Lab Assignment 3

Password checker

Maros Cuninka 07.01.2023

GitHub repository: https://github.com/mcuninka/PasswordChecker-Generator.git

1. Introduction

The task of this project was to build a program that :

a) Checks the complexity of a password obtained by a user and displays all requirements which are not met.

b) Generates a random strong password that meets all requirements.

c) Receives a text file with passwords and generates a new text file that will list all requirements that are not met for each password.

d) Generates a list of random weak passwords, the number of passwords is provided by a user.

Password requirements:

- Password should have at least a length of 12 characters
- Password should contain at least 1 number
- Password should contain at least 1 special characters
- Password should contain at least 1 upper case letter
- Password should contain at least 1 lower
- Password does not exist in the list of the top 100 common passwords https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-100.txt

2. Setup

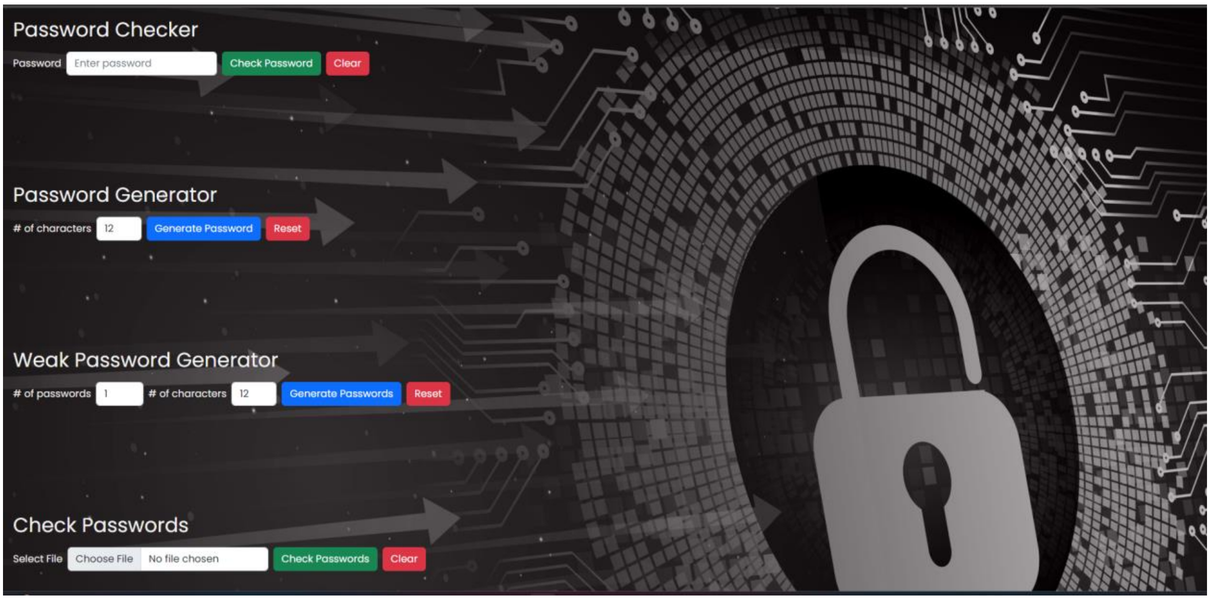
I have created a single-page web application, Python with flask was used for the back end, and React JS for the front end. In order to run the server, you must have python installed on your machine, together with requests and flask libraries (pip install requests & pip install flask). For the client, make sure to have npm installed on your computer.

To start the application, navigate yourself to the flask_server directory (cd flask_server) and start the server by executing ‘python3 main.py’. Afterwards in the different terminal, navigate yourself to the client directory (cd client) and run ‘npm start’. Before running the client, run npm ci or npm i to install all necessary node modules. Once you have all node modules, you do not have to run these commands again and you can just start the client with ‘npm start’.

```
cd flask_server -> python3 main.py
cd client -> npm ci -> npm start
```

3. Implementation

In Figure 1 you can see the whole application. It is divided into 4 parts, one part for each task. Since this was the whole functionality of the application, I decided to have it on a single page, instead of having a navigation bar, to improve user experience.



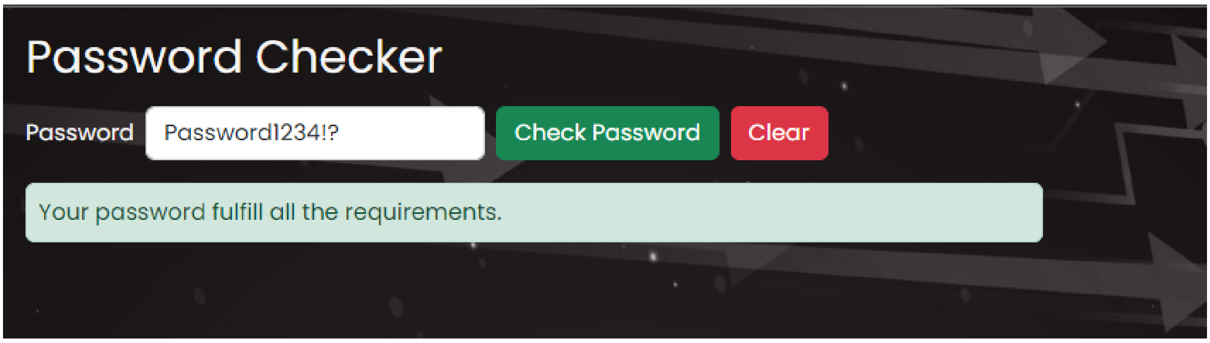
# of characters 12

# of passwords 1 # of characters 12

Figure 1 Password checker web application

3.1. Password Checker

The first part consists of a password checker. A user can type in an arbitrary password and check its complexity by pressing the green check password button. If the password meets all requirements, a success status message is shown (Figure 2).

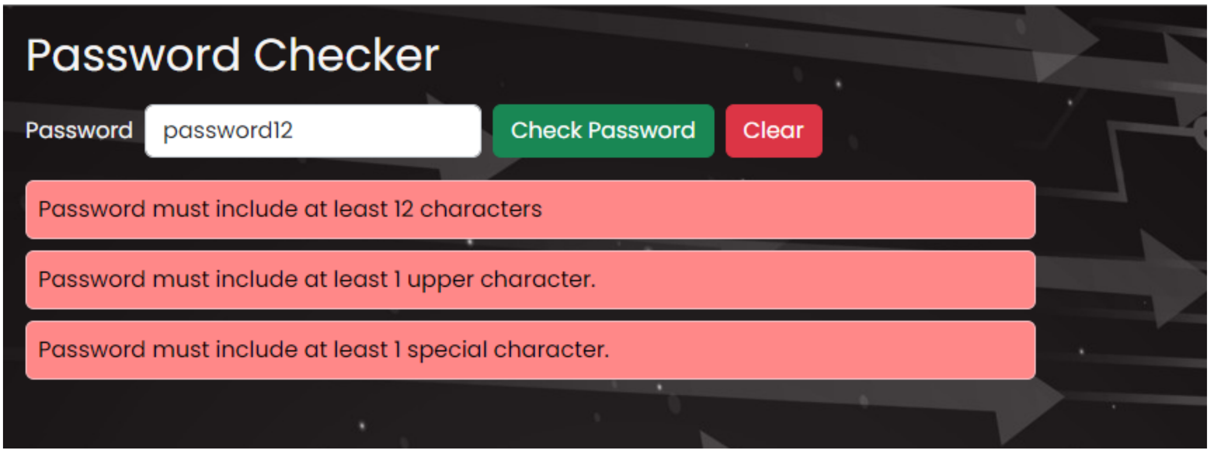


Password Password1234!?

Your password fulfill all the requirements.

Figure 2 Password with all requirements

On the other hand, if it does not meet all the requirements, an error message for each not met requirement is displayed as shown in Figure 3. Additionally, there is also a clear button that clears the user input and all status messages.



Password password12

Password must include at least 12 characters

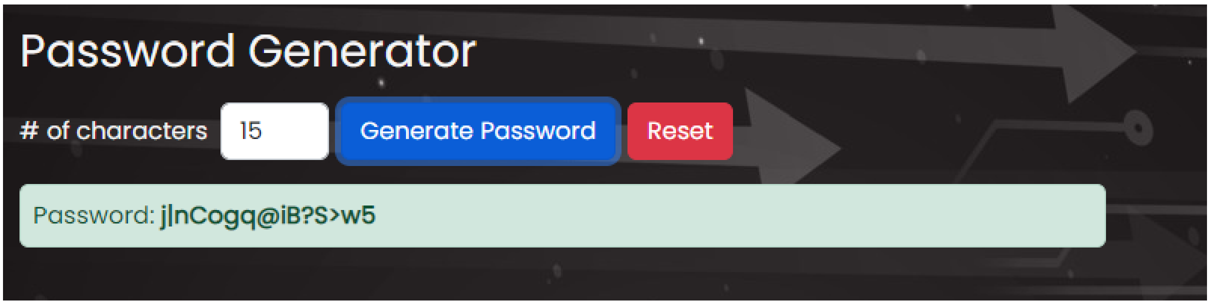
Password must include at least 1 upper character.

Password must include at least 1 special character.

Figure 3 Short password with missing 1 upper and special character

3.2. Strong Password Generator

The next part is a password generator, which generates a strong password fulfilling all requirements. A user can select how many characters the password should have and then after pressing the blue generate password button a randomly generated password is displayed (Figure 4).

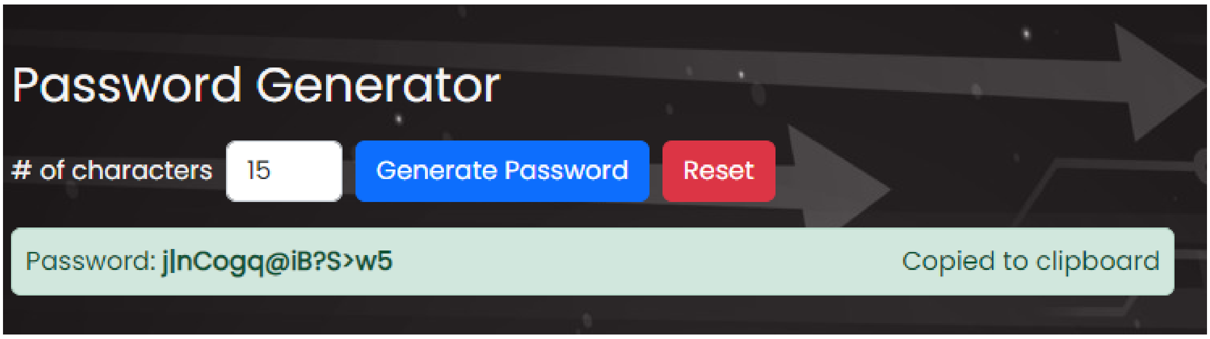


# of characters 15

Password: jlnCogq@iB?S>w5

Figure 4 Randomly generated strong password

There is also a click-to-copy functionality where a user can click anywhere in the green rectangle to copy the randomly generated password. After clicking, a message on the right is displayed (Figure 5).



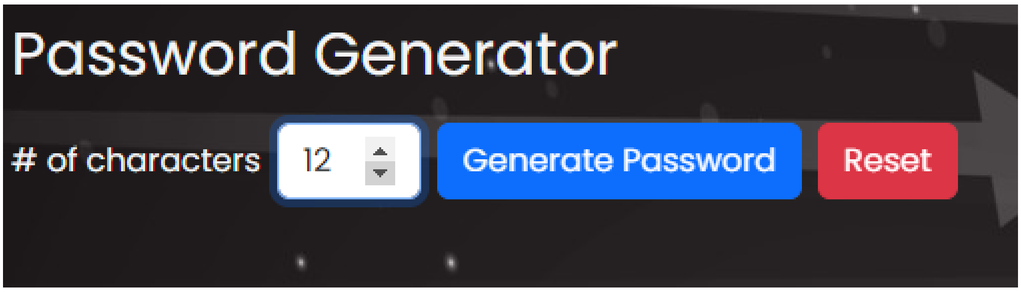
# of characters 15

Password: jlnCogq@iB?S>w5

Copied to clipboard

Figure 5 Click to copy functionality

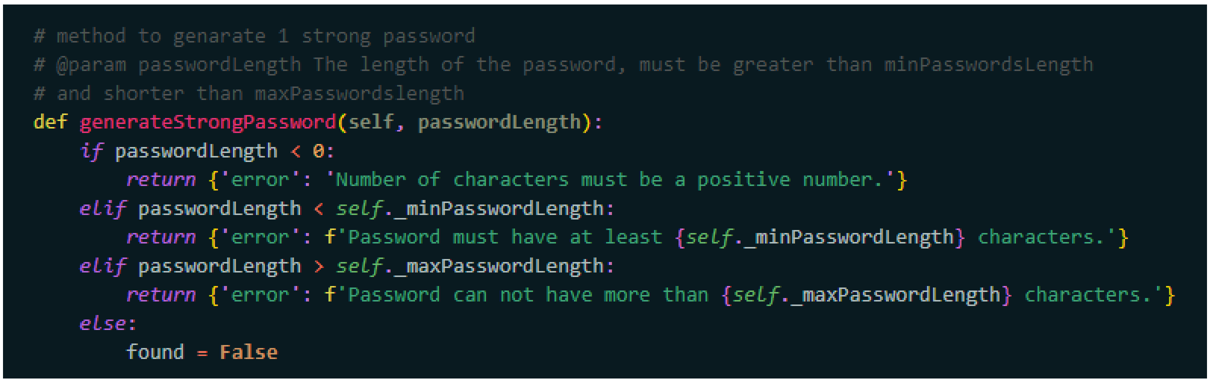
One of the requirements was that the minimum length of the password is 12 characters. Therefore, the default value for the input of a number of characters is set to 12. The maximum is set to 50 for now. The input field does not allow a user to go below 12 or above 50 by pressing the down/up arrow or by pressing the down/up arrow key on a keyboard (Figure 6). However, for testing purposes of the server, it is possible to manually enter a value below 12 by typing the number directly in the input field. In that case, an error message saying 'Password must have at least 12 characters.' is displayed. If a user tries to enter a negative value, the value is converted directly to a positive value. The same applies to a value above 50, after entering a value higher than 50, it is immediately changed back to 50.



# of characters 12

Figure 6 Limit for number of characters

Even though a user cannot enter a value below 12 and above 50 on the front end, there is also a check on the back end to prevent any misbehaving of the system (Figure 7). Finally, a reset button is implemented, which clears status messages and resets the number of characters back to its default value – 12.

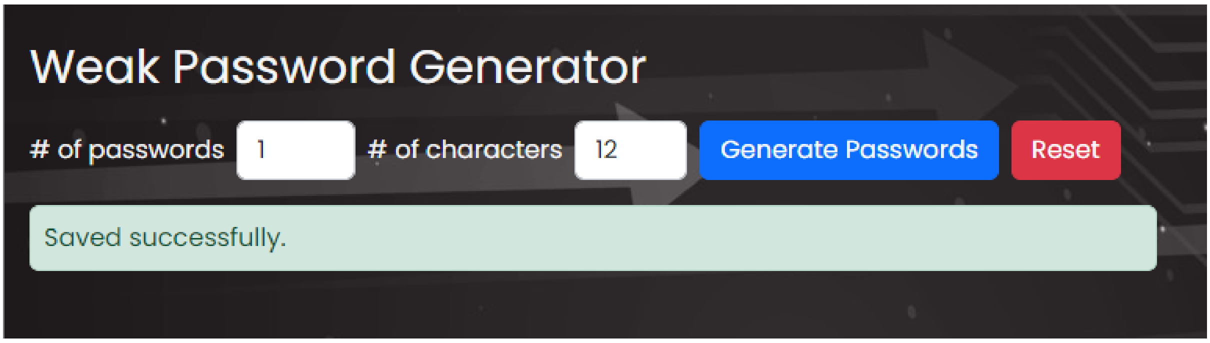


```python
# method to genarate 1 strong password
# @param passwordLength The length of the password, must be greater than minPasswordsLength
# and shorter than maxPasswordslength
def generateStrongPassword(self, passwordLength):
    if passwordLength < 0:
        return {'error': 'Number of characters must be a positive number.'}
    elif passwordLength < self._minPasswordLength:
        return {'error': f'Password must have at least {self._minPasswordLength} characters.'}
    elif passwordLength > self._maxPasswordLength:
        return {'error': f'Password can not have more than {self._maxPasswordLength} characters.'}
    else:
        found = False
```

Figure 7 Method to generate a strong password on back end

3.3. Weak Passwords Generator

The third part is focused on a generator of weak passwords, which means that at least one of the requirements is not satisfied. A user can select how many passwords he/she wants to generate and what should be the length of those passwords. Currently, the maximum number of passwords is set to 100 and the maximum number of characters is the same as in the previous case – 50. The minimum for both inputs is set to 1. If a user tries to type in a negative value, the value is converted to the positive representation of that value and if a user tries to type in a value greater than the maximum, the value is changed to the max value. After pressing generate passwords button while having both input fields with correct values a green success message is displayed (Figure 8), and the txt file is saved in the current directory of the back end. This could be later modified to allow users to save it anywhere on a computer. In a case of an error, a red error message is displayed describing the problem. Lastly, the reset button clears all status messages and resets the number of passwords and the number of characters inputs back to their default values.



Weak Password Generator

# of passwords 1

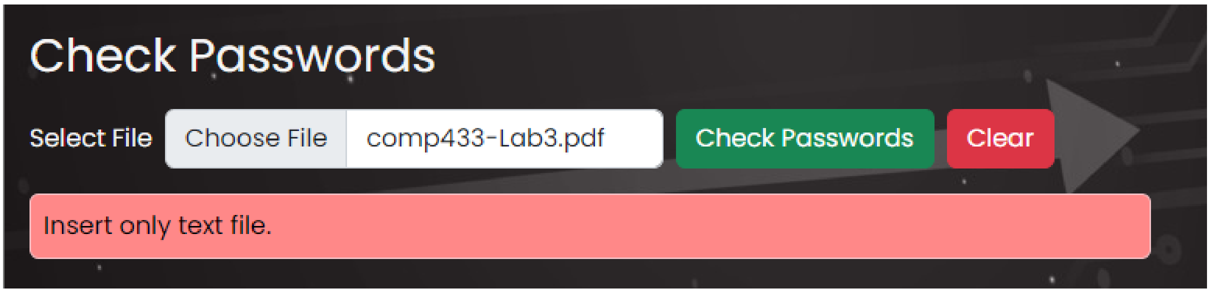
# of characters 12

Saved successfully.

Figure 8 Weak password generator

3.4. Check Passwords from the File

The last functionality of the application is the possibility to upload a txt file with passwords, one on each line, to check their complexity. Then a new txt file is created and saved in the current directory of the server with all passwords and all the requirements which are not met per each password. The back end only allows txt files, so if a user tries to upload any other document, an error message describing the problem is displayed (Figure 9). The clear button clears all status messages and the file input.



comp433-Lab3.pdf

Insert only text file.

Figure 9 Wrong file type for check password functionality