Department of Telecoms and Networking

COURSE TITLE: Cryptography

TERM 1 | YEAR 3

វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
INSTITUTE OF DIGITAL TECHNOLOGY

Project Title: User Privileges and Access

Lecturer: Mr. Mr. Meas Sothearath

Submission date: [29/11/2025]

STUDENT NAME: THA ROTHSOCHEATA

ID: IDTB100038

I. Overview

PassGuard is a web-based application designed to help users evaluate and enhance the security of their passwords while learning essential cryptographic concepts. The system analyzes password strength, calculates entropy, estimates cracking time, detects common dictionary patterns, and generates cryptographic hashes such as MD5, SHA-1, and SHA-256 based on user input.

The application is developed using Python (Flask) for a lightweight, maintainable backend, and a responsive HTML, CSS, and JavaScript frontend to deliver real-time feedback and an easy-to-use interface. PassGuard aims to improve user awareness of secure password practices and demonstrate how hashing functions operate in practical scenarios.

II. Problem Statements

Many users still create weak passwords because they do not fully understand what makes a password secure or how password hashing works. Common issues include:

- People often reuse simple passwords or use easy-to-guess dictionary words.
- Users do not understand the idea of entropy, so they do not know why longer and more complex passwords are harder to crack.
- There is confusion about one-way hash functions, which are used to protect passwords during storage.
- Existing password check tools are either too technical or do not explain the cryptography behind the results.

As a result, users are more vulnerable to hackers, account takeovers, identity theft, and data breaches.

III. Solutions

The solution is to build a simple, beginner-friendly web application that helps users understand and improve their passwords. The app will provide real-time analysis and easy explanations so that anyone can learn the basics of password security and hashing.
The web app will include:

- Real-time password analysis that shows if a password is Weak, Medium, or Strong as the user types.
- Entropy calculation and an easy-to-read estimate of how long it would take to crack the password using common attack speeds.
- Detection of weak patterns, such as dictionary words, repeated characters (aaaa), or sequences (123123).

- Hash generator for MD5, SHA-1, and SHA-256, with simple explanations of why hashing is one-way.
- Clear tooltips and help messages that explain concepts using plain language.
- The app will use Python (Flask) as the backend for additional features like dictionary checking or optional APIs (e.g., HaveIBeenPwned), while keeping most analysis on the client side for privacy and speed.

User experience features:

- A real-time strength meter that updates instantly.
- One-click "Copy" buttons for hash output.
- Optional password generator for strong suggestions.
- Short educational notes that appear when users interact with each section.
-

IV. Technology Use

Backend (Python)

- Flask – A lightweight Python web framework used to create routes, API endpoints, and optional logging features.
- Werkzeug / bcrypt (optional) – Can be used to demonstrate secure hashing methods like bcrypt, allowing comparison with weaker hashes such as MD5 or SHA-1.
- Requests (optional) – Used if integrating external services like HaveIBeenPwned for checking breached passwords.

Frontend

- HTML / CSS – Builds the structure and style of the website, designed to be responsive and mobile-friendly.
- Vanilla JavaScript – Handles real-time password analysis, validation, and interactive UI elements.
- CryptoJS or Web Crypto API – Performs client-side hashing (MD5, SHA-1, SHA-256) directly in the browser for fast and private processing.
- Bootstrap or Tailwind (optional) – Used to create a clean, modern interface quickly.

Data

- dictionary.json – A small local dictionary used to detect weak or common words inside the user's password.
- localStorage – Stores user preferences or history on the client side (but never stores raw passwords).

Development Tools

- Python 3.10+ – Main programming language for backend development.
- pip & virtualenv – For installing and managing project dependencies.
- Git – For version control and project management.

V. Expected Outcome

Functional

- A working Flask web app with routes serving the UI and API endpoints.
- Client-side real-time password analysis and strength meter.
- Hash generator producing MD5, SHA-1, SHA-256 outputs with copy-to-clipboard.
- Entropy calculation and human-readable cracking-time estimate.
- Dictionary detection and repeated-pattern detection.
- Optional: password suggestion generator, HaveIBeenPwned check (server-side).

Educational

- Users learn why certain passwords are weak and how to make better ones.
- Students understand the basics of hash functions and secure password storage.
- A demonstration of combining Python backend with modern frontend development.

VI. Motivation

- Teach practical cybersecurity concepts in a simple, interactive way so users can understand password strength, hashing, and security risks.
- Encourage stronger password habits by giving users instant, easy-to-understand feedback on their password choices.
- Provide a useful educational tool for students who are learning about cryptography, hashing algorithms, and secure password practices.
- Combine backend Python development with frontend interactivity, making this project an excellent full-stack learning experience.