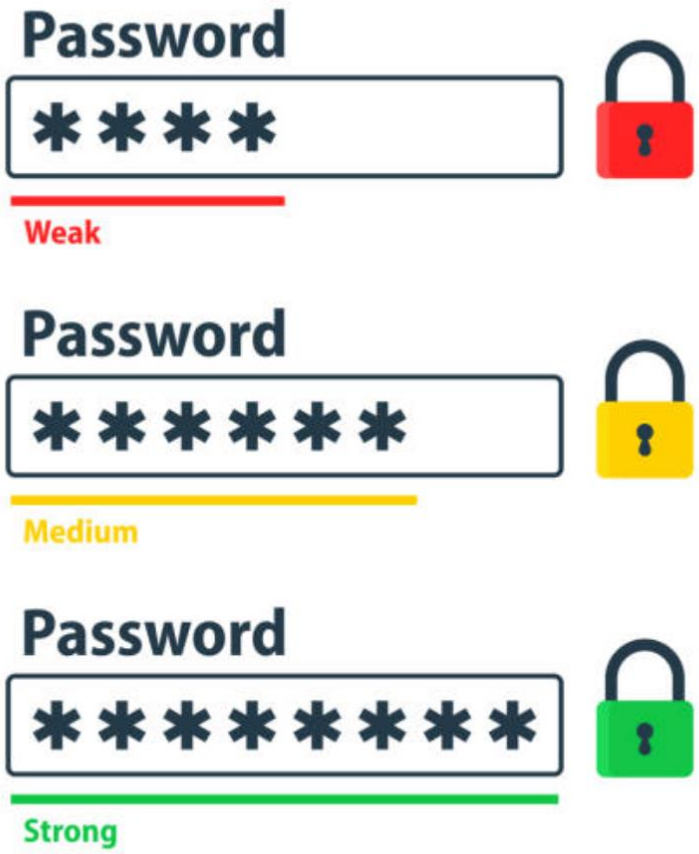
PROJECT

TOPIC: PASSWORD STRENGTH CHECKER



Student Name	**Aarna Manoj**
Course Name	**Introduction to Problem Solving and Programming**
Course Code	**CSE1021**
Faculty Name	**Ms. AVR Mayuri**
Institution	**VIT Bhopal**

1.Introduction

A Password Strength Checker checks how strong your password is, then shows ways to make it better. So you can avoid common traps like short phrases or easy guesses, this tool checks randomness levels and spots predictable sequences. While testing each entry against risky combos, it guides toward smarter choices - making hacking attempts way harder. Rules get applied on the fly so what works stays safe over time.

Objectives:

Check how strong a password is - fast and right.

Give clear tips that people can actually use to get better.

Hook up quick with sign-up steps or when resetting passwords instead.

2.Problem Statement

People usually creates bad passwords - like simple words or short combos - that hackers can guess easily. This effort focuses on creating a solid tool to test how strong a password really is.

Finds passwords that are easy to guess or unsafe.

Guesses how long it'd take to break a password or checks its randomness level.

Shows people how to make better passwords - using tips that actually help. While giving steps that are simple to follow along with.

3. Functional Requirements

These are the specific features and capabilities of the Password Strength Checker. The system includes three major functional modules, clear input/output behavior, and a logical user workflow.

Major Functional Modules:

- Password Input & Basic Validation Module
- Takes password input from the user.
- Checks minimum criteria (length, empty input, spaces, etc.).
- Output: "valid/invalid input" message before further checks.

Strength Evaluation Module

- Evaluates password using rules such as:

✧ Length
✧ Use of uppercase, lowercase, numbers, and special characters
✧ Repeated characters or common patterns

- Optionally checks against a common-password list.
- Outputs a score (0–4) and a strength label (Very Weak → Very Strong).

Feedback & Suggestions Module

- Generates user-friendly improvement tips.
- Examples:

✧ "Add more characters"
✧ "Use at least one special character"
✧ "Avoid common words"

- Outputs a list of suggestions based on weaknesses found.

Input / Output Structure

Input:

- A password string entered by the user.

Outputs:

- Strength score (numeric)
- Strength label (text)
- Feedback messages
- (Optional) entropy or crack-time estimate

Logical Workflow

- User enters a password into the system.
- System performs quick local checks.
- System evaluates the password strength using defined rules.
- System assigns a strength score and label.
- System displays feedback and suggestions to the user.
- User modifies password (if needed) based on suggestions.

4. Non-Functional Requirements

Since the Password Strength Checker is a client-side utility, these requirements focus heavily on speed, user experience, and resource efficiency.

• Usability: Clarity of Feedback, Responsiveness, Accessibility
• Reliability: Deterministic checks and clear messages
• Maintainability: Modular code structure, Technology Stack
• Performance: Instant checks on input, Resource Efficiency
• Security: No passwords are stored or logged

5. System Architecture

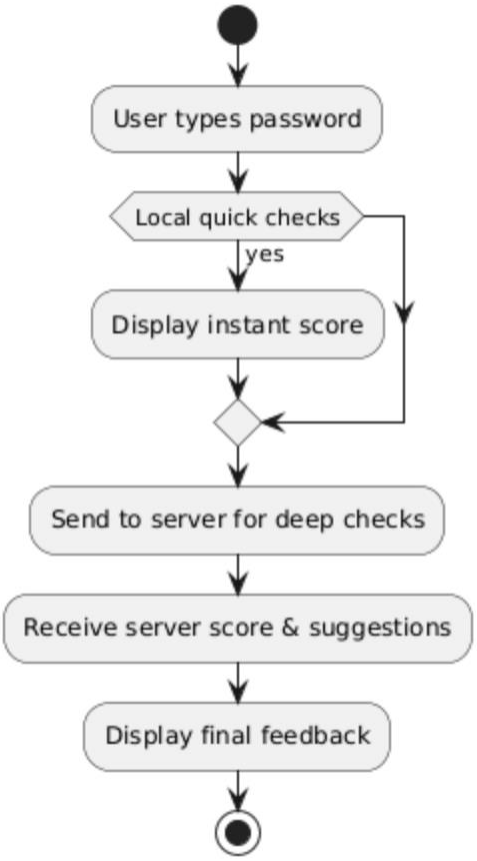
A simple and robust architecture consists of:

Client (Frontend): HTML plus CSS with JS tool runs fast checks - shows tips right away using live feedback loops.

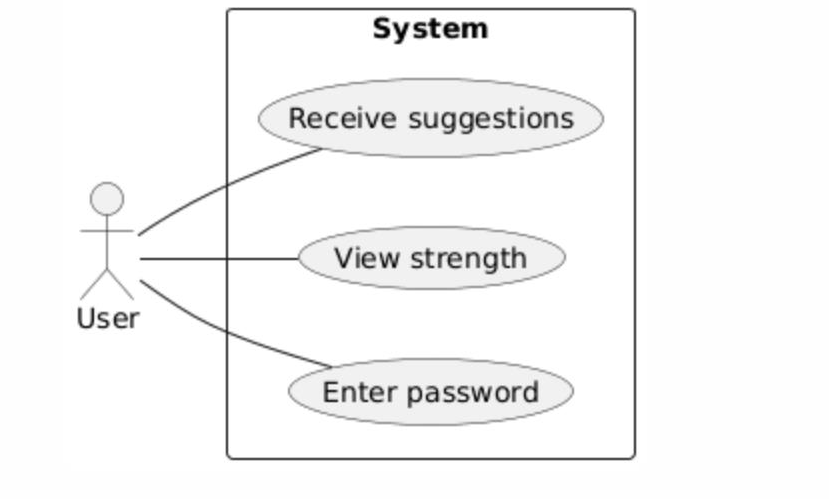
Server (Backend API): A protected REST interface checking passwords more deeply - like scanning blacklists, measuring guess difficulty, or spotting random patterns.

Storage (optional): A tiny database - say, PostgreSQL or Redis - to keep blocked lists, track request limits, or save anonymous usage data.

6. Workflow Diagram



7. Design Diagrams



8. Implementation Details

The project uses Python functions to separate checks (length, uppercase, lowercase, digits, special chars). A scoring system aggregates the results and determines the final rating. Suggestions are generated based on missing criteria.

Tech Stack:

Frontend: HTML, CSS, basic JavaScript

Backend: Python (FastAPI)

Storage: Not required for basic version

Libraries: zxcvbn for strength calculation

9. Code (main.py)

See the included main.py in the GitHub package. Core logic checks for length, character classes, and computes a score.

10. Testing Approach

- Unit tests: For entropy calculation, blacklist checks, and feedback generation.
- Integration tests: API endpoint tests (valid/invalid payloads, rate-limiting behavior).
- Performance tests: Simulate concurrent requests; ensure median response under 200 ms.
- Security tests: Ensure passwords are not logged; check for injection vulnerabilities; run static analysis tools.
- Usability tests: Quick user-testing sessions (5–10 users) to ensure feedback is understood and actionable.

11.Screenshots

```
enter ur password: aarna manoj
password: aarna manoj
{'strength': 'Very Weak', 'score': 2, 'feedback': 'Missing: Uppercase letter (A-Z), Digit (0-9), Special character (!@#$...)'}
```

```
enter ur password: aarna.25MIM.10050
password: aarna.25MIM.10050
{'strength': 'Moderate', 'score': 7, 'feedback': 'Password meets all complexity requirements.'}
```

12. Challenges Faced

- Balancing **usability** (giving helpful suggestions) with **security** (not exposing evaluation internals which may help attackers).
- Creating accurate crack-time estimates without overclaiming precision.
- Ensuring no raw passwords are accidentally logged by 3rd-party libs or during error handling.

13. Learnings & Future Enhancements

- Learned how password strength is measured using patterns and entropy.
- Understood how libraries like **zxcvbn** simplify complexity.
- Gained experience in building an API and connecting frontend–backend.
- Improved understanding of safe handling of passwords (n## 13. Future Enhancements
- Add a more detailed strength meter with animations.
- Include passphrase suggestions.
- Add dark mode UI.
- Offline version using only client-side JS.
- Optional feature: Check password against a larger breached password list securely.

14. References

- Course material
- OWASP Password Security Guidelines
- Dropbox zxcvbn project