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

[Planning 3](#_Toc150638067)

[Task Definition 3](#_Toc150638068)

[Hardware and Software Requirements 4](#_Toc150638069)

[Storyboard 5](#_Toc150638070)

[Data Dictionary 7](#_Toc150638072)

[Algorithm Design 8](#_Toc150638073)

[GANTT Chart 9](#_Toc150638074)

[Implementation 10](#_Toc150638075)

[GitHub Repository URL 10](#_Toc150638076)

[Testing 11](#_Toc150638077)

# 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 pwned 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 this risk 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.

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=TaHWaK).

# Implementation

## GitHub Repository URL

# 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, | * 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. |