

# Cohort 8 Group Members and Roles

1. Tiffany Kosgei – Project Lead
2. Sherry Obare – Role
3. Wayne Asava – Role

Mr. John Kuria

**CODEHOUND**

**PROBLEM BACKGROUND**

In the year 2023, several online newspapers including [The Star](https://beta.the-star.co.ke/) and Citizen compiled a report for The Communications Authority of Kenya regarding cyber threats in the country. The report showed that 860 million cyber-attacks were recorded that year. 79% of these attacks were attributed to flaws and vulnerabilities in the organizations’ systems.

According to a report by [Business Daily](https://www.businessdailyafrica.com/bd/corporate/technology/cyberattacks-targeting-kenya-rise-5-times-on-ai-driven-threats--4601942), they indicate that the growth of cyber-attacks has spiked five times over the past year. This growth is attributed to the use of AI to bypass traditional security measures through techniques such as sophisticated social engineering and exploiting zero day vulnerabilities. These cyber-attacks lead to the crippling of Internet Service Providers (ISPs), cloud service providers, government, health and the education sector. CodeHound narrows down to focus on cyber-attacks on web applications. This is mainly because web applications attacks have had an increase of 175% during the first quarter of 2024 according to the [Communications Authority of Kenya.](https://ke-cirt.go.ke/wp-content/uploads/2024/04/2023-24-Q3-Cyber-Security-Report.pdf) Majority of these attacks were targeted at government systems and the ICT sector. Attackers targeted user login credentials, vulnerable web browsers and database servers belonging to government and Internet Service Providers (ISPs). These attacks are mainly SQL Injection and Cross-Site Scripting (XSS) as stated by a report by [OWASP](https://www.imperva.com/blog/the-top-3-owasp-risks-to-the-financial-services-sector-in-2021-and-how-to-mitigate-them/) in 2021. These attacks are mainly brought about by vulnerability oversight caused by the developers focus on customer satisfaction thus neglecting security demands [[ Fonseca, 2007](https://bdigital.ipg.pt/dspace/bitstream/10314/3533/1/Fo)].

A couple of approaches have been curated to overcome the issue of vulnerability oversight in web applications including Web Application Vulnerability Scanners and Deep learning methods. For Web Application Vulnerability Scanners, they may produce false positives or even miss some vulnerabilities thus cannot be relied fully for accuracy. WAV Scanners also have limited depth in certain areas thus cannot fully comprehend a vulnerability if flagged. On the other hand, deep learning methods as a way of flagging vulnerabilities may lead to false positives, may miss a vulnerability and may struggle to understand the semantics of a code thus cannot be relied upon for accuracy.

CodeHound is yet another vulnerability scanner for web applications. CodeHound aims at thoroughly understanding code specifically written in JavaScript and thus specializes in overseeing the development of web applications developed with JavaScript and flagging vulnerabilities if any during the development phase. It also offers suggestions on how these errors that are flagged can be rectified. This will both fill the gap of lack of accuracy in WAV Scanners and deep learning methods as well as the gap of understanding a particular area in depth.

How will we ensure that CodeHound stands out among all the other vulnerability scanners?

**MARKET OPPORTUNITY**

Two current alternative situations/solutions that solve the same problem include:

●Deep learning vulnerability scanners for XSS and SQL injection analyze web application inputs and outputs using neural networks trained on extensive datasets of attack patterns. These models learn to recognize subtle variations of malicious payloads, context-sensitive injection points, and evasion techniques. They examine how user inputs are processed and output, identifying potential vulnerabilities where insufficient sanitization or encoding could allow attackers to inject malicious scripts or SQL commands. The scanners adapt to new attack vectors by continually updating their understanding of exploit patterns and application behaviors.

●Web application vulnerability scanners for XSS and SQL injection work by sending crafted inputs to web application entry points, like form fields and URL parameters. They analyze the application’s responses to detect if malicious scripts (XSS) or unauthorized database queries (SQL injection) can be executed. The scanners use predefined patterns and payload libraries to generate test cases, attempting to bypass security filters and identify vulnerabilities in the application’s input handling and output encoding processes.

**Below are some of the things that can be added or improved in addressing the solution:**

●Real-time analysis by detecting issues as developers code, providing immediate feedback, which is faster than post-development scanning.

●Proactive approach by addressing vulnerabilities during development is more efficient than finding them later in the software lifecycle.

●Specific fix suggestions by providing concrete solutions, not just identifying problems.

**From this gap, CodeHound can improve:**

●Contextual understanding: Deep learning models often lack specific context. Your real-time analysis can leverage the full context of the code being written, including surrounding functions, imported libraries, and project-specific patterns.

●Immediate feedback loop: Traditional scanners operate post-development. Your tool’s ability to provide instant feedback allows for a tighter, more efficient security-minded development cycle

●Reduced false positives: By analyzing code in-context as it’s written, you can potentially achieve higher accuracy than broad-sweep scans, reducing false positives that plague many existing tools.

●Tailored fix suggestions: Most scanners only identify issues. Your ability to offer specific, contextually relevant fix suggestions is a significant advantage.

●Developer education: Unlike passive scanners, your real-time approach can actively teach developers about security best practices as they code.

●Integration of security into development: CodeHound’s approach embeds security directly into the development process, addressing the gap between security and development teams that many organizations face.

CodeHound targets the market for web development companies in Kenya, particularly those focusing on web application security, which is significant due to the increasing reliance on digital platforms. The demand for vulnerability scanners targeting SQL injection and XSS attacks is driven by the rising number of web-based attacks, which have escalated dramatically in recent years([Lakshmi et al., 2024](https://typeset.io/papers/a-proactive-approach-for-detecting-sql-and-xss-injection-4wdiizzgdd))([Bhojak et al., 2015](https://typeset.io/papers/sql-injection-and-xss-vulnerability-detection-in-web-4yhzwwk1yy)). Key aspects of this market include:

**Market Size and Revenue**

●The global web application security market is projected to reach USD 12.5 billion by 2025, with a substantial portion attributed to emerging markets like Kenya.

●Local web development companies are increasingly investing in security solutions, indicating a growing revenue stream for vulnerability scanners. Target Audience

●The primary audience includes web application developers in Kenya, who often lack adequate security training ([Lakshmi et al., 2024](https://typeset.io/papers/a-proactive-approach-for-detecting-sql-and-xss-injection-4wdiizzgdd)).

●Approximately 70% of developers report insufficient knowledge of security practices, highlighting a critical need for effective tools ([Maini et al., 2019](https://typeset.io/papers/automated-web-vulnerability-scanner-44fc1d0sr4)). Economic Contribution

●Web development contributes significantly to Kenya’s economy, with the tech sector growing at an annual rate of 10% ([Zangana, 2024](https://typeset.io/papers/exploring-the-landscape-of-website-vulnerability-scanners-2bpq72p8az)).

●Enhanced security measures can lead to increased consumer trust and, consequently, higher economic activity in the digital space.

**SOLUTION IDEA**

*Target User*

Our target users are web application developers because according to our research, there has been an increase in web application attacks. The [OWASP 10 report](https://owasp.org/www-project-top-ten/) indicates SQL Injection and XSS are the most prevalent web application attacks.

Therefore, our project is aimed to help the web application developers to scan and detect the vulnerabilities during development to avoid their products going live with vulnerabilities. Our focus is mainly of web application developers because they tend to focus more on the functionality while neglecting security aspects.

Web application attacks often lead to data breaches hence affecting the users of the applications i.e companies and customers. However, we choose to focus on web application developers because they are the ones who develop the web applications and through early detections, SQL injection and XSS can be prevented during early stages.

*Solution Prototype*

The solution is a browser extension that functions as a real-time vulnerability scanner for web applications. It leverages Abstract Syntax Tree (AST) analysis to dissect the source code as the developer types, identifies potential security threats related to XSS and SQL Injection, and provides actionable feedback and remediation suggestions. This proactive approach enhances code security by catching vulnerabilities early in the development process, reducing the risk of deployment issues and potential attacks.

**The technology choice includes:**

**●Abstract Syntax Tree (AST) Generator:** An AST allows for an in-depth analysis of JavaScript code structure, enabling precise identification of potential vulnerabilities. By representing code in a tree format, it simplifies the process of understanding code flow and structure, making it easier to pinpoint areas prone to XSS and SQL Injection attacks.

●**JavaScript:** As the primary language for web development, using JavaScript allows for seamless integration with existing web applications. It ensures that the extension can effectively analyze the code being developed and interact directly with the JavaScript environment.

●**Web Extensions API:** This API provides a standardized way to create extensions that work across multiple browsers. It allows your extension to monitor and interact with web pages and code editors, facilitating real-time feedback for developers.

●**Local Database** (e.g., IndexedDB or LocalStorage): A local database can efficiently store the rules for detecting XSS and SQL Injection vulnerabilities. Using a client-side database allows for quick access and updates, ensuring the extension performs efficiently without requiring constant server communication.

●**Rule Engine** (JavaScript-based): Implementing a rule engine in JavaScript allows you to define and manage the vulnerability detection rules directly within the same environment as the code being analyzed. This enhances the maintainability and flexibility of the rules as new vulnerabilities are discovered.

●**Regular Expressions** (RegEx): Regular expressions are invaluable for pattern matching in code, especially for identifying common vulnerabilities. They allow for quick scanning of input/output operations that may expose the application to XSS and SQL Injection.

●**API (**RESTful or WebSocket): An API facilitates communication between the AST generator and the rule database. Using a RESTful or WebSocket approach allows for real-time updates and efficient data exchange, ensuring that the extension remains responsive to code changes.

●**Frontend Framework** (e.g., React or Vue): A modern frontend framework can help create an intuitive user interface for the extension. This enhances the user experience by making it easy for developers to view alerts, suggestions, and detailed information about detected vulnerabilities.

**Below is a description of how the solution works and its flowchart:**

The process begins with a Parser, which reads the source code and breaks it down into its syntactical elements.

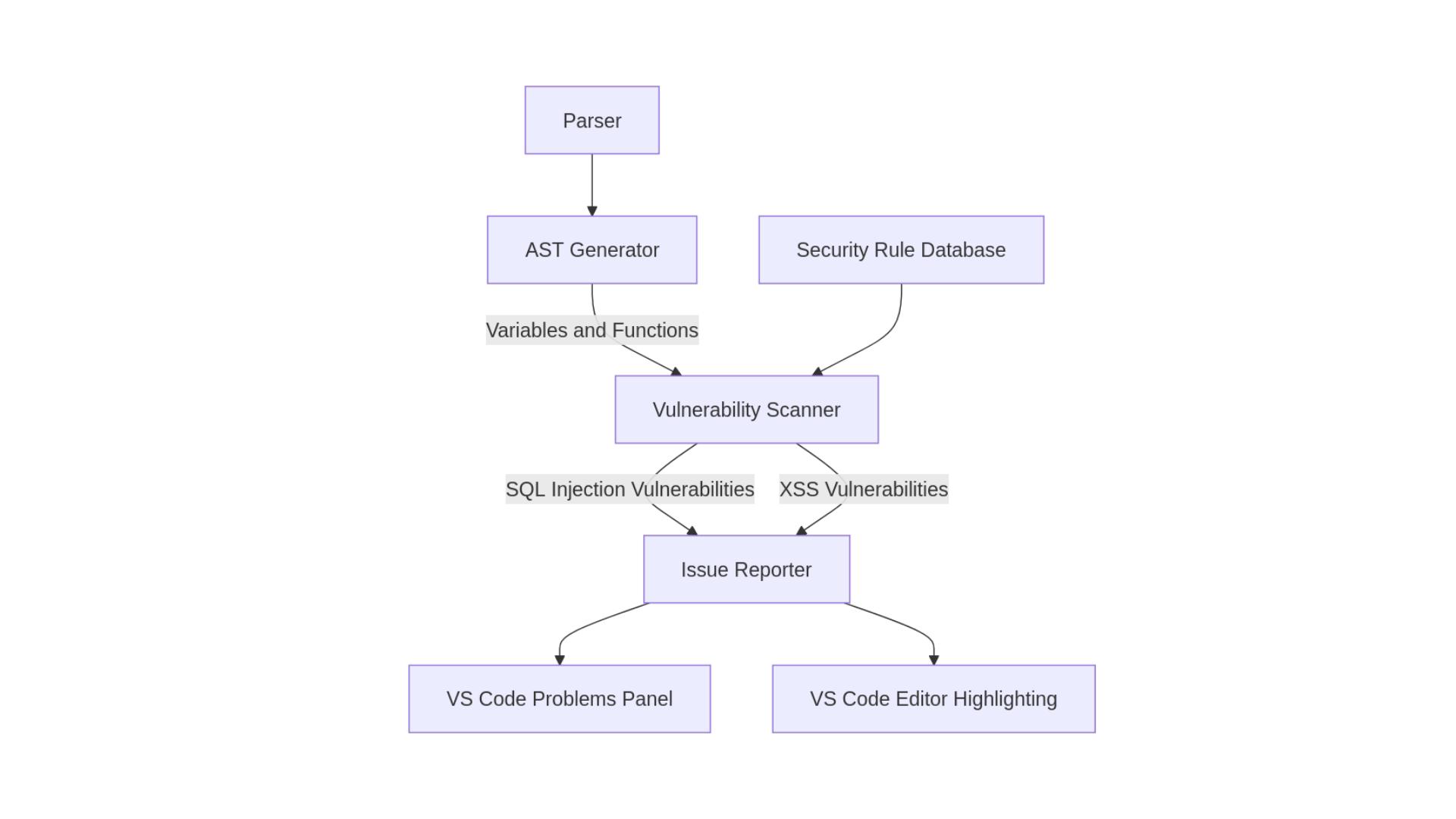
These elements are then fed into the AST Generator, which creates an Abstract Syntax Tree (AST). This tree represents the structure of the code, allowing the system to analyze variables, functions, and expressions.

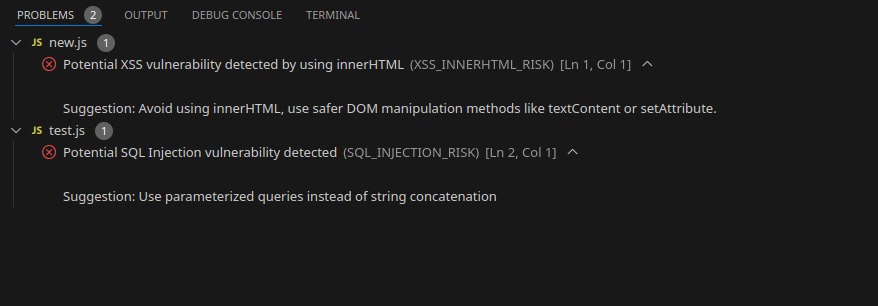
The Security Rule Database contains predefined patterns and rules for identifying vulnerabilities like SQL Injection and XSS.

The Vulnerability Scanner uses both the AST (variables and functions) and the security rules to scan the code. It checks for patterns that match known vulnerabilities, specifically focusing on SQL Injection and XSS.

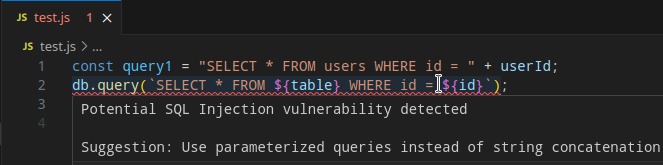
Once vulnerabilities are identified, the Issue Reporter is responsible for reporting these issues.

It outputs the results in two ways: to the VS Code Problems Panel, where developers can see a list of problems, and through VS Code Editor Highlighting, where vulnerable code sections are directly highlighted within the editor. This makes it easier for developers to identify and fix the vulnerabilities.

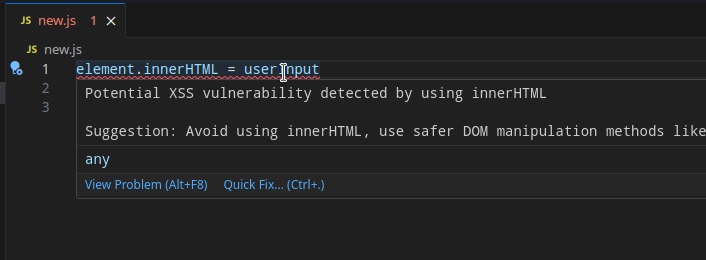


**Below are screencasts that align with how our solution works:**

1. Screencast of a potential XSS threat detected and suggestion on how to rectify it.



1. Screencast of a potential SQL Injection threat detected and suggestion on how to rectify it.



1. Screencast of how the errors are flagged and suggestions offered.

CodeHound addresses vulnerability oversight in web application development by directly integrating into the development workflow and offering real-time error identification and rectification suggestions. Here’s how it tackles the issue of rushed development and the associated security risks:

●**Immediate Detection**: By scanning the code during development, your solution detects potential SQL Injection and XSS vulnerabilities early in the process. This proactive approach ensures that these critical security flaws don’t go unnoticed due to tight deadlines.

●**Actionable Suggestions**: Instead of merely flagging vulnerabilities, your tool provides corrective suggestions. This feature minimizes the time developers need to research or troubleshoot, allowing them to fix issues quickly, even under time constraints.

●**Integration with Developer Tools**: Since it integrates with environments like VS Code, developers can seamlessly receive alerts and fixes within their regular workflow. This integration prevents vulnerabilities from being overlooked while enabling rapid resolution, balancing speed with security.

***Assumptions***

●Web application attacks i.e SQL Injection and XSS will continue to rise

●Web developers will always build web applications

●Most developers are not fully trained in identifying or fixing vulnerabilities like XSS or SQL injection during coding.

●Vulnerabilities in a system will always go live undetected unless thorough security measures are implemented.

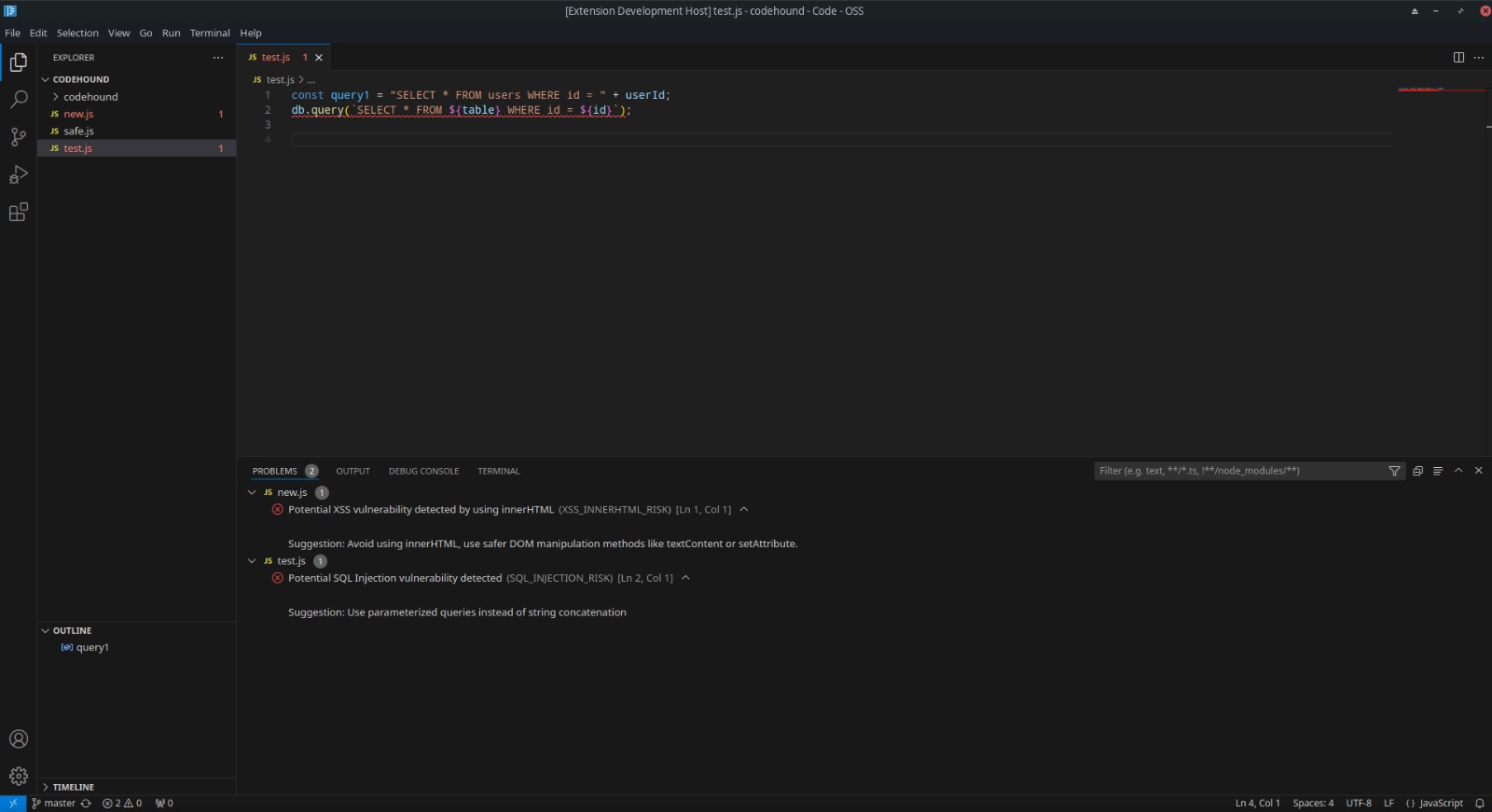
**VALUE PROPOSITION**

CodeHound aims to provide developers with an efficient tool for identifying and resolving security vulnerabilities like SQL Injection and XSS in real-time, minimizing oversight during rapid development cycles. This value is delivered to web developers in web development companies who need to ensure secure code without sacrificing speed.

**DESIGN SOLUTION**

The technologies used to build the solution are:

* Abstract Syntax Tree(AST)
* JavaScript
* Web Extensions API
* Local Database
* Rule Engine
* Regular Expressions
* API
* Frontend Framework

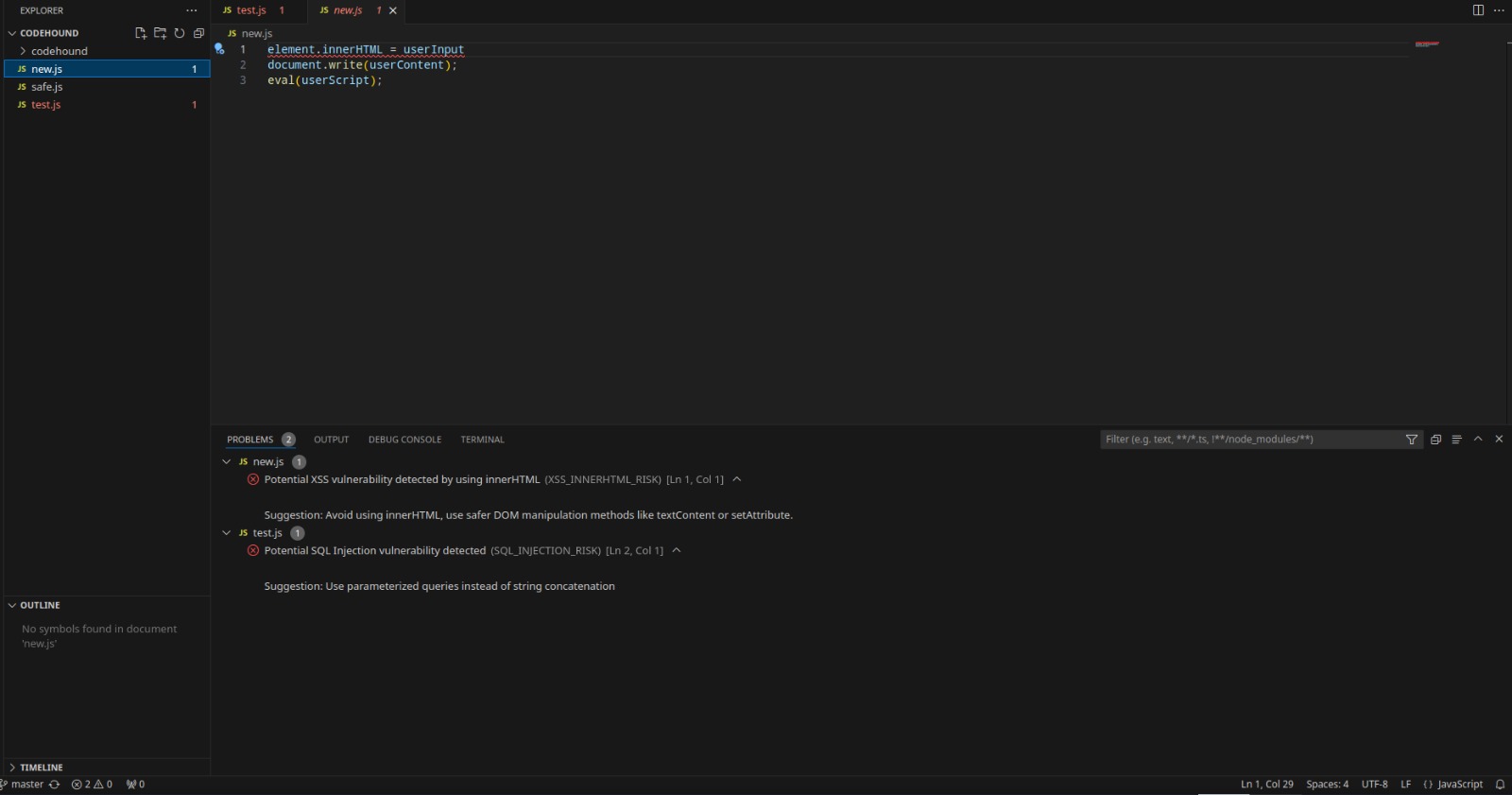
Screenshots of modules include:

1. **Security Vulnerability Analyzer**

This module scans the open JavaScript document and identifies potential security risks. It does so by parsing the code into an Abstract Syntax Tree (AST) and traversing it, looking for patterns that match the defined security rules (e.g., SQL Injection, XSS).

Key Features:

* Analyzes code in real-time as you type or modify JavaScript files.
* Detects dangerous coding patterns, such as SQL queries constructed via string concatenation or unsafe DOM manipulations.



**2. Security Fix Provider (SecurityFixProvider).**

This module offers Quick Fixes for security issues identified by CodeHound. When a vulnerability is detected, the provider suggests fixes that improve security.

Key Features:

Integrated with VS Code’s Quick Fix system, allowing developers to apply suggested fixes

with a single click.

Provides context-based suggestions, such as using parameterized queries instead of

concatenation.

**Link to the solution**

**GitHub -** [**https://github.com/AsavaAsava/codehound-extension**](https://github.com/AsavaAsava/codehound-extension)

**BUSINESS MODEL**

CodeHound will implement the freemium model to gain profit. The subscriptions will be monthly and annually with a 14-day trial on the premium features.

Free features include:

●Limited scan to JavaScript code

●Provide general suggestions to any vulnerabilities that may be detected Premium features include:

●Scan all programming languages for vulnerabilities

●Generate a detailed and personalized report for the vulnerabilities found.

●Allows the user to scan the whole code after completing their project just to ensure there are no loopholes

**RESPONSIBLE COMPUTING**

The following are the responsible computing practices that are practiced by CodeHound:

●**Inclusion:** CodeHound aims to include all developers, regardless of their level of security expertise. By offering vulnerability detection and providing suggestions for fixes, it ensures that even less experienced developers can write secure code, enhancing overall inclusion in web application security.

●**Accessibility:** CodeHound is accessible to developers from various backgrounds and technical levels. It ensures easy integration with popular IDEs (like VS Code) and providing clear explanations of both the vulnerabilities and the suggested solutions will enhance accessibility.

●**Biasness:** While CodeHound is still an upcoming extension that only helps developers using JavaScript to develop web applications and only focuses on XSS and SQL Injection threats, we aim at expanding it to cover all languages and most of the threats that web applications face. This will therefore eradicate the biasness.

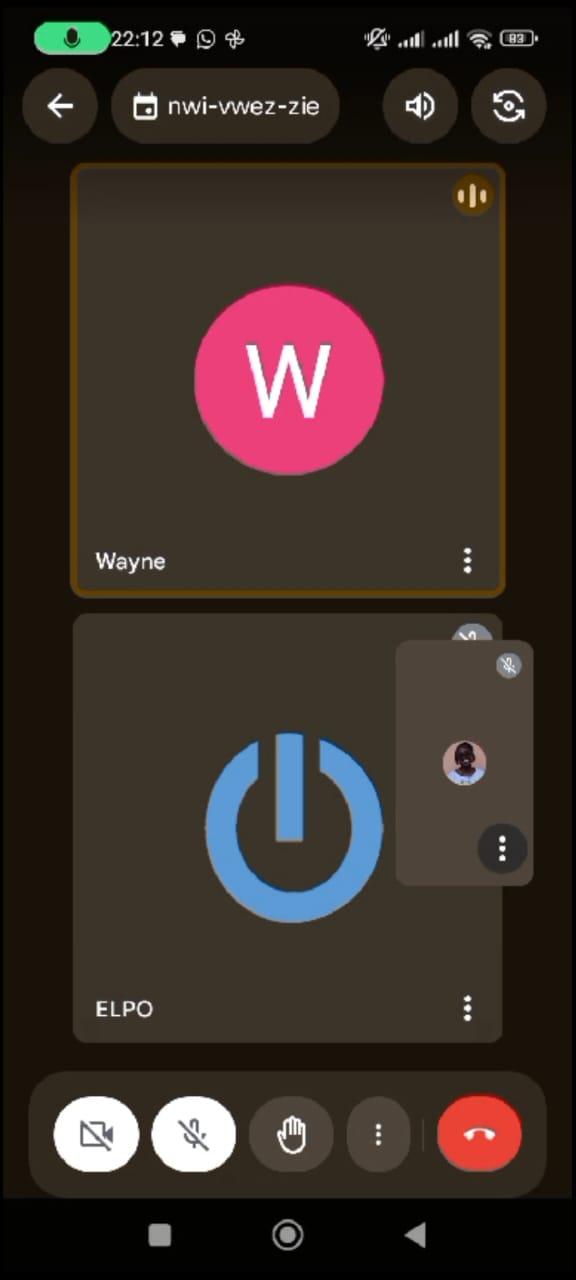
●**Privacy and Security:** CodeHound ensures that any data used (e.g., code snippets or patterns flagged) is securely handled, with no unauthorized access to user code or sensitive data. It’s essential that user data is anonymized or protected to prevent exploitation or misuse. Additionally, the scanner itself should be secure and not introduce any vulnerabilities.

**TRACTION**

Our team managed to talk to two web application developers; Steve Tom and Duncan Mukwe. Below are screenshots from the Google Meet we had.

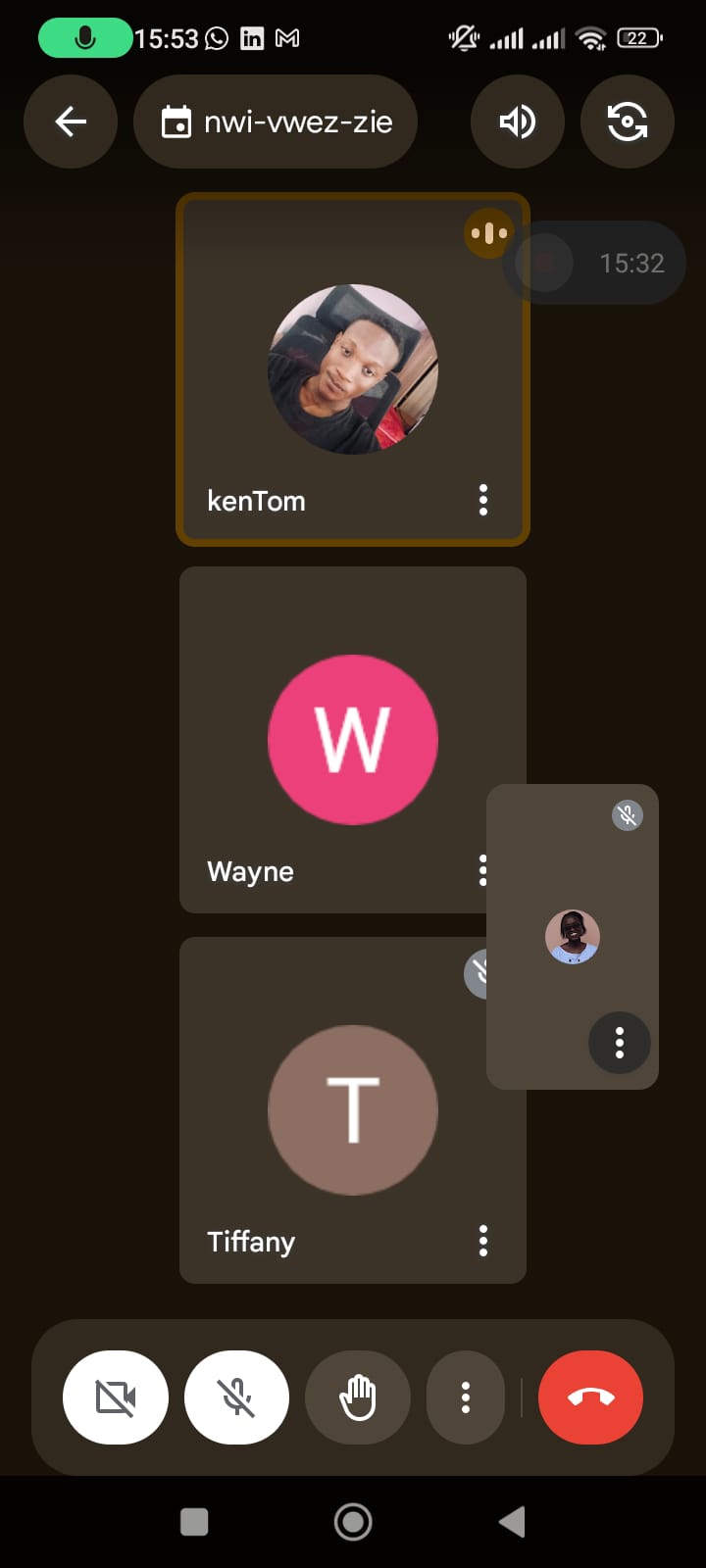
●Duncan Mukwe

[Duncan](https://ke.linkedin.com/in/dancan-mukwe-96652a206) is a web application developer working at Elpo Tech company. He narrated to us a time when his company faced an attack on their website. The attackers found a loophole in their system due to outdated software and the attackers were able to lead Elpo Tech’s clients to their website. This incident made the company take security seriously and the employees then were trained in general security practices. However, he agreed that our solution will make their work easier as well as the incoming junior developers.



●Steve Tom

[Steve](https://raccoon-porfolio-v2.vercel.app/) is a computer science graduate and a passionate software developer, dedicated to crafting efficient and scalable web applications. During our virtual call, he explained to us that when implementing security on his applications, he relies heavily on the frameworks that he uses to automatically implement security measures which are always in-built.

****

**CODEHOUND’S TEAM**

**Wayne Asava – Software developer**

Wayne Asava is a proficient full-stack developer with a strong focus on JavaScript and its frameworks, including React and Node.js. His expertise in both front-end and back-end technologies enables him to create secure and efficient web applications that meet functional requirements while prioritizing security. Wayne’s ability to collaborate effectively with cross-functional teams enhances the development process, ensuring that security measures are seamlessly integrated into the software lifecycle.

Wayne’s commitment to cybersecurity complements his development skills, driving him to integrate security measures throughout the software development lifecycle. He actively participates in workshops on secure coding practices, ensuring he stays informed about the latest threats and mitigation strategies. This unique combination of full-stack development expertise and cybersecurity knowledge makes Wayne an essential contributor to the project. His knowledge of JavaScript frameworks will be instrumental in implementing effective vulnerability scanning features within the VS Code extension, allowing the team to deliver a reliable solution that proactively addresses potential threats in web applications.

****

**Sherry Obare – Vulnerability assessor**

Sherry is a vulnerability assessor with extensive expertise in using various vulnerability assessment tools, including Veracode, Checkmarx, and Grype. Her understanding in using these tools enables her to identify, analyze, and prioritize security vulnerabilities effectively, ensuring that potential threats are addressed promptly. Sherry’s technical skills encompass a deep understanding of secure coding practices, risk assessment methodologies, and remediation strategies, which are essential for safeguarding web applications against attacks such as SQL injection and XSS.

Beyond her technical capabilities, her strong analytical thinking allows her to evaluate complex security issues systematically, while her communication skills facilitate clear reporting and collaboration with development teams. Additionally, her attention to detail ensures that no vulnerability goes unnoticed. Sherry’s unique combination of technical expertise and interpersonal skills makes her play a crucial part in ensuring the success of the project by recommending effective mitigation strategies within the VS Code extension.

****

**Tiffany Kosgei – Web application penetration tester**

Tiffany Kosgei is a skilled web application penetration tester with extensive experience in identifying and exploiting security vulnerabilities in web applications. Her proficiency with tools such as Burp Suite, OWASP ZAP, and Metasploit allows her to perform thorough penetration tests that uncover potential weaknesses, including SQL injection and XSS vulnerabilities.

Tiffany possesses essential soft skills that enhance her effectiveness as a penetration tester.

Her strong problem-solving abilities enable her to think creatively when simulating attacks, while her excellent communication skills facilitate clear reporting of findings to both technical and non-technical stakeholders. Tiffany’s attention to detail ensures that all vulnerabilities are meticulously documented and addressed. Her skills play a critical part in ensuring the

success of the project by rigorously testing the VS Code extension for vulnerabilities and providing recommendations for remediation.