Build a Password Strength Analyzer Tool

Algorithm Explanation

This program is a **Password Strength Analyzer** with enhanced security features. It assesses password strength and provides **hashed versions** using **BCrypt** and **Argon2**, two of the most secure hashing algorithms available.

1. Password Strength Analysis

The algorithm evaluates password security based on five key factors:

- ✅ **Length:** At least 8 characters.
- ✅ **Digits:** Includes at least one numerical digit (0-9).
- ✅ **Uppercase Letters:** Contains at least one uppercase letter (A-Z).
- ✅ **Lowercase Letters:** Contains at least one lowercase letter (a-z).
- ✅ **Special Characters:** Includes at least one special character (!@#$%^&*() etc.).

Each fulfilled condition **increases the strength score**:

- **Very Strong** (5/5 criteria met)
- **Strong** (4/5)
- **Moderate** (3/5)
- **Weak** (2/5)
- **Very Weak** (0-1/5)

2. Password Hashing

Instead of insecure hashing methods like MD5 or SHA256, this program uses BCrypt and Argon2, which are highly resistant to brute-force attacks.

🔹 **BCrypt Hashing**

- Salting & Adaptive Cost: BCrypt automatically generates a salt and includes a cost factor, making brute-force attacks harder.
- Secure Against Rainbow Tables: Due to its built-in salting mechanism.
- **Usage in Code:**

```
salt = bcrypt.gensalt()

hashed = bcrypt.hashpw(password.encode(), salt)
```

🔹 Argon2 Hashing

- More Advanced than BCrypt: Argon2 is the winner of the 2015 Password Hashing Competition.
- Resistant to GPU-based Attacks: It uses memory-intensive operations to slow down attackers.
- **Usage in Code:**

```
argon2_hasher = argon2.PasswordHasher()

hashed = argon2_hasher.hash(password)
```

3. User Interface Features

- **Password Visibility Toggle**: Users can view or hide their password input.
- **Strength Indicator**: Color-coded label and progress bar show password strength.
- **Copy Hashed Passwords**: Users can easily copy secure hashes.
- **Clear Button**: Resets the input field and strength meter.

Effectiveness of the Algorithm

This implementation **greatly enhances password security** compared to traditional methods.

✅ Strengths

- **Strong Hashing Mechanisms**: Uses **BCrypt** and **Argon2** instead of outdated MD5/SHA256.
- **User-Friendly UI**: Provides **clear feedback** on password strength.
- **Real-Time Analysis**: Strength is calculated instantly upon input.
- **Copy Functionality**: Allows users to store hashed passwords securely.

Limitations

- **No Common Password Check**: The program doesn’t check against leaked password databases (e.g., rockyou.txt or "Have I Been Pwned").
- **No Password Generation**: It doesn’t suggest strong passwords if the input is weak.

Possible Improvements

- **Integrate a breached password API** (e.g., "Have I Been Pwned") to check if a password has been leaked.
- **Add a password generator** to suggest secure passwords when a weak one is entered.

File Edit Selection View Go Run Terminal Help

Search

PSWD_Strenght_Check.py

D: > Future Intern > CS1 Task 2 > PSWD_Strenght_Check.py > ...

```python
import tkinter as tk
from tkinter import messagebox, ttk
import re
import bcrypt
import argon2
from ttkbootstrap import Style

# Function to analyze password strength
def analyze_password():
    password = entry_password.get()

    if not password:
        messagebox.showwarning("Input Error", "Please enter a password.")
        return

    strength = check_password_strength(password)
    hashed_password_bcrypt = hash_password_bcrypt(password)
    hashed_password_argon2 = hash_password_argon2(password)

    result_text.set(f"Password Strength: {strength}\nHashed Password (BCrypt): {hashed_password_bcrypt}\nHashed Password (Argon2): {hashed_password_argon2}")
    update_strength_color(strength)
    update_strength_meter(strength)

# Function to check password strength
def check_password_strength(password):
    length_criteria = len(password) >= 8
    digit_criteria = re.search(r"\d", password) is not None
    upper_criteria = re.search(r"[A-Z]", password) is not None
    lower_criteria = re.search(r"[a-z]", password) is not None
    special_criteria = re.search(r"[!@#$%^&*(),.?\":{}|<>]", password) is not None

    criteria_met = [length_criteria, digit_criteria, upper_criteria, lower_criteria, special_criteria]
    strength = sum(criteria_met)

    if strength == 5:
        return "Very Strong"
    elif strength == 4:
        return "Strong"
    elif strength == 3:
        return "Moderate"
    elif strength == 2:
        return "Weak"
    else:
        return "Very Weak"

# Function to hash the password using BCrypt
def hash_password_bcrypt(password):
    # Generate a salt and hash the password
    salt = bcrypt.gensalt()
    hashed = bcrypt.hashpw(password.encode(), salt)
    return hashed.decode('utf-8')
```

0 0 BLACKBOX Chat Add Logs CyberCoder Improve Code Share Code Link

Ln 7, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.0 BLACKBOXAI: Open Chat

File Edit Selection View Go Run Terminal Help

Search

PSWD_Strenght_Check.py

D: > Future Intern > CS1 Task 2 > PSWD_Strenght_Check.py > ...

```python
# Function to hash the password using BCrypt
def hash_password_bcrypt(password):
    # Generate a salt and hash the password
    salt = bcrypt.gensalt()
    hashed = bcrypt.hashpw(password.encode(), salt)
    return hashed.decode('utf-8')

# Fu
(function) def hash_password_argon2(password: Any) -> str
def hash_password_argon2(password):
    # Create an Argon2 hasher object and hash the password
    argon2_hasher = argon2.PasswordHasher()
    hashed = argon2_hasher.hash(password)
    return hashed

# Function to update the color based on strength
def update_strength_color(strength):
    color_map = {
        "Very Strong": "green",
        "Strong": "blue",
        "Moderate": "orange",
        "Weak": "red",
        "Very Weak": "darkred"
    }
    label_result.config(foreground=color_map[strength])

# Function to update the strength meter
def update_strength_meter(strength):
    strength_meter['value'] = {
        "Very Strong": 100,
        "Strong": 80,
        "Moderate": 60,
        "Weak": 40,
        "Very Weak": 20
    }[strength]

# Function to toggle password visibility
def toggle_password_visibility():
    if show_password_var.get():
        entry_password.config(show="")
    else:
        entry_password.config(show="*")

# Function to copy hashed passwords to clipboard
def copy_to_clipboard():
    hashed_password_bcrypt = hash_password_bcrypt(entry_password.get())
    hashed_password_argon2 = hash_password_argon2(entry_password.get())
    root.clipboard_clear()
    root.clipboard_append(f"BCrypt: {hashed_password_bcrypt}\nArgon2: {hashed_password_argon2}")
    messagebox.showinfo("Copied to Clipboard", "Hashed passwords copied to clipboard.")

# Function to clear input
```

0 0 BLACKBOX Chat Add Logs CyberCoder Improve Code Share Code Link

Ln 7, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.0 BLACKBOXAI: Open Chat

File Edit Selection View Go Run Terminal Help

Search

PSWD_Strenght_Check.py

D: > Future Intern > CS1 Task 2 > PSWD_Strenght_Check.py > ...

```python
def copy_to_clipboard():
    messagebox.showinfo("Copied to Clipboard", "Hashed passwords copied to clipboard.")

# Function to clear input
def clear_input():
    entry_password.delete(0, tk.END)
    result_text.set("")
    strength_meter['value'] = 0

# Create the main window
root = tk.Tk()
style = Style(theme='cosmo')  # Using 'cosmo' theme for a different look
root.title("Password Analyzer")
root.geometry("450x450")
root.configure(bg=style.colors.bg)

# Create and place the input field frame
frame_input = ttk.Frame(root, padding="20")
frame_input.pack(pady=10)

label_password = ttk.Label(frame_input, text="Enter Password:", style='primary.TLabel')
label_password.grid(row=0, column=0, padx=5)

entry_password = ttk.Entry(frame_input, show="*")
entry_password.grid(row=0, column=1, padx=5)

show_password_var = tk.BooleanVar()
check_show_password = ttk.Checkbutton(frame_input, text="Show Password", variable=show_password_var, command=toggle_password_visibility)
check_show_password.grid(row=1, column=0, columnspan=2, pady=5)

# Create and place the analyze button
button_analyze = ttk.Button(root, text="Analyze", command=analyze_password, style='primary.TButton')
button_analyze.pack(pady=10)

# Create and place the result frame
frame_result = ttk.Frame(root, padding="10")
frame_result.pack(pady=5)

result_text = tk.StringVar()
label_result = ttk.Label(frame_result, textvariable=result_text, font=("Helvetica", 10), style='primary.TLabel')
label_result.pack(pady=5)

# Create and place the strength meter
strength_meter = ttk.Progressbar(root, length=300, mode='determinate')
strength_meter.pack(pady=10)

# Create and place the copy and clear buttons
frame_buttons = ttk.Frame(root, padding="10")
frame_buttons.pack(pady=10)

button_copy = ttk.Button(frame_buttons, text="Copy Hashed Passwords", command=copy_to_clipboard, style='primary.TButton')
```

⊗ 0 ⚠ 0 BLACKBOX Chat Add Logs CyberCoder Improve Code Share Code Link

Ln 7, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.0 BLACKBOXAI: Open Chat

File Edit Selection View Go Run Terminal Help

Search

PSWD_Strenght_Check.py

D: > Future Intern > CS1 Task 2 > PSWD_Strenght_Check.py > ...

```python
style = Style(theme='cosmo')  # Using 'cosmo' theme for a different look
root.title("Password Analyzer")
root.geometry("450x450")
root.configure(bg=style.colors.bg)

# Create and place the input field frame
frame_input = ttk.Frame(root, padding="20")
frame_input.pack(pady=10)

label_password = ttk.Label(frame_input, text="Enter Password:", style='primary.TLabel')
label_password.grid(row=0, column=0, padx=5)

entry_password = ttk.Entry(frame_input, show="*")
entry_password.grid(row=0, column=1, padx=5)

show_password_var = tk.BooleanVar()
check_show_password = ttk.Checkbutton(frame_input, text="Show Password", variable=show_password_var, command=toggle_password_visibility)
check_show_password.grid(row=1, column=0, columnspan=2, pady=5)

# Create and place the analyze button
button_analyze = ttk.Button(root, text="Analyze", command=analyze_password, style='primary.TButton')
button_analyze.pack(pady=10)

# Create and place the result frame
frame_result = ttk.Frame(root, padding="10")
frame_result.pack(pady=5)

result_text = tk.StringVar()
label_result = ttk.Label(frame_result, textvariable=result_text, font=("Helvetica", 10), style='primary.TLabel')
label_result.pack(pady=5)

# Create and place the strength meter
strength_meter = ttk.Progressbar(root, length=300, mode='determinate')
strength_meter.pack(pady=10)

# Create and place the copy and clear buttons
frame_buttons = ttk.Frame(root, padding="10")
frame_buttons.pack(pady=10)

button_copy = ttk.Button(frame_buttons, text="Copy Hashed Passwords", command=copy_to_clipboard, style='primary.TButton')
button_copy.grid(row=0, column=0, padx=5)

button_clear = ttk.Button(frame_buttons, text="Clear", command=clear_input, style='primary.TButton')
button_clear.grid(row=0, column=1, padx=5)

# Start the main event loop
root.mainloop()
```

0 0 BLACKBOX Chat Add Logs CyberCoder Improve Code Share Code Link

Ln 7, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.0 BLACKBOXAI: Open Chat

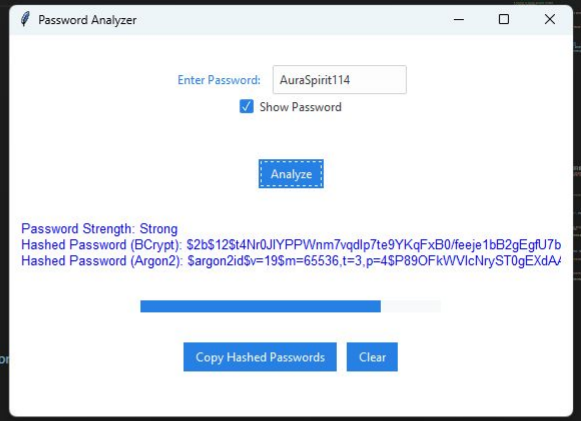
File Edit Selection View Go Run Terminal Help

Search

PSWD_Strenght_Check.py

D: > Future Intern > CS1 Task 2 > PSWD_Strenght_Check.py > ...

```python
import tkinter as tk
from tkinter import messagebox, ttk
import re
import bcrypt
import argon2
from ttkbootstrap import Style

# Function to analyze password strength
def analyze_password():
    password = entry_password.get()

    if not password:
        messagebox.showwarning("Input Error", "Please enter a password.")
        return

    strength = check_password_strength(password)
    hashed_password_bcrypt = hash_password_bcrypt(password)
    hashed_password_argon2 = hash_password_argon2(password)

    result_text.set(f"Password Strength: {strength}\nHashed Password (BCrypt): {hashed_password_bcrypt}\nHashed Password (Argon2): {hashed_password_argon
    update_strength_color(strength)
    update_strength_meter(strength)

# Function to check password strength
def check_password_strength(password):
    length_criteria = len(password) >= 8
    digit_criteria = re.search(r"\d", password) is not None
    upper_criteria = re.search(r"[A-Z]", password) is not None
    lower_criteria = re.search(r"[a-z]", password) is not None
    special_criteria = re.search(r"[!@#$%^&*(),.?\":{}|<>]", password) is not None

    criteria_met = [length_criteria, digit_criteria, upper_criteria, lower_criteria, special_criteria]
    strength = sum(criteria_met)

    if strength == 5:
        return "Very Strong"
    elif strength == 4:
        return "Strong"
    elif strength == 3:
        return "Moderate"
    elif strength == 2:
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

```
PS C:\Users\Tarun> & C:/Users/Tarun/AppData/Local/Programs/Python/Python313/python.exe "d:/Future Intern/CS1 Task 2/PSWD_Strenght_Check.py"
```

0 0 BLACKBOX Chat Add Logs CyberCoder Improve Code Share Code Link

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.0 BLACKBOXAI: Open Chat