myresearchgo Volume 1, November Issue 8, 2025, ISSN: 3107-3816 (Online)

Secure Password Manager Website

[1] Atharva R. Khot, [2] Jiteshree P. Raut
[1] Student, [2] Assistant Professor

Department of Information Technology

S. D. S. M. College, Palghar, Maharashtra, India

***Abstract*:** In the digital era, managing and safeguarding passwords has become a critical challenge due to the increasing number of online accounts and the rising threat of cyberattacks. This project, titled "Secure Password Manager," is designed to provide a safe, user-friendly, and efficient solution for password storage and retrieval. The system is developed using a full-stack architecture consisting of an Angular frontend, a Django backend, and a MySQL database hosted on WAMP Server. **The core functionality revolves around** secure password encryption and decryption **using the** Fernet encryption algorithm from Python's cryptography library. Passwords are never stored in plain text; instead, they are encrypted using a secret Fernet key before being stored in the database, ensuring complete confidentiality. The application avoids the use of browser local storage to enhance data security and minimize exposure to client-side attacks. The backend implements JWT-based authentication to ensure that only authorized users can access their password vaults. The frontend provides an intuitive interface for users to manage their credentials—add, view, update, and delete passwords—while maintaining a seamless and responsive experience through Angular routing and Material Design components. Overall, this project demonstrates a secure, scalable, and modern web-based password management system that emphasizes data protection, encryption integrity, and secure authentication**,** offering users a reliable tool to manage digital credentials safely.

Keywords: password, password manager, secure password, security, JWT, Django

1. Introduction

Password management is a critical component of personal and organizational security. Users have multiple online accounts and often use weak or reused passwords, exposing them to account takeover and data breaches. A secure password manager stores credentials encrypted, enforces strong password policies, and provides secure sharing capabilities.

With the rapid expansion of digital services, users are required to maintain multiple online accounts across banking, social media, e-commerce, and enterprise platforms. Each of these accounts demands secure authentication, commonly implemented through passwords. However, due to the difficulty of remembering many complex passwords, users often adopt insecure habits such as reusing passwords, choosing weak or predictable passwords, or storing them in unsafe locations. These practices significantly increase vulnerability

to cyber-attacks such as credential stuffing, brute-force attacks, phishing, and data breaches. Therefore, there is a critical need for a secure and convenient method to store and manage passwords.

The system is implemented using **Angular** for the frontend and **Django REST Framework** with **MySQL** for the backend. To ensure data confidentiality, user passwords are encrypted using the **Fernet symmetric encryption algorithm**, meaning the actual password is never stored in plain text at any stage. Only the encrypted form of each password is stored in the database, and decryption is performed only for authenticated users.

The application also adopts **JWT-based authentication** with tokens managed securely through **HttpOnly cookies**, preventing client-side access and reducing risk of token theft or session hijacking. The frontend provides an intuitive interface for adding, editing, searching, and retrieving stored passwords, while ensuring that no authentication data is exposed through local storage or browser memory.

By combining strong cryptographic techniques, modern authentication practices, and a user-friendly interface, the Secure Password Manager enhances both **security and usability** in password handling. This system encourages users to adopt safer password behaviors without compromising convenience, contributing to improved personal and organizational cybersecurity hygiene.

2. Objectives

The objectives of the project are:
• To securely store user credentials using authenticated encryption (AES-GCM) with per-entry salts and IVs.
• To provide a secure sharing mechanism using hybrid RSA encryption so users can safely share credentials.
• To enforce password strength checks and provide users with constructive feedback.
• To protect the application against common web attacks (XSS, CSRF, SQL injection) and implement secure session handling.
• To design a system where a leaked database does not expose plaintext credentials (key separation and secure key storage).

3. Problem Statement

Users frequently reuse passwords or keep them in insecure places (notes, email), which exposes them to credential stuffing and phishing. Even when a password manager exists, improper handling of cryptographic keys, weak KDF parameters, or storage of secrets in plaintext can lead to catastrophic data breaches. This project aims to minimize these risks by implementing a design that protects against offline attacks when the database is leaked and defends against common web-based attacks.

4. Hypothesis

If credentials are encrypted using strong authenticated encryption (AES-GCM) with keys derived from a high-cost KDF (Argon2/PBKDF2) using per-user salts, and if private asymmetric keys are protected (encrypted

under the user's master key or stored in a secure KMS), then a leaked database alone will not allow an attacker to recover plaintext passwords. Combining HttpOnly cookie-based sessions with CSRF protection will reduce the chance of token theft and session hijacking.

5. Limitations

• Master password is not stored; if forgotten, user data cannot be recovered unless they previously exported encrypted backups or used an out-of-band recovery procedure.
• The demo implementation performs encryption on the server (requires sending master password). Production must move encryption client-side (WebCrypto) for zero-knowledge.
• Deployment requires HTTPS (TLS) and secure key management (KMS/HSM) for production-grade security.
• Performance: KDFs with high memory/CPU cost (Argon2) increase client/server CPU use but are necessary for security.

6. Methodology

Architecture overview (three tiers):
1. Frontend — Angular (non-standalone): UI for registration, login, vault management, sharing and strength analysis. Client performs password-strength checks and (ideally) client-side encryption.
2. Backend — Django REST Framework: Authentication, key management, encryption APIs (if server-side), and storage logic. Uses custom cookie-based JWT authentication and CSRF protection.
3. Database — MySQL (WAMP): Stores encrypted ciphertexts, salts, IVs, public keys, and audit logs.

7. Future Scope

• Add biometric unlock (WebAuthn/FIDO2) to protect keys on devices.
• Create browser extensions for autofill and site detection.
• Integrate with external KMS (AWS KMS/HashiCorp Vault) to avoid storing private keys in DB.
• Add breach-detection services and password reuse alerts.
• Build offline encrypted backups and secure export/import features.

8. Review of Literature

This paper provides best practices for password handling and storage, endorsing strong KDFs and the use of authenticating encryption.[1]

The paper recommends secure password policies, KDF usage, and advice on password storage and recovery.[2]
This research indicates client-side encryption provides better security guarantees (zero-knowledge) than server-side encryption. This project follows that guidance for future improvements.[3]

The Solutions such as Bitwarden and 1Password use client-side encryption, zero-knowledge servers, and hybrid-sharing techniques – these informed the design choices here.[4]

9. Conclusion

The Secure Password Manager successfully addresses the growing need for safe and efficient password handling in an increasingly digital world. With the rising number of online accounts and cyber threats, users face challenges in managing strong, unique passwords without compromising usability. This project provides a secure, scalable, and user-friendly solution that allows users to store, view, and manage their passwords with confidence.

Additionally, the implementation of JWT-based authentication with HttpOnly cookies prevents token theft and strengthens user session security. The system's interface allows users to efficiently add, update, and retrieve passwords, promoting the use of stronger and unique credentials across different platforms.

Overall, the project demonstrates how modern encryption, secure authentication mechanisms, and thoughtful system design can significantly enhance password security without sacrificing ease of use. The Secure Password Manager not only improves personal cybersecurity practices but also lays a strong foundation for future enhancements such as two-factor authentication, biometric login support, cloud synchronization, and mobile application integration. This establishes the system as a practical and adaptable tool for real-world password management needs.

10. References

1. OWASP Foundation, 'Password Storage Cheat Sheet'.
2. NIST Special Publication 800-63B, 'Digital Identity Guidelines'.
3. Django REST Framework Documentation. https://www.django-rest-framework.org/
4. Angular Documentation. https://angular.io/docs
5. RFC 8018 - PKCS #5: Password-Based Cryptography Specification.
6. PyCryptodome and cryptography library documentation.