Intelligent XSS Analyzer

Presented by

- Sakshi Khalate – 25
- Layba Khan – 26
- Parshwa Mehta– 27
- Atharva Mohite– 28

Under the guidance
of
Dr. Jayasudha Koti

Department of Electronics and Telecommunication Engineering
St. Francis Institute of Technology, Mumbai

Fun Facts

1. XSS is one of the top three most commonly detected vulnerabilities, still how many users identify it ?

2. A study by **Palo Alto Networks** in 2025 saw that on analyzing 750,000 applications 65% had at least one XSS vulnerability, so what do you think how safe is your data on various browsers and applications ?

3. A report by **XBOW** in 2025 found that only 25% of XSS vulnerability detected is mitigated , the rest are ignored as users don’t have enough resources to mitigate it.

4. According to a report by **contrast security** in 2020 nearly half i.e (46%) of XSS attacks had a 6% success rate of mitigating XSS.

Contents

Introduction

This project is about building a simple tool to help developers and technical users find and fix **Reflected XSS i.e (Reflected Cross Site Scripting)** in web applications. The application runs a scan of the URL of any website and finds out if the vulnerability is present or not. If yes then it provides the necessary steps to mitigate the vulnerability.

Motivation

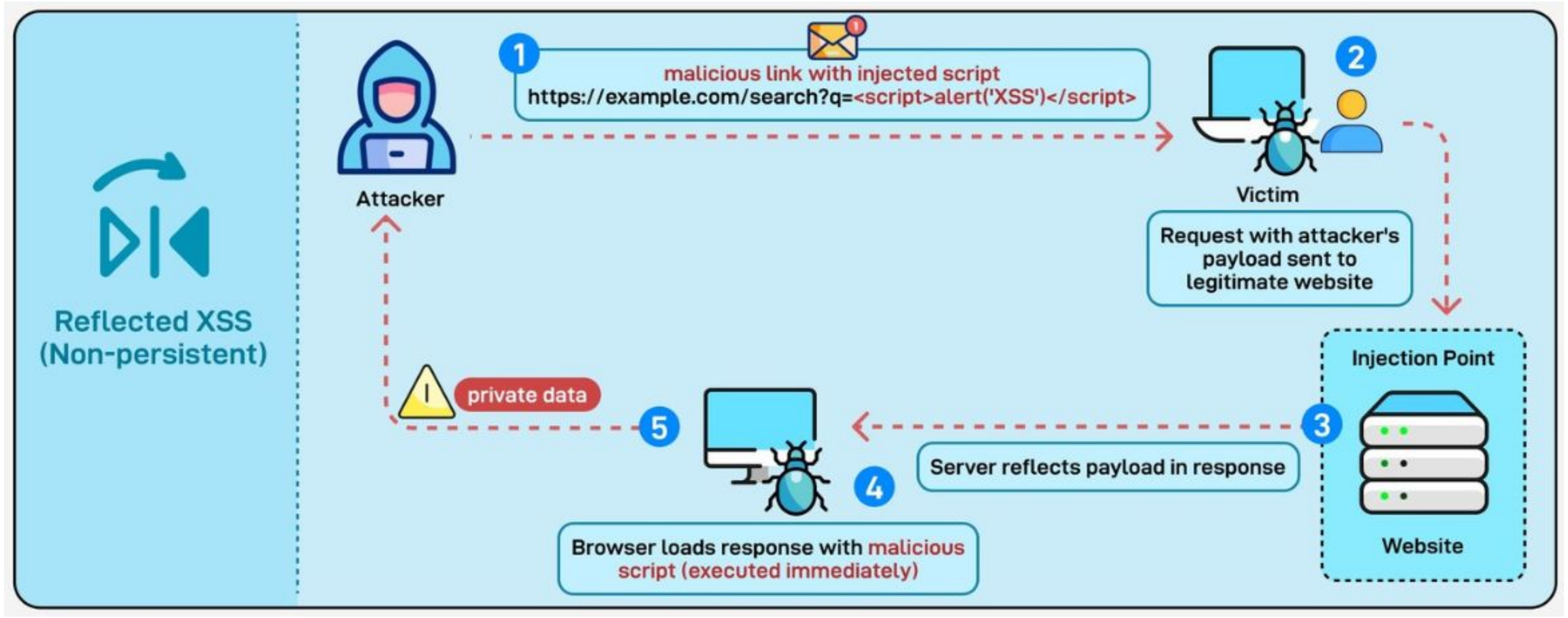
The main reason we are making this is because most security reports are very technical and hard to understand. This makes it difficult for users to fix security issues quickly and correctly, leaving their websites at risk. We want to make security easier for the users (Developers, DevOps and people who work for SMEs) whose data is very important and confidential.

Problem Statement

Existing XSS scanners detect flaws but fail at remediation, offering cryptic reports that create a time-consuming "mitigation gap" and leave applications exposed.

Reflected XSS : Vulnerability Insights

- Reflected XSS is the simplest variety of cross-site scripting. It arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.
- A web vulnerability where malicious scripts are **reflected** off a web application to a victim's browser.
- The attack is delivered through a link (e.g., in a phishing email or message).
- It is **non-persistent**, meaning the malicious script is **not** permanently stored on the server.


Reflected XSS (Non-persistent)
Attacker
1
malicious link with injected script
https://example.com/search?q=<script>alert('XSS')</script>
2
Victim
Request with attacker's payload sent to legitimate website
Injection Point
3
Website
Server reflects payload in response
4
Browser loads response with malicious script (executed immediately)
5
private data

Objectives

1.**To Develop Custom Scanners**: To build a custom Python-based scanning engine from scratch, capable of Identifying **Reflected XSS** vulnerabilities by crawling web pages, injecting safe payloads into forms, and checking for reflection.

2.**To Build a Robust Backend System:** To create a complete backend application using Python (Flask) that handles user registration/login, orchestrates the different scan modules, and saves scan results and user data to a PostgreSQL database.

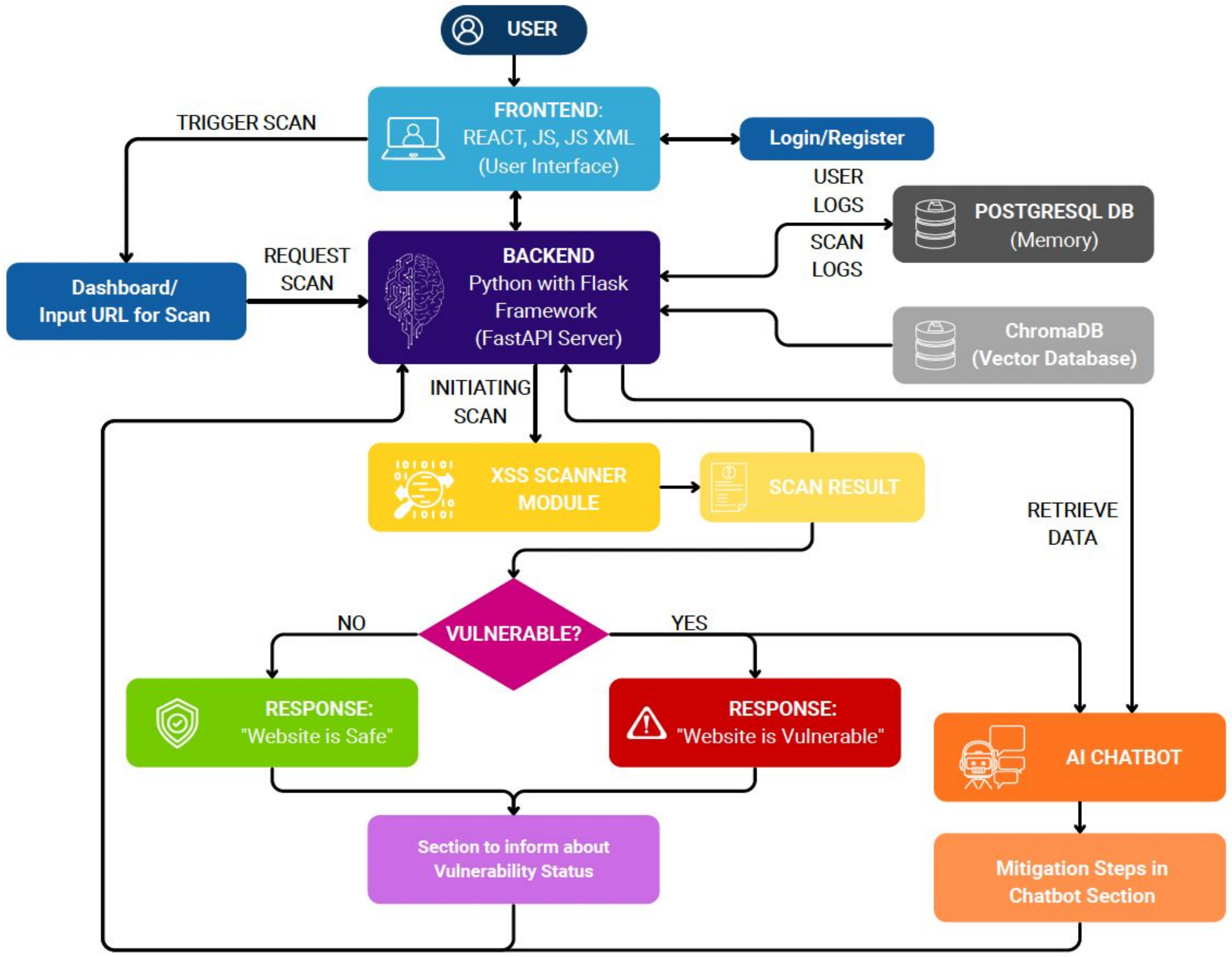
3.**To Integrate an Intelligent AI Advisor:** To implement an AI chatbot using Phi-3 LLM combined with a Retrieval-Augmented Generation (RAG) system. This AI will use your custom knowledge base to provide users with clear, expert, step-by-step mitigation advice tailored to the vulnerabilities found.

Objectives

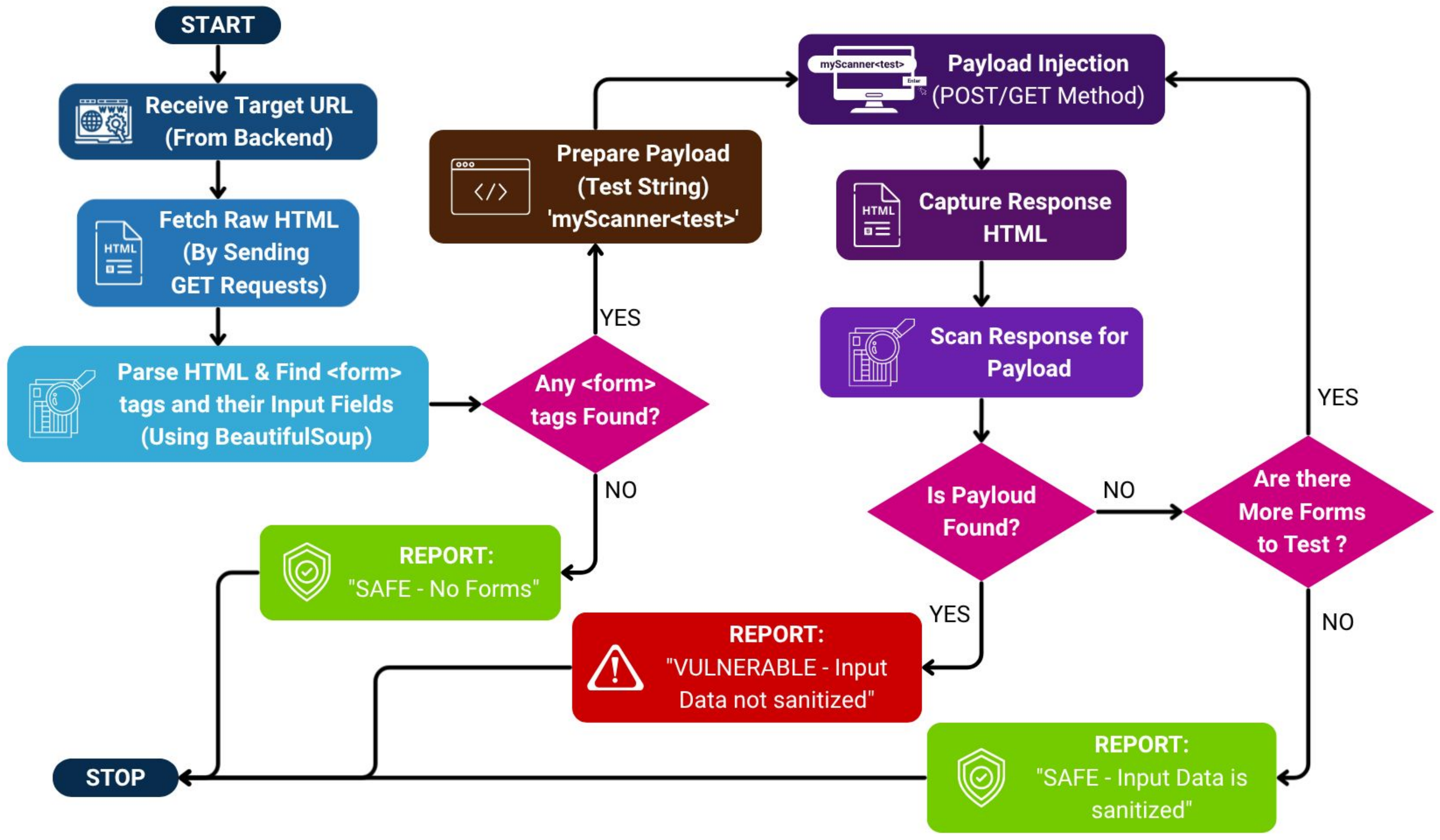
4.To Create a User-Friendly Interface: To design and build an intuitive frontend with React.js that allows users to easily submit their website URLs or dependency files, view their scan reports on a clear dashboard, and interact with the AI advisor in a conversational chat window.

5.To Bridge the "Usability Gap": To effectively translate complex, technical vulnerability data into simple, actionable, and educational advice, empowering both non-technical users and developers to understand and fix security flaws.

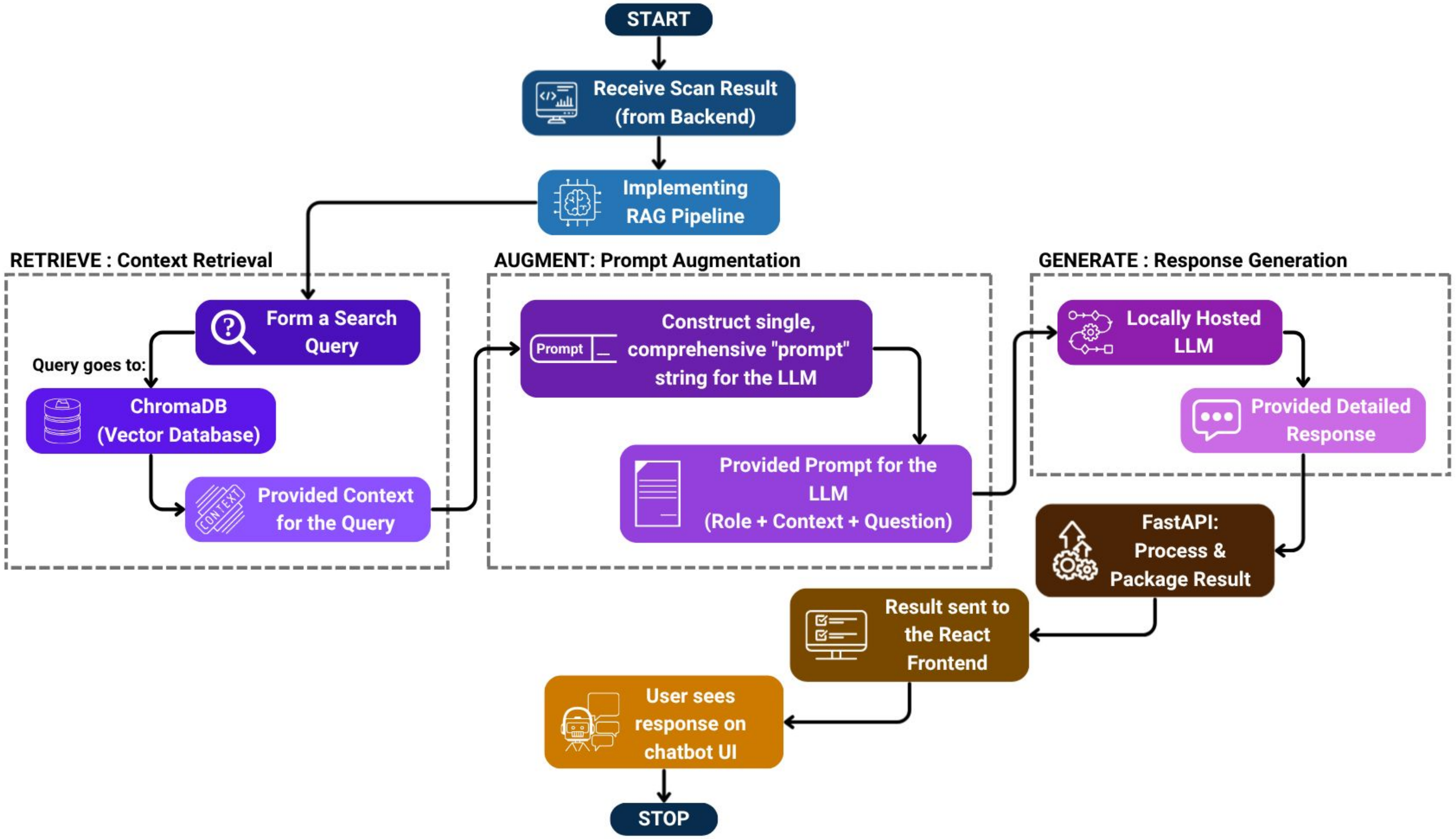
Architecture Block Diagram



Working of Vulnerability Scanner


START
Receive Target URL (From Backend)
Fetch Raw HTML (By Sending GET Requests)
Parse HTML & Find <form> tags and their Input Fields (Using BeautifulSoup)
Any <form> tags Found?
YES
NO
Prepare Payload (Test String) 'myScanner<test>'
myScanner<test>
Enter
Payload Injection (POST/GET Method)
HTML
Capture Response HTML
Scan Response for Payload
Is Payloud Found?
YES
NO
Are there More Forms to Test ?
YES
NO
REPORT: "SAFE - No Forms"
REPORT: "VULNERABLE - Input Data not sanitized"
REPORT: "SAFE - Input Data is sanitized"
STOP

Working of AI Chatbot


START
Receive Scan Result (from Backend)
Implementing RAG Pipeline
RETRIEVE : Context Retrieval
Form a Search Query
Query goes to:
ChromaDB (Vector Database)
Provided Context for the Query
CONTEXT
AUGMENT: Prompt Augmentation
Prompt
Construct single, comprehensive "prompt" string for the LLM
Provided Prompt for the LLM (Role + Context + Question)
GENERATE : Response Generation
Locally Hosted LLM
Provided Detailed Response
FastAPI: Process & Package Result
Result sent to the React Frontend
User sees response on chatbot UI
STOP

Software Requirements

- **Backend Framework**: Python with FastAPI for high-performance API development and LLM integration.
- **Machine Learning Libraries**: Transformers (Hugging Face) for loading and managing the pre-trained LLM.
- **Natural Language Processing**: ChromaDB for vector storage and retrieval, and generating response .This stack will be used to build a custom RAG pipeline.
- **Self-Hosted LLM**: Phi-3-mini as the locally-run generative model for providing mitigation advice.
- **Frontend Development**: React.js (using Vite) for the client-side user interface, managing state, and handling API calls.
- **Database Systems**: PostgreSQL for user accounts, scan history logs and ChromaDB as the dedicated vector database for the AI's knowledge base.
- **Scanning Engine**: Custom-built scanner using Python libraries requests (for HTTP communication) and BeautifulSoup (for parsing HTML and XML).
- **Development Environment**: VS Code for coding and Github for version control and repository management.
-

Requirement of Security in Applications

- **To protect your users private data**, like personal info and financial details, that can be exploited.
- **To build and keep customer trust**, because people won't use a service they think is unsafe or unprotected.
- **To follow laws and regulations** (like GDPR) and avoid getting large fines.
- **To keep your business running**, as an attack can shut down your website and stop operations leading to big losses.
- **To defend against new and smarter attacks** that are created by cybercriminals every day and protect our data.

Why 'Intelligent XSS Analyzer'?

1. **We Don't Just Give You Data, We Give Answers**
   Instead of complex, technical reports, our app provides clear, easy-to-understand explanations in a simple conversation.
2. **We Focus on the Solution, Not Just the Problem**
   Other tools just tell you *what's* wrong. We use an AI chatbot to show you *how* to fix it with simple, step-by-step guidance.
3. **We Provide Expert, Tailored Advice**
   Our AI (using Google Gemini + RAG) is trained with our own expert knowledge. It gives you a specific solution for *your* exact problem, not just a generic tip.
4. **We Are Built for Developers & Non-Experts**
   We designed this tool for the person who has to *actually write the fix*, making security less scary and accessible to everyone, not just security professionals.

Sample prompts by the user to the Chatbot

- "Can you explain that in simpler terms?"
- "What is 'output encoding'?"
- "How do I fix this in a XML/HTML file?"
- "Can you show me a code example for Python?"
- "What's the difference between Reflected XSS,Stored XSS and DOM XSS ?"

Target Audience

1.Developers:-

The primary users who are responsible for *fixing* code. Our app provides clear, step-by-step mitigation advice directly, rather than just a complex error report.

2.DevOps & SREs:-

Professionals managing the CI/CD pipeline. They need a fast, automated, and clear-cut tool to integrate security checks without slowing down deployment.

3.Technical Users (QA, IT Admins):-

Users who test applications or manage infrastructure. Our tool allows them to quickly find and report vulnerabilities with clear, understandable explanations.

4.Small & Medium Enterprises (SMEs):-

Businesses that need robust web security but lack the budget for a dedicated, full-time security team or expensive enterprise software.

Conclusion

This project successfully demonstrates a complete and modern solution for developer-centric security. We have built a web application that bridges the critical gap between vulnerability DETECTION and SOLUTION.

Our custom-built scanners for XSS effectively identify high-risk, common flaws. More importantly, the integration of a chatbot transforms those technical findings into actionable, easy-to-understand solutions.

This project proves that by combining smart, targeted scanning with intelligent AI, we can empower developers of all skill levels to not only find security flaws but to confidently fix them, making security an accessible and manageable part of the development lifecycle.

Literature Review

Sr. No.	Title of the Paper	Descriptions	Results	Limitations
1.	**XSS Vulnerability Scanner**[1]	This project, titled "XSS Vulnerability Scanner," is an automated tool designed to detect Cross-Site Scripting (XSS) vulnerabilities in web applications. The scanner works by taking a website link from the user , crawling the site , and then injecting JavaScript code with special HTML.	The authors state that their XSS scanner **"has performed much better than other web risk scanners"** in terms of accuracy and false-positive standards. They also report that in terms of the *number* of vulnerabilities detected, their tool was **"as successful as"** competing structures. The report attributes this improved performance to the use of the fuzzy inference system.	**Context-Blind:** The scanner's logic relies on only seven DOM properties , so it's likely blind to the *context* of a reflection (e.g., in an HTML attribute) and would miss it. **Simple Payloads:** The tool only tests "special HTML characters". This simple method would fail against filters that are vulnerable to encoded or complex bypasses. **Limited Vectors:** Its focus on DOM features means it would miss XSS injected via HTTP headers or reflected in non-HTML content like JSON.
2.	**CROSSCAN: REFLECTED XSS SCANNER WEBSITE TO SCAN FOR REFLECTED XSS VULNERABILITY** [2]	This project, titled "CROSSCAN," is an automated scanner designed specifically to find **Reflected Cross-Site Scripting (XSS) vulnerabilities** in web applications. The tool is intended for developers and security professionals to streamline vulnerability detection.	The scanner was tested on various web applications and showed strong performance in several areas such as detection rate,accuracy and usability.	**Fails on modern apps:** The scanner struggles with advanced JavaScript and Single-Page Applications (SPAs). **Misses new attacks:** It relies on a fixed list of known payloads, so it can miss new or sophisticated XSS patterns. **Needs client-side improvement:** The authors state it needs future work to get better at handling client-side vulnerabilities.

Literature Review

Sr. No.	Title of the Paper	Descriptions	Results	Limitations
3.	**AI-Based Web Vulnerability Scanner: A Comprehensive Review [3]**	The document reviews AI-powered web vulnerability scanners that improve accuracy by analyzing web traffic and code patterns to detect threats like SQL injection and XSS. It highlights a scorecard system that prioritizes vulnerabilities by risk, enabling efficient remediation.	The conclusion highlights that AI-enhanced vulnerability scanners significantly improve web security by increasing detection accuracy, reducing false positives, and prioritizing risks effectively through a scorecard system. This approach enables faster, focused remediation of critical threats and adapts continuously to emerging vulnerabilities.	**Lack of Labeled Data:** A scarcity of high-quality training data can reduce the AI's accuracy and lead to "overfitting." **Handling Novel Attacks:** The AI struggles to detect new, unknown attack vectors (zero-day threats) until they are added to its training set. **Model Interpretability:** The AI is a "black box," making it difficult to understand *how* it reaches a decision, which is a problem for teams who need to trust and verify its findings.
4.	**An Automatic Vulnerability Scanner for Web Applications with Firewall Techniques [4]**	Key features of a web vulnerability scanner include automated asset discovery, comprehensive vulnerability detection, risk-based prioritization, and detailed actionable reports.	The paper concludes that combining automated vulnerability scanning tools with manual testing provides the most effective cybersecurity assessment. The developed scanner is flexible and customizable, capable of detecting various web and network threats.	**Requires Manual Testing:** The tool supplements, but does not replace, manual expert testing. **Dependent on Updates:** Effectiveness relies on constant updates to its vulnerability database. **Finds Only Known Flaws:** It is signature-based, so it cannot detect new or zero-day exploits.

References

1. AI in Cybersecurity: Latest Developments + How It's Used in 2025 - https://secureframe.com/blog/ai-in-cybersecurity
2. Vulnerabilities of Web Applications: Good Practices and New Trends - https://www.acigjournal.com/Vulnerabilities-of-Web-Applications-Good-Practices-and-New-Trends,199521,0,2.html
3. Web Application Scanning & API Security - https://www.qualys.com/apps/web-app-scanning/
4. Study of Vulnerability Scanners for Detecting SQL Injection and XSS Attack in Websites - https://ojs.bonviewpress.com/index.php/AIA/article/view/754/489
5. AI in Cybersecurity: Key Benefits, Defense Strategies, & Future Trends - https://www.fortinet.com/resources/cyberglossary/artificial-intelligence-in-cybersecurity
6. CROSSCAN: REFLECTED XSS SCANNER WEBSITE TO SCAN FOR REFLECTED XSS VULNERABILITIES - https://www.journalijar.com/uploads/2025/05/68344e5066f45_IJAR-51522.pdf
7. XSS Vulnerability Scanner - https://www.scribd.com/document/811759841/XSS-Vulnerability-Scanner
8. DAST- https://www.getastra.com/blog/dast/what-is-dast/
9. OWASP XSS -https://owasp.org/www-community/attacks/xss/