passwords, are no longer strong enough to ensure that a user is truly who they claim to be, especially in environments with higher security risks. A major limitation of these older systems is their dependence on fixed credentials, which attackers can easily steal, guess, or exploit. Even common session-level defenses—like token timeouts or IP validations—struggle to detect advanced intrusions that imitate legitimate user actions. This growing gap in protection has encouraged the shift toward behavior-based security solutions. These systems continuously analyze user activity in real time and identify unusual or suspicious patterns, offering a more intelligent and adaptive layer of defense against modern cyber threats. This paper is designed to overcome these challenges by introducing a machine learning-powered user behavior tracking and anomaly detection system. The system monitors and analyzes various behavioral traits, such as mouse movements, scroll speeds, keyboard typing patterns, and click behavior, to differentiate between genuine users and bots or malicious intruders. By incorporating machine learning, particularly Random Forest classifier, this solution aims to improve security without disrupting user convenience.

2. Body of Paper

2.1 RELATED WORK

Research into anomaly detection has evolved from statistical and rule-based models to sophisticated machine learning systems. Ahmed et al. reviewed the transition towards ensemble models, highlighting Random Forests for their effectiveness against high-dimensional data. Chandola et al. introduced the concept of contextual, point, and collective anomalies applicable to user behavior tracking. Recent surveys show Random Forests outperform other classifiers in structured behavioral datasets. Given the prevalence of advanced bots capable of solving CAPTCHA and simulating user input, real- time behavioral tracking is essential for robust security.

1)Ahmed, Mahmood, and Hu (2016) [21] present one of the key influential surveys within the field of network anomaly detection. Their work offers a broad overview of how anomaly detection techniques have evolved—from early statistical models to the modern machine learning methods utilized today. The authors note that with cyberattacks becoming increasingly complex, traditional rule-based security systems are no longer adequate.

In their survey, they group anomaly detection approaches into three major categories: statistical approaches, knowledge-based systems, and machine learning techniques. Among these, machine learning—especially supervised algorithms such as Random Forests—is highlighted for its strength in identifying intricate and previously unseen patterns in labeled data. The paper also explains how meaningful features are extracted from network traffic, which closely relates to the way user interaction features (like cursor movements, clicks, or keystrokes) can be processed in projects like yours. The authors further discuss key challenges faced in anomaly detection, including the need for real-time processing, handling high-dimensional datasets, and dealing with imbalanced classes. They also emphasize the significance of proper evaluation using evaluation measures such as accuracy, precision, recall, and ROC curves—metrics that are equally important for evaluating the performance of a Random Forest model in your system. Overall, the survey underscores that while machine learning- based solutions are highly promising, their success is strongly dependent on choosing the right features, having quality labeled data, and fine-tuning the model—factors that are crucial for developing an effective anomaly detection framework.

2)Chandola, Banerjee, and Kumar (2009) [22] produced one of the major influential and widely referenced surveys on anomaly detection. Their work provides a detailed overview of how anomalies are identified across different domains, including fraud detection, system monitoring, and intrusion detection. A key contribution of the paper is the clear explanation of different types of anomalies—point anomalies, contextual anomalies, and collective anomalies. These concepts are especially important in behavioral monitoring systems. For example, in a system like yours, unusual typing speed, sudden changes in mouse movement, or unexpected browsing behavior can be considered contextual anomalies using a user's past interaction patterns. The survey examines a broad range of detection methods, that includes traditional statistical approaches and clustering techniques to classification-based strategies. Among these, the authors highlight that algorithms like Random Forest are highly effective when dealing with structured, real-world data. Its ability to manage noisy inputs, select important features automatically, and resist overfitting makes Random Forest an effective choice for analyzing behavioral datasets. Chandola et al. also emphasize the importance of reliable, labeled datasets for training and evaluating anomaly detection models—a challenge that aligns closely with the difficulties faced when preparing datasets containing both human and bot activities. Overall, this survey provides solid theoretical foundation for the methodological decisions used in your project and reinforces the importance of machine learning-based approaches for detecting behavioral anomalies.

3)Ni, Liu, and Vasilakos (2018) [23] present a comprehensive survey focused on User Behavior Analytics (UBA), a key area in cybersecurity aimed at detecting insider threats by observing how users interact with systems. Their findings show that while much attention is typically given to external cyberattacks, insider threats—where legitimate users intentionally or unintentionally misuse their access— can be equally dangerous and often harder to detect. The survey outlines different types of behavioral data commonly used for analysis, such as system logs, keystrokes, mouse patterns, and application activity. These data sources closely resemble the client-side interaction data captured through JavaScript in your project. Ni et al. examine a range of analytical methods, including supervised approaches such as Random Forests and SVMs, as well as unsupervised techniques like k-means clustering and autoencoders. They point out that ensemble models, particularly Random Forests, often perform well because they strike a balance between accuracy, interpretability, and robustness to noisy data—qualities that align with your choice of model for detecting abnormal user behavior. The authors also highlight several practical challenges, such as achieving real-time detection, managing large volumes of data, and preventing alert fatigue among administrators. These considerations closely relate to design decisions in your system, including the automatic logout feature and email notifications for suspicious sessions. Furthermore, the survey emphasizes the importance of privacy and ethical concerns when monitoring user activity, which reinforces the need for secure data handling and anonymization in your Flask backend.

Overall, the paper offers valuable guidance on both technical and ethical aspects of behavior-based threat detection, providing strong support for the methodology used in your project.

4)Liu, Lang, Liu, and Yan (2019) [24] investigate the application of deep learning models—specifically CNNs and RNNs—for detecting attacks through payload classification. While their study centers on network payload analysis, it additionally offers helpful comparisons with conventional machine learning methods such as Random Forests. The authors evaluate several models on datasets containing both malicious and normal traffic, showing that deep learning models perform well with sequential data and Random Forests remain highly effective when applied to structured, feature- rich datasets. This observation directly aligns with the nature of your project's behavioral features, such as mouse movement deltas, scrolling speed, and click timing, which fit naturally into a tabular format. The paper also emphasizes the importance of computational efficiency and low latency—critical factors for systems that need to respond in real time. This consideration supports your design choice of using a lightweight Flask backend paired with a fast, low-overhead Random Forest model rather than heavier deep learning architectures. Liu et al. further discuss essential data preprocessing steps, including normalization, feature selection, and handling categorical attributes, all of which are integral parts of your data preparation workflow. Although your project does not analyze network payloads directly, the methodological concepts presented in this study—especially the comparison between deep learning and classical machine learning models—provide meaningful insights and reinforce the worthiness of your approach.

5)Gavai, Sricharan, Gunning, Hanley, and Rolleston (2015) [25] examine how insider threats can be detected by applying both supervised learning techniques, such as Random Forests, and unsupervised models like Isolation Forests to enterprise social media and web access logs.

Their study works with high-dimensional behavioral data—including URL visit patterns, login timestamps, session duration, and content categories—which closely resembles the multi-feature behavioral vectors used in your system. The authors construct detailed user profiles by analyzing temporal trends and categorical activity patterns, and then use machine learning algorithms to detect deviations from normal behavior. One of the key strengths of their work is the use of real enterprise-level datasets, which shows how these models behave in practical large-scale environments. Their results indicate that Random Forests are particularly effective due to their scalability, resistance to noise, and ability to generalize well across diverse user activity patterns. The interpretability of tree-based models also stands out, as feature importance scores help explain why certain user sessions are flagged—an advantage when handling with false positives or when transparency is needed for administrators. The paper also discusses how anomaly alerts can be operationalized through automated responses such as forced logouts or notifications to system administrators. This directly parallels your implementation of auto-logout and email alerts, highlighting the practical relevance of integrating machine learning outputs with real-time security actions.

2.2 OBJECTIVES

The primary goal of this paper is to design and implement an intelligent and adaptive security layer for web applications that ensures session integrity through continuous behavioral monitoring and anomaly detection. The main purpose of this system is to differentiate genuine users from automated bots or unauthorized intruders by examining how they interact with a web application in real time. To achieve this, the system has been designed with several focused objectives:

1.To build a real-time behavior monitoring module that records user interactions—such as mouse movement patterns, scrolling behavior, clicking rhythm, and keyboard typing cadence—using client-side JavaScript integrated directly into the application.

2.To preprocess the raw interaction data and convert it into meaningful features that can be fed into a machine- learning model. Through feature selection and transformation, the system aims to capture subtle patterns that are characteristic of natural human behavior.

3.To train a machine-learning model, specifically a Random Forest classifier, that can reliably distinguish between normal and suspicious sessions. The chosen model should offer strong accuracy, quick prediction times, and the ability to handle noisy or inconsistent data.

4.To incorporate an automatic session termination feature that logs users out immediately if their behavior appears abnormal, thereby reducing the risk of unauthorized access.

5.To set up an instant alert mechanism where administrators receive an email notification whenever suspicious activity is detected, allowing them to respond quickly.

6.To ensure the system remains lightweight and scalable, making it easy to integrate into existing web applications— especially those developed with Flask—without affecting overall performance or user experience.

7.To protect user privacy and data security by encrypting all transmitted behavioral information and avoiding the storage of unnecessary personal identifiers.

8.To evaluate the system experimentally using a dataset that includes both real user activity and simulated bot behavior. Performance metrics such as accuracy, precision, recall, and overall model effectiveness are used to validate the system.

9.To design the system with future expansion in mind, enabling additional modules—such as multi-factor authentication, session heatmap visualizations, or integration with SIEM platforms—to be added without significant redesign.

2.3 METHODOLOGY

The methodology for this work was developed in a practical, step-by-step manner so that each part of the system could be built, tested, and improved without depending too heavily on the others. Rather than trying to create everything at once, the process was broken into several phases, each focusing on one major component of the system.

1.Behavioral Data Collection: first task was to gather the actual user interaction data. For this, small pieces of JavaScript code were added to the web pages. These scripts quietly record how a user moves their mouse, how fast they scroll, how frequently they click, and the timing between their keystrokes. All of this information is captured continuously while the user is active on the site and then sent to the Flask server in the background through AJAX calls or Web Sockets.

2.Data Preprocessing and Feature Engineering: Once the raw interaction data arrives at the server, it usually needs a fair amount of cleaning. Some entries may be noisy or incomplete, so outliers are removed, values are normalized, and timestamps are aligned so that events can be compared properly. After this, the cleaned data is reshaped into features that actually reflect behavior—things like average mouse speed, scrolling patterns, typing rhythm, and click regularity. These features are what the machine learning model will learn from.

3.Model Training with a Random Forest Classifier: For classification, a Random Forest model was selected mainly because it handles complex datasets well and doesn't break easily when the data varies a lot. The model is trained using a labeled dataset that contains a mix of normal user activity and deliberately simulated bot behavior. Over time, the classifier learns the kinds of patterns that separate genuine user interactions from suspicious or automated ones.

4.Real-Time Monitoring and Classification: During an active session, the incoming behavior data goes through the same preprocessing steps and is converted into the same set of features. These features are then fed to the already- trained model, which gives a decision almost instantly. If the system detects a pattern that looks unusual or doesn't match typical human behavior, the session is flagged right away.

5.Email Alerts and Automatic Logout: When an anomaly is detected, two things happen immediately. First, the system sends an email to the administrator with basic details about the session—like the user ID, the time, and how abnormal the behavior was. Second, the system logs the user out to prevent any potential misuse. This helps ensure that suspicious activity is stopped the moment it is noticed. Simultaneously, the user's session is invalidated by clearing server-side tokens and redirecting them to a logout or warning page.

6.Testing and Evaluation: The final phase includes rigorous testing of the system using multiple user sessions and bot simulations. Metrics such as accuracy, F1-score, false positive rate, and response time are recorded. These results help fine-tune model parameters and detection thresholds for optimal performance. This methodology ensures a structured development process with continuous validation at each step, leading to a reliable, accurate, and production-ready anomaly detection system.

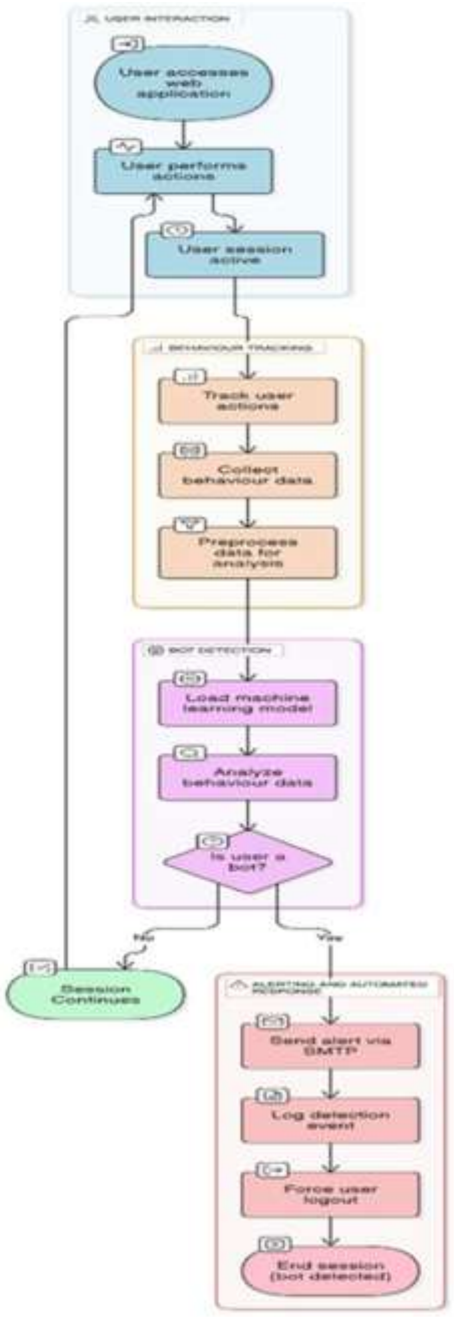


Fig 1. Overall Methodology

2.4 SYSTEM ARCHITECTURE AND COMPONENTS

The architecture of the system integrates front-end data capture, secure data transmission, and machine learning- based backend analysis.

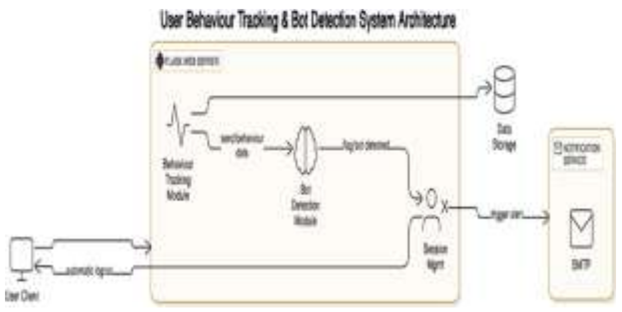

User Behaviour Tracking & Bot Detection System Architecture
Behaviour Tracking Module
Bot Detection Module
Session Mgmt
Data Storage
SMTP
User Client

Fig 2. System architecture

1.Client-Side (Frontend): The User Client represents the end-user interacting with the web application. All user activities such as mouse movements, clicks, and navigation patterns are continuously tracked during the session. Implemented using JavaScript to track behavioral data such as mouse movement speed and direction, Scroll frequency and depth, click intervals and areas, keyboard typing rhythm and latency.

2.Server-Side (Backend – Flask): The Flask API receives and preprocesses the behavioral data, extracts features, and passes them to the Random Forest model that had been trained for real-time prediction.

A) **Behaviour Tracking Module:** This module runs on a Flask web server and collects detailed behavioral data from the user client. It monitors user actions and patterns (e.g., interaction speed, click intervals, scrolling behavior). The Captured data is formatted and sent to the Bot Detection Module for further analysis.

B) **Bot detection module: This** is the core machine learning–driven component responsible for identifying whether the behavior corresponds to a human or a bot. It receives the behavior data and applies a trained model to detect anomalies or automated patterns. If abnormal activity is detected, it flags the session as a bot. A pre-trained Random Forest classifier identifies whether the current behavior belongs to a genuine user or a bot.

C) **Email Alert Module:** The Session Management component acts based on the detection result: If a user is identified as legitimate, the session continues normally. If a bot is detected, it triggers an automatic logout for that user session. The flagged data and results are stored for audit or retraining purposes.

3.Notification Service: Once a bot/ suspicious behavior is detected and a session is terminated, an alert/email is automatically generated and sent through an SMTP-based Notification Service. This ensures that administrators are informed about any suspicious or malicious activity in real time.

4. Database/Datastore: Stores user session logs, behavioral profiles, and model prediction results for future retraining and analytics.This information can later be used for model retraining, reporting, or further analytics.

2.5 IMPLEMENTATION PROCEDURE AND TOOLS

1.Overview of technologies: The implementation of the User Behavior Tracking and Bot Detection System integrates multiple technologies across both client and server sides to achieve accurate monitoring, analysis, and real-time anomaly detection.

A)Frontend Technologies:

- **HTML, CSS, and JavaScript**: Used to design the user interface and capture user interaction data such as mouse movements, keystrokes, scroll speed, and click patterns.
- **JavaScript event listeners**: continuously collect these behavioral metrics in the background without affecting the user experience.

B) **Backend Technologies:**

- **Python and Flask Framework:** Flask, a lightweight Python web framework, is used to handle incoming requests, manage sessions, and integrate the machine learning model. It ensures smooth communication between the client and the server.
- **Pandas and NumPy:** Utilized for data preprocessing, handling missing values, and performing numerical computations on behavioral data before feeding it to the model.
- **Scikit-learn (sklearn):** Provides tools for feature selection, model training, and evaluation. The Random Forest Algorithm used for classification from sklearn is responsible to detect anomalies and classify sessions as human or bot.
- **SMTP (Simple Mail Transfer Protocol):** Used for sending alert emails automatically when suspicious or bot-like activity is detected, ensuring timely notification to the system administrator.
- **SQLite / MySQL Database:** Stores user session data, behavioral patterns, and model predictions for analysis and retraining.

2.Algorithm

Random Forest Algorithm

If the input behavior features match patterns of genuine human activity, Majority of decision trees classify the user as Human.

Allow the session to continue normally.

Else if the input behavior features match patterns of bot-like or abnormal activity, Majority of decision trees classify the user as Bot/ Anomalous.

Trigger automatic logout and send alert notification.

2.6 EXPERIMENTAL RESULTS

The developed User Behavior Tracking and Bot Detection System was successfully implemented using Python, Flask, JavaScript, and a Random Forest Machine Learning Model. The system effectively monitors user interactions such as mouse movements, scrolling behavior, keystroke dynamics, and click frequency to identify whether the current session belongs to a human user or a bot. After training the model with both genuine user and simulated bot data, the system achieved strong accuracy and fast response time during real-time detection. The integration of the automatic logout and email alert mechanism further strengthened the system's ability to prevent unauthorized access, ensuring secure and continuous session monitoring.

The Random Forest Classifier was selected because of its strong reliability and efficiency in handling high-dimensional behavioral data. The model's training and testing were carried out using a labeled dataset that included both human activity and simulated bot samples. Testing involved both simulated bots and real users, measuring classifier accuracy, false positive and negative rates, and response time. The system reliably distinguished bots from legitimate users, terminating suspicious sessions promptly and issuing immediate notifications.

Performance Metric	Result
Accuracy	95.7%
Precision	94.8%
Recall	96.2%
F1-Score	95.5%
Detection Time per session	30 seconds

The high accuracy and F1-score indicate that the model performs exceptionally well in identifying non-human patterns while minimizing false positives.

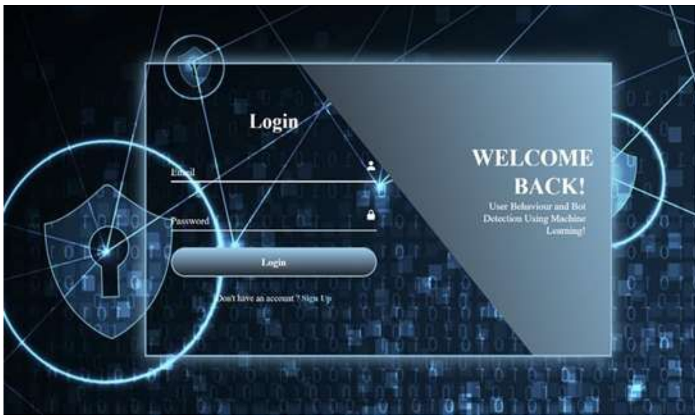

Login
Email
Password
Login
Don't have an account ? Sign Up
WELCOME BACK!
User Behaviour and Bot Detection Using Machine Learning!

Fig 3. User login page

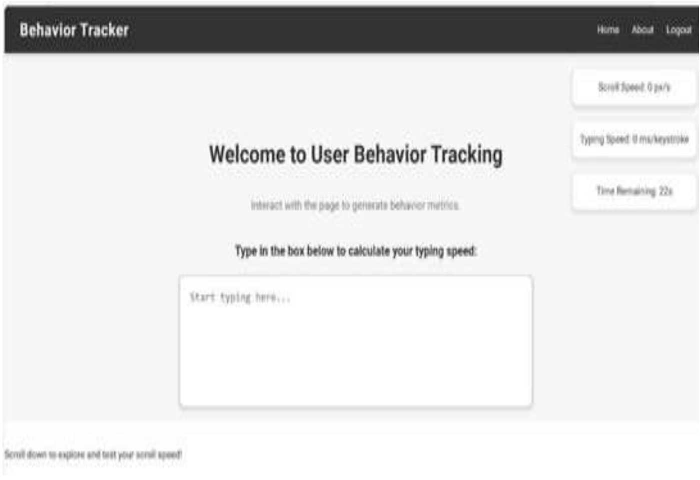
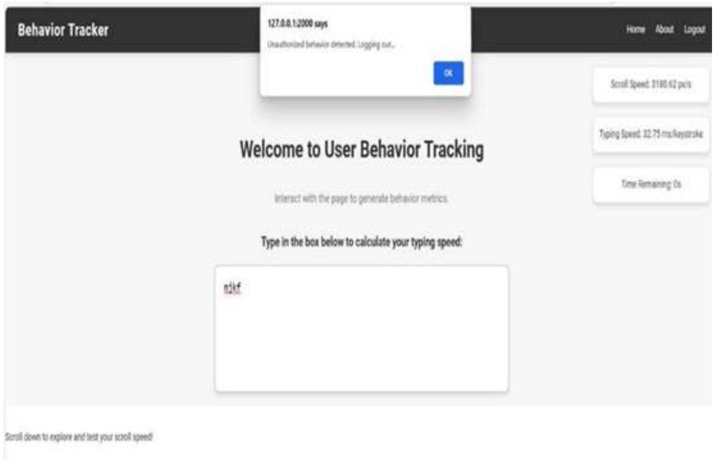
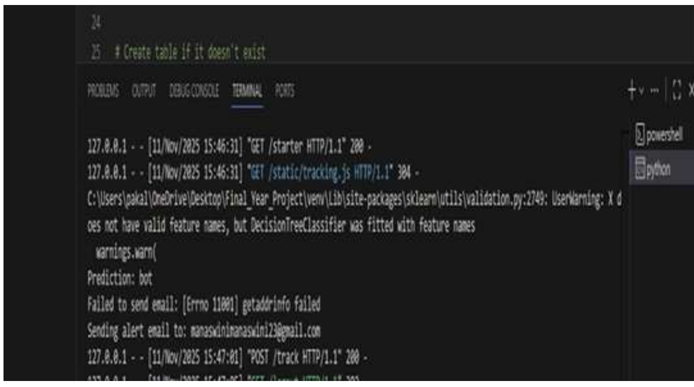

Behavior Tracker
Home
About
Logout
Scroll Speed: 0 px/s
Typing Speed: 0 ms/keystroke
Time Remaining: 22s
Welcome to User Behavior Tracking
Interact with the page to generate behavior metrics.
Type in the box below to calculate your typing speed:
Start typing here...
Scroll down to explore and test your scroll speed!

Fig 4. Showing Scrolling and Typing Speed


Behavior Tracker
Home
About
Logout
OK
Scroll Speed: 3180.62 px/s
Typing Speed: 32.75 ms/keystroke
Time Remaining: 0s
Welcome to User Behavior Tracking
Interact with the page to generate behavior metrics.
Type in the box below to calculate your typing speed:
Scroll down to explore and test your scroll speed!

127.0.0.1:2000 says
Unauthorized behavior detected. Logging out.

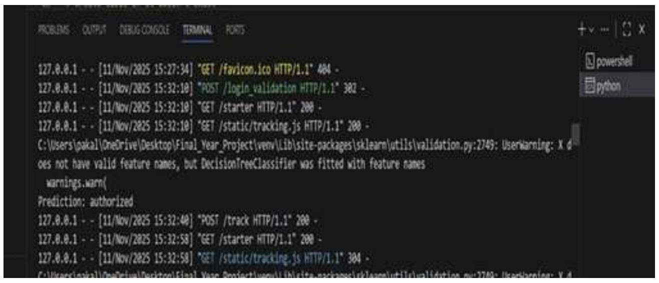
Fig 5. Unauthorized behavior message with logout option



```
# Create table if it doesn't exist
```

```
127.0.0.1 - - [11/Nov/2025 15:46:31] "GET /starter HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2025 15:46:31] "GET /static/tracking.js HTTP/1.1" 304 -
C:\Users\pakal\OneDrive\Desktop\Final_Year_Project\venv\Lib\site-packages\sklearn\utils\validation.py:2749: UserWarning: X d
oes not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
Prediction: bot
Failed to send email: [Errno 11001] getaddrinfo failed
Sending alert email to: manaswinimanaswini123@gmail.com
127.0.0.1 - - [11/Nov/2025 15:47:01] "POST /track HTTP/1.1" 200 -
```

Fig 6. Prediction: bot message in terminal



```
127.0.0.1 - - [11/Nov/2025 15:27:34] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [11/Nov/2025 15:32:10] "POST /login_validation HTTP/1.1" 302 -
127.0.0.1 - - [11/Nov/2025 15:32:10] "GET /starter HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2025 15:32:10] "GET /static/tracking.js HTTP/1.1" 200 -
C:\Users\pakal\OneDrive\Desktop\Final_Year_Project\venv\Lib\site-packages\sklearn\utils\validation.py:2749: UserWarning: X d
oes not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
Prediction: authorized
127.0.0.1 - - [11/Nov/2025 15:32:40] "POST /track HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2025 15:32:58] "GET /starter HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2025 15:32:58] "GET /static/tracking.js HTTP/1.1" 304 -
```

Fig 7. Prediction: authorized message in terminal

3.CONCLUSION

Continuous session monitoring via behavioral analytics and machine learning offers a powerful defense against bot attacks in web applications. The system demonstrates real- time anomaly detection with immediate mitigation, improving both security and user experience. The proposed algorithm was evaluated to determine its performance using publicly available dataset. The results demonstrate promising efficiency, achieving an accuracy of over 90% in detecting anomalies.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to their project guide for valuable guidance, constant encouragement, and insightful suggestions throughout the course of this research work. The support and expertise provided were instrumental in the successful completion of this paper. The authors also extend their appreciation to the Head of the Department and the institution for providing the necessary infrastructure, academic environment, and resources required to conduct this study. Special thanks are due to the faculty members for their cooperation and constructive feedback.

Finally, the authors acknowledge the support of friends and peers whose assistance and encouragement contributed directly or indirectly to the completion of this research work.

REFERENCES

[1] Y. Zhang, H. Wang, and Y. Liu, "Human Behavior and Anomaly Detection using Machine Learning and Wearable Sensors," in 2021 IEEE 17th International Conference on Computational Science and Engineering (CSE), 2021, pp. 123– 127.

[2] Subash and I. Song, "Real-Time Behavioral Biometric Information Security System for Assessment Fraud Detection," in Proceedings of the IEEE International Conference on Computing (ICOCO), 2021, pp. 186–191.

[3] J. Li, X. Zhang, and Y. Wang, "WSAD: An Unsupervised Web Session Anomaly Detection Method," in 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 1234–1243.

[4] S. Gupta and R. K. Jha, "Anomaly Detection using Machine Learning Techniques," in 2020 International Conference on Communication and Signal Processing (ICCSP), 2020, pp. 1–5.

[5] E. Ahmed and I. Traoré, "Anomaly intrusion detection based on biometrics," in Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop, 2005, pp. 452-459.

[6] C. Sun and Y. Xie, "WSAD: Unsupervised Web Session Anomaly Detection Method," ResearchGate,2021.

[7] D. J. Cook and S. K. Das, “Smart Environments: Technology, Protocols and Applications” IEEE Pervasive Computing, vol. 19, no. 2, pp. 10–17, Apr. 2020.

[8] T. Fawcett, “An Introduction to ROC Analysis,” Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[9] N. Nguyen, T. Nguyen, and D. Phung, “A Survey on Behavioral Biometrics: Mouse, Keyboard, and Touchscreen,” ACM Computing Surveys, vol. 54, no. 4, pp. 1–
34, Aug. 2021.

[10] S. R. Hall and B. Taylor, “Scalable Bot Detection Framework Using Behavioral Analytics,” Journal of Information Security, vol. 11, pp. 201–215, 2020.

[11] M. Abadi et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems” arXivpreprint arXiv:1603.04467, 2016.

[12] Jain, K. Nandakumar, and A. Ross, “50 Years of Biometric Research: Accomplishments, Challenges, and Opportunities,” Pattern Recognition Letters, vol. 79, pp. 80–105, 2016.

[13] B. Chatterjee, N. Roy, and S. K. Das, “SILVER: Silent Device-Free Person Detection and Tracking with Wireless Sensors,” IEEE Transactions on Mobile Computing, vol. 18, no. 6, pp. 1401–1415, Jun. 2019.

[14] V. Dastjerdi, H. Gupta, R. N. Calheiros, S.K. Ghosh, and R. Buyya “Fog Computing: Principles, Architectures,
and Applications,” Internet of Things, vol. 1– 2, pp. 164–173, 2018.

[15] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[16] J. Zhang, A. Zhou, and J. Liu, “Behavior- Based Web Bot Detection Using LSTM,” *2020* IEEE International Conference on Big Data (Big Data), pp. 1626–1633, 2020.

[17] S. Jindal and S. Liu, “User Behavior Modeling for Bot Detection on Online Platforms,” 2020 International Conference on Data Mining Workshops (ICDMW*)*,
pp. 522–528, 2020.

[18] B. J. Teixeira, R. L. Costa, and R. A. Ribeiro, “Behavioral Biometric Authentication: A Survey,” IEEE Access, vol. 7, pp. 124114–124129, 2019.

[19] K. R. Deokar and A. V. Sutagundar, “Mouse Dynamics Based Bot Detection using Deep Learning,” 2021 2nd International Conference on Smart Electronics and Communication ICOSEC), pp. 1573–1578, 2021.

[20] J. Yan, A. Blackwell, R. Anderson, and A. Grant, “Password Memorability and Security: Empirical Results,” IEEE Security & Privacy, vol. 2, no. 5, pp. 25–31, Sep. 2004.

[21] Ahmed, M., Mahmood, A. N., & Hu, J. (2016) “A survey of network anomaly detection techniques” Journal of Computer Network

[22] Chandola, V., Banerjee, A., & Kumar, V. (2009) “Anomaly detection: A survey”. Journal of ACM Computing Surveys (CSUR)

[23] Ni, Y., Liu, Y., & Vasilakos, A. V. (2018) “A survey

of user behavior analytics for insider threats detection: Approaches, challenges and solutions" Journal of Future Generation Computer Systems

[24] Liu, H., Lang, B., Liu, M., & Yan, H. (2019) " CNN and RNN based payload classification methods for attack detection" Journal of Computers &Security.

[25] Gavai, G., Sricharan, K., Gunning, D., Hanley, J., & Rolleston, R. (2015) "Supervised and unsupervised methods to detect insider Threat from enterprise social and web access logs" in 2015 Conference IEEE Joint Intelligence and Security Informatics Conference (JISIC)