WEBSCAN PRO

– AUTOMATED WEB APPLICATION SECURITY TESTING TOOL

The Problem

- Web applications are highly vulnerable to attacks such as SQL Injection, XSS, and IDOR due to poor validation and access control.

Solution

WebScanPro is an automated security testing tool that scans a given URL, detects vulnerabilities, and generates a structured security report.

Objectives

- Detect common vulnerabilities (OWASP Top 10)
- Automate penetration testing
- Provide mitigation suggestions
- Generate visual security reports

Tools and test platforms

- **Languages:** Python

- **Libraries:** Requests, BeautifulSoup, Streamlit.
- **Testing Platforms:**
    - DVWA
    - OWASP Juice Shop
    - bWAPP

- **Output:** HTML/PDF Security Report

Week-1

Project Initialization & Setup

Problem

Manual security testing is time-consuming and inconsistent.

Approach

Defined project goals

Studied OWASP Top vulnerabilities

Set up vulnerable applications locally

Week-1

Solution Implemented

Installed DVWA / Juice Shop using XAMPP or Docker

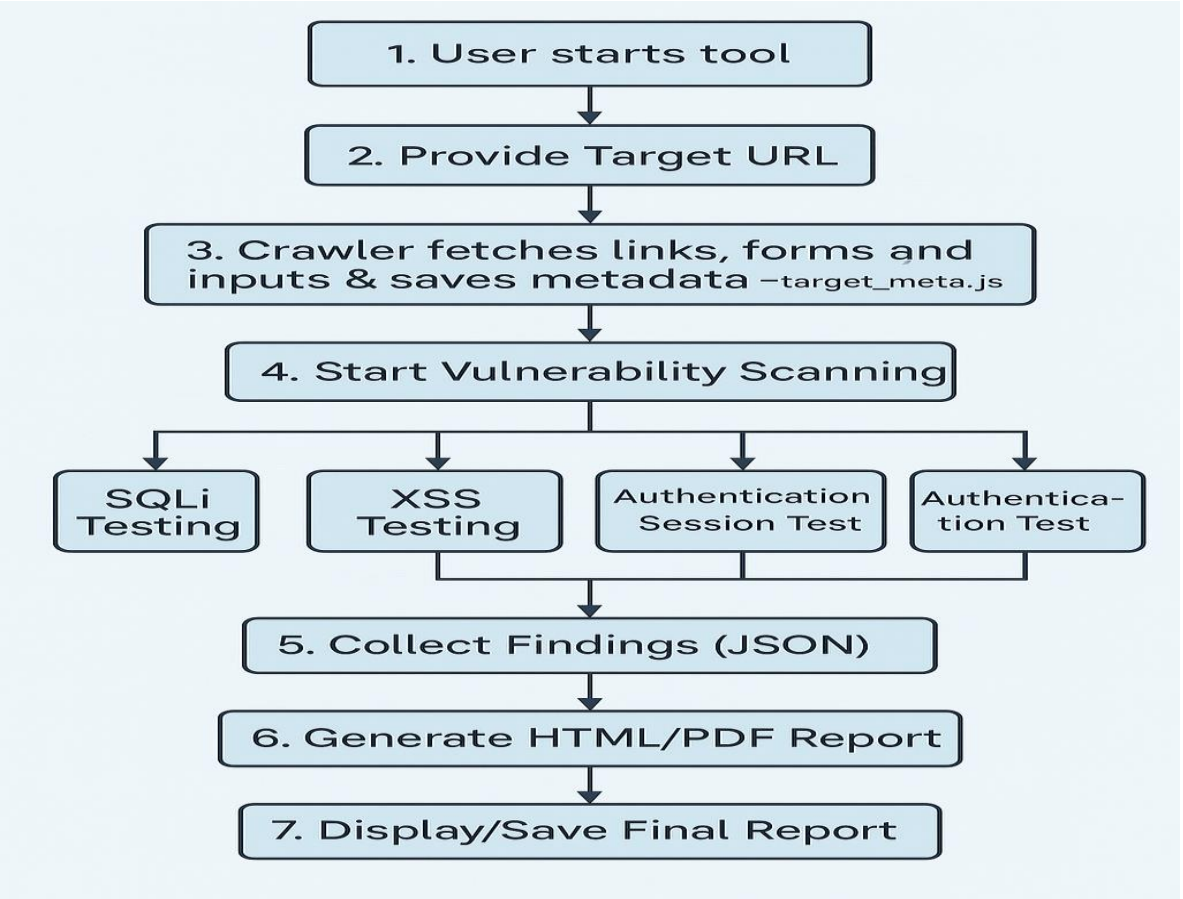
Explored pages, forms, URLs, parameters

Output

Working vulnerable web app environment

Identified attack surfaces

Workflow:


1. User starts tool
2. Provide Target URL
3. Crawler fetches links, forms and inputs & saves metadata –target_meta.js
4. Start Vulnerability Scanning
SQLi Testing
XSS Testing
Authentication Session Test
Authentica-tion Test
5. Collect Findings (JSON)
6. Generate HTML/PDF Report
7. Display/Save Final Report

Week-2

Use tools like BeautifulSoup or Selenium to extract interactive elements.

Problem

- Security testing requires knowing all entry points (forms, links, inputs).

Approach

- Web crawling
- HTML parsing

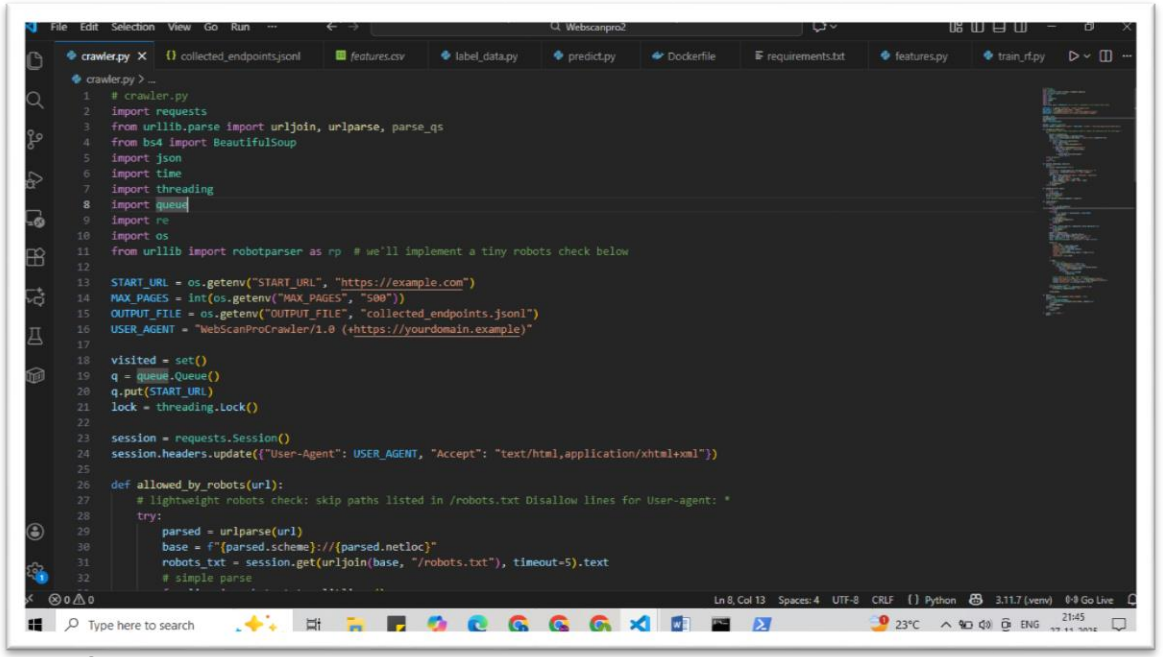
Solution Implemented

- Developed crawler using **BeautifulSoup**

Week-2

Extracted:

- URLs
- Forms
- Input fields
- Parameters



Output

Structured metadata of target website

Week-3

SQL Injection Testing

Problem

Improper input validation leads to SQL Injection attacks.

Approach

- Payload-based testing
- Error response analysis

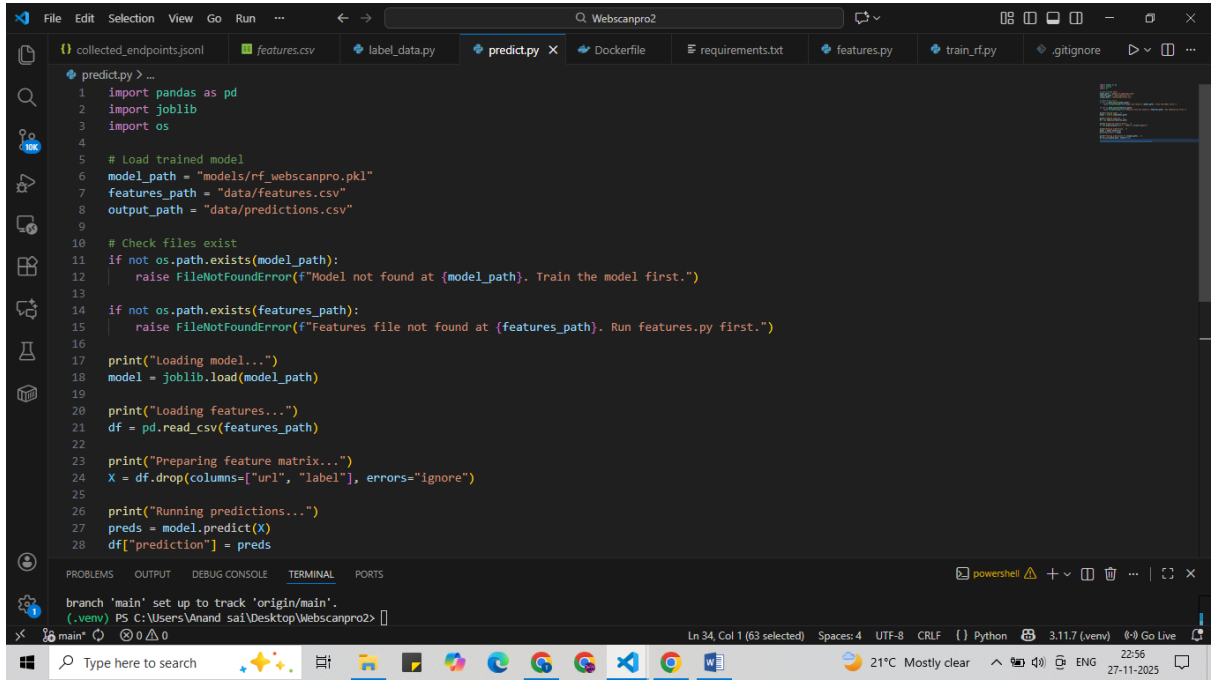
Solution Implemented

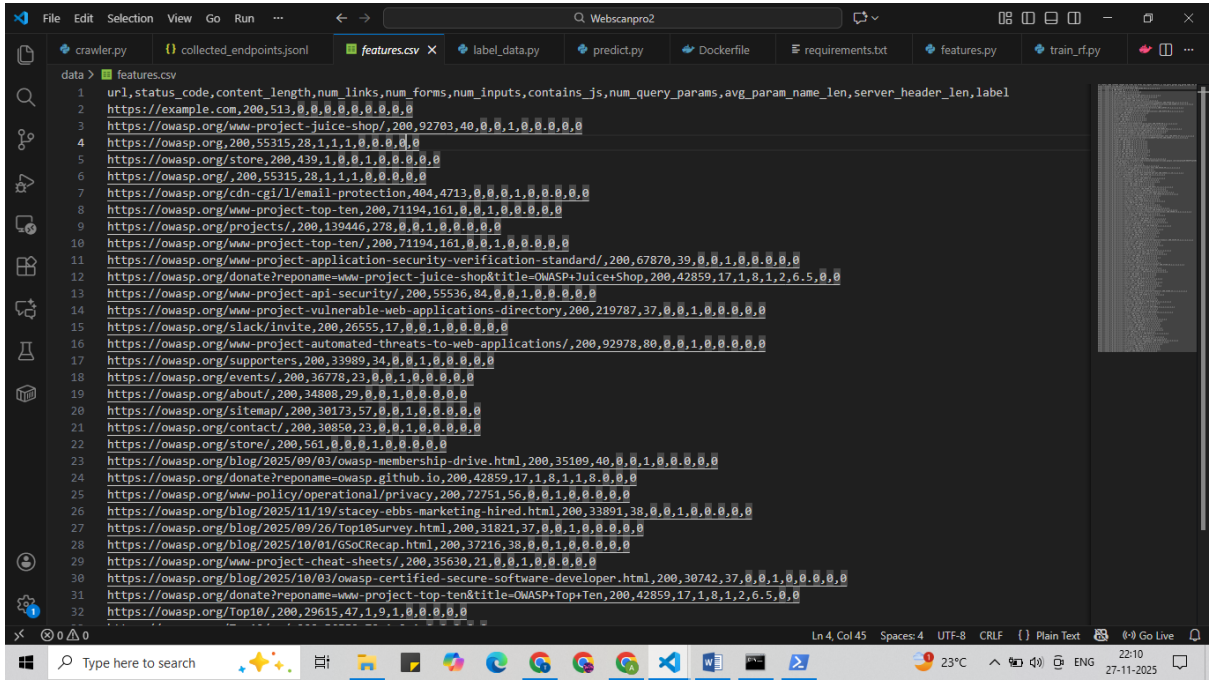
- Injected SQL payloads (' OR 1=1 --)
- Analyzed server responses
- Detected vulnerable endpoints

Output

- SQL Injection findings
- Severity classification
- Fix suggestions (Prepared Statements)

Output:


predict.py
import pandas as pd
import joblib
import os
# Load trained model
model_path = "models/rf_webscanpro.pkl"
features_path = "data/features.csv"
output_path = "data/predictions.csv"
# Check files exist
if not os.path.exists(model_path):
raise FileNotFoundError(f"Model not found at {model_path}. Train the model first.")
if not os.path.exists(features_path):
raise FileNotFoundError(f"Features file not found at {features_path}. Run features.py first.")
print("Loading model...")
model = joblib.load(model_path)
print("Loading features...")
df = pd.read_csv(features_path)
print("Preparing feature matrix...")
X = df.drop(columns=["url", "label"], errors="ignore")
print("Running predictions...")
preds = model.predict(X)
df["prediction"] = preds
branch 'main' set up to track 'origin/main'.
(.venv) PS C:\Users\Anand sai\Desktop\Webscanpro2>


features.csv
url,status_code,content_length,num_links,num_forms,num_inputs,contains_js,num_query_params,avg_param_name_len,server_header_len,label
https://example.com,200,513,0,0,0,0,0,0,0,0,0
https://owasp.org/www-project-juice-shop/,200,92703,40,0,0,1,0,0,0,0,0
https://owasp.org,200,55315,28,1,1,1,0,0,0,0,0
https://owasp.org/store,200,439,1,0,0,1,0,0,0,0,0
https://owasp.org/,200,55315,28,1,1,1,0,0,0,0,0
https://owasp.org/cdn-cgi/l/email-protection,404,4713,0,0,0,1,0,0,0,0,0
https://owasp.org/www-project-top-ten,200,71194,161,0,0,1,0,0,0,0,0
https://owasp.org/projects/,200,139446,278,0,0,1,0,0,0,0,0
https://owasp.org/www-project-top-ten/,200,71194,161,0,0,1,0,0,0,0,0
https://owasp.org/www-project-application-security-verification-standard/,200,67870,39,0,0,1,0,0,0,0,0
https://owasp.org/donate?reponame=www-project-juice-shop&title=OWASP+Juice+Shop,200,42859,17,1,8,1,2,6.5,0,0
https://owasp.org/www-project-api-security/,200,55536,84,0,0,1,0,0,0,0,0
https://owasp.org/www-project-vulnerable-web-applications-directory,200,219787,37,0,0,1,0,0,0,0,0
https://owasp.org/slack/invite,200,26555,17,0,0,1,0,0,0,0,0
https://owasp.org/www-project-automated-threats-to-web-applications/,200,92978,80,0,0,1,0,0,0,0,0
https://owasp.org/supporters,200,33989,34,0,0,1,0,0,0,0,0
https://owasp.org/events/,200,36778,23,0,0,1,0,0,0,0,0
https://owasp.org/about/,200,34808,29,0,0,1,0,0,0,0,0
https://owasp.org/sitemap/,200,30173,57,0,0,1,0,0,0,0,0
https://owasp.org/contact/,200,30850,23,0,0,1,0,0,0,0,0
https://owasp.org/store/,200,561,0,0,0,1,0,0,0,0,0
https://owasp.org/blog/2025/09/03/owasp-membership-drive.html,200,35109,40,0,0,1,0,0,0,0,0
https://owasp.org/donate?reponame=owasp.github.io,200,42859,17,1,8,1,1,8.0,0,0
https://owasp.org/www-policy/operational/privacy,200,72751,56,0,0,1,0,0,0,0,0
https://owasp.org/blog/2025/11/19/stacey-ebbs-marketing-hired.html,200,33891,38,0,0,1,0,0,0,0,0
https://owasp.org/blog/2025/09/26/Top10Survey.html,200,31821,37,0,0,1,0,0,0,0,0
https://owasp.org/blog/2025/10/01/GSoCRecap.html,200,37216,38,0,0,1,0,0,0,0,0
https://owasp.org/www-project-cheat-sheets/,200,35630,21,0,0,1,0,0,0,0,0
https://owasp.org/blog/2025/10/03/owasp-certified-secure-software-developer.html,200,30742,37,0,0,1,0,0,0,0,0
https://owasp.org/donate?reponame=www-project-top-ten&title=OWASP+Top+Ten,200,42859,17,1,8,1,2,6.5,0,0
https://owasp.org/Top10/,200,29615,47,1,9,1,0,0,0,0,0

Week-4

Cross-Site Scripting (XSS) Testing

Problem

XSS allows attackers to execute malicious scripts in user browsers.

Approach

- JavaScript payload injection
- DOM & response inspection

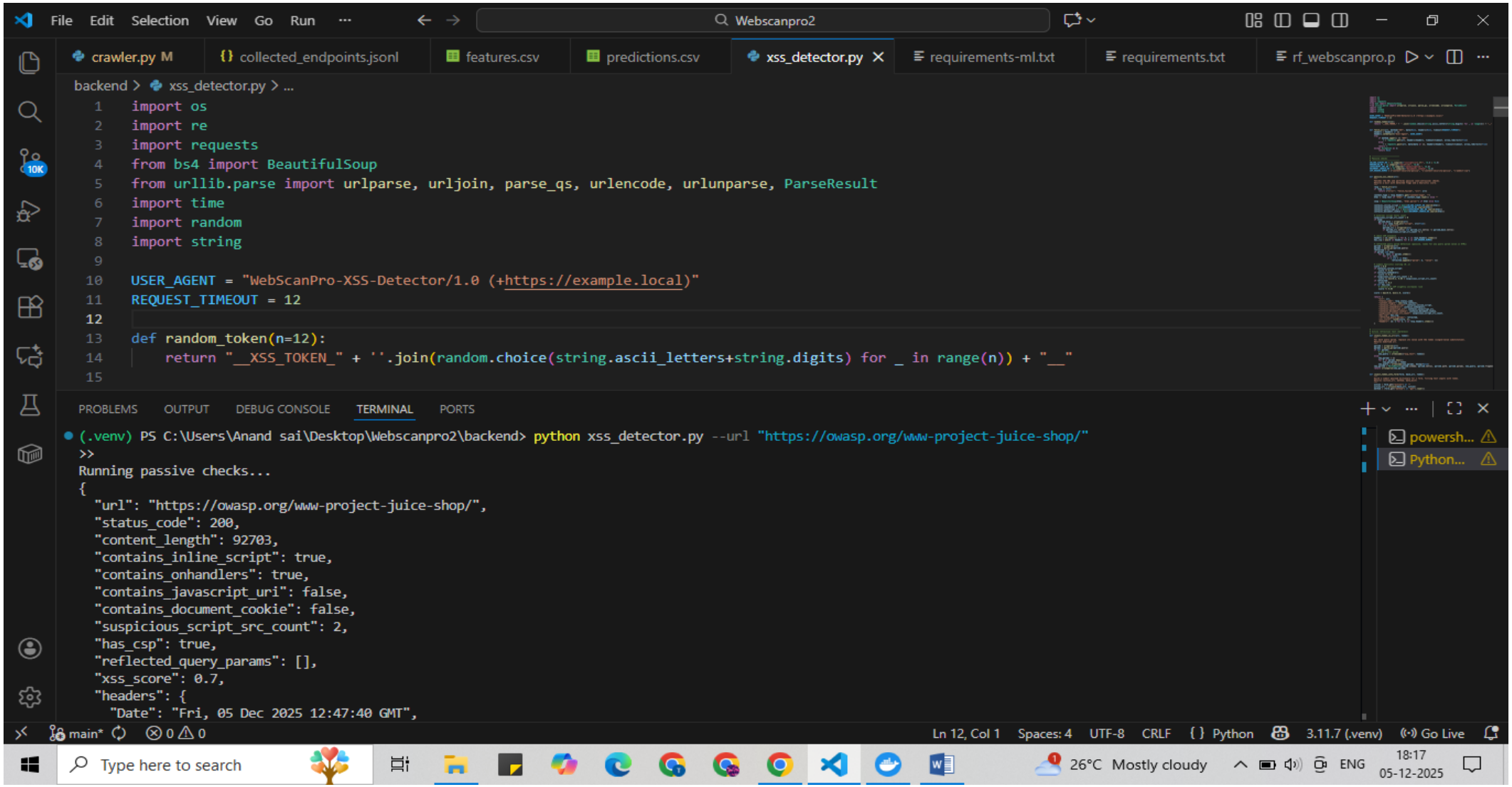
Solution Implemented

- Injected XSS payloads (<script>alert()</script>)
- Detected reflected & stored XSS

Output

- Vulnerable input fields identified
- Mitigation tips (Input sanitization, CSP)

Output:


File Edit Selection View Go Run
Webscanpro2
crawler.py M
collected_endpoints.jsonl
features.csv
predictions.csv
xss_detector.py
requirements-ml.txt
requirements.txt
rf_webscanpro.p
backend > xss_detector.py > ...
import os
import re
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse, urljoin, parse_qs, urlencode, urlunparse, ParseResult
import time
import random
import string
USER_AGENT = "WebScanPro-XSS-Detector/1.0 (+https://example.local)"
REQUEST_TIMEOUT = 12
def random_token(n=12):
return "__XSS_TOKEN_" + ''.join(random.choice(string.ascii_letters+string.digits) for _ in range(n)) + "__"
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(.venv) PS C:\Users\Anand sai\Desktop\Webscanpro2\backend> python xss_detector.py --url "https://owasp.org/www-project-juice-shop/"
>>
Running passive checks...
{
"url": "https://owasp.org/www-project-juice-shop/",
"status_code": 200,
"content_length": 92703,
"contains_inline_script": true,
"contains_onhandlers": true,
"contains_javascript_uri": false,
"contains_document_cookie": false,
"suspicious_script_src_count": 2,
"has_csp": true,
"reflected_query_params": [],
"xss_score": 0.7,
"headers": {
"Date": "Fri, 05 Dec 2025 12:47:40 GMT",
powersh...
Python...
main*
Ln 12, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.11.7 (.venv)
Go Live
Type here to search
26°C Mostly cloudy
ENG
18:17
05-12-2025

Week-5

Authentication & Session Testing

Problem

Weak authentication leads to account compromise.

Approach

Credential testing

Session analysis

Solution Implemented

Tested weak/default credentials

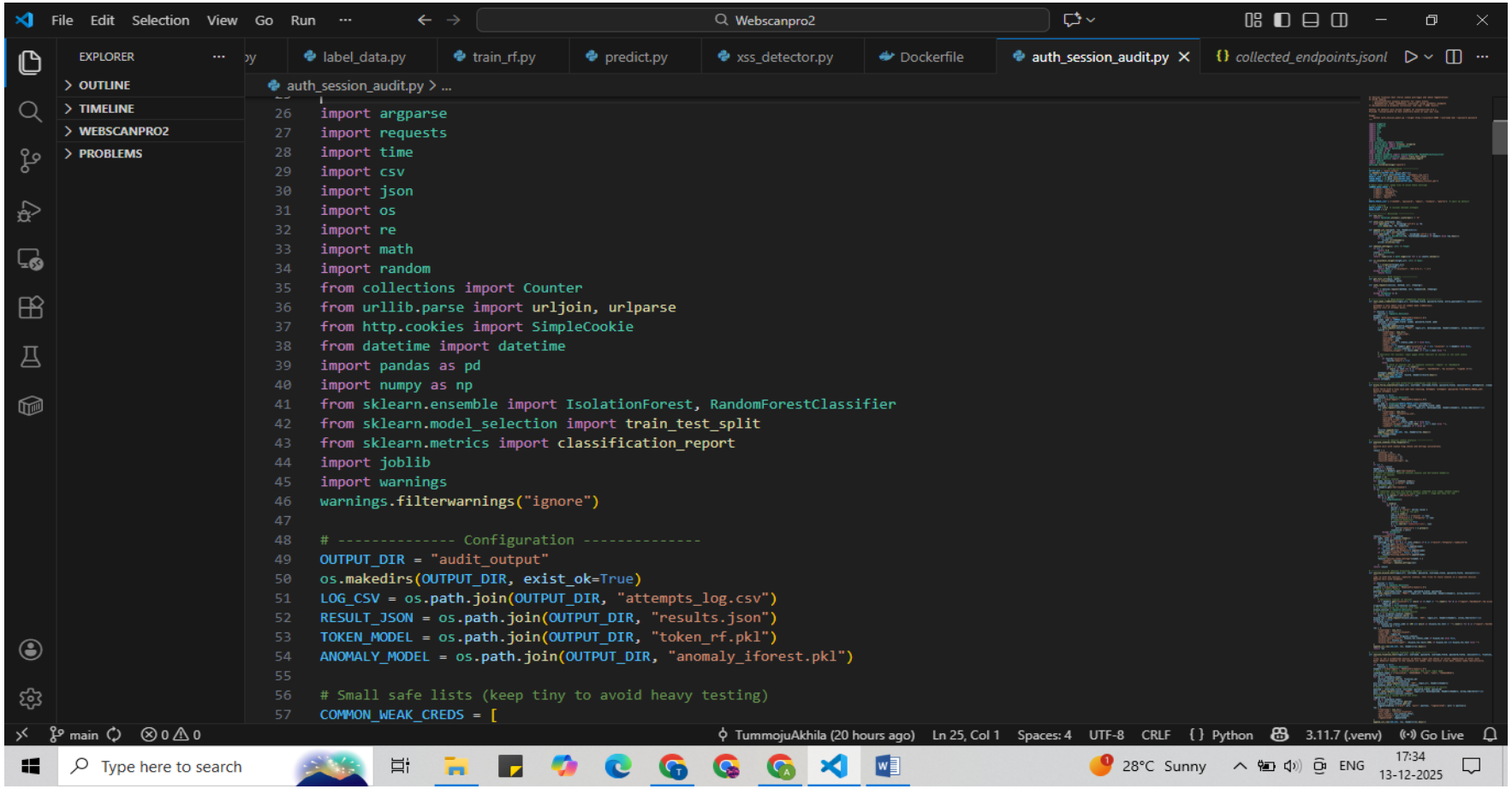
Checked cookies & session fixation

Simulated brute-force attempts

Output:

Authentication flaws

Secure session recommendations



Week-6

Access Control & IDOR Testing

Problem

- Unauthorized users accessing restricted resources.

Approach

- Parameter manipulation
- Role-based testing

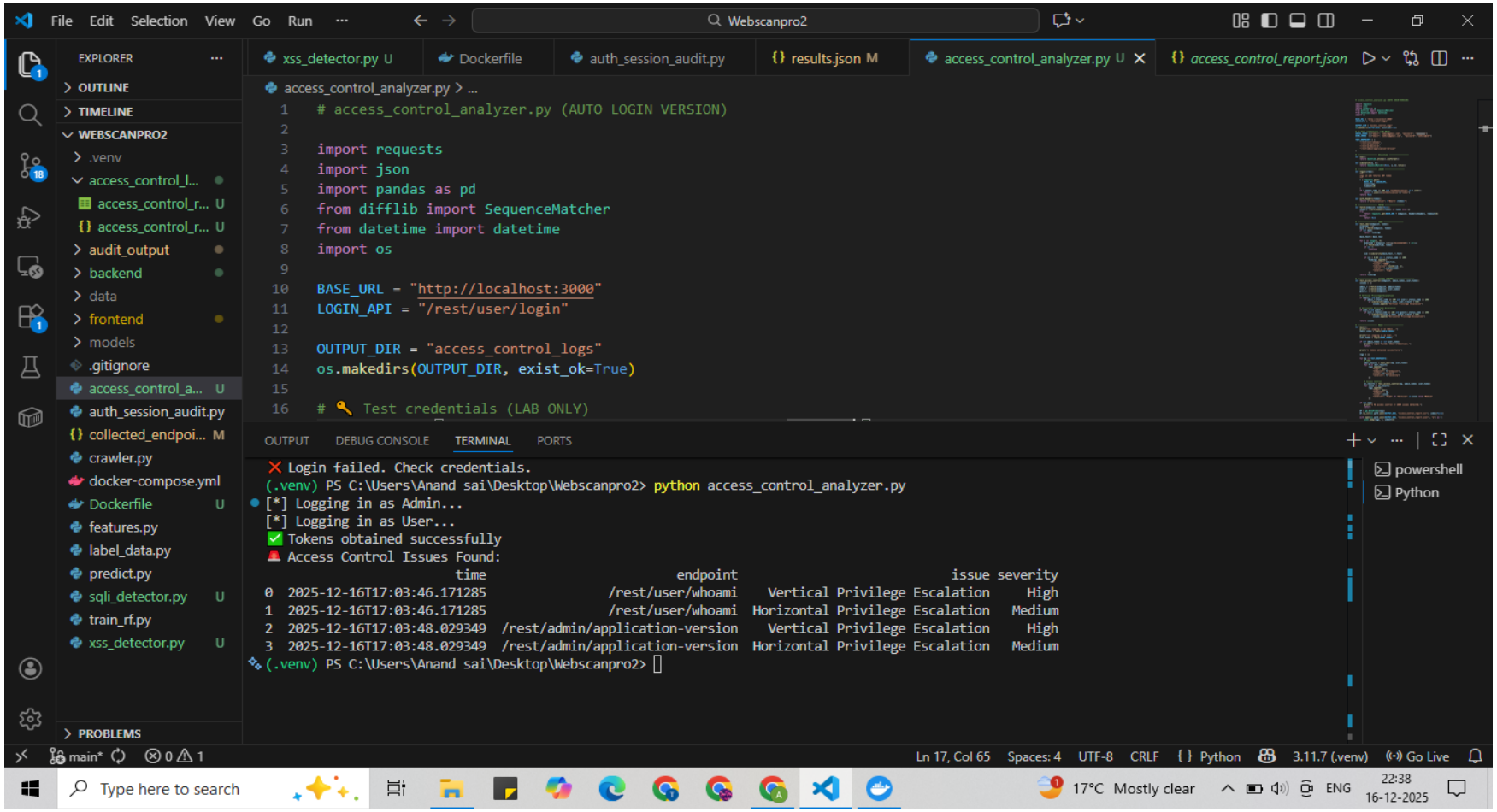
Solution Implemented

- Modified user IDs in URLs
- Tested horizontal & vertical privilege escalation

Output

- IDOR vulnerabilities detected
- Access control fixes (RBAC, ABAC)

Output:


File Edit Selection View Go Run
Webscanpro2
EXPLORER
OUTLINE
TIMELINE
WEBSCANPRO2
.venv
access_control_l...
access_control_r... U
access_control_r... U
audit_output
backend
data
frontend
models
.gitignore
access_control_a... U
auth_session_audit.py
collected_endpoi... M
crawler.py
docker-compose.yml
Dockerfile U
features.py
label_data.py
predict.py
sqli_detector.py U
train_rf.py
xss_detector.py U
PROBLEMS
xss_detector.py U
Dockerfile
auth_session_audit.py
results.json M
access_control_analyzer.py U
access_control_report.json
access_control_analyzer.py > ...
1 # access_control_analyzer.py (AUTO LOGIN VERSION)
2
3 import requests
4 import json
5 import pandas as pd
6 from difflib import SequenceMatcher
7 from datetime import datetime
8 import os
9
10 BASE_URL = "http://localhost:3000"
11 LOGIN_API = "/rest/user/login"
12
13 OUTPUT_DIR = "access_control_logs"
14 os.makedirs(OUTPUT_DIR, exist_ok=True)
15
16 # 🔧 Test credentials (LAB ONLY)
OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python
❌ Login failed. Check credentials.
(.venv) PS C:\Users\Anand sai\Desktop\Webscanpro2> python access_control_analyzer.py
[*] Logging in as Admin...
[*] Logging in as User...
✅ Tokens obtained successfully
🚨 Access Control Issues Found:
time endpoint issue severity
0 2025-12-16T17:03:46.171285 /rest/user/whoami Vertical Privilege Escalation High
1 2025-12-16T17:03:46.171285 /rest/user/whoami Horizontal Privilege Escalation Medium
2 2025-12-16T17:03:48.029349 /rest/admin/application-version Vertical Privilege Escalation High
3 2025-12-16T17:03:48.029349 /rest/admin/application-version Horizontal Privilege Escalation Medium
(.venv) PS C:\Users\Anand sai\Desktop\Webscanpro2>
main* 0 1
Ln 17, Col 65 Spaces: 4 UTF-8 CRLF { } Python 3.11.7 (.venv) Go Live
Type here to search
17°C Mostly clear
ENG
22:38
16-12-2025

Week-7

Security Report Generation

Problem

- Raw vulnerability data is hard to understand.

Approach

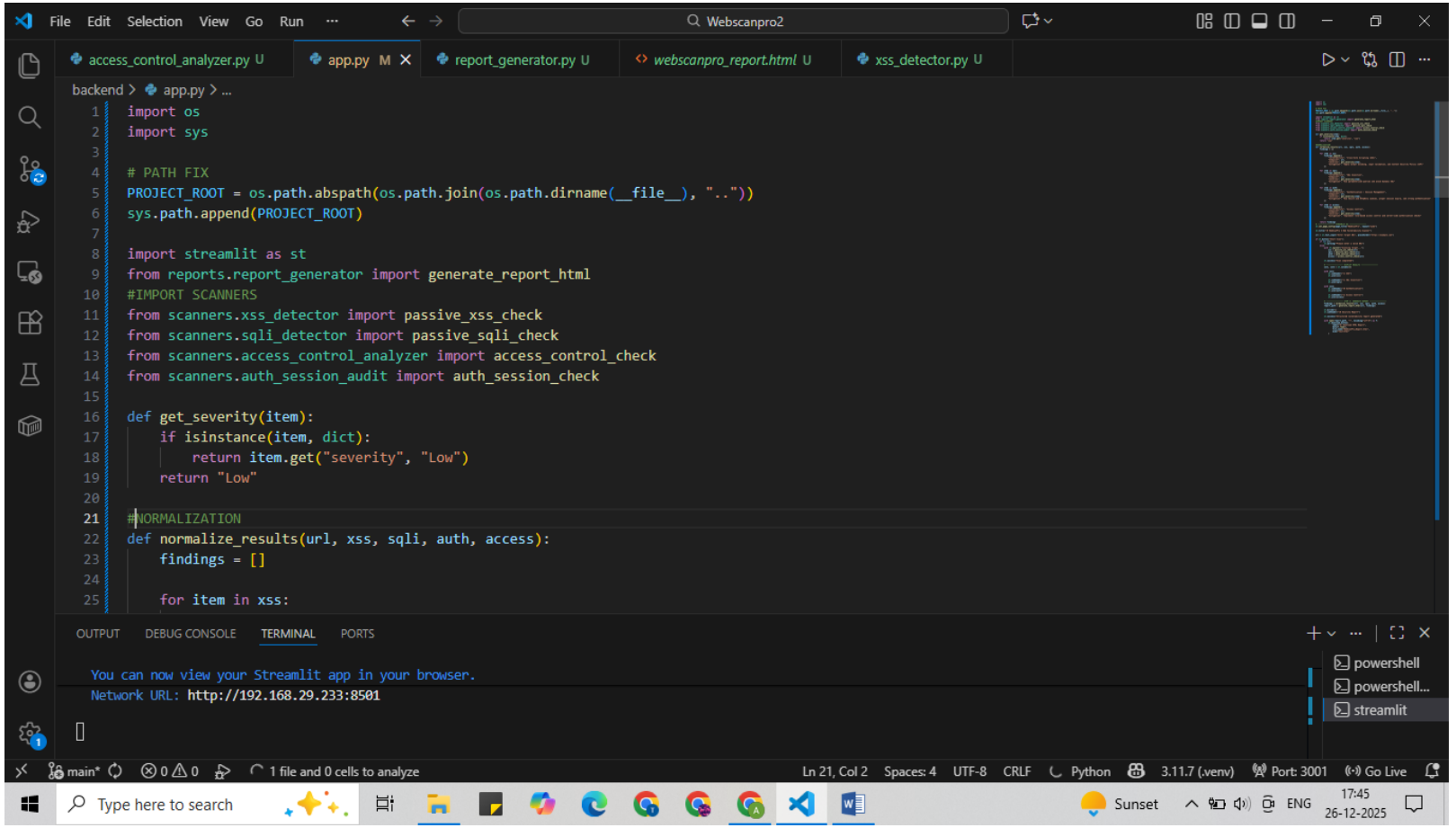
- Structured reporting
- Severity classification

Solution Implemented

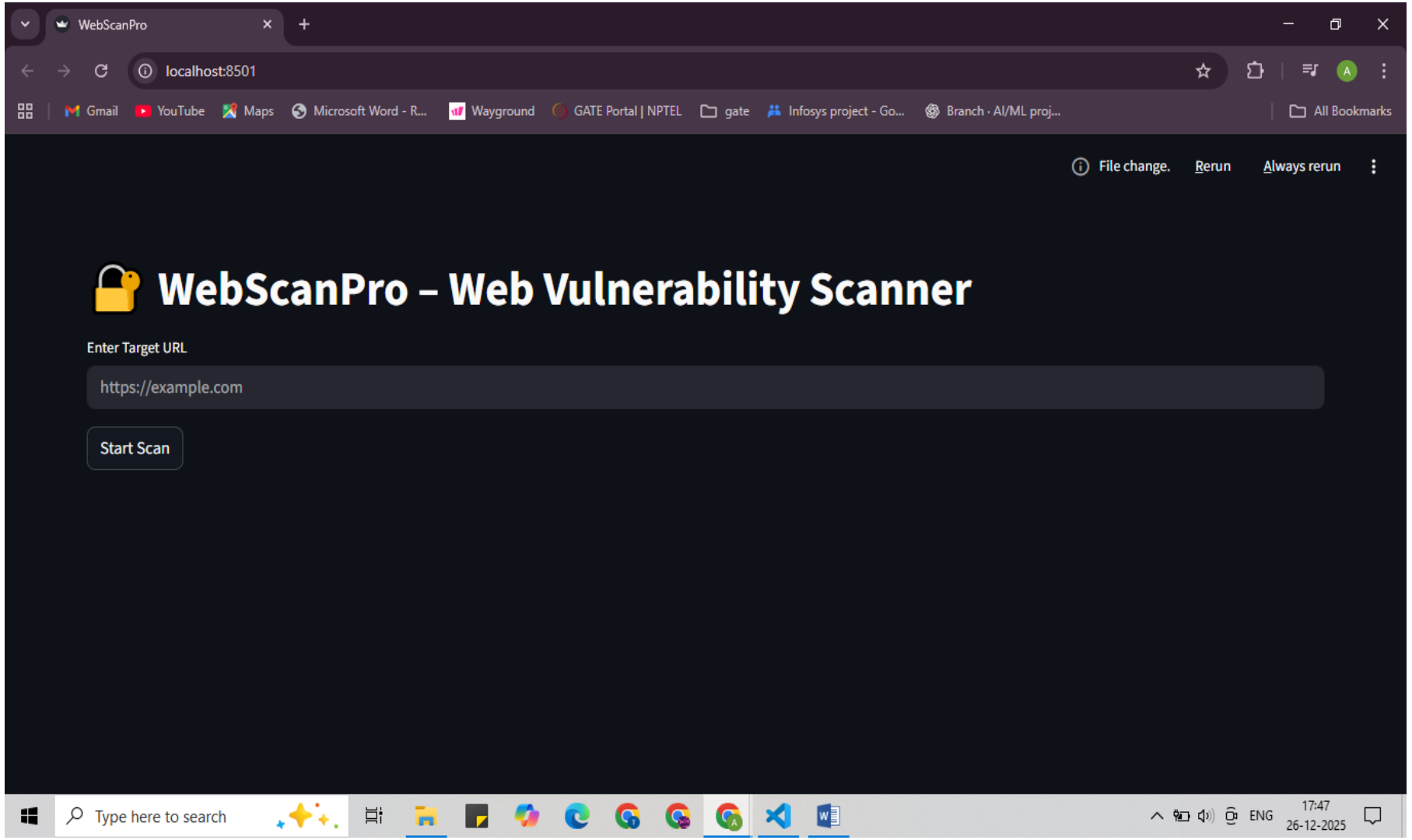
- Generated report with:
- Vulnerability Type
- Affected Endpoint
- Severity (Low / Medium / High)
- Mitigation

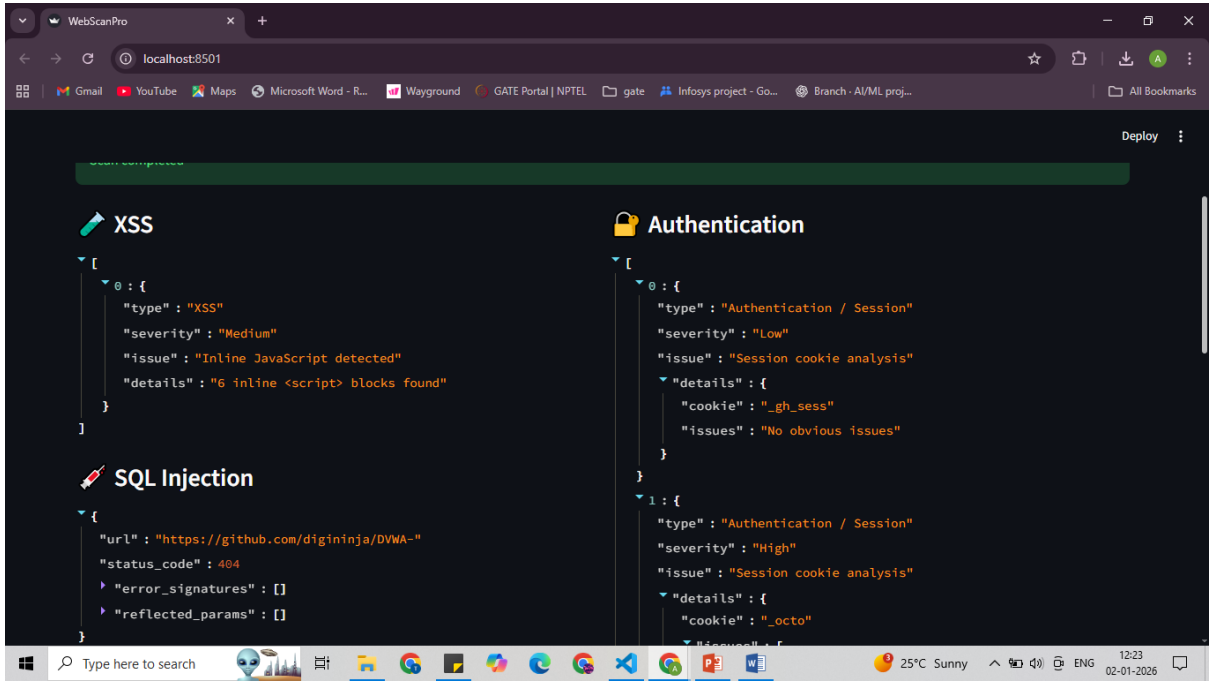
Output

- HTML / PDF security report
- Visual summaries



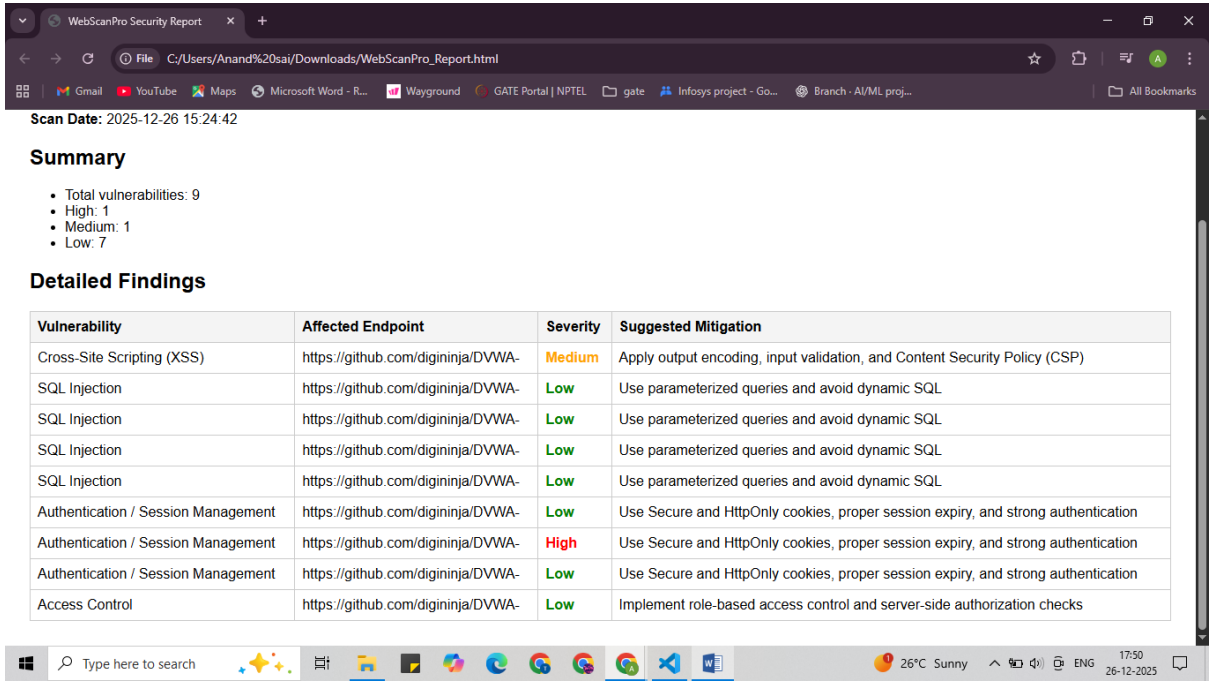
Final Output:





Vulnerabilites detected

Generated Report



THANK YOU