National Forensic Sciences University
Knowledge | Wisdom | Fulfilment
An Institution of National Importance
(Ministry of Home Affairs, Government of India)

PROJECT(MBACS-SIV-1) REPORT

ON

"Vuln_Hub" Submitted To

School of Management Studies,

National Forensic Sciences University

MASTER OF BUSINESS ADMINISTRATION

In

CYBER SECURITY MANAGEMENT

Submitted By Ansh

Modi

0123004000011002007

Under the Supervision of

Dr.Siddharth Dhabade

National Forensic Sciences University, Gandhinagar Campus,

Gandhinagar – 382009, Gujarat, India.

ACKNOWLEDGEMENTS

I'd like to take a moment to thank most sincerely my illustrious supervisor and mentor, whose remarkable guidance, unwavering support and unsurpassed learning have been the foundation for which this project has stood. You've been so patient and insightful through this journey.

I am hugely grateful to my peers who have given their invaluable feedback, ready suggestions and collaborated with me to make the quality and depth of my work very high. I have been constantly inspired and motivated by your support.

I am extraordinarily grateful to every single person who, in any way, helped complete this project. Your involvement either through words of encouragement or technical assistance or sharing resources has been a beautiful illustration of the power of teamwork and shared vision. Without your kindness, you would not have accomplished this.

With Sincere Regards,

Ansh Modi

MBA in Cyber Security Management

TABLE OF CONTENTS

ABSTRACT

- The growing complexity of cybersecurity threats has increased the demand for automated, AI-driven vulnerability scanning tools. Traditional security solutions such as Nessus, Metasploit, and OWASP ZAP require extensive configuration and high-end computing resources, limiting their accessibility to non-experts and small organizations.
- This project, AI-Powered Cloud-Based Vulnerability Scanner, aims to bridge this gap by integrating React.js, Flask, OWASP ZAP API, and Google Gemini AI to provide real-time security scanning with automated remediation suggestions.
- The project evolved through three development phases:
- Phase 1: Command-Line Interface (CLI) scanner
- Phase 2: GUI-based vulnerability scanner with WHOIS Lookup
- Phase 3: React.js migration and AI-powered remediation
- This AI-driven system enables users to scan web applications, retrieve domain details, analyze security risks, and receive AI-generated solutions for detected vulnerabilities. The future roadmap includes AI-powered phishing detection, AI-based vulnerability scanning, automated exploitation techniques, and mobile PWA deployment.

INTRODUCTION

With the rise of cyber threats, vulnerability scanning has become a fundamental practice for securing digital assets. However, traditional security tools come with several challenges:

- **High technical expertise required** – Many tools demand deep security knowledge for setup and configuration.
- **Resource-intensive setup** – Platforms like Nessus and ZAP require dedicated operating systems like Kali Linux.
- **Manual analysis needed** – Even after scanning, users must manually interpret results Project Objective

This project aims to simplify security analysis by providing a cloud-based, AI-enhanced security scanner that is:

- **Accessible:** No OS-specific requirements; works on any browser.
- **User-Friendly:** Simple UI via React.js for easy navigation.
- **AI-Powered:** Uses Google Gemini AI to generate automated remediation steps.
- **Cloud-Based:** Eliminates hardware dependencies, making security tools accessible on any device and find remediation steps.

PROBLEM STATEMENT

- Challenges in Cybersecurity Practices:
- **Technical Complexity:**
    - Vulnerability scanning tools require substantial technical expertise to configure and use effectively.
    - CLI-based tools, though powerful, can be intimidating for users unfamiliar with command-line interfaces.
- **Resource Requirements:**
    - Tools like Nessus, Metasploit, and even OWASP ZAP often demand high computational resources and operating systems like Kali Linux.
    - Users without such resources find it difficult to perform practical vulnerability assessments.
- **Standalone Tools:**
    - WHOIS, DNS lookup, and other cybersecurity utilities are often standalone tools requiring separate installation or subscription.
    - Consolidating multiple tools into one platform is not common.
- **Solution:**
    - Automated AI-based remediation using Google Gemini AI.
    - A streamlined UI for easy security analysis.
    - Centralized security scanning, WHOIS lookup, and vulnerability assessment.

LITERATURE SURVEY

Sr. No.	Paper Title and Author(s)	Details of Publication	Findings/Outcome
1	IoT Security and Privacy: A Survey, Shojafar, M., et al.	2017 IEEE Communications Surveys & Tutorials	Discusses IoT security challenges and potential solutions.
2	Comprehensive Study on Security Issues in IoT: A Survey, Vyas, A., et al.	2018 SET Conference	Highlights the need for robust security mechanisms in IoT systems.
3	IoT Security: A Survey, Taxonomy, and Open Issues, Alaba, F. A., et al.	2017 Future Generation Computer Systems	Categorizes IoT security challenges and explores open research issues.
4	Frameworks for Cloud-Based Cybersecurity Tools, K. Lee, F. Zhao	2022 Springer Cybersecurity Advances	Proposes cloud platforms for low-resource cybersecurity practices.

Sr. No.	Paper Title and Author(s)	Details of Publication	Findings/Outcome
5	Expanding Vulnerability Scanners with Auxiliary Tools, Chen, J., Kim, H.	2018 IEEE Security Symposium	Integrates WHOIS and DNS lookups into scanning platforms for improved functionality.

COMPONENT / TOOLS USED

- **Frontend:**
  - **React JS** for creating the web interface.
  - **Tailwind CSS** for responsive design and layout.
- **Backend:**
  - **Flask** as the Python-based backend framework for handling HTTP requests and responses.
  - **Python Libraries**: whois, genai(Google Gemini API).
- **Database/Storage:**
  - **NoSQL** for storing scan history and user preferences (if necessary).
- **Tools Integrated:**
  - **ZAP** for vulnerability scanning.
  - **WHOIS** for domain information.
  - **Google Gemini API** for AI based Remediation Steps.

DIAGRAM

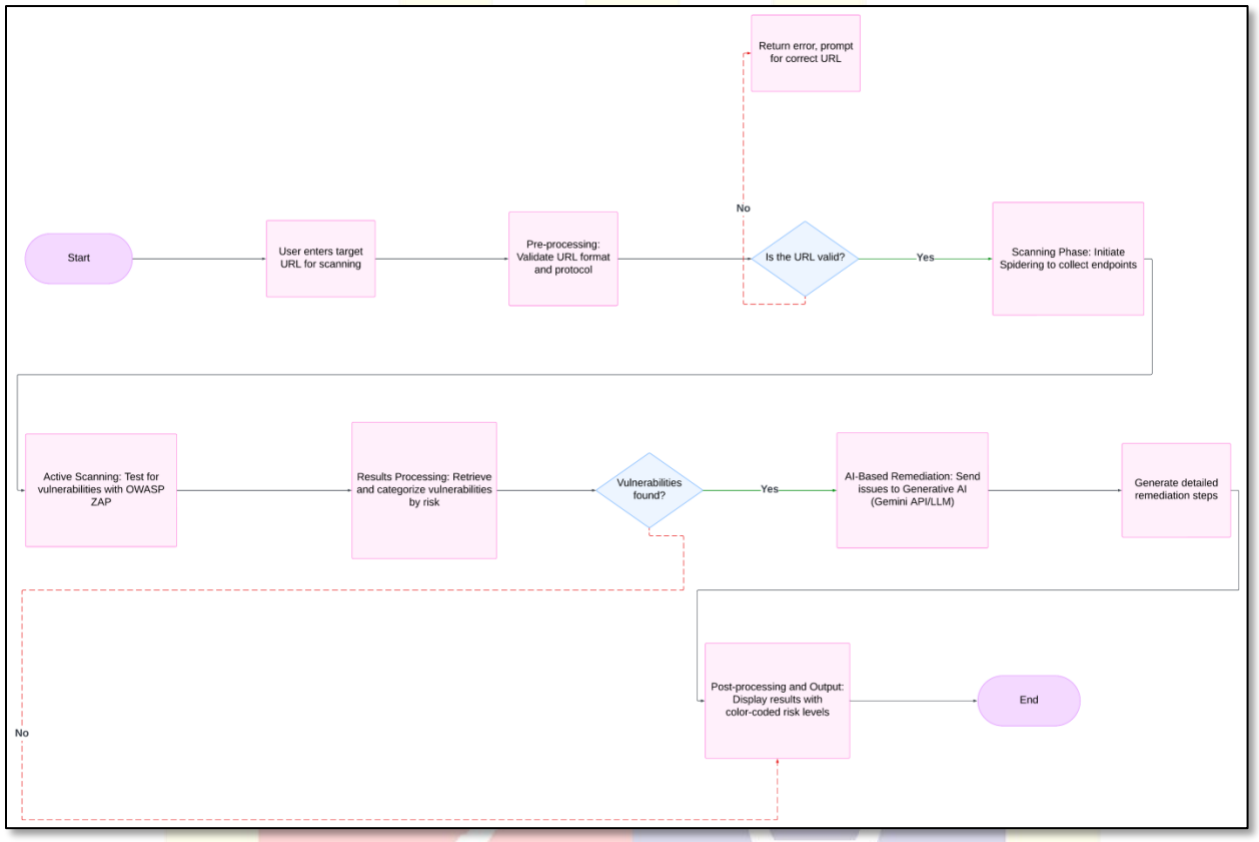

Fig 1: Flow Chart

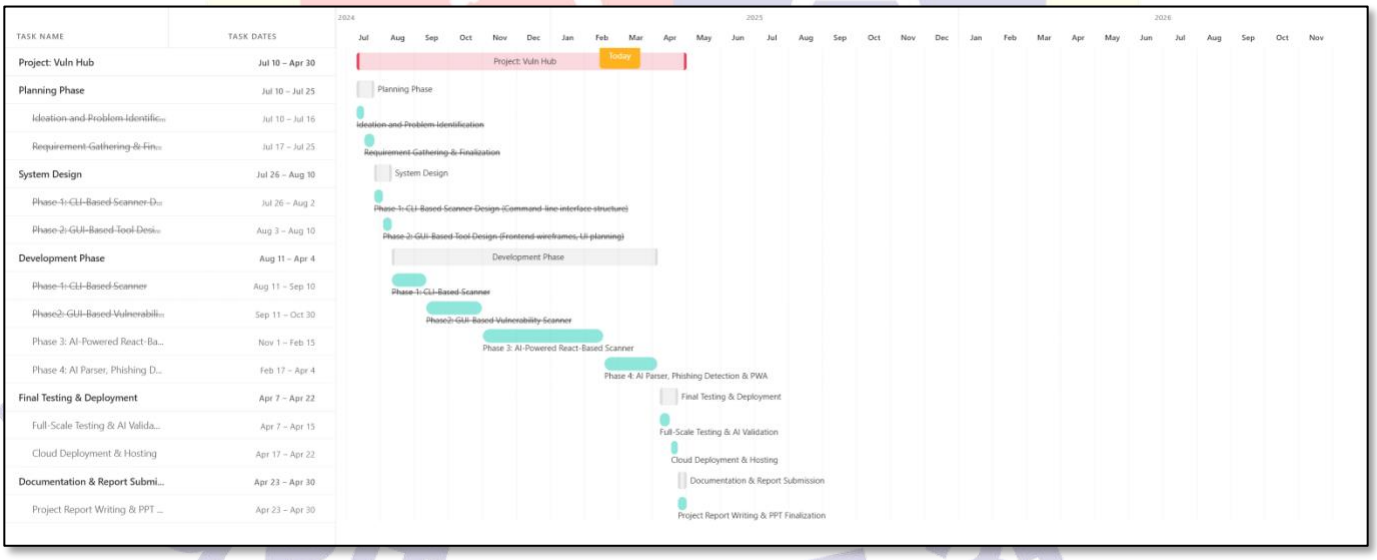

Fig 2: Gantt Chart

IMPLEMENTATION

- Phase 1: Development of the CLI-based scanner using Python.
- Phase 2: Building the user interface with HTML, CSS, and JavaScript, Flask Backend, WHOIS Integration.
- Phase 3:
    - Migration to React.JS:
        - Transitioned from Flask-Based UI to React.js for a dynamic and responsive experience.
        - Used Vite.js instead of Webpack for faster build times and improved performance.
        - Implemented React Router for seamless navigation between Scanner and WHOIS Lookup.
        - Integrated state management using Jotai to handle real-time scan results efficiently.
    - AI-Powered Remediation:
        - Integrated Google Gemini AI API to automate security fix recommendations.
        - AI dynamically analyzes detected vulnerabilities and generates customized remediation steps.
        - Implemented real-time vulnerability reporting with AI-powered fixes displayed instantly.
    - Improved UI/UX:
        - Old Retro, clean, and responsive UI built with Tailwind CSS.
        - Improved terminal output design for better readability of scan results.

CHALLENGES

- Phase 1: CLI Based Scanner
    - Limited User Accessibility
    - Manual Analysis Required
    - Resource Constraints
- Phase 2: GUI Based Scanner (HTML & Flask)
    - Complex GUI Development:
        - Transitioning from CLI to a graphical interface required additional frameworks (Flask, Bootstrap).
        - Designing a user-friendly, intuitive interface without compromising security features was challenging.
    - Handling Real-Time Scan Results:
        - Initially, results were displayed only after the scan completed, leading to poor user experience.
        - Streaming live scan results dynamically using Flask's SSE (Server-Sent Events) was complex to implement.
    - WHOIS & API Integration Issues:
        - Fetching WHOIS data required multiple APIs, and different domain registrars returned inconsistent data formats.
        - Implementing data parsing mechanisms to extract relevant information efficiently took additional effort.
- Phase 3: GUI Based, AI Integrated Version
    - Migration from Flask Frontend to React.js:
        - Flask's templating engine (Jinja2) was not scalable, requiring a complete rewrite of the UI using React.js.
        - State management issues arose when handling real-time scan
    - Integrating OWASP ZAP API with React & Flask:
        - OWASP ZAP scans are asynchronous and long-running (sometimes over 5 minutes).
        - Handling continuous updates in React without blocking UI interactions was a major technical challenge.
        - Solution: Implemented WebSockets for real-time scan progress updates, reducing delays.
    - AI-Based Remediation with Google Gemini API:

        - I try another AI API but:
            - OpenAI API (limited free tier and high pricing for large-scale queries).
            - Hugging Face API (free models lacked accuracy for specific security use cases).
        - Generating context-aware vulnerability remediation steps required prompt engineering and API optimization.
        - Some vulnerabilities lacked proper AI-generated solutions, leading to incomplete remediation suggestions.
        - Solution: Designed a structured prompt format to improve AI accuracy and ensured fall-back security guidelines when AI failed.
    - Ensuring Data Consistency in Remediation Reports:
        - Some vulnerabilities appeared multiple times due to repeated scans or similar attack vectors.
        - Solution: Implemented deduplication logic to prevent duplicate vulnerability listings.
    - Dynamic UI Issues in React.js:
        - UI sometimes froze during long scans due to improper state handling in React hooks.
        - Solution: Used React Suspense & Lazy Loading to prevent UI blocking and improve performance.
    - Handling Large-Scale Scan Outputs in React's Terminal Box:
        - Displaying thousands of lines of scan results in the terminal caused performance issues.
        - Solution: Implemented virtualized rendering to dynamically load only visible content.
- Future Challenges:
    - AI Parser for Better Remediation Reading:
        - AI-generated remediation often lacked structured readability.
        - Need to develop an AI-powered parser to format AI recommendations more effectively.
    - AI-Based Vulnerability Scanning:
        - Current scanning depends on OWASP ZAP, which has limitations.

    - Future ML-based scanning models will require extensive dataset training & validation.
- AI-Powered Exploitation & Phishing Detection:
    - Implementing AI-driven exploit simulation is complex and requires ethical constraints.
    - Phishing detection will require NLP-based analysis to recognize suspicious content in emails & URLs.
- Mobile App (PWA) Development:
    - Converting the platform into a PWA requires optimizing performance, storage handling, and offline support.
- Deployment to Cloud:
    - Finding a cost-effective or free cloud deployment for AI-powered cybersecurity tools is challenging.
    - Many cloud providers restrict AI-based security applications due to computational costs and ethical concerns.
    - Exploring self-hosted solutions or grant-based cloud access for long-term deployment.

RESULT

- Successful transition from CLI to GUI to AI-powered React.js Scanner
- Integrated WHOIS lookup for domain analysis
- AI-generated remediation steps improve security insights
- Real-time scanning feedback through a dynamic UI

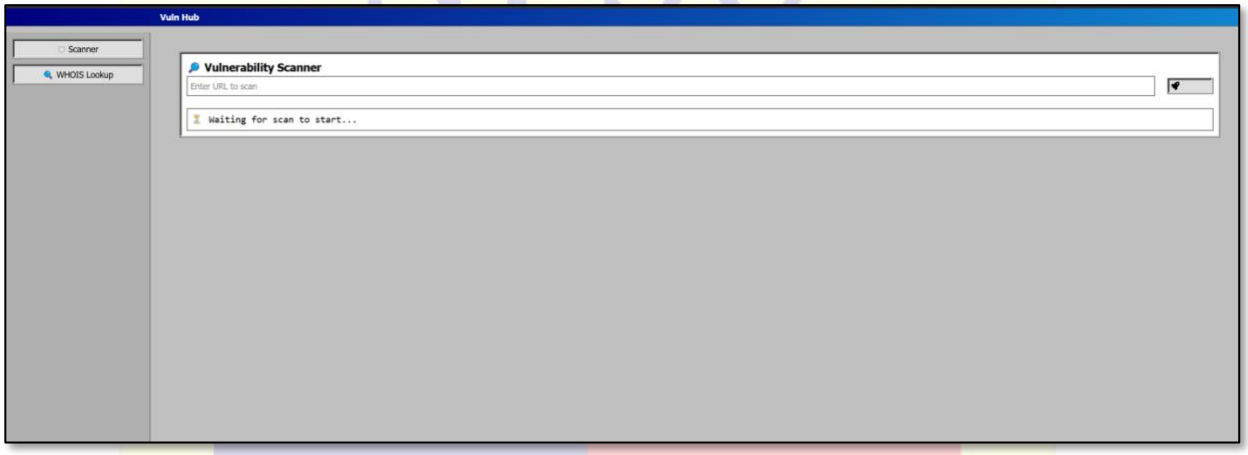

Fig 3: Home Page

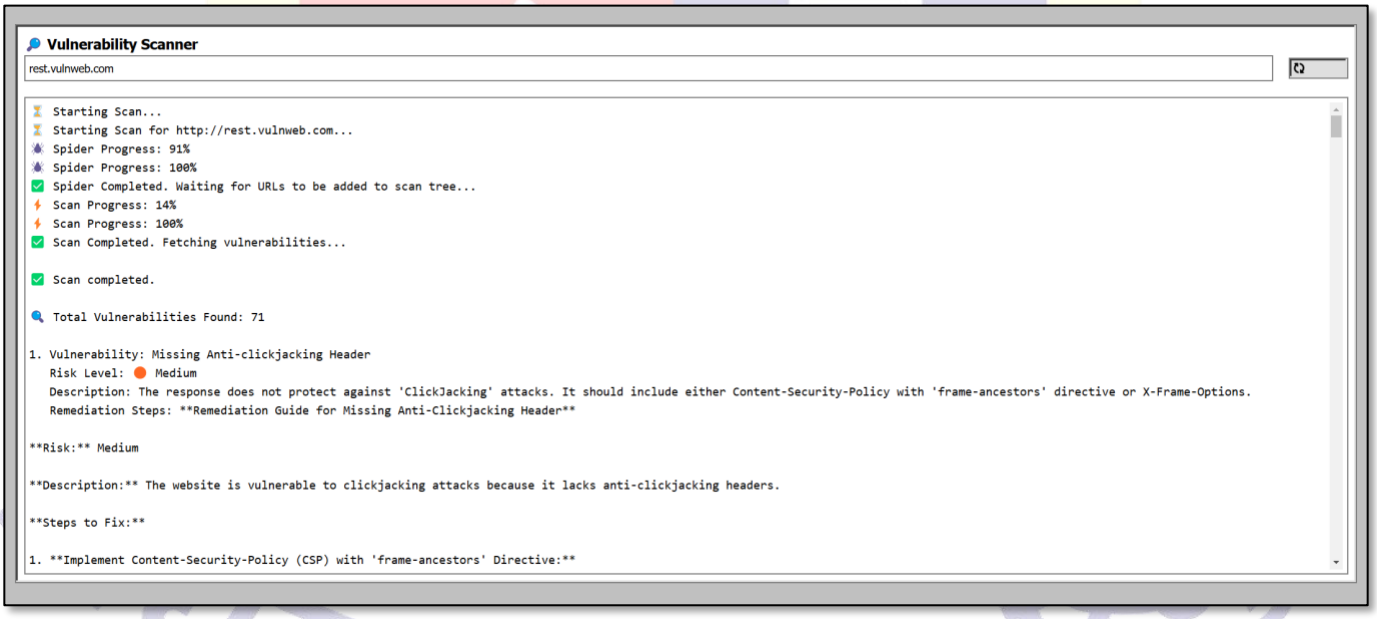

Fig 5: Output

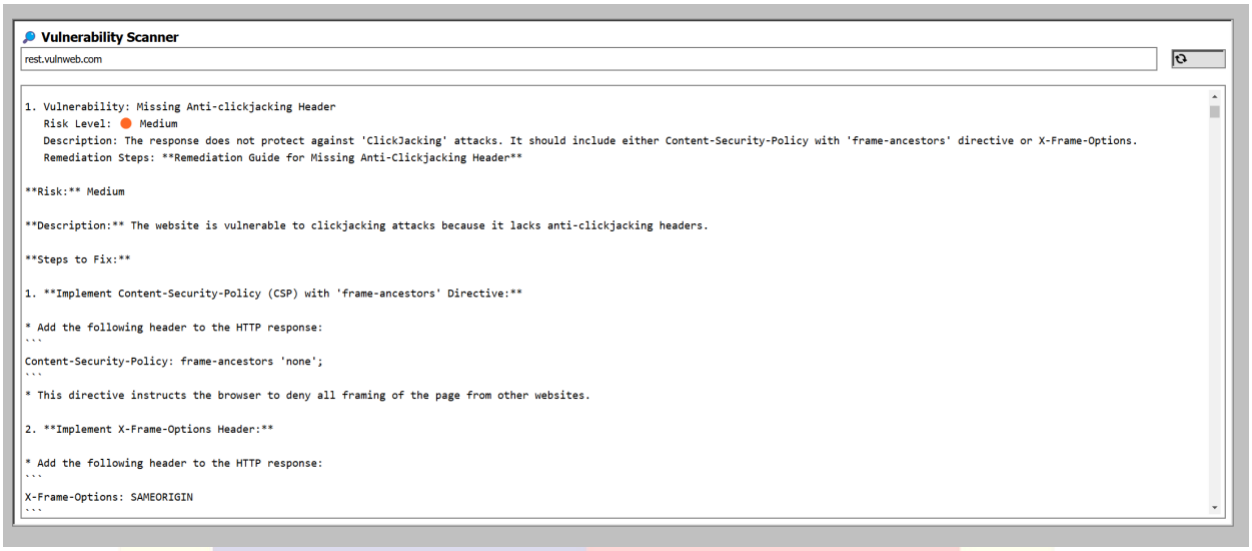

Fig 6: AI Based Remediation Steps

FUTURE WORK

- **Phase 4:** AI-powered remediation interpretation for better security insights.
- **Phase 5:** AI-based vulnerability scanning beyond OWASP ZAP, using ML-based detection models.
- **Phase 6:** AI-driven exploitation simulation for penetration testing.
- **Phase 7:** AI-powered phishing detection for fraud prevention.
- PWA Development: Convert the platform into a Progressive Web App (PWA) for mobile access.

CONCLUSION

This The AI-Powered Cloud-Based Vulnerability Scanner simplifies security assessments by automating remediation suggestions and eliminating the need for complex security tool configurations. With React.js, Google Gemini AI, and OWASP ZAP, this project has successfully:

- Enhanced vulnerability detection capabilities
- Automated security recommendations using AI
- Improved user experience through a dynamic frontend

Future developments will focus on AI-powered security intelligence with machine learning-based vulnerability scanning, phishing detection, and automated penetration testing.

REFERENCES

- Books, Papers, and Articles:
  - "The Web Application Hacker's Handbook" by Dafydd Stuttard and Marcus Pinto

    A comprehensive guide on web application vulnerabilities and exploitation techniques, providing foundational knowledge for vulnerability scanners.
  - OWASP ZAP: A Beginner's Guide to Automated Security Testing Published on Medium and OWASP Blog, this guide introduces the OWASP ZAP tool and its integration for web security testing.
  - "Understanding WHOIS and Its Role in Domain Name System" Published by ICANN, explaining WHOIS records' importance in cybersecurity and domain ownership transparency.
- Websites and Tools:
  - **Nmap Official Website** - https://nmap.org/
  - **Flask Documentation** - https://flask.palletsprojects.com/
  - **Tailwind CSS** - https://tailwindcss.com/
  - **OWASP ZAP** - https://www.zaproxy.org/
  - **Google Gemini AI API:** https://ai.google.dev/
  - **React.js Documentation:** https://react.dev/