PYTHON: VIT-YARTHI PROJECT

TOPIC- PASSWORD STRENGTH CHECKER

PRESENTED BY:

PEREPU SOUMYA
25BCE10126
(2025-26)

HTTPS://GITHUB.COM/PEREPU-8/PEREPUSOUMYA/BLOB/MAIN/PASSWORDCHECK.PY

HTTPS://VITYARTHI.COM/PLAY-COURSE/PYTHON-ESSENTIALS-2/417

INTRODUCTION

IN THE DIGITAL ERA, PASSWORDS ACT AS THE FIRST LINE OF DEFENSE FOR PROTECTING PERSONAL DATA, ONLINE ACCOUNTS, AND SENSITIVE INFORMATION. HOWEVER, MOST USERS STILL RELY ON WEAK, PREDICTABLE, OR REPETITIVE PASSWORDS, MAKING THEM EASY TARGETS FOR CYBER-ATTACKS SUCH AS BRUTE-FORCE ATTEMPTS, CREDENTIAL THEFT, AND UNAUTHORIZED ACCESS. AS ONLINE THREATS CONTINUE TO EVOLVE, ENSURING PASSWORD STRENGTH IS NO LONGER OPTIONAL—IT IS A FUNDAMENTAL REQUIREMENT FOR MAINTAINING DIGITAL SECURITY.

THIS PROJECT, PASSWORD STRENGTH CHECKER, AIMS TO ADDRESS THIS GROWING CONCERN BY PROVIDING A SIMPLE, EFFICIENT, AND INTELLIGENT PYTHON-BASED TOOL THAT EVALUATES THE STRENGTH OF ANY PASSWORD ENTERED BY A USER. IT ANALYZES MULTIPLE KEY PARAMETERS SUCH AS LENGTH, CHARACTER DIVERSITY, NUMERICAL INCLUSION, USE OF SPECIAL SYMBOLS, AND REPETITION PATTERNS. BASED ON THESE CHECKS, THE SYSTEM CATEGORIZES THE PASSWORD INTO LEVELS (WEAK, MEDIUM, OR STRONG) AND GIVES CLEAR SUGGESTIONS FOR IMPROVEMENT.

THROUGH THIS PROJECT, USERS CAN UNDERSTAND THE WEAKNESSES IN THEIR PASSWORDS AND LEARN HOW TO CREATE STRONGER, MORE SECURE CREDENTIALS. IT ALSO PROMOTES AWARENESS ABOUT CYBERSECURITY AND ENCOURAGES SAFE DIGITAL PRACTICES. THE TOOL IS LIGHTWEIGHT, USER-FRIENDLY, AND DESIGNED TO HELP INDIVIDUALS ENHANCE THE SECURITY OF THEIR ONLINE IDENTITIES.

PROBLEM STATEMENT

DESPITE INCREASING AWARENESS, A LARGE NUMBER OF USERS CONTINUE TO RELY ON SIMPLE PASSWORDS DUE TO CONVENIENCE, LACK OF KNOWLEDGE, OR POOR SECURITY HABITS. THIS CREATES SIGNIFICANT VULNERABILITIES ACROSS PERSONAL, ACADEMIC, AND ORGANIZATIONAL SYSTEMS. THE PROBLEM BECOMES MORE CRITICAL BECAUSE USERS OFTEN DO NOT HAVE AN IMMEDIATE WAY TO EVALUATE THE STRENGTH OF THEIR PASSWORDS OR IDENTIFY WHICH ELEMENTS ARE MISSING—SUCH AS LENGTH, DIGITS, UPPERCASE/LOWERCASE LETTERS, OR SPECIAL CHARACTERS. WITHOUT CLEAR FEEDBACK, THEY UNKNOWINGLY CONTINUE USING PASSWORDS THAT CAN BE CRACKED WITHIN SECONDS. THIS PROJECT AIMS TO ADDRESS THIS ISSUE BY PROVIDING A SIMPLE YET EFFECTIVE PYTHON-BASED TOOL THAT ANALYZES PASSWORD STRENGTH AND GIVES INSTANT, ACTIONABLE FEEDBACK. IT HELPS USERS UNDERSTAND THE WEAKNESSES IN THEIR PASSWORDS AND ENCOURAGES THE CREATION OF SECURE AND ROBUST CREDENTIALS, ULTIMATELY PROMOTING BETTER CYBERSECURITY PRACTICES.THIS PROJECT SOLVES THAT PROBLEM BY EVALUATING THE STRENGTH OF PASSWORDS AND GUIDING USERS TO CREATE MORE SECURE ONES

01. Users often create weak or predictable passwords due to lack of awareness, making their accounts vulnerable to cyber-attacks and unauthorized access.

02. There is no simple tool that instantly evaluates password strength and provides clear feedback on how to improve it.

03. A system is required that analyzes password parameters (length, character variety, numbers, symbols) and classifies strength while giving actionable suggestions.

FUNCTIONALREQUIREMENTS

PASSWORD INPUT MODULE:

- THE SYSTEM MUST ALLOW THE USER TO ENTER ANY PASSWORD FOR EVALUATION.
- IT SHOULD ACCEPT ALL CHARACTER TYPES (LETTERS, NUMBERS, SPECIAL SYMBOLS.)

PASSWORD STRENGTH ANALYSIS:
THE SYSTEM MUST EVALUATE THE PASSWORD BASED ON:

- LENGTH
- UPPERCASE LETTERS
- LOWERCASE LETTERS
- DIGITS
- SPECIAL CHARACTERS
- REPETITION OR PREDICTABLE PATTERNS

STRENGTH CLASSIFICATION

- THE SYSTEM MUST CATEGORIZE THE PASSWORD AS:
  - WEAK
  - MEDIUM
  - STRONG

FEEDBACK & SUGGESTIONS MODULE

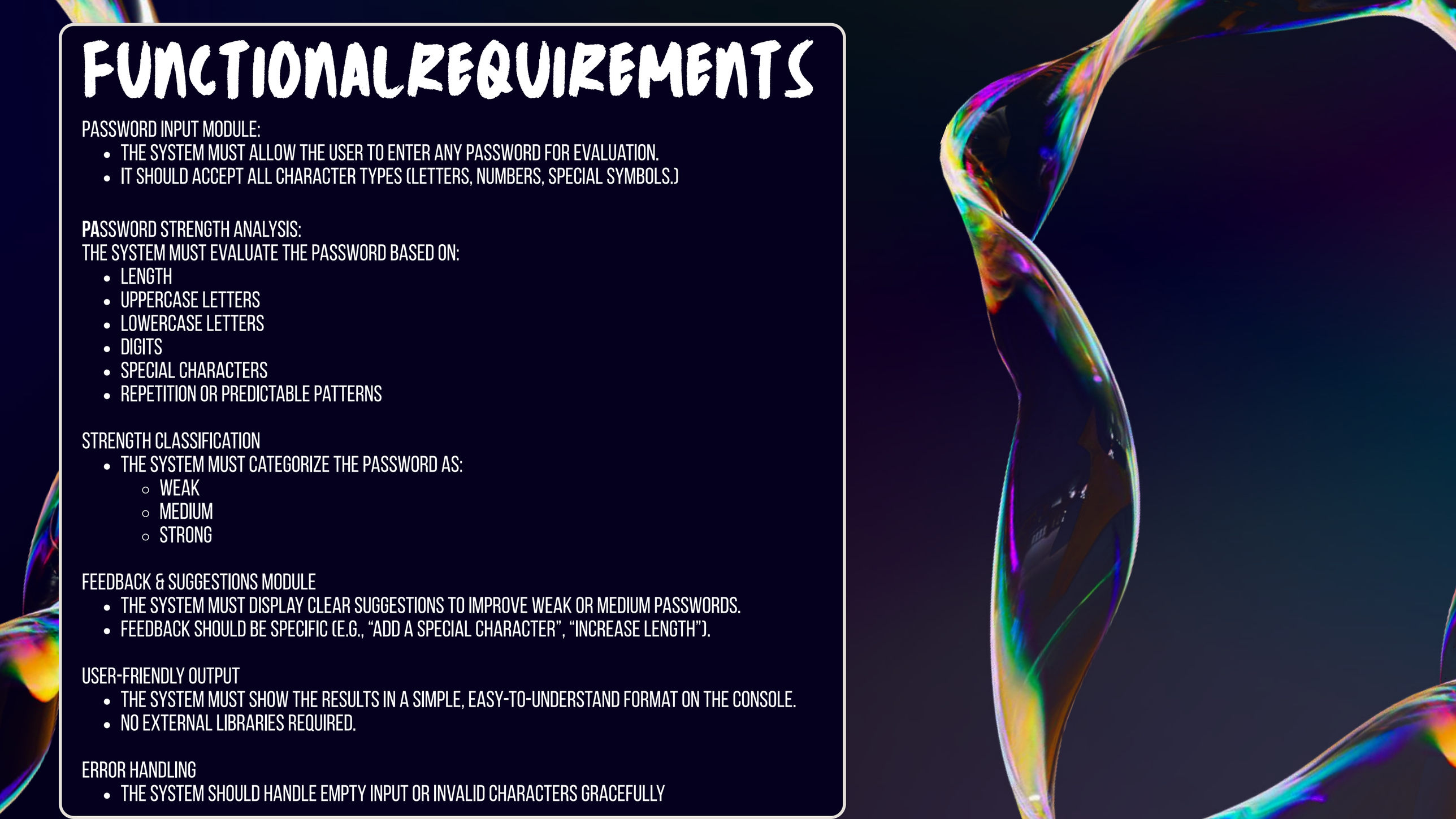
- THE SYSTEM MUST DISPLAY CLEAR SUGGESTIONS TO IMPROVE WEAK OR MEDIUM PASSWORDS.
- FEEDBACK SHOULD BE SPECIFIC (E.G., “ADD A SPECIAL CHARACTER”, “INCREASE LENGTH”).

USER-FRIENDLY OUTPUT

- THE SYSTEM MUST SHOW THE RESULTS IN A SIMPLE, EASY-TO-UNDERSTAND FORMAT ON THE CONSOLE.
- NO EXTERNAL LIBRARIES REQUIRED.

ERROR HANDLING

- THE SYSTEM SHOULD HANDLE EMPTY INPUT OR INVALID CHARACTERS GRACEFULLY



NON-FUNCTIONAL REQUIREMENTS

01

PERFORMANCE

- THE SYSTEM MUST EVALUATE THE PASSWORD AND GENERATE RESULTS INSTANTLY WITHOUT NOTICEABLE DELAY.
- SHOULD RUN SMOOTHLY ON LOW-SPEC MACHINES SINCE NO HEAVY LIBRARIES ARE USED.

02

USABILITY

- THE OUTPUT MUST BE SIMPLE AND EASY FOR BEGINNERS TO UNDERSTAND.
- FEEDBACK SHOULD BE READABLE, CLEAR, AND FREE FROM TECHNICAL JARGON.

03

MAINTAINABILITY

- CODE MUST BE MODULAR AND EASY TO UPDATE OR EXTEND (E.G., ADDING MORE PASSWORD RULES).
- COMMENTS AND CLEAN STRUCTURE SHOULD SUPPORT FUTURE ENHANCEMENTS

04

SECURITY AND RELIABILITY

- THE TOOL MUST NOT STORE OR LOG THE USER'S PASSWORD AT ANY STAGE.
- ALL EVALUATION MUST HAPPEN LOCALLY, ENSURING USER PRIVACY
- THE SYSTEM MUST PROVIDE CONSISTENT RESULTS FOR THE SAME PASSWORD INPUT

SYSTEM ARCHITECTURE DIAGRAM :

I. THE PASSWORD STRENGTH CHECKER IS A LIGHTWEIGHT, MODULAR PYTHON APPLICATION THAT EVALUATES PASSWORD STRENGTH LOCALLY AND PROVIDES IMMEDIATE, ACTIONABLE FEEDBACK. THE ARCHITECTURE FOCUSES ON SEPARATION OF CONCERNS: INPUT HANDLING, PASSWORD ANALYSIS, FEEDBACK GENERATION, AND PRESENTATION. THE SYSTEM IS DESIGNED FOR EASY INTEGRATION INTO OTHER APPLICATIONS AND FOR SIMPLE CLI USAGE.

II. HIGH-LEVEL COMPONENTS

1. USER INTERFACE (UI)
2. INPUT VALIDATION MODULE
3. RULES & SCORING MODULE.
4. FEEDBACK GENERATOR
5. OUTPUT FORMATTED

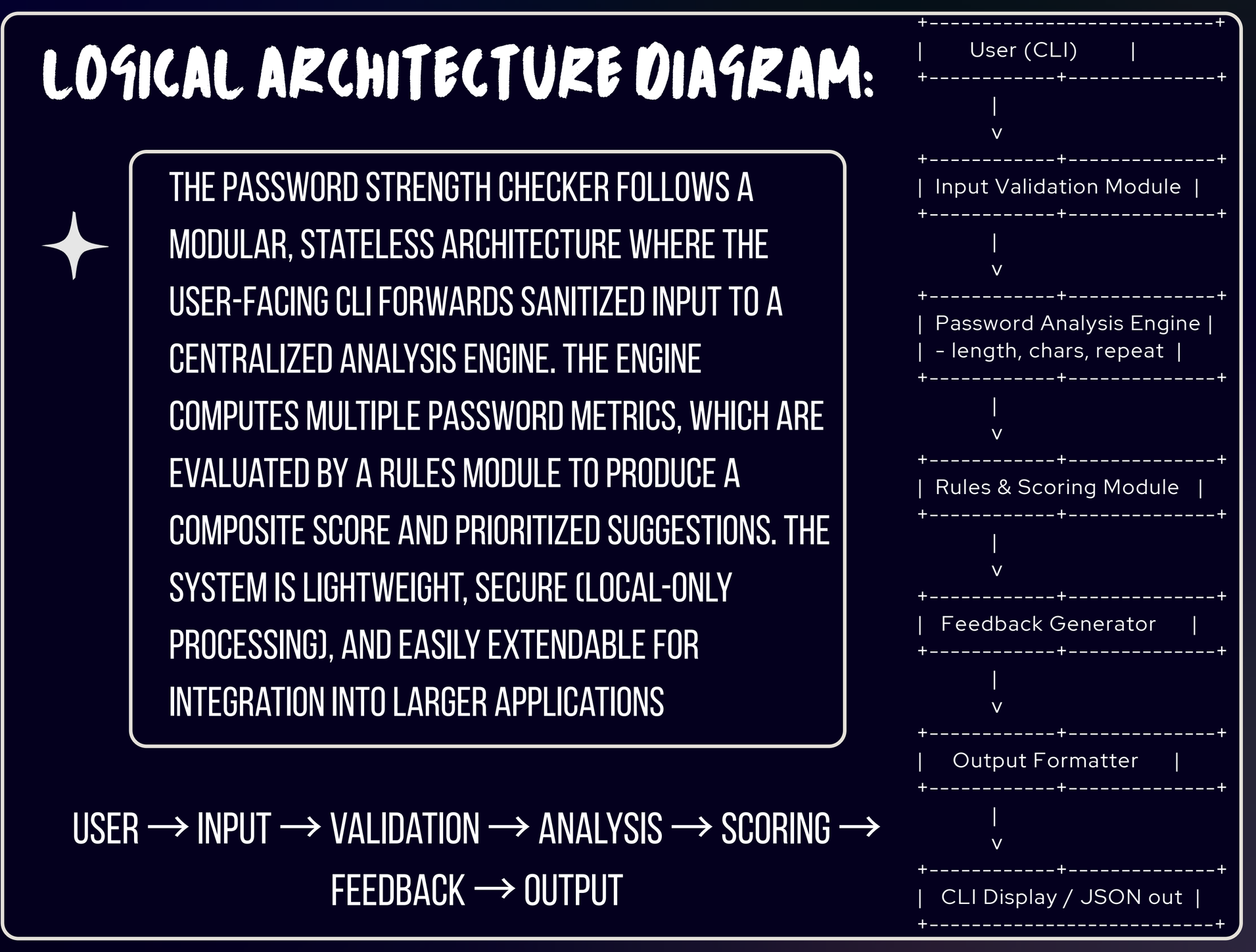
III. COMPONENT INTERACTION / DATA FLOW (STEP-BY-STEP):

- USER ENTERS PASSWORD VIA CLI (UI).
- INPUT VALIDATION MODULE CHECKS INPUT AND PASSES SANITIZED PASSWORD TO ANALYSIS ENGINE.
- PASSWORD ANALYSIS ENGINE RUNS A SET OF RULE CHECKS AND COMPUTES RAW METRICS.
- RULES & SCORING MODULE CONVERTS METRICS INTO A COMPOSITE SCORE AND CLASSIFICATION (WEAK/MEDIUM/STRONG).
- FEEDBACK GENERATOR CREATES ACTIONABLE SUGGESTIONS BASED ON FAILED/LOW-SCORING RULES.
- OUTPUT FORMATTER PRINTS A FINAL REPORT TO THE CLI (SCORE, CLASSIFICATION, SUGGESTIONS).
- OPTIONAL: RETURN A JSON OBJECT FOR PROGRAMMATIC CONSUMPTION WHEN USED AS A LIBRARY.

LOGICAL ARCHITECTURE DIAGRAM:

THE PASSWORD STRENGTH CHECKER FOLLOWS A MODULAR, STATELESS ARCHITECTURE WHERE THE USER-FACING CLI FORWARDS SANITIZED INPUT TO A CENTRALIZED ANALYSIS ENGINE. THE ENGINE COMPUTES MULTIPLE PASSWORD METRICS, WHICH ARE EVALUATED BY A RULES MODULE TO PRODUCE A COMPOSITE SCORE AND PRIORITIZED SUGGESTIONS. THE SYSTEM IS LIGHTWEIGHT, SECURE (LOCAL-ONLY PROCESSING), AND EASILY EXTENDABLE FOR INTEGRATION INTO LARGER APPLICATIONS

USER → INPUT → VALIDATION → ANALYSIS → SCORING → FEEDBACK → OUTPUT



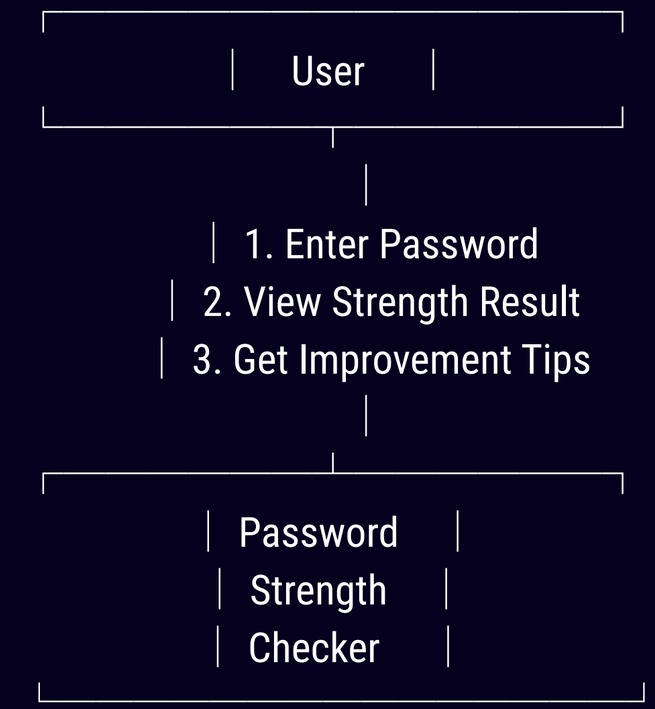
DESIGN DIAGRAM

01. USE CASE DIAGRAM:
Shows how the user interacts with the password checker — entering a password, receiving strength report, and getting suggestions.

02. WORKFLOW DIAGRAM:
Represents the step-by-step flow from password input to generating output results

03. SEQUENCE DIAGRAM:
Displays the order of interactions between the CLI, validator, analyzer, scoring engine, and feedback generator.

USE CASE DIAGRAM:



WORKFLOW DIAGRAM:

Start → Input Password → Validate Length → Check Character Types
↓
Display Result ← Calculate Score ← Check Common Patterns

DESIGN DECISIONS AND RATIONALE:

01

DECISION: SCOORING SYSTEM:

RATIONALE: PROVIDES GRANULAR FEEDBACK INSTEAD OF BINARY PASS/FAIL

02

DECISION: MULTIPLE CHECKS:

RATIONALE: COMPREHENSIVE SECURITY ASSESSMENT COVERING ALL VULNERABILITY TYPE

03

DECISION: NO PASSWORD CHANGE:

RATIONALE: PRIVACY PROTECTION - PASSWORDS AREN'T SAVED OR TRANSMITTED

04

DECISION: REAL-TIME FEEDBACK:

RATIONALE: IMMEDIATE USER GUIDANCE FOR BETTER PASSWORD CREATION

IMPLEMENTATION DETAILS:

- Implemented in Python using built-in functions and the string module.
- Code organized into logical blocks for validation, analysis, scoring, and feedback.
- Uses condition checks (any(), islower(), isdigit(), etc.) to evaluate password structure.
- Final output printed to console as strength level + improvement tips.

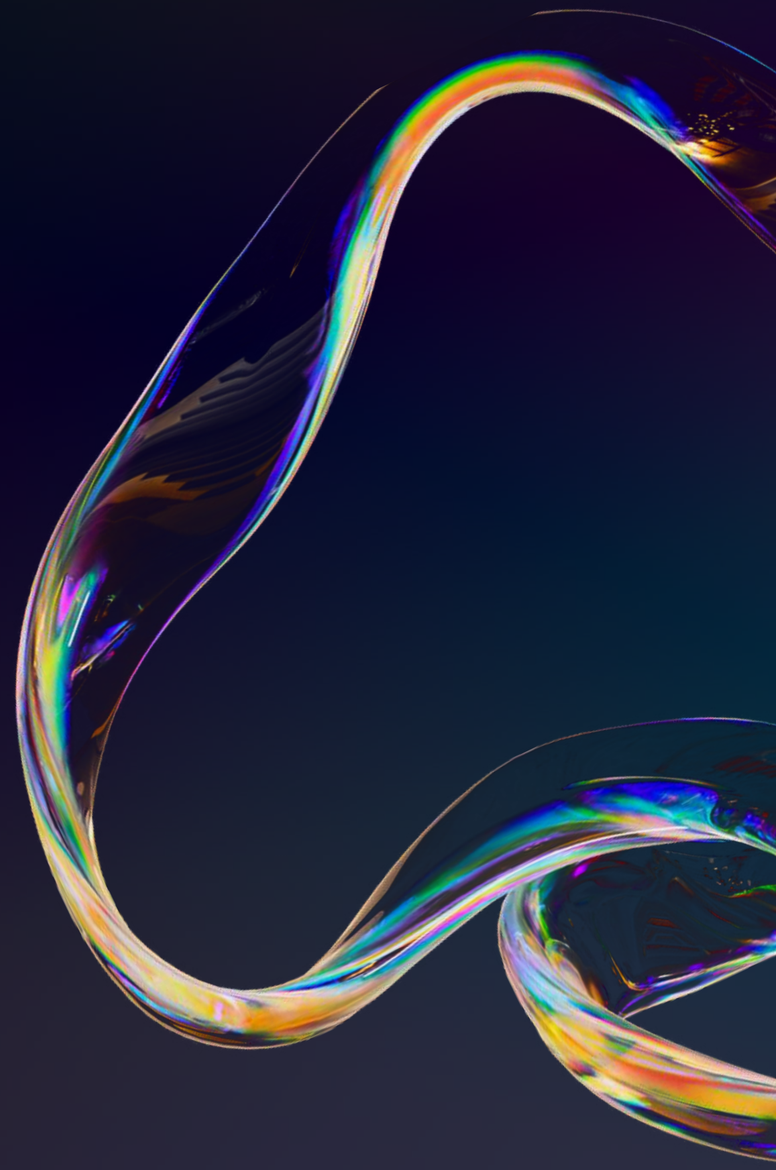
```python
from string import punctuation as spl
def passwordchecker(password):
    temp=0
    if(len(password)>=8):
        temp+=1
    else:
        print("PASSWORD TOO SHORT. SHOULD CONTAIN ATLEAST 8 CHARACTERS.")
    if(any(chr.isdigit() for chr in password)):
        temp+=1
    else:
        print("PASSWORD SHOULD CONTAIN ATLEAST ONE DIGIT.")
    if(any(chr.islower() for chr in password)):
        temp+=1
    else:
        print("PASSWORD SHOULD CONTAIN ATLEAST ONE LOWERCASE LETTER.")
    if(any(chr.isupper() for chr in password)):
        temp+=1
    else:
        print("PASSWORD SHOULD CONTAIN ATLEAST ONE UPPERCASE LETTER.")
    if(any( ((chr in spl)for chr in password))):
        temp+=1
    else:
        print("PASSWORD SHOULD CONTAIN ATLEAST ONE SPECIAL CHARACTER.")
    if temp>=5:
        return "STRONG 🔥"
    elif temp>=3:
        return"MEDIUM 😐(CAN IMPROVE)"
    else:
        return"WEAK 🚫"

Password=input("ENTER A PASSWORD: ")
data=passwordchecker(Password)
```

SCREENSHOTS AND RESLUTS:

```python
from string import punctuation as spl
def passwordchecker(password):
    temp=0
    if(len(password)>=8):
        temp+=1
    else:
        print("PASSWORD TOO SHORT. SHOULD CONTAIN ATLEAST 8 CHARACTE
    if(any(chr.isdigit() for chr in password)):
        temp+=1
    else:
        print("PASSWORD SHOULD CONTAIN ATLEAST ONE DIGIT.")
    if(any(chr.islower() for chr in password)):
        temp+=1
    else:
        print("PASSWORD SHOULD CONTAIN ATLEAST ONE LOWERCASE LETTER.
    if(any(chr.isupper() for chr in password)):
        temp+=1
    else:
        print("PASSWORD SHOULD CONTAIN ATLEAST ONE UPPERCASE LETTER.
    if(any( ((chr in spl)for chr in password))):
        temp+=1
    else:
        print("PASSWORD SHOULD CONTAIN ATLEAST ONE SPECIAL CHARACTER
    if temp>=5:
        return "STRONG 🔥"
    elif temp>=3:
        return"MEDIUM 😐(CAN IMPROVE)"
    else:
        return"WEAK 🚫"

Password=input("ENTER A PASSWORD: ")
data=passwordchecker(Password)
```



OUTPUT AND TESTING APPROACH:

```
ENTER A PASSWORD: 2812
PASSWORD TOO SHORT. SHOULD CONTAIN ATLEAST 8 CHARACTERS.
PASSWORD SHOULD CONTAIN ATLEAST ONE LOWERCASE LETTER.
PASSWORD SHOULD CONTAIN ATLEAST ONE UPPERCASE LETTER.
PASSWORD SHOULD CONTAIN ATLEAST ONE SPECIAL CHARACTER.
PASSWORD: WEAK 🚫
PS C:\Users\perep\OneDrive\Desktop\Perepusoumya> & C:/User
asswordcheck.py
ENTER A PASSWORD: Soumya@2812
PASSWORD: STRONG 🔥
PS C:\Users\perep\OneDrive\Desktop\Perepusoumya> & C:/User
asswordcheck.py
ENTER A PASSWORD: Soumya2812
PASSWORD SHOULD CONTAIN ATLEAST ONE SPECIAL CHARACTER.
PASSWORD: MEDIUM 😊(CAN IMPROVE)
PS C:\Users\perep\OneDrive\Desktop\Perepusoumya>
```

CHALLENGES FACED:

1. BALANCING STRICTNESS VS USABILITY - TOO STRICT MIGHT FRUSTRATE USERS
2. DETECTING SOPHISTICATED PATTERNS BEYOND SIMPLE SEQUENCES
3. CREATING MEANINGFUL FEEDBACK THAT USERS CAN ACTUALLY USE
4. HANDLING SPECIAL CHARACTER VARIATIONS ACROSS KEYBOARD LAYOUTS

LEARNING AND KEY TAKEAWAYS:

- UNDERSTANDING OF MODERN PASSWORD SECURITY PRINCIPLES
- IMPORTANCE OF USER EDUCATION IN CYBERSECURITY
- STRING MANIPULATION TECHNIQUES IN PYTHON
- ALGORITHM DESIGN FOR MULTI-CRITERIA EVALUATION SYSTEMS

FUTURE ENHANCEMENTS:

- DATABASE INTEGRATION TO CHECK AGAINST PREVIOUSLY BREACHED PASSWORDS
- GUI IMPLEMENTATION WITH VISUAL STRENGTH METER
- PASSWORD GENERATOR WITH CUSTOMIZABLE REQUIREMENTS
- API DEVELOPMENT FOR INTEGRATION WITH OTHER APPLICATIONS
- MULTI-LANGUAGE SUPPORT FOR INTERNATIONAL USERS

REFERENCES:

- NIST SPECIAL PUBLICATION 800-63B - DIGITAL IDENTITY GUIDELINES
- OWASP AUTHENTICATION CHEAT SHEET
- PYTHON STRING MODULE DOCUMENTATION
- CYBERSECURITY BEST PRACTICES FROM CISA
- GEEKSFORGEEKS & W3SCHOOLS (BASIC STRING HANDLING REFERENCES)
- CLASSROOM ASSIGNMENT INSTRUCTIONS (VITYARTHI)

PEREPPU SOUMYA
25BCE10126
2025 BATCH

The End

https://github.com/Perepu-8/Perepusoumya/blob/main/passwordcheck.py

https://github.com/Perepu-8/Perepusoumya/blob/main/README.md

https://github.com/Perepu-8/Perepusoumya/blob/main/STATEMENT.md