Brainwave

MATRIX SOLUTIONS

منة الله اشرف على محمد

Task 2

Password Strength Checker Project Report

This report summarizes all the steps and enhancements implemented during the task.

---

1. Environment Setup

- **Python & Virtual Environment**
    - Installed Python 3.8+ and created a virtual environment using python3 -m venv venv.
    - Activated virtual environment: source venv/bin/activate (or . env\Scripts\Activate.ps1 on Windows).
- **Dependencies**
    - Installed zxcvbn-python for entropy calculation and feedback: pip install zxcvbn-python.
    - Prepared common_passwords.txt by downloading the top 10,000 common passwords from SecLists.

---

2. Core Function: password_strength.py

- **Loading Common Passwords**
    - Implemented load_common_passwords() to read common_passwords.txt into a Python set.
- **evaluate_password(password: str) -> dict**
    - Checks if password exists in common list; returns score=0 if so.
    - Uses zxcvbn() to obtain score (0–4), guesses, and guesses_log10.
    - Computes **entropy** in bits: entropy = guesses_log10 * log2(10).
    - Analyzes character categories: lower, upper, digits, symbols.
    - Classifies overall strength: Very Weak, Weak, Fair, or Strong based on score, length, and categories.
    - Returns a dictionary with all metrics and feedback suggestions.

- **Standalone CLI Test**
    - Added a __main__ block for interactive testing:

python password_strength.py

---

3. Command-Line Interface: cli.py

- **Argument Parsing** (argparse)
    - Positional argument password (optional).
    - --json flag to print full JSON result.
    - -o/--output FILE option to save output as text or JSON.
- **Workflow**
    1. Read password from CLI or prompt.
    2. Call evaluate_password().
    3. Format results as plain text or JSON.
    4. Print to console or write to file.
- **Usage Examples**
- python cli.py   # prompts for password
- python cli.py "P@ssw0rd" --json

python cli.py "Pass1234" -o report.txt

---

4. Web Frontend: Flask Application (app.py + templates/index.html)

- **Flask Setup**
    - Installed Flask: pip install Flask.
- **app.py**
- from flask import Flask, render_template, request
- from password_strength import evaluate_password
-
- app = Flask(__name__)

- @app.route('/', methods=['GET','POST'])
- def index():
-     result, pw = None, ''
-     if request.method == 'POST':
-         pw = request.form['password']
-         result = evaluate_password(pw)
-     return render_template('index.html', password=pw, result=result)
-
- if __name__ == '__main__':

app.run(debug=True)

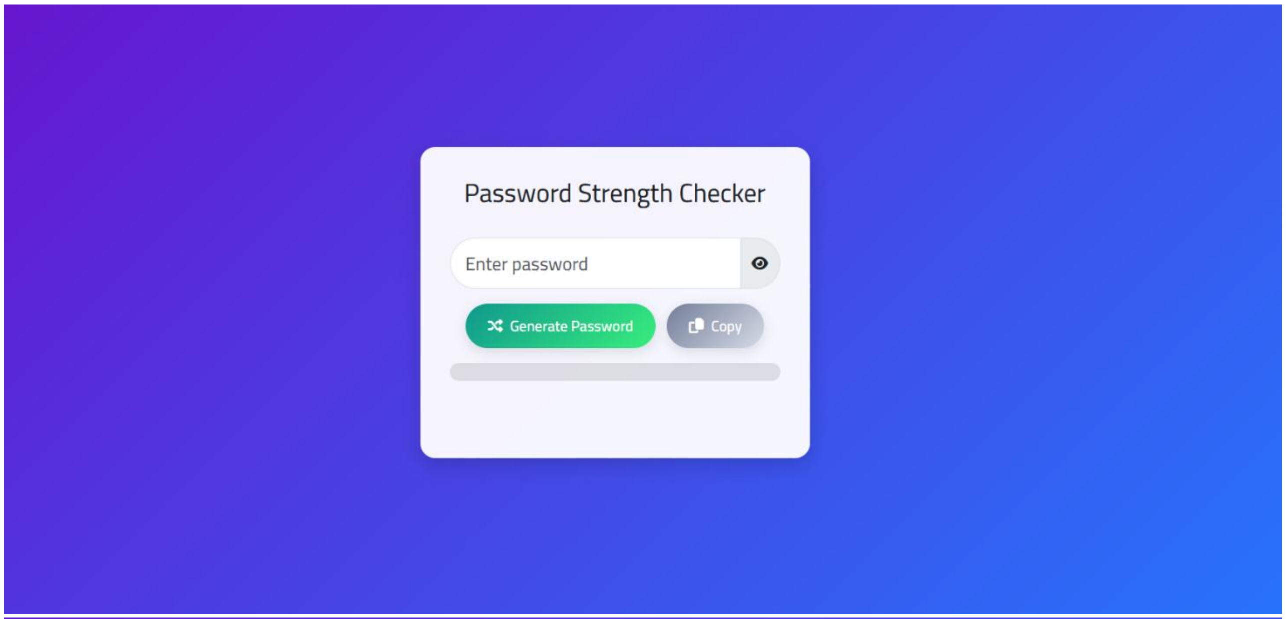
- **Template: templates/index.html**
  - Built using **Bootstrap 5** and **FontAwesome**.
  - Includes:
    - Password input with show/hide eye icon.
    - Buttons: **Check Strength**, **Generate Password**, **Copy**.
    - Real-time strength **progress bar** and label.
    - Result panel showing score, entropy, length, classes, warnings, and suggestions.

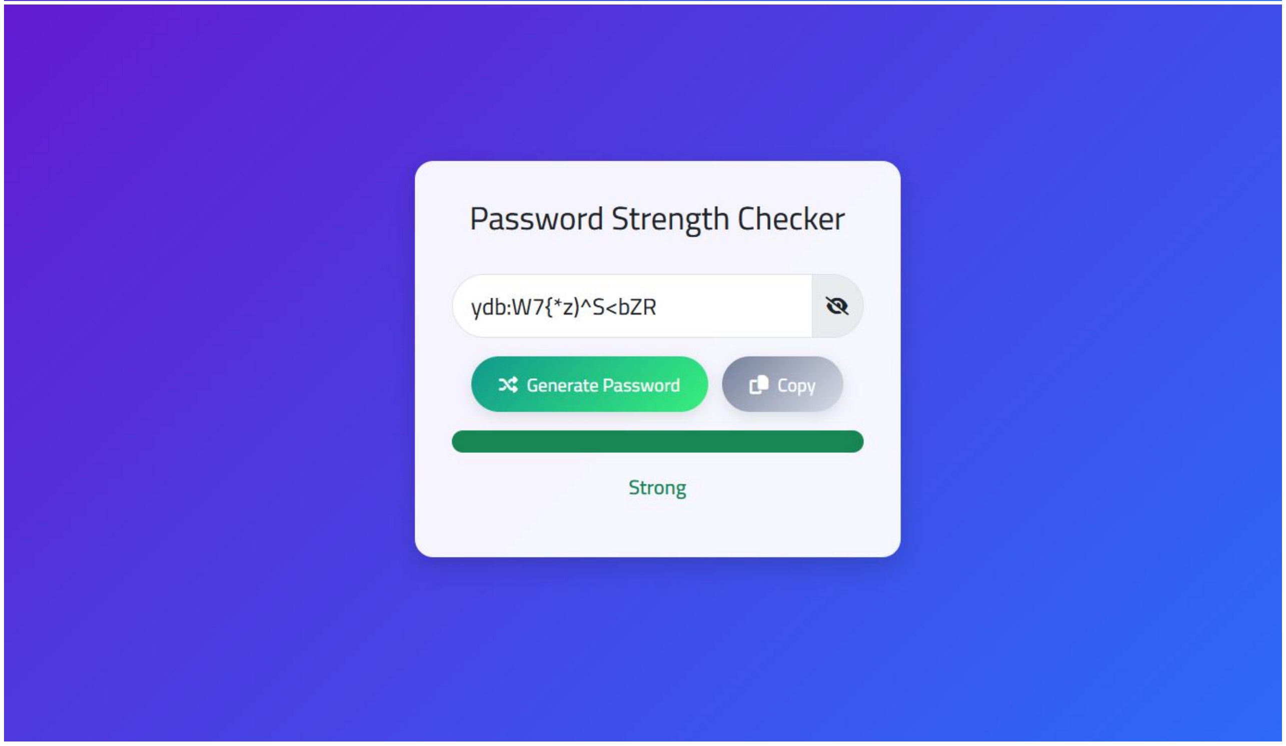
---

5. Design & UI Enhancements

- **Responsive Card Layout**
  - Centered on page with gradient background.
  - Rounded corners and subtle box-shadow.
- **Custom Buttons**
  - Circular pill shape with distinct gradients:
    - **Check**: purple→blue
    - **Generate**: teal→green

        - **Copy**: gray gradient
    - Hover opacity effect.
- **Eye Icon**
    - Placed inside an input-group-text on the right of the password field.
    - Toggles between fa-eye and fa-eye-slash on click.
- **Progress Bar & Text**
    - Updates in real-time on input.
    - Color-coded:
        - 0–1: danger (red)
        - 2: warning (orange)
        - 3: info (blue)
        - 4: success (green)
- **Result Section**
    - Displays after clicking **Check Strength**.
    - Bordered with color matching strength.
    - Suggestions wrap correctly for long text (word-break enabled).





ydb:W7{*z)^S<bZR

Strong