Saint Augustine’s College, Sydney

**Saint Augustine’s College, Sydney**

**Software Engineering Year 11: Object Oriented Programming**

Cruz Leung “Password Strengthener” project

Table of Contents

[Task Definition 4](#_Toc201445624)

[Hardware and Software Requirements 5](#_Toc201445625)

[Storyboard 6](#_Toc201445626)

[Algorithm Design 7](#_Toc201445628)

[Password checker 7](#_Toc201445629)

[Flowchart 8](#_Toc201445630)

[Implementation 9](#_Toc201445631)

[GitHub Repository URL 9](#_Toc201445632)

[Testing 10](#_Toc201445635)

[Project Reflection 12](#_Toc201445636)

# Planning

## Task Definition

I have been assigned the task of developing a front-end and back-end Python application to address a common cybersecurity issue: weak passwords. The application, named "Password Strengthener", is a user-friendly webpage designed to check and validate passwords, such as detecting breached passwords, common passwords, or performing standard password validations. The effectiveness of a password is often undermined by its simplicity, predictability, and vulnerability to breaches. By generating effective feedback for positive password reinforcements, "Password Strengthener" strives to teach users the ability and skill to construct a strong and effective password, contributing to overall cybersecurity awareness. The application also aims to mitigate the risk of poorly designed passwords by implementing a strong password generator to create passwords that are difficult to decipher, providing users an easy way to create secure passwords.

The four simple, core functionalities of the application are as follows:

* Password Checker: Programmed using professional methods, my password checker allows users to check their password strength, as well as getting instant feedback for security improvements.
* Breach check: Powered by Troy Hunt’s HaveIbeenPwned, the function allows users to check if their passwords have been involved in data breaches.
* Password generator: My tool will produce 12-character passwords that are a mix of lower- and upper-case alphanumeric characters, digits, and symbols to ensure complexity and strength.
* Save to clipboard: Users can copy the generated password directly to their clipboard for easy use when creating or updating their accounts.

## Hardware and Software Requirements

In order to develop LockSmithy, my development environment will need to meet the following hardware and software requirements.

|  |  |
| --- | --- |
| Software | Requirements |
| Operating System | Any OS compatible with Python; Windows, macOS, or Linux |
| Python Version | Designed with **Python 3.12.4**, compatible with **Python 3.0 and above**. |
| GooeyPie Framework | Latest compatible version with Python 3.x |
| Python Package Manager | PIP for installing GooeyPie and any other dependencies |
| IDE | Visual Studio Code |
| Version Control | Git for version control and GitHub for repository hosting |

|  |  |
| --- | --- |
| Hardware | Requirements |
| Processor | Intel Core i3 (8th Gen or later) / AMD Ryzen 3 or better |
| Memory | 4GB RAM or higher for optimal performance |
| Storage | Minimum of 1GB free space for project and software |

## Storyboard

This storyboard presents a preliminary design of the user interface planned for development utilizing Python 3.12.4 and the [Gooeypie](https://www.gooeypie.dev/start) framework. The envisioned application is presented featuring a user-friendly interface allowing users to enter passwords and receive feedback for improvements. An integrated common password and breached password check is implemented to maximise strength and reduce credential stuffing. A password generator tool is also included, allowing users to smoothly create a secure password. The about screen is designed to offer concise information about the software, highlighting usage information and app version.

## A screenshot of a computer AI-generated content may be incorrect.

This diagram was created and generated using GooeyPie Prototyping Template on [Powerpoint](https://saintaugsydney-my.sharepoint.com/:p:/g/personal/pleung_student_saintaug_nsw_edu_au1/Eeb0KVjCsc1GgSLgrumvAVwB9E_cRHptRJBKDSQKcxNeIA?e=7wtS85), and edited with .drawio. A copy of template can be found [here](https://saintaugsydney-my.sharepoint.com/:i:/g/personal/pleung_student_saintaug_nsw_edu_au1/EcsS1-rxaeNKujbltXg_g04BUwy7LSzYqyRhXfDlER9ucw?e=0qqxFd).

## Algorithm Design

The provided pseudocode delineates the Pass strength check tool. It constitutes the program's most intricate algorithm, involving logic, validations and cross-site checks to assess the password according to programmed methods.

## Password checker

FUNCTION check\_password:

SET password = user input

INITIALIZE score = 0

INITIALIZE lists: required\_components, weakness\_feedback, feedback

INITIALIZE major\_weakness\_count = 0

INITIALIZE requirement\_fail = False

IF password is empty:

DISPLAY "No Password" and return

IF length >= 14:

score += 4

ELSE IF length >= 12:

score += 3

ADD suggestion to feedback

ELSE IF length >= 10:

score += 2

ADD suggestion to feedback

ELSE IF length >= 8:

score += 1

ADD suggestion to feedback

ELSE IF length >= 6:

ADD weakness message

INCREMENT major\_weakness\_count

ELSE:

ADD required component

SET requirement\_fail = True

IF password contains digits:

score += 1

ELSE:

ADD suggestion to feedback

IF password contains uppercase letters:

score += 1

ELSE:

ADD suggestion to feedback

INCREMENT major\_weakness\_count

IF password contains lowercase letters:

score += 1

ELSE:

ADD suggestion to feedback

IF password contains special characters:

score += 1

ELSE:

ADD suggestion to feedback

CALL check\_common\_pwds, return requirement\_fail and required\_components

CALL check\_dictionary\_words, return requirement\_fail and required\_components

CALL repeated\_pattern\_check, return major\_weakness\_count and weakness\_feedback

CALL check\_password\_pwned, return breach\_count

CALCULATE score

IF password is breached:

score = 0

DISPLAY "Breach Status: Breached"

ELSE:

DISPLAY "Breach Status: Not Breached"

CALL strength\_status(score)

CALL progress\_bar\_update(score)

DISPLAY required\_components

DISPLAY weakness\_feedback

DISPLAY feedback

## Flowchart

The following flowchart was created to display the main functions and logic of the password strength checker and the scoring system. Additional functions such as the GUI implementations and password generator was excluded.

A diagram of a company

AI-generated content may be incorrect.

This diagram was created and generated with .drawio. A copy of template can be found [here](https://saintaugsydney-my.sharepoint.com/:i:/g/personal/pleung_student_saintaug_nsw_edu_au1/Ebm2_0AlL6FAiEbWogSu_M4BZehyFPFTbcn7Tr0OnYvowg?e=HrNHeS).

# Implementation

## GitHub Repository URL

<https://github.com/Cruz-Leung/pwd_checker.git>

## 

## A screenshot of a computer program AI-generated content may be incorrect.

*This GitHub README.md was created using* [*readme*](https://readme.so/)*.*

# Testing

The below test table was created after production of code to test for bugs and issues in the code.

If found additional bugs or queries, please report to [cruzczuchry@gmail.com](mailto:cruzczuchry@gmail.com).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Input Password** | **Rationale** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Screenshot** |
| TC 1 | Password | Common, short, dictionary word, functional test | * Breached * fails many checks * displays never use this password * found in common passwords * found in dictionary words | Same as expected | Pass | A screenshot of a computer  AI-generated content may be incorrect. |
| TC 2 | Empty password “” | Test how app handles no input | * Status: no password * Progressbar = 0% * Feedbacks display null | Same as expected | Pass | A screenshot of a computer  AI-generated content may be incorrect. |
| TC 3 | P@ssw0rd123 | Common variation with required types, better mix | * Breached * Sequential pattern warning * Displays never use | Same as expected +  Repeated pattern detected | Fail | A screenshot of a computer  AI-generated content may be incorrect. |
| TC 4 | P@ssw0rd123 | Retest on fixed code | * Breached * Sequential pattern warning * Displays never use * Doesn’t have repeated pattern detected | Same as expected | Pass |  |
| TC 5 | abcabcabc | Repeated sequence pattern | * Detect repeated sequence * Breached * Many failed checks * Display “Never use” | Same as expected | Pass | A screenshot of a computer  AI-generated content may be incorrect. |
| TC 6 | ThisIsA$StrongPass197! | Long, diverse, possible crash or miscalculati-on of score | * Very strong * No warnings | Same as expected | Pass | A screenshot of a computer  AI-generated content may be incorrect. |
| TC 7 | Aksdhfksd@jsdh5230dsgsdgsdsd26 | Very long for possible crashes, 30 characters | * Repeated sequence * Strong but not very strong | Same as expected | Pass |  |
| TC 8 | "A!b2C@d3E#" | Very complex, strong | * Strong * Suggests more characters * No major weakness | Same as expected | Pass | A screenshot of a computer  AI-generated content may be incorrect. |
| TC 9 | QWErty123! | Test API for breach, looks complex but isn’t | * Breached * Display never use * Repeated sequence * Common pwds | Same as expected | Pass |  |
| TC 10 | Z1! | Very short with multiple character types | * Displays never use * Fails many checks * Length fail | Same as expected | Pass | A screenshot of a computer  AI-generated content may be incorrect. |

# Project Reflection

My planning phase of the algorithm began withdrawing multiple physical sketches and storyboards of my user-interface (GUI) for my application. This was different from the last assessment where I began with direct coding, as were my previous experiences with long projects. By learning through challenges and problems I faced last project, I was able to plan out an initial design before beginning to code for the GUI using a Gooeypie library. This gave me the ability to code with a vision and plan, whereas I was met with confusion and many errors as my code sized increased in my previous projects. However, I was still met with algorithmic and logic problems as I began my code for logic and the algorithm of the Password Checker, as I began with direct coding for this part. This helped me realize that charts such as flowcharts should be created and planned out before the project, in order to have a structured plan and direction to my heavy and confusing algorithmic code.

At the start of my coding journey for this project, I was met with difficulties such as familiarizing with Gooeypie, a GUI library that I’ve never used. With millions of online resources available, I was able to watch and learn professionals use GUI libraries, especially Gooeypie, which greatly assisted me with my GUI code. My drawn-out storyboards were also able to assist me in the direction of my designs, giving me a structured plan to design my user-interface.

I also encountered confusion with the many validation functions in my code, such as the common password check, dictionary check. I was quite unfamiliar with the concept of return values and function parameters, which was crucial to create a complex password checking tool. Overtime, I started to be more familiar with Python and the functions, particularly in calling out functions with the use of different parameters, as well as creating and iterating through large documents and files, such as txt and csv files. By breaking down each function one by one and connecting it with my main function, I was able to resolve any system and syntax errors within my code.

Lastly, I encountered many problems and errors when trying to implement a HaveIbeenPwned API, where server limitations and restriction, with the combination of poor school network and strict firewall, made my code extremely slow, and unusable. After many attempts of different APIs and servers, I was still left with an extremely slow application, where my API still only worked for about 40% of the time. In the end, I was able to get assistant by my teacher, Mr Fong, who helped me find an API that doesn’t require “pip installs”, allowing my code to reach the server without needing to use my device as a user-agent. To my surprise, my code worked very smoothly and accurately with the new API, and I was able to continue with my other goals of the projects after.

The GitHub repository management was another rewarding aspect of the project, particularly with the utilization of readme.io. This provided me a good base for efficient documentation throughout the project, which gave me a good, structured idea on how I wrote my code before. Not only was I able to retrace deleted code that I needed to reuse, it also greatly improved efficiency on future bug fixes and patch notes on earlier codes. However, I was met with various GitHub repository issues where I was unable to create a new repository, as well as locating my new files into it. This led to some earlier codes to be saved in my repositories for my previous project, which was annoying and difficult to retrace. This project allowed me to be more familiarize with the inner workings and the usage of GitHub, skills that can significantly assist me in my future projects.

The addition of the pseudocode and test tables were also very refreshing and effective. As I’ve never written any pseudocode before, I was able to learn and begin writing pseudocode for the first time. Although it didn’t contribute to my overall code writing during the project, I can now begin my planning stage using pseudocode instead of direct coding in my next project, which could help me reduce any confusion that I might have in my algorithm. The test table was very effective in my test and debugging stage. Before testing, I was confident that I have created a bug free application. However, I was met with various hidden and underlying errors and bugs when I tried testing it with different and unorthodox inputs. This allowed me to resolve many bugs and potential errors that could’ve been prompted to the user, had I publicly released an untested code. This was a great learning opportunity for me, as I will begin to incorporate a test table in my future projects in order to achieve a bug-free application.

My proficiency and skill in Python and designing GUIs, particularly with Gooeypie, have advanced and developed considerably through this project. I have mastered complex algorithmtic structures, as well as incorporating code into a front-end web application. With my new understanding of user-interface designs and the usage of functions, I am looking forward to exploring beyond the confines of a functional approach next term.