```python
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import string
import random

class PasswordGenerator:
    def __init__(self, master):
        self.master = master
        self.master.title("Password Generator")
        self.master.geometry("1000x1000")

        # GUI elements
        font_style = ("Arial", 12, "normal")
        bold_font_style = ("Arial", 12, "bold")

        # Add a centered title
        self.label_title = tk.Label(master, text="GUI PASSWORD GENERATOR", font=("Arial", 14, "bold"), bg='#f0f0f0')
        self.label_title.pack(pady=10)

        self.label_length = tk.Label(master, text="Enter Password Length:", bg='#f0f0f0', font=bold_font_style)
        self.label_length.pack()

        self.length_var = tk.StringVar()
        self.entry_length = tk.Entry(master, textvariable=self.length_var, font=font_style)
        self.entry_length.pack()

        self.label_options = tk.Label(master, text="Password Options:", bg='#f0f0f0', font=bold_font_style)  # Bold font for Password Options
        self.label_options.pack()

        self.lower_var = tk.IntVar()
        self.check_lower = tk.Checkbutton(master, text="Lowercase Letters", variable=self.lower_var, bg='#f0f0f0', font=font_style)
        self.check_lower.pack()

        self.upper_var = tk.IntVar()
        self.check_upper = tk.Checkbutton(master, text="Uppercase Letters", variable=self.upper_var, bg='#f0f0f0', font=font_style)
        self.check_upper.pack()

        self.digit_var = tk.IntVar()
        self.check_digit = tk.Checkbutton(master, text="Digits", variable=self.digit_var, bg='#f0f0f0', font=font_style)
        self.check_digit.pack()

        self.special_var = tk.IntVar()
        self.check_special = tk.Checkbutton(master, text="Special Characters", variable=self.special_var, bg='#f0f0f0', font=font_style)
        self.check_special.pack()

        self.generate_button = tk.Button(master, text="Generate Password", command=self.generate_password, bg='#4caf50', fg='white', font=bold_font_style)
        self.generate_button.pack()

        self.copy_button = tk.Button(master, text="Copy to Clipboard", command=self.copy_to_clipboard, bg='#2196F3', fg='white', font=font_style)
        self.copy_button.pack(pady=5)

        self.toggle_visibility_button = tk.Button(master, text="Toggle Visibility", command=self.toggle_visibility, bg='#2196F3', fg='white', font=font_style)
        self.toggle_visibility_button.pack(pady=5)

        self.progress_var = tk.DoubleVar()
        self.progressbar = ttk.Progressbar(master, variable=self.progress_var, mode='indeterminate', length=300)
        self.progressbar.pack()

        self.label_strength = tk.Label(master, text="Password Strength:", bg='#f0f0f0', font=font_style)
        self.label_strength.pack()

        self.strength_var = tk.StringVar()
        self.label_strength_value = tk.Label(master, textvariable=self.strength_var, font=("Arial", 10, "bold"), bg='#f0f0f0')
        self.label_strength_value.pack()

        self.password_var = tk.StringVar()
        self.entry_password = tk.Entry(master, textvariable=self.password_var, show='*', state='readonly', font=font_style)  # Password is hidden by default
        self.entry_password.pack()

        # Add a Reset button
        self.reset_button = tk.Button(master, text="Reset", command=self.reset_password, bg='#2196F3', fg='white', font=font_style)
        self.reset_button.pack(pady=10)

        # Add a Listbox for password history
        self.history_listbox = tk.Listbox(master, selectmode=tk.SINGLE, font=font_style, height=5)
        self.history_listbox.pack(pady=5)

        # Add an Entry for the name of the password in history
        self.history_name_var = tk.StringVar()
        self.entry_history_name = tk.Entry(master, textvariable=self.history_name_var, font=font_style)
        self.entry_history_name.pack(pady=5)

        # Add a label to display the total number of passwords in history
        self.label_total_passwords = tk.Label(master, text="Total Passwords in History: 0", bg='#f0f0f0', font=font_style)
        self.label_total_passwords.pack(pady=5)

        # Bind a callback to handle password selection from history
        self.history_listbox.bind('<Double-Button-1>', self.select_from_history)

    def generate_password(self):
        try:
            length = int(self.length_var.get())
            if length <= 0:
                raise ValueError("Password length must be greater than 0.")

            options = {
                'lowercase': string.ascii_lowercase if self.lower_var.get() else '',
                'uppercase': string.ascii_uppercase if self.upper_var.get() else '',
                'digits': string.digits if self.digit_var.get() else '',
                'special': string.punctuation if self.special_var.get() else ''
            }

            characters = ''.join(options.values())
            if not characters:
                raise ValueError("Select at least one option for password generation.")

            self.progressbar.start(10)

            password = ''.join(random.choice(characters) for _ in range(length))
            self.password_var.set(password)
            strength = "Weak" if length < 8 else "Medium" if any([self.lower_var.get(), self.upper_var.get(), self.digit_var.get(), self.special_var.get()]) else "Strong"
            self.strength_var.set(strength)

            # Update the password history
            history_name = self.history_name_var.get()
            if history_name:
                entry = f"{history_name}: {password}"
            else:
                entry = password
            self.update_password_history(entry)

            # Prompt a thank you message
            messagebox.showinfo("Thank You", "Thank you for using the Password Generator!")

        except ValueError as e:
            messagebox.showerror("Password Generator", str(e))
        finally:
            self.progressbar.stop()

    def update_password_history(self, password):
        # Add the generated password to the history
        self.history_listbox.insert(0, password)
        if self.history_listbox.size() > 5:
            self.history_listbox.delete(5, tk.END)

        # Update the total number of passwords in history
        total_passwords = self.history_listbox.size()
        self.label_total_passwords.config(text=f"Total Passwords in History: {total_passwords}")

    def reset_password(self):
        # Ask for confirmation before resetting
        response = messagebox.askquestion("Reset Confirmation", "Are you sure you want to reset?")
        if response == 'yes':
            # Reset the generated password and clear the options
            self.password_var.set("")
            self.length_var.set("")
            self.lower_var.set(0)
            self.upper_var.set(0)
            self.digit_var.set(0)
            self.special_var.set(0)
            self.strength_var.set("")
            self.history_name_var.set("")
            self.history_listbox.delete(0, tk.END)
            self.label_total_passwords.config(text="Total Passwords in History: 0")

    def copy_to_clipboard(self):
        password_to_copy = self.password_var.get()
        if password_to_copy:
            self.master.clipboard_clear()
            self.master.clipboard_append(password_to_copy)
            self.master.update()
            messagebox.showinfo("Copy to Clipboard", "Password copied to clipboard!")

    def toggle_visibility(self):
        current_state = self.entry_password["show"]
        new_state = "" if current_state == "*" else "*"
        self.entry_password["show"] = new_state

    def select_from_history(self, event):
        selected_index = self.history_listbox.curselection()
        if selected_index:
            selected_password = self.history_listbox.get(selected_index[0])
            # Extract password from the history entry (if there is a name associated)
            extracted_password = selected_password.split(": ", 1)[-1]
            self.password_var.set(extracted_password)

def main():
    root = tk.Tk()
    app = PasswordGenerator(root)
    root.configure(bg="skyblue")

    root.mainloop()

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