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

## 

## Algorithm Design

The provided pseudocode delineates the method for the generatePassword method. It constitutes the program's most intricate algorithm, involving string operations and iterative constructs to assemble the password according to specified parameters.

#### generate\_password

BEGIN generate\_password(length, use\_lowercase, use\_uppercase, use\_numbers, use\_special)

SET characters = ''

SET password = ''

IF use\_lowercase THEN

APPEND 'abcdefghijklmnopqrstuvwxyz' TO characters

ENDIF

IF use\_uppercase THEN

APPEND 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' TO characters

ENDIF

IF use\_numbers THEN

APPEND '0123456789' TO characters

ENDIF

IF use\_special THEN

APPEND '!@#$%^&\*()\_+-=[]{}|;:,.<>/?' TO characters

ENDIF

FOR i FROM 1 TO length

SET randomIndex = RANDOM INTEGER BETWEEN 1 AND LENGTH(characters)

APPEND characters[randomIndex] TO password

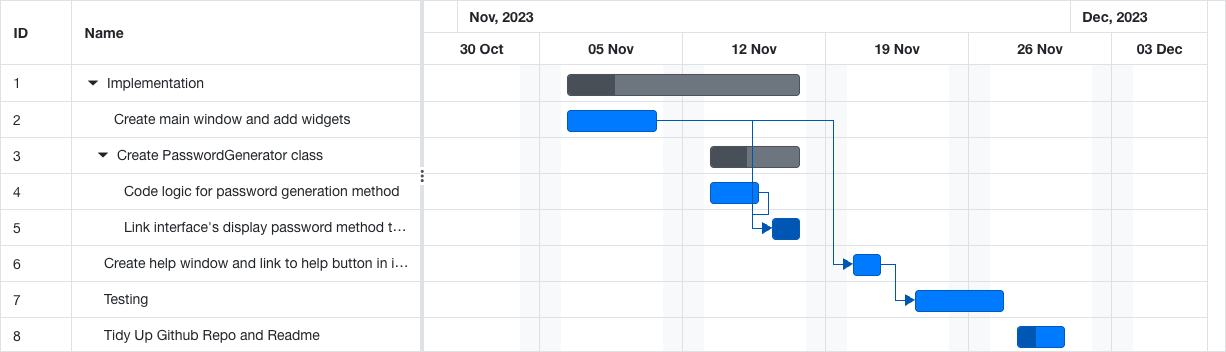
ENDFOR

RETURN password

END generate\_password

## GANTT Chart

The following GANTT chart was created at the beginning of my project and includes predicted timescales for each of the main tasks surrounding implementation, testing and release of my software.



*I created this GANTT chart using the free GANTT chart creator* [*https://www.onlinegantt.com/#/gantt*](https://www.onlinegantt.com/#/gantt)

# Implementation

## GitHub Repository URL

# Testing