LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS6P05NI Final Year Project

Assessment Weightage & Type

40% FYP Final Report

Year and Semester

2022/23 Spring

Password Analyzer Using Machine Learning

Student Name: Sujen Shrestha

London Met ID: 20049250

College ID: NP01NT4S210105

Internal Supervisor: Aaditya Khwakhwali

External Supervisor: Bijay Limbu Senihang

Assignment Due Date: 19th April 2023

Assignment Submission Date: 18th April 2023

Word Count: 8872

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

I would like to express my profound gratitude to Mr. Aaditya Khwakhwali, my internal supervisor, and Mr. Bijay Limbu Senihang, my external supervisor, for their invaluable guidance and support throughout the process of developing this project. Their insights and expertise have been extremely valuable in helping me understand the subject matter, overcome the challenges and complete this project report.

I am deeply grateful to Mr. Aaditya Khwakhwali for his continuous encouragement, constructive feedback, and commitment to my progress. His guidance has been essential in developing the skills and knowledge required to accomplish this project. I would also like to extend my appreciation to Mr. Bijay Limbu Senihang for providing me his expert advice, guidance, and techniques to efficiently carry out this project. His perspective has been critical in ensuring the success and enhancing quality of the final report.

Lastly, I would like to acknowledge the support and dedication of my friends and teachers who have encouraged, guided, and inspired me throughout this project. Their collective contributions have been extremely significant for the successful completion of this project.

Abstract

This report presents the development and analysis of a password strength analyzer and random password generator tool, designed to enhance user account security and promote strong password creation practices. The study explores the current state of password security, the need for a robust password management tool, and a detailed understanding of the proposed solution. The password strength analyzer evaluates user passwords based on a variety of factors, including length, character composition, and prevalence in known data breaches. In parallel, the random password generator creates secure passwords adhering to best practices in password security.

The report also includes extensive testing and analysis, covering both unit and system testing, to ensure the effectiveness and accuracy of the password strength analyzer and random password generator. Furthermore, it addresses the legal, social, and ethical issues associated with password security, as well as the advantages and limitations of the developed tool. Lastly, the report identifies potential areas of future work, such as enhanced machine learning algorithms, integration with password managers, and the development of an API to improve the tool's reach and usability. Overall, this report demonstrates the potential of the password strength analyzer and random password generator as a valuable tool for users to create and maintain strong, secure passwords, ultimately safeguarding their online accounts from unauthorized access and data breaches.

Table of Contents

List of Figures

List of Table

1. Introduction

1.1 Project Description

Passwords have become a major component of modern life. A typical computer user may also need passwords for a variety of purposes, such as accessing computer accounts, obtaining email from servers, transferring money, making purchases online, having access to applications, databases, networks, and websites, or simply reading the morning newspaper online. Therefore, the passwords are the barriers we use to protect our sensitive personal data online. As our society increasingly transitions into the digital realm, it is widely recognized that we require multiple passwords to safeguard our online activities and sensitive personal information. The consequences of such data falling into the wrong hands can be severe, and at times it is imperative to maintain anonymity while online. With the internet and cybercrime landscape constantly evolving, security vulnerabilities are becoming more widespread and pervasive. Choosing a strong password has become vital for users as a result of the rise in cyberattacks.

This project helps to tackle this issue by creating a password strength analyzer that uses machine learning, which is a valuable tool designed to assess the strength of a password. This tool uses machine learning algorithms that have been trained on vast datasets consisting of passwords and their corresponding strength ratings. These algorithms analyze the characteristics of a password and use the training data to predict its strength. Specifically, the password analyzer considers factors such as password length, complexity, and uniqueness in order to evaluate the password's strength. In essence, this machine learning-based password strength classifier assists users in selecting robust passwords and discourages the use of weak passwords that are vulnerable to being guessed or cracked by malicious actors.

1.2 Current Scenario

It is found in a study that 43% of passwords are reused by people for multiple accounts while researching the password practices of end users who appeared or reappeared in several relevant credential dumps (Das, et al., 2014). By analysing the same password reuse across 21 prestigious American colleges, research was conducted on the effects of password reuse across various sites and its poor practices leading to security or personal data breaches (Abbott, et al., 2018).

During the year 2020, the cybersecurity organization Blackbaud encountered a data breach that impacted millions of individuals worldwide. The company became aware of the breach upon noticing anomalous activity on its systems and subsequently identifying those malevolent actors had obtained access to a huge amount of sensitive personal and financial information, including names, addresses, and social security numbers. Blackbaud later revealed that many of the compromised passwords were weak and had likely been easily guessed by the attackers (Mills, 2021).

Figure 1: Password Breach Statistics since 2017 (Chang, 2022).

1.3 Problem Domain and Project as a Solution

In the current digital age, weak passwords continue to be a major security concern. They are a common target for hackers and other cybercriminals, who use various techniques such as dictionary attacks, brute force attacks, and social engineering to try to guess or crack passwords and gain access to sensitive information.

One major issue with weak passwords is their susceptibility to being easily guessed or cracked, which results to them being compromised. Passwords that are simple and commonly used words or phrases or that include personal information like names or dates of birth can be effortlessly guessed and cracked through dictionary attacks or similar methods.

The problem domain of this project revolves around the increasing need for robust password security in an era of growing digital threats. With the rapid expansion of online services, users are required to maintain multiple accounts, each with a unique password. However, users often struggle to create strong passwords that are both secure and memorable. Weak or compromised passwords make it easier for hackers and malicious actors to access sensitive data, leading to data breaches, identity theft, and other forms of cybercrime.

To address this issue, Password Strength Analyzer web application has been developed as a solution, providing users with an accessible and user-friendly tool to analyze their password strength, ensure compliance with established guidelines, and generate secure passwords.

This project provides various features, each of which are listed below:

- **Password Strength Analyzer:**

The application includes a password analysis tool that uses machine learning to evaluate the strength of a user's password, providing real-time feedback on its effectiveness. By checking the password against a pre-trained model, the app can determine whether the password is weak, average, or strong, helping users identify areas for improvement.

- **NIST Compliance Guidelines:**

The app displays password compliance guidelines based on recommendations from the National Institute of Standards and Technology (NIST), ensuring that users understand the best practices for creating secure passwords. These guidelines provide actionable steps that users can follow to enhance the security of their passwords.

- **Password Breach Check:**

The app integrates with the Have I Been Pwned API to determine if a user's password has been previously compromised in a data breach. This feature helps users understand the importance of regularly updating their passwords and avoiding the use of passwords that have been exposed in previous breaches.

- **Random Password Generator:**

The application offers a strong password generator that creates secure and unique passwords based on user-specified length. This feature assists users in generating strong passwords which cannot be easily guessed or cracked by hackers.

1.4 Aim and Objectives

1.4.1 Aim

The aim of this project is to build a tool that is accurate, easy to use, and secure, and that helps users choose strong, unique passwords and prevent the use of weak passwords that can be easily guessed or cracked by attackers. This will help individuals to minimize the risk of being hacked or have their personal information be compromised.

1.4.2 Objectives

Some specific goals or objectives that this project aims to achieve include:

- **Building a classifier that is accurate and reliable:** The primary goal of the project is to build a password strength analyzer that is accurate and reliable in its assessments of password strength. This involves training the analyzer on a large dataset of passwords and their associated strength scores, and testing the analyzer to ensure that it is able to accurately predict the strength of new passwords.

- **Providing users with an easy-to-use tool:** The analyzer should be easy for users to use, with a clear and intuitive interface that allows them to easily input their passwords and receive feedback about their strength.

- **Improving over time:** The tool should be designed to improve over time, with the ability to learn from new data and become more accurate in its assessments of password strength.

- **Ensuring the security and privacy of user data:** The project should prioritize the security and privacy of user data, ensuring that passwords are stored and transmitted in a secure manner and that user privacy is protected.

1.5 Structure of the Report

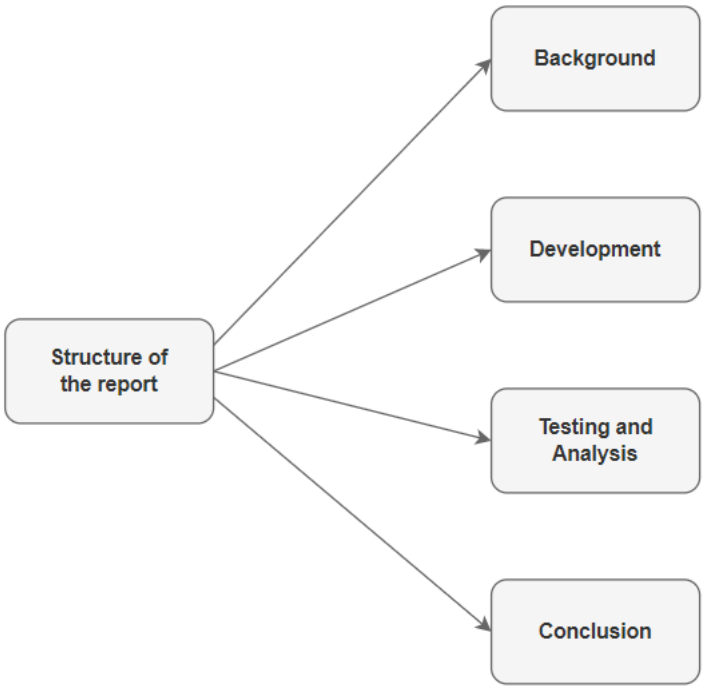


Figure 2: Structure of the report

1.5.1 Background

This chapter delves into the background information necessary for understanding the project. It begins with a discussion of the end users, detailing their needs and requirements. It then moves on to an overview of the proposed solution and how it addresses those needs. The chapter also includes a literature review of similar projects, highlighting their key findings and the impact they have on the current project. A comparison table and analysis of the comparisons are provided to offer insights into how the current project builds upon or differs from existing work in the field.

1.5.2 Development

This chapter outlines the entire development process for the project. It starts with a presentation of the considered methodologies for project development, followed by an explanation of the selected methodology and its phases. The requirement analysis section discusses the hardware and software requirements for the project, followed by the design phase, which includes flowcharts, wireframes, and use case diagrams. The implementation section details the creation of the frontend and backend components of the system, while the backend development section provides a brief overview of the code used for this project.

1.5.3 Testing and Analysis

This chapter focuses on the various testing methods employed to evaluate the system's performance and functionality. The test plan section outlines the strategies for both unit testing and system testing, followed by the execution of these tests and the presentation of their results. A critical analysis is also included to discuss the project's strengths, weaknesses, and areas for improvement.

1.5.4 Conclusion

This chapter summarizes the project's outcomes, reflecting on its objectives and achievements. It discusses the legal, social, and ethical issues related to the project, highlighting potential challenges and considerations for implementation. The advantages and limitations of the project are also explored, offering a balanced view of its overall impact. Lastly, this chapter identifies areas for future work and potential enhancements to the system.

2. Background

2.1 About the End Users

The end users for this password strength analyzer and random password generator project are diverse, as the system aims to cater to a wide range of individuals and organizations. The primary goal is to enhance the security of users' online accounts and sensitive data by encouraging the use of strong, unique passwords. The following sections provide a detailed description of the key end users who stand to benefit from this project:

- **Individual Users:**

Internet users across the globe require strong and secure passwords to protect their personal and financial information in various online platforms. These may include social media accounts, email services, online banking, e-commerce websites, and more. The password strength analyzer and random password generator will help individual users assess the strength of their existing passwords and generate new, complex passwords, reducing the likelihood of unauthorized access and potential identity theft.

- **Small and Medium-sized Businesses (SMBs):**

Small and medium-sized businesses often lack the resources to implement advanced security measures, making them vulnerable to cyber threats. By using the password strength analyzer and random password generator, SMBs can help their employees create and maintain secure passwords for their work accounts, which may include email, project management tools, and customer relationship management (CRM) systems. This will aid in minimizing the risk of data breaches, which can have severe financial and reputational consequences for businesses.

- **Large Enterprises:**

Although large enterprises typically have more robust cybersecurity measures in place, they also have a larger attack surface due to the sheer number of employees and accounts. The password strength analyzer and random password generator can be used by employees in these organizations to maintain secure passwords for their various work accounts, reducing the risk of unauthorized access and data breaches.

- **Educational Institutions:**

Schools, colleges, and universities require secure passwords for their students, faculty, and staff to protect sensitive information such as grades, personal records, and research data. The password strength analyzer and random password generator will help users within educational institutions to create and maintain strong passwords, thereby safeguarding their critical data from potential cyber threats.

- **Government Agencies:**

Government agencies handle sensitive information related to national security, public services, and more. The password strength analyzer and random password generator can help government employees maintain secure passwords for their various accounts, ensuring that critical data remains protected from unauthorized access and cyber-attacks.

In summary, the end users of the password strength analyzer and random password generator project encompass a wide range of individuals and organizations seeking to improve their online security by using strong, unique passwords. By providing an accessible and user-friendly tool, the project aims to promote better password hygiene, contributing to a safer and more secure digital environment for all.

2.2 Understanding the Solution

The proposed solution, which includes a password strength analyzer and random password generator, seeks to significantly enhance users' online security by providing a comprehensive and user-friendly method for managing passwords. This section provides the specifics of each component and explains how they work together to create a robust and reliable solution for password security.

2.2.1 Password Strength Analyzer

The password strength analyzer is a sophisticated tool designed to evaluate the security of users' passwords based on a range of factors such as length, character variety, and overall complexity. By examining these elements, the analyzer can offer users a clear understanding of the strength of their current password and whether it requires improvement. The tool employs advanced machine learning algorithms to detect patterns and trends in password security, which enables it to deliver accurate and trustworthy feedback to users. Armed with this knowledge, users can make informed decisions about updating their existing passwords to meet the necessary security standards, effectively decreasing the likelihood of unauthorized account access.

The password strength analyzer operates through the following steps:

- **Accepting user input:** The user enters their existing password into the provided text field, which is then analyzed by the tool.

- **Analyzing the password:** The analyzer assesses the entered password based on several factors, including length, character variety (i.e., the presence of uppercase and lowercase letters, numbers, and special characters), and the use of easily guessable words or patterns.

- **Providing feedback:** Based on the analysis, the tool assigns a strength score (e.g., weak, moderate, strong) to the password and offers suggestions for improvement if necessary.

2.2.2 Random Password Generator

The random password generator serves as a valuable addition to the password strength analyzer by helping users create new, secure passwords that are less likey to be hacked. The generator produces complex and unique passwords by combining a mix of character types (i.e., uppercase and lowercase letters, numbers, and special characters) and ensuring that the resulting password complies to the user-defined length parameters.

The random password generator functions as follows:

- **User-defined parameters:** The user inputs their desired password length (within a predefined minimum and maximum range to guarantee password security).

- **Generating the password:** The generator creates a random combination of characters that corresponds to the specified length and character types.

- **Displaying the result:** The user is presented with the generated password, which can then be used for their online accounts.

2.3 Similar Projects

2.3.1 Project 1: Building a Multi-class Password Strength Generator and Classifier Model by Augmenting Supervised Machine Learning Techniques

Authors: Sakya Sarkar and Mauparna Nandan

In this project, a dataset was first created using a password generator. The goal of the study was to use machine learning algorithms to predict the strength of passwords, which was modelled as a classification task. Multiple supervised machine learning algorithms were implemented and used for training and testing the dataset. During the testing phase, the XGBoost algorithm outperformed the other algorithms, achieving an accuracy of 94%. This work is significant because it introduces a new method for generating the dataset using a password generator and applies various machine learning algorithms, including Adaboost, XGBoost (XGB), Stochastic Gradient Descent (SGD), and Multilayer Perceptron (MLP), for training and testing. The results indicate that machine learning approaches can effectively classify passwords into different strength categories, such as very weak, weak, medium, strong, and very strong (Sarkar & Nandan, 2022).

2.3.2 Project 2: Real Time Password Strength Analysis on a Web Application Using Multiple Machine Learning Approaches

Author: Umar Farooq

This project uses a common and efficient method for protecting against online and offline attacks by forcing users to choose strong passwords through the implementation of multiple machine learning algorithms, such as Decision Tree, Naïve Bayes, Linear Regression, Random Forest, and Neural Network, on a web application in real-time. This means that a user will only be able to log in to their account if the password strength from all algorithms is considered strong. During testing, the model achieved the best results with Decision Tree, with an accuracy of 99%, and the worst results with Naïve Bayes, with an accuracy of 87%. The model was also tested against three types of password cracking attacks, including Brute Force Attack, Dictionary Attack, and Reverse Brute Force Attack, using the Burp Suite software on a web application with 250 accounts, 150 of which had strong passwords and 100 of which had weak passwords. None of the strong passwords were cracked by these attacks, while 86 of the weak passwords were cracked (Farooq, 2020).

2.3.3 Project 3: Proactive Password Strength Analyzer Using Filters and Machine Learning Techniques

Authors: Suganya G, Karpagavalli S and Christina V

This work proposed a framework to proactively analyze the strength of passwords using filters and a support vector machine. The framework was intended to be implemented as a submodule of an access control system. The problem of choosing weak, easily exploited passwords and the need to enforce the selection of strong passwords was identified as an important issue that needed to be addressed. As a solution, the Proactive Password Strength Analyzer was designed with various filters and an RBF kernel to classify the chosen password. The performance of various machine learning algorithms was studied, and the best model was employed in the Proactive Password Strength Analyzer. The proposed analyzer was intended to be implemented as a submodule of an access control mechanism to help users choose strong passwords that would protect their identity and resources (G, et al., 2010).

2.4 Comparisons

2.4.1 Comparison Table

S.N.	Features	Project 1	Project 2	Project 3	This Project
1.	HIgh Accuracy	✓	✓	✗	✓
2.	Responsive User Interface	✗	✓	✗	✓
3.	Local / Web Host	✗	✓	✓	✓
4.	Application	✗	✓	✓	✓
5.	Random Password Generator	✗	✗	✗	✓
6.	Password Breach Checker	✗	✗	✗	✓
7.	Multiple Algorithm Testing	✓	✓	✓	✓
8.	Password Conceal Button	✗	✗	✗	✓
9.	Password Guidelines	✗	✗	✗	✓

Table 1: Similar project comparison table

2.4.2 Analysis and Conclusion of the Comparison

Projects 1 and 2 have had a significant impact on the development of this project. These projects have significantly influenced the selection of machine learning model required for this project. All the similar projects can classify the password strength however this project stands out in comparison to them as it implements various additional features. The additional functionalities which are implemented in this project includes a random password generator which can create a password of length 10 to 64 as entered by user. Another significant feature of this project is that it checks the entered password has been found in any previous data breach using the API from “haveibeenpwned”. Due to such capabilities, this project will help the users to create robust passwords to secure their online presence.

3. Development

3.1 Considered Methodologies

3.1.1 Waterfall Methodology

The Waterfall methodology, sometimes referred to as the Waterfall model, is a structured approach to project development in which each phase must be completed before moving on to the next. This process involves a series of sequential steps, such as analysis, design, programming, and testing, which flows in a linear fashion, like a waterfall (Adobe Communication , 2022).

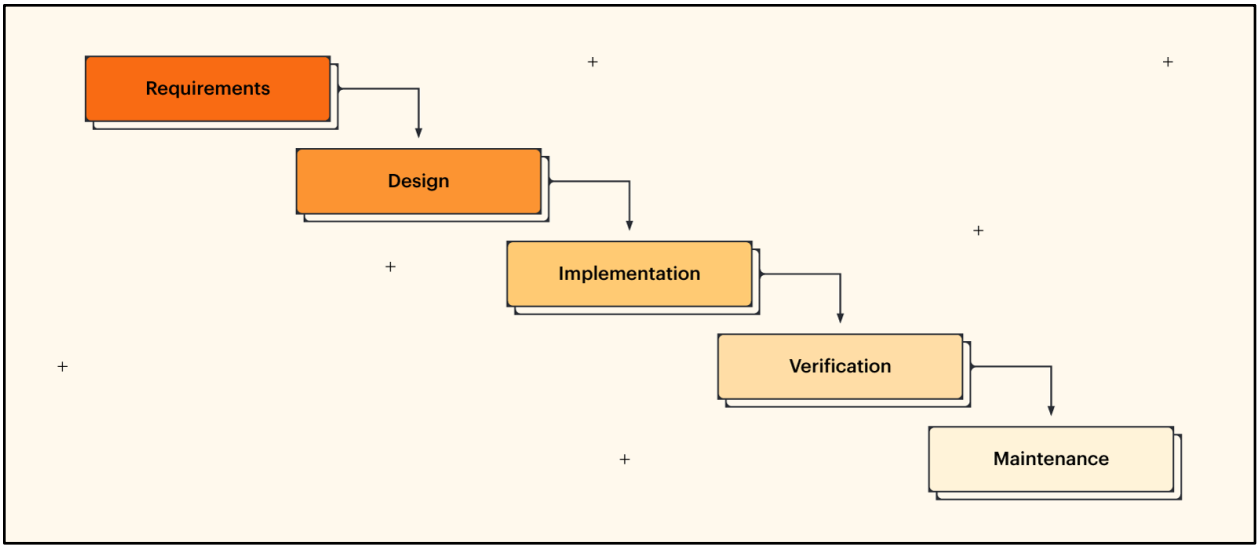


Figure 3: Waterfall Methodology (Lucid Content, 2021).

3.1.2 Scrum Methodology

The Scrum methodology will be used for the development of this system. This approach involves iteratively improving the project through incremental steps, and is particularly useful for developing websites and software. It allows for flexibility and adaptability, enabling scope adjustments to be made mid-project in order to continually improve the work (Martin, 2022).

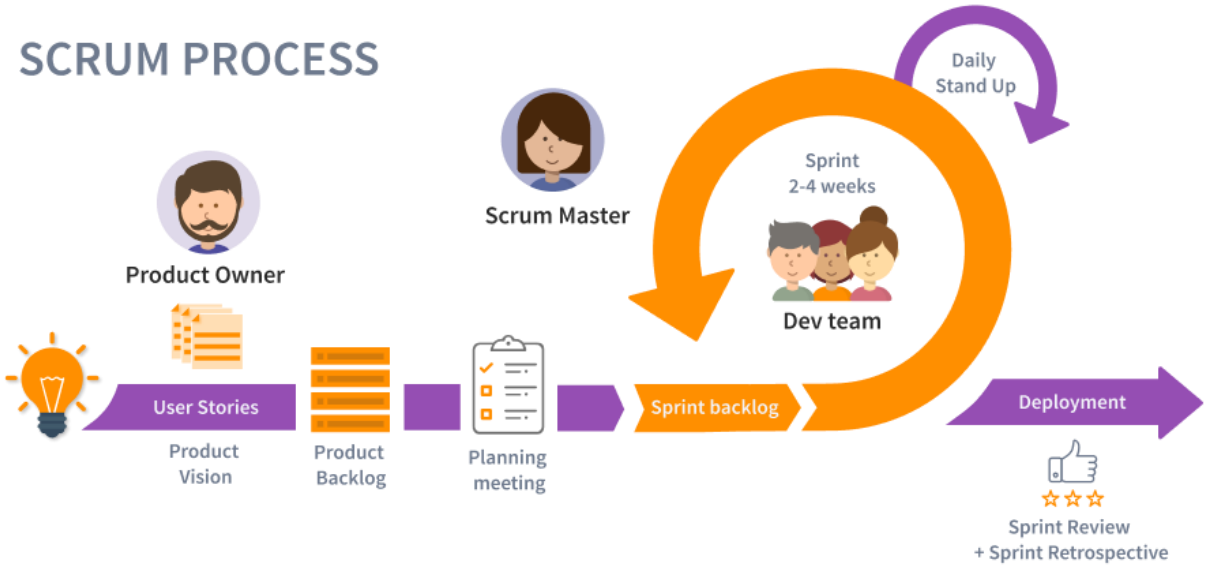


Figure 4: Scrum Methodology (Tuleap, 2022).

3.1.3 Prototype Methodology

The Prototype methodology is a software development approach that involves creating a prototype, or a preliminary version of the final product, which is tested and refined until it meets the necessary requirements. This process involves an iterative, trial-and-error approach in which both the developer and the client work together to identify and address any issues or changes that need to be made. The Prototype methodology is useful when the project's requirements are not fully understood and helps to build the foundation for the final system or software (Martin, 2022).

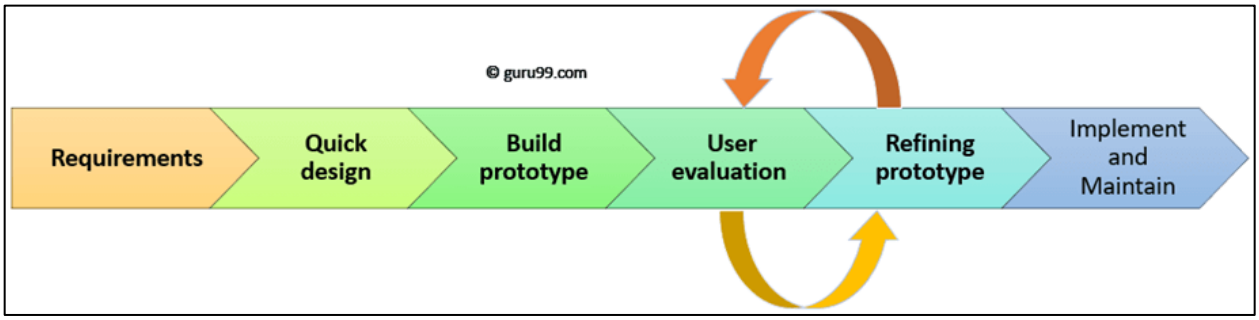


Figure 5: Prototype Methodology (Martin, 2022).

3.2 Selected Methodology

3.2.1 Scrum Methodology

The Scrum methodology will be used for the development of this system. This approach involves iteratively improving the project through incremental steps, and is particularly useful for developing websites and software. It allows for flexibility and adaptability, enabling scope adjustments to be made mid-project in order to continually improve the work (Martin, 2022).

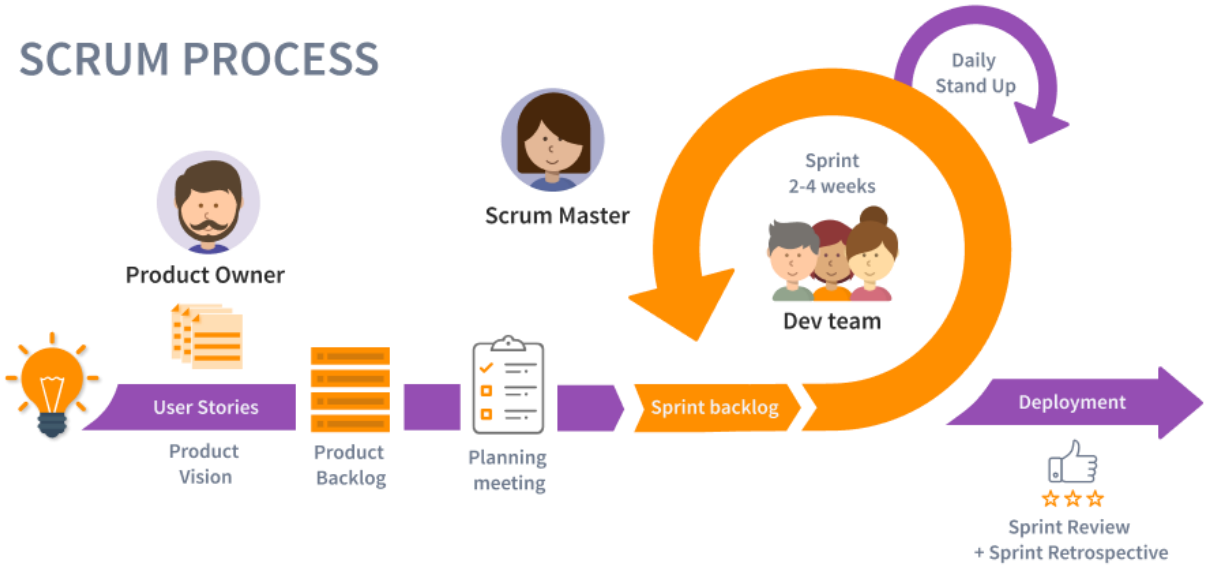


Figure 6: Selected Methodology (Tuleap, 2022).

The Scrum methodology is focused on meeting the client's objectives and is able to accommodate changing requirements throughout the development process. It involves regular sprint cycles that include planning, development, testing, and review phases. This approach ensures timely deliverables, strong project visibility, and continuous improvement.

The agile approach, specifically the Scrum methodology, is well-suited for this project which involves developing a password analyser using machine learning, training

and testing the system with a dataset, and reviewing the results. If additional features are needed in the future, the methodology can be used to iteratively improve the project through additional sprint cycles using the same process.

3.3 Phases of Methodology

The project was carried out using the four phases of the Scrum methodology, which are described below:

3.3.1 Planning

In this phase, research was conducted to determine the hardware and software requirements for the project, as well as the steps needed for initiation, development, and deployment.

3.3.2 Designing

During this phase, the proposed system's wireframes was created using Balsamiq wireframe software and flowcharts and use case diagrams was created using draw.io.

3.3.3 Development

In this phase, the project was developed, including the creation of both the front end and back end of the system and their integration into a functional web application.

3.3.4 Testing

Lastly, the functionality, performance, and testing of the system was carried out using various methodologies such as white box, black box, and grey box testing, and the results have been documented.

3.4 Requirement Analysis

3.4.1 User requirements:

The user must have a computer with internet connection and a web browser to access the web application.

3.4.2 Developer requirements:

To develop this web application, the developer must have knowledge of various tools required for programming. The tools used to develop this project are listed below:

- **Programming Language – Backend: Python**

Python is a powerful, high-level programming language that is known for its simplicity, flexibility, and versatility. It is interpreted, meaning that it does not need to be compiled before it is run, and it supports object-oriented programming, which allows for the creation of reusable code. Python's built-in data structures and support for dynamic typing and binding make it well-suited for rapid application development and the integration of existing components. It's easy-to-learn syntax and focus on readability help to keep the maintenance cost of programs low (Python Software Foundation, 2022).

- **Frontend: Streamlit**

Streamlit is a Python-based framework that enables machine learning engineers to quickly and easily build and share visually appealing web applications for data science and machine learning. It is open-source and free to use, and requires only a minimal amount of code to create impressive apps. Streamlit is specifically designed for machine learning professionals and makes it simple to create visually striking apps in a short amount of time (Mhadhbi, 2021).

- **Text Editor: Visual Studio Code**

VS Code is a powerful source code editor that is optimized for speed and efficiency. It supports a wide range of programming languages and includes features like syntax highlighting, automatic indentation, and bracket matching to help users write code quickly and accurately. VS Code also integrates with various tools for building and scripting, allowing users to perform common tasks without leaving the editor. Additionally, VS Code has built-in support for Git, allowing users to manage source control and view diffs of pending changes directly from the editor (Visual Studio Code, 2022).

- **API: Pwned Passwords**

Pwned Passwords is a database containing hundreds of millions of real-world passwords that have been exposed in data breaches. These passwords are not safe to use because they are more likely to be targeted by attackers trying to gain access to other accounts (Hunt, 2022).

- **Version Control System: GitHub**

GitHub is a web-based platform designed for version control and collaborative software development. It provides a centralized location for developers to store and manage their code, track changes to it over time, collaborate with others, and deploy their applications to the cloud. Using GitHub, developers can host and manage their Git repositories, which are collections of files and folders that make up a project. They can create branches to work on specific features or changes, merge changes from different branches, and track issues and bugs. In addition to version control, GitHub offers a range of project management tools, such as wikis, issue tracking, and project boards. These tools help teams work together more efficiently and effectively, allowing them to deliver higher quality software products in less time (Gaba, 2023).

3.5 Design

3.5.1 Flowchart

A flowchart is a visual representation of the steps involved in a process, in this case, the password project. It displays the sequence of actions and decision points, helping to identify the flow of information and interactions between different components of the system (Chaudhuri, 2020).

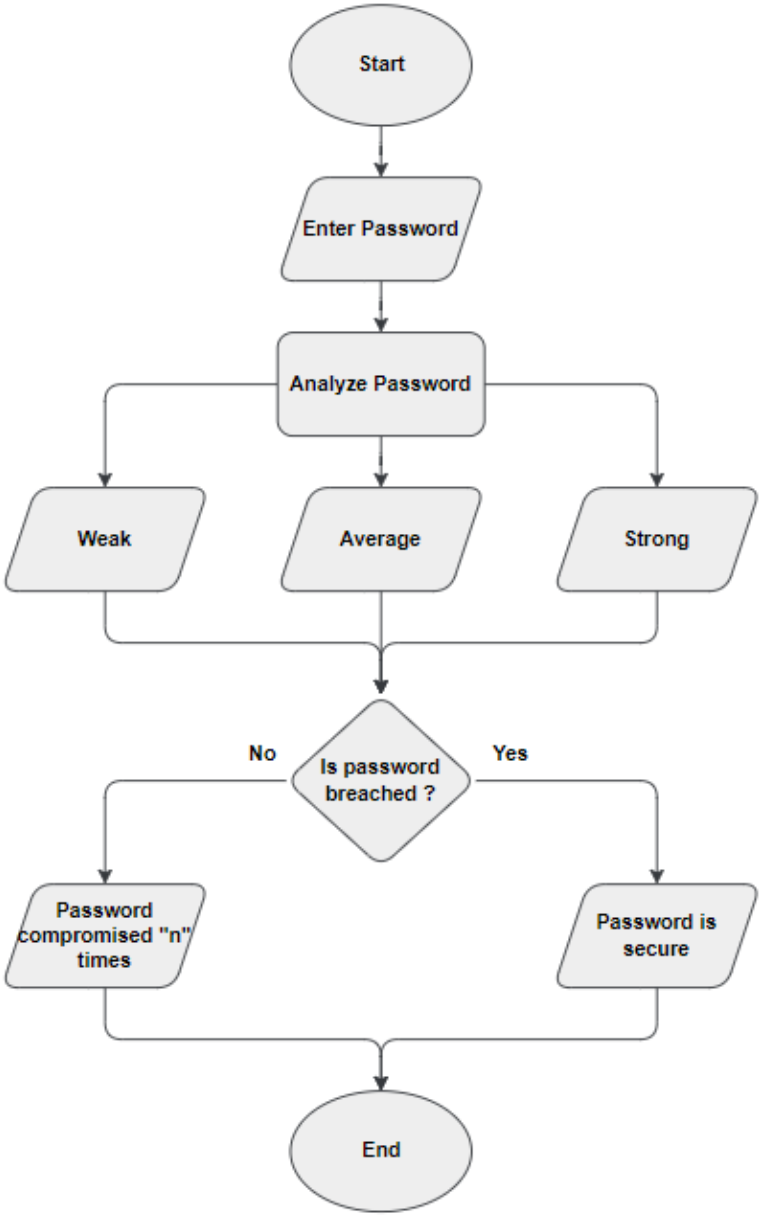


Figure 7: Flowchart of Password Analyzer

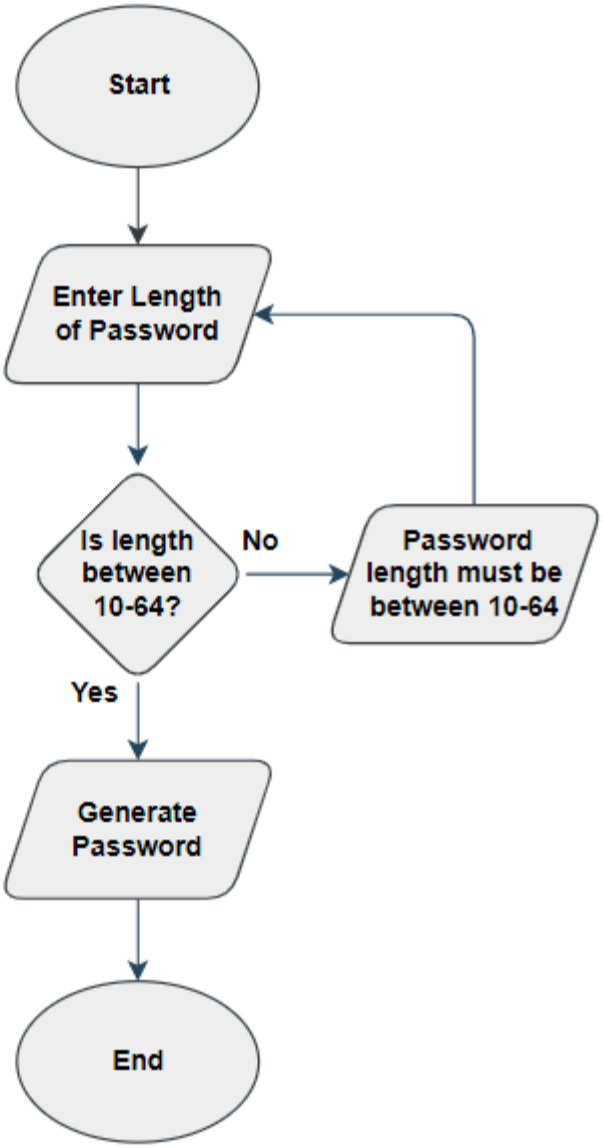


Figure 8: Flowchart of Password Generator

The stepwise process flow of the application which is required to understand the working mechanism of the project is represented in the form of a flowchart. This is explained in the appendix section of the report. **(Flowchart: Appendix C)**

3.5.2 Use Case Diagram

A use case diagram is a visual representation of a system's functionality from the perspective of its users (called "actors") and their interactions with the system. Use case diagrams are part of the Unified Modeling Language (UML) and are used primarily during the requirements gathering and analysis phase of software development. They help developers and stakeholders understand how the system will be used and the relationships between different use cases (Visual Paradigm, 2023).

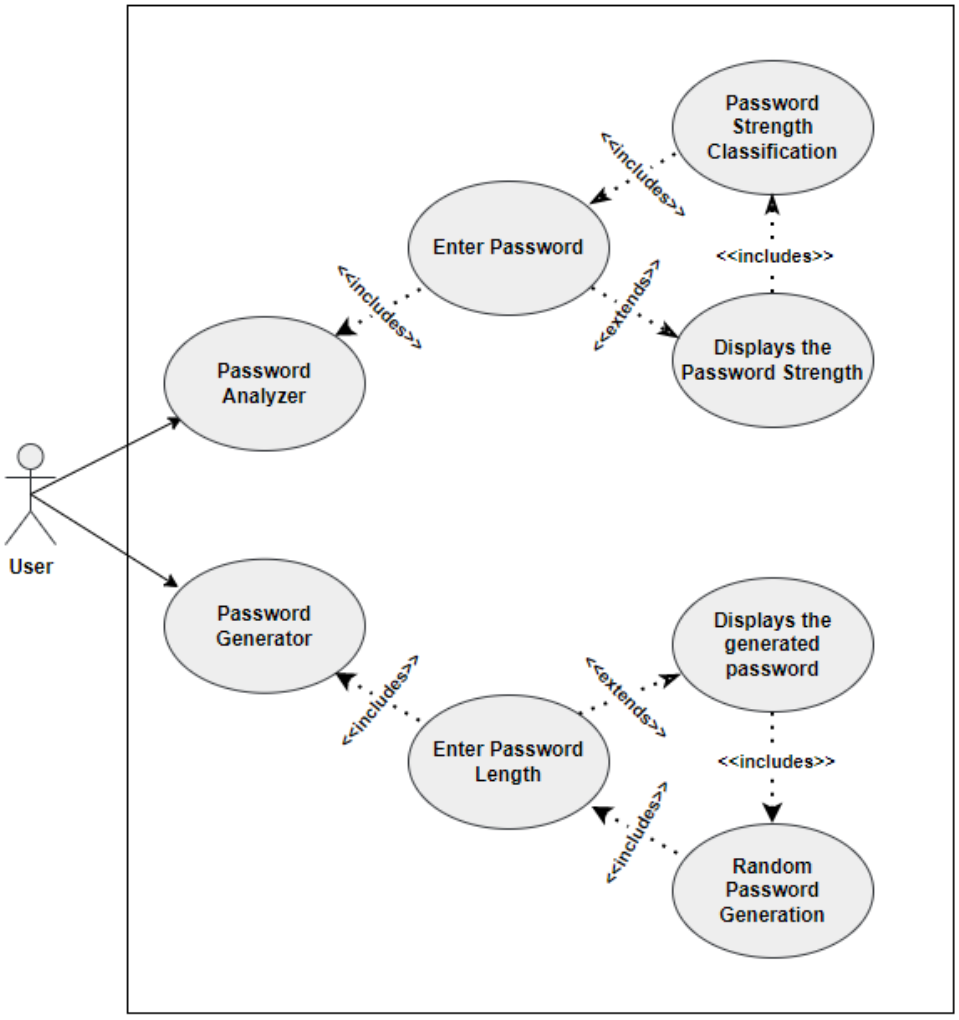


Figure 9: Use Case diagram of the system

The various scenarios in which the application can be used are illustrated in the form of use case diagram. The detailed explanation of the use case diagram is described in the appendix section. **(Use Case Diagram: Appendix C)**

3.5.3 Wireframes

A wireframe is a visual representation of the user interface for the password project. It outlines the layout and arrangement of various elements on each screen, including buttons, input fields, navigation menus, and other UI components. Wireframes help to establish the overall structure and functionality of the application, ensuring that it meets user needs and expectations (Balsamiq, 2023).

.

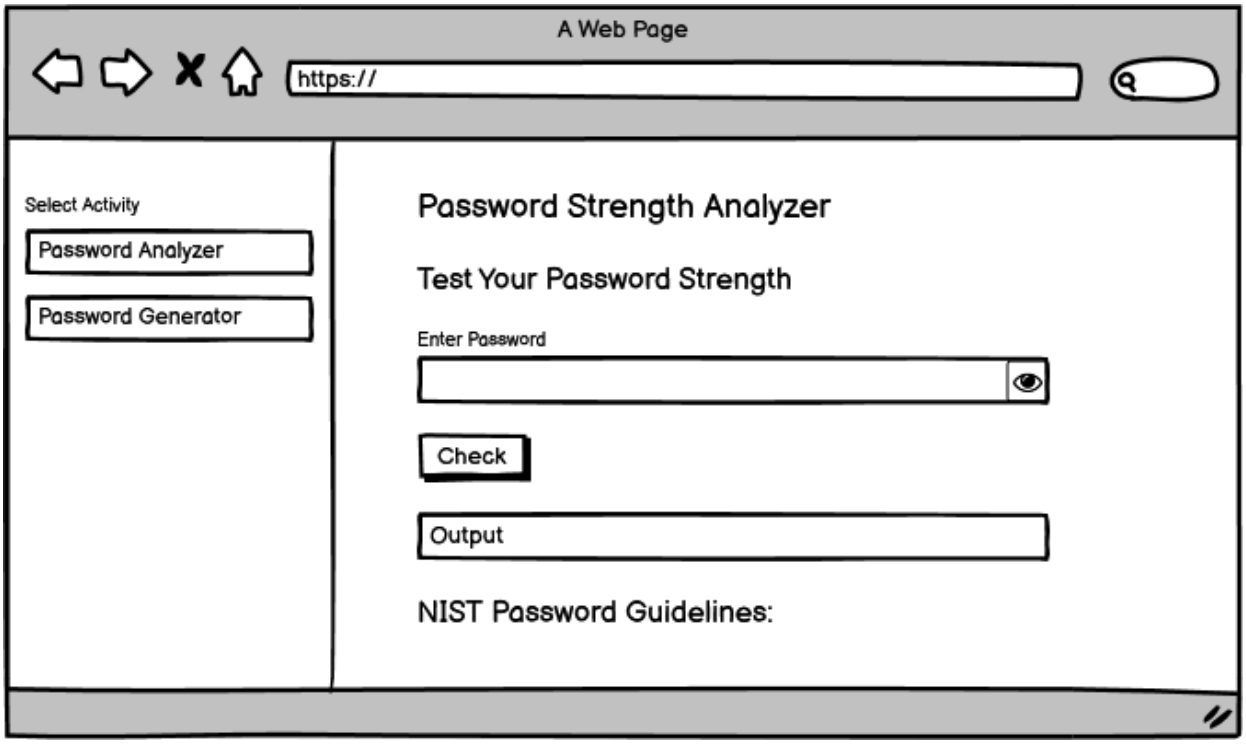


Figure 10: Wireframe of Password Analyzer

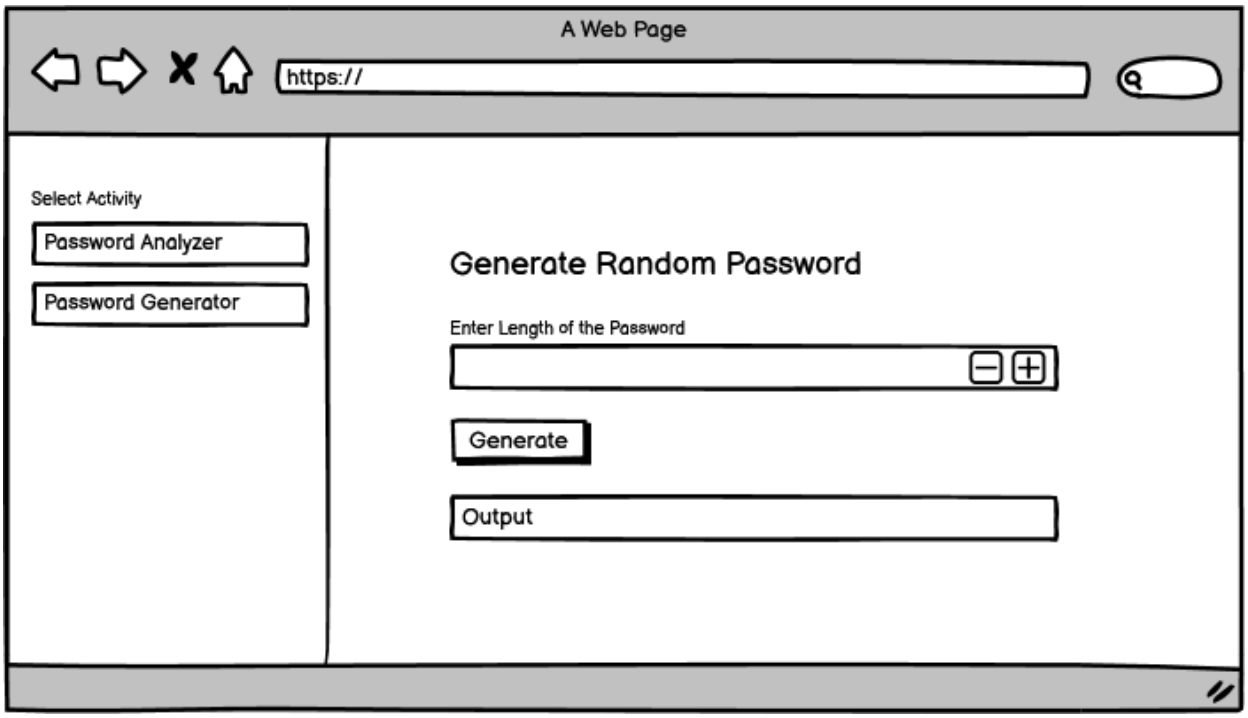


Figure 11: Wireframe of Password Generator

The design and wireframes of the Graphical User Interface of the application have been elaborated in the appendix section. **(Wireframes: Appendix C)**

3.6 Implementation

The deployment section of the report is divided into two main subsections, covering the process of initializing a Git repository, committing changes, pushing the files to GitHub, and ultimately deploying the application using Streamlit Cloud. This process ensures that the app is hosted on a reliable platform, making it accessible to users via the internet.

3.6.1 Version Control with Git and GitHub

The process began with creating a local Git repository within the project directory using the “git init” command. This command sets up a new Git repository, allowing for the tracking of changes to the files within the project. Following the initialization of the repository, the project files were added to the Git staging area using the git add command, which signifies that they are ready to be committed.

Once the files were staged, the “git commit” command was used to create a snapshot of the project at that point in time, with a brief description of the changes made. This process of committing changes allows for easier tracking of project history and the ability to revert to a previous version if necessary.

To maintain a backup of the project, the committed files were pushed to a remote GitHub repository from Visual Studio Code using the “git push” command. This action ensures that the project files are accessible to others, allowing them to view, clone, or contribute to the project. Additionally, the commit history can be viewed using the “git log” command or by navigating to the commit history page in the GitHub repository.

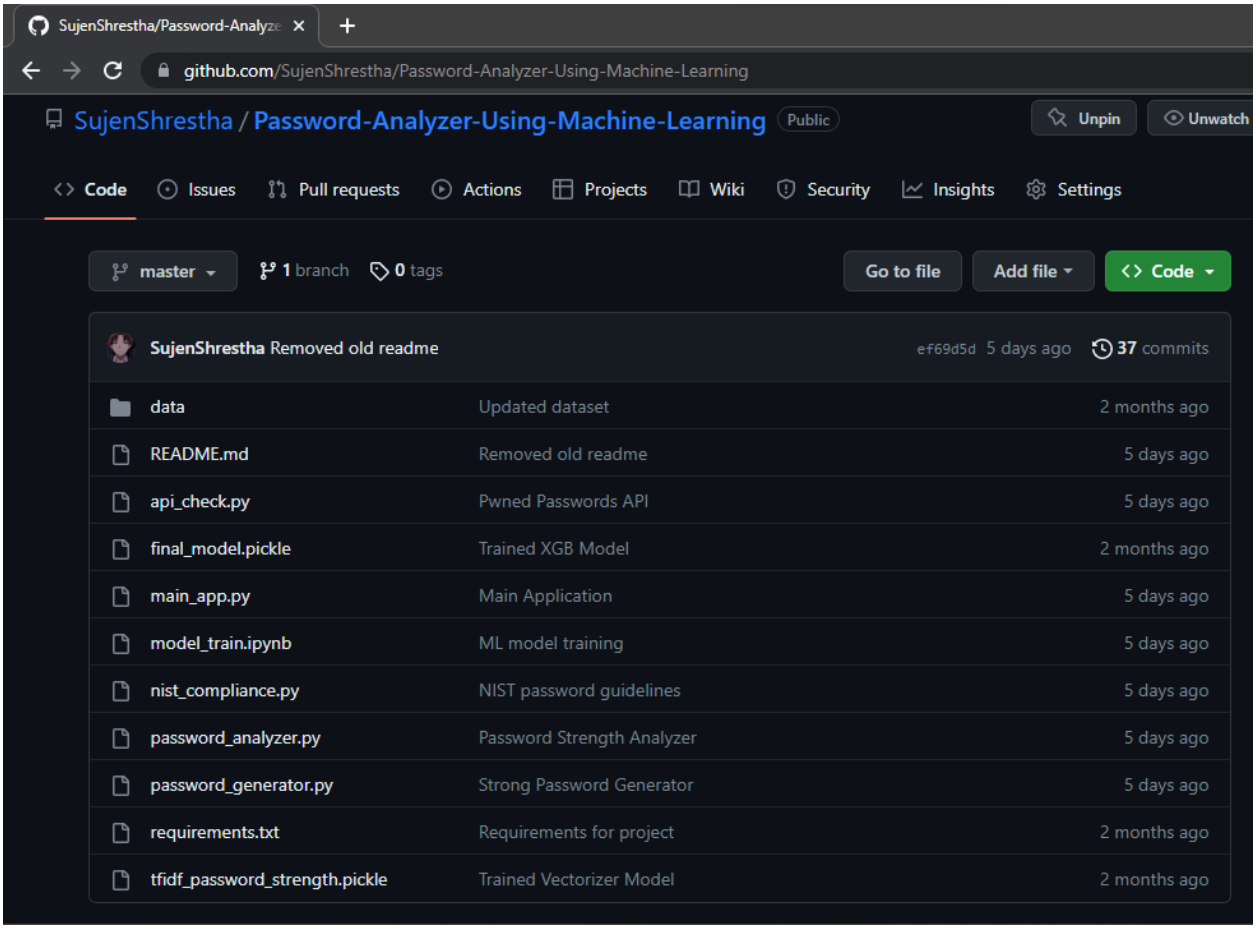


Figure 12: Code pushed to GitHub Repository

3.6.2 Deployment with Streamlit Cloud

Streamlit Community Cloud was chosen as the hosting platform for the application, as it provides a user-friendly environment for deploying web applications built with streamlit framework. The deployment process involved a few key steps to ensure a successful launch of the app.

First, the Streamlit Cloud account was connected to the GitHub repository containing the project files. This connection allows Streamlit Cloud to access the necessary files for deploying the application. Then, the appropriate repository and the main python file containing the code of application were selected within the Streamlit Cloud interface to specify the path containing code files from GitHub.

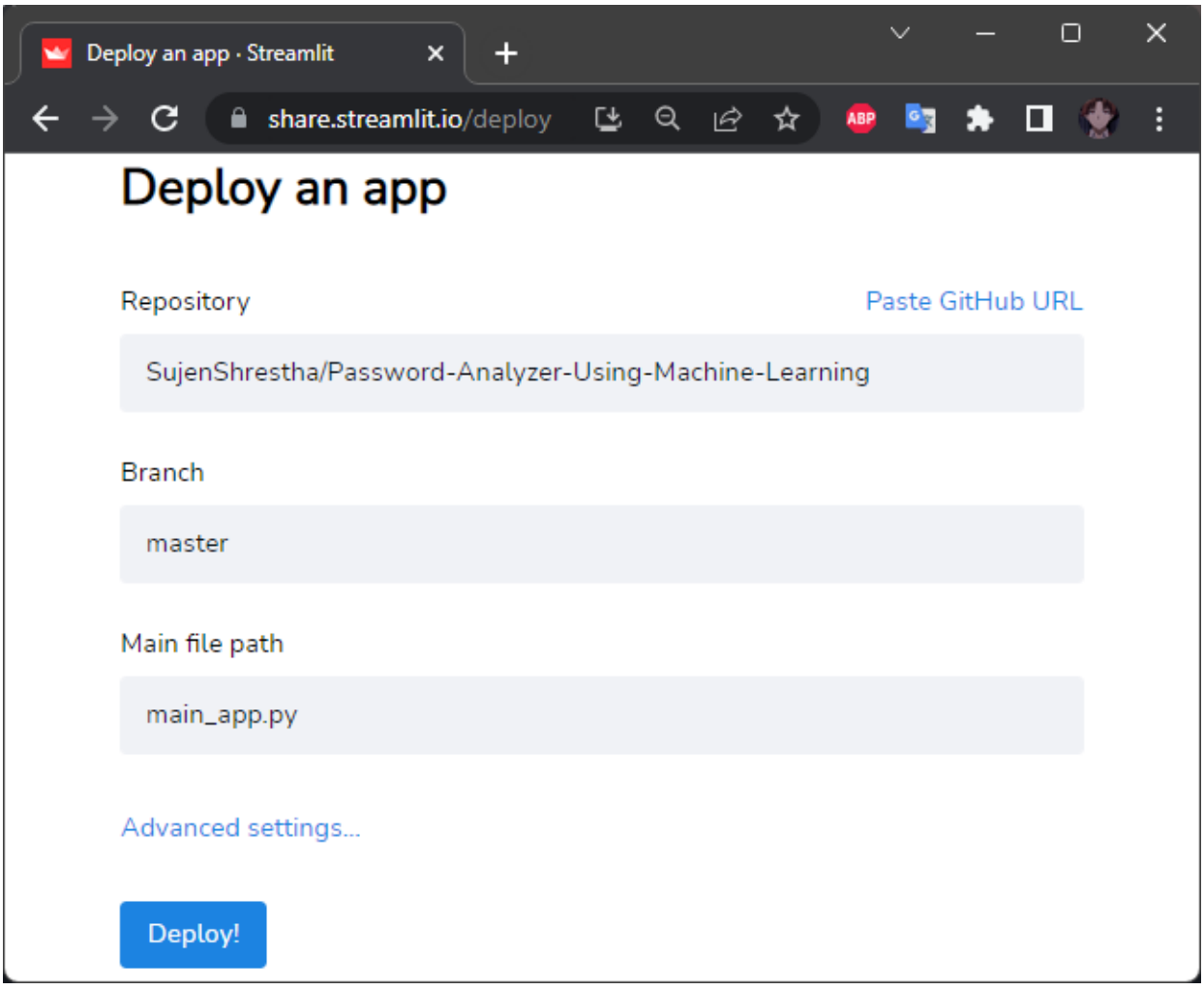


Figure 13: Repository Selection for Deployment using Streamlit Cloud

After configuring the settings, the "Deploy" button was clicked within Streamlit Cloud to initiate the deployment process. During this process, Streamlit Cloud automatically installed the required dependencies listed in the "requirements.txt" file from the GitHub repository, ensuring that the app would function correctly once deployed.

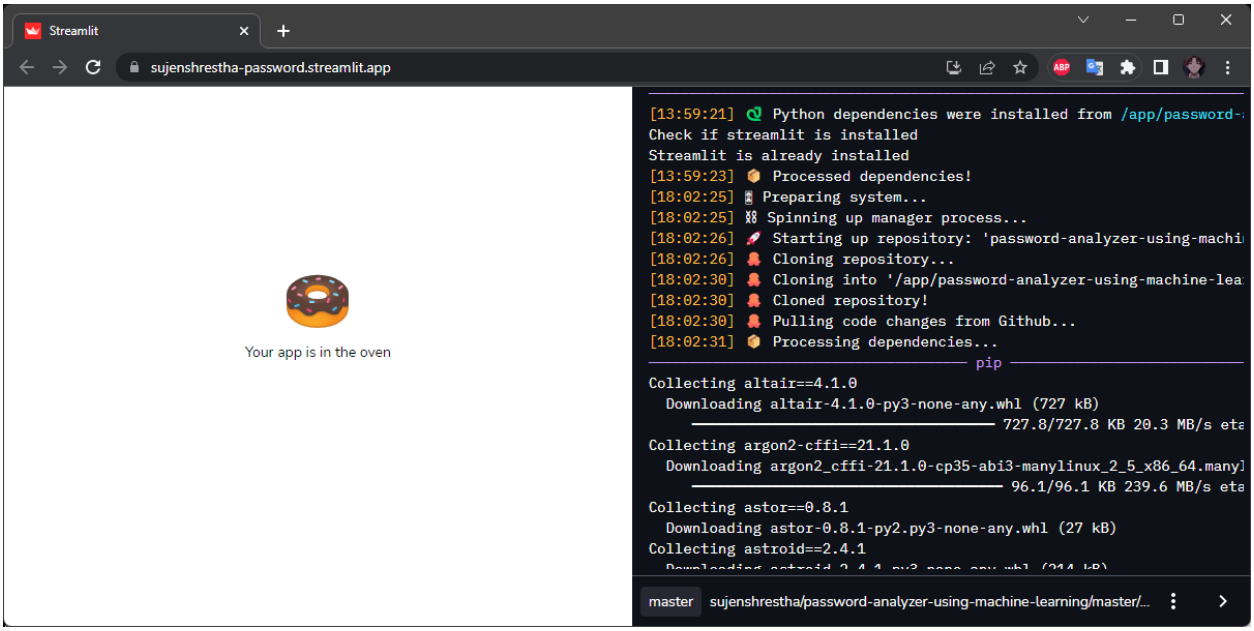


Figure 14: Application initializing in Streamlit Cloud

Upon the completion of the deployment process, the application became publicly accessible via a unique URL generated by Streamlit Cloud. This URL can be shared with users, allowing them to access and interact with the web application from any device with an internet connection.

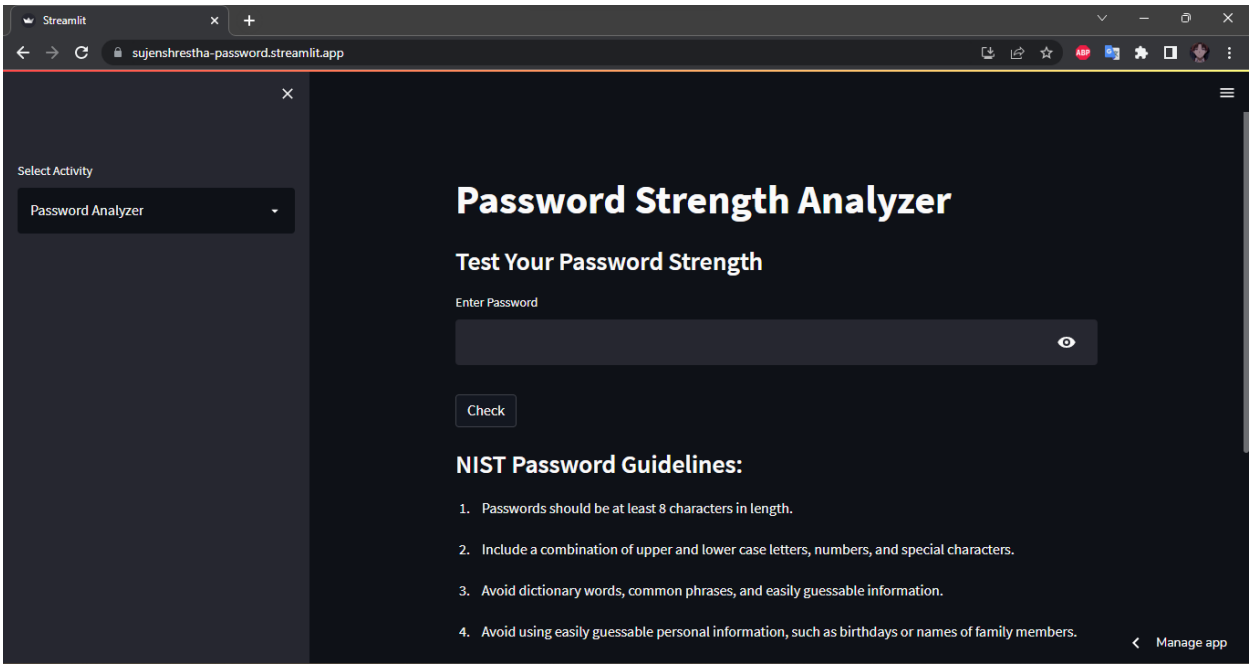


Figure 15: Application Running in Streamlit Cloud

In summary, by following this deployment process and utilizing Git, GitHub, and Streamlit Cloud, the application was successfully deployed, providing users with a reliable and accessible tool for testing and generating passwords.

The various technologies and libraries used for developing the solution are elaborated in the appendix section. **(Implementation: Appendix B)**

4. Testing and Analysis

4.1 Test Plan

The test plan for the password strength analyzer application consists of two main testing phases: Unit Testing and System Testing. The purpose of these tests is to ensure the functionality and reliability of the application, as well as to identify and resolve any potential issues or bugs. The following sections outline the test plan for each phase.

4.1.1 Unit Testing, Test Plan

Unit testing, or white box testing, is a method that focuses on checking individual modules of an application to make sure they work as intended (Panigrahi, 2023). In this project, the unit testing process tested buttons and interactive elements within the program's user interface. Custom triggers were used to display specific values when buttons and widgets functioned correctly, ensuring users could interact with the graphical user interface (GUI) without any problems. This testing approach played a crucial role in confirming the reliability and functionality of the system, ultimately leading to a smooth user experience.

Test Case	Objectives
1	To test the functionality of “eye” button to show or hide the password in the text field.
2	To test the functionality of “Check” button to get the result of password strength analysis.
3	To test the functionality of sidebar navigation menu to switch between password analyzer and password generator.
4	To test the functionality of “+” button to increase the length of password.
5	To test the functionality of “-“ button to decrease the length of password.
6	To test the functionality of “Generate” button to get the result of generated password.
7	To test whether password below the length of specified threshold can be generated.
8	To test whether the password above the length of specified threshold can be generated.
9	To test the functionality of “close” button of the navigation sidebar.
10	To test the “expand” button of the navigation sidebar.

Table 2: Test Plan for Unit Testing

4.1.2 System Testing, Test Plan

System testing, often referred to as black box testing, is a method employed to validate the overall functionality of a software product. This approach encompasses both basic module testing and integration testing including multiple modules, setting it apart from unit testing (Panigrahi, 2023). In this project, system testing was used to assess key features, checking the if the system was functioning as intended and if the desired output was obtained. The results from trained machine learning algorithm and Pwned Passwords API were tested for each required output. By conducting system testing, the project ensured that all features were functioning correctly and that the software operated as a cohesive, reliable system, ultimately delivering a seamless user experience.

Test Case	Objectives
1	To test the output when entered a weak password.
2	To test the output when entered an average password.
3	To test the output when entered a strong password.
4	To test the output when a password is entered which has been breached before.
5	To test the output when a password is entered which has not been breached before.

Table 3: Test Plan for System Testing

4.2 Unit Testing

4.2.1 Test Case 1

Test Case 1	
Objective	To test the functionality of “eye” button to show or hide the password in the text field.
Action	To click on the “eye” button and see whether the hidden text is visible.
Expected Result	The hidden text in the password field should become visible.
Actual Result	The hidden text in the password field was visible.
Conclusion	The test is successful.

Table 4: Testing the eye button functionality

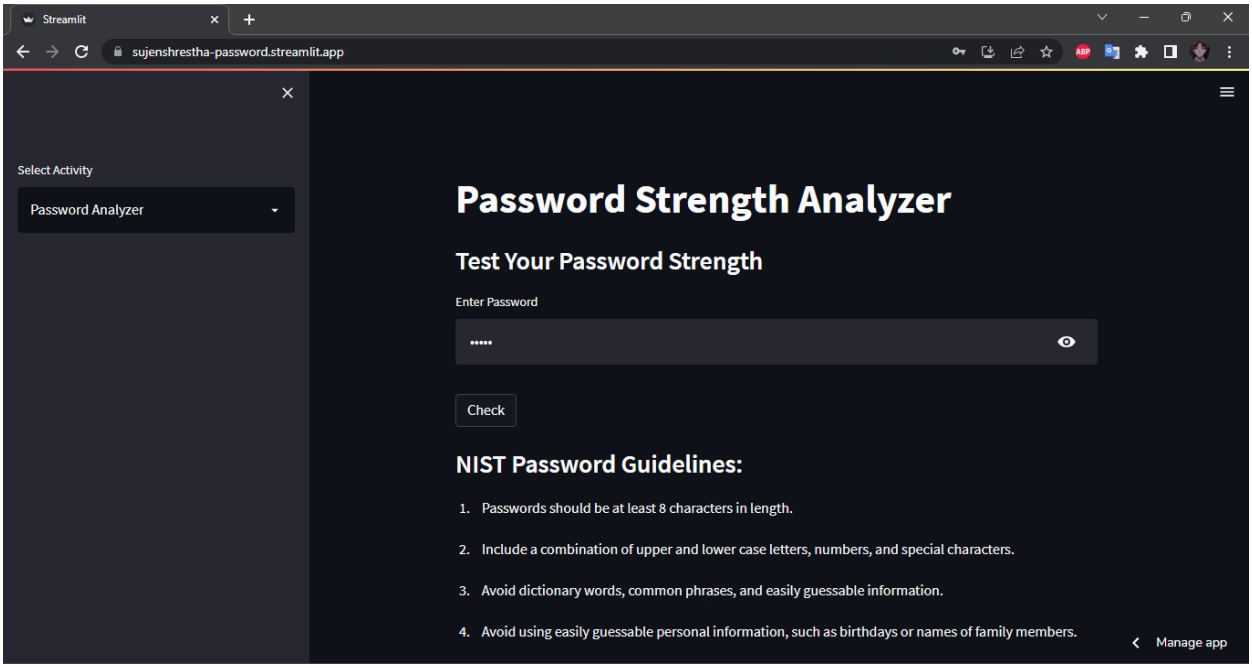


Figure 16: Password hidden in the text field

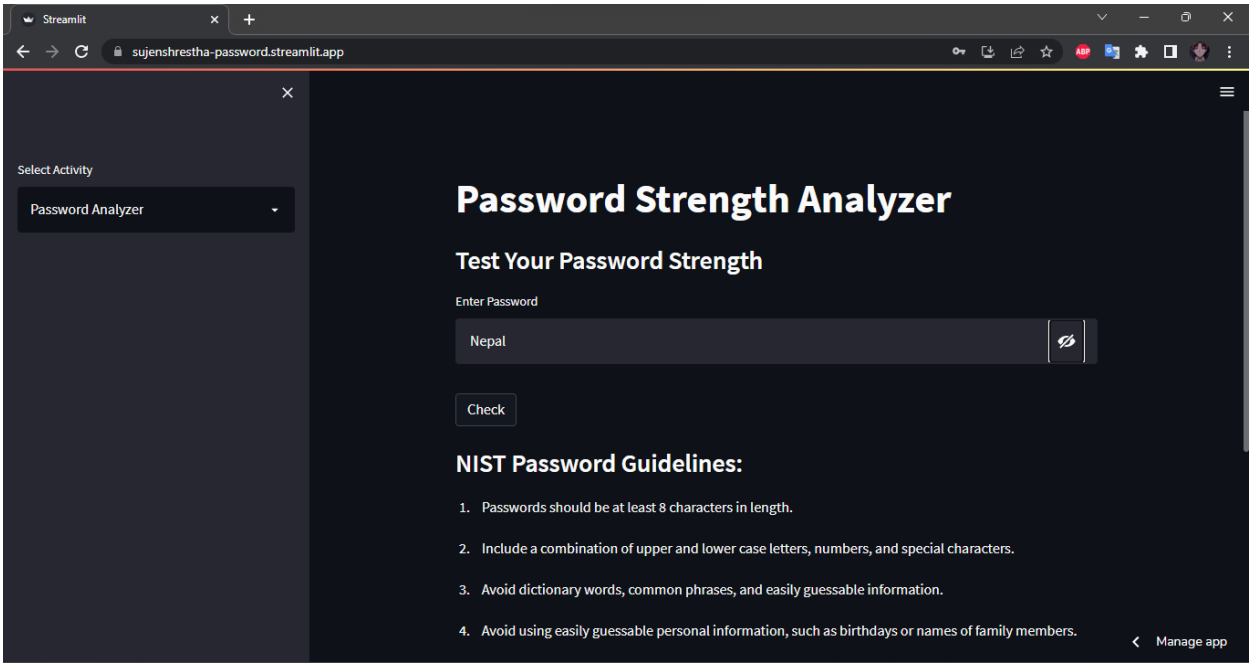


Figure 17: Password displayed after clicking the eye button

4.2.2 Test Case 2

Test Case 2	
Objective	To test the functionality of “Check” button to get the result of password strength analysis.
Action	To click on the “Check” button and see whether the result is obtained.
Expected Result	The result of the password strength and breach status should be displayed.
Actual Result	The result of the password strength and breach status were displayed.
Conclusion	The test is successful.

Table 5: Testing the Check button functionality

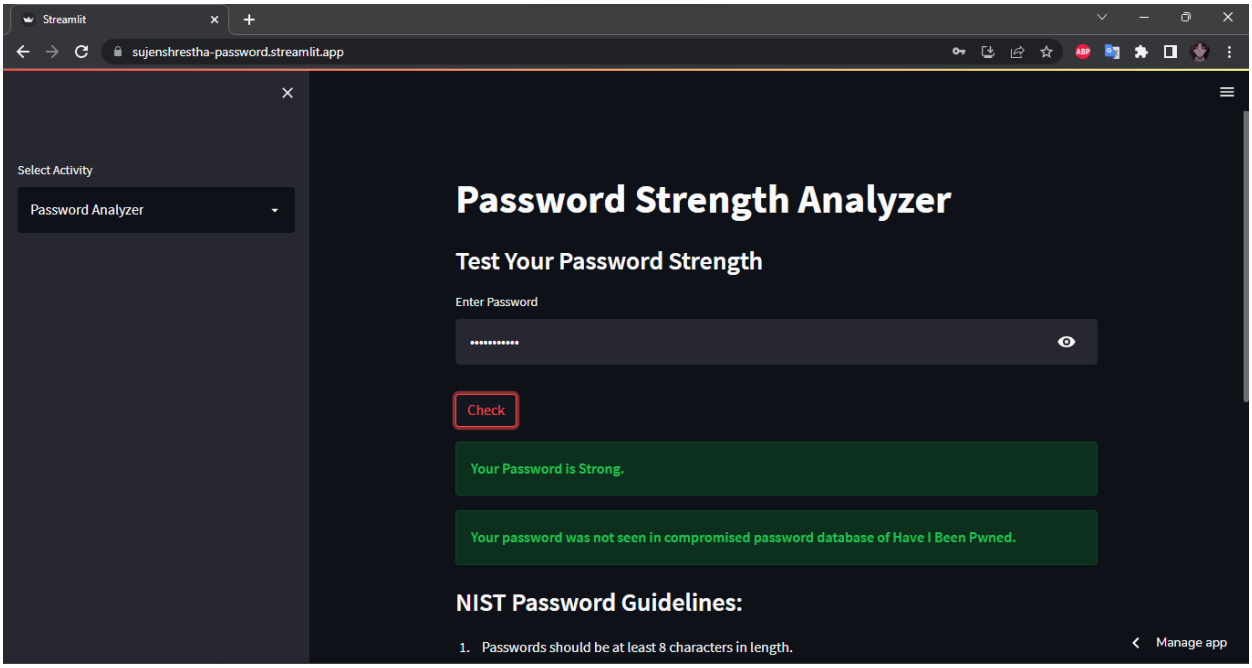


Figure 18: Result obtained after clicking the Check button

4.2.3 Test Case 3

Test Case 3	
Objective	To test the functionality of sidebar navigation menu to switch between password analyzer and password generator.
Action	To click on the down arrow in the navigation menu and select option from sub-menu to see whether another page is loaded.
Expected Result	The web page should be changed to the option selected in the sub menu.
Actual Result	The web page was changed to the option as selected in the sub menu.
Conclusion	The test is successful.

Table 6: Testing the sidebar navigation functionality

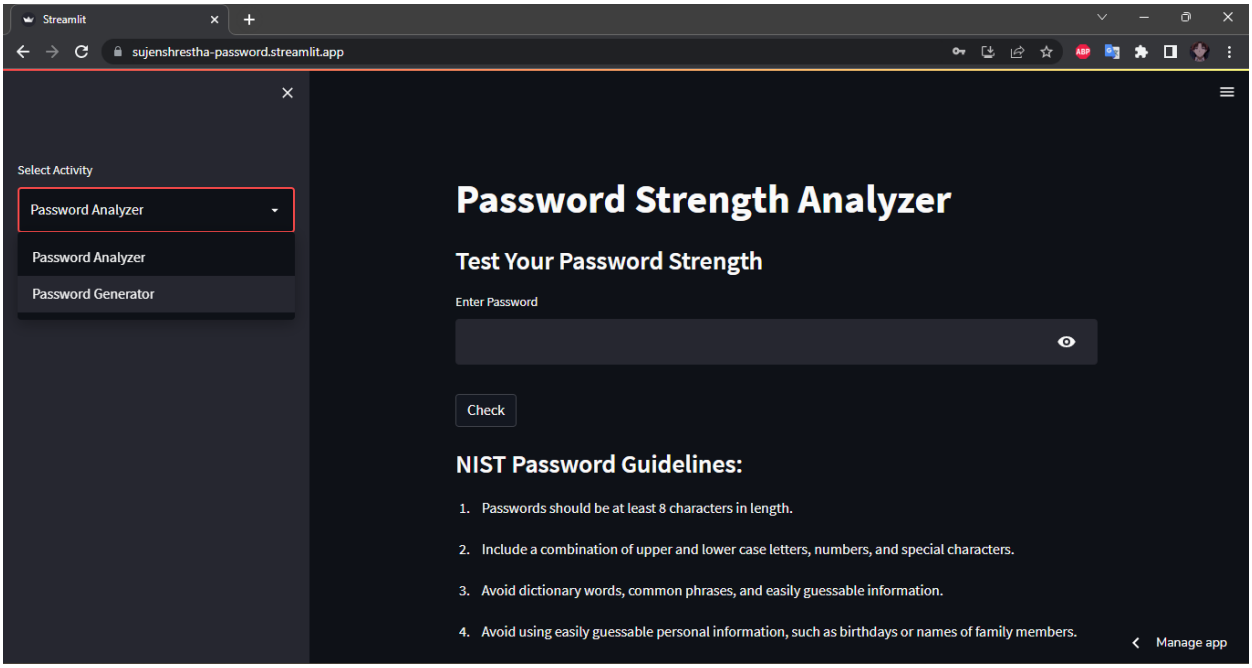


Figure 19: Selecting the sidebar navigation sub-menu

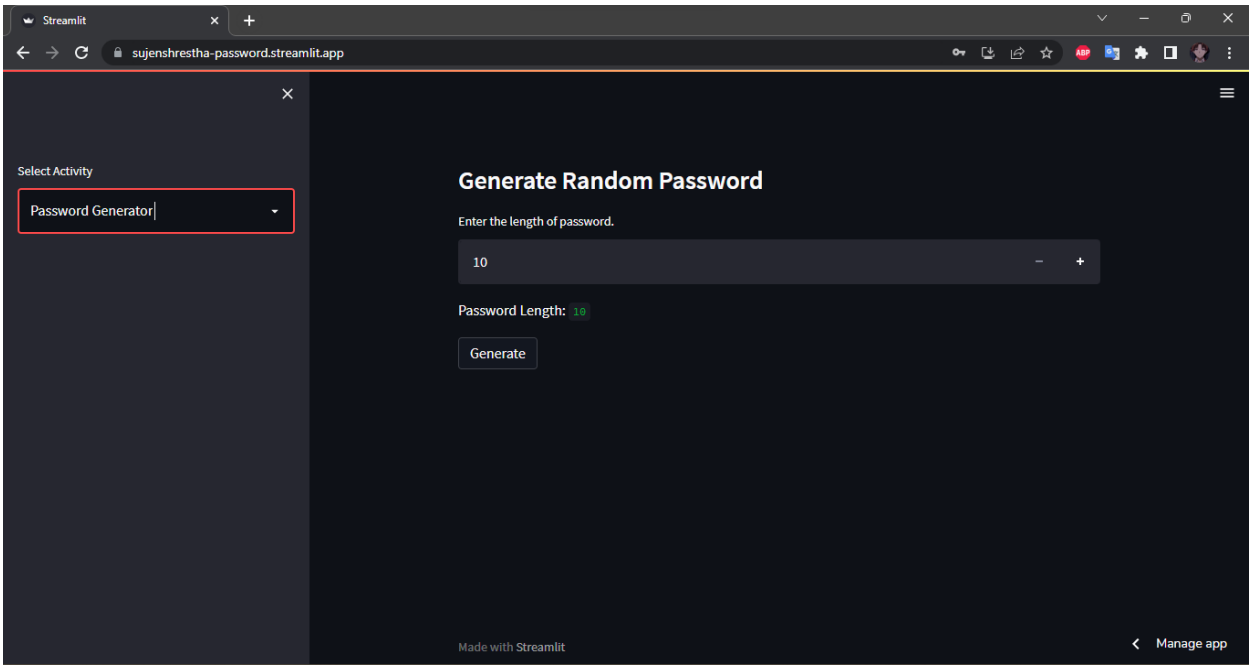


Figure 20: Web page loaded after clicking the option from navigation sidebar

4.2.4 Test Case 4

Test Case 4	
Objective	To test the functionality of “+” button to increase the length of password.
Action	To click on the “+” button and see if the length of password is increased.
Expected Result	The length of the password should be increased when the “+” button is clicked.
Actual Result	The length of the password was increased when the “+” button was clicked.
Conclusion	The test is successful.

Table 7: Testing the length increment button functionality

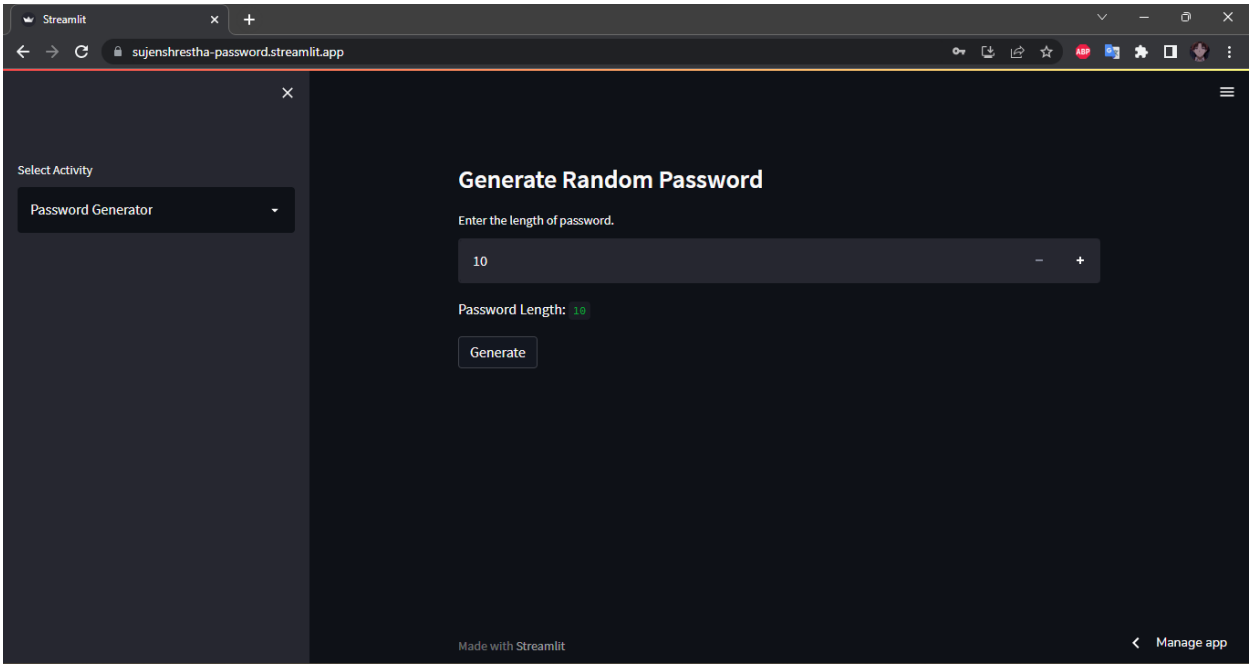


Figure 21: The length of password before clicking the "+" button

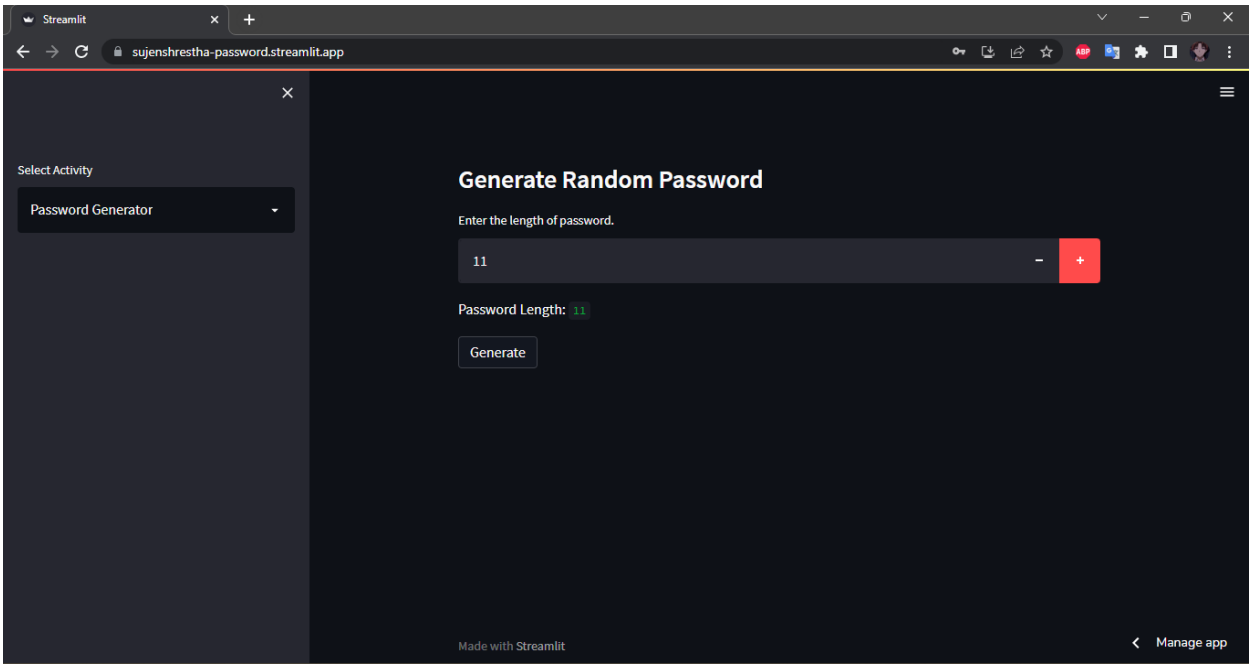


Figure 22: The length of password after clicking the "+" button

4.2.5 Test Case 5

Test Case 5	
Objective	To test the functionality of "-" button to decrease the length of password.
Action	To click on the "-" button and see if the length of the password is decreased.
Expected Result	The length of the password should be decreased when the "-" button is clicked.
Actual Result	The length of the password was decreased when the "-" button was clicked.
Conclusion	The test is successful.

Table 8: Testing the length reduction button functionality

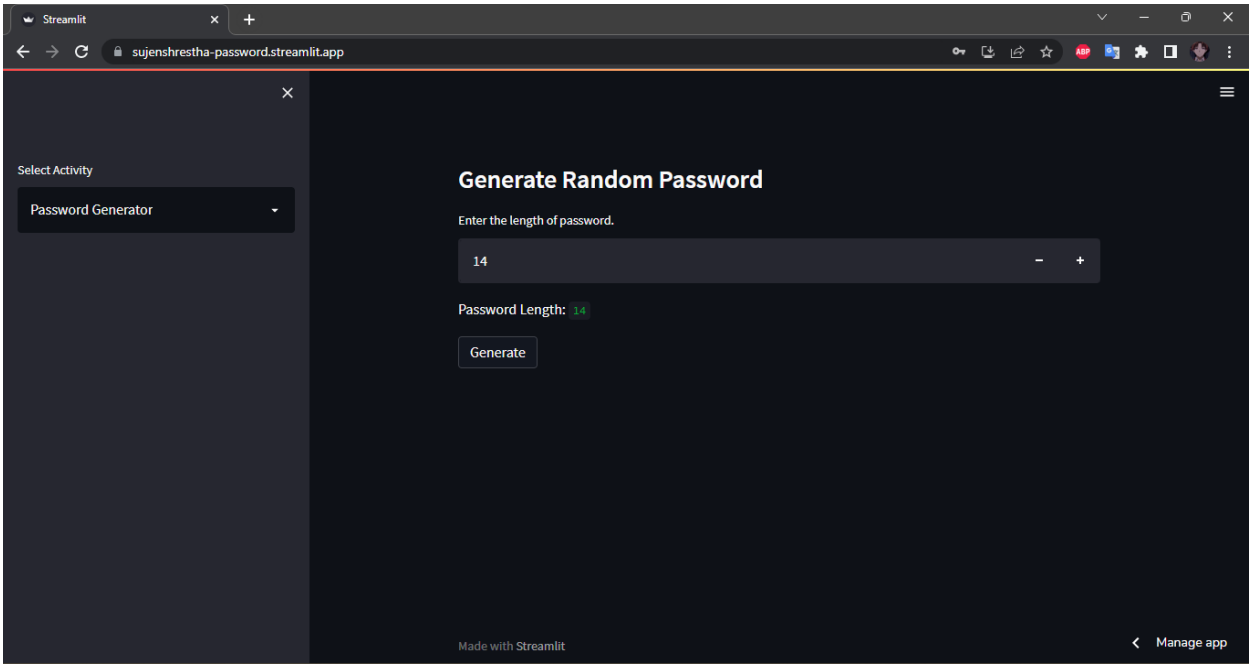


Figure 23: The length of password before clicking the "-" button

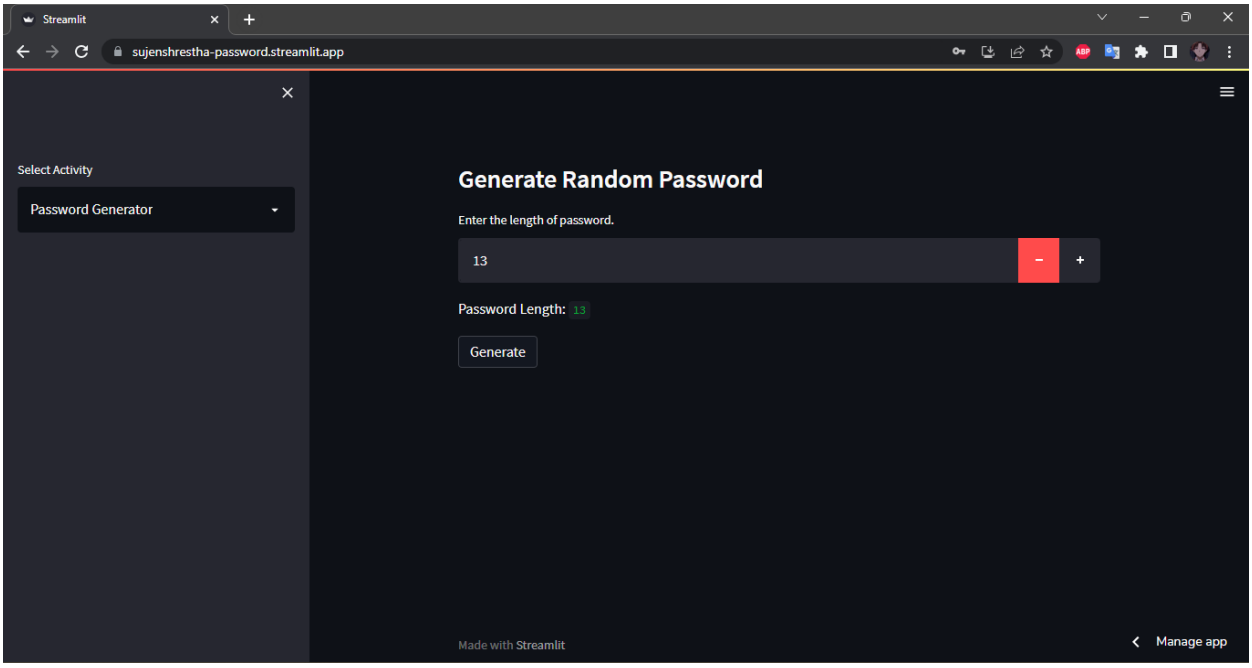


Figure 24: The length of password after clicking "-" button

4.2.6 Test Case 6

Test Case 6	
Objective	To test the functionality of “Generate” button to get the result of generated password.
Action	To click on the “Generate” button and see if a password is generated.
Expected Result	The password should be generated according to the length specified by the user.
Actual Result	The password was generated according to the length specified by the user.
Conclusion	The test is successful.

Table 9: Testing the functionality of Generate button

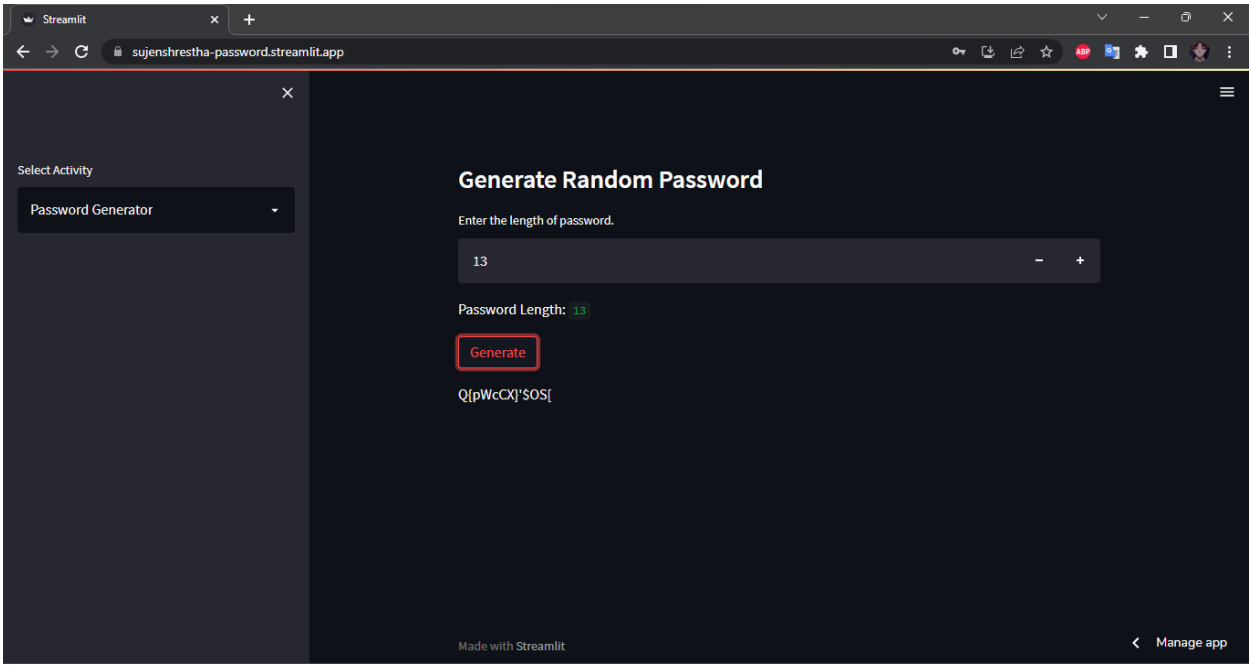


Figure 25: Result obtained after clicking Generate button

4.2.7 Test Case 7

Test Case 7	
Objective	To test whether password below the length of specified minimum threshold limit can be generated.
Action	To enter a number less than minimum threshold and see whether the password is generated.
Expected Result	When a number less than specified minimum threshold is entered, an error message should be displayed.
Actual Result	When a number less than specified minimum threshold was entered, an error message was displayed.
Conclusion	The test is successful.

Table 10: Testing the minimum password generation limit

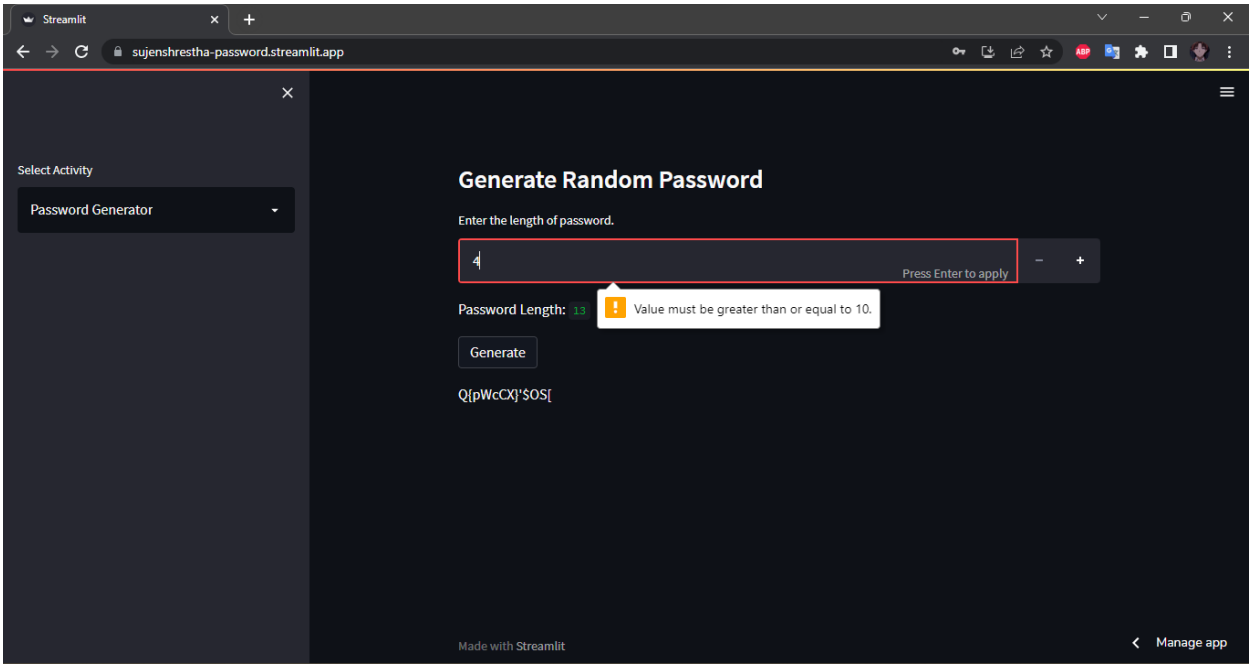


Figure 26: Password length entered below the minimum threshold limit

4.2.8 Test Case 8

Test Case 8	
Objective	To test whether the password above the specified maximum threshold limit can be generated.
Action	To enter a number greater than specified maximum threshold and see whether the password is generated.
Expected Result	When a number greater than specified maximum threshold is entered, an error message should be displayed.
Actual Result	When a number greater than 64 was entered, an error message was displayed.
Conclusion	The test is successful.

Table 11: Testing the maximum password generation limit

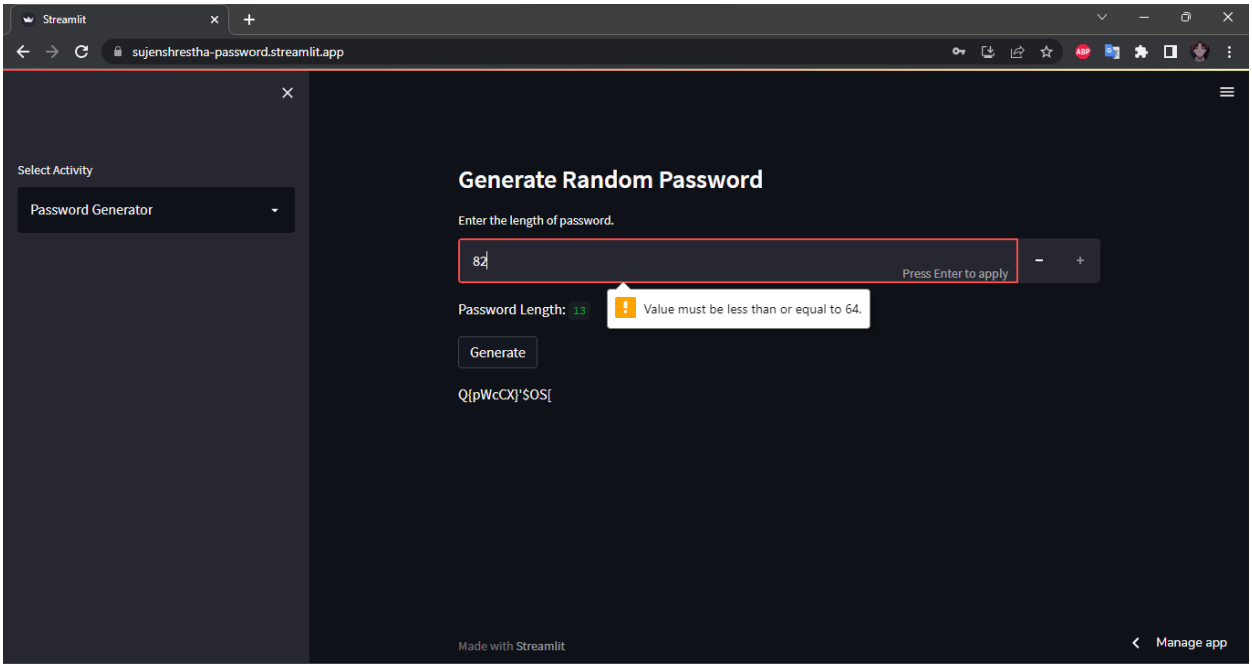


Figure 27: Password length entered beyond the maximum threshold limit

4.2.9 Test Case 9

Test Case 9	
Objective	To test the functionality of “close” button of the navigation sidebar.
Action	To click on the “close” button and see if the sidebar navigation menu gets collapsed.
Expected Result	When the “close” button is clicked, the sidebar should get collapsed.
Actual Result	When the “close” button was clicked, the sidebar got collapsed.
Conclusion	The test is successful.

Table 12: Testing the sidebar collapse button functionality

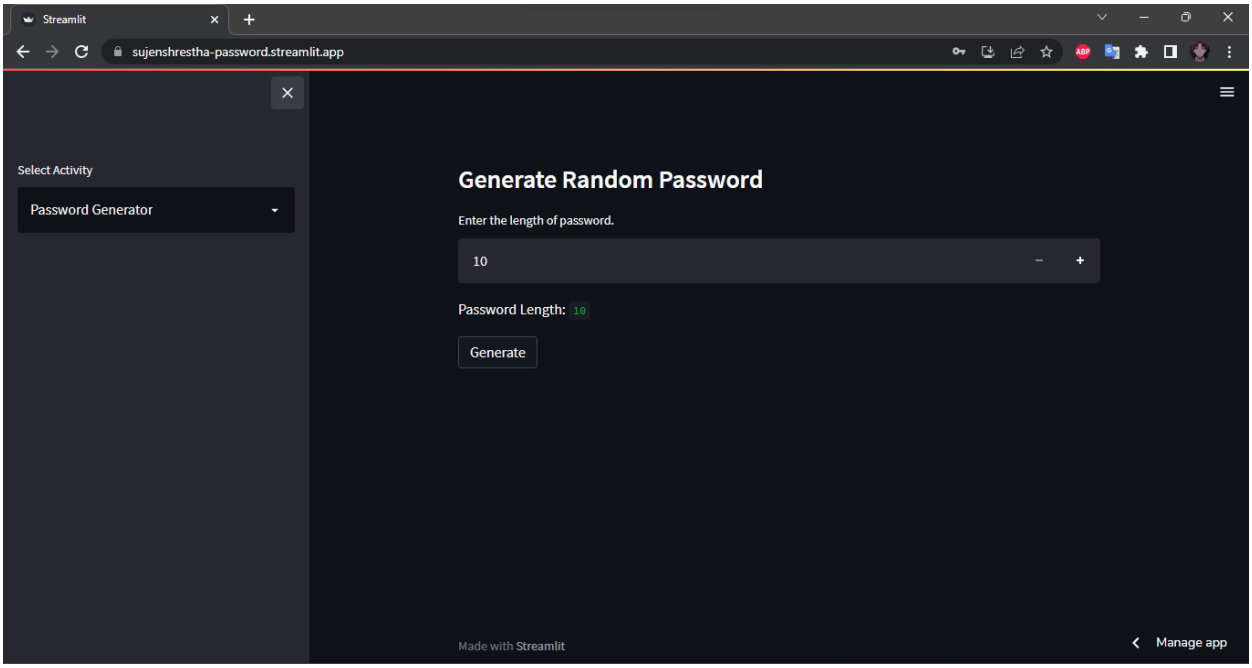


Figure 28: Sidebar menu before clicking the "Close" button

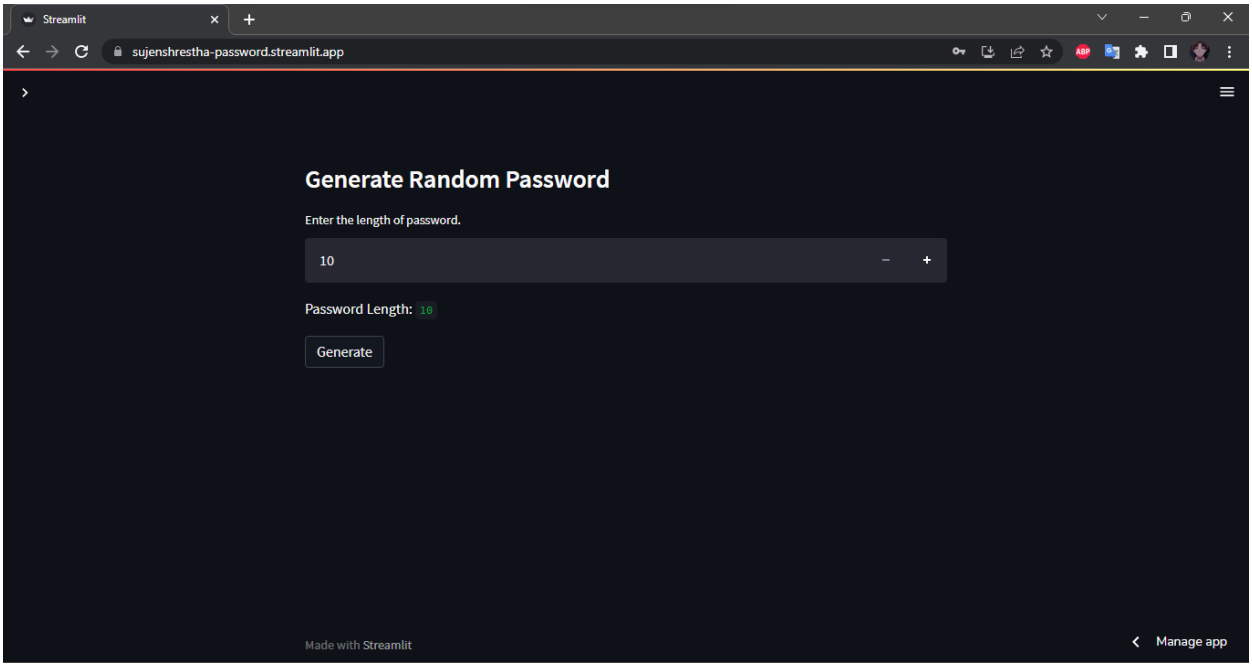


Figure 29: Sidebar menu collapsed after clicking the "Close" button

4.2.10 Test Case 10

Test Case 10	
Objective	To test the “expand” button of the navigation sidebar.
Action	To click on the “expand” button and see if the sidebar navigation menu gets expanded.
Expected Result	When the “expand” button is clicked, the sidebar should get expanded.
Actual Result	When the “expand” button was clicked, the sidebar got expanded.
Conclusion	The test is successful.

Table 13: Testing the sidebar expand button functionality

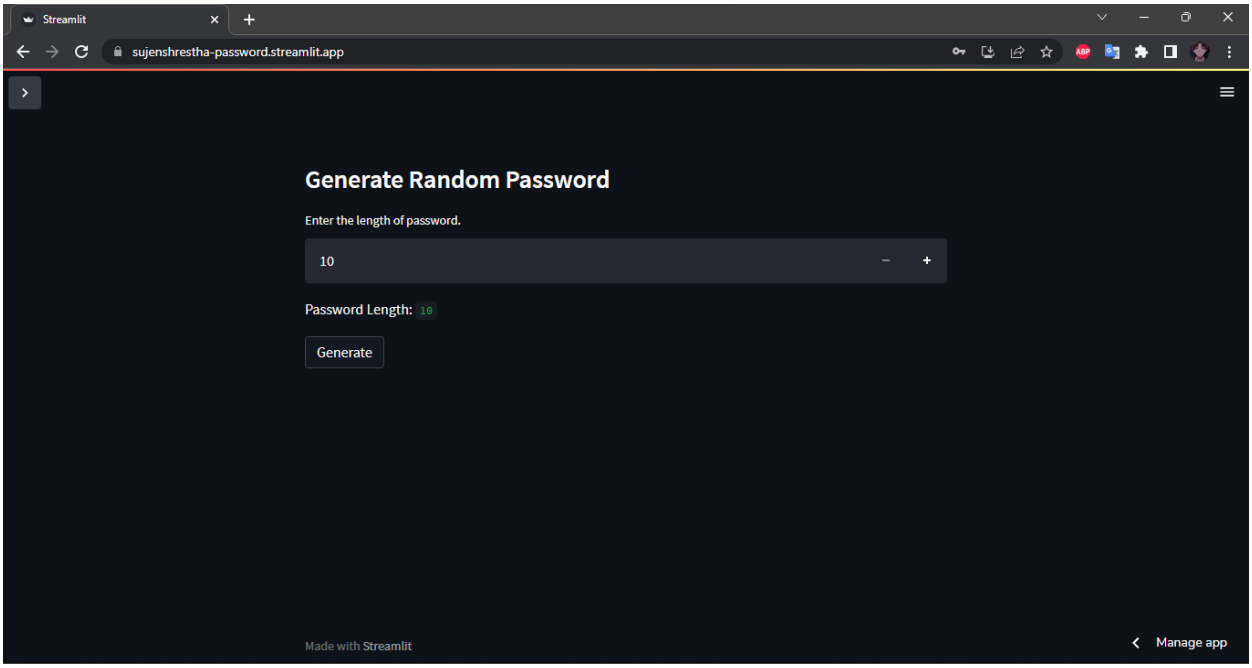


Figure 30: Webpage before clicking the "Expand" sidebar button

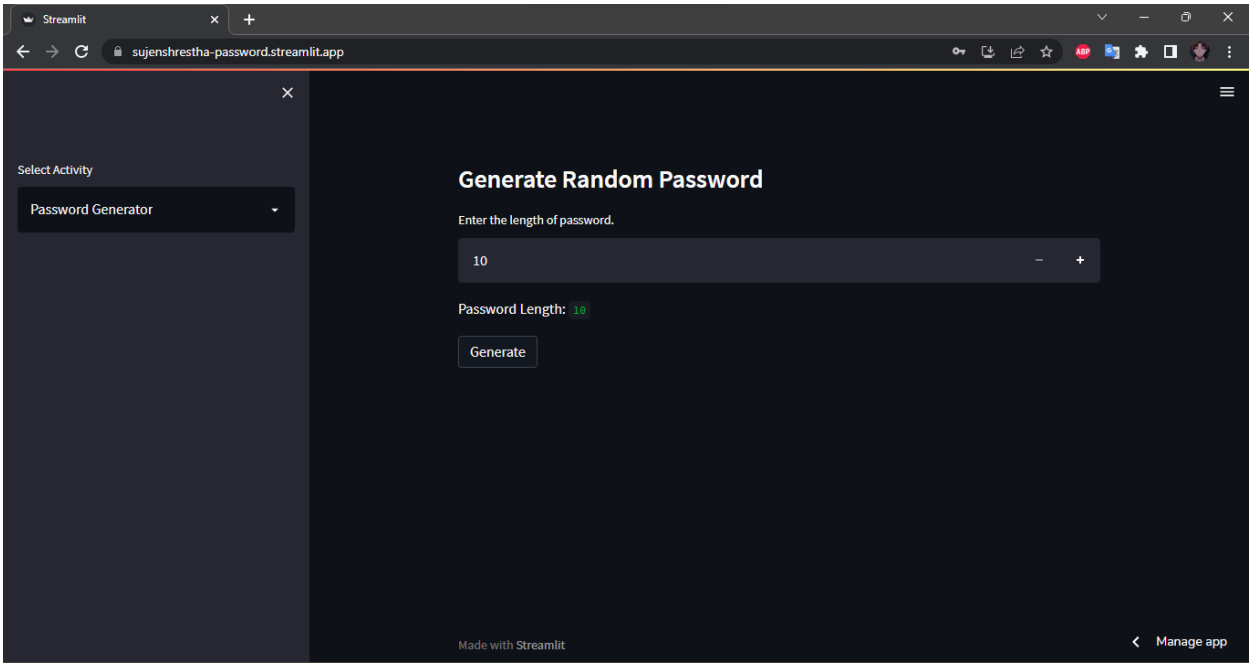


Figure 31: Webpage after clicking the "Expand" sidebar button

4.3 System Testing

4.3.1 Test Case 1

Test Case 1	
Objective	To test the output when entered a weak password.
Action	To enter a password of small length and non-unique characters and check the output.
Expected Result	The result should display that the entered password is weak and suggestions to improve the password should be provided.
Actual Result	The result displayed that the entered password was weak and suggestions to improve the password were provided.
Conclusion	The test is successful.

Table 14: Testing the condition for weak password

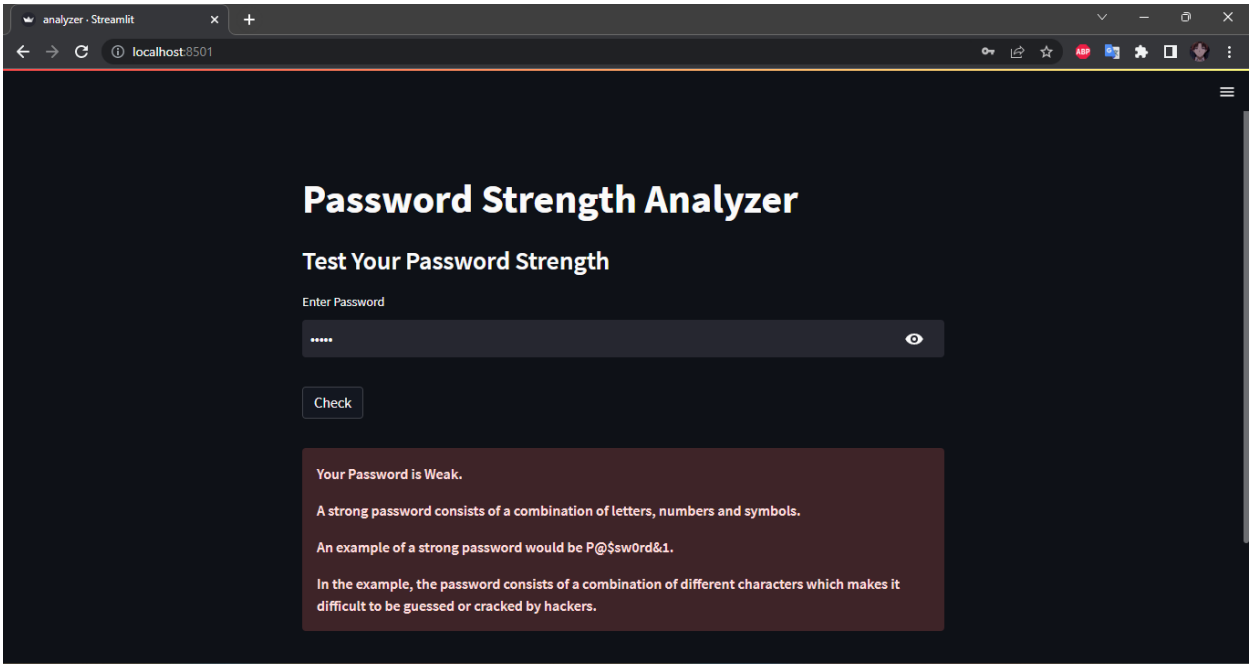


Figure 32: Output after weak password is entered

4.3.2 Test Case 2

Test Case 2	
Objective	To test the output when entered an average password.
Action	To enter a password of average length and some unique characters.
Expected Result	The result should display that the entered password is average and suggestions to improve the password should be provided.
Actual Result	The result displayed that the entered password was average and suggestions to improve the password were provided.
Conclusion	The test is successful.

Table 15: Testing the condition for average password

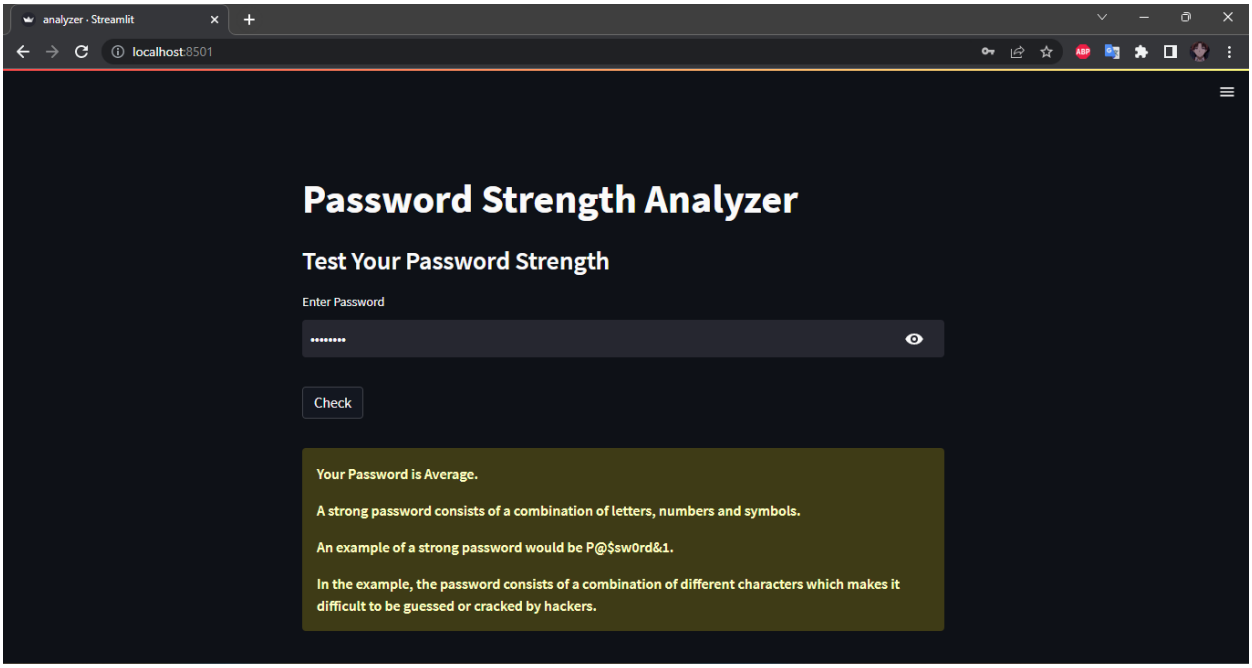


Figure 33: Output after average password is entered

4.3.3 Test Case 3

Test Case 3	
Objective	To test the output when entered a strong password.
Action	To enter a password of high length and various unique characters.
Expected Result	The result should display that the entered password is strong.
Actual Result	The result displayed that the entered password was strong.
Conclusion	The test is successful.

Table 16: Testing the condition for strong password

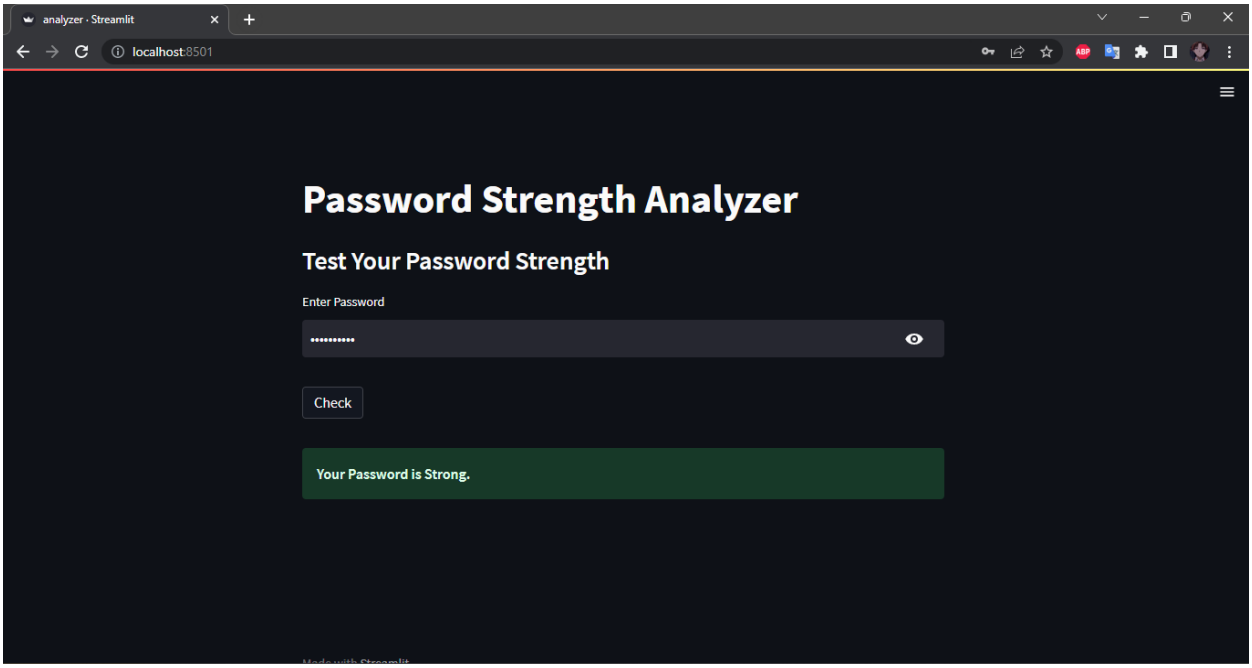


Figure 34: Output after strong password is entered

4.3.4 Test Case 4

Test Case 4	
Objective	To test the output when a password is entered which has been breached before.
Action	To enter a common dictionary word as password which might have been breached before and check the result.
Expected Result	The result should display that the password has been breached along with the number of times it has been seen in a breach.
Actual Result	The result displayed that the password had been breached along with the number of times it was seen in a breach.
Conclusion	The test is successful.

Table 17: Testing the condition for compromised password

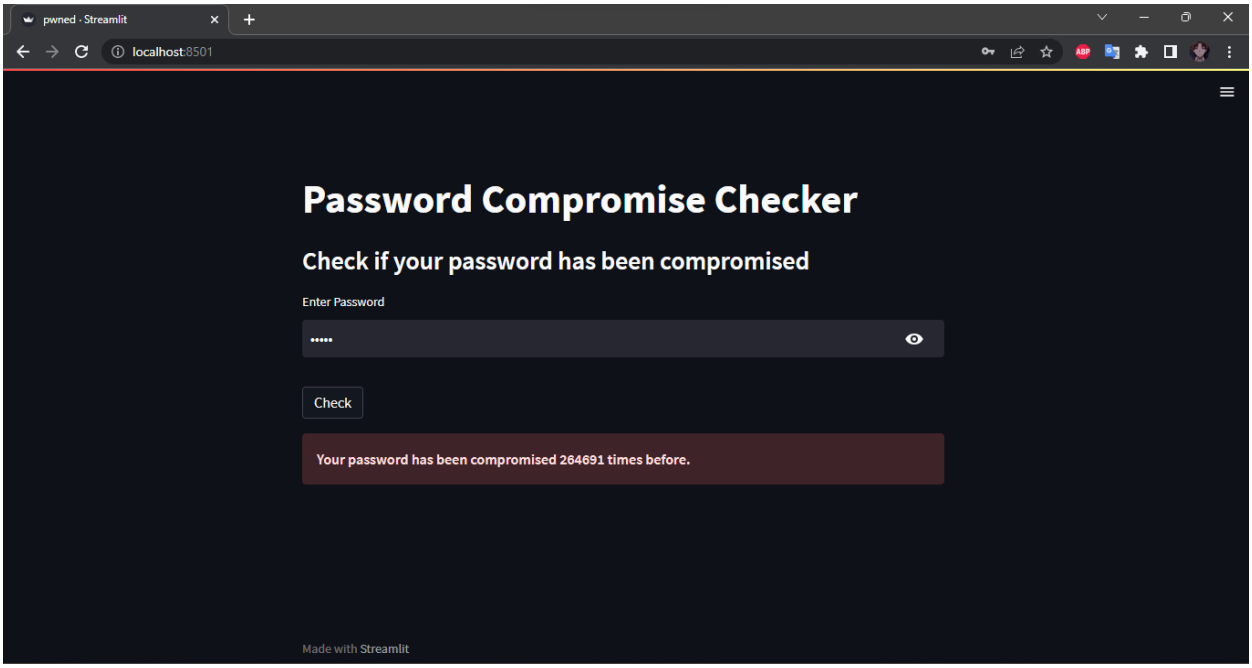


Figure 35: Output when compromised password is entered

4.3.5 Test Case 5

Test Case 5	
Objective	To test the output when a password is entered which has not been breached before.
Action	To enter a unique password with different character combination and check the output.
Expected Result	The result should display that the entered password is not found in Have I Been Pwned database.
Actual Result	The result displayed that the entered password was not found in Have I Been Pwned database.
Conclusion	The test is successful.

Table 18: Testing the condition for secure password

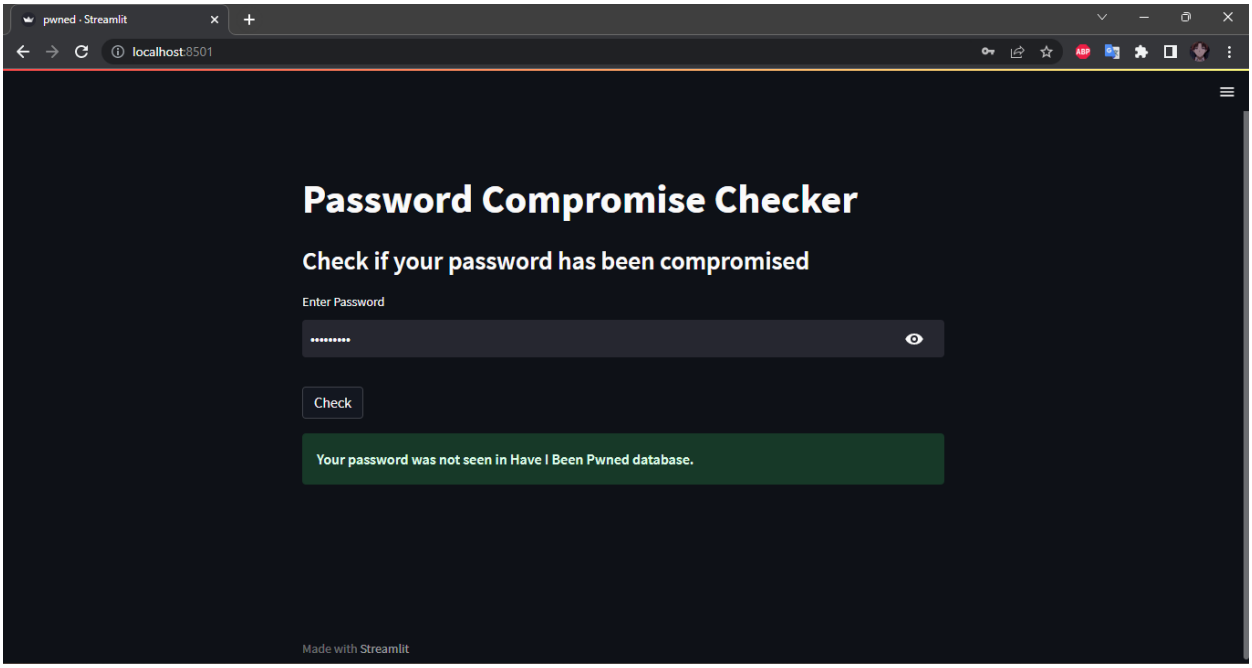


Figure 36: Output when unique password is entered

4.3.6 Test Case 6

Test Case 6	
Objective	To test if the web application works in mobile device.
Action	To use a mobile device and test the functionality of the web application.
Expected Result	The web page should be responsive and work similarly on the mobile device.
Actual Result	The web page was responsive and worked similarly on the mobile device.
Conclusion	The test is successful.

Table 19: Testing the web application on mobile device

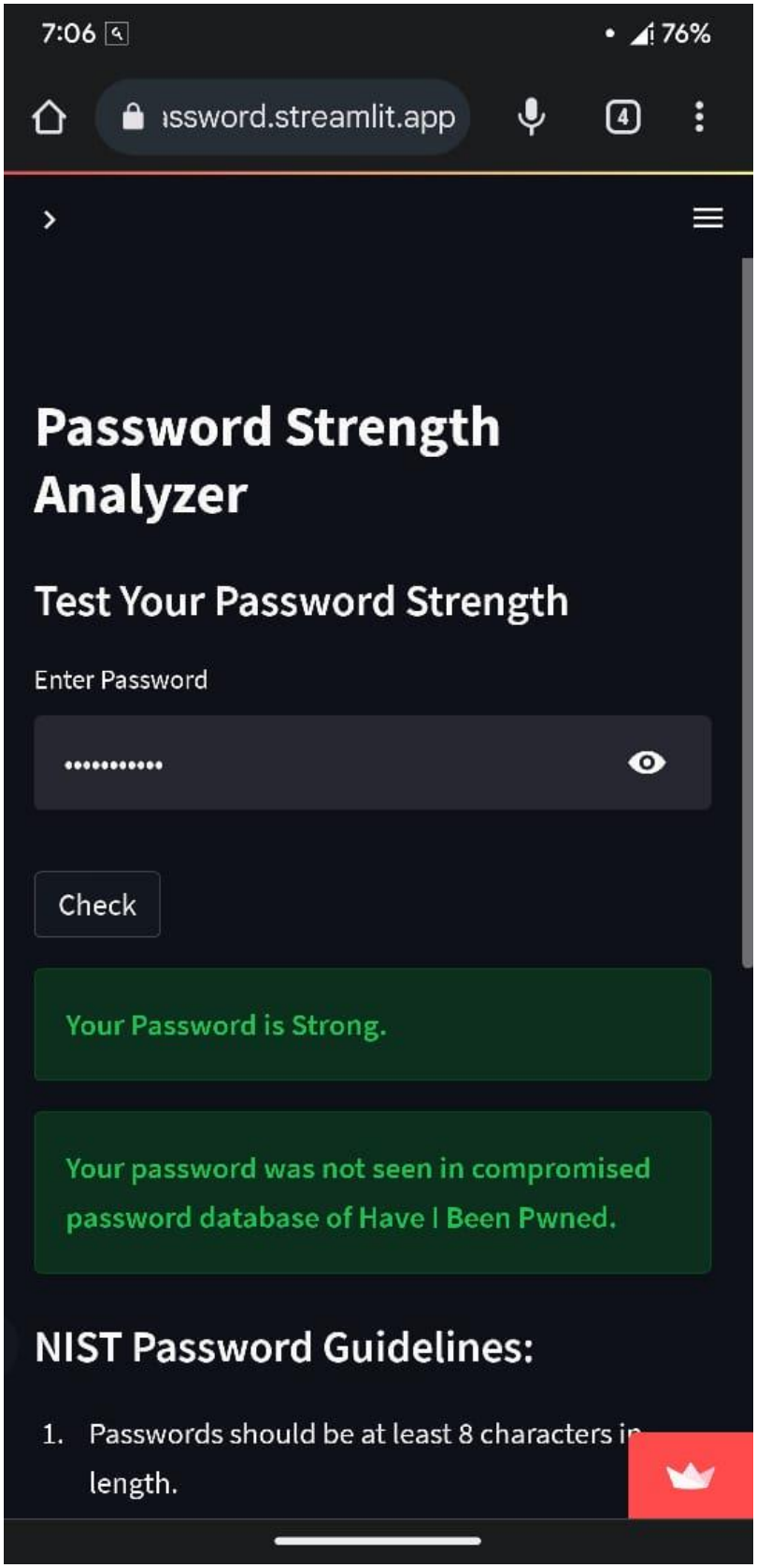


Figure 37: Password analyzer in mobile device

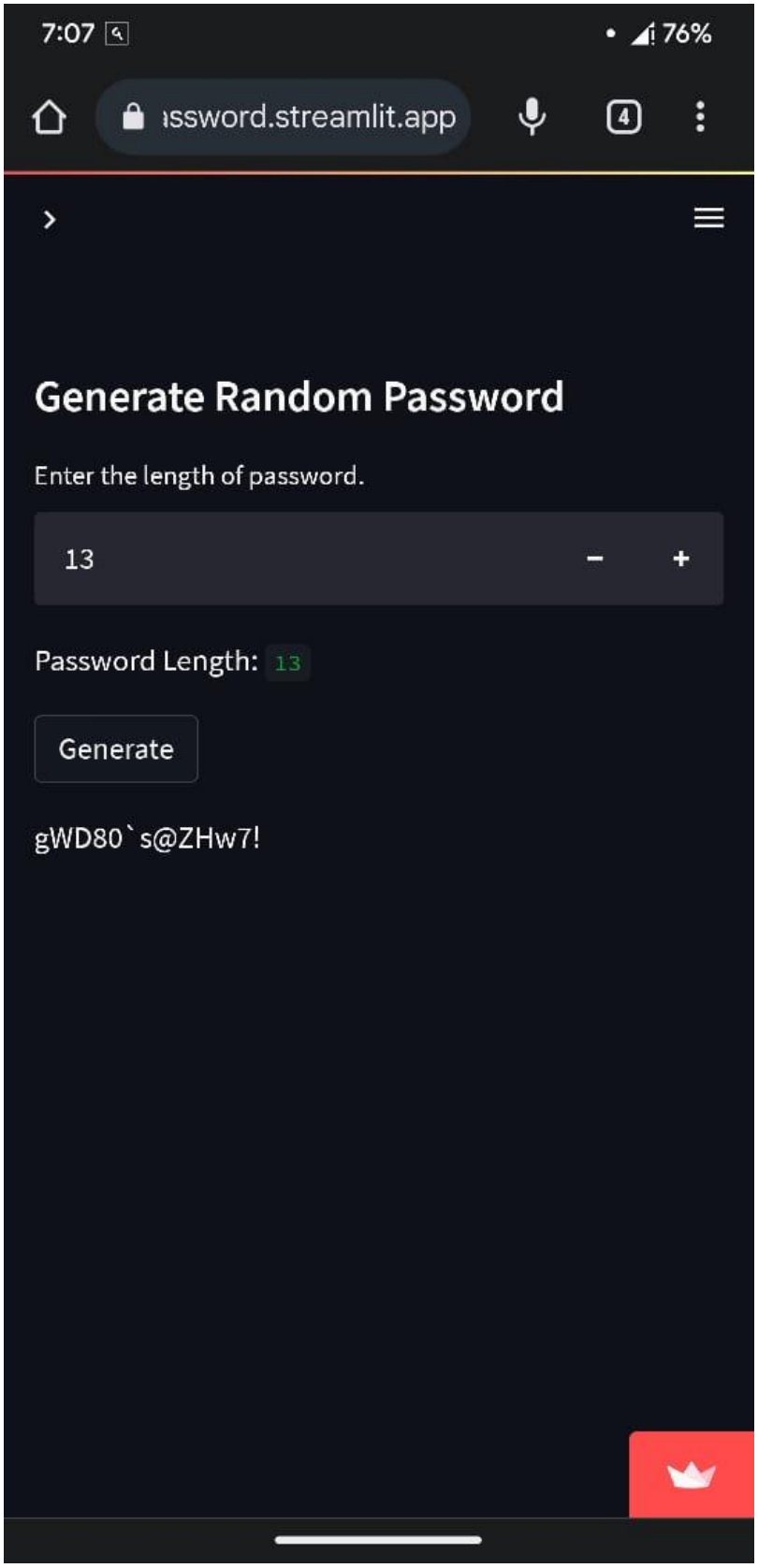


Figure 38: Password generator in mobile device

4.4 Critical Analysis

This section provides a comprehensive critical analysis of the password strength analyzer and generator, focusing on its effectiveness, efficiency, and potential areas of improvement. This analysis will help to evaluate the project's success and identify areas where future enhancements can be made.

4.4.1 Strengths of the application

The password strength analyzer and generator effectively address the need for secure passwords by assessing password strength and providing a tool for generating strong passwords. The application evaluates passwords based on multiple criteria, including length, character variety, and potential breaches in the Have I Been Pwned database. By providing real-time feedback on password strength and breach status, users can make informed decisions about their password choices. Furthermore, the password generator allows users to create strong, randomized passwords with customizable length, enhancing security.

The application's user interface is designed to be intuitive and user-friendly, making it easy for users to understand and interact with the application. The responsive design ensures that the application is accessible on various devices, including desktops, tablets, and smartphones. While the application performs well overall, there is potential for performance optimization to handle larger volumes of password analysis requests and provide faster results to users.

4.4.2 Areas for Improvement

- **Expanded Character Support:** To improve password strength and variety, the application should support a broader range of Unicode characters and special symbols.

- **Integration with Password Managers:** Integrating the application with popular password managers would streamline the process of storing and managing generated passwords, making it more efficient for users.

- **Multi-factor Authentication Options:** Adding support for multi-factor authentication, such as biometrics or one-time codes, would further enhance the security provided by the application.

- **Enhanced Suggestions:** The suggestions provided to users for improving password strength could be more specific and tailored to the user's input to offer better guidance.

In conclusion, the password strength analyzer and generator is a successful project that effectively addresses the need for secure passwords. The application is efficient and user-friendly, providing users with the tools they need to create and evaluate strong passwords. By addressing the areas for improvement outlined above, the application can become an even more powerful and comprehensive tool for enhancing password security.

5. Conclusion

The primary objective of this project was to design and develop a robust password strength analyzer and random password generator application, providing users with an effective tool for maintaining secure and strong passwords. Through a thorough understanding of password security concepts and the implementation of cutting-edge technologies, the project has successfully achieved its goals.

The application's user interface is intuitive and easy to navigate, making it accessible to users with varying levels of technical expertise. The password strength analyzer accurately evaluates the security of user-provided passwords, offering helpful suggestions for improvement when necessary. The random password generator allows users to create strong, unique passwords of varying lengths, ensuring a high level of customization and security.

Throughout the project, an emphasis was placed on rigorous testing and critical analysis to identify potential areas for improvement. While the application performs well in its core functionalities, there is always room for growth and enhancement. Future iterations of the application could explore the incorporation of additional security features, such as two-factor authentication or biometric authentication, to further bolster user account security. Expanding the application's breach database sources and integrating a password manager feature would also serve to improve its overall utility and user experience.

In conclusion, the password strength analyzer and random password generator application provides users with a valuable tool for safeguarding their online accounts and personal information. This final year project demonstrates the importance of password security and highlights the role of well-designed applications in promoting safer digital practices. By continuously refining and expanding upon the application's features, it can continue to serve as a reliable resource for users seeking to protect their digital assets in an increasingly connected world.

5.1 Legal, Social and Ethical Issues

In developing and implementing a password strength analyzer and random password generator, it was crucial to address various legal, social, and ethical issues associated with the project to ensure responsible use and compliance with appropriate guidelines and regulations.

5.1.1 Legal Issues

Data Privacy:

The tool deals with users' passwords, which are highly sensitive data. Ensuring data privacy is maintained throughout the entire process is critical. The system must not store, share, or transmit any user passwords without their explicit consent. Compliance with data protection regulations, such as the General Data Protection Regulation (GDPR) in Europe and the California Consumer Privacy Act (CCPA) in the United States, is essential to avoid legal repercussions. This project ensures that all necessary measures are taken to protect user data, including hashing, anonymization, and secure data handling practices.

Copyright and Licensing:

During the development process, the use of open-source libraries, frameworks, and software components were adhered to their respective licensing agreements. Proper attribution and compliance to the terms of use are necessary to avoid potential legal disputes. This project was built by researching various licenses and their requirements to ensure compliance and avoid copyright infringement.

5.1.2 Social Issues

Digital Literacy:

The effectiveness of the password strength analyzer and random password generator relies on users' understanding of the importance of secure passwords and their willingness to implement the recommendations provided by the tool. It is essential to educate users about the potential risks of weak passwords and promote the use of password management tools to encourage a more secure digital environment. This was achieved through educational materials, user guides, and practical examples that demonstrate the potential consequences of using weak passwords.

Accessibility:

Ensuring that the password strength analyzer and random password generator are accessible to users with varying levels of digital literacy and diverse backgrounds was crucial. This project was built in a way that it offered a user-friendly interface, clear instructions, and compatibility with minimalistic UI with non tech savvy individuals in mind. This project considered factors, varying technical abilities when designing the user interface and functionality of the tool to ensure that it is easy to use, inclusive and accessible to all potential users.

5.1.3 Ethical Issues

Security:

While the tool aims to enhance users' online security, it is vital to consider the potential for malicious actors to exploit the system. This application was built in a way that it does not store data entered by the users. This means that any input received in the application cannot be stolen as it is not stored in any database. This ensures the security concerns of the users and makes the tool safe to use by all kinds of users without having to worry about their data being leaked.

Trust:

The users trust the password strength analyzer and random password generator with sensitive information, and it is crucial to maintain that trust by providing accurate and reliable results. This project ensures that the machine learning algorithms and analysis techniques used in the tool are highly accurate, up-to-date, and unbiased which is essential for maintaining user trust and confidence in the system. The application was regularly reviewed and validated for performance and improvements were made as necessary to ensure that the tool remained effective and trustworthy.

In conclusion, addressing legal, social, and ethical issues was a fundamental aspect of developing and implementing a password strength analyzer and random password generator. By considering these aspects, this project was created ensuring a responsible and effective solution that adheres to appropriate regulations, fosters trust, and contributes to a more secure and accessible digital landscape. Following these practices enhanced the tool's overall efficacy which will help to ensure its continued acceptance and use by individuals and organizations.

5.2 Advantages

The password strength analyzer and random password generator offer numerous advantages to their users, organizations, and the broader digital community. These benefits contribute to a more secure and reliable online environment, which encourages higher trust in digital systems and services.

Enhancing Security:

The primary advantage of this tool is its ability to assess the strength of a user's password and recommend improvements or generate a random, secure password. By encouraging users to adopt stronger passwords, this tool plays a crucial role in enhancing overall online security and reducing the risk of unauthorized access to user accounts, data breaches, and identity theft.

Providing User-friendly Interface:

The password strength analyzer and random password generator feature an intuitive, easy-to-use interface that serves to users of all skill levels and backgrounds. This allows users to quickly and effortlessly assess their passwords' strength and make necessary changes without requiring extensive technical knowledge.

Improving Password Management:

By providing users with an accessible and easy-to-use tool for password analysis and generation, the tool promotes better password security practices. The users are more likely to create unique and secure passwords for different accounts, reducing the likelihood of password reuse and their associated risks.

Ensuring Accessibility and Inclusivity:

The tool is designed to be accessible to a diverse range of users, including those with disabilities or limited digital literacy. By ensuring compatibility with assistive technologies and providing clear instructions in multiple languages, this tool reaches a broad audience and contributes to a more inclusive digital environment.

Offering Real-time Feedback:

The password strength analyzer provides real-time feedback to users, allowing them to make immediate adjustments to their passwords and see the impact on their security. This interactive approach to password analysis helps users understand the importance of strong passwords and the factors that contribute to their security.

Enabling Scalability and Flexibility:

The design of this tool allows for easy updates and improvements, ensuring that it remains relevant and effective as new security threats and password best practices emerge. The use of machine learning algorithms and modular components facilitate the ongoing development and refinement of the tool, enabling it to adapt to changing user needs and expectations.

In summary, the password strength analyzer and random password generator offer numerous advantages, including enhanced security, user-friendly design, improved password management, accessibility, real-time feedback, and scalability. These benefits make the tool a valuable asset in promoting a safer and more inclusive digital environment.

5.3 Limitations

While the password strength analyzer and random password generator provide valuable assistance in enhancing password security, it is important to recognize their limitations. The following points discuss these limitations in detail:

Limited Scope of Analysis:

The password strength analyzer focuses on assessing passwords based on predefined criteria, such as length, complexity, and the use of special characters. However, it may not account for all potential security risks or consider all possible attack vectors. For instance, it does not identify passwords that are susceptible to social engineering attacks, dictionary attacks, or more advanced techniques that exploit human psychology instead of brute force methods. Therefore, users must remain vigilant and adopt additional security measures to protect their accounts.

Overemphasis on Password Complexity:

The tool places a strong emphasis on password complexity, which is undoubtedly a crucial aspect of password security. Nevertheless, this might lead users to overlook other important security practices, such as using multi-factor authentication, and avoiding the reuse of passwords across multiple accounts. A comprehensive approach to account security should encompass all these aspects.

Dependence on User Input:

The effectiveness of the password strength analyzer relies on the user's willingness to enter their actual password for analysis. The users might be hesitant to input their real passwords due to privacy concerns or fear of data breaches, limiting the tool's ability to accurately assess password security.

Potential False Sense of Security:

After utilizing the password strength analyzer and random password generator, users might develop a false sense of security, believing that their accounts are now impervious to attacks. This overconfidence could lead them to neglect other essential security measures, such as monitoring their accounts for suspicious activity, being cautious about sharing their passwords, and staying informed about the latest cybersecurity threats.

Compatibility with Different Systems:

Although the tool has been designed to be accessible and compatible with various devices, operating systems, and browsers, technical issues or incompatibilities might hinder its functionality in certain cases. To address this limitation, developers should continuously update the tool to ensure compatibility with new software versions and hardware specifications.

Lack of Personalized Recommendations:

The password strength analyzer provides general feedback based on predefined criteria, but it does not offer personalized recommendations that consider an individual's specific context, habits, or risk factors. To enhance its effectiveness, future iterations of the tool could incorporate machine learning algorithms that adapt to users' unique circumstances and offer tailored suggestions for improving password security.

In conclusion, the password strength analyzer and random password generator offer numerous benefits in promoting password security. However, it is crucial to recognize their limitations and understand that they should be used in conjunction with a broader approach to online security. Users should adopt additional security measures, such as multi-factor authentication and regular password updates, to ensure comprehensive protection for their online accounts.

5.4 Future Work

In order to improve and expand the password strength analyzer and random password generator, several areas of potential growth and enhancement can be identified. The following sections provide a detailed explanation of various opportunities for future work:

Personalized Recommendations:

To enhance the password strength analyzer, the development of machine learning algorithms that adapt to unique circumstances, habits, and risk factors of users is essential. By incorporating these algorithms, the tool can provide personalized recommendations for improving password security, ensuring it caters to a diverse range of users. This could involve analyzing users' password creation patterns, common keyboard combinations, and personal information usage to develop customized password strength criteria and advice.

Continuous Learning:

To maintain the effectiveness of the password strength analyzer, it is crucial to implement a system that continuously learns from user interactions and updates its password strength assessment algorithms accordingly. This would allow the tool to adapt to the latest password attack trends and techniques and stay up to date with the ever-evolving threat landscape. By leveraging machine learning, natural language processing, and data analysis techniques, the tool can dynamically adjust its evaluation criteria and provide accurate and relevant feedback to users.

Enhanced User Interface:

Improving the user interface is another area of focus for future development. The goal is to make it more engaging and interactive, while still maintaining simplicity and ease of use. This could involve implementing visual representations of password strength, such as progress bars or color-coded indicators, as well as providing helpful tips and suggestions to users. Additionally, the tool could offer an option to compare the strength of multiple passwords, allowing users to make more informed decisions about their password choices.

Collaboration with Security Experts:

Working closely with cybersecurity experts and researchers can ensure that the tool's algorithms and recommendations are in line with the latest advancements in the field. By collaborating with experts, the tool can be regularly updated based on new research findings, attack vectors, and security best practices. This partnership will help maintain the tool's effectiveness in evaluating password security and providing accurate, actionable advice to users.

User Education:

Incorporating educational content into the tool can help raise awareness about password security and best practices. This could include tutorials, infographics, and interactive quizzes designed to help users understand the importance of strong passwords and learn how to create them effectively. By providing educational resources, the tool can empower users to take control of their password security and make informed decisions about their account protection.

The various other improvements which can me made are further elaborated in the appendix section. **(Future Work: Appendix E)**

6. References

Abbott, J., Calarco, D. & Camp, L., 2018. *Factors Influencing Password Reuse: A Case Study.* s.l.:Indiana University.

Adobe Communication , 2022. *Adobe Experience Cloud Blog.* [Online]
Available at: https://business.adobe.com/blog/basics/waterfall
[Accessed 14 October 2022].

Balsamiq, 2023. *What Are Wireframes?.* [Online]
Available at: https://balsamiq.com/learn/articles/what-are-wireframes/
[Accessed 28 Jan 2023].

Chang, J., 2022. *55 Important Password Statistics You Should Know: 2022 Breaches & Reuse Data.* [Online]
Available at: https://financesonline.com/password-statistics/

Chaudhuri, A. B., 2020. *Flowchart and Algorithm Basics The Art of Programming.* Virginia: MERCURY LEARNING AND INFORMATION.

Das, A. et al., 2014. *The Tangled Web of Password Reuse.* San Diego: NDSS.

Escardó, M. H., 2019. *Data Structures and Algorithms.* Birmingham: University of Birmingham.

Farooq, U., 2020. Real Time Password Strength Analysis on a Web Application Using Multiple Machine Learning Approaches. *International Journal of Engineering Research & Technology (IJERT),* 9(12), pp. 359-364.

Gaba, I., 2023. *What is GitHub And How To Use It?.* [Online]
Available at: https://www.simplilearn.com/tutorials/git-tutorial/what-is-github
[Accessed 16 Feb 2023].

Gantt.com, 2022. *What is a Gantt Chart? Gantt Chart Software, Information and History.* [Online]
Available at: https://www.gantt.com/

G, S., S, K. & V, C., 2010. Proactive Password Strength Analyzer Using Filters and Machine Learning Techniques. *International Journal of Computer Applications,* 7(14), pp. 1-5.

Hunt, T., 2022. *Have I Been Pwned: Pwned Passwords.* [Online]
Available at: https://haveibeenpwned.com/Passwords
[Accessed 27 December 2022].
Ives, B., Walsh, K. R. & Schneider, H., 2004. Communications of the ACM. *The domino effect of password reuse,* pp. 75-78.
Lucid Content, 2021. *Lucidchart.* [Online]
Available at: https://www.lucidchart.com/blog/waterfall-project-management-methodology
[Accessed 14 October 2022].
Martin, J., 2022. *coolblueweb.* [Online]
Available at: https://coolblueweb.com/blog/guide-to-agile-methodology/
[Accessed 14 October 2022].
Martin, M., 2022. *Guru99.* [Online]
Available at: https://www.guru99.com/software-engineering-prototyping-model.html
[Accessed 14 October 2022].
Mhadhbi, N., 2021. *Python Tutorial: Streamlit.* [Online]
Available at: https://www.datacamp.com/tutorial/streamlit
Mills, R., 2021. *Blackbaud data breach: What you need to know.* [Online]
Available at: https://givebutter.com/blog/blackbaud-data-breach
[Accessed 23 Novermber 2022].
Panigrahi, K. K., 2023. *Difference between Unit Testing and System Testing,* Hyderabad: Tutorials Point India Private Limited.
Python Software Foundation, 2022. *What is Python? Executive Summary.* [Online]
Available at: https://www.python.org/doc/essays/blurb/
Rosenthal, M., 2022. *tessian.* [Online]
Available at: https://www.tessian.com/blog/phishing-statistics-2020/
[Accessed 17 December 2022].
Sarkar, S. & Nandan, M., 2022. *Building a Multi-class Password Strength Generator and Classifier Model by Augmenting Supervised Machine Learning Techniques,* Durham: Research Square.

Tuleap, 2022. *Understanding Agile Scrum in 10 minutes.* [Online]
Available at: https://www.tuleap.org/agile/agile-scrum-in-10-minutes
Visual Paradigm, 2022. *Visual Paradigm.* [Online]
Available at: https://www.visual-paradigm.com/guide/project-management/what-is-work-
[Accessed 15 October 2022].
Visual Paradigm, 2023. *What is Use Case Diagram?.* [Online]
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
[Accessed 12 Jan 2023].
Visual Studio Code, 2022. *Why Visual Studio Code ?.* [Online]
Available at: https://code.visualstudio.com/docs/editor/whyvscode
Wilson, J., 2022. *Techopedia.* [Online]
Available at: https://www.techopedia.com/definition/3533/python
[Accessed 17 December 2022].

7. Appendices

7.1 Appendix A: Sample codes

```python
import streamlit as st
from password_analyzer import check_password_strength, load_vectorizer, load_model
from api_check import check_password_api
from nist_compliance import display_password_compliance
from password_generator import generate_password
```

Figure 39: Importing all the project modules

```python
# Convert a given word (string) to a list of individual characters
def word_to_char(word):
    return list(word)

def main():
    # Define the available activities in the application
    activities = ["Password Analyzer", "Password Generator"]

    # Create a sidebar selection box for the user to choose an activity
    choice = st.sidebar.selectbox("Select Activity", activities)

    # If the user chooses "Password Analyzer" activity
    if choice == "Password Analyzer":
        st.title("Password Strength Analyzer")
        st.subheader("Test Your Password Strength")

        # Get the user's input for the password
        password = st.text_input("Enter Password", type="password")
        st.write("\n\n")
```

Figure 40: Code for sidebar and password input field

```
        # When the user clicks the "Check" button
        if st.button("Check"):
            # Load the vectorizer and model
            vectorizer = load_vectorizer()
            model = load_model()

            # Check and display the password strength
            strength = check_password_strength(password, vectorizer, model)
            if strength == 0:
                st.error("**Your Password is Weak.** \n\n **A strong password consists of a combination of letters, numb
            elif strength == 1:
                st.warning("**Your Password is Average.** \n\n **A strong password consists of a combination of letters,
            elif strength == 2:
                st.success("**Your Password is Strong.**")

            # Check the password against the API and display the result
            compromised_count = check_password_api(password)
            if compromised_count:
                st.error(f"**Your password has been compromised {compromised_count} times before.**")
            else:
                st.success("**Your password was not seen in compromised password database of Have I Been Pwned.**")

        # Display the password compliance guidelines
        guidelines = display_password_compliance()
        st.subheader("NIST Password Guidelines:")
        for i, guideline in enumerate(guidelines, start=1):
            st.write(f"{i}. {guideline}")
```

Figure 41: Code for checking password strength, breach status and NIST guidelines

```
    # If the user chooses "Password Generator" activity
    elif choice == "Password Generator":
        st.subheader("Generate Random Password")

        # Get the user's input for the password length
        password_length = st.number_input("Enter the length of password.", 10, 64)
        st.write("Password Length: ", password_length)

        # When the user clicks the "Generate" button
        if st.button("Generate"):
            # Generate a random password based on the user's input
            custom_password = generate_password(password_length)

            # Display the generated password
            st.write(custom_password)

# Execute the main function
if __name__ == '__main__':
    main()
```

Figure 42: Code for password generator field and length limit

password_analyzer.py > ...

```python
import numpy as np
import pickle

# Load the pre-trained vectorizer from a pickle file
def load_vectorizer():
    with open("tfidf_password_strength.pickle", 'rb') as file:
        saved_vectorizer = pickle.load(file)
    return saved_vectorizer

# Load the pre-trained model from a pickle file
def load_model():
    with open("final_model.pickle", 'rb') as file:
        final_model = pickle.load(file)
    return final_model

# Check and display the strength of the input password
def check_password_strength(password_input, vectorizer, model):
    X_password = np.array([password_input])
    # Transform the input password using the pre-trained vectorizer
    X_predict = vectorizer.transform(X_password)
    # Make predictions using the pre-trained model
    y_pred = model.predict(X_predict)
    return y_pred[0]
```

Figure 43: Code for password analyzer module

password_generator.py > ...

```python
import random
import string

# Generate a random password of given length
def generate_password(length):
    # Combine digits, letters, and punctuation as possible characters for the password
    characters = string.digits + string.ascii_letters + string.punctuation
    # Randomly select characters and join them to form the password
    generated_password = "".join(random.choice(characters) for _ in range(length))
    return generated_password
```

Figure 44: Code for password generator module

api_check.py > ...

```python
import requests
import hashlib

API_URL = 'https://api.pwnedpasswords.com/range/'

# Request data from the API
def request_api_data(query_char):
    url = API_URL + query_char
    response = requests.get(url)

    # Raise an error if the API request is unsuccessful
    if response.status_code != 200:
        raise RuntimeError(f'ERROR FETCHING: {response.status_code}, CHECK API AND TRY AGAIN')
    return response

# Get the count of compromised passwords
def get_compromised_count(hashes, tail_hash):
    # Split the response text into lines and then into hash and count pairs
    hashes = (line.split(':') for line in hashes.text.splitlines())
    for hash, count in hashes:
        if hash == tail_hash:
            return count
    return 0

# Check if the password has been compromised using the API
def check_password_api(password):
    # Calculate the SHA-1 hash of the password
    sha1_password = hashlib.sha1(password.encode('utf-8')).hexdigest().upper()
    first5_chars, tail = sha1_password[:5], sha1_password[5:]
    response = request_api_data(first5_chars)
    return get_compromised_count(response, tail)
```

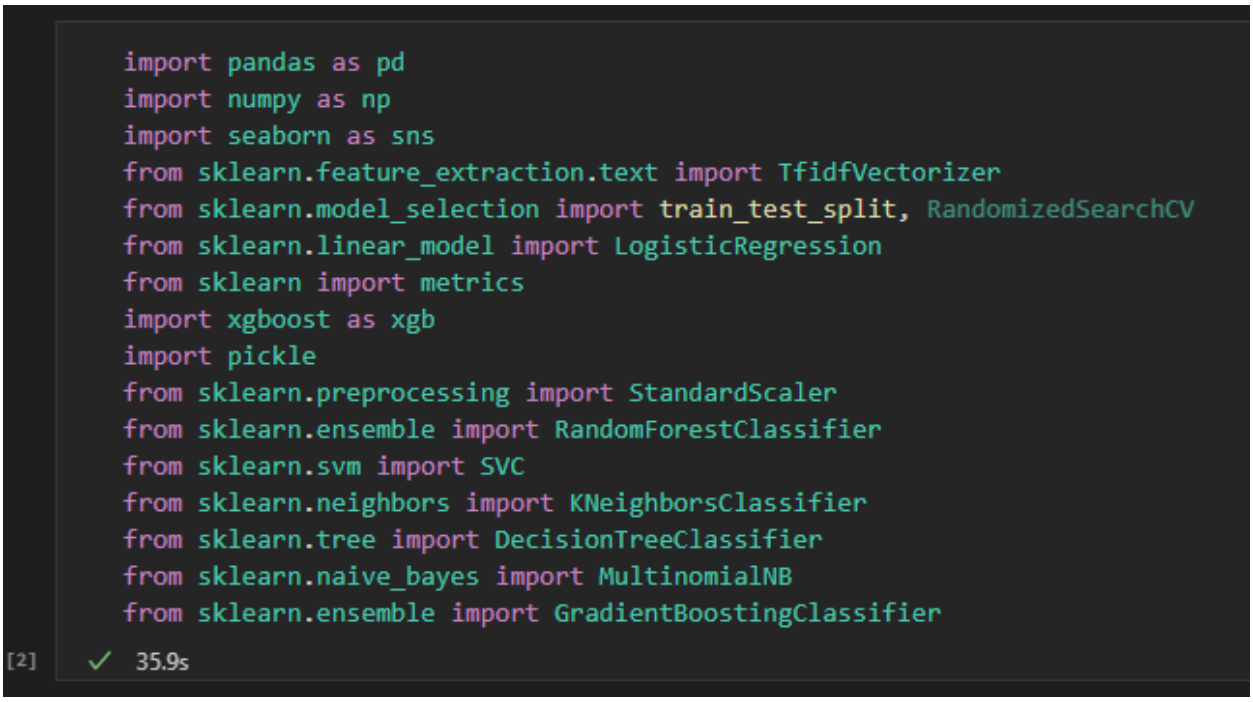
Figure 45: Code for API check module

nist_compliance.py > display_password_compliance

```python
def display_password_compliance():
    nist_guidelines = [
        "Passwords should be at least 8 characters in length.",
        "Include a combination of upper and lower case letters, numbers, and special characters.",
        "Avoid dictionary words, common phrases, and easily guessable information.",
        "Avoid using easily guessable personal information, such as birthdays or names of family members.",
        "Use a unique password for each account.",
        "Do not write down or share passwords with others.",
        "Enable multi-factor authentication (MFA) when available.",
        "Do not use passwords that have been compromised in data breaches."
    ]

    return nist_guidelines
```

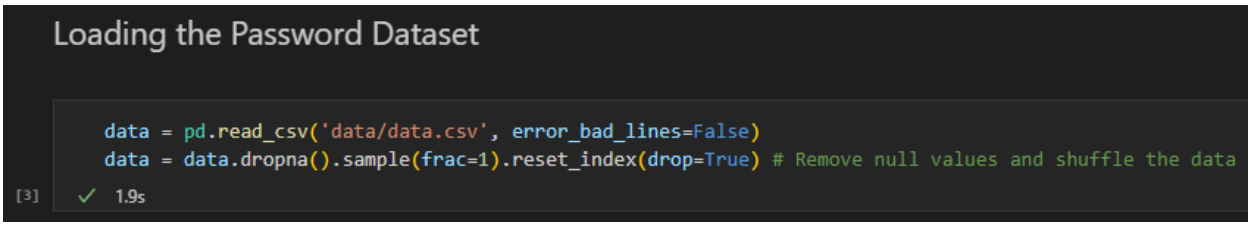
Figure 46: Code for NIST compliance module



```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import xgboost as xgb
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import GradientBoostingClassifier
```

[2] ✓ 35.9s

Figure 47: Importing all the libraries required for machine learning

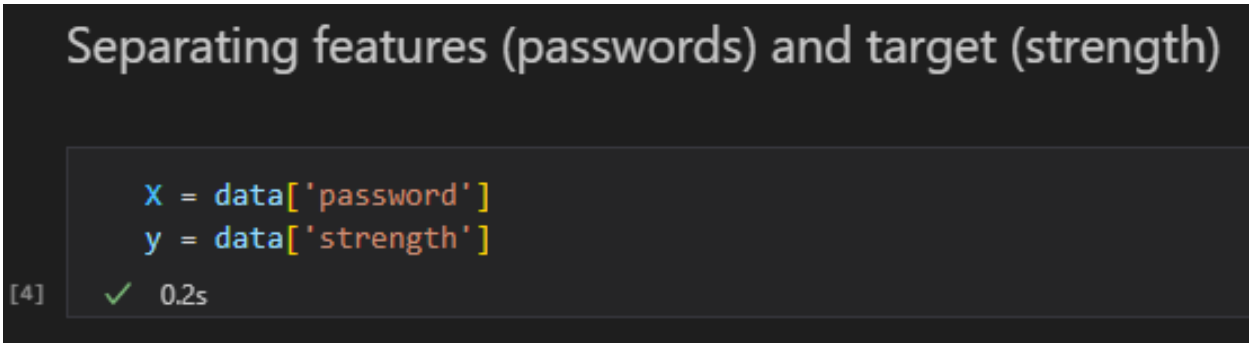


Loading the Password Dataset

```
data = pd.read_csv('data/data.csv', error_bad_lines=False)
data = data.dropna().sample(frac=1).reset_index(drop=True) # Remove null values and shuffle the data
```

[3] ✓ 1.9s

Figure 48: Loading the password dataset



Separating features (passwords) and target (strength)

```
X = data['password']
y = data['strength']
```

[4] ✓ 0.2s

Figure 49: Separating columns for passwords and their strengths

Plot the distribution of password strength

```
[10]  sns.countplot(x='strength', data=data)
      ✓ 0.4s
...   <AxesSubplot:xlabel='strength', ylabel='count'>
```

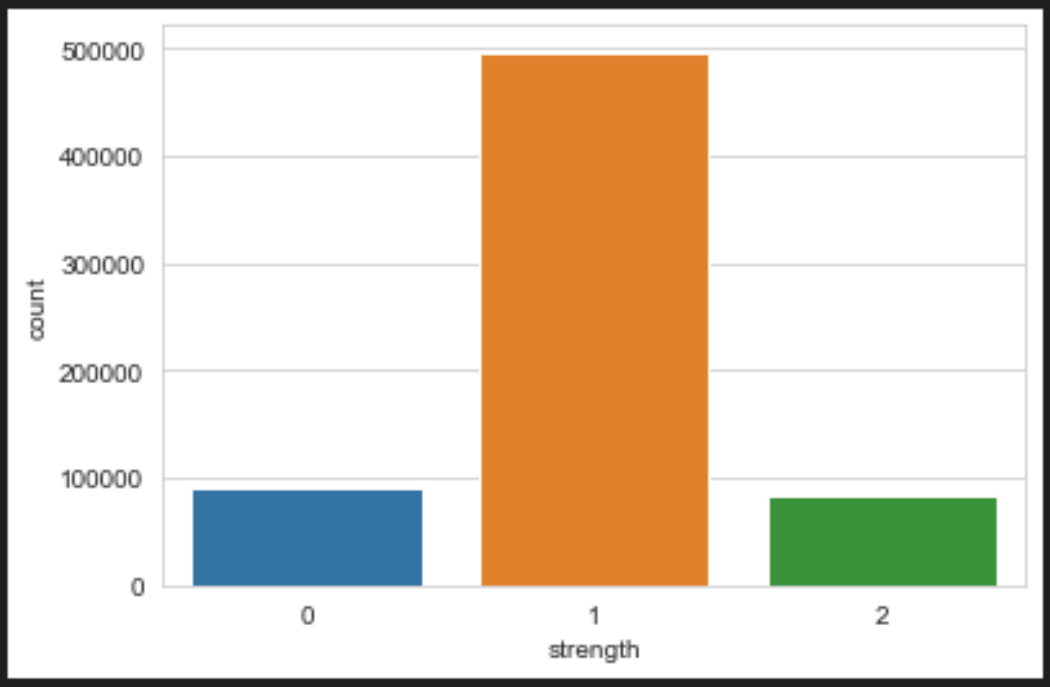


Figure 50: Plotting the distribution of password strength

Convert a given word (string) to a list of individual characters

```
[11]  def word_to_char(word):
          return list(word)
      ✓ 0.0s
```

Figure 51: Code for converting words to list of individual characters

Create the TfidfVectorizer and fit_transform the passwords

```
vectorizer = TfidfVectorizer(tokenizer=word_to_char)
X = vectorizer.fit_transform(X)
```

[12] ✓ 8.8s

Figure 52: Code for creating vectorizer

Split the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

[13] ✓ 0.7s

Figure 53: Code for splitting data into train and test set

Scaling the data

```
scaler = StandardScaler(with_mean=False)
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[14] ✓ 0.5s

Figure 54: Code for scaling the data

Train and evaluate a logistic regression model

```
[13]
log_clf = LogisticRegression(penalty='l2', multi_class='ovr', solver='liblinear')
log_clf.fit(X_train, y_train)
y_pred = log_clf.predict(X_test)
print('Accuracy (Logistic Regression):', metrics.accuracy_score(y_test, y_pred))
```

```
... Accuracy (Logistic Regression): 0.8112941281882803
```

Figure 55: Code for training and evaluating logistic regression model

Train and evaluate a K-Nearest Neighbors model

```
[10]
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)
y_pred = knn_clf.predict(X_test)
print('Accuracy (K-Nearest Neighbors):', metrics.accuracy_score(y_test, y_pred))
```

```
... Accuracy (K-Nearest Neighbors): 0.7786721223343886
```

Figure 56: Code for training and evaluating K-Nearest Neighbors model

Train and evaluate a Support Vector Machine model

```
[11]
svc_clf = SVC()
svc_clf.fit(X_train, y_train)
y_pred = svc_clf.predict(X_test)
print('Accuracy (Support Vector Machine):', metrics.accuracy_score(y_test, y_pred))
```

```
... Accuracy (Support Vector Machine): 0.8886939250940804
```

Figure 57: Code for training and evaluating Support Vector Machine model

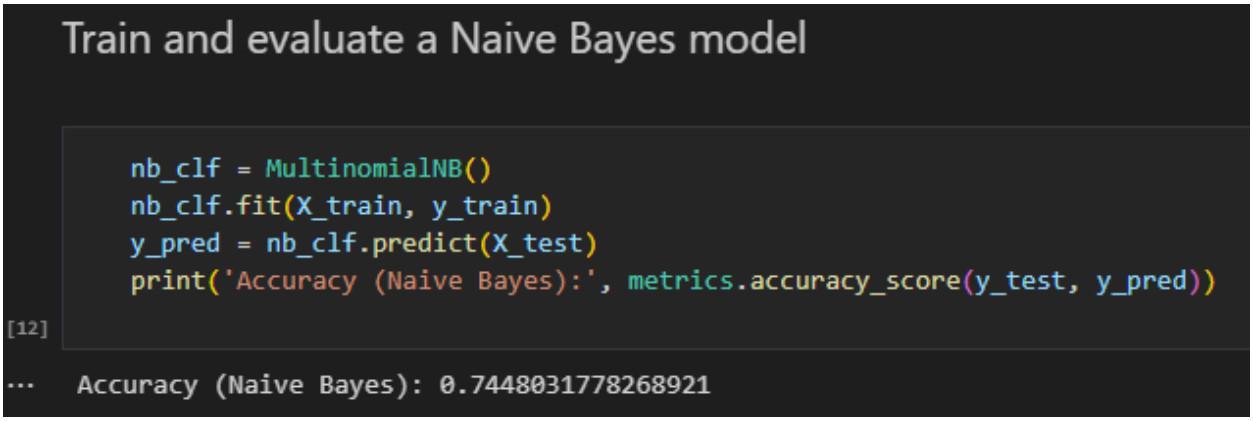


Figure 58: Code for training and evaluating Naive Bayes model

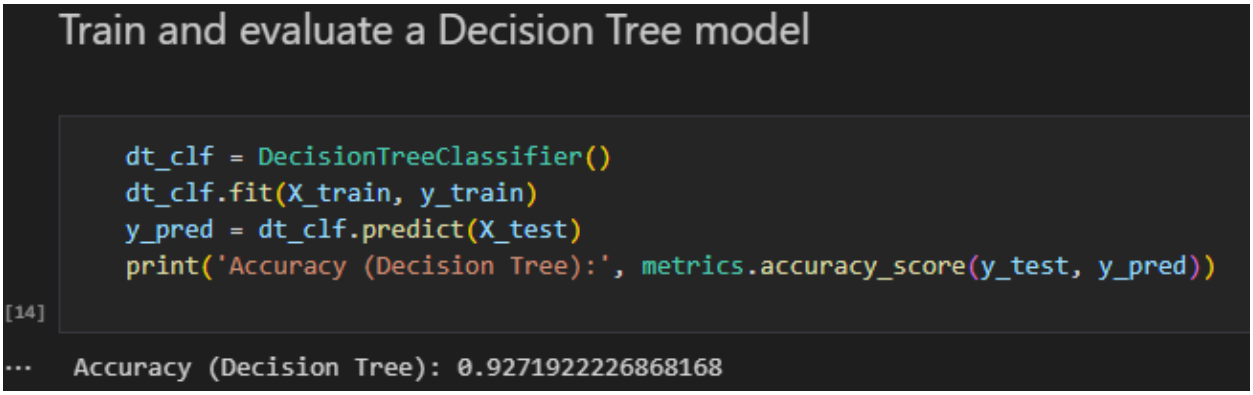


Figure 59: Code for training and evaluating Decision Tree model

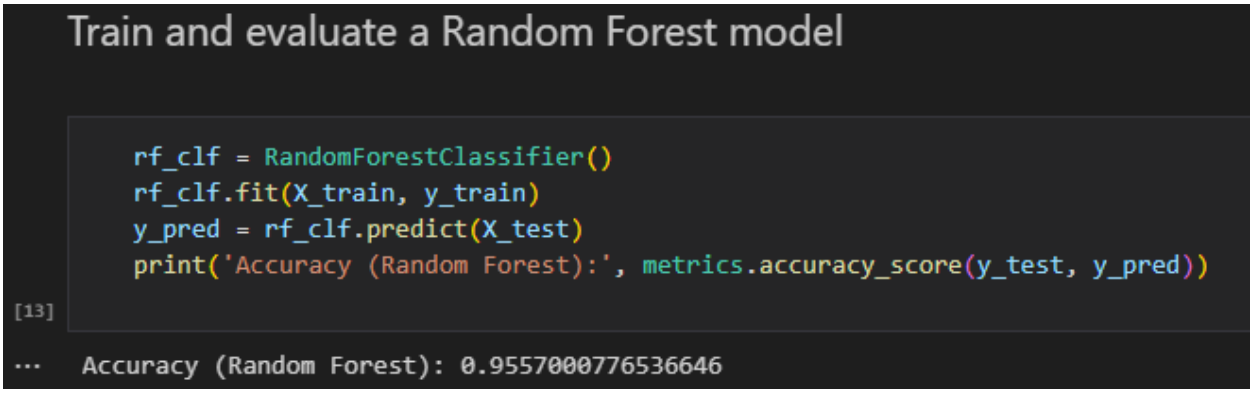


Figure 60: Code for training and evaluating Random Forest model

Train and evaluate a Gradient Boosting Classifier model

```
gb_clf = GradientBoostingClassifier()
gb_clf.fit(X_train, y_train)
y_pred = gb_clf.predict(X_test)
print('Accuracy (Gradient Boosting Classifier):', metrics.accuracy_score(y_test, y_pred))
```

[14]

```
Accuracy (Gradient Boosting Classifier): 0.9127516277402784
```

Figure 61: Code for training and evaluating Gradient Boosting Classifier model

Train and evaluate an XGBoost classifier

```
xgb_clf = xgb.XGBClassifier()
xgb_clf.fit(X_train, y_train)
y_pred = xgb_clf.predict(X_test)
print('Accuracy (XGBoost):', metrics.accuracy_score(y_test, y_pred))
```

[15]

```
Accuracy (XGBoost): 0.9810271190490413
```

Figure 62: Code for training and evaluating XGBoost Classifier model

Saving the vectorizer and XGBoost classifier as pickle files

```
pickle.dump(vectorizer, open("tfidf_password_strength.pickle", "wb"))
pickle.dump(xgb_clf, open("final_model.pickle", "wb"))
```

[18] ✓ 0.2s

Figure 63: Code for saving the vectorizer and XGBoost classifier

Load the saved vectorizer and XGBoost classifier

```
with open("tfidf_password_strength.pickle", 'rb') as file:
    saved_vectorizer = pickle.load(file) # Load the vectorizer from the pickle file

with open("final_model.pickle", 'rb') as file:
    final_model = pickle.load(file) # Load the final model (XGBoost classifier) from the pickle file
```

[19] ✓ 0.3s

Figure 64: Code for loading the saved vectorizer and XGBoost classifier

Defining a function to test the final model

```
def test_password_strength(password, vectorizer, model):
    X_password = np.array([password]) # Convert the password to a numpy array
    X_predict = vectorizer.transform(X_password) # Transform the password using the loaded vectorizer
    y_pred = model.predict(X_predict) # Predict the password strength using the loaded model
    return y_pred
```

[20] ✓ 0.0s

Figure 65: Code for defining a function to test the final model

Testing sample passwords

```
# Print the first password and its predicted strength
password1 = 'abc'
strength1 = test_password_strength(password1, saved_vectorizer, final_model)
print(f'Password: {password1}, Strength: {strength1}')
```

[19]

```
Password: abc, Strength: [0]
```

```
# Print the second password and its predicted strength
password2 = 'abc@123'
strength2 = test_password_strength(password2, saved_vectorizer, final_model)
print(f'Password: {password2}, Strength: {strength2}')
```

[20]

```
Password: abc@123, Strength: [1]
```

```
# Print the third password and its predicted strength
password3 = 'abc@123$##'
strength3 = test_password_strength(password3, saved_vectorizer, final_model)
print(f'Password: {password3}, Strength: {strength3}')
```

[21]

```
Password: abc@123$##, Strength: [2]
```

Figure 66: Code for testing password strength output

7.2 Appendix B: Implementation

The project was developed using various software tools and modules. To understand the how the solution works, the technologies, libraries, and platform used during the development process must be understood.

7.2.1 Technologies and Libraries

The Password Strength Analyzer application was developed using the following technologies and libraries:

- **Python:** A versatile and widely used programming language which served as the foundation for the app's development.

- **Streamlit:** A python library used to create the web app interface, facilitating the design and implementation of interactive web applications.

- **Random:** The random module was used to generate random passwords in the password generator component of the web application.

- **String:** The string module was used to access predefined character sets (letters, digits, and punctuation) when creating random passwords in the password generator component.

- **Hashlib:** The hashlib module was used to generate hashed representations of user-input passwords when making API calls to the Have I Been Pwned service, ensuring secure and anonymous communication.

- **Requests:** A python library used to handle API calls to the Have I Been Pwned service, allowing the app to check if a password has been compromised.

- **NumPy:** NumPy was used for handling arrays and performing numerical operations during the data pre-processing and model training phases of the password strength classifier.

- **Pandas:** The Pandas library was used for loading and manipulating the dataset during the data preparation phase. It facilitated the process of filtering, transforming, and cleaning the data before it was used for training and evaluating the password strength classifier.

- **Scikit-learn:** Scikit-learn was used for various tasks throughout the project, including data pre-processing (such as TF-IDF vectorizer) and model evaluation (using metrics like accuracy and confusion matrix). It played an essential role in building and evaluating the password strength classifier.

- **XGBoost:** The XGBoost library was used for training the password strength classifier. It provided an efficient and effective gradient boosting framework for the project, which resulted in a high-performing classifier.

- **Pickle:** The pickle library was used to save and load trained models and vectorizers, enabling the app to utilize the pre-trained classifier. This allowed for easy reuse of the trained models in the web app without the need for retraining.

7.2.2 Implementation Process

The implementation process for the application involved the following steps:

- **Data preparation:** The dataset used to train and evaluate the password strength classifier was loaded and pre-processed.

- **Model training and evaluation:** The XGBoost classifier was trained using the pre-processed data, and its performance was evaluated on a test set.

- **Saving and loading models:** The Pickle library was used to save and load the trained classifier and TF-IDF vectorizer.

- **API integration:** API calls to the Have I Been Pwned service were implemented using the Requests library.

- **Deployment:** The app was deployed using Streamlit Community Cloud for public access and usage.

By following these steps, the Password Strength Analyzer application was successfully implemented, providing users with an interactive and easy-to-use tool for checking password strength and generating secure passwords.

In conclusion, this comprehensive solution effectively addresses the various aspects of password management. The users can quickly identify flaws in their current passwords and effortlessly generate strong, unique replacements. This, in turn, reduces the risk of unauthorized access to sensitive information and contributes to a more secure online experience for all users The integration of these tools not only heightens individual users' security but also encourages a more secure digital environment for organizations, educational institutions, and government entities.

(Continue to previous topic: Go Back)

7.3 Appendix C: Designs

7.3.1 Gantt Chart

A Gantt chart is a graphical representation of a project's timeline, showing the start and end dates of each task, as well as the duration of each task. It is presented in a horizontal format, allowing users to easily see which tasks have been completed and which are still scheduled to be completed. The chart also indicates the length of time required for each task to be completed. It is a useful tool for visualizing the progress of a project and for managing resources and schedules (Gantt.com, 2022).

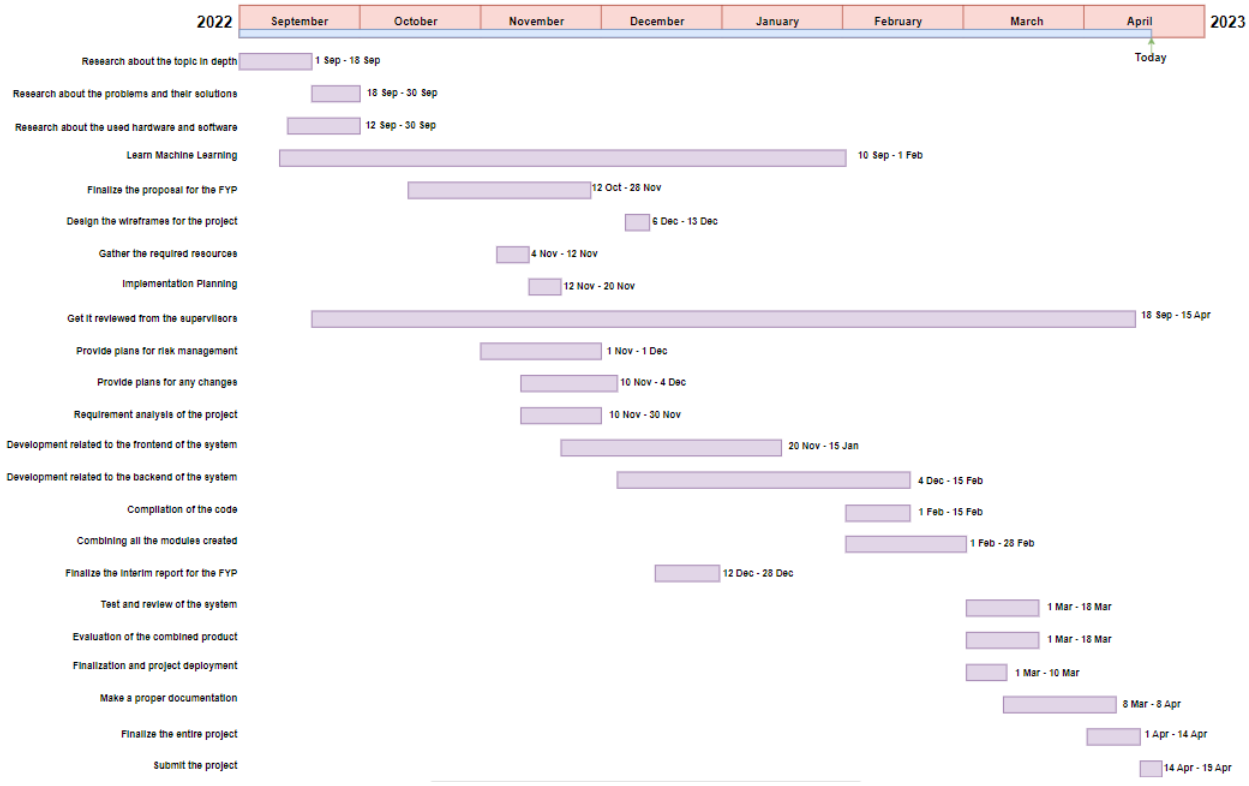


Figure 67: Gantt Chart

7.3.2 Work Breakdown Structure

A Work Breakdown Structure (WBS) is a hierarchical breakdown of the work that needs to be done in order to complete a project and create its required deliverables. It helps establish a common understanding of the project scope and is used to decompose the work to be executed by the project team. The WBS is a detailed description of all the tasks and subtasks involved in completing the project. It is used to organize and structure the work, and is often presented in the form of a visual diagram or outline (Visual Paradigm, 2022).

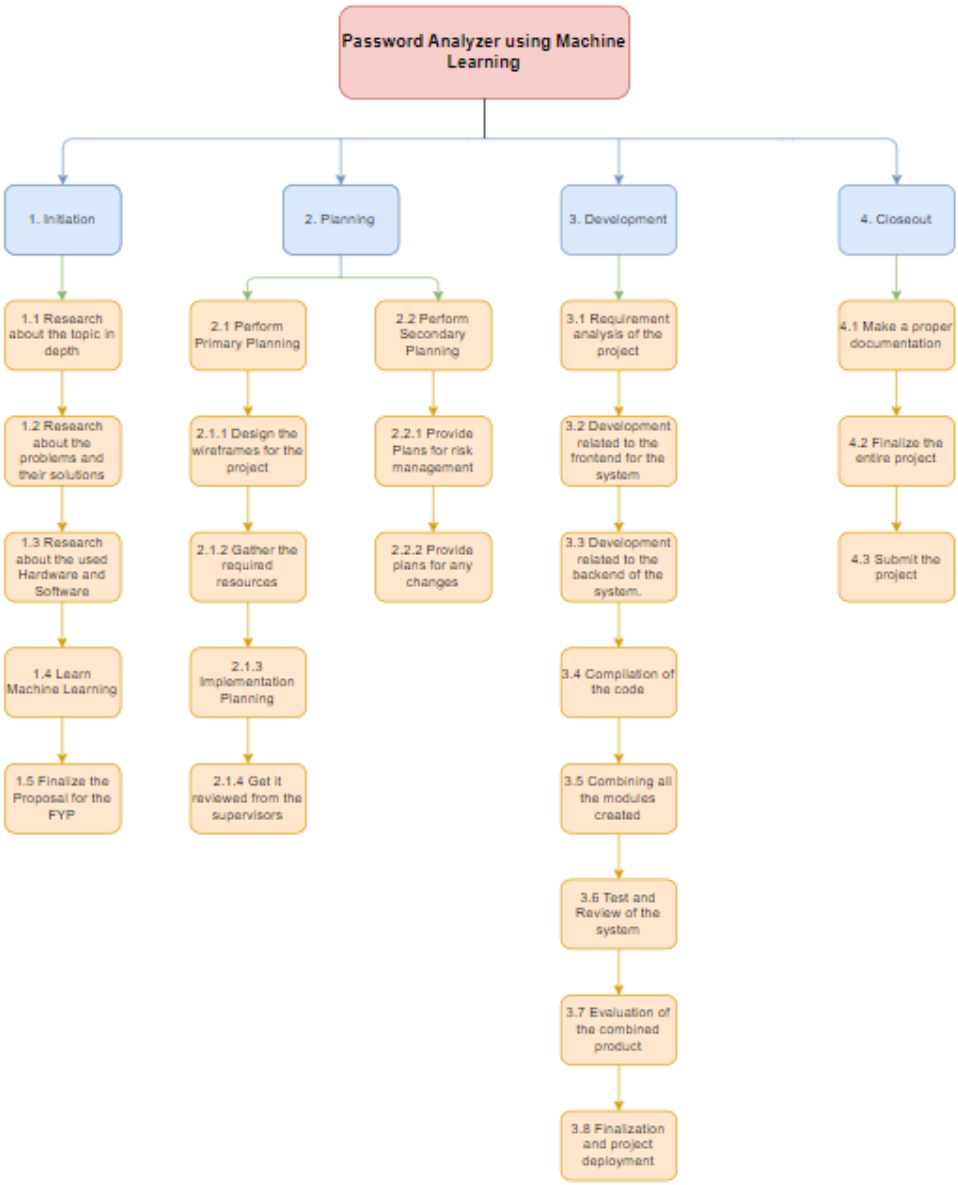


Figure 68: Work Breakdown Structure

The work breakdown structure was divided and organized based on the timeline shown in the Gantt chart. The project had several milestones that were identified and achieved at various points during the project. The project took approximately 7 months to complete, with the timeline starting on the date the topic was selected and ending with the submission of the project.

Tasks	Start Date	End Date	Duration
Research about the topic in depth	01/09/2022	18/09/2022	18 Days
Research about the problems and their solutions	18/09/2022	30/09/2022	13 Days
Research about the used hardware and software	12/09/2022	30/09/2022	19 Days
Learn Machine Learning	10/09/2022	01/02/2023	143 Days
Finalize the proposal for the FYP	12/10/2022	28/11/2022	48 Days
Design the wireframes for the project	06/12/2022	13/12/2022	7 Days
Gather the required resources	04/11/2022	12/11/2022	9 Days
Implementation Planning	12/11/2022	20/11/2022	9 Days
Get it reviewed from the supervisors	18/09/2022	15/04/2023	209 Days
Provide plans for risk management	01/11/2022	01/12/2022	31 Days
Provide plans for any changes	10/11/2022	04/12/2022	25 Days
Requirement analysis of the project	10/11/2022	30/11/2022	21 Days
Development related to the frontend of the system	20/11/2022	15/01/2023	56 Days
Development related to the backend of the system	04/11/2022	15/02/2023	103 Days
Compilation of the code	01/02/2023	15/02/2023	15 Days
Combining all the modules created	01/02/2023	28/02/2023	28 Days
Finalize the interim report for the FYP	12/12/2022	28/12/2022	16 Days
Test and review of the system	01/04/2023	18/04/2023	18 Days
Evaluation of the combined product	01/04/2023	18/04/2023	18 Days
Finalization and project deployment	01/04/2023	10/04/2023	10 Days
Make a proper documentation	08/03/2023	08/04/2023	32 Days
Finalize the entire project	01/04/2023	14/04/2023	14 Days
Submit the project	14/04/2023	19/04/2023	6 Days

Table 20: Tabular form of dates in Gantt Chart

7.3.3 Algorithms & Flowcharts

An algorithm can be described as a structured set of instructions designed to solve a problem or accomplish a specific task. Similar to a recipe, it provides a step-by-step procedure to follow, ensuring that the desired outcome is reached. Algorithms serve as the foundation for computer programs, guiding the process of problem-solving and decision-making (Escardó, 2019).

A flowchart, on the other hand, is a visual representation of an algorithm or process. It uses various symbols and shapes to illustrate the sequence of steps, decisions, and actions required to complete a task or solve a problem. Flowcharts help simplify complex processes, making it easier for individuals to understand the logic and flow of information. They are particularly useful in designing and analyzing computer programs, as they provide a clear and concise way to present the structure and organization of an algorithm (Chaudhuri, 2020).

Process flow of Machine Learning Model Selection:

1. The dataset was opened and loaded in Jupyter Notebook.
2. The loaded dataset was pre-processed and split into train and test data.
3. The data was trained and tested using different ML algorithms.
4. The model with the highest accuracy was saved to use in the project.

The flow of this process is illustrated in the flowchart below:

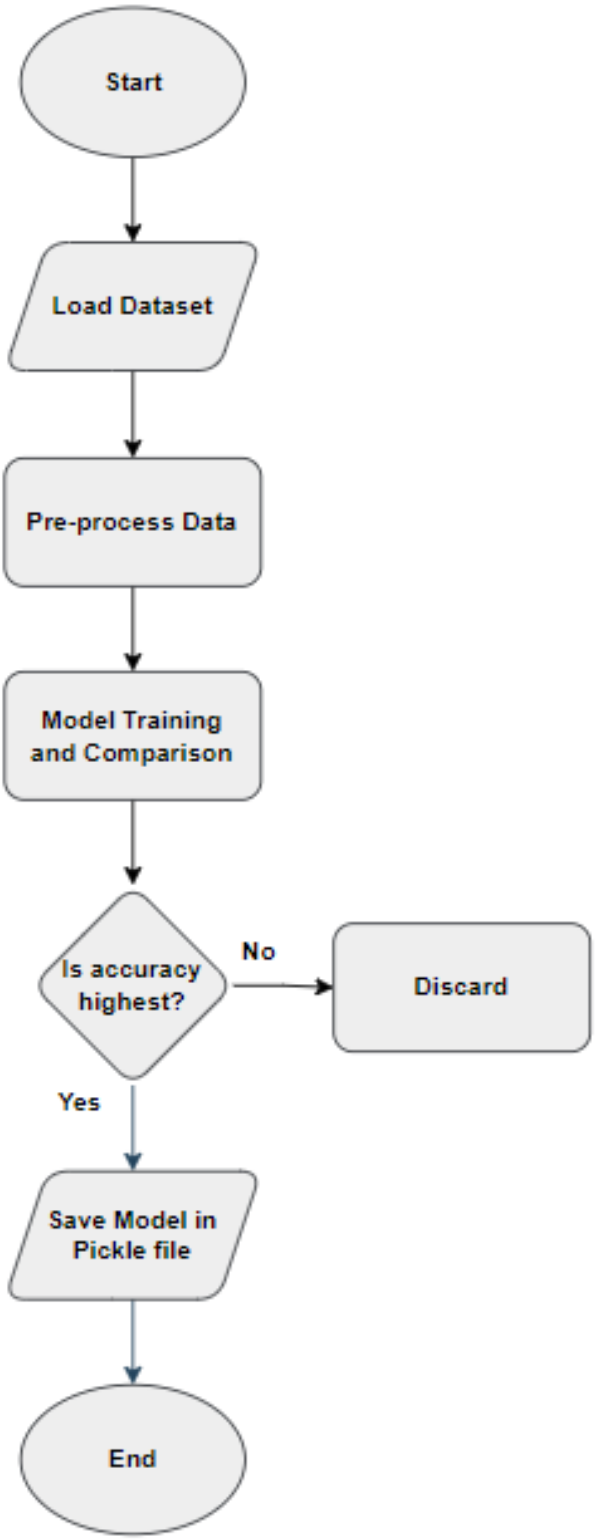


Figure 69: Flowchart of machine learning model

In this project, when a user accesses the application, they have the option to use either the password analyzer or password generator.

Process flow of password analyzer:

1. The user enters a password.
2. The user clicks on the Check button.
3. The system classifies strength of the password as weak, average or strong from the pre-trained ML model.
4. The password strength is displayed with suggestions for improvement.
5. The system calls the Pwned Passwords API form Have I Benn Pwned with the hash from the entered password.
6. The result of breach status is displayed.
7. The user may choose to repeat this process as many times as they want.

The flow of this process is illustrated in the flowchart below:

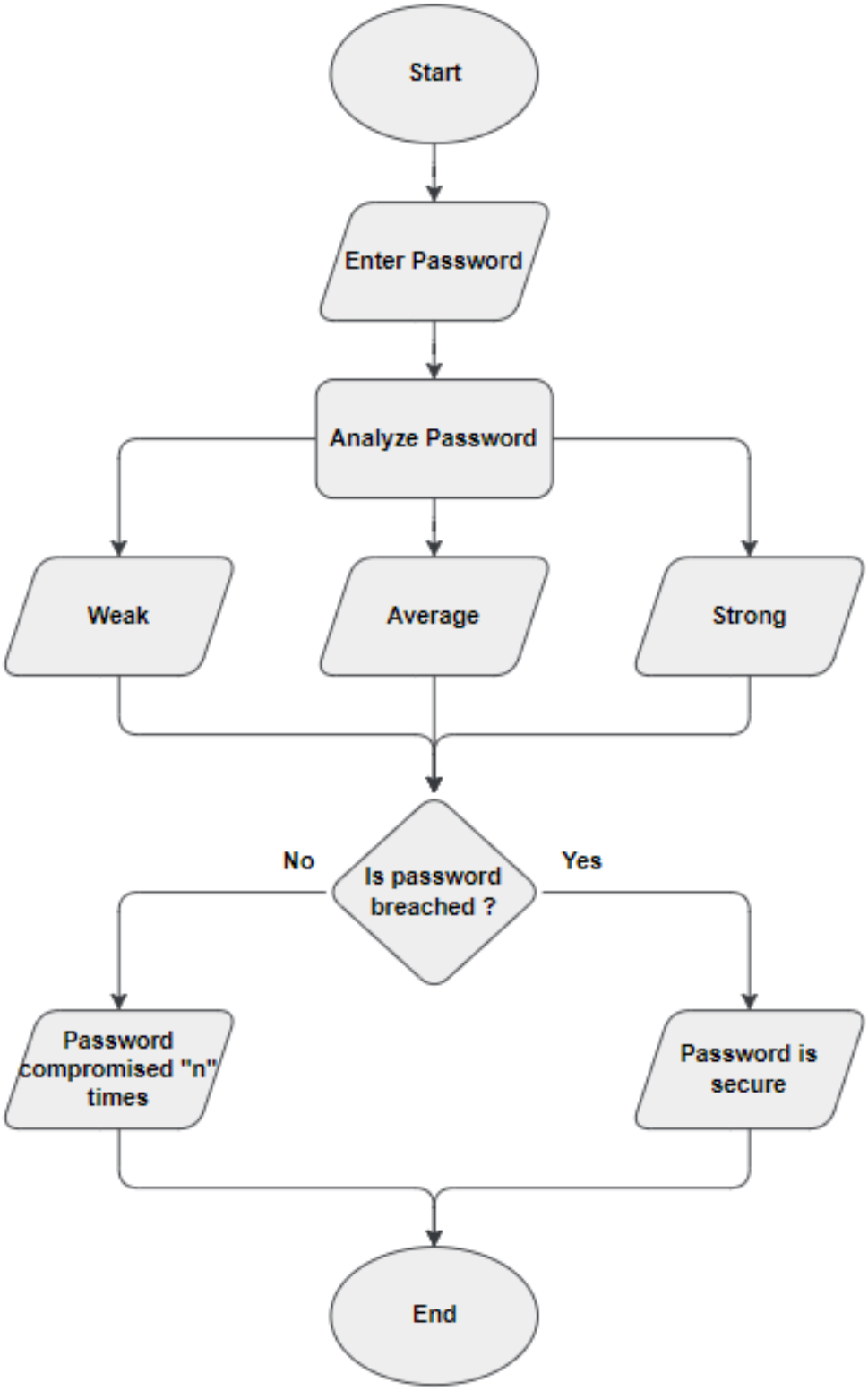


Figure 70: Flowchart for Password Strength Analyzer

Process flow of password analyzer:

1. The user enters a desired password length.
2. The user clicks on the Generate button.
3. If the length entered is not between 10-64, an error message is displayed.
4. If the length entered is between 10-64, the password is generated.
5. The user may choose to repeat this process as many times as they want.

The flow of this process is illustrated in the flowchart below:

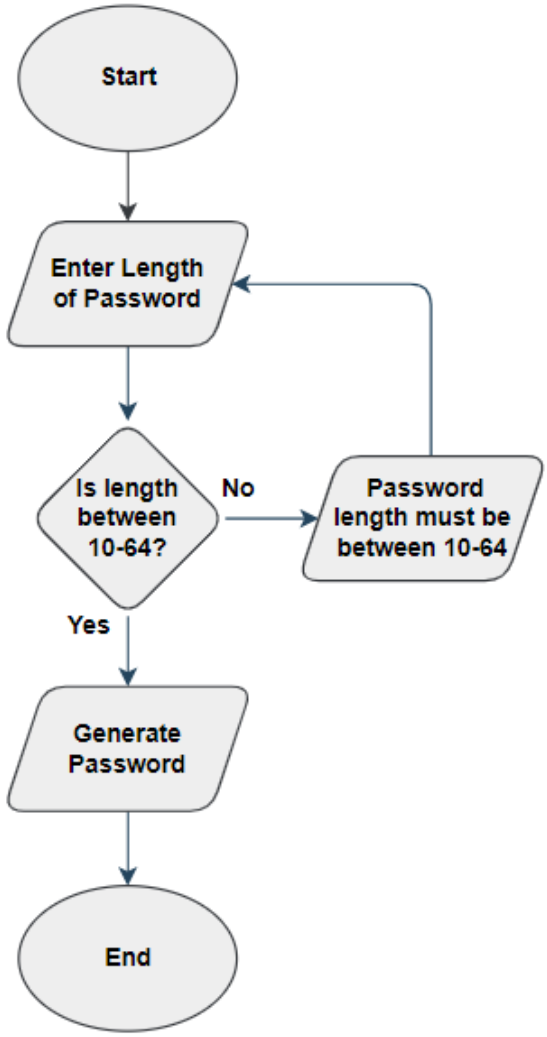


Figure 71: Flowchart for Random Password Generator

(Continue to previous topic: Go Back)

7.3.4 Use Case Diagrams

The use case diagram below depicts the functions that a user can perform using the application. The user interacts with two main components of the application: Password Analyzer and Password Generator. The goal is to evaluate the strength of the user's existing password and potentially generate a new, more secure password based on their input.

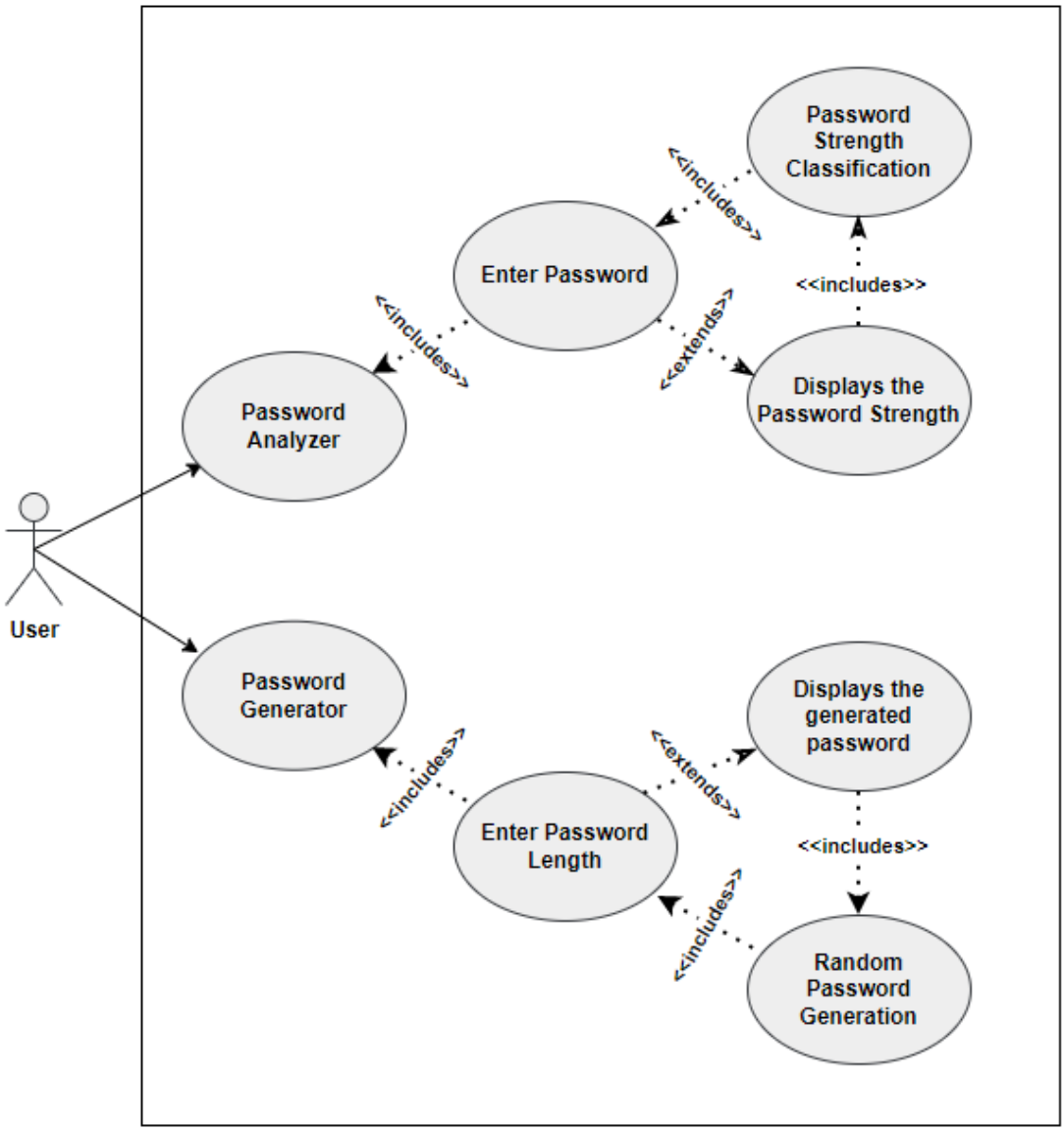


Figure 72: Use Case Diagram of application

Password Analyzer:

1. The user begins by accessing the Password Analyzer feature of the application.
2. They input their current password into the designated input field.
3. The system processes the input and calculates the strength of the password based on predefined criteria such as length, complexity, and the use of special characters.
4. The application displays the calculated password strength to the user, including feedback and recommendations on how to improve their password security.
5. The user may choose to update their password based on the recommendations provided or input a new password to analyze its strength. This step can be repeated as many times as needed until the user is satisfied with their password strength.

Password Generator:

1. The user accesses the Password Generator feature of the application.
2. They input the desired password length into the designated input field.
3. The system processes the input and generates a random, secure password according to the specified length and predefined criteria for strong passwords.
4. The application displays the generated password to the user.
5. The user may choose to generate another password if they are not satisfied with the first one. This step can be repeated as many times as needed until the user is content with the generated password.

By providing these two functionalities, the application allows users to both assess the security of their current passwords and generate new, secure passwords that meet their specific requirements. This use case diagram illustrates the primary interactions a user performs with the application to enhance their online security and protect their accounts from unauthorized access.

(Continue to previous topic: Go Back)

7.3.5 Wireframes

The wireframe for this project is designed to be straightforward, as most of the work on this project focuses on the backend of the system. The wireframe has been created using "Balsamiq Wireframes" and features a text field for users to enter their passwords.

There is a button labelled "Check" that, when pressed, provides the output for the strength of the password. The output includes the password's strength rating (weak, average, or strong) and suggestions for improvement, if applicable. The interface also offers users the option to reveal or hide their password by clicking an "eye" icon next to the text field.

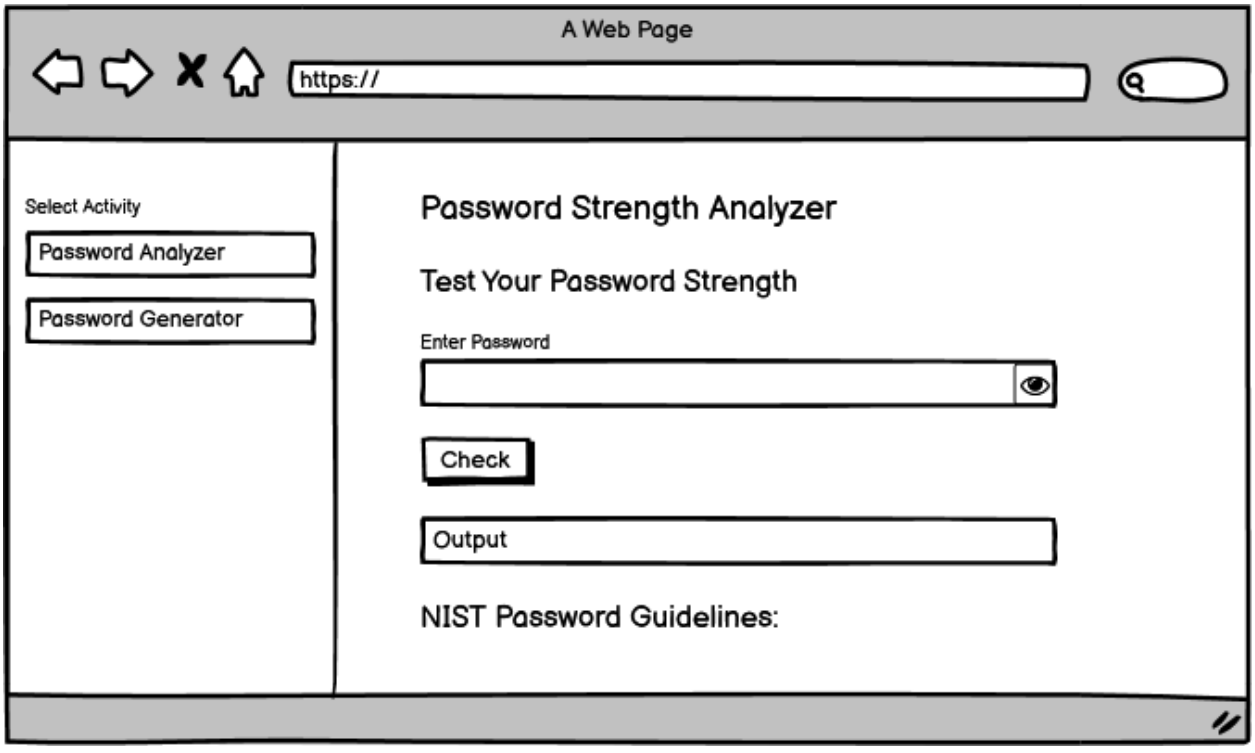


Figure 73: Wireframe of Password Analyzer

Additionally, there is a separate interface for generating random passwords. Users can input their desired password length in a designated field. After entering the desired length, they can press the "Generate" button, which will display a randomly generated password that meets the specified criteria. This password generator interface also includes "+" and "-" buttons to incrementally increase or decrease the desired password length.

The wireframe also features a navigation menu, allowing users to easily switch between the password strength analyzer and password generator interfaces. Overall, the design ensures a seamless and user-friendly experience, enabling users to quickly analyze and generate secure passwords.

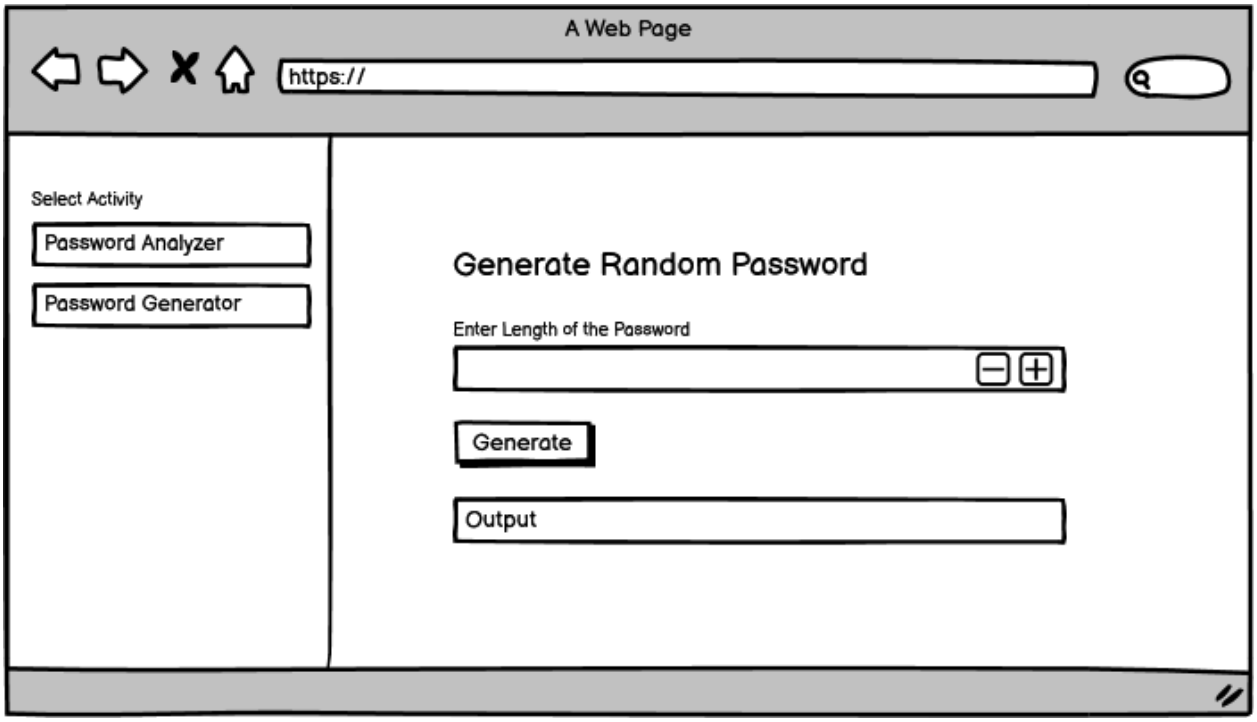


Figure 74: Wireframe of Password Generator

(Continue to previous topic: Go Back)

7.4 Appendix D: Screenshots of the System

Password Strength Analyzer

This page contains an input field where users can enter their password to check its strength as well as the status of compromise from the Have I Been Pwned database. The password text is hidden by default and can be made visible by clicking the eye icon on the right side of the text field. It also contains the password compliance requirements as recommended by NIST (National Institute of Standards and Technology).

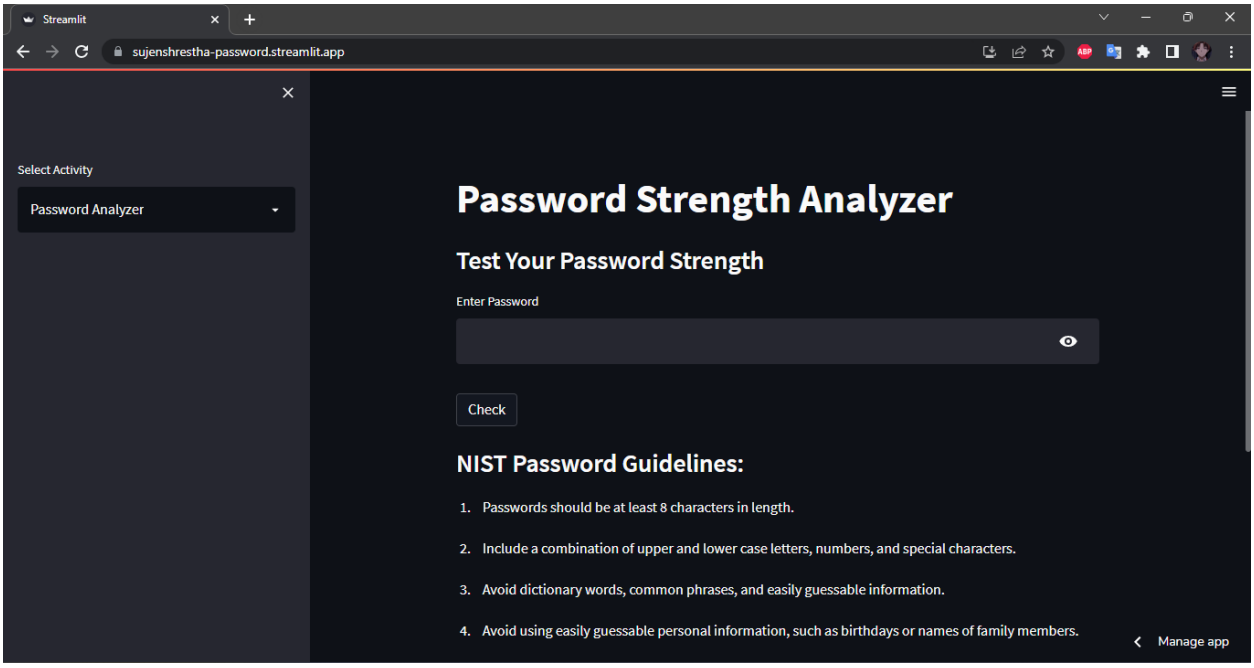


Figure 75: Interface for Password Strength Analyzer

Random Password Generator

This page contains an input field where users can enter a number to generate a secure password consisting of letters, numbers and symbols combined in a random order for the entered length. The minimum number that can be entered is 10 and maximum 64. The number can be entered manually or by using the buttons to increase or decrease the length by 1 beside the text field.

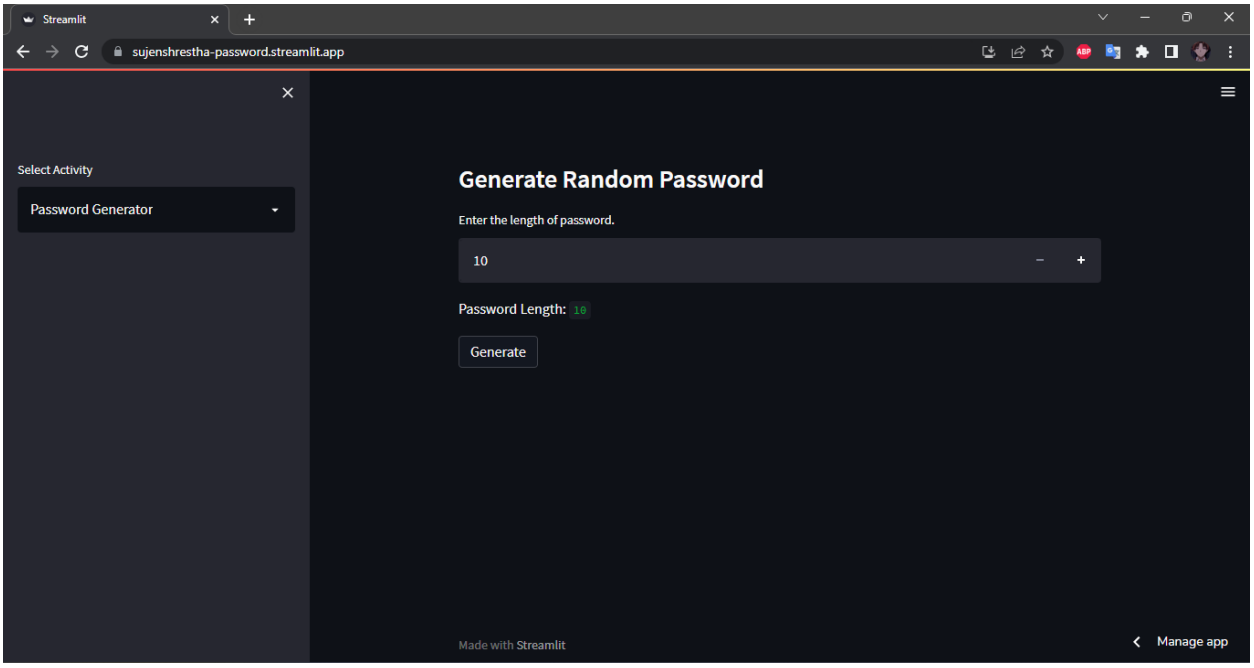


Figure 76: Interface for Random Password Generator

Appearance Settings

In the settings menu, the appearance of the application can be tweaked to make it wider, and the theme of the application can be changed to dark or light as preferred by the user.

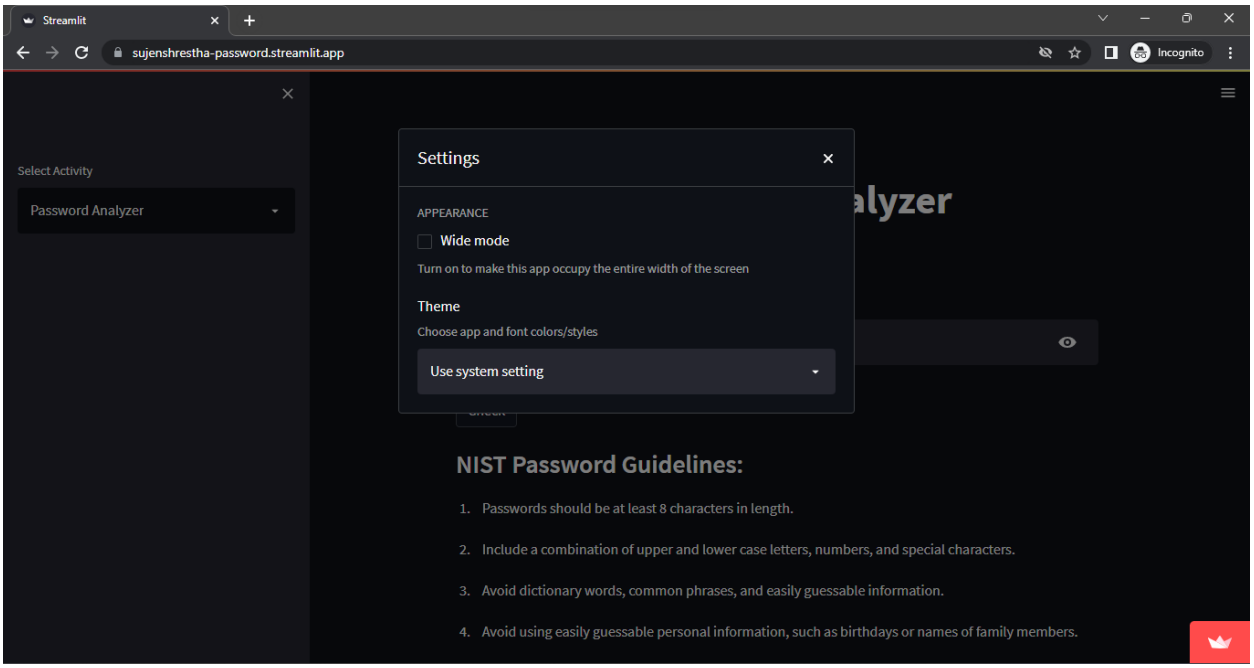


Figure 77: Appearance settings of the application

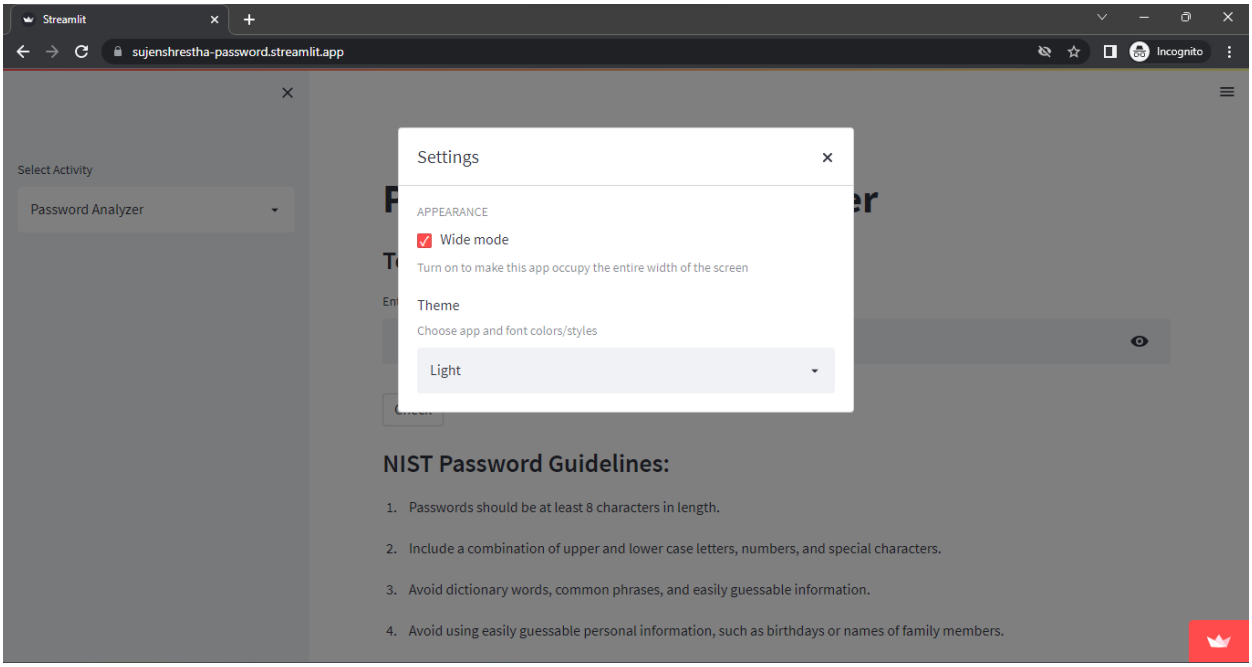


Figure 78: Changing the appearance settings of the application

7.5 Appendix E: Future Work

User Feedback and Iterative Development:

Actively seeking user feedback and incorporating it into the development process can lead to significant improvements in the tool's functionality and user experience. By encouraging users to provide feedback on their experience, any issues, bugs, or areas for improvement can be identified and addressed. This iterative development process ensures that the tool remains effective and user-centric, ultimately resulting in a more successful and widely adopted password security solution.

Language Support:

In order to cater to a global user base, it is essential to provide language support for the password strength analyzer and random password generator. By offering the tool in multiple languages, it can reach a wider audience and better serve users from different regions and backgrounds. This can be achieved by incorporating translation features, localization, and culturally appropriate content, ensuring that the tool is accessible and user-friendly for a diverse range of users.

Integration with Multi-factor Authentication:

Multi-factor authentication (MFA) systems provide an additional layer of security, making it more difficult for attackers to compromise accounts. Developing integration capabilities with various MFA systems, such as biometric identification, hardware tokens, or software-based authenticators, can help users enhance their account security. By incorporating MFA into the password strength analyzer, the tool can prompt users to enable additional security measures when setting up their accounts or changing their passwords, thus improving overall account security.

Performance Optimization:

To maintain a seamless user experience, it is crucial to continuously optimize the tool's performance. This includes addressing any issues related to loading times, server capacity, and response times. By regularly monitoring and improving the tool's performance, users can enjoy a smooth and efficient experience when using the password strength analyzer and random password generator.

API Development:

Developing an API for the password strength analyzer and random password generator can help increase the tool's reach and usability. By offering an API, other developers and organizations can integrate the tool's functionality into their own applications, websites, or platforms. This can lead to increased adoption of the tool and the promotion of better password security practices across various platforms.

Compatibility and Accessibility:

Ensuring compatibility with new software versions, hardware specifications, and browser updates is essential for maintaining the tool's usability and effectiveness. Regular updates should be conducted to address any compatibility issues that may arise. Additionally, prioritizing accessibility for users with disabilities is crucial. This can be achieved by incorporating features such as screen reader compatibility, keyboard navigation, and adherence to accessibility guidelines and standards.

By focusing on these areas of future work, the password strength analyzer and random password generator can become a more comprehensive and robust tool for users. By continuously improving its functionality, user experience, and security features, the tool can better serve its purpose of helping users create strong and secure passwords. This, in turn, will lead to enhanced account security, reducing the risk of data breaches and unauthorized access.