Password strength checker

Project Summary: Password Strength Checker

Objective:
The objective of this project is to create a password strength checker application that evaluates the strength of a password and provides suggestions for improvement.

Key Features:

1. Password Strength Evaluation: The application checks the password strength based on various criteria such as length, presence of digits, uppercase and lowercase letters, special characters, and common patterns.
2. Suggestions for Improvement: The application provides suggestions for improving the password strength.
3. Argon2 Hashing: The application hashes the password using Argon2, a secure password hashing algorithm.
4. GUI Interface: The application has a user-friendly GUI interface built using tkinter.

Functionality:

1. Password Input: The user can enter a password in the GUI interface.
2. Password Strength Check: The application checks the password strength and displays the result.
3. Suggestions: The application provides suggestions for improving the password strength.
4. Argon2 Hash: The application hashes the password using Argon2 and displays the hash.

Benefits:

1. Improved Password Security: The application helps users create stronger passwords by providing suggestions for improvement.
2. Secure Password Storage: The application uses Argon2 hashing to securely store passwords.
3. User-Friendly Interface: The GUI interface makes it easy for users to check their password strength and get suggestions for improvement.

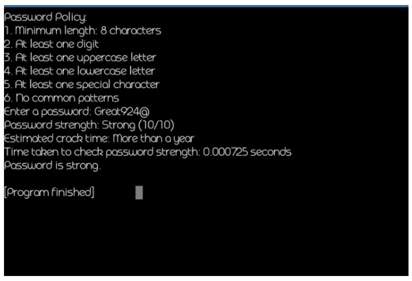
Technical Details:

1. Programming Language: Python
2. GUI Library: tkinter
3. Password Hashing Algorithm: Argon2
4. Password Strength Criteria*: Length, presence of digits, uppercase and lowercase letters, special characters, and common patterns.

Outcome:
The outcome of this project is a functional password strength checker application that provides users with a secure and user-friendly way to check their password strength and get suggestions for improvement.

```
import re
import time

def check_password_strength(password):
    strength = 0
    errors = []
    score = 0
    crack_time = ""

    # Check password length
    if len(password) < 8:
        errors.append("Password is too short. It should be at least 8 characters.")
    else:
        strength += 1
        score += 2

    # Check for digits
    if not re.search(r"\d", password):
        errors.append("Password should have at least one digit.")
    else:
        strength += 1
        score += 2

    # Check for uppercase letters
    if not re.search(r"[A-Z]", password):
        errors.append("Password should have at least one uppercase letter.")
    else:
        strength += 1
        score += 2

    # Check for lowercase letters
    if not re.search(r"[a-z]", password):
        errors.append("Password should have at least one lowercase letter.")
    else:
        strength += 1
        score += 2

    # Check for special characters
    if not re.search(r"[^a-zA-Z0-9]", password):
        errors.append("Password should have at least one special character.")
    else:
        strength += 1
        score += 2

    # Check for common patterns
    common_patterns = ["abc", "123", "qwerty", "password"]
    for pattern in common_patterns:
        if pattern in password.lower():
            errors.append("Password should not contain common patterns.")
            score -= 2
            break

    # Estimate crack time
    if score <= 4:
        crack_time = "Less than a minute"
    elif score <= 6:
        crack_time = "Several minutes to hours"
    elif score <= 8:
        crack_time = "Several hours to days"
    else:
        crack_time = "More than a year"

    if strength == 5:
        strength_level = "Strong"
    elif strength >= 3:
        strength_level = "Medium"
    else:
        strength_level = "Weak"

    return strength_level, errors, score, crack_time

def main():
    print("Password Policy:")
    print("1. Minimum length: 8 characters")
    print("2. At least one digit")
    print("3. At least one uppercase letter")
    print("4. At least one lowercase letter")
    print("5. At least one special character")
    print("6. No common patterns")

    password = input("Enter a password: ")
    start_time = time.time()
    strength_level, errors, score, crack_time = check_password_strength(password)
    end_time = time.time()

    print(f"Password strength: {strength_level} ({score}/10)")
    print(f"Estimated crack time: {crack_time}")
    print(f"Time taken to check password strength: {end_time - start_time:.6f} seconds")

    if strength_level == "Strong":
        print("Password is strong.")
    elif strength_level == "Medium":
        print("Password is medium strength.")
        print("Suggestions:")
        for error in errors:
            print("*", error)
    else:
        print("Password is weak.")
        print("Suggestions:")
        for error in errors:
            print("*", error)

if __name__ == "__main__":
    main()
```

Output

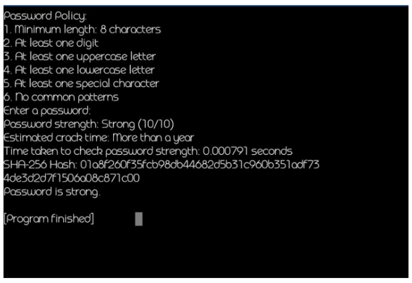


```
Password Policy:
1. Minimum length: 8 characters
2. At least one digit
3. At least one uppercase letter
4. At least one lowercase letter
5. At least one special character
6. No common patterns
Enter a password: Great924@
Password strength: Strong (10/10)
Estimated crack time: More than a year
Time taken to check password strength: 0.000725 seconds
Password is strong.

[Program finished]
```

Now SHA 256 is used to improve password strength and suggestions

```
import re
import time
import getpass
import hashlib

def check_password_strength(password):
    strength = 0
    errors = []
    score = 0
    crack_time = ""
    suggestions = []

    # Check password length
    if len(password) < 8:
        errors.append("Password is too short. It should be at least 8 characters.")
        suggestions.append("Increase password length to at least 8 characters.")
    else:
        strength += 1
        score += 2

    # Check for digits
    if not re.search(r"\d", password):
        errors.append("Password should have at least one digit.")
        suggestions.append("Add at least one digit to the password.")
    else:
        strength += 1
        score += 2

    # Check for uppercase letters
    if not re.search(r"[A-Z]", password):
        errors.append("Password should have at least one uppercase letter.")
        suggestions.append("Add at least one uppercase letter to the password.")
    else:
        strength += 1
        score += 2

    # Check for lowercase letters
    if not re.search(r"[a-z]", password):
        errors.append("Password should have at least one lowercase letter.")
        suggestions.append("Add at least one lowercase letter to the password.")
    else:
        strength += 1
        score += 2

    # Check for special characters
    if not re.search(r"[^a-zA-Z0-9]", password):
        errors.append("Password should have at least one special character.")
        suggestions.append("Add at least one special character to the password.")
    else:
        strength += 1
        score += 2

    # Check for common patterns
    common_patterns = ["abc", "123", "qwerty", "password"]
    for pattern in common_patterns:
        if pattern in password.lower():
            errors.append("Password should not contain common patterns.")
            suggestions.append("Avoid using common patterns in the password.")
            score -= 2
            break

    # Estimate crack time
    if score <= 4:
        crack_time = "Less than a minute"
    elif score <= 6:
        crack_time = "Several minutes to hours"
    elif score <= 8:
        crack_time = "Several hours to days"
    else:
        crack_time = "More than a year"
    if strength == 5:
        strength_level = "Strong"
    elif strength >= 3:
        strength_level = "Medium"
    else:
        strength_level = "Weak"

    # Hash the password
    hashed_password = hashlib.sha256(password.encode()).hexdigest()

    return strength_level, errors, score, crack_time, suggestions, hashed_password

def main():
    print("Password Policy:")
    print("1. Minimum length: 8 characters")
    print("2. At least one digit")
    print("3. At least one uppercase letter")
    print("4. At least one lowercase letter")
    print("5. At least one special character")
    print("6. No common patterns")

    password = getpass.getpass("Enter a password: ")
    start_time = time.time()
    strength_level, errors, score, crack_time, suggestions, hashed_password =
check_password_strength(password)
    end_time = time.time()

    print(f"Password strength: {strength_level} ({score}/10)")
    print(f"Estimated crack time: {crack_time}")
    print(f"Time taken to check password strength: {end_time - start_time:.6f} seconds")
    print(f"SHA-256 Hash: {hashed_password}")

    if strength_level == "Strong":
        print("Password is strong.")
    elif strength_level == "Medium":
        print("Password is medium strength.")
        print("Suggestions:")
        for suggestion in suggestions:
            print("*", suggestion)
    else:
        print("Password is weak.")
        print("Suggestions:")
        for suggestion in suggestions:
            print("*", suggestion)

if __name__ == "__main__":
    main()
```

Output



```
Password Policy:
1. Minimum length: 8 characters
2. At least one digit
3. At least one uppercase letter
4. At least one lowercase letter
5. At least one special character
6. No common patterns
Enter a password:
Password strength: Strong (10/10)
Estimated crack time: More than a year
Time taken to check password strength: 0.000791 seconds
SHA-256 Hash: 01a8f260f35fcb98db44682d5b31c960b351adf73
4de3d2d7f1506a08c871c00
Password is strong.

[Program finished]
```

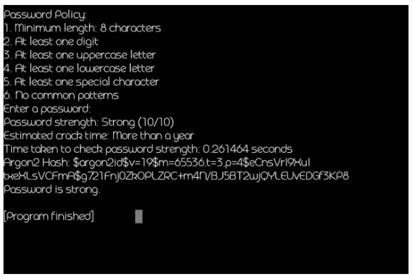
Using argon2 hash for more secured storage

```
import re
import time
import getpass
from argon2 import PasswordHasher

def check_password_strength(password):
    strength = 0
    errors = []
    score = 0
    crack_time = ""
    suggestions = []

    # Check password length
    if len(password) < 8:
        errors.append("Password is too short. It should be at least 8 characters.")
        suggestions.append("Increase password length to at least 8 characters.")
    else:
        strength += 1
        score += 2

    # Check for digits
    if not re.search(r"\d", password):
        errors.append("Password should have at least one digit.")
        suggestions.append("Add at least one digit to the password.")
    else:
        strength += 1
        score += 2

    # Check for uppercase letters
    if not re.search(r"[A-Z]", password):
        errors.append("Password should have at least one uppercase letter.")
        suggestions.append("Add at least one uppercase letter to the password.")
    else:
```

```
    strength += 1
    score += 2

# Check for lowercase letters
if not re.search(r"[a-z]", password):
    errors.append("Password should have at least one lowercase letter.")
    suggestions.append("Add at least one lowercase letter to the password.")
else:
    strength += 1
    score += 2

# Check for special characters
if not re.search(r"[^a-zA-Z0-9]", password):
    errors.append("Password should have at least one special character.")
    suggestions.append("Add at least one special character to the password.")
else:
    strength += 1
    score += 2

# Check for common patterns
common_patterns = ["abc", "123", "qwerty", "password"]
for pattern in common_patterns:
    if pattern in password.lower():
        errors.append("Password should not contain common patterns.")
        suggestions.append("Avoid using common patterns in the password.")
        score -= 2
        break

# Estimate crack time
if score <= 4:
    crack_time = "Less than a minute"
elif score <= 6:
    crack_time = "Several minutes to hours"
elif score <= 8:
    crack_time = "Several hours to days"
else:
    crack_time = "More than a year"

if strength == 5:
    strength_level = "Strong"
elif strength >= 3:
    strength_level = "Medium"
else:
    strength_level = "Weak"

# Hash the password using Argon2
ph = PasswordHasher()
hashed_password = ph.hash(password)
```

```
    return strength_level, errors, score, crack_time, suggestions, hashed_password

def main():
    print("Password Policy:")
    print("1. Minimum length: 8 characters")
    print("2. At least one digit")
    print("3. At least one uppercase letter")
    print("4. At least one lowercase letter")
    print("5. At least one special character")
    print("6. No common patterns")

    password = getpass.getpass("Enter a password: ")
    start_time = time.time()
    strength_level, errors, score, crack_time, suggestions, hashed_password =
check_password_strength(password)
    end_time = time.time()

    print(f"Password strength: {strength_level} ({score}/10)")
    print(f"Estimated crack time: {crack_time}")
    print(f"Time taken to check password strength: {end_time - start_time:.6f} seconds")
    print(f"Argon2 Hash: {hashed_password}")

    if strength_level == "Strong":
        print("Password is strong.")
    elif strength_level == "Medium":
        print("Password is medium strength.")
        if suggestions:
            print("Suggestions:")
            for suggestion in suggestions:
                print("*", suggestion)
    else:
        print("Password is weak.")
        if suggestions:
            print("Suggestions:")
            for suggestion in suggestions:
                print("*", suggestion)

if __name__ == "__main__":
    try:
        main()
    except Exception as e:
        print("An error occurred: ", str(e))
```

Output

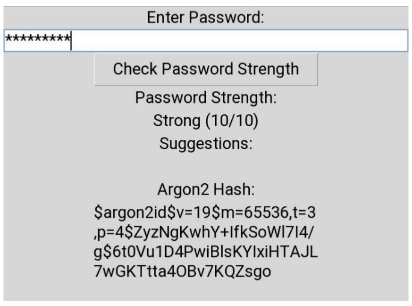


```
Password Policy:
1. Minimum length: 8 characters
2. At least one digit
3. At least one uppercase letter
4. At least one lowercase letter
5. At least one special character
6. No common patterns
Enter a password:
Password strength: Strong (10/10)
Estimated crack time: More than a year
Time taken to check password strength: 0.261464 seconds
Argon2 Hash: $argon2id$v=19$m=65536,t=3,p=4$eCnsVrI9XuI
txeXLsVCFmA$g721FnjOZkOPLZRC+m4N/BJ5BT2wjQYLEUvEDGf3KP8
Password is strong.

[Program finished]
```

Now the GUI application allows user to check the password strength

```
import re
import time
import getpass
from argon2 import PasswordHasher
import tkinter as tk
from tkinter import messagebox

class PasswordStrengthChecker:
    def __init__(self):
        self.window = tk.Tk()
        self.window.title("Password Strength Checker")

        self.password_label = tk.Label(self.window, text="Enter Password:")
        self.password_label.pack()

        self.password_entry = tk.Entry(self.window, show="*", width=50)
        self.password_entry.pack()

        self.check_button = tk.Button(self.window, text="Check Password Strength",
command=self.check_password_strength)
        self.check_button.pack()

        self.strength_label = tk.Label(self.window, text="Password Strength:")
        self.strength_label.pack()

        self.strength_result = tk.Label(self.window, text="")
        self.strength_result.pack()

        self.suggestions_label = tk.Label(self.window, text="Suggestions:")
        self.suggestions_label.pack()

        self.suggestions_result = tk.Label(self.window, text="", wraplength=400,
justify=tk.LEFT)
        self.suggestions_result.pack()

        self.hash_label = tk.Label(self.window, text="Argon2 Hash:")
        self.hash_label.pack()
        self.hash_result = tk.Label(self.window, text="", wraplength=400, justify=tk.LEFT)
        self.hash_result.pack()

    def check_password_strength(self):
        password = self.password_entry.get()

        strength = 0
        errors = []
        score = 0
        crack_time = ""
        suggestions = []

        # Check password length
        if len(password) < 8:
            errors.append("Password is too short. It should be at least 8 characters.")
            suggestions.append("Increase password length to at least 8 characters.")
        else:
            strength += 1
            score += 2

        # Check for digits
        if not re.search(r"\d", password):
            errors.append("Password should have at least one digit.")
            suggestions.append("Add at least one digit to the password.")
        else:
            strength += 1
            score += 2

        # Check for uppercase letters
        if not re.search(r"[A-Z]", password):
            errors.append("Password should have at least one uppercase letter.")
            suggestions.append("Add at least one uppercase letter to the password.")
        else:
            strength += 1
            score += 2

        # Check for lowercase letters
        if not re.search(r"[a-z]", password):
            errors.append("Password should have at least one lowercase letter.")
            suggestions.append("Add at least one lowercase letter to the password.")
        else:
            strength += 1
            score += 2

        # Check for special characters
        if not re.search(r"[^a-zA-Z0-9]", password):
            errors.append("Password should have at least one special character.")
            suggestions.append("Add at least one special character to the password.")
        else:
            strength += 1
            score += 2

        # Check for common patterns
        common_patterns = ["abc", "123", "qwerty", "password"]
        for pattern in common_patterns:
            if pattern in password.lower():
                errors.append("Password should not contain common patterns.")
                suggestions.append("Avoid using common patterns in the password.")
                score -= 2
                break

        # Estimate crack time
        if score <= 4:
            crack_time = "Less than a minute"
        elif score <= 6:
            crack_time = "Several minutes to hours"
        elif score <= 8:
            crack_time = "Several hours to days"
        else:
            crack_time = "More than a year"

        if strength == 5:
            strength_level = "Strong"
        elif strength >= 3:
            strength_level = "Medium"
        else:
            strength_level = "Weak"

        # Hash the password using Argon2
        ph = PasswordHasher()
        hashed_password = ph.hash(password)

        self.strength_result.config(text=f"{strength_level} ({score}/10)")
        self.suggestions_result.config(text="\n".join(suggestions))
        self.hash_result.config(text=hashed_password)

    def run(self):
        self.window.mainloop()

if __name__ == "__main__":
    password_strength_checker = PasswordStrengthChecker()
    password_strength_checker.run()
```

Output


Enter Password:

****

Password Strength:

Strong (10/10)

Suggestions:

Argon2 Hash:

```
$argon2id$v=19$m=65536,t=3
,p=4$ZyzNgKwhY+IfkSoWl7I4/
g$6t0Vu1D4PwiBlsKYIxiHTAJL
7wGKTtta4OBv7KQZsgo
```