Saint Augustine’s College, Sydney

**Saint Augustine’s College, Sydney**

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

**Benjamin Robertson: Password strength Checker**

Table of Contents

[Planning 3](#_Toc201515496)

[Task Definition 3](#_Toc201515497)

[Hardware and Software Requirements 4](#_Toc201515498)

[Storyboard 5](#_Toc201515499)

[Algorithm Design 6](#_Toc201515500)

[Implementation 7](#_Toc201515501)

[GitHub Repository URL 7](#_Toc201515502)

[Data Dictionary 8](#_Toc201515503)

[Testing Table 9](#_Toc201515504)

[FlowChart 10](#_Toc201515505)

[Project Reflection 11](#_Toc201515507)

# Planning

## Task Definition

With the growing risks linked to unsecure and predictable passwords, this project entails the creation of a Python GUI application for evaluating as well as enhancing password security. A recent study conducted by cybersecurity firm NordPass revealed the ongoing susceptibility of individuals and even top institutions as a result of the extensive usage of unsecure passwords. This highlights the requirement for readily available, easy-to-use tools that will actively help users in strong, secure password generation and maintenance.

My solution to this problem is a graphical user interface (GUI)-based password generator and analyser tool, which I have developed using customtkinter for a clean and modern appearance. The software's core function is two-fold:

Password generation allows users to create random passwords that contain a combination of upper- and lower-case letters, numbers, and special characters. Furthermore, the users can also opt to adjust the length of the generated passwords to fulfill specific requirements.

Password strength analysis, through calculations of entropy to provide users with an insight into how strong their passwords are. Constructive criticism, as well as recommendations for enhancement, are provided based on password security best practices.

For better usability, the application features a copy-to-clipboard feature, thereby enabling users to conveniently copy generated passwords while creating or renewing accounts.

What sets this project apart is the fact that there are hidden easter eggs and interactive minigames that are embedded within the interface. These elements boost exploration, thereby creating an engaging experience while subtly reinforcing healthy cybersecurity habits. This element of fun turns a commonly boring security activity into a more delightful user experience.

Python was chosen for the purpose of this project due to its readability, large collection of standard libraries, and easy interfacing with third-party modules like customtkinter to design the graphical user interface, random and string to generate secure passwords, and JSON files to hold information. Additionally, the software will be built based on a modular structure, thus ensuring code legibility, maintainability, and adherence to best programming standards. This project aims to give a compelling and interactive tool for the implementation of stronger password security best practices through the union of technical expertise and creative engagement.

## Hardware and Software Requirements

In order to develop this application, 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 | Python 3.x |
| 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 | Modern CPU capable of efficient multitasking |
| Memory | 4GB RAM or higher for optimal performance |
| Storage | Minimum of 1GB free space for project and software |

## Storyboard

This storyboard outlines the planned design of a modern, interactive GUI developed in Python 3.x using the CustomTkinter framework. The interface will feature a central input field for password entry, strength checking, generation, and copy-to-clipboard functionality, all presented in a clean, user-friendly layout.

A dedicated Help screen will provide clear instructions, while the About page shares developer information and useful links. To enhance the experience, random programming jokes will be displayed to add humour and engagement.

The application also features an Achievements window, encouraging users to discover hidden secrets and goals, adding a layer of gamification. Additionally, a hidden Developer Security Area can be unlocked by completing a quiz, granting access to a secret webpage with further easter eggs and challenges.

*A screenshot of a computer screen

AI-generated content may be incorrect.* *This diagram was generated using* [*.drawio*](https://www.drawio.com/)*, a link to the template can be found* [*here*](https://app.diagrams.net/#G1Nrq-Wh2iyLPM1n0gFWWfUcL9bp7b1my5)*.*

## Algorithm Design

The provided pseudocode delineates the method for the generate\_secure\_password method. It constitutes one of the program's most used algorithm, involving string operations and iterative constructs to assemble the password according to specified parameters. I would do check password strength but is over 150 lines and that is far to much to convert to pseudo code

#### generate\_secure\_password

BEGIN generate\_secure\_password

SET characters = 16

Set Special characters =  [!@#$%^&\*()-\_=+[]{};:,.?/](mailto:!@#$%^&*()-_=+[]{};:,.?/)

Set Letters = “abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ”

Set Numbers = “1234567890”

Set Characters = Special characters + Letters + Numbers

Set Password = “”

FOR i FROM 1 TO length

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

APPEND characters[randomIndex] TO password

ENDFOR

RETURN password

Set password\_entry to password # sets the password field in the app to display the password

END generate\_secure\_password

# Implementation

## [GitHub Repository URL](https://github.com/Robertson-B/Password-Checker)

https://github.com/Robertson-B/Password-Checker

# Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Data Type | Description | Example Value |
| COLORS | Dictionary | Stores the colour palette for the app’s UI elements (background, header, buttons, etc.). has to be American spelling because that’s what CTK uses | "background":"#F0F4F8",  "header": "#005A9E", |
| [ALLOWED\_PASSWORD\_CHARS](vscode-file://vscode-app/c:/Users/benja/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | String | String of all characters allowed in password input. | “ABCabc123^$#” |
| secret\_theme\_on | Boolean | Indicates if the secret dark theme is enabled. | True |
| password\_checks | Integer | Number of times the user has checked a password | 17 |
| Start\_time | Float | Timestamp when the app started (used for "Quitter" achievement). | 13.47 |
| konami\_progress | String | Tracks user input for the Konami code easter egg. | “upupdown” |
| password\_entry | CTKentry/string | The current inputted password in the password entry field | “nevergonnagiveyouup” |
| [copy\_button](vscode-file://vscode-app/c:/Users/benja/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | CTKbutton | Button to copy the password to clipboard. | NA |

# Testing Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No. | Input Password | Test Description | Expected Output | Actual Output | Pass/Fail |
| 1 | 1234123412341234 | Check to see if the password handles only numbers | Add an uppercase/lowercase /special character | Add an uppercase/lowercase /special character | Pass |
| 2 | abcdefghijklmno | Check to see if the password handles only lowercase letters | Add an uppercase/number /special character | Add an uppercase/number /special character | Pass |
| 3 | ABCDEFGHIJKLM | Check to see if the password handles only upper letters | Add a lowercase/number /special character | Add a lowercase/number /special character | Pass |
| 4 | abcABC123!@# | Check to see if the password handles all types of characters | Strong | Strong | Pass |
| 5 | password | Checks to see if it handles common passwords | Common | Common | Pass |
| 6 | nevergonnagiveyouup | Checks to see if the easter egg works | Rickrolled  Opens rickroll link | Rickrolled  Opens rickroll link | Pass |

# Flowchart

# A diagram of a flowchart AI-generated content may be incorrect.

*This diagram was generated using* [*.drawio*](https://www.drawio.com/)*, a link to the template can be found* [*here*](https://app.diagrams.net/#G1FL-e8EYIvEmQwhZ3faVG8hwfB4bku9qV)*.*

# Project Reflection

The development of this password strength checker application was both a technically challenging and creatively rewarding experience. The project achieved its primary goal of creating an accessible, secure, and engaging tool for password generation and strength evaluation. By integrating features such as entropy-based strength calculation, user progress tracking, and API-based breach checking, I was able to deliver a comprehensive solution that aligned with both the project requirements and modern cybersecurity standards.

One of the key successes of this project was the successful implementation of real-time password breach checking through API integration. By connecting to external breach databases, the application offers users meaningful, evidence-based feedback about the security of their passwords, extending its usefulness beyond local strength checks. Another major success was the addition of user progress tracking, tied to the achievement system. This feature encouraged ongoing engagement with the application by allowing users to unlock easter eggs, find hidden minigames, and track their progress over multiple sessions, transforming the app into an interactive experience rather than a purely functional tool.

I also consider the entropy-based password strength evaluation to be a significant achievement. Implementing entropy calculations required independent research into cryptographic principles and adapting them to work within the program’s real-time feedback system. This demonstrated my ability to engage with complex, real-world cybersecurity concepts and translate them into accessible features for users.

However, the development journey was not without challenges. Working with CustomTkinter presented a steep learning curve, especially when managing multiple interface windows, hidden features, and ensuring consistent design across the application. Debugging these features taught me the importance of modular programming and iterative refinement. Additionally, balancing the entertainment aspects of the application with its serious security functionality required careful design. I addressed this by keeping the primary interface clean and professional while hiding playful features behind unlockable achievements and quizzes.

For future development, improvements could focus on expanding the scope of the achievement system to include more interactive challenges, potentially incorporating online leaderboards or personalized goals to deepen user engagement. The visual design could also be enhanced by implementing more sophisticated layout management and better responsiveness for different screen resolutions.

Overall, this project has significantly strengthened both my technical and creative skills. Through the integration of API handling, data security principles, GUI development, and user experience design, I have grown as a programmer and problem-solver. This project has taught me how to combine practical functionality with creative engagement, preparing me for more advanced, user-focused software projects in the future.