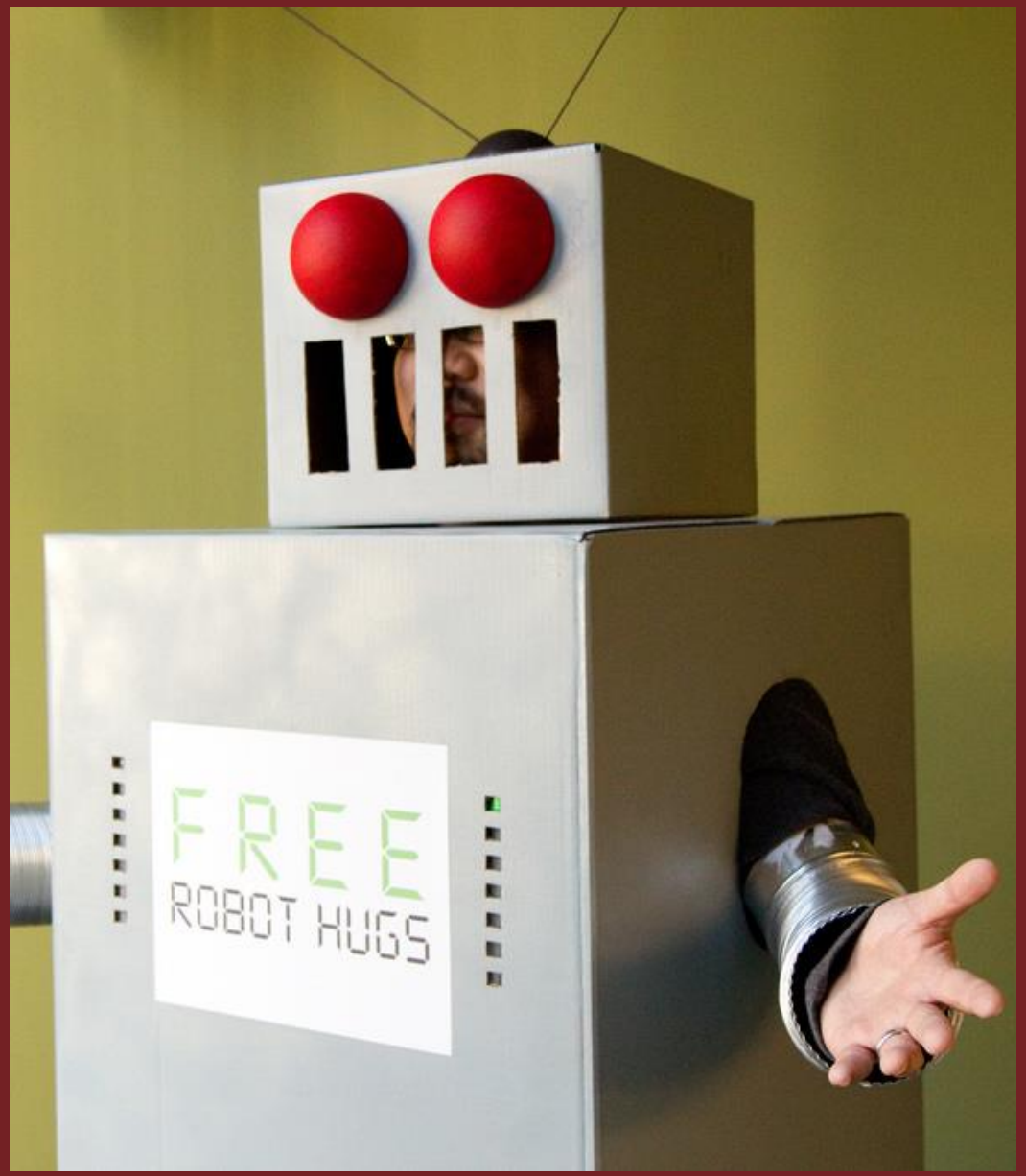
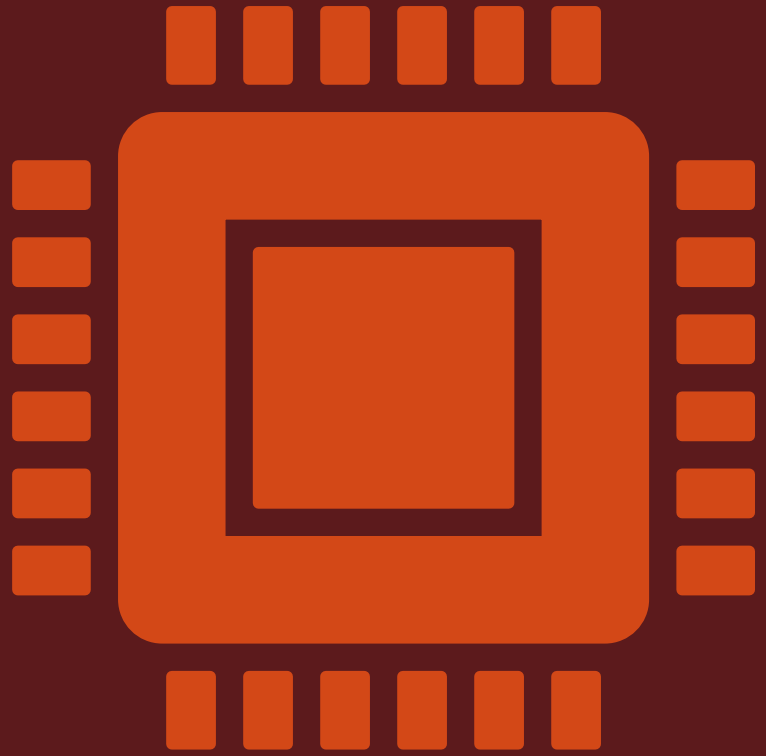
# PROJECT 1 – SECURITY VULNERABILITY MITIGATION CHATBOT

*Using Excel*



# INTRODUCTION

- Brief Overview of the Chatbot: Introduce the chatbot as an intelligent system designed to help users identify and mitigate security vulnerabilities in software. Importance of Security Vulnerability Mitigation: Discuss the critical nature of cybersecurity and the risks posed by unaddressed vulnerabilities. Purpose and Scope of the Chatbot Project: Explain the goals of the project, such as improving security practices, providing quick responses to security queries, and aiding in the education of users about potential security threats.



# PROBLEM STATEMENT

- Common Security Vulnerabilities: List common types of security vulnerabilities (e.g., SQL injection, XSS, CSRF).Challenges in Identifying and Mitigating Vulnerabilities: Highlight the difficulty in consistently identifying and addressing security issues, particularly for those without specialized knowledge. Need for an Automated Solution: Discuss why automation is essential for scalable, efficient, and effective security vulnerability management.

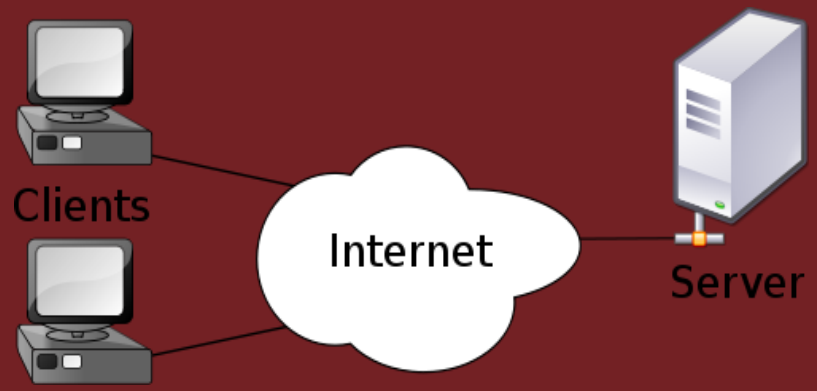
# SOLUTION OVERVIEW

- Introduction to the Chatbot: Describe what the chatbot is and its primary functions. Key Features and Functionalities: List key features such as real-time response, natural language understanding, and specific security advice. Addressing the Problem: Explain how the chatbot can assist users in identifying vulnerabilities and provide actionable mitigation strategies.

# ARCHITECTURE

- High-Level Overview: Provide a diagram or description of the overall system architecture. Client-Side Components: Describe the HTML, CSS, and JavaScript components that handle the user interface. Server-Side
- Components: Describe the server-side implementation using Python and Flask, handling the chatbot logic and response generation.
- Data Flow: Explain the flow of data from user input to server processing and back to the client.

# SERVER-SIDE IMPLEMENTATION AND CLIENT-SIDE IMPLEMENTATION



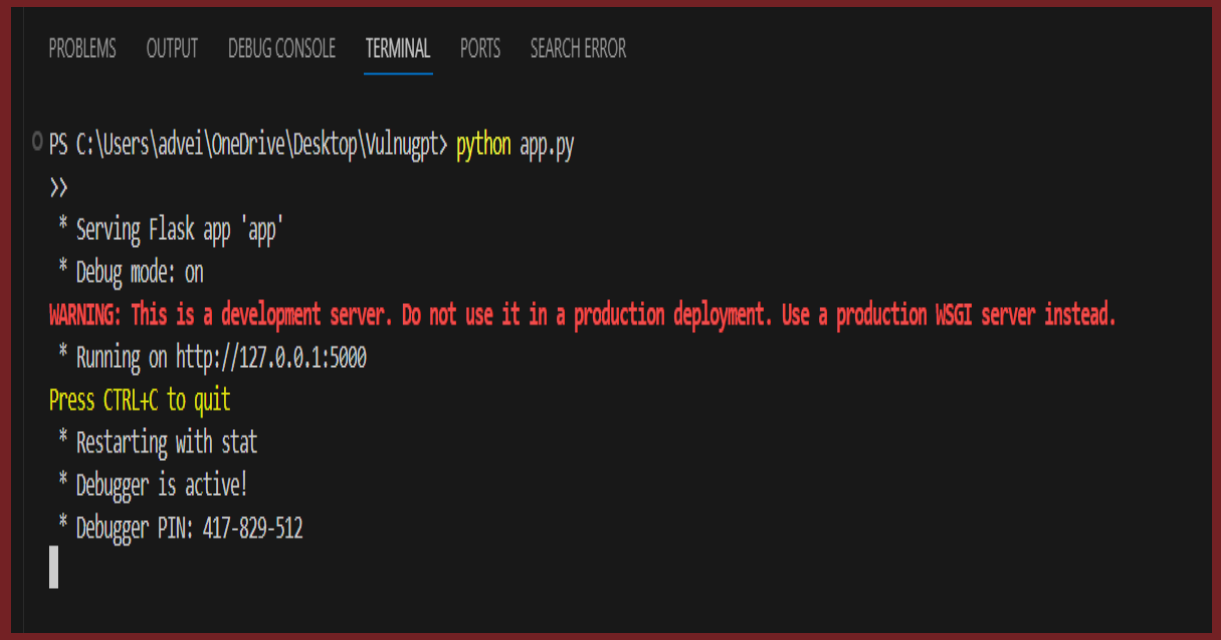

## Server-Side Implementation

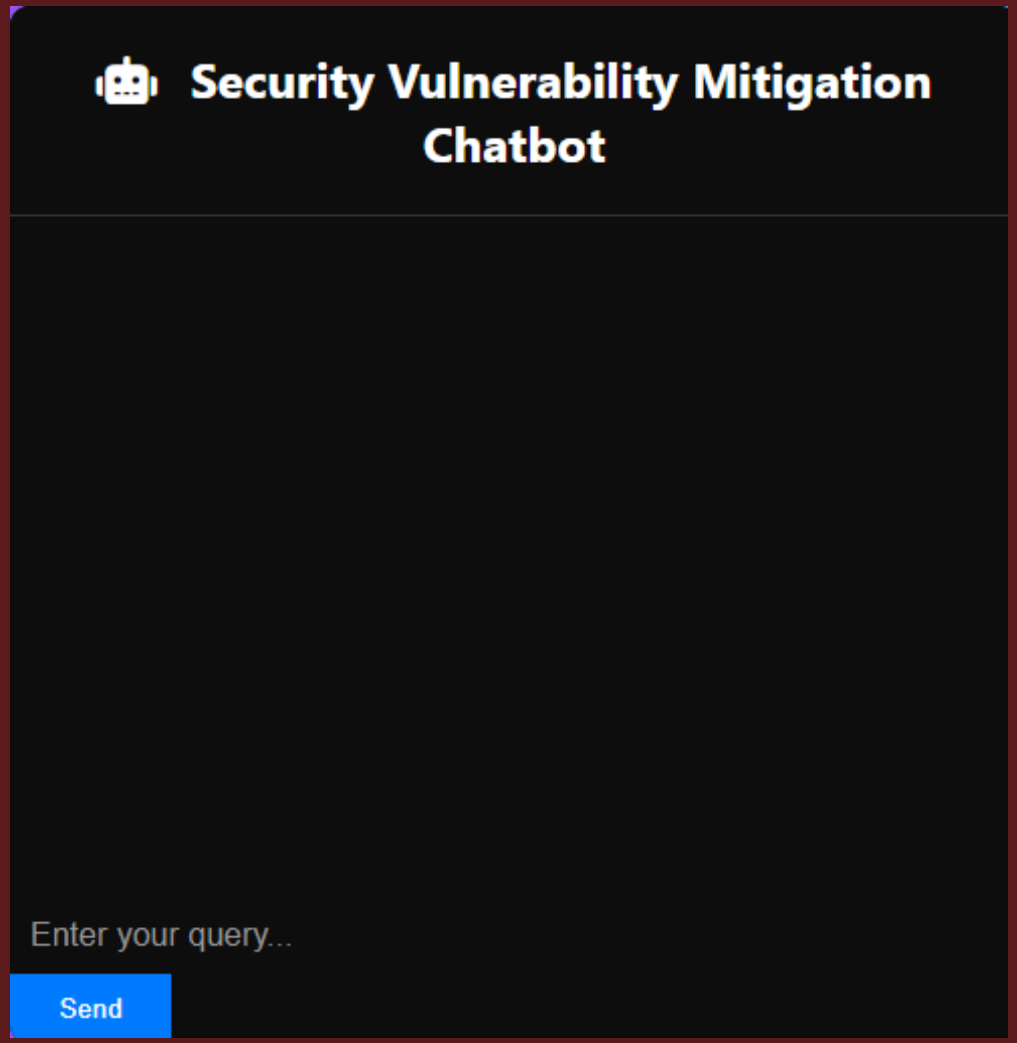
- Setting Up Flask Server: Installing Flask and setting up the environment Basic Flask application structure Defining the /chat Endpoint: Creating an endpoint to handle incoming messages Example of defining a Flask route Processing User Input: Logic for processing user input and generating responses Integration with the chatbot model or response logic Example Code Snippets: Key snippets of Python code to illustrate these points Example of Flask route definition and input processing Example of sending a response back to the client

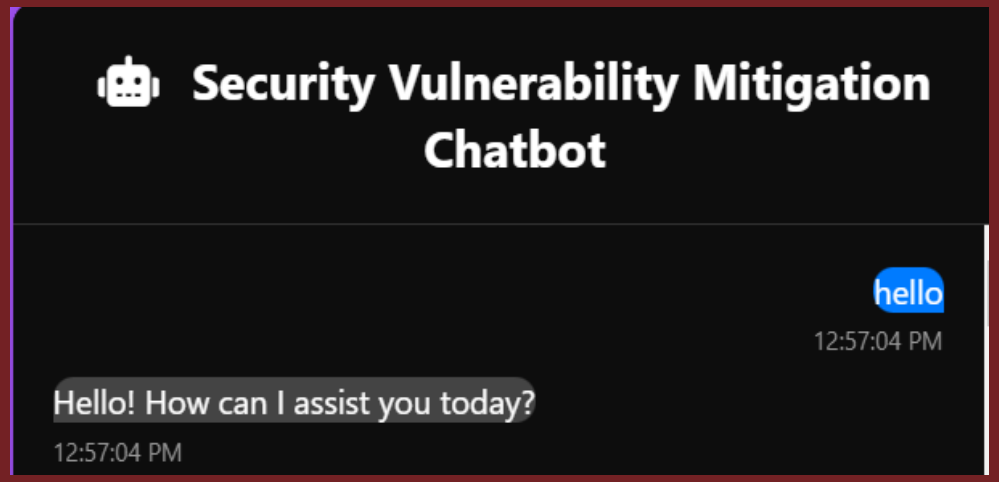
## Client-Side Implementation

- HTML Structure: Role of HTML in the UI Basic structure of the HTML document Key HTML elements used (e.g., input fields, buttons, containers)CSS Styling: CSS used for styling the chatbot interfascicular scheme, font choices, and layout design Responsive design considerations JavaScript Functionality: JavaScript functions that handle user inputs ending data to the server using fetch API Displaying responses in the UI Handling asynchronous communication with the server

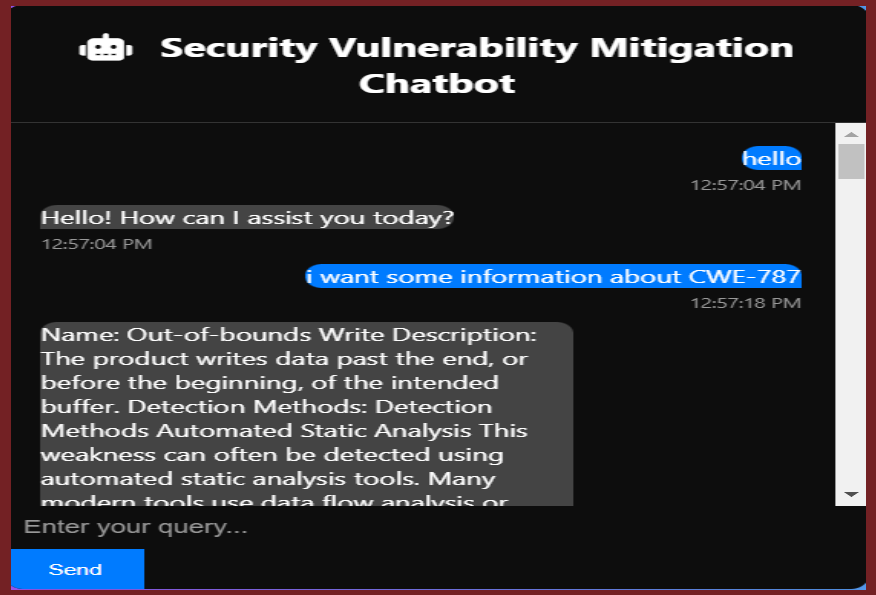
# SOME GLANCE OF THE CHATBOT AND THE WORKING






```
PS C:\Users\advei\OneDrive\Desktop\Vulnugpt> 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: 417-829-512
127.0.0.1 - - [06/Jun/2024 12:56:47] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Jun/2024 12:56:47] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [06/Jun/2024 12:57:04] "POST /chat HTTP/1.1" 200 -
```




```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

PS C:\Users\advei\OneDrive\Desktop\Vulnugpt> 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: 417-829-512
127.0.0.1 - - [06/Jun/2024 12:56:47] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Jun/2024 12:56:47] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [06/Jun/2024 12:57:04] "POST /chat HTTP/1.1" 200 -
127.0.0.1 - - [06/Jun/2024 12:57:18] "POST /chat HTTP/1.1" 200 -
```

# REFERENCES -

*Click here to get the link*