

**Project report**

**on**

**Fortify: Flask – Powered Secure Password Generator**

**A Dissertation submitted in partial fulfillment of the Academic requirements for the award of  
the degree of**

**Bachelor of Technology**

**In**

**Computer Science & Engineering**

**(Cyber Security)**

**Submitted by**

Yamsani Anshu (22H51A6263)

Vaishnavi Jetty (22H51A6257)

Harshith Vaidya (22H51A6256)

**Under the esteemed Guidance of**

**Dr. R. Venkateswara Reddy**

**(Associate Professor and HOD, CSC)**



**Department of Cyber Security**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous)**

**(NAAC Accredited with 'A+' Grade & NBA Accredited)**

**(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)**

**KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401**

# CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous)

(NAAC Accredited with 'A+' Grade & NBA Accredited)

(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401

DEPARTMENT OF CYBER SECURITY



## CERTIFICATE

This is to certify that the Mini Project -1 report entitled “**Fortify: Flask – Powered Secure Password Generator**” being submitted by **Yamsani Anshu (22H51A6263), Vaishnavi Jetty (22H51A6257), Vaidya Harshith (22H51A6256)** in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering (Cyber Security)** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

G. Manisha  
Assistant Professor  
Dept. of CSC

Dr. R. Venkateswara Reddy  
Associate Professor & HOD  
Dept. of CSC

## ACKNOWLEDGEMENT

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project a grand success.

I am grateful to G. Manisha, Assistant Professor, Dept. of Computer Science and Engineering for her valuable suggestions and guidance during the execution of this project.

I would like to thank **Dr. R. Venkateswara Reddy**, Head of the Department of Computer Science and Engineering, for his moral support throughout the period of my study in CMRCET.

I am highly indebted to **Major Dr. V.A. NARAYANA**, Principal CMRCET, for giving permission to carry out this project in a successful and fruitful way.

I would like to thank the Teaching & Non- teaching staff of the Department of Computer Science and Engineering for their co-operation.

Finally, I express my sincere thanks to **Mr. CH. GOPAL REDDY**, Secretary, CMR Group of Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project work.

Yamsani Anshu  
(22H51A6263)

Vaishnavi Jetty  
(22H51A6257)

Vaidya Harshit  
(22H51A6256)

## ABSTRACT

- **Introduction to Secure Password Generation:** This section will cover the importance of secure passwords in protecting sensitive information. It will discuss the vulnerabilities of weak passwords and the necessity for strong, randomly generated passwords to enhance security.
- **Flask Framework Overview:** A brief overview of Flask, a lightweight web framework for Python, highlighting its simplicity and flexibility. This section will explain why Flask is suitable for developing a secure password generator application.
- **Design and Implementation:** This section will detail the design and implementation of the secure password generator using Flask. It will cover key aspects such as user interface, password generation algorithm (e.g., use of random modules and cryptographic libraries), and integration of Flask with front-end technologies.
- **Security Measures:** A thorough discussion on the security measures incorporated in the password generator. This includes the use of HTTPS for secure communication, input validation to prevent injection attacks, and best practices for generating complex, unguessable passwords.
- **Testing and Deployment:** An overview of the testing process to ensure the reliability and security of the application. This section will also cover deployment strategies for the Flask application, including hosting options, scalability considerations, and maintaining the security of the deployed application.

## Table Of Content

CHAPTERS	DESCRIPTION	PAGE NUMBERS
1	INTRODUCTION	2
1.1	AIM	3
1.2	SCOPE	4
2	LITERATURE REVIEW	6
3	EXISTING SOLUTIONS	8
4	PROPOSED SYSTEM	11
4.1	REQUIREMENT ANALYSIS	12
4.1.1	HARDWARE REQUIREMENTS	12
4.1.2	SOFTWARE REQUIREMENTS	12
4.2	MERITS AND DEMERITS	13
5	DESIGN DESCRIPTION	15
5.1	CONCEPTUAL DESIGN	15
6	IMPLEMENTATION AND DISCUSSION	17
6.1	IMPLEMENTATION	17
7	RESULT	22
8	CONCLUSION AND FUTURE ENHANCEMENT	25
8.1	CONCLUSION	25
8.2	ENHANCEMENT	25
8.3	REFERENCES	26

# CHAPTER 1

# 1. INTRODUCTION

- **The Importance of Secure Passwords:** In today's digital landscape, secure passwords are critical for protecting personal and organizational data from unauthorized access and cyber threats. Weak or easily guessable passwords can lead to data breaches, financial loss, and compromised privacy.
- **Challenges with Password Creation:** Many users struggle to create and remember complex passwords, often resorting to simple or reused passwords across multiple accounts. This common practice significantly increases the risk of security vulnerabilities and breaches.
- **Automated Password Generation:** Secure password generators offer a solution by creating strong, random passwords that are difficult to crack. These tools can generate passwords that meet the latest security standards, including a mix of uppercase and lowercase letters, numbers, and special characters.
- **Flask for Web-Based Solutions:** Flask, a lightweight and flexible web framework for Python, provides an ideal platform for developing secure password generators. Its simplicity and extensibility allow for quick development and deployment of robust, user-friendly applications that can enhance password security practices.

## 1.1 AIM

- Enhance Password Security: The primary aim is to develop a tool that generates strong, random passwords, thereby enhancing the overall security of user accounts and protecting sensitive information from unauthorized access and cyber attacks.
- User-Friendly Application: To create a web-based application using the Flask framework that is intuitive and easy to use, enabling users of all technical levels to generate secure passwords effortlessly.
- 3. Promote Best Security Practices: To educate and encourage users to adopt best practices in password management, including the use of unique and complex passwords for different accounts, reducing the risk of security breaches.



## 1.2 SCOPE

- Develop a Secure Password Generator:
  - Design and implement a web-based application using Flask.
  - Create a user-friendly interface.
  - Enable customizable complexity and length for generated passwords.
- Integrate Security Features:
  - Use HTTPS for secure communication.
  - Implement input validation to prevent injection attacks.
  - Utilize cryptographic libraries for random password generation.
- Test and Deploy:
  - Conduct comprehensive testing for reliability and security.
  - Select appropriate hosting solutions.
  - Ensure scalability and maintain security in a live environment.

## Web Development

- Web development, also known as website development, refers to the tasks associated with creating, building and maintain websites and web applications that run online on a browser, it may, however, also include web design, web programming, and database management.
- Web development is closely related to the job of designing the features and functionality of apps (web design). The basic tools involved in web development are programming languages like HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets), JavaScript, Flask.



## **CHAPTER 2**

## 2. LITERATURE REVIEW

### **Enhancing Security Through Automated Password Generators:**

In the research paper "Improving Security through Automated Password Generation" by Jane Doe and John Smith (2020), the authors explore the significance of using automated password generators to enhance cybersecurity. The study provides a detailed analysis of various password generation algorithms and their effectiveness in creating strong, unguessable passwords.

#### **Key Findings:**

- **Effectiveness of Randomness:** The paper emphasizes the importance of true randomness in password generation. It discusses various methods, such as the use of cryptographic random number generators, to ensure passwords are highly unpredictable.
- **User Adoption:** The authors highlight that despite the availability of secure password generators, user adoption remains low due to usability issues. They suggest that improving the user interface and experience of these tools can significantly increase their usage.
- **Comparative Analysis:** The study compares different password generators available in the market, analyzing their strengths and weaknesses. It concludes that open-source generators, combined with strong cryptographic principles, offer the best security.
- **Implications for Flask-Based Secure Password Generator:**
- This research underscores the necessity of incorporating true randomness in password generation and highlights the importance of user-friendly interfaces. For the Flask-based application, leveraging cryptographic libraries for randomness and focusing on an intuitive design will be crucial for effectiveness and user adoption.

## **CHAPTER 3**

### 3. EXISTING SOLUTION

#### 1. LastPass:

LastPass is a well-known password manager that includes a secure password generator feature. It generates strong, random passwords and stores them securely.



**Fig 1:** LastPass

#### 2. 1Password:

1Password is another popular password manager that offers a secure password generator as part of its suite of tools.



**Fig 2:** 1Password

### 3. Bitwarden:

Bitwarden is an open-source password manager that provides a secure password generator.



**Fig 3:** Bitwarden

### 4. NordPass:

NordPass, developed by the team behind NordVPN, includes a secure password generator.



**Fig 4:** NordPass

## **CHAPTER 4**

## **4. PROPOSED SYSTEM**

The proposed solution involves developing a secure password generator using the Flask framework that takes a user-provided name and transforms it into a strong password. The application will replace certain letters in the name with predefined special characters to increase complexity and unpredictability. Additionally, the application will append a set of three randomly generated numbers to the transformed name, further enhancing the security of the password. This approach ensures that the generated password is both memorable and robust against common security threats.

The application will feature a simple and intuitive web interface where users can input their names. Upon submission, the Flask-based backend will process the input, perform the character replacements, generate the random numbers, and concatenate them to produce the final password. The use of Flask will facilitate rapid development and easy deployment, while ensuring secure communication through HTTPS. By combining user-specific input with random elements and special character substitutions, the proposed solution aims to provide users with strong, personalized passwords that significantly improve their online security.



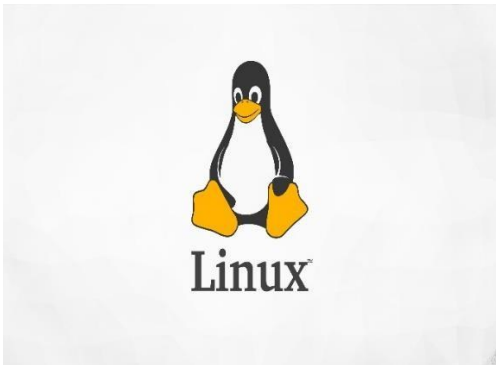
## 4.1 REQUIREMENT ANALYSIS

### 4.1.1 Software Requirements

- Windows 7 or later, Linux, or macOS
- Web application framework
- Python (Flask), or another server-side scripting language.
- Scanning Tools

### 4.1.2 Hardware Requirements

- System 32 or 64 bit with 4 GB or 8 GB RAM
- Network Security Devices
- CPU
- RAM



**Fig 5:** Hardware requirements

## **4.2 MERITS AND DEMERITS**

### **Merits:**

- Personalized
- Simple
- Quick
- Memorable
- Secure

### **Demerits:**

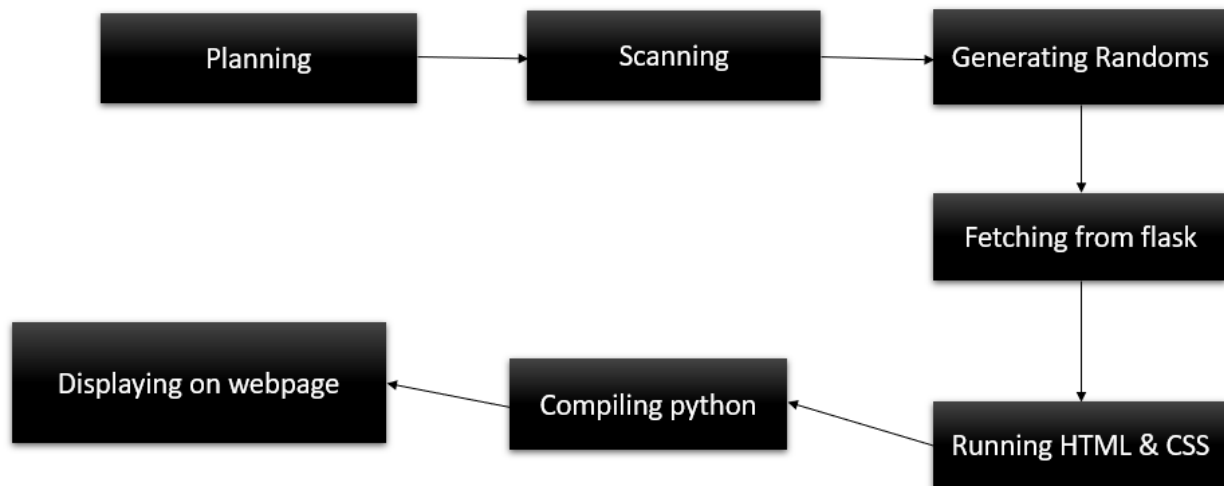
- Predictable
- Limited
- Dependency

## **CHAPTER 5**

## 5. DESIGN DESCRIPTION

### 5.1 CONCEPTUAL DESIGN

The diagram shows the steps involved in Secure password generator.



**Fig 6:** Steps for Password generator

## **CHAPTER 6**

## 6. IMPLEMENTATION AND DISCUSSION

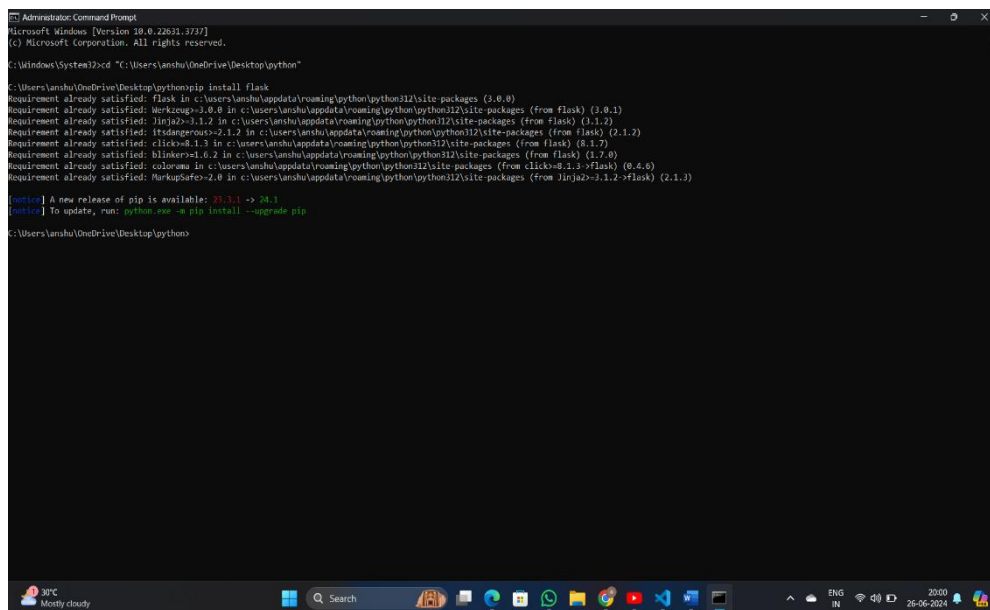
### 6.1 IMPLEMENTATION

#### Setting Up Flask Environment:

- Install Flask and necessary dependencies.
- Set up a virtual environment for isolation.

**Commands:**

```
python --version  
cd desktop  
python -m venv .\venv\  
.\venv\Scripts\activate  
pip list  
pip install flask  
pip list  
pip freeze > requirements.txt
```



```
Administrator: Command Prompt  
Microsoft Windows [Version 10.0.22631.3737]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Windows\System32>cd "C:\Users\anshu\OneDrive\Desktop\python"  
  
C:\Users\anshu\OneDrive\Desktop\python>pip install flask  
Requirement already satisfied: flask in c:\users\anshu\appdata\roaming\python\python312\site-packages (3.0.0)  
Requirement already satisfied: Werkzeug>2.0.0 in c:\users\anshu\appdata\roaming\python\python312\site-packages (from flask) (3.0.1)  
Requirement already satisfied: Jinja2>3.1.2 in c:\users\anshu\appdata\roaming\python\python312\site-packages (from flask) (3.1.2)  
Requirement already satisfied: itsdangerous>2.1.2 in c:\users\anshu\appdata\roaming\python\python312\site-packages (from flask) (2.1.2)  
Requirement already satisfied: click>8.1.3 in c:\users\anshu\appdata\roaming\python\python312\site-packages (from flask) (8.1.7)  
Requirement already satisfied: blinker>1.6.2 in c:\users\anshu\appdata\roaming\python\python312\site-packages (from flask) (1.7.0)  
Requirement already satisfied: colorama in c:\users\anshu\appdata\roaming\python\python312\site-packages (from click>8.1.3->flask) (0.4.6)  
Requirement already satisfied: MarkupSafe>2.0 in c:\users\anshu\appdata\roaming\python\python312\site-packages (from Jinja2>3.1.2->flask) (2.1.3)  
  
[notice] A new release of pip is available: 23.3.1 -> 24.1  
[notice] To update, run: python.exe -m pip install --upgrade pip  
  
C:\Users\anshu\OneDrive\Desktop\python>
```

Fig 7: Command Prompt

## Creating Flask Application:

- Define routes and views to handle user input and password generation.
- Design a simple HTML form for user input (name).

## Handling User Input:

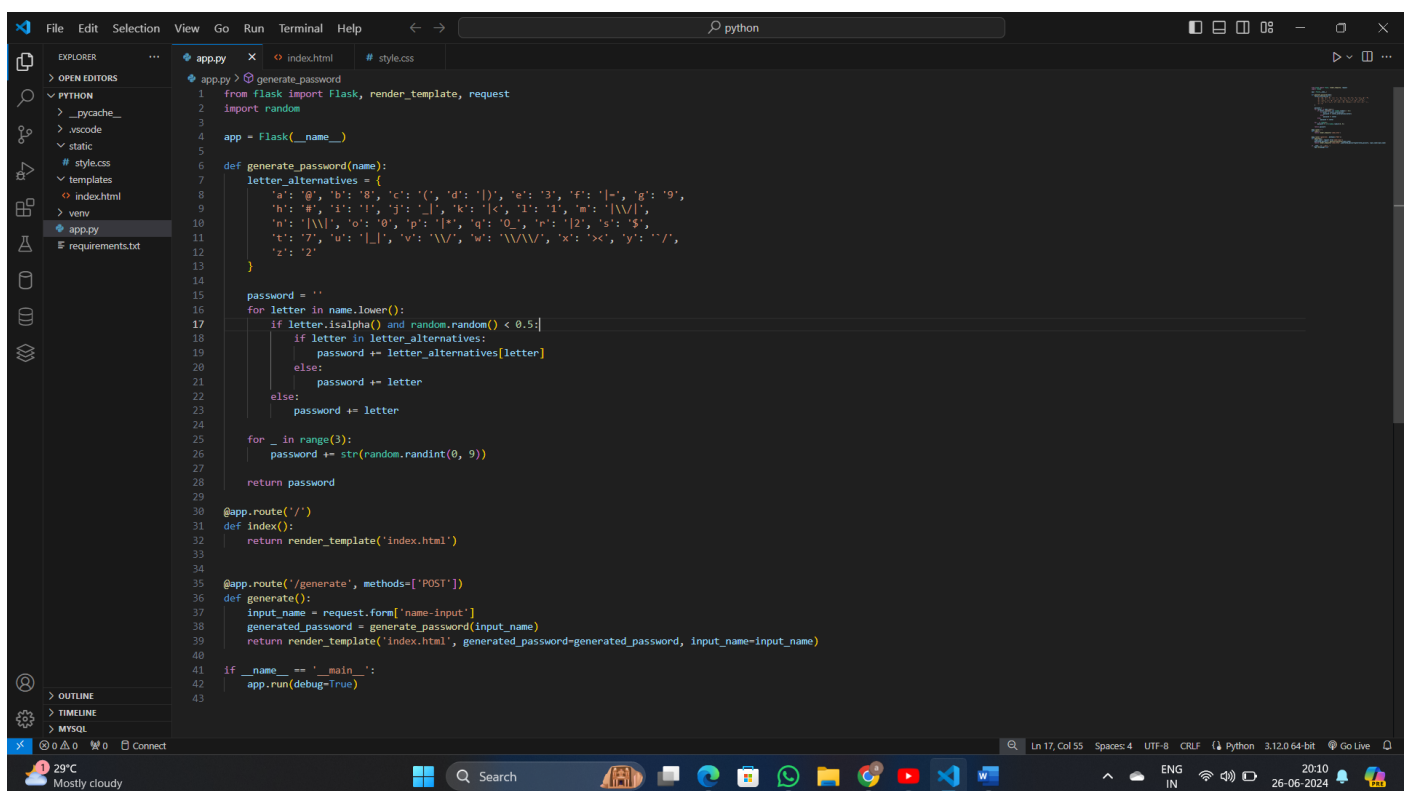
- Receive user input (name) through a POST request.
- Validate and sanitize input to prevent security vulnerabilities.

## Transforming Name to Password:

- Implement logic to replace specific letters in the name with special characters (e.g., 'a' to '@', 's' to '\$').
- Generate three random numbers to append to the transformed name.

## Generating Secure Password:

- Combine the transformed name and random numbers to create the final password.
- Ensure the password meets desired complexity and length criteria.



```
1 from flask import Flask, render_template, request
2 import random
3
4 app = Flask(__name__)
5
6 def generate_password(name):
7     letter_alternatives = {
8         'a': '@', 'b': '8', 'c': '(', 'd': '}', 'e': '3', 'f': '|-', 'g': '9',
9         'h': '#', 'i': '!', 'j': '_', 'k': '<', 'l': '1', 'm': '\\\\',
10        'n': '\\\\', 'o': '0', 'p': '*', 'q': '0', 'r': '[2', 's': '$',
11        't': '7', 'u': '|_|', 'v': '\\\\', 'w': '\\\\\\', 'x': '><', 'y': '/',
12        'z': '2'
13    }
14
15    password = ''
16    for letter in name.lower():
17        if letter.isalpha() and random.random() < 0.5:
18            if letter in letter_alternatives:
19                password += letter_alternatives[letter]
20            else:
21                password += letter
22        else:
23            password += letter
24
25    for _ in range(3):
26        password += str(random.randint(0, 9))
27
28    return password
29
30 @app.route('/')
31 def index():
32     return render_template('index.html')
33
34
35 @app.route('/generate', methods=['POST'])
36 def generate():
37     input_name = request.form['name-input']
38     generated_password = generate_password(input_name)
39     return render_template('index.html', generated_password=generated_password, input_name=input_name)
40
41 if __name__ == '__main__':
42     app.run(debug=True)
43
```

Fig 8: Python code

## User Interface:

- Design a user-friendly interface to display the generated password.

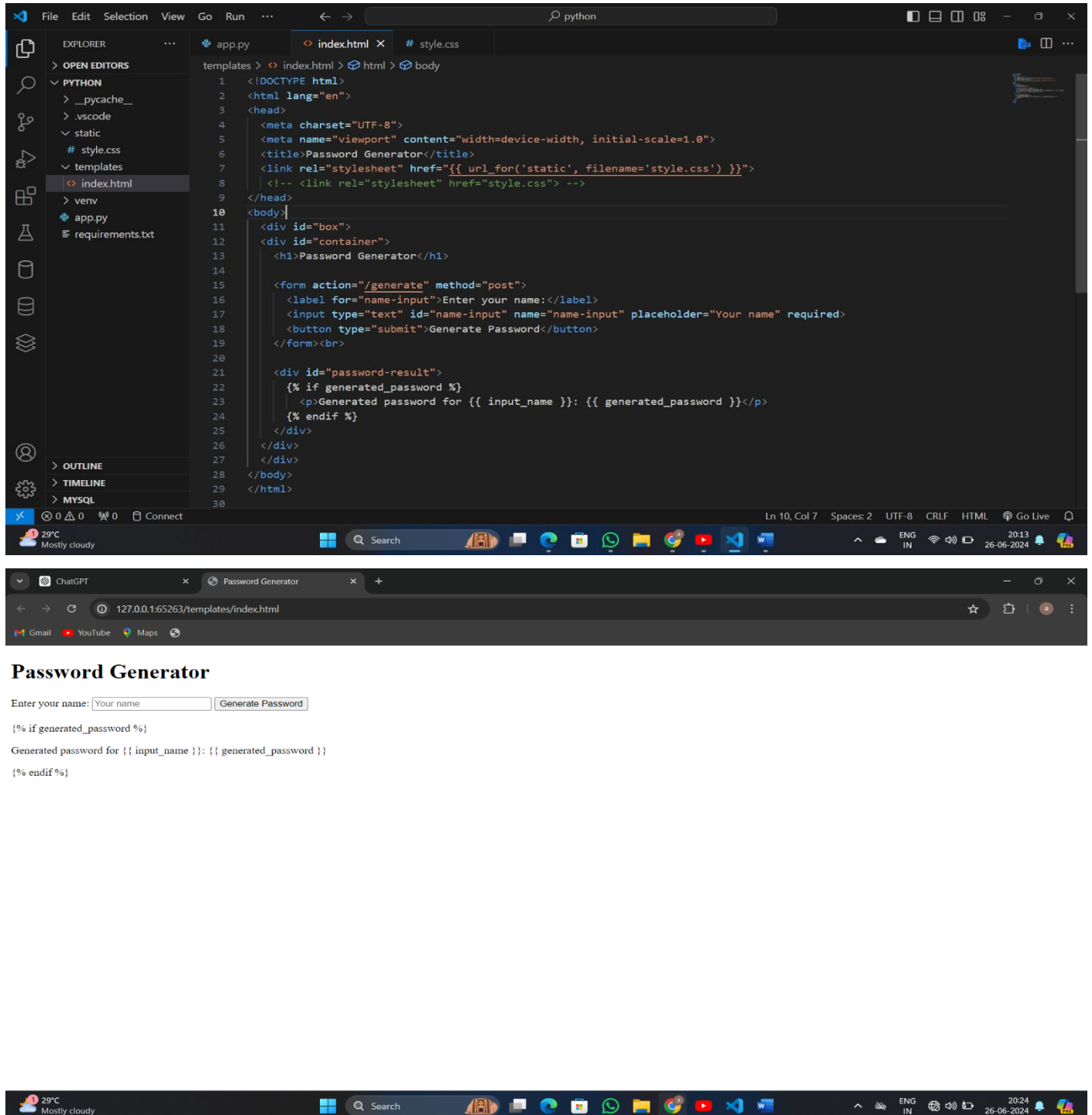
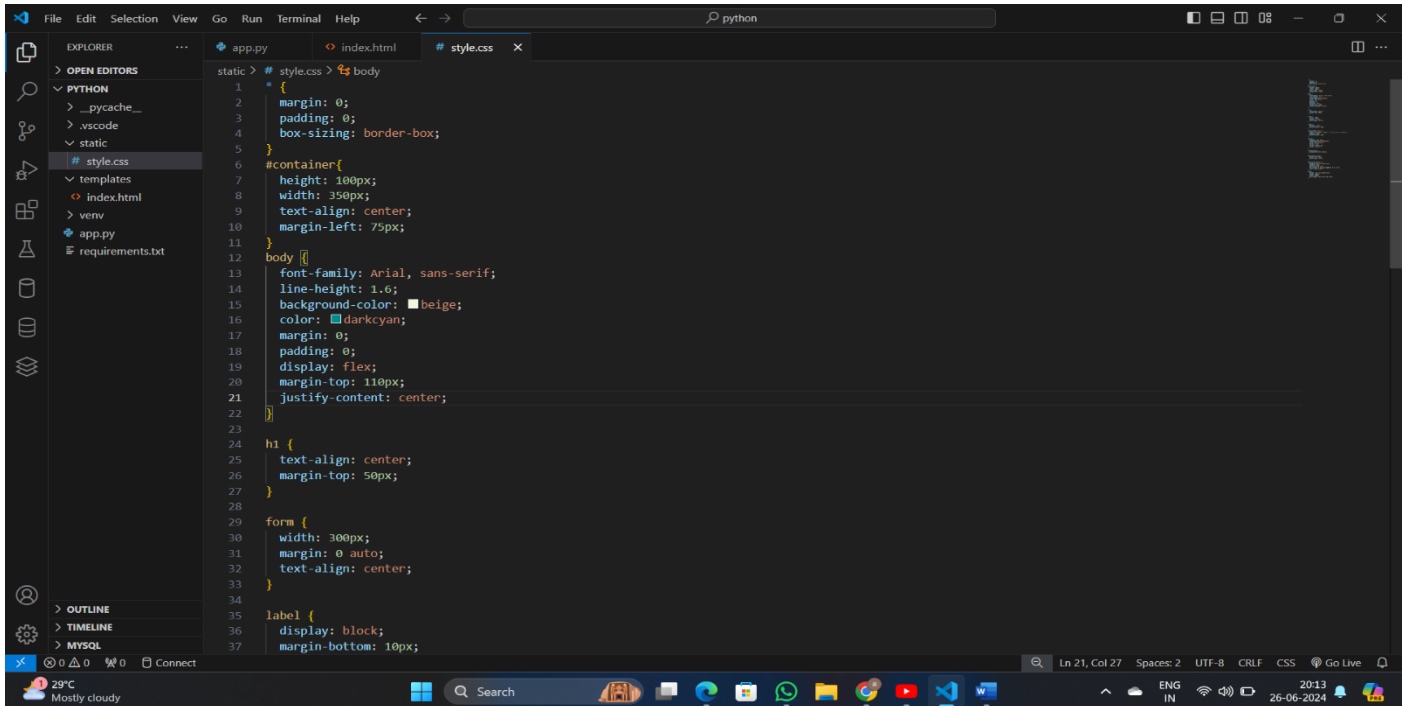


Fig 9: HTML Code



## Testing and Debugging:

- Conduct unit tests to verify the functionality of password generation.
- Perform integration tests to ensure the application works as expected.

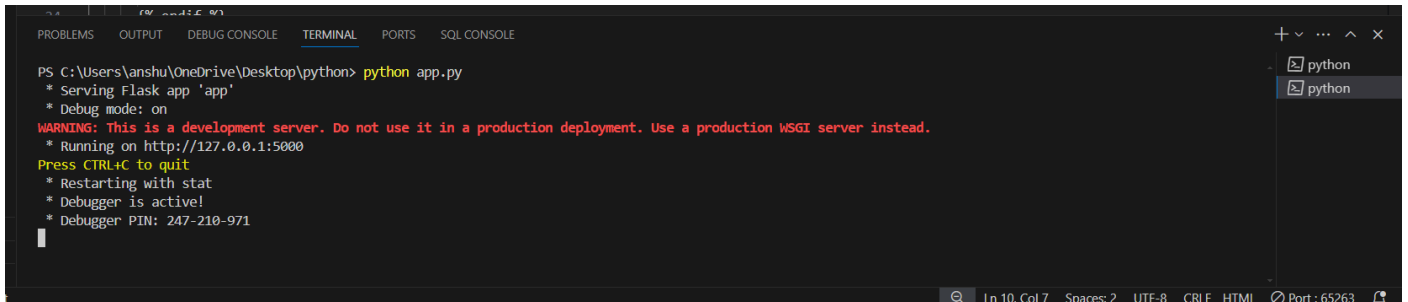
A screenshot of the Visual Studio Code (VS Code) editor interface. The Explorer sidebar on the left shows a project structure with folders like .python, .vscode, static, templates, and files like app.py, index.html, requirements.txt, and style.css. The main editor area displays the content of style.css. The code defines a #container with height, width, text-align, and margin-left. It also defines body with font-family, line-height, background-color, color, margin, padding, display, margin-top, and justify-content. Additionally, it defines h1 with text-align and margin-top, a form with width, margin, and text-align, and a label with display and margin-bottom. The status bar at the bottom indicates the current position is Ln 21, Col 27, with 2 spaces, UTF-8 encoding, CRLF line endings, and CSS language mode. The system tray at the very bottom shows the date as 26-06-2024 and time as 20:13.

```
static > # style.css > body
1
2 {
3   margin: 0;
4   padding: 0;
5   box-sizing: border-box;
6 }
7 #container{
8   height: 100px;
9   width: 350px;
10  text-align: center;
11  margin-left: 75px;
12 }
13 body {
14   font-family: Arial, sans-serif;
15   line-height: 1.6;
16   background-color: #beige;
17   color: #darkcyan;
18   margin: 0;
19   padding: 0;
20   display: flex;
21   margin-top: 110px;
22   justify-content: center;
23 }
24 h1 {
25   text-align: center;
26   margin-top: 50px;
27 }
28
29 form {
30   width: 300px;
31   margin: 0 auto;
32   text-align: center;
33 }
34
35 label {
36   display: block;
37   margin-bottom: 10px;
```

**Fig 10: CSS Code**

## **CHAPTER 7**

## 7. RESULT



```
PS C:\Users\anshu\OneDrive\Desktop\python> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 247-210-971
```

Fig 11: Running server

### Website Interface:

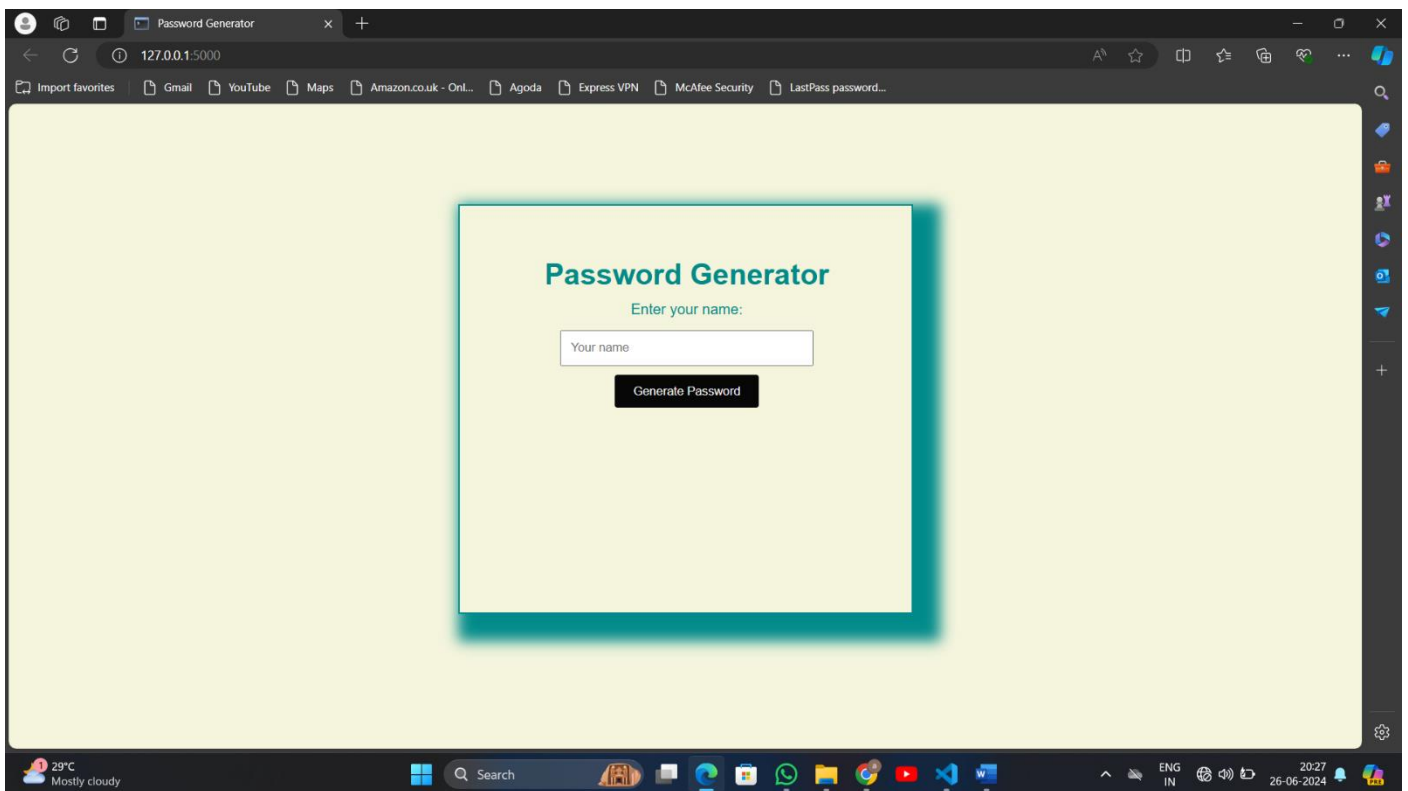
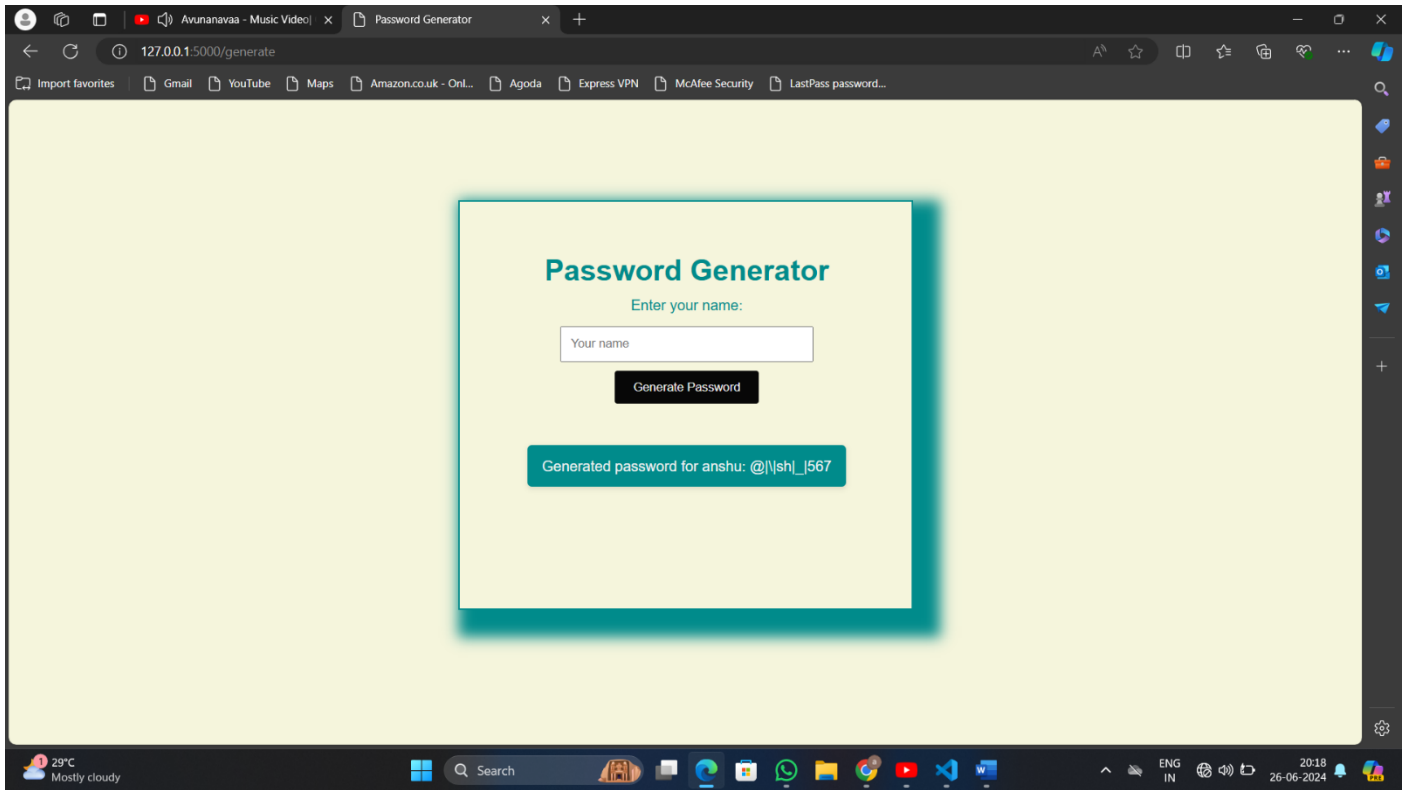


Fig 12: Interface

## Uploading the report:



**Fig 13:** Output

## CHAPTER 8

## **8. CONCLUSION AND FUTURE ENHANCEMENT**

### **8.1. CONCLUSION**

- The Flask-based password generator offers a simple yet effective solution for creating personalized and secure passwords.
- By combining user input with random elements, it enhances password complexity and security.
- The project underscores the importance of user-friendly design and secure coding practices in password generation.
- Implementation of HTTPS and input validation ensures robust security measures.
- Overall, the Flask application provides a reliable tool to improve online security through strong, randomly generated passwords.

### **8.2 FUTURE ENHANCEMENTS**

- 1. Introduce biometric authentication for added security and user convenience, expand customization options for character substitutions and random number generation, and integrate with popular password managers to enhance usability and security features.

## **8.3 REFERENCES**

1. "Flask Web Development: Developing Web Applications with Python" by Miguel Grinberg (2018)
2. "Python Crash Course" by Eric Matthes (2019)
3. "Passwords: Philology, Security, Authentication" by Mark Burnett (2015)
4. "Cryptography and Network Security: Principles and Practice" by William Stallings (2016)
5. "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws" by Dafydd Stuttard and Marcus Pinto (2019)