

Project 1 : SentinelShield – Advanced Intrusion Detection & Web Protection System

- **Project Overview :**

SentinelShield is a lightweight Intrusion Detection and Web Protection System developed as a practical cybersecurity project. The system simulates the behavior of a basic Web Application Firewall (WAF) by inspecting incoming HTTP requests, detecting malicious patterns, logging security events, and generating alerts.

This project helps in understanding how real-world security systems monitor, analyze, and respond to web-based attacks.

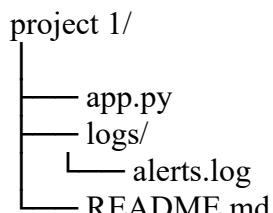
- **Features :**

- Detects SQL Injection attempts
- Detects Cross-Site Scripting (XSS) attacks
- Logs all malicious activities
- Generates real-time alerts
- Simple and lightweight implementation

- **Technologies Used :**

- Operating System: Windows / Kali Linux
- Programming Language: Python
- Framework: Flask
- Tools: VS Code, Terminal/PowerShell, Web Browser

- **Project Structure**



- How to Run the Project :
- Step 1: Install Dependencies

pip install flask or

.\venv\Scripts\python.exe -m pip install flask

The screenshot shows the VS Code interface with the following details:

- Project Structure:** Shows files `app.py` and `index.html`.
- Terminal Output:**

```
(.venv) PS C:\Users\ishika.soni\OneDrive\Documents\unified mentor\project 1> & "C:/Users/ishika.soni/OneDrive/Documents/unified mentor/project 1/.venv/Scripts/python.exe" "C:/Users/ishika.soni/OneDrive/Documents/unified mentor/project 1/app.py"
* Debug mode: on
● (.venv) PS C:\Users\ishika.soni\OneDrive\Documents\unified mentor\project 1> .\venv\Scripts\python.exe -m pip install flask
Requirement already satisfied: flask in \.\venv\lib\site-packages (3.1.2)
Requirement already satisfied: blinker>=1.9.0 in \.\venv\lib\site-packages (from flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in \.\venv\lib\site-packages (from flask) (8.3.1)
Requirement already satisfied: itsdangerous>=2.2.0 in \.\venv\lib\site-packages (from flask) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in \.\venv\lib\site-packages (from flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in \.\venv\lib\site-packages (from flask) (3.0.3)
Requirement already satisfied: werkzeug>=3.1.0 in \.\venv\lib\site-packages (from flask) (3.1.5)
Requirement already satisfied: colorama in \.\venv\lib\site-packages (from click>=8.1.3>flask) (0.4.6)
```
- Bottom Status Bar:** Shows Upcoming Earnings, system icons, and the date/time (21:24 05-02-2026).

- Step 2: Run the Application

python app.py

The screenshot shows the VS Code interface with the following details:

- Project Structure:** Shows files `app.py` and `index.html`.
- Terminal Output:**

```
from flask import Flask, request
import logging
app = Flask(__name__)
logging.basicConfig(
    filename="logs/alerts.log",
    level=logging.INFO
)
def detect_sql(payload):
    keywords = ["'", "or", "1=1", "union", "select", "--"]
    for word in keywords:
        if word in payload.lower():
            return True
    return False
def detect_xss(payload):
    if "script" in payload.lower():
        return True
    return False
@app.route("/")
def home():
    return "SentinelShield is Running!"
@app.route("/login")
def login():
    data = request.args.get("user", "admin")
    if detect_sql(data):
        logging.info(f"SQL Injection Attempt: {data}")
        return "⚠ Attack Detected!"
    if detect_xss(data):
        logging.info(f"XSS Attempt: {data}")
        return "⚠ Attack Detected!"
```
- Bottom Status Bar:** Shows system icons, the date/time (21:27 05-02-2026), and the Python version (3.14.2 (venv)).

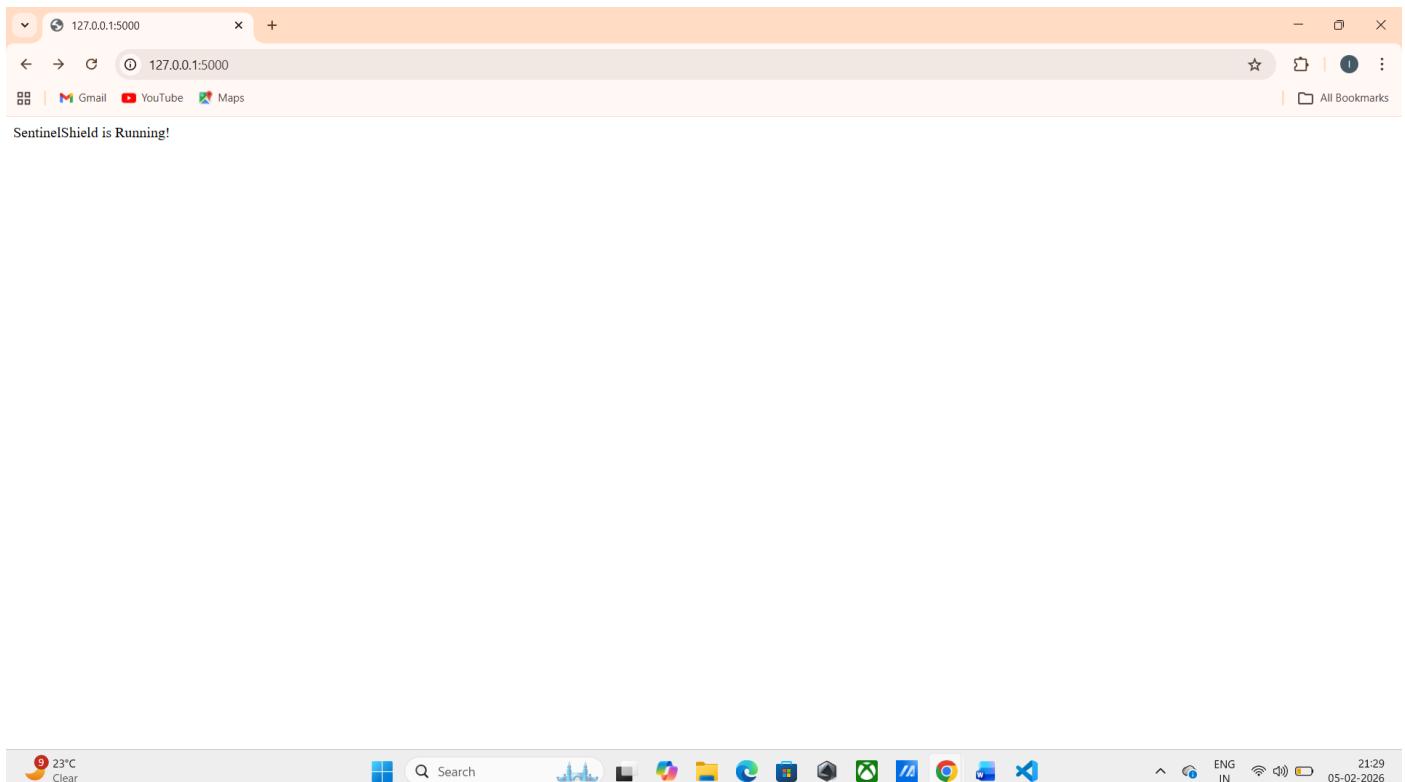
```
File Edit Selection View Go Run ... project 1 File Explorer Task List Terminal Help

app.py index.html
app.py ...
8     level=logging.INFO
9 )
10
11 def detect_sql(payload):
12     keywords = ["' or", "1=1", "union", "select", "--"]
13     for word in keywords:
14         if word in payload.lower():
15             return True
16     return False
17
18 def detect_xss(payload):
19     if "<script>" in payload.lower():
20         return True
21     return False
22
23 @app.route("/")
24 def home():
25     return "SentinelShield is Running!"
26
27 @app.route("/login")
28 def login():
29     data = request.args.get("user", "admin")
30
31     if detect_sql(data):
32         logging.info(f"SQL Injection Attempt: {data}")
33         return "⚠ Attack Detected!"
34
35     if detect_xss(data):
36         logging.info(f"XSS Attempt: {data}")
37         return "⚠ Attack Detected!"
38
39     return "Request Safe"
40
41 if __name__ == "__main__":
42     app.run(debug=True)

Ln 22, Col 1 Spaces: 4 UTF-8 CRLF [ ] Python 3.14.2 (venv) 21:27
23°C Clear Search ENG IN 05-02-2026
```

○ Step 3: Open in Browser

<http://127.0.0.1:5000>



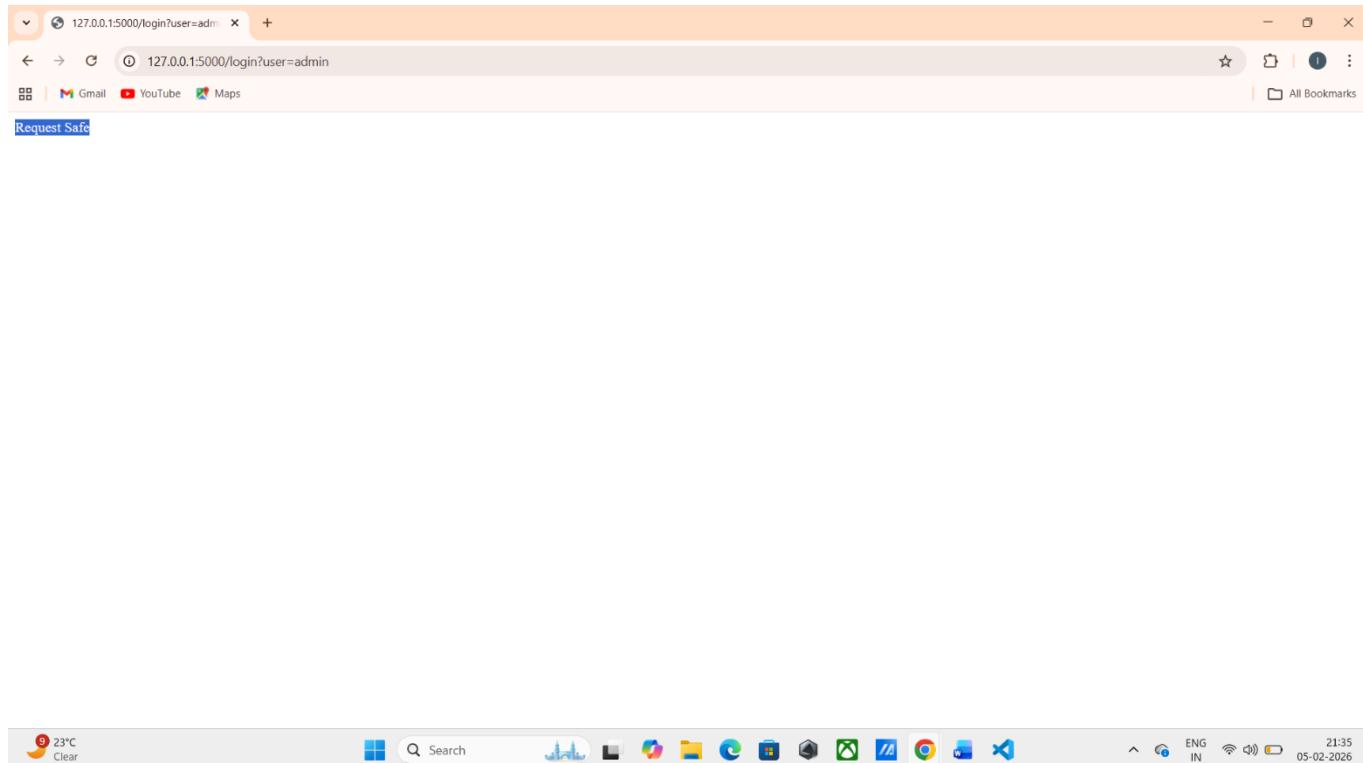
- **Testing the System :**

- **Normal Request**

<http://127.0.0.1:5000/login?user=admin>

- **Output:**

Request Safe

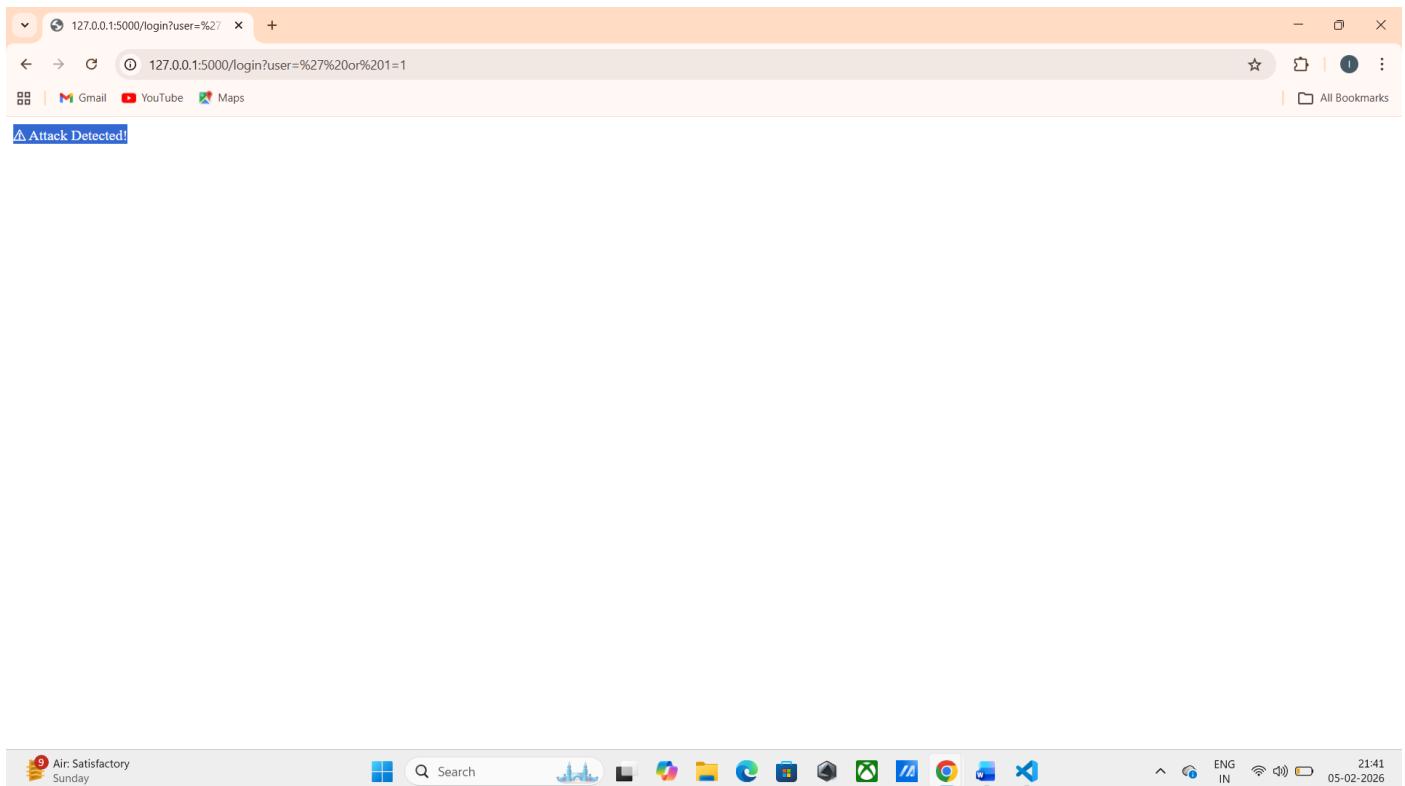


- **SQL Injection Test**

<http://127.0.0.1:5000/login?user=' or 1=1>

- **Output:**

Attack Detected!

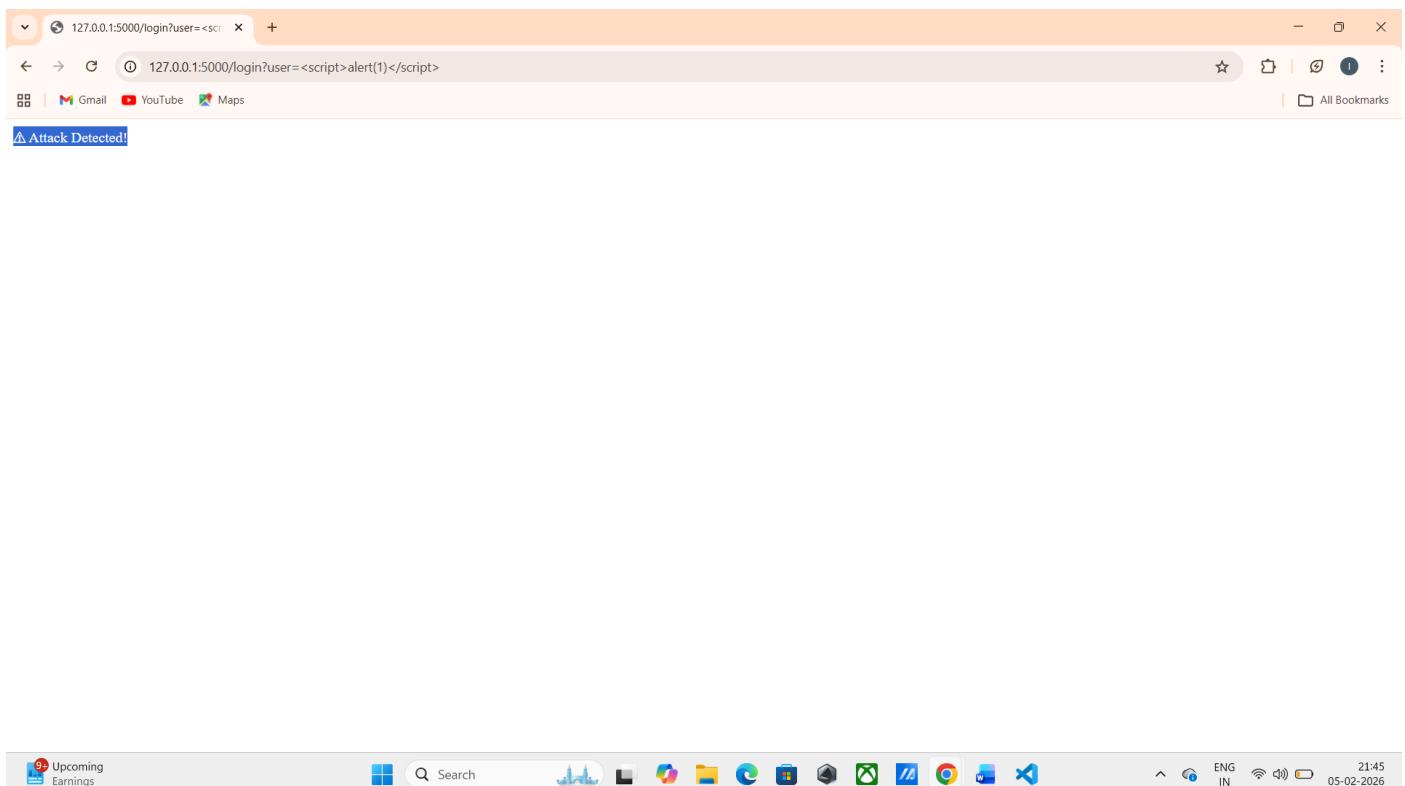


- **XSS Test**

`http://127.0.0.1:5000/login?user=<script>alert(1)</script>`

- **Output:**

Attack Detected!



• Log Output

All detected attacks are stored in:

- logs/alerts.log
- Example:

SQL Injection Attempt: ' or 1=1

XSS Attempt: <script>alert(1)</script>

```
alerts.log
File Edit View
INFO:werkzeug: * Restarting with stat
INFO:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 292-483-760
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 18:47:26] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 18:47:28] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 18:48:04] "GET / HTTP/1.1" 200 -
INFO:werkzeug: * Detected change in 'C:\Users\ishika soni\OneDrive\Documents\unified mentor\project 1\app.py', reloading
INFO:werkzeug: * Restarting with stat
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 292-483-760
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 18:48:04] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 18:48:04] "GET /login?user=admin HTTP/1.1" 200 -
INFO:root:SQL Injection Attempt: ' or 1=1
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:41] "GET /login?user=%20or%201=1 HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:41] "GET /login?user=%20or%201=1 HTTP/1.1" 200 -
INFO:werkzeug: * This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
INFO:werkzeug: * Running on http://127.0.0.1:5000
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:11] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:29] "GET /login?user=admin HTTP/1.1" 200 -
INFO:root:SQL Injection Attempt: ' or 1=1
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:41] "GET /login?user=%20or%201=1 HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:41] "GET /login?user=%20or%201=1 HTTP/1.1" 200 -
INFO:werkzeug: * This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
INFO:werkzeug: * Running on http://127.0.0.1:5000
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:41] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:41] "GET /login?user=admin HTTP/1.1" 200 -
INFO:root:SQL Injection Attempt: ' or 1=1
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:41] "GET /login?user=%20or%201=1 HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 20:06:41] "GET /login?user=%20or%201=1 HTTP/1.1" 200 -
INFO:root:XSS Attempt: <script>alert(1)</script>
INFO:werkzeug:127.0.0.1 - - [05/Feb/2026 21:45:14] "GET /login?user=<script>alert(1)</script> HTTP/1.1" 200 -
```

Ln 43, Col 1 874 of 2,689 characters Plain text 100% Windows (CRLF) UTF-8
24°C Clear Search 21:48 05-02-2026 ENG IN

• Learning Outcomes :

- Understanding of web application security
- Practical knowledge of intrusion detection
- Hands-on experience with Flask framework
- Basic implementation of logging and monitoring
- Real-world simulation of IDS/WAF concepts

• Future Enhancements :

- IP-based blocking system
- Brute force attack detection
- Web dashboard for monitoring
- Email or SMS alert system
- Integration with SIEM tools

