Project Report on Password Generator (C Programming)

Project Information

Project Title: Password Generator

Name’s : Swagato Pandith, Shahariyer Nafis, Sumaiya Jannat Kuasha, Mymuna Sultana

ID’s : 2252081040, 2252081041, 2252081042, 2252081077

Department : CSE

Semester : 1st(Summer-2025)

Batch : 64(B)

Introduction

Passwords are essential for protecting data and online accounts. Weak or reused passwords increase the risk of hacking. A password generator helps create strong, random, and secure passwords. Strong passwords are difficult for attackers to guess or crack. Using a generator saves time and avoids human bias in choosing passwords. It enhances online security by preventing identity theft and data breaches. Password generators often include features like length customization and symbol inclusion, reducing the chances of using predictable patterns.

Project Objectives

✔ To design a program that generates strong and random passwords
✔ Ensure that passwords include a mix of:
- Uppercase letters (A–Z)
- Lowercase letters (a–z)
- Numbers (0–9)
- Special characters (!, @, #, …)

✔ Provide users with customizable password length

Tools and Language Used

- Programming Language: C
- Compiler: Code::Blocks IDE
- Libraries Used: stdio.h, stdlib.h, time.h

Methodology

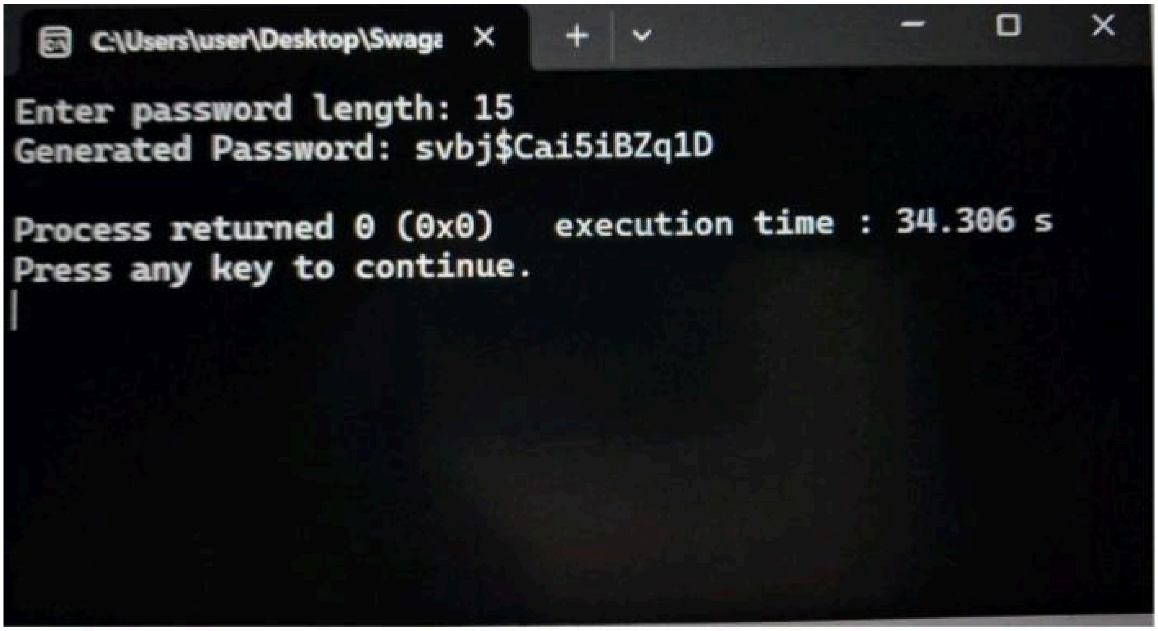
The following steps were implemented to develop the password generator:

1. Input desired password length from user.
2. Use random number generation function (rand()).
3. Select characters from different sets (letters, numbers, symbols).
4. Combine them into a password string.
5. Display the generated password to the user.

Implementation

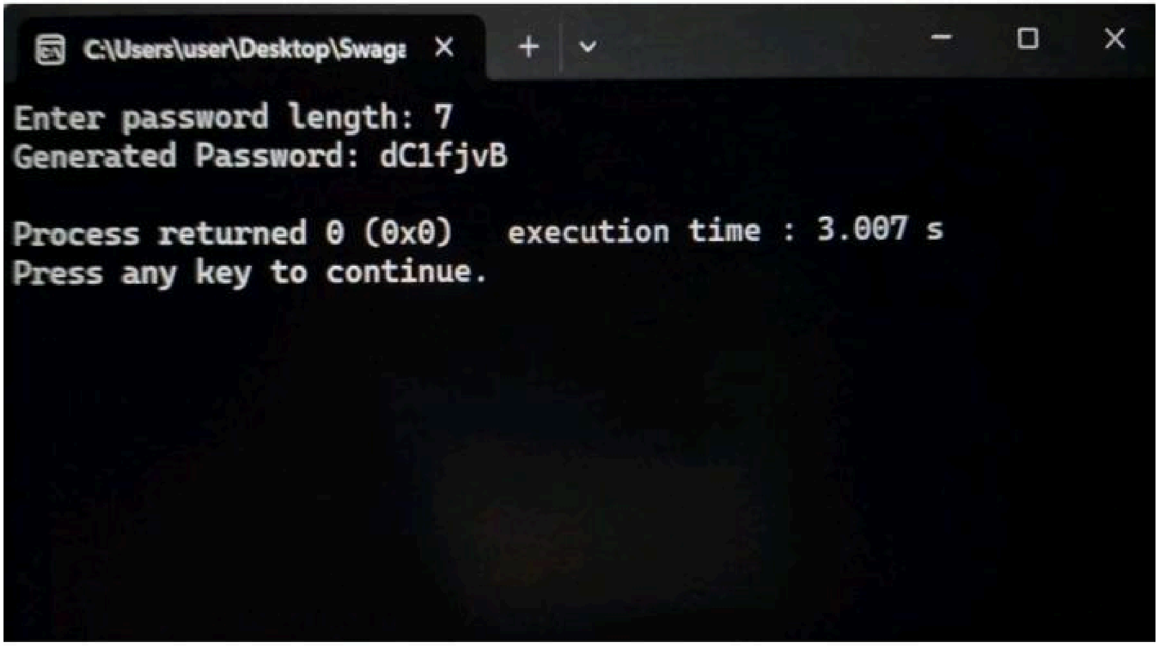
The program was implemented in C using the standard libraries mentioned above.

Sample Output



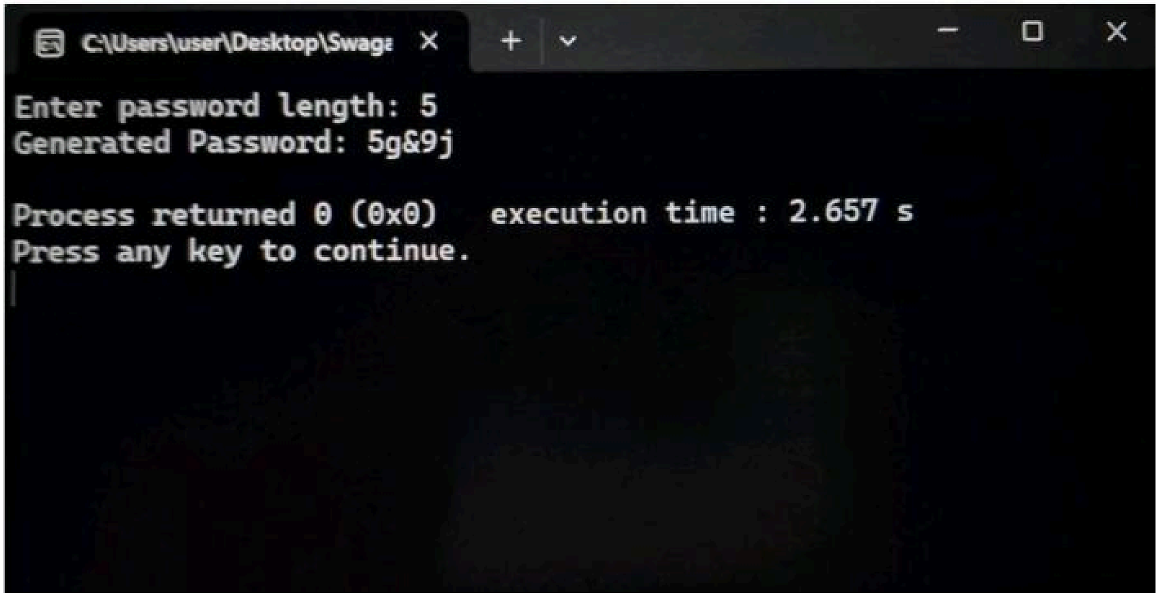
```
Enter password length: 15
Generated Password: svbj$Cai5iBZq1D

Process returned 0 (0x0)   execution time : 34.306 s
Press any key to continue.
```



```
Enter password length: 7
Generated Password: dC1fjvB

Process returned 0 (0x0)   execution time : 3.007 s
Press any key to continue.
```



```
Enter password length: 5
Generated Password: 5g&9j

Process returned 0 (0x0)   execution time : 2.657 s
Press any key to continue.
```

Advantages

- Generates strong and unique passwords.
- Saves time compared to manual password creation.
- Reduces chances of hacking or brute-force attacks.
- Ensures use of uppercase, lowercase, numbers, and special characters.
- Provides customizable password length.
- Easy to use and user-friendly.
- Works offline.
- Randomization ensures different passwords every time.

Limitations

- Difficult for users to remember very strong/random passwords.
- Does not store passwords (manual saving required).
- Limited character set.
- Cannot prevent password theft if stored insecurely.
- Weak passwords possible if length is too small.
- Generates only one password at a time.

Conclusion

This project demonstrates how C programming can be used to solve real-life problems such as generating secure passwords. By using libraries like stdio.h, stdlib.h, and time.h, we created a program that produces strong, unique, and random passwords within seconds. The system saves time, strengthens security, and minimizes risks of hacking or brute-force attacks. Although it has some limitations, the project serves as a foundation for more advanced password management systems in the future.