Password Checker Application

Overview

The Password Checker Application is a cross-platform GUI tool designed to enhance password security by providing two primary functionalities:

1. **Check Password Strength**:
    - Users can input their passwords to receive a strength assessment (Weak, Moderate, or Strong) along with helpful suggestions to improve the password's security.
2. **Generate Strong Passwords**:
    - Users can generate secure, randomized passwords of a specified length.
    - The generated password can be copied to the clipboard for easy use.

This application is built using **PyQt5**, ensuring a modern and consistent user interface on both **Windows** and **macOS**.

Features

1. Cross-Platform Compatibility

- The same Python codebase works seamlessly on Windows and macOS.
- The application adapts to the system theme, supporting both light and dark modes.

2. Intuitive User Interface

- **Main Window**:
    - Large, readable buttons for the primary functionalities.
    - Styled with a dark theme for a professional look.
- **Custom Dialogs**:
    - Consistent fonts and button sizes in all dialog windows.
    - Easily readable prompts and input fields.

3. Password Strength Assessment

- Analyzes the input password and classifies it as Weak, Moderate, or Strong.

- Provides actionable suggestions to improve password strength, such as including uppercase letters, numbers, or symbols.

4. Password Generation

- Generates a strong, random password with a user-defined length (between 8 and 32 characters).
- Combines uppercase letters, lowercase letters, digits, and symbols for maximum security.
- Allows users to copy the password directly to the clipboard.

Technical Details

Built With

- **Python**
- **PyQt5** for the graphical user interface
- **PyInstaller** for packaging the application into executables for Windows and macOS.

Design Considerations

1. **Dark Theme**:
    - The app uses a Fusion style with a custom dark palette to ensure a sleek look.
2. **Custom Dialogs**:
    - Replaced default dialogs (e.g., QInputDialog) with custom QDialog components to provide full control over layout and styling.

Packaging

For Windows:

1. Install PyInstaller:

```
pip install pyinstaller
```

2. Generate the executable:

```
pyinstaller --onefile --windowed --icon=path/to/icon.ico --add-data="path/to/icon.ico;." passcheck.py
```

3. The .exe file will be located in the dist folder.

- **Explanation of --add-data:** This ensures the .ico file is bundled with the application so that it is available to both the main window and dialog boxes.
- Modify your script to reference the bundled icon using this snippet:
- import os
- if hasattr(sys, "_MEIPASS"):
-     icon_path = os.path.join(sys._MEIPASS, "pass.ico")
- else:

  icon_path = "path/to/pass.ico"

For macOS:

1. Install PyInstaller:

pip3 install pyinstaller

2. Generate the .app:

pyinstaller --onefile --windowed --icon=path/to/icon.icns passcheck.py

3. The .app file will be located in the dist folder.

Creating an Icon for the App

Windows Icons (.ico Format)

For Windows, the app requires an icon in .ico format. Follow these steps:

1. **Prepare the Base Image**:
   - Start with a square .png file, preferably 256x256 pixels.
2. **Convert to .ico**:
   - Use an online converter (e.g., CloudConvert) or tools like GIMP to convert the .png file to .ico format.
3. **Use the .ico File**:
   - Specify the path to the .ico file using the --icon option when running PyInstaller.

macOS Icons (.icns Format)

For macOS, the app requires an icon in .icns format. Below are the steps:

1. **Prepare the Base Image**:
    - Ensure your starting image is a **square .png file** (e.g., 1024x1024 pixels).
    - Place the image in a working directory (e.g., ~/Desktop/AppIcon.png).
2. **Create the AppIcon.iconset Folder**:
    - Open the terminal and run:

```
mkdir AppIcon.iconset
```

3. **Resize the Image**:
    - Use the sips command to create all required icon sizes:
    - sips -z 16 16 ~/Desktop/AppIcon.png --out AppIcon.iconset/icon_16x16.png
    - sips -z 32 32 ~/Desktop/AppIcon.png --out AppIcon.iconset/icon_32x32.png
    - sips -z 128 128 ~/Desktop/AppIcon.png --out AppIcon.iconset/icon_128x128.png
    - sips -z 256 256 ~/Desktop/AppIcon.png --out AppIcon.iconset/icon_256x256.png
    - sips -z 512 512 ~/Desktop/AppIcon.png --out AppIcon.iconset/icon_512x512.png

```
cp ~/Desktop/AppIcon.png AppIcon.iconset/icon_512x512@2x.png
```

    - Each command resizes the base image into required dimensions and places them in the AppIcon.iconset folder.

4. **Generate the .icns File**:
    - Run the following command to convert the AppIcon.iconset folder into an .icns file:

```
iconutil -c icns AppIcon.iconset
```

    - The resulting AppIcon.icns file will be created in the same directory.

5. **Use the .icns File in PyInstaller**:

  - When building the macOS app, specify the path to the .icns file using the --icon option.

Key Difference Between .ico and .icns

- **Windows Icons (.ico)**:
  - .ico files are simpler and primarily used for executables and shortcuts on Windows.
  - A single .ico file can contain multiple resolutions (e.g., 16x16, 32x32, 256x256).
- **macOS Icons (.icns)**:
  - .icns files are specific to macOS and contain a series of icon sizes optimized for various macOS UI elements.
  - The AppIcon.iconset folder is a macOS-specific requirement for generating .icns files.

How to Use

1. **Run the Application**:
   - Double-click the .exe (Windows) or .app (macOS) file to launch.
2. **Check Password Strength**:
   - Click the "Check Password Strength" button.
   - Enter your password in the dialog box.
   - View the password strength and suggestions.
3. **Generate Strong Passwords**:
   - Click the "Generate Strong Password" button.
   - Enter the desired password length (8-32 characters).
   - Copy the generated password from the dialog box.
4. **Exit the Application**:
   - Click the "Exit" button on the main window.

Data Privacy and Security

This application is designed with user privacy and security in mind.

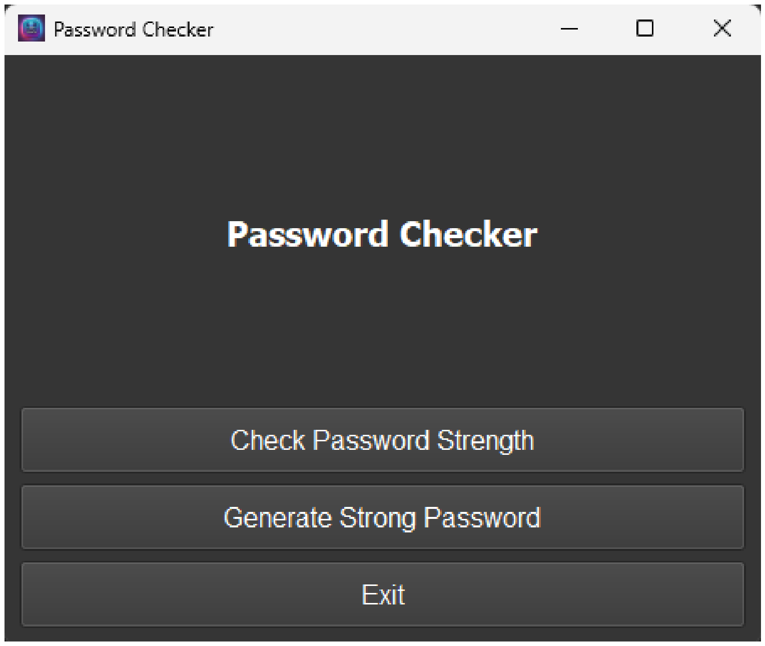
1. **No Data Collection**:
   - The application does not store, log, or transmit any user-entered passwords or generated passwords.
   - All password-related operations are performed locally on the user's device.
2. **Transparency**:
   - The application's source code is available on [Your GitHub Repository]. Users can review the code to verify that no data collection occurs.
3. **Best Practices**:
   - Users are encouraged to use the app in a secure environment and avoid sharing passwords with untrusted parties.

Screenshots

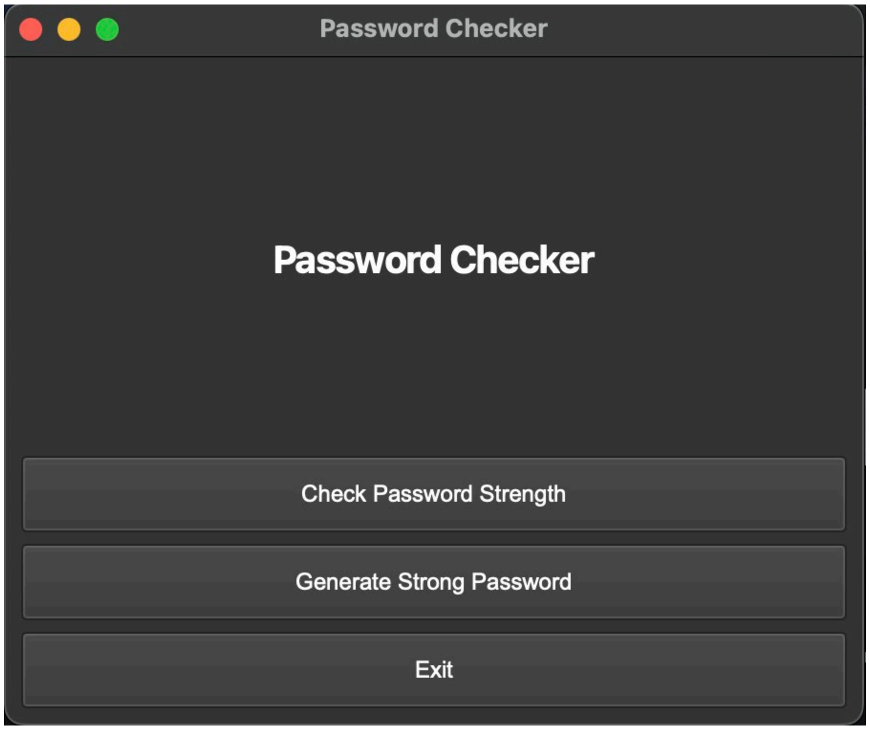
Main Window

The main interface with buttons for all functionalities.

(Windows)



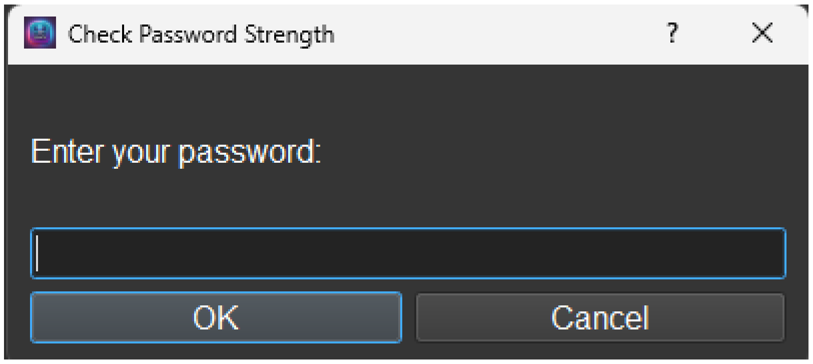
(MacOS)



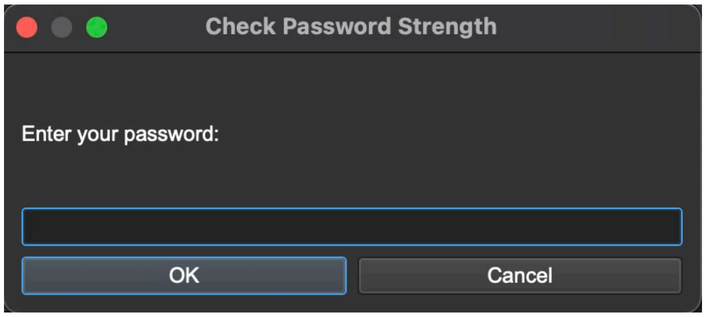
Check Password Strength

Dialog for entering a password and viewing strength analysis.

(Windows)



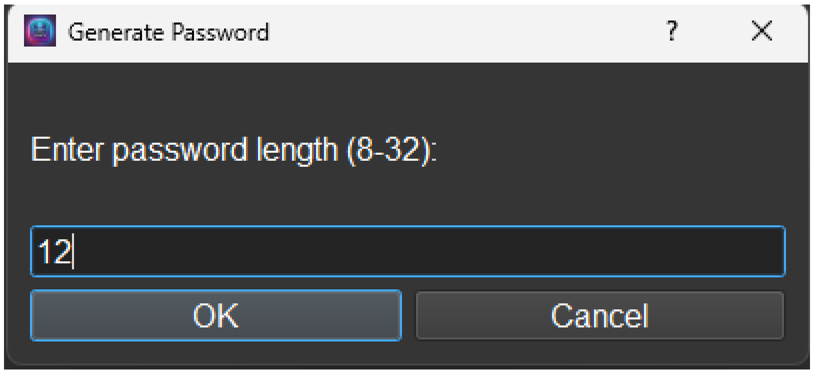
(MacOS)



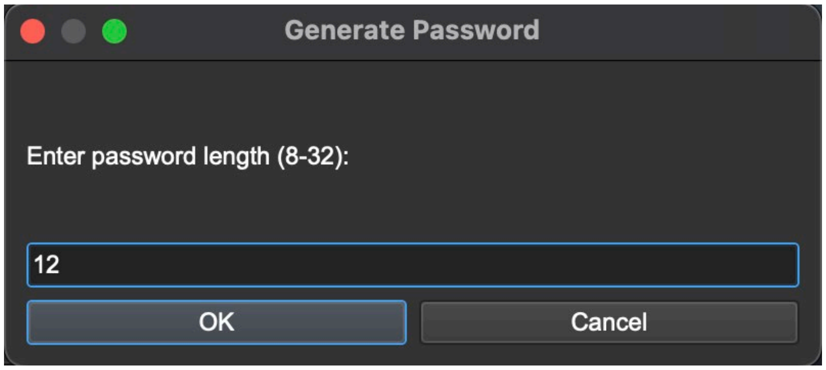
Generate Password

Dialog for generating a strong password and copying it to the clipboard.

(Windows)



(MacOS)



Future Enhancements

- Add support for customizable password generation rules (e.g., exclude certain characters).
- Implement a password history feature to save previously generated passwords.

- Enhance the interface with animations or transitions for better user experience.

Author

- Developed by Bruce Matrix

License