**LAB #8 – Python Web Page Code (Password Update form, logs and**

**cryptographic algorithms)**

Name - Herman Mann

Course – SDEV 300

Date – 05/10/2022

Table of Contents

[**Test Plan** 3](#_Toc103011419)

[**Test Case # 1 – Checking and testing the Password update form creation** 3](#_Toc103011420)

[**Test Case # 2 – Checking and testing of inputting a wrong password for username and checking the log** 5](#_Toc103011421)

[**Test Case # 3 – Checking and testing of entering new password and then entering a different password after** 7](#_Toc103011422)

**Part 2 - Using the Decrypting Secret Messages sites found in this week’s readings, decrypt the following messages……………** 10

**Pylint results for** **Python Web Page Code Program (Password Update form, and more) …**………………….………………………………………….17

# **Test Plan**

|  |  |
| --- | --- |
| **Test Case # 1 - Checking and testing the Password update form creation** | |
| **Input** | cd C:\Users\herma\Desktop\sdev300\_lab8\_additionalUpdatesto\_lab7  set FLASK\_APP=website\_app.py  flask run  Old Password: SoftwareDevelopment45@  New Password: Cybersecurity45#  Re-enter New Password: Cybersecurity45# |
| **Expected Output** | The website\_app server starts successfully and does not encounter any errors/issues  Gives server link upon successful execution: <http://127.0.0.1:5000>  Successfully opens login template of the login form page. User will login and then will be given the option at the top left-corner to Change Password. User will click on “Change Password” button and the change\_update\_password form will be brought up successfully for the current user to change his/her password to a new one. |
| **Actual Output** | The website\_app server starts successfully and does not encounter any errors/issues  Gives server link upon successful execution: <http://127.0.0.1:5000>  Successfully opens login template of the login form page. User will login and then will be given the option at the top left-corner to Change Password. User will click on “Change Password” button and the change\_update\_password form will be brought up successfully for the current user to change his/her password to a new one. |
| **Pass?** | **YES** |
| **Screenshots**  **A message**  **“Password has been Updated” will be displayed on index page after successful new password creation.** | Graphical user interface, text, application  Description automatically generated  Graphical user interface, application  Description automatically generated |

|  |  |
| --- | --- |
| **Test Case # 2 - Checking and testing of inputting a wrong password for username and checking the log** | |
| **Input** | cd C:\Users\herma\Desktop\sdev300\_lab8\_additionalUpdatesto\_lab7  set FLASK\_APP=website\_app.py  flask run  Username: german123  Password: sdev300 |
| **Expected Output** | The website\_app server starts successfully and does not encounter any errors/issues  Gives server link upon successful execution: <http://127.0.0.1:5000>  Goes to the Login form, and user enters a wrong password for username and message gets displayed:  Invalid username or password entered.  Then the login attempt gets logged in the “user\_failed\_logins” file as a failed login attempt by the user. |
| **Actual Output** | The website\_app server starts successfully and does not encounter any errors/issues  Gives server link upon successful execution: <http://127.0.0.1:5000>  Goes to the Login form, and user enters a wrong password for username and message gets displayed:  Invalid username or password entered.  Then the login attempt gets logged in the “user\_failed\_logins” file as a failed login attempt by the user. |
| **Pass** | **YES** |
| **Screenshots** | Graphical user interface  Description automatically generated |

|  |  |
| --- | --- |
| **Test Case # 3 – Checking and testing of entering new password and then entering a different password after** | |
| **Input** | cd C:\Users\herma\Desktop\sdev300\_lab8\_additionalUpdatesto\_lab7  set FLASK\_APP=website\_app.py  flask run  Old Password: Cybersecurity45#  New Password: PythonProgramming45@  Re-enter New Password: JavaProgramming45@ |
| **Expected Output** | The website\_app server starts successfully and does not encounter any errors/issues  Gives server link upon successful execution: <http://127.0.0.1:5000>  A message gets displayed “The new passwords do not match” after user enters one password for entering a new password and then re-entering a different password. Works successfully and as expected. |
| **Actual Output** | The website\_app server starts successfully and does not encounter any errors/issues  Gives server link upon successful execution: <http://127.0.0.1:5000>  A message gets displayed “The new passwords do not match” after user enters one password for entering a new password and then re-entering a different password. Works successfully and as expected. |
| **Pass** | **YES** |
| **Screenshots**  A message, “The new passwords do not match” gets displayed after user enters a different password in “New Password” box and a different password in the ”Re-enter New Password” box. | **Graphical user interface, text, application  Description automatically generated**  **Graphical user interface, text, application  Description automatically generated** |

**Part 2 - Using the Decrypting Secret Messages sites found in this week’s readings, decrypt the following messages**

a.

- .... .. ... / ... -.. . ...- / ...-- ----- ----- / -.-. .-.. .- ...

... / .... .- ... / ... --- -- . / ... - .-. .- -. --. . / .-. . --.-

..- . ... - ... .-.-.-

Decoded Message: THIS SDEV 300 CLASS HAS SOME STRANGE REQUESTS.

Should be: THIS / SDEV / 300 / CLASS / HAS / SOME / STRANGE / REQUESTS.

Cipher: Morse Code

So, I got the following above after decrypting the message. The Cipher is Morse Code, and the overall decoded/decrypted message is as represented above.

Link: <http://rumkin.com/tools/cipher/morse.php>

Graphical user interface, text, application

Description automatically generated

b.

U28gdGhpcyBpcyBiYXNlNjQuIE5vdyBJIGtub3cu

Decoded Message: So this is base64. Now I know.

Cipher: Base64

So, I got the following above for part b after decrypting the overall message. The cipher is Base64, and the decoded/decrypted message is as represented above. I found the website by entering “guess the encryption” and then searched the first website that came across my eyes which was the one below. I tested the requirement by first entering the cipher text, and then selecting the type of cipher it was coming close to and went with that and finally got the expected result I was looking for part B.

Graphical user interface, text, application, Teams

Description automatically generated

Link: <https://www.boxentriq.com/code-breaking/base64-decoder>

Graphical user interface, application, Teams

Description automatically generated

c.

--- Psuwb Ysm ----

W oa gc qzsjsf. Bc cbs qcizr dcggwpzm twuifs hvwg cih.

--- Sbr Ysm ---

Decoded Message:

--- Begin Key ----

I am so clever. No one could possibly figure this out.

--- End Key ---

Cipher: Rotational Cipher (Referred as ROT13)

So, I got the following for part c after decrypting the message. This was a challenging message overall to decrypt and had to use another external resource besides the one I used to decrypt the overall message.

Link: <https://www.dcode.fr/caesar-cipher>

Graphical user interface, text, application

Description automatically generated

I used a hint from the other website to find this website to decrypt the main overall message, as it was extremely challenging. Since I figured and found out that this specific message was a “Rotational Cipher” I searched Rotational Cipher and found this website. Then, I decrypted the message of part c on this website directly.

Link: <https://rot13.com/>

Chart

Description automatically generated

Discussing the log file, without the file it would be extremely easy to steal a password and hack the account, so it is extremely important to keep track of failed logins in a separate file of a username that is currently in session.

Pylint results for Python Web Page Code Program

Graphical user interface, text

Description automatically generated with medium confidence

Throughout the completion of this lab which focused on adding a little more Flask Functionality such as the creation of the login form, the registration form, and the table and images for the website I got to expand my further knowledge in the content areas of Python Flask, HTML, CSS, and using more advanced Python language functionality. Moreover, it was great to understand how to create a password complexity for the login form of my website as I did not know how to do this before and showed me how to be more prepared for the near future for creating more forms in HTML and CSS and using Python programming functionality itself. Discussing my PYLINT results in great depth, I had a lot of “trailing whitespace” issues which dealt with the Convention section of errors, I had 1 refactor error which was not using “with” for opening a file I just did “open (NEW\_PASS\_FILE, "w").close()” and it should’ve been “with open(NEW\_PASS\_FILE, "w") as file:

file.close()”

Also, I encountered the error of “Unable to import ‘passlib.hash’” as I was using statements from this import file and I was very confused and surprised why I was unable to import this specific file in my Python environment. I still had the issue towards the end, and I could not find any ways to come about and fix the error overall. In the future for the last lab I look forward to fixing this major issue and getting it solved so my PYLINT score will result in a perfect “10.00/10”. Overall, I tried fixing the PYLINT score, but could not and therefore had to end up with the same score of a “9.58/10”. So, I still had a lot of difficulty for some reason to import “passlib.hash”.