1. Title Page

Project Title:
SecurePass – A Simple Password Generator

Name:
Vempalle Mohammad Rahil

Organization:
SLASH MARK IT Solutions

Submission Date:
11 June 2025

2. Abstract

In the modern digital age, the importance of creating secure passwords cannot be overstated. Weak or reused passwords are one of the leading causes of cybersecurity breaches. This project aims to address this problem by developing a simple yet effective password generator using Python.

The main objectives are to generate strong, random passwords based on user preferences (length, inclusion of symbols, numbers, etc.) and ensure the ease of use for non-technical users. The approach uses Python's random and string libraries to create passwords dynamically. The system includes a basic graphical interface using Tkinter.

The key results of the project demonstrate the creation of passwords that meet standard security guidelines. The project is lightweight, customizable, and can be integrated into other security tools or websites.

In conclusion, this project shows that even basic tools can be used to solve real-world security problems effectively with the right approach.

3. Table of Contents

4. Introduction

With the increasing number of online accounts, the average user struggles to create and remember strong, unique passwords. Most people tend to reuse weak passwords, putting their digital identity at risk. This project is aimed at simplifying the process of generating secure passwords using an intuitive Python application.

5. Problem Statement

Users often struggle to create strong passwords. Common passwords like "123456" or "password" are highly insecure. The project aims to build a customizable tool that generates strong passwords with minimal effort.

6. Scope of the Project

Inclusions:

- Password generation using Python
- Option to include symbols, numbers, uppercase/lowercase letters

Exclusions:

- No password storage or database integration

Constraints:

- Only local application; no online version
- Basic GUI

Assumptions:

- Users prefer simplicity over complexity
- Python environment is already set up

7. Literature Review (Optional)

Various online tools exist but many do not allow user customization or transparency in how passwords are generated. This project builds on basic Python libraries to ensure complete transparency and customization.

8. Methodology

Approach:

- Requirement gathering
- Designing GUI and logic
- Coding the password generation
- Testing and final review

Technologies Used:

- Python 3
- Tkinter for GUI
- string and random modules for generation

Workflow Diagram:

css

CopyEdit

[Start] → [User Inputs] → [Password Logic] → [Password Output] → [End]

9. System Design and Architecture

System Overview:

- Simple script-based application with GUI interface

Architecture Diagram:

pgsql

CopyEdit

User Input

↓

Password Generator Logic

↓

Display Password

Data Flow:
User input → Generate password → Display password on screen

10. Implementation

Modules:

- Input length and preferences
- Generate password using string pools
- Display on screen

Code Snippet:

```
import random
import string

def generate_password(length=10):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for i in range(length))
    return password
```

Interface Screenshot:
(Insert screenshot of Tkinter app here if available)

11. Testing

Testing Types:

- Manual testing
- Input validation

Sample Test Case:

Test ID	Input Length	Include Symbols	Expected Result	Status
TC01	12	Yes	Password with 12 chars	Pass

12. Results and Discussion

- The tool successfully generated secure passwords based on user preferences.
- Easy to use interface allows flexibility.
- Performance is instant for small-scale use.

13. Challenges Faced

- GUI alignment using Tkinter was a bit tricky initially.
- Balancing randomness while ensuring readability in password length.

14. Conclusion

This project successfully delivers a functional password generator. It highlights how simple Python logic can be leveraged to address critical cybersecurity issues. The tool is user-friendly, secure, and adaptable.

15. Future Scope

- Add password strength meter
- Enable password copy to clipboard
- Convert to a web-based version using Flask

16. References/Bibliography

- Python Documentation: https://docs.python.org/3/
- Tkinter GUI Docs
- OWASP Password Policy Guide

17. Appendices

- Full Python code
- GUI Layout wireframe
- Output screenshots

18. Acknowledgments

I would like to thank **SLASH MARK IT Solutions** for their guidance and the support from my mentors who provided valuable insights during the project.