Advanced Software Engineering
Secure Password Management System

Team Matrix
Github: https://github.com/RohanKumarSachdeva/COMSW4156-TeamMatrix

Mausam Sanjay Agrawal - msa2213

Nirvishi Shrivastava - ns3499

Rishabh Gupta - rg3334

Rohan Kumar Sachdeva - rks2158

Part 1:

Mentor: Claire (Yumeng) Luo {yl4655}
Date of meeting: 23rd October 2021

Part 2:

Functionality:
Our service is a secure password generation and management service. We let the users manage the passwords of the various accounts they possess. Specifically, we enable the user to do the following:

1. Authenticate and login to our service for accessing user specific information.
2. Create a password for their application or use our service to generate one which is then saved in encrypted format.
3. Retrieve, update and delete previously generated passwords.
4. In case the user provides us with a password, we let them know how secure it is and suggest they use a stronger one.

Users:
The following types of users can make use of this service:

1. Users who do not want to manage/remember the passwords of the various accounts they have.
2. Users who are concerned about storing passwords in plaintext on their own system.

3. Users who do not want to go into the hassles of creating a secure password by themselves.
4. Users who want to know how secure their passwords are.

Data Creation:
Our tool will require each user to register and login before using our services for which we will be using the OAuth protocol from Auth0 API to authenticate the users. The user’s passwords along with the applications that they correspond to will be saved in our database. **The passwords will be saved in encrypted format to avoid leakage of any sensitive information**.

User Stories:

1. As a user who doesn’t want to manage remembering or storing my passwords, I want to offload this task to a service. My conditions of satisfaction are:
    a) I would like the flexibility of providing my own password.
    b) My password should be handled in a secure fashion.
    c) My password shouldn’t be visible to other users.

2. As a user who has previously saved passwords on this management service, I want to access/update them. My conditions of satisfaction are:
    a) I must receive the password for the specific application I am looking for.
    b) I can update or delete that particular password if I wish to.

3. As a user who is concerned with creating weak passwords, I want to know if my current password is weak or not. My conditions of satisfaction are:
    a) I should be informed if my password is weak.
    b) I should receive recommendations for a strong password.

External APIs used:

1. Auth0: We will use this for authenticating users for our service.

2. Password generation: We will be using an external API to create a randomly generated secure password. For now, we are planning to use the API described here: https://github.com/fawazsullia/password-generator/. This API accepts inputs such as the password length, choice of types of characters and case. We might switch to a better API if we find one.

3. Password Strength: We will be using an API to check how secure a password is. For now, we are considering to use the following API, https://haveibeenpwned.com/API/v2#APIVersion which informs us if the provided password was part of any data breach. This could also change if we find an API which better suits our purpose.

Part 3:

Demonstration:

- User visits the webpage
- User registers/logs in to our application
    - This will ensure that each client's data is protected from other clients
- User selects an option of creating password or generating password
- If create password is selected, the user provides the application name and his own password which is validated based on the following conditions:
    - Minimum length should be 8 characters
    - Minimum kinds of characters (1 uppercase, 1 lowercase, 1 number & 1 special character)
    - We validate the password strength
- If generate password is selected, the user will give the application name and a password will be generated by the service
- After creation/generation of password, we encrypt it and save it in the database
- User can retrieve the password for a particular application or all passwords for his entire list of applications

- User can choose to update existing password of any application ensuring that the updated password meets the above conditions
- User can choose to delete any application name and its corresponding password from his list of passwords
    - For retrieve/delete/update operations on the password of a particular application, we check if the app name provided by the user actually matches with the entries for that user in the database and service the request accordingly (cover cases like empty or invalid app names)

Testing:

Here, the tester is a user making calls to our API from Postman or making curl calls from a terminal

- User login and the associated distinguishing between clients
    - The tester registers as a user
    - The tester provides the same password that was used during registration, otherwise it is barred access to the service
    - This protects the data of a client from other clients using the service
    - The tester can distinguish between clients using the token/key that comes on successfully validating against the OAuth API

- Password Creation/Generates & Corresponding Data Persistence
    - The tester logs in as a user
    - The tester generates a password for application 'abc'
    - The tester logs out
    - The tester returns back and validates against the OAuth API
    - The tester then requests for the password of application 'abc'
    - The tester will pass this test if it is able to retrieve the password and it matches with the one that was originally generated

- Password Update
    - The tester retrieves the list of application names for which it has passwords stored
    - The tester updates the password for one of those apps

    - On subsequent password retrieval request for the same app, the tester will pass the test if it receives the updated password

- Password Deletion
    - The tester retrieves the list of application names for which it has passwords stored
    - The tester deletes the password for one of those apps
    - On subsequent password retrieval request for the same app, the tester will pass the test if it receives an error message from the service that the password doesn't exist anymore

Part 4:

Frameworks/Libraries: Python project built using Flask
Style Checker: Flake8
Static Analysis Bug Checker: Pychecker/Pylint
Test Runner and Coverage Tracker: unittest & coverage python libraries