Project Report: Password Strength Analyzer (Streamlit GUI)

Developer: Vishal Prajapati
Course: B.Tech CSE (AI & ML)
Institute: Krishna Institute of Engineering and Technology, Ghaziabad

---

Objective:

To develop a secure web-based Password Analyzer that evaluates user-entered passwords using rule-based validation and entropy-based strength checking. It also checks whether the password is present in known public data breaches.

Technologies Used:

- **Python** – Core programming logic
- **Streamlit** – Web UI framework
- **zxcvbn** – Password strength analyzer by Dropbox
- **requests** – To fetch breach data using API
- **re (regex)** – Pattern matching for rule enforcement
- **HaveIBeenPwned API** – For checking if password was exposed in breaches

---

Features:

1. Password Strength Analysis:

- Uses the zxcvbn library to calculate entropy and strength score (0 to 4)
- Strength score converted to percentage: `(score + 1) * 10`

2. Rule-Based Validation (R1 to R9):

Rule ID	Description
R1	Minimum 8 characters required
R2	At least one uppercase letter
R3	At least one lowercase letter
R4	At least one digit
R5	At least one special character (@#$%^&*)
R6	Avoid common patterns like ‘123’, ‘qwerty’, etc.
R7	Avoid repeating characters like ‘aaa’, ‘111’
R8	Avoid sequential characters like ‘1234’, ‘abcd’
R9	Do not include your own name/email in password

- Passed rules contribute additional percentage to total strength: `(passed_rules / 9) * 50`

3. Breach Check:

- SHA-1 hash of password generated
- Only first 5 characters of hash sent to the HaveIBeenPwned API
- If suffix found in response, the password is flagged as breached

4. Streamlit Web UI:

- Clean, browser-based interface
- Password and username/email inputs
- Strength shown as progress bar and percentage
- Detailed rule-based suggestions
- API breach result shown

Output Example:

Input Password: `Vishal@123`

- **Strength Score:** 78.33%
- **Suggestions:** Avoid common patterns like '123', 'admin'
- **Breach Status:** Found in public breaches over 2,900 times

---

How to Run:

1. Install dependencies:

```
pip install streamlit zxcvbn requests
```

2. Run the app:

```
streamlit run password_app.py
```

3. Browser will open at `http://localhost:8501/`

Benefits:

- Secure password feedback in real-time
- Breach verification using real-world leak database
- Easy-to-use, modern web interface
- Doesn't store or send full password data (privacy-safe)

Future Improvements:

- Add password generator tab
- Export report as PDF or JSON
- Deploy to cloud (Streamlit Cloud / Render / Vercel)
- Add dark mode and mobile responsiveness

Conclusion:

This tool is a practical and educational example of how password policies, entropy, and threat intelligence can be combined into one tool. It helps users create secure passwords and alerts them of known breaches, all through a fast and friendly UI.