3rd Sem Mini Project Report on

Password Strength Assessment Tool

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted by:

Student Name	University Roll No.
Suryansh Chauhan	2418014

Under the Guidance of
MR NISHANT
(Assistant Professor)

Department of Computer Science and Engineering
Graphic Era Hill University
Dehradun, Uttarakhand 2024-25

Graphic Era Hill University

DEHRADUN • BHIMTAL • HALDWANI

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project report entitled "Password Strength Assessment Tool" in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering in the Department of Computer Science and Engineering of the Graphic Era Hill University, Dehradun shall be carried out by the undersigned under the supervision of Mr. Nishant (Assistant Professor), Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun .

Name	University Roll no
Suryansh Chauhan	**2418014**

The above-mentioned student shall be working under the supervision of the undersigned on the "Password Strength Assessment Tool"

Supervisor **Head of the Department**

Examination

Name of the Examiners: **Signature with Date**

1.

2.

Table of Contents

Chapter 1

Introduction and Problem Statement

1.1 Background Information :

The increasing reliance on digital communication necessitates robust password protection to prevent unauthorized access. Many users struggle to create strong, secure passwords due to a lack of knowledge or tools that assess password strength. This project aims to provide an AI-powered solution for evaluating the strength of user passwords and educating them about secure password practices.

1.2 Aim :

The Password Strength Assessment Tool's aim is to offer a user-friendly application for checking the strength of password. The tool lets users to find the strength of password and suggest how should the password should be created if its weak.

1.3 Objectives :

The goal of this project is to develop a password strength assessment tool that:

- Assess the strength of user passwords based on predefined criteria.
- Educate users on creating secure passwords.
- Detect commonly used, weak, or predictable password patterns.
- Suggest stronger alternatives to weak passwords.

1.4 Problem Statement :

Weak passwords are a primary reason for many cybersecurity breaches. Users often fail to understand what constitutes a strong password or lack the tools to evaluate them effectively. This project will bridge this gap by offering a tool that evaluates password strength and provides actionable recommendations, making the process intuitive and educational.

Chapter 2

Methodology

2.1 Requirement Analysis :

To create a robust password strength assessment tool, the following requirements are identified:

- Input: A password provided by the user.
- Output: A strength rating (e.g., weak, moderate, strong) and improvement suggestions.
- Criteria: Evaluate based on length, complexity, dictionary words, common patterns, and previously breached passwords.
- Accessibility: Intuitive GUI for easy interaction.
- Additional: Option to generate random secure passwords.

2.2 System Design :

Frontend (GUI):

- A user-friendly interface modern web technologies like **HTML, CSS, and JavaScript** for a browser-based tool.
- Fields for password input, results display, and suggestion outputs.
- Buttons for strength assessment and secure password generation.

Backend:

- A password strength algorithm using Python libraries such as **flask** for heuristic analysis or machine learning models for advanced assessments.
- Random password generator with customizable options (e.g., length, character types).

2.3 Implementation :

The tool can be implemented with the following steps:

- **Password Input Validation**: Ensure valid input and handle empty or null cases gracefully.
- **Strength Assessment**: Evaluate using predefined criteria (length, complexity, patterns)

or machine learning.

- **Feedback and Suggestions**: Provide actionable recommendations for creating stronger passwords.
- **GUI Development**: Use a web-based interface for user interactions.
- **Random Password Generation**: Generate strong passwords on demand.

2.4 Libraries Used :

- **flask**: For password strength estimation.
- **HTML/CSS/JavaScript**: For GUI design.

2.5 Testing and Debugging :

- Perform unit testing for individual functions like strength assessment, suggestion generation, and password validation.
- Conduct usability testing to refine the GUI and ensure intuitive design.
- Debug edge cases, such as very long passwords or passwords containing unexpected characters.

Chapter 3

Project Work Carried Out

3.1 Architectural Design :

- ***Frontend (GUI):***

- Input field for password entry.
- Buttons for assessing strength and generating random passwords.
- Result area displaying the strength score, visual indicators (e.g., color-coded bars), and suggestions for improvement.

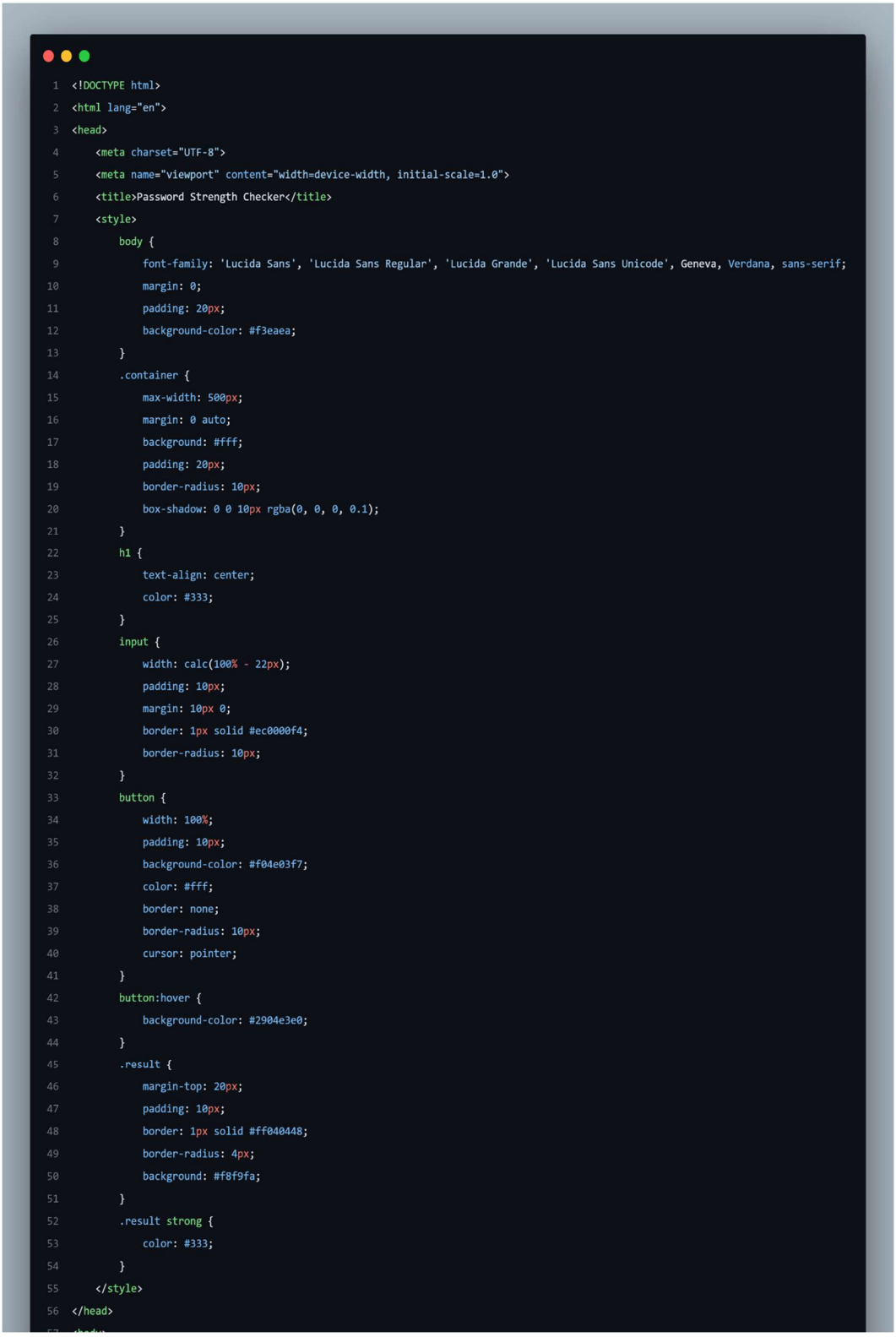
- ***Backend:***

- Strength assessment using flask or a similar library.
- Breach checking via API calls.
- Random password generator logic.

3.2 Implementation of Objectives :

- **Password Strength Assessment**: Use the backend logic to evaluate the entered password's security.
- **Feedback Mechanism**: Provide detailed feedback, including strength metrics and improvement tips.
- **Password Generation**: Allow users to generate secure, random passwords with adjustable criteria.
- **GUI Interaction**: Streamline user experience with intuitive layouts and responsive design.

3.3 Source Code :

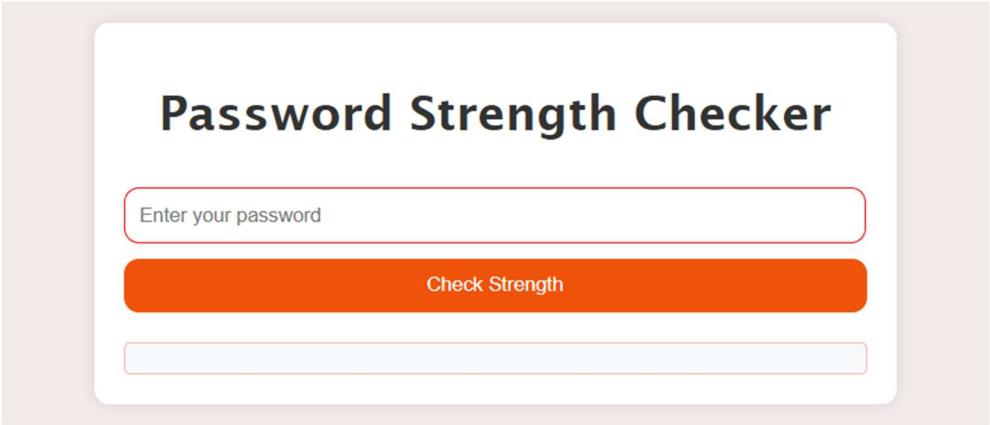


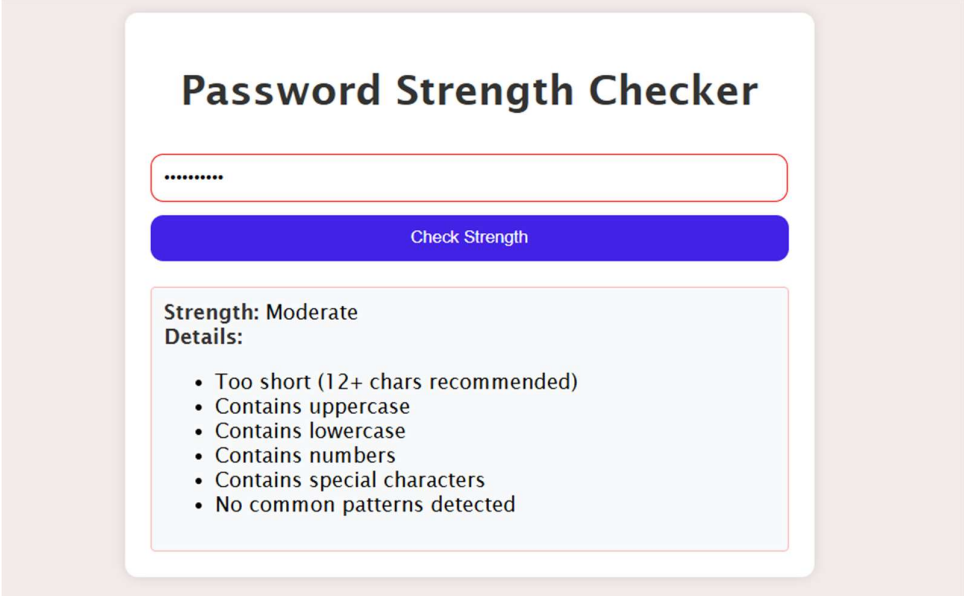
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Password Strength Checker</title>
    <style>
        body {
            font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f3eaea;
        }
        .container {
            max-width: 500px;
            margin: 0 auto;
            background: #fff;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            text-align: center;
            color: #333;
        }
        input {
            width: calc(100% - 22px);
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ec0000f4;
            border-radius: 10px;
        }
        button {
            width: 100%;
            padding: 10px;
            background-color: #f04e03f7;
            color: #fff;
            border: none;
            border-radius: 10px;
            cursor: pointer;
        }
        button:hover {
            background-color: #2904e3e0;
        }
        .result {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #ff040448;
            border-radius: 4px;
            background: #f8f9fa;
        }
        .result strong {
            color: #333;
        }
    </style>
</head>
```

```
<body>
    <div class="container">
        <h1>Password Strength Checker</h1>
        <input type="password" id="password" placeholder="Enter your password" />
        <button onclick="checkPassword()">Check Strength</button>
        <div class="result" id="result"></div>
    </div>

    <script>
        async function checkPassword() {
            const password = document.getElementById("password").value;
            const resultDiv = document.getElementById("result");

            if (!password) {
                resultDiv.innerHTML = "<strong>Please enter a password.</strong>";
                return;
            }

            const response = await fetch('/assess_password', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ password })
            });

            if (response.ok) {
                const data = await response.json();
                resultDiv.innerHTML = `
                    <strong>Strength:</strong> ${data.strength}<br>
                    <strong>Details:</strong><br>
                    <ul>
                        <li>${data.details.length}</li>
                        <li>${data.details.uppercase}</li>
                        <li>${data.details.lowercase}</li>
                        <li>${data.details.digit}</li>
                        <li>${data.details.special}</li>
                        <li>${data.details.common_patterns}</li>
                    </ul>
                `;
            } else {
                const error = await response.json();
                resultDiv.innerHTML = `<strong>Error:</strong> ${error.error}`;
            }
        }
    </script>
</body>
</html>

```

```python
from flask import Flask, render_template, request, jsonify
import re

app = Flask(__name__)

def calculate_password_strength(password):
    # Criteria for strength
    length = len(password) >= 12
    has_upper = bool(re.search(r'[A-Z]', password))
    has_lower = bool(re.search(r'[a-z]', password))
    has_digit = bool(re.search(r'\d', password))
    has_special = bool(re.search(r'[!@#$%^&*(),.?":{}|<>]', password))

    # Common patterns check
    common_patterns = ["123456", "password", "qwerty", "admin"]
    common = any(pattern in password.lower() for pattern in common_patterns)

    # Scoring
    score = sum([length, has_upper, has_lower, has_digit, has_special]) - int(common)

    if score < 3:
        strength = "Weak"
    elif score < 5:
        strength = "Moderate"
    else:
        strength = "Strong"

    return {
        "strength": strength,
        "details": {
            "length": "Good" if length else "Too short (12+ chars recommended)",
            "uppercase": "Contains uppercase" if has_upper else "Missing uppercase letters",
            "lowercase": "Contains lowercase" if has_lower else "Missing lowercase letters",
            "digit": "Contains numbers" if has_digit else "Missing numbers",
            "special": "Contains special characters" if has_special else "Missing special characters",
            "common_patterns": "Contains common patterns" if common else "No common patterns detected"
        }
    }

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/assess_password", methods=["POST"])
def assess_password():
    data = request.json
    password = data.get("password")
    if not password:
        return jsonify({"error": "Password is required"}), 400

    result = calculate_password_strength(password)
    return jsonify(result)

if __name__ == "__main__":
    app.run(debug=True)

```

3.4 Output :


Password Strength Checker
Enter your password
Check Strength


Password Strength Checker
Check Strength
Strength: Moderate
Details:
Too short (12+ chars recommended)
Contains uppercase
Contains lowercase
Contains numbers
Contains special characters
No common patterns detected

Chapter 4

Results and Discussion

- **Password Strength Assessment**:
  The AI-driven tool successfully evaluates passwords using predefined criteria such as length, character diversity, and the presence of common patterns. It uses a heuristic algorithm to provide a reliable strength rating (e.g., weak, moderate, strong).

- **AI-Enhanced Feedback**:
  The AI component identifies weaknesses like dictionary words, sequential patterns, and previously breached passwords. It provides detailed feedback on why a password is weak and actionable suggestions to improve it, such as adding complexity or avoiding predictable patterns.

- **User Interface**:
  The GUI is intuitive, allowing users to input passwords and view strength ratings in real time. Features include visual indicators (e.g., color-coded bars for strength levels) and a suggestion box for improvement tips. Users can also generate secure passwords with customizable parameters.

- **Password Generation**:
  The tool includes a robust random password generator that creates secure passwords meeting user-specified criteria (e.g., length, inclusion of special characters). Generated passwords are evaluated to ensure they meet strong security standards.

- **API Integration**:
  Integration with external services like the **Have I Been Pwned? API** checks whether the entered password has been exposed in known breaches, further enhancing the tool's effectiveness.

- **Performance**:
  The AI model evaluates passwords swiftly, offering real-time feedback even when analyzing complex password patterns or integrating breach-checking APIs.

Chapter 5

Conclusion and Future Work

5.1 Conclusion :

An AI-driven password strength assessment tool can significantly enhance cybersecurity by identifying weak passwords and guiding users towards better practices. By integrating machine learning, it adapts to evolving threats, providing a robust solution for both individual and enterprise security needs.

5.2 Future Work :

- ***Advanced Password Analysis***:
  Incorporate additional layers of analysis, such as entropy calculation and pattern recognition using advanced AI techniques like neural networks.

- ***Cross-Platform Support***:
  Expand the tool's availability by developing versions for mobile apps, browser extensions, and desktop platforms to provide seamless password evaluation across all devices.

- ***Multilingual Support***:
  Expand usability by supporting multiple languages for both the interface and feedback messages.

- ***Integration with Password Managers***:
  Partner with password management services to evaluate and strengthen user-stored passwords directly within their vaults.

- ***Real-Time Security Alerts***:
  Implement real-time notifications when a user's password is found in a breach database, offering immediate recommendations to replace it.

- ***User Customization***:
  Allow users to define their own password policies based on organizational requirements or personal preferences, such as minimum length, required character sets, or restrictions on reuse.