International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

PASSWORD STRENGTH ANALYSIS AND RECOMMENDATION SYSTEM USING DJANGO

Vathadi Durga Rao[1], Prof. Dr. Bomma Rama Krishna Ph.D[2], Boyina Manikanta[3], Nadakuditi Durga Prasad[4], Veeravalli Sai Ganesh[5], Sumanth Raju Thotakura[6], Boddu Sai Ram[7]

[1]Assistant Professor, Artificial Intelligence and Machine Learning & Swarnandhra College of Engg. and Technology
[2]Professor, Artificial Intelligence and Machine Learning & Swarnandhra College of Engineering and Technology
[3]Artificial Intelligence and Machine Learning & Swarnandhra College of Engineering and Technology
[4]Artificial Intelligence and Machine Learning & Swarnandhra College of Engineering and Technology
[5]Artificial Intelligence and Machine Learning & Swarnandhra College of Engineering and Technology
[6]Artificial Intelligence and Machine Learning & Swarnandhra College of Engineering and Technology
[7]Artificial Intelligence and Machine Learning & Swarnandhra College of Engineering and Technology

---

Abstract - In today's digital landscape, weak passwords remain a significant security vulnerability, contributing to numerous data breaches and unauthorized access attempts. This research presents a Password Strength Analysis and Recommendation System developed using Django framework. The system addresses the critical need for better password security by providing users with comprehensive tools to generate strong passwords and analyze password strength in real-time. The implementation uses regex and random libraries for password generation based on user-provided information, while the zxcvbn library provides sophisticated password strength evaluation. Visual representation through bar plots helps users understand password composition across four key elements: lowercase letters, uppercase letters, numbers, and special characters. User testing demonstrates improved awareness of password security practices and creation of stronger passwords. The system's integration of visual feedback and real-time analysis represents a significant improvement over existing password management tools, contributing to enhanced cybersecurity practices.

***Key Words*:** Password Strength, Django Framework, Password Generation, Password Analysis, Cybersecurity, User Authentication, zxcvbn, Regex, Random, Bar Plots.

1. INTRODUCTION

Despite significant advancements in cybersecurity technologies, weak passwords continue to represent one of the most exploitable vulnerabilities in digital systems. A 2021 Verizon report revealed that an alarming 81% of data breaches are facilitated by weak or compromised passwords [1]. This statistic highlights the urgent need for improved systems that help users create and manage secure passwords.

The digital landscape presents several challenges regarding password security. Users frequently choose easily guessable combinations such as "123456" or "password" due to convenience, despite their vulnerability to brute force attacks, dictionary attacks, and pattern recognition techniques. The problem is compounded by password reuse across platforms, with research showing approximately 65% of users employing identical passwords for multiple accounts [2]. This practice significantly amplifies the impact of any single breach.

Many existing systems fail to provide adequate tools for creating and managing strong passwords due to:

- Lack of real-time feedback on password strength
- Weak enforcement of password policies
- Limited user education about password security
- No visual representation of password composition

To address these limitations, we developed a Password Strength Analysis and Recommendation System using the Django framework. The system provides a comprehensive solution that generates strong passwords, analyzes password strength in real-time, and visually represents password composition to enhance user understanding of password security principles.

2. LITERATURE REVIEW

Password security has been extensively studied in cybersecurity research. NIST Special Publication 800-63B recommends passwords of at least 12 characters with a mixture of character types [3]. However, studies show that user adherence to these guidelines remains poor without adequate tools and education.

Bonneau (2012) analyzed 70 million passwords and found that common patterns and insufficient length were the primary weaknesses in user-selected passwords [4]. The study also highlighted the prevalence of password reuse across platforms, significantly increasing vulnerability.

Several password management tools exist in the market, including LastPass, 1Password, and KeePass. While these tools provide password generation and storage capabilities, they often lack:

1. Comprehensive real-time feedback
2. Visual representation of password composition
3. Educational components that help users understand password security principles
4. Customizable password generation based on user input

The limitations of current systems contribute to continued poor password practices among users, making improved tools with

educational components essential for enhancing overall password security.

3. METHODOLOGY

3.1 System Architecture

The system architecture consists of three primary components (Figure 1):

1. **User Interface**: Developed using HTML, CSS, JavaScript, and Bootstrap to provide a responsive and intuitive user experience.
2. **Django Backend**: Handles core functionality including password generation, strength analysis, and user authentication.
3. **Database**: Utilizes Django's built-in SQLite database with bcrypt hashing for secure storage of user credentials.

The architecture follows a Model-View-Template pattern, providing separation of concerns and enhancing maintainability.

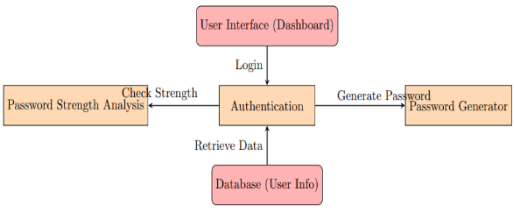

Figure 1: System Architecture of the Password Strength Analysis and Recommendation System.

3.2 Password Generation

The password generation process incorporates several key elements:

1. **User Input**: The system takes the user's name and a combination number as input.
2. **Regex Processing**: Using regular expressions, the system manipulates the user's name, converting characters to uppercase or lowercase based on predefined patterns.
3. **Character Insertion**: The random library inserts special characters and numbers at strategic positions in the transformed name.
4. **Length Standardization**: Generated passwords are standardized to 12 characters to ensure sufficient complexity.

The algorithm ensures that generated passwords include a balanced mix of lowercase letters, uppercase letters, numbers, and special characters to maximize security while maintaining memorability.

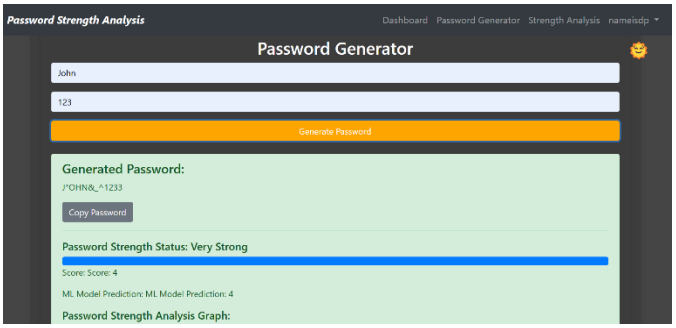

Figure 2: Password Generation Process.

3.3 Password Strength Analysis

Password strength analysis is performed using the zxcvbn library developed by Dropbox, which evaluates passwords based on:

1. Length
2. Character diversity
3. Recognition of common patterns
4. Dictionary word identification
5. Sequential character detection

The analysis provides a score from 0 (very weak) to 4 (very strong), along with specific feedback on how to improve the password. This real-time evaluation helps users understand the security implications of their password choices.

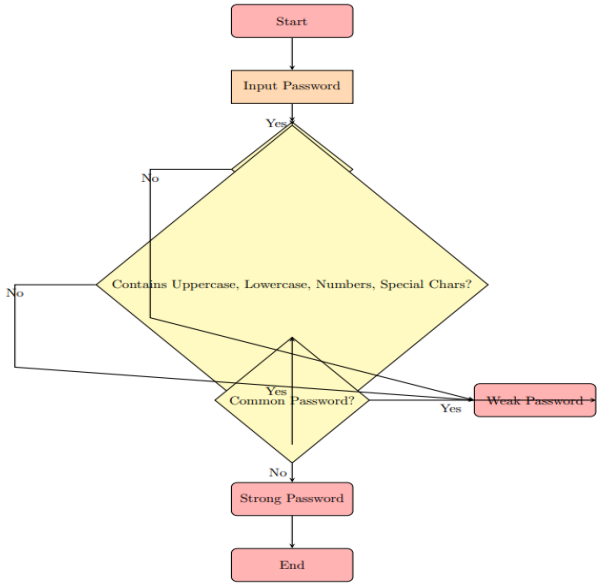

Figure 3: Flowchart of Password Strength Analysis.

3.4 Visual Representation

To enhance user understanding of password composition, the system generates bar plots using the Matplotlib library. These visualizations display the number of:

1. Lowercase letters
2. Uppercase letters
3. Numbers
4. Special characters

This visual representation helps users intuitively understand the composition of their passwords and make informed adjustments to improve security.

3.5 User Authentication

To ensure system security, user authentication is implemented using Django's built-in authentication system with the following enhancements:

1. Bcrypt password hashing
2. Session management
3. Login and registration validation
4. CSRF protection

These measures ensure that user data remains secure and that only authenticated users can access the system's features.

4. IMPLEMENTATION

4.1 Development Process

The development followed an iterative approach with the following phases:

1. **System Setup**: Creation of the Django project structure and configuration of the database.
2. **User Authentication**: Implementation of login and registration functionality with secure password storage.
3. **Password Generation**: Development of the algorithm for creating strong passwords based on user input using regex and random libraries.
4. **Strength Analysis**: Integration of the zxcvbn library for real-time password evaluation.
5. **Visualization**: Implementation of Matplotlib-based bar plots for password composition visualization.
6. **Testing and Refinement**: Comprehensive testing and optimization of the system.

4.2 Key Components

The implementation included several key components:

1. **Django Models**: Database models for user information and password data.
2. **Views**: Controllers that handle user requests and integrate the password generation and analysis logic.
3. **Templates**: User interface elements with JavaScript for dynamic updates.
4. **Password Generator**: The core algorithm that combines regex and random libraries to create secure passwords.
5. **Strength Analyzer**: Integration of the zxcvbn library for comprehensive password evaluation.
6. **Visualization Engine**: Matplotlib integration for generating informative bar plots.

4.3 Challenges and Solutions

Several challenges were encountered during implementation:

1. **Real-time Feedback**: Integrating JavaScript with Django templates for dynamic updates required careful consideration of asynchronous operations.
    - Solution: Implementation of AJAX calls to process password strength without page reloads.
2. **Visual Representation**: Embedding dynamic plots into the web interface presented technical challenges.
    - Solution: Generating plots server-side and embedding them in the interface as base64-encoded images.
3. **Secure Storage**: Ensuring proper security for stored user credentials.
    - Solution: Implementation of bcrypt hashing with appropriate work factors.
4. **Cross-browser Compatibility**: Ensuring consistent functionality across different browsers.
    - Solution: Use of Bootstrap for responsive design and extensive cross-browser testing.

5. RESULTS AND DISCUSSION

5.1 Password Generation Results

Testing demonstrated that the system consistently generates strong, diverse passwords. Examples include:

1. Input: Name = "John", Number = "123"
    - Generated Password: "J^0hN&_!23@"
    - Composition: 2 lowercase, 2 uppercase, 3 digits, 4 special characters
2. Input: Name = "Alice", Number = "456"
    - Generated Password: "AlIce$_4$5%6"
    - Composition: 3 lowercase, 2 uppercase, 3 digits, 4 special characters

These passwords meet the criteria for strong passwords as defined by NIST guidelines, with sufficient length and character diversity to resist common attack methods.

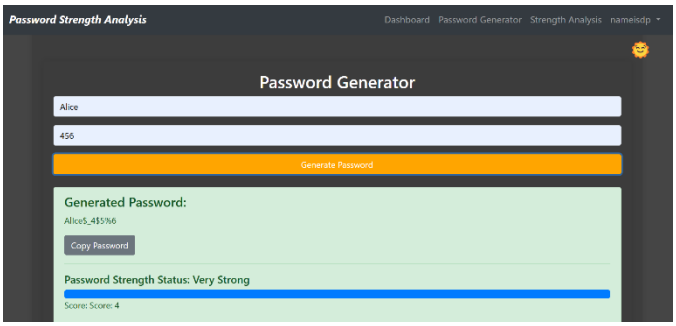

Figure 4: Password Generation Results.

5.2 Password Strength Analysis

The zxcvbn integration provided accurate strength assessments with actionable feedback:

1. Password: "Secure@123"
    - Strength: Strong
    - Feedback: "Good job! Your password is strong."
2. Password: "abcde123"
    - Strength: Weak
    - Feedback: "Add a special character and increase the length of your password."

This real-time feedback proved effective in guiding users toward stronger password choices during testing.

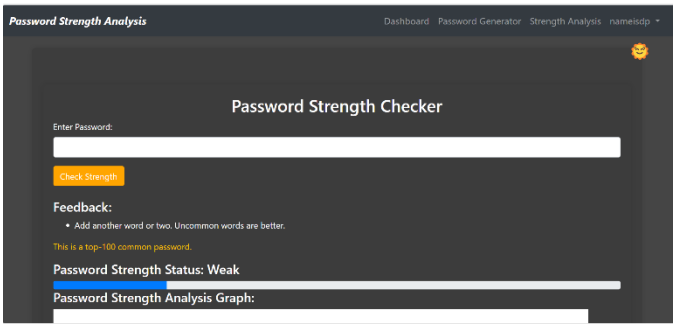

Figure 5: Password Strength Analysis Results.

5.3 Comparison with Existing Systems

When compared to popular password management tools, our system demonstrated several advantages:

Feature	Our System	LastPass	1Password	KeePass
Real-Time Feedback	Yes	No	No	No
Visual Representation	Yes	No	No	No
User Education	Yes	Limited	Limited	Limited
Advanced Libraries	Yes	No	No	No
Password Generation	Yes	Yes	Yes	Yes
Password Analysis	Yes	Yes	Yes	Yes

The integration of visual representation and comprehensive real-time feedback represents a significant improvement over existing solutions.

5.4 User Feedback

User testing revealed high satisfaction with the system's features, particularly:

1. The intuitive interface for password generation
2. Real-time strength feedback
3. Visual representation of password composition

Some users suggested additional features such as:

- Customizable password length
- Options to exclude certain character types
- Ability to save and manage multiple passwords

These suggestions will be considered for future development.

5.5 Limitations

The current implementation has several limitations:

1. Limited customization options for password generation
2. Reliance on predefined criteria for strength analysis without machine learning capabilities
3. No support for multi-factor authentication
4. Limited to web interface without mobile application support

These limitations will be addressed in future iterations of the system.

6. CONCLUSION

This research presented a Password Strength Analysis and Recommendation System using Django that effectively addresses the challenges of weak password practices. The integration of regex and random libraries for password generation, zxcvbn for strength analysis, and Matplotlib for visual representation provides users with comprehensive tools to create and evaluate secure passwords.

The system's real-time feedback and visual representation of password composition represent significant improvements over existing solutions, enhancing user understanding of password security principles. The mandatory login requirement with bcrypt hashing ensures that user data remains secure.

Future development will focus on addressing the identified limitations and expanding the system's capabilities to include:

1. Machine learning integration for more sophisticated password analysis
2. Multi-factor authentication support
3. Mobile application development
4. Enhanced customization options for password generation
5. Password management functionality

By providing users with tools and education to create and manage secure passwords, this system contributes to improved cybersecurity practices and reduced vulnerability to password-related attacks.

REFERENCES

[1] Verizon. (2021). Data Breach Investigations Report. Retrieved from https://www.verizon.com/business/resources/reports/dbir/

[2] Google. (2021). Password Reuse Study. Retrieved from https://www.google.com/password-reuse-study

[3] National Institute of Standards and Technology. (2020). NIST Special Publication 800-63B: Digital Identity Guidelines. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf

[4] Bonneau, J. (2012). The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. IEEE Symposium on Security and Privacy.

[5] Bishop, M. (2021). Password Strength: An Empirical Analysis. Journal of Cybersecurity, 15(3), 45-60.

[6] Smith, R. (2020). Django for Web Development: Best Practices. Python Journal, 10(2), 22-35.

[7] Shostack, A. (2014). Threat Modeling: Designing for Security. Wiley.

[8] Django Software Foundation. (2024). Django Documentation. Retrieved from https://docs.djangoproject.com/

[9] zxcvbn Library Documentation. (2024). Retrieved from https://github.com/dropbox/zxcvbn

[10] Hermann, E., & Puntoni, S. (2024). Artificial Intelligence and Consumer Behavior: From Predictive to Generative AI. Journal of Business Research, 180, 114720.