```
import re
import sys
import time
import getpass

# ANSI color codes
RED = "\033[91m"
GREEN = "\033[92m"
YELLOW = "\033[93m"
CYAN = "\033[96m"
MAGENTA = "\033[95m"
BOLD = "\033[1m"
RESET = "\033[0m"

# Typing effect for messages
def type_print(text, delay=0.02):
    for char in text:
        sys.stdout.write(char)
        sys.stdout.flush()
        time.sleep(delay)
    print()

# Password strength bar
def strength_bar(score):
    bar = f"{BOLD}Strength Meter: "
    levels = ["▢", "▣", "▣", "▣", "▣", "▣"]
    for i in range(5):
        if i < score:
            bar += GREEN + "█" + RESET
        else:
            bar += RED + "░" + RESET
    return bar

# Suggestions based on missing elements
def get_suggestions(password):
    suggestions = []
    if not re.search(r"[A-Z]", password):
        suggestions.append("Add at least one UPPERCASE letter.")
    if not re.search(r"[a-z]", password):
        suggestions.append("Include some lowercase letters.")
    if not re.search(r"\d", password):
        suggestions.append("Insert at least one number.")
    if not re.search(r"[!@#$%^&*()\-_=+{};:,<.>]", password):
        suggestions.append("Use a special character (e.g., !, @, #).")
    if len(password) < 8:
        suggestions.append("Make your password at least 8 characters long.")
    return suggestions

def check_password_complexity(password):
    has_uppercase = bool(re.search(r"[A-Z]", password))
    has_lowercase = bool(re.search(r"[a-z]", password))
    has_digit = bool(re.search(r"\d", password))
    has_special = bool(re.search(r"[!@#$%^&*()\-_=+{};:,<.>]", password))
    length_ok = len(password) >= 8

    score = sum([has_uppercase, has_lowercase, has_digit, has_special, length_ok])

    if score == 5:
        strength = f"{GREEN}🔒 Excellent{RESET}"
    elif score >= 3:
        strength = f"{YELLOW}⚠️  Fair{RESET}"
    else:
        strength = f"{RED}❌ Weak{RESET}"

    report = f"""
{BOLD}{CYAN}🔍 Password Check Result:{RESET}
- Length: {'✅' if length_ok else '❌'}
- Uppercase: {'✅' if has_uppercase else '❌'}
- Lowercase: {'✅' if has_lowercase else '❌'}
- Digit: {'✅' if has_digit else '❌'}
- Special Char: {'✅' if has_special else '❌'}

{strength_bar(score)}
{BOLD}🔐 Overall Strength: {strength}{RESET}
"""

    suggestions = get_suggestio
    if suggestions:
        report += f"\n{MAGENTA}{BOLD}
        for tip in suggestions:
            report += f"  - {tip}\n"
```

```
        else:
            report += f"\n{GREEN}✅ Your password looks great. No suggestions needed!{RESET}"

        return report


    # Main function
    if __name__ == "__main__":
        type_print(f"{BOLD}{CYAN}Welcome to the Enhanced Password Strength Checker! 🔐{RESET}")
        time.sleep(0.3)

        password = getpass.getpass(f"{CYAN}Enter your password (hidden): {RESET}")
        show = input(f"{YELLOW}Do you want to see your password? (y/n): {RESET}").strip().lower()
        if show == 'y':
            print(f"{MAGENTA}🔎 You entered: {password}{RESET}")

        print()
        result = check_password_complexity(password)
        type_print(result, delay=0.01)
```

```
Welcome to the Enhanced Password Strength Checker! 🔐
Enter your password (hidden): ·····
Do you want to see your password? (y/n): y
🔎 You entered: vasantha@846


🔍 Password Check Result:
- Length: ✅
- Uppercase: ❌
- Lowercase: ✅
- Digit: ✅
- Special Char: ✅

Strength Meter: ███▒
🔐 Overall Strength: ⚠️  Fair

💡 Suggestions to Improve:
  - Add at least one UPPERCASE letter.
```