**PROJECT SYNOPSIS**

Hashed Password Cracker Simulation

# Project Overview

The 'Hashed Password Cracker Simulation' project is a tool designed to simulate the process of cracking hashed passwords using different attack strategies. These strategies include Dictionary Attack, Brute-Force Attack, and Mask-Based Attack. The tool supports various hashing algorithms such as MD5, SHA-1, and SHA-256. This project aims to demonstrate password security testing and crack hashed passwords by exploiting weaknesses in commonly used hashing techniques.

# Objectives

The main objectives of this project are:  
1. To provide a tool for hashing passwords using popular hashing algorithms (MD5, SHA-1, SHA-256).  
2. To implement various password-cracking techniques, including:  
 - Dictionary Attack: A method where a wordlist is used to test against the hashed password.  
 - Brute-Force Attack: A method where all possible combinations of characters up to a defined length are tested.  
 - Mask-Based Attack: A method that uses a user-defined pattern or mask to limit the character space and attempt cracking.  
3. To implement multithreading to speed up brute-force and mask-based attacks for better performance.  
4. To create an interactive command-line interface where users can hash passwords or attempt to crack hashed passwords.

# Methodology

The tool operates in a step-by-step process where users can choose between hashing a password or attempting to crack a given hash. The project utilizes Python's built-in `hashlib` library for hashing and various techniques for cracking passwords:  
1. \*\*Hashing a Password\*\*: The user inputs a password, selects a hashing algorithm (MD5, SHA-1, or SHA-256), and the password is hashed.  
2. \*\*Cracking a Password Hash\*\*: The user inputs a hash value and selects an attack method. The available attack types are:  
 - \*\*Dictionary Attack\*\*: A wordlist is used to check against the hashed password.  
 - \*\*Brute-Force Attack\*\*: A multi-threaded brute-force approach that generates all possible combinations of characters up to a specified length.  
 - \*\*Mask-Based Attack\*\*: A custom pattern (mask) is defined to reduce the search space and crack the password.  
3. \*\*Multithreading\*\*: The brute-force and mask-based attacks are optimized by using multithreading for concurrent password attempts, making the process faster and more efficient.

# Results

The program successfully implements the three attack methods, with the following results:  
1. \*\*Dictionary Attack\*\*: Cracks passwords that are present in the provided wordlist.  
2. \*\*Brute-Force Attack\*\*: Attempts all combinations of characters, based on the maximum length specified by the user, to crack the password.  
3. \*\*Mask-Based Attack\*\*: Efficiently reduces the search space by using a custom mask, allowing users to specify character sets for each part of the password.  
4. \*\*Performance\*\*: Multithreading ensures faster password cracking for brute-force and mask-based attacks, improving performance over traditional single-threaded methods.

# Conclusion

The 'Hashed Password Cracker Simulation' project demonstrates a functional password-cracking tool that simulates real-world password cracking methods used by security professionals to test the strength of hashed passwords. By using dictionary attacks, brute-force attacks, and mask-based attacks, the tool helps users understand how various attack strategies work in practice. The inclusion of multithreading significantly improves the efficiency of brute-force and mask-based attacks, making the tool suitable for password security testing in a realistic environment.  
  
This project serves as an educational tool to raise awareness about password security and the importance of using strong, complex passwords. It also highlights the limitations of popular hashing algorithms like MD5 and SHA-1, which can be cracked relatively easily using modern computing power.

# Future Work

Future improvements to this project could include:  
1. \*\*Enhanced Attack Strategies\*\*: Implementing more advanced attack techniques such as Rainbow Table attacks or Hybrid attacks.  
2. \*\*GUI Implementation\*\*: Developing a graphical user interface (GUI) to make the tool more user-friendly.  
3. \*\*Expanded Hashing Algorithms\*\*: Supporting additional hashing algorithms such as bcrypt, scrypt, and Argon2 for more realistic security testing.  
4. \*\*Optimization\*\*: Further optimization of brute-force and mask-based attacks by incorporating more advanced methods like GPU-based computation or distributed cracking.  
5. \*\*Logging and Reporting\*\*: Adding functionality to log the attack attempts and generate reports on attack results.