INTERNATIONAL JOURNAL OF RESEARCH, EXPLORATION & KNOWLEDGE (IJREK)
Official Scholarly Journal Portal: **https://www.ijrek.org** | DOI: **https://doi.org/10.5281/zenodo.23642669**

AI BASED PASSWORD STRENGTH ANALYZER

Mr.Suryawanshi A.M.[1], Dhumal Smita[2]

[1] Faculty Guide & Supervisor | [2] Dattakala Group Of Institutions Faculty Of Engineering Swami - Chincholi tal. Daund dist. Pune, India

Corresponding author: **dhumalsmita6@gmail.com**

ARTICLE ID	ARTICLE TYPE	RESEARCH AREA	PUBLICATION
IJREK-2026-00162	Research Article	Artificial Intelligence	Volume 1, Issue Issue 1 (2026)
RECEIVED	**REVISED**	**ACCEPTED**	**PUBLISHED**
25 October 2026	30 October 2026	26 October 2026	02 November 2026

Abstract

Passwords remain one of the most common methods of user authentication, but weak and reused passwords continue to expose accounts to guessing, credential stuffing, and brute-force attacks. This paper proposes an AI Based Password Strength Analyzer that evaluates password quality and provides understandable recommendations for improvement. The system combines conventional password-strength indicators—such as length, character variety, predictable patterns, and common-password matches—with a machine-learning component that can learn patterns associated with weak passwords from appropriately prepared datasets. Rather than treating a password as simply valid or invalid, the analyzer assigns a strength category and explains which characteristics influenced the assessment. Privacy is a central design requirement: password text should be processed locally whenever possible, should not be stored or logged, and should never be transmitted to a server without a clear security justification. The paper describes the proposed architecture, analysis workflow, implementation approach, evaluation plan, and limitations. The system is intended as a decision-support tool that encourages safer password choices; it does not replace multi-factor authentication, password managers, secure storage, or established authentication controls.

Keywords: *Artificial Intelligence; Password Security; Password Strength; Machine Learning; Authentication; Cybersecurity*

2. PROBLEM STATEMENT

Many users choose passwords that appear complex but are still predictable, such as a common word followed by a year or a symbol. A simple checklist may reward character variety without sufficiently accounting for length, common patterns, or likely guessing behavior. At the same time, an opaque machine-learning score may confuse users if it does not explain why a password is considered weak. The problem is to design a password-strength analyzer that combines dependable security heuristics with AI-assisted pattern recognition and communicates actionable feedback without collecting or exposing users' passwords. 3. Objectives The main objectives of the proposed system are: • To assess password length, character diversity, repetition, sequences, and predictable structures. • To identify passwords that match known common-password lists or obvious variants, where permitted. • To explore a machine-learning classifier that distinguishes likely weak patterns from stronger patterns. • To provide a strength label and concise recommendations that users can understand. • To protect privacy by avoiding password storage, logging, and unnecessary network transmission. • To evaluate the analyzer for classification quality, consistency, response time, and user comprehension.

4. SCOPE OF THE SYSTEM

The proposed application is designed as an advisory tool that can be integrated into a sign-up page, password-change screen, or standalone security utility. It estimates password quality at the time of entry. It does not verify a user's identity, test whether an account has been compromised, or guarantee that a password cannot be guessed. It also does not replace secure password hashing, rate limiting, multi-factor authentication, or a password manager.

5. RELATED WORK

Password-security guidance has long recommended adequate length, avoidance of common and compromised passwords, and support for password managers. Research on password guessing has shown that attackers exploit human choice patterns rather than trying all possible strings uniformly. This makes predictable phrases, substitutions, and common suffixes important signals. Rule-based strength meters are easy to explain and implement, but their results depend on the quality of their rules. Statistical and machine-learning approaches can learn patterns from datasets, yet they may inherit dataset bias, produce misleading confidence, or encourage users to optimize for a score rather than actual security. The proposed design therefore combines transparent rules with an optional model and treats the output as an estimate rather than a guarantee.

6. PROPOSED SYSTEM

The AI Based Password Strength Analyzer receives a password through a protected input field, evaluates it using deterministic checks, optionally applies a trained model, and returns a strength category with specific suggestions. The user sees feedback such as "increase length," "avoid a common phrase," or "remove a predictable sequence," rather than only a numerical score.

6.1 MAIN MODULES

Input and Privacy Layer The input layer accepts a password through a masked field and ensures that the value is not written to application logs, analytics events, browser storage, or debugging output. Local processing is preferred. If a remote model is required, the design should disclose that fact and use an appropriately reviewed privacy and security mechanism. Feature Extraction Module This module calculates non-secret descriptive features such as total length, character classes, repeated-character runs, sequential patterns, dictionary-like components, and common-password matches. It should avoid retaining the raw password after analysis. Feature design must be reviewed carefully because even derived data can reveal information about the password. Rule-Based Scoring Engine A transparent rules engine checks basic indicators. Length is treated as a major factor; character variety is considered alongside length rather than as a substitute for it. Common passwords, predictable substitutions, repeated characters, and keyboard sequences reduce the estimated strength. AI Prediction Module A machine-learning classifier

may estimate whether a password resembles weak patterns observed in a training dataset. Candidate models include logistic regression, decision trees, or gradient-boosted trees over engineered features. Model output is combined cautiously with rule-based findings, and the interface should not display unjustified precision. Feedback and Recommendation Module The final module converts analysis into a category—such as Weak, Moderate, Strong, or Very Strong—and gives actionable guidance. Feedback should not disclose a full password in error messages or send it to third-party services.

6.2 EXPECTED OUTPUT

The result screen may show a strength category, a short explanation of the most important contributing factors, and a list of suggested improvements. The user should be encouraged to use a unique, sufficiently long password or passphrase and to store it in a reputable password manager.

7. SYSTEM ARCHITECTURE

The proposed architecture consists of five logical stages: secure input, feature extraction, rule-based assessment, optional AI inference, and feedback generation. The stages can run entirely in the browser or local application where feasible. A modular design allows the rule engine to remain usable even when the model is unavailable. Stage Responsibility

1. SECURE INPUT

Accept masked input; avoid logging or persistence.

2. FEATURE EXTRACTION

Measure length, repetition, sequences, and common patterns.

TABLE I

Stage	Responsibility
1. Secure Input	Accept masked input; avoid logging or persistence.
2. Feature Extraction	Measure length, repetition, sequences, and common patterns.
3. Rule Engine	Apply explainable password-strength heuristics.
4. AI Model (optional)	Estimate weakness patterns learned from prepared data.
5. Feedback UI	Display category and practical improvement tips.

3. RULE ENGINE

Apply explainable password-strength heuristics.

4. AI MODEL (OPTIONAL)

Estimate weakness patterns learned from prepared data.

5. FEEDBACK UI

Display category and practical improvement tips. 8. Methodology Step 1: Requirement Analysis Identify the target use case, supported platform, privacy requirements, and intended users. Define which feedback is necessary and ensure the application does not require users to submit real passwords to a central service. Step 2: Dataset Preparation For model development, use appropriately licensed, ethically sourced password-strength datasets or synthetic examples. Do not collect passwords from real users. Remove duplicates, document dataset origin, and separate training, validation, and test data to reduce leakage. Training examples should represent both common weak patterns and stronger, longer choices. Step 3: Feature Engineering Extract features such as length, repeated runs, predictable sequences, common-word indicators, and character distribution. Avoid relying only on the number of character classes. Do not use personal information or hidden user attributes as features. Step 4: Model Training Train a simple baseline model first, then compare candidate classifiers using the same data splits. Select a model based on generalization, calibration, interpretability, and computational cost—not only training accuracy.

8. METHODOLOGY (CONTINUED)

Step 5: Hybrid Decision Logic Combine the model prediction with deterministic security rules. A common-password match or highly predictable pattern should trigger a clear warning even if the model score appears favorable. Thresholds should be selected using validation data and documented. The system should avoid presenting an exact cracking time unless it has a defensible threat model and assumptions. Step 6: User Feedback Translate the analysis into plain language. Recommendations should focus on meaningful changes, such as increasing length, choosing an unrelated passphrase, avoiding common phrases, and using a unique password. Avoid forcing unnecessary symbol substitutions if they make a password harder to remember without materially improving resistance to guessing. Step 7: Testing Test known weak examples, long passphrases, repeated strings, keyboard sequences, common substitutions, and randomly generated strings. Include boundary tests, malformed inputs, empty input, Unicode characters, and very long entries. Confirm that password values are not written to logs or retained after analysis.

9. TOOLS AND TECHNOLOGIES

A prototype can be implemented using HTML, CSS, and JavaScript for the user interface. Python may be used for dataset preparation and model training, with libraries such as scikit-learn for conventional machine-learning models. A browser-based rule engine can support local analysis. A backend is optional and should be avoided unless required for a justified purpose. Version control can be managed with Git. Unit tests can verify scoring rules and edge cases. Security review should include checking console output, analytics, network requests, error reports, and storage behavior. The selected tools should match the development environment and project requirements.

10. PRIVACY AND SECURITY CONSIDERATIONS

Password text is highly sensitive. The safest default is to analyze it locally and discard it immediately after producing feedback. The application must not save passwords in a database, send them to an external AI service, or include them in telemetry. If a remote service is unavoidable, users must be clearly informed and the design must undergo a privacy and security review. A strength analyzer is not an authentication system. Production services should store passwords using a suitable password-hashing scheme with unique salts and appropriate work factors, protect reset flows, apply rate limits, and offer multi-factor authentication where appropriate. The analyzer should not reveal whether a password belongs to a known account or expose data from breach corpora. 11. Limitations The model's quality depends on its training data and may not generalize to every language, keyboard, or user population. A strength label is an estimate and cannot predict all attacker behavior. Common-password lists may be incomplete or outdated. A high score can also create false confidence if users reuse a password or if the service storing it is compromised.

12. EVALUATION PLAN

Since this paper describes a proposed system, numerical performance results are not claimed. The prototype should be evaluated using a held-out test set and a set of manually reviewed examples.

12.1 CLASSIFICATION METRICS

Accuracy may be reported, but it should not be used alone when weak and strong classes are imbalanced. Precision, recall, F1-score, confusion matrix, and—where suitable—ROC-AUC can help characterize the model. The cost of incorrectly labeling a weak password as strong should receive particular attention.

12.2 USABILITY METRICS

A small usability study can assess whether users understand the strength category, can identify the recommended change, and can improve a sample password after reading the feedback. Collect only the minimum data needed and do not ask participants to disclose their personal passwords.

12.3 PERFORMANCE AND PRIVACY TESTS

Measure response time for typical and unusually long inputs. Verify that analysis works without unnecessary network access and that password values do not appear in browser storage, application logs, crash reports, or analytics. Confirm that the interface remains accessible and usable with keyboard navigation and screen readers.

13. EXPECTED RESULTS

The expected outcome is a working prototype that provides immediate, understandable feedback and detects common weak-password patterns more consistently than a simple character-count rule. The hybrid design is expected to retain the transparency of deterministic checks while allowing a trained model to identify additional patterns. These are design expectations, not measured findings; actual performance must be established through testing. 14. Applications • Registration and password-change forms for websites and web applications. • Educational tools that teach users how to create stronger passwords. • Internal security awareness exercises using synthetic examples. • Prototype authentication interfaces for academic projects. • Password policy testing during application development.

15. FUTURE SCOPE

Future improvements may include better multilingual pattern detection, calibration of model confidence, accessibility testing, and integration with password-manager guidance. The project could compare rule-only, model-only, and hybrid approaches under a documented threat model. Further work may also investigate privacy-preserving ways to check common or compromised passwords without transmitting the user's password, provided that the method is independently reviewed. 16. Conclusion The AI Based Password Strength Analyzer is proposed as a privacy-conscious tool for helping users make better password choices. By combining explainable rules with optional machine-learning pattern recognition, it can offer more useful feedback than a simplistic complexity meter. Its design prioritizes length, predictability, common-password detection, clear recommendations, and safe handling of sensitive input. The analyzer should be treated as one layer of a broader security strategy. It cannot guarantee password safety and must not replace unique passwords, password managers, secure password hashing, rate limiting, or multi-factor authentication. A complete implementation should be validated with independent test data and usability checks, and should demonstrate that password text is never unnecessarily retained or transmitted. 17. References [1] National Institute of Standards and Technology (NIST), Digital Identity Guidelines: Authentication and Lifecycle Management, NIST Special Publication 800-63B. Available at: https://pages.nist.gov/800-63-4/sp800-63b.html [2] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms," IEEE Symposium on Security and Privacy, 2012. [3] J. Bonneau, "The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords," IEEE Symposium on Security and Privacy, 2012. [4] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password Cracking Using Probabilistic Context-Free Grammars," IEEE Symposium on Security and Privacy, 2009. [5] OWASP Foundation, "Authentication Cheat Sheet." Available at: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html [6] OWASP Foundation, "Password Storage Cheat Sheet." Available at: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

Note: This paper describes a proposed design and evaluation plan. Any implementation claims or performance figures should be added only after the prototype has been built and tested.

Publication archival note: Official scholarly contribution published under CC BY 4.0 International License. Permanent record on **https://www.ijrek.org** | Article ID: IJREK-2026-00162