Password Manager Project Report

Name: Achutha Surrneni

Domain: Python

Introduction

The Password Manager project, developed using Python, is an application designed to securely store, manage, and retrieve passwords. With increasing online accounts, managing numerous complex passwords has become a critical need. This project aims to provide a secure, user-friendly solution to this problem by leveraging Python's capabilities.

Objectives

- To provide a secure way to store and manage passwords.
- To ensure that users can generate and retrieve strong passwords easily.
- To develop a user-friendly interface for easy accessibility.
- To implement robust security measures to protect stored passwords.

Development Tools and Technologies

- **Programming Language**: Python
- **Libraries**:
  - **Tkinter**: For creating the graphical user interface (GUI).
  - **SQLite**: For local database storage.
  - **Cryptography**: For encryption and decryption of passwords.
  - **Hashlib**: For hashing the master password.

Features

1. **Secure Storage**:
   - All passwords are stored in an encrypted format using AES encryption.
   - Passwords can only be decrypted and accessed by the user with the correct master password.
2. **User Authentication**:
   - A master password is required to access the application.
   - The master password is hashed and stored securely.

3. **Password Generation**:
    - The application includes a feature to generate strong, random passwords.
    - Users can specify criteria such as length and inclusion of special characters.
4. **Password Retrieval**:
    - Users can easily retrieve stored passwords.
    - A search functionality is provided to quickly find the desired credentials.
5. **User Interface**:
    - Developed using Tkinter, the GUI is simple and intuitive.
    - Provides a seamless user experience even for non-technical users.
6. **Cross-Platform Compatibility**:
    - The application is designed to run on Windows, macOS, and Linux.
7. **Database Integration**:
    - Utilizes SQLite for fast and reliable local database storage.
    - Ensures that user data is efficiently managed.
8. **Backup and Restore**:
    - Features options to backup and restore the password database.
    - Protects against data loss.
9. **Security Measures**:
    - Implements automatic logout after a period of inactivity.
    - Protects against brute-force attacks by limiting login attempts.

Architecture

The application follows a modular architecture with a clear separation of concerns:

- **User Interface Module**: Handles all interactions with the user through Tkinter.
- **Database Module**: Manages storage and retrieval of encrypted passwords using SQLite.
- **Security Module**: Handles encryption, decryption, and hashing functionalities.

Implementation

1. **Encryption and Decryption**:
    - The **cryptography** library is used to encrypt passwords before storing them in the database.
    - Decryption is performed when the user needs to retrieve a password.
2. **Hashing**:
    - The **hashlib** library hashes the master password using a secure algorithm (e.g., SHA-256).
3. **Database Management**:
    - SQLite is used to create a local database where encrypted passwords are stored.
    - The database schema includes fields for account names, usernames, encrypted passwords, and other metadata.
4. **Graphical User Interface**:
    - Tkinter is used to build the GUI, providing input fields for account details and buttons for actions like saving, retrieving, and generating passwords.

Testing

- Extensive testing was conducted to ensure the security and functionality of the application.
- Unit tests were written for each module to verify correctness.
- User acceptance testing was performed to gather feedback on usability.

Potential Improvements

- **Cloud Synchronization**: Adding features to sync passwords across multiple devices securely.
- **Browser Integration**: Developing extensions for browsers to auto-fill and save passwords.
- **Multi-Factor Authentication**: Enhancing security by supporting multi-factor authentication for accessing the password manager.

Conclusion

The Password Manager project, developed using Python, successfully provides a secure and user-friendly solution for managing passwords. With features like encryption, a master password system, and an intuitive GUI, it addresses the need for secure password management effectively. Future enhancements can further improve its functionality and security, making it an indispensable tool for users.

This project demonstrates the effective use of Python and its libraries to develop a practical application that meets modern security needs.