# SecureCode AI

## *Design and Implementation of a Unified Static Application and Dependency Security Analysis Platform*

---

**1. Problem Statement:**

With the rapid growth of software-driven systems, applications are increasingly exposed to security vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), Command Injection, insecure memory usage, and hardcoded credentials. These vulnerabilities often originate during the development phase and remain undetected until deployment, where they may lead to data breaches, system compromise, and financial losses.

In addition to vulnerabilities in application source code, modern software systems are also exposed to risks through **third-party libraries and dependencies**. Even if application logic is secure, the use of vulnerable packages can introduce known Common Vulnerabilities and Exposures (CVEs). Developers often fail to audit these dependencies due to fragmented tooling and lack of integrated security solutions.

Existing challenges include:

- Reliance on simple pattern matching, resulting in high false positives
- Limited support for multi-line and data-flow-based vulnerability detection
- Separate tools for code security and dependency security
- High cost and complexity of commercial security tools

Hence, there is a need for a **unified security analysis platform** that can detect both **code-level vulnerabilities and package-level security risks** early in the Software Development Life Cycle (SDLC).

---

**2. Proposed Solution and Methodology**

**2.1 Proposed Solution**

SecureCode AI is a **unified Static Application Security Testing (SAST) platform** designed to analyze application source code and third-party dependencies without executing the program.

The solution provides:

- Automated detection of source code vulnerabilities
- Package and dependency security analysis
- Multi-language support
- Severity scoring and CWE mapping
- Clear explanations and remediation guidance
- A single interface for holistic application security assessment

---

### 2.2 Methodology

SecureCode AI follows a **static analysis and rule-based methodology**, combining taint-based data-flow analysis with dependency inspection.

The methodology consists of:

1. Identifying user-controlled input sources
2. Tracking tainted data across variable assignments
3. Detecting sensitive operations (sinks)
4. Applying security rules to classify vulnerabilities
5. Analyzing third-party packages for known security risks
6. Aggregating and reporting results in a unified format

This methodology allows detection of vulnerabilities that span multiple lines and components.

---

### 2.3 Package Security Analysis (Dependency Scanning)

In addition to source code analysis, SecureCode AI includes a **package security scanner** that examines third-party dependencies used by an application. This module identifies insecure or vulnerable packages that may introduce known vulnerabilities even when application code is secure.

The package scanner:

- Accepts package information provided by the user
- Analyzes dependencies using predefined vulnerability rules and known insecure patterns
- Integrates results with code-level findings
- Eliminates the need for separate dependency scanning tools

This approach provides a **holistic view of application security**, covering both internal code and external components.

---

### 2.5 Technical Explanation of the Solution

#### 2.5.1 Code Security Analysis Engine

The core of SecureCode AI is a **taint-based static analysis engine** implemented in Python.

Key technical concepts include:

- **Taint Sources:** User-controlled inputs such as HTTP parameters, form inputs, cookies, request objects, and command-line arguments.
- **Taint Propagation:** Tracking how tainted data flows through variable assignments across multiple lines of code.
- **Sinks:** Security-critical operations such as database query execution, system command execution, HTML rendering, and unsafe memory functions.

- **Rule Engine:** A collection of vulnerability detection rules mapped to CWE identifiers, severity levels, and remediation guidance.

Unlike simple scanners, SecureCode AI performs **cross-line and cross-variable analysis**, enabling accurate detection of vulnerabilities such as SQL Injection.

---

### 2.5.2 Package Scanner – Technical Explanation

The package security scanner operates by processing dependency information supplied by the user. The system evaluates these dependencies against predefined vulnerability rules and known insecure package patterns.

Key characteristics:

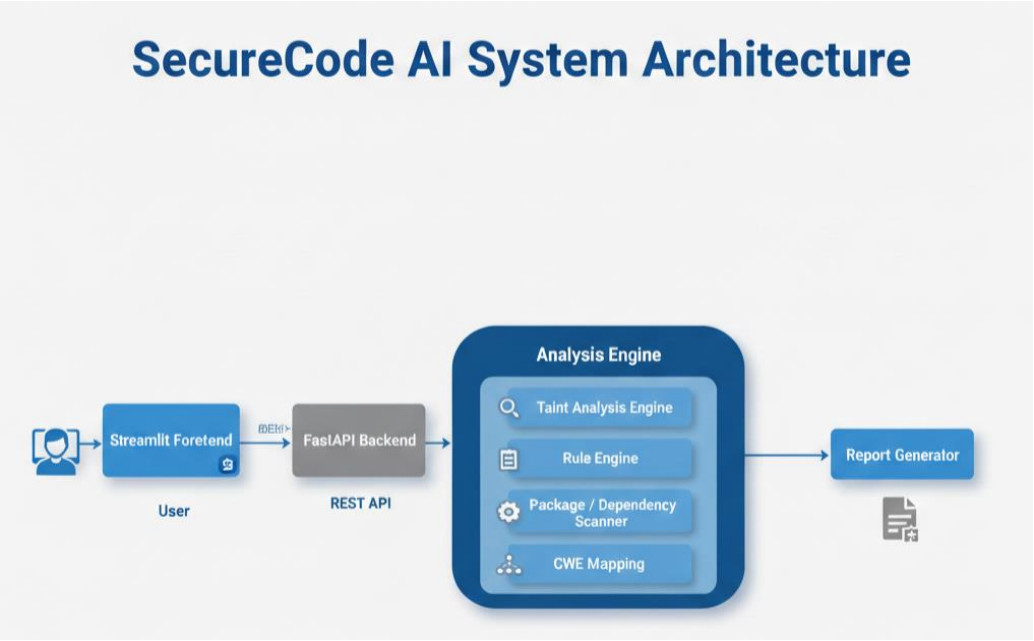
- Operates within the same backend infrastructure as code analysis
- Requires no additional backend services
- Produces consistent severity scoring and reporting
- Integrates seamlessly with code-level vulnerability results

This design ensures both code and dependency risks are assessed under a single security framework.

---

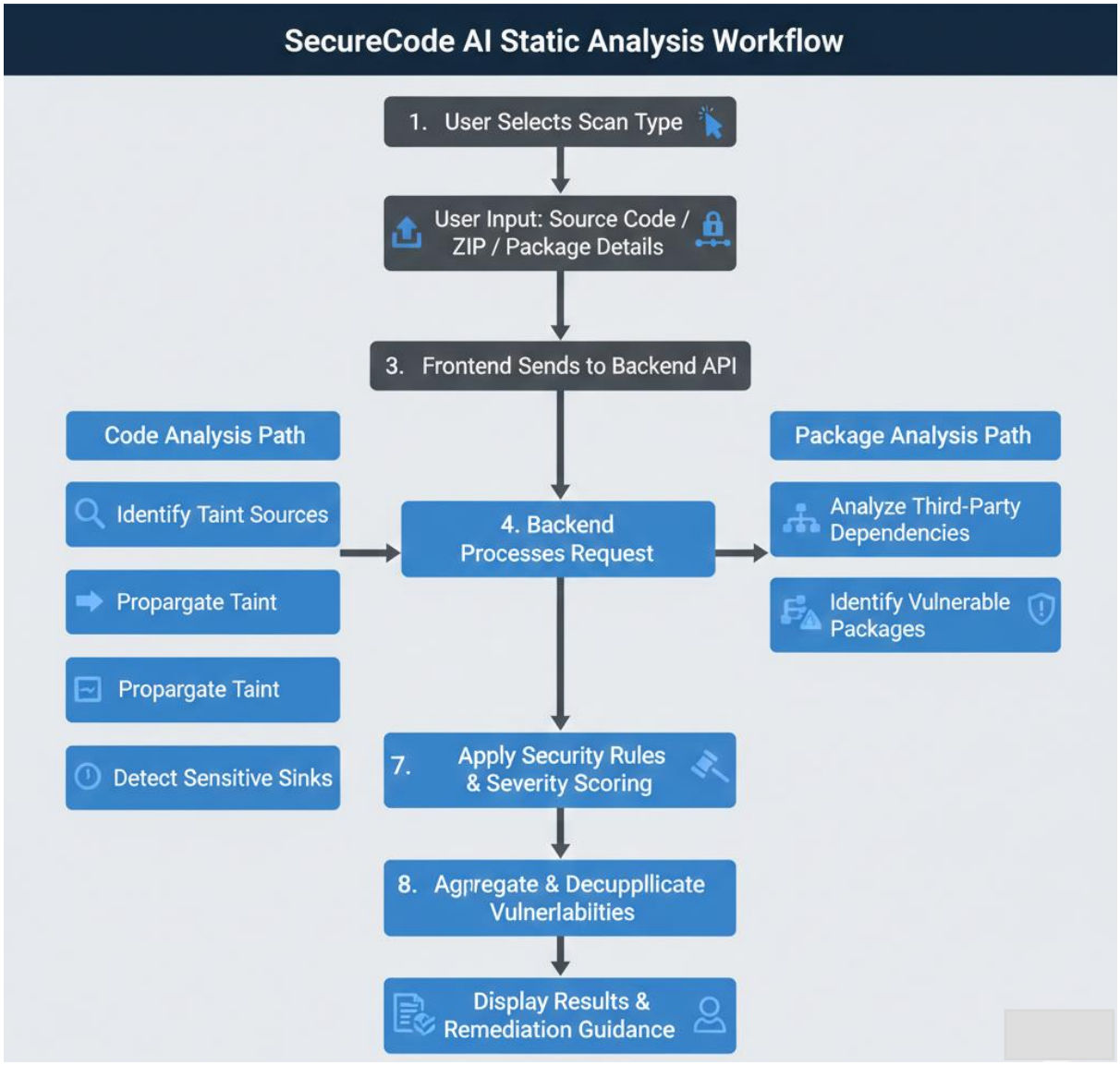
## 3. System Architecture

### 3.1 Architecture Diagram




---

### 3.2 Workflow Diagram




---

**4. Technology Stack and Libraries Used**

**Frontend**

- **Streamlit**
  Used to develop an interactive and lightweight dashboard for security analysis.

**Backend**

- **FastAPI**
  Used for building high-performance REST APIs.

**Core Language**

- **Python**
  Chosen for its strong static analysis capabilities, extensive security ecosystem, and rapid development support.

**Libraries and Tools**

- re – Pattern matching and rule enforcement
- json – Structured report generation
- requests – API communication
- python-multipart – File upload handling

---

**5. Impact on the Real World**

SecureCode AI contributes to real-world security by:

- Detecting vulnerabilities early in the SDLC
- Reducing post-deployment security costs
- Improving developer awareness of secure coding
- Identifying risks introduced by third-party dependencies
- Supporting cybersecurity education and training

---

**6. Innovation and Novelty**

The innovative aspects of SecureCode AI include:

- Taint-based static analysis implemented at an academic level
- Unified analysis of both source code and dependencies
- Cross-language vulnerability detection
- Integration of CWE mapping and remediation guidance
- Modular and extensible architecture

---

**7. Existing Solutions and Uniqueness**

**Existing Solutions**

- Commercial SAST tools: Powerful but costly and complex
- Dependency scanners: Focus only on packages
- Regex-based scanners: High false positives and limited accuracy

**Uniqueness of SecureCode AI**

- Combines code security and package security in one platform
- Lightweight and easy to deploy
- Educational and research-focused design
- No reliance on proprietary tools or licenses

---

**8. Business Analysis**

**Target Users**

- Educational institutions
- Small development teams
- Cybersecurity students and trainees
- Startups requiring basic security assessment

**Value Proposition**

- Low-cost application security analysis
- Early vulnerability detection
- Reduced tooling complexity

**Future Business Scope**

- SaaS-based secure code and dependency scanning
- CI/CD pipeline integration
- Organizational dashboards
- Advanced analytics and reporting

---

**9. Additional Considerations**

**Ethical Use**

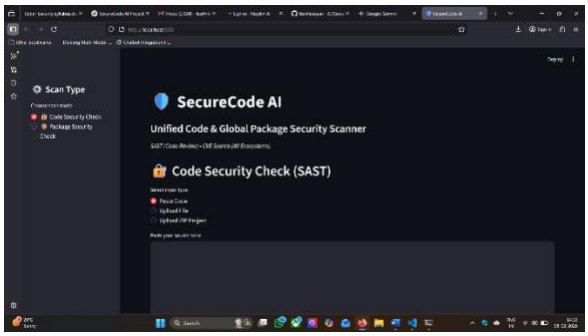
SecureCode AI is intended strictly for educational and authorized security analysis. Unauthorized scanning of third-party systems is discouraged.

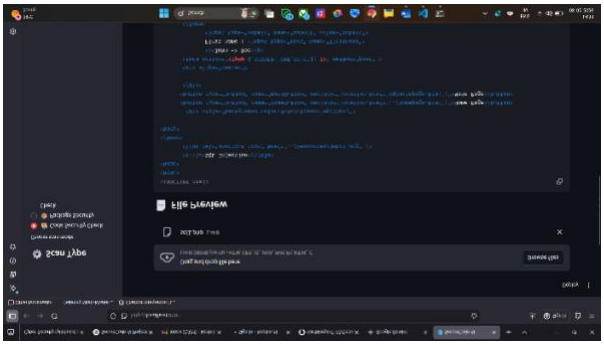
**Limitations**

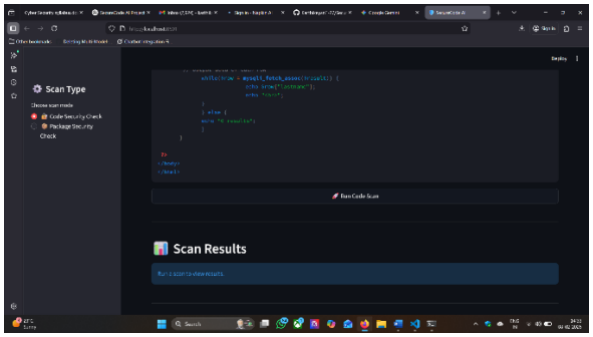
- Static analysis only
- Intra-procedural analysis
- Limited control-flow awareness
- Potential false positives and false negatives

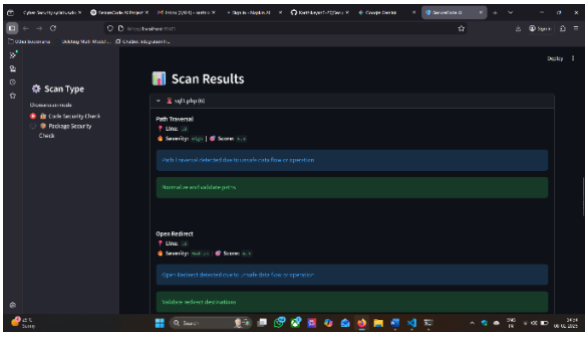
---

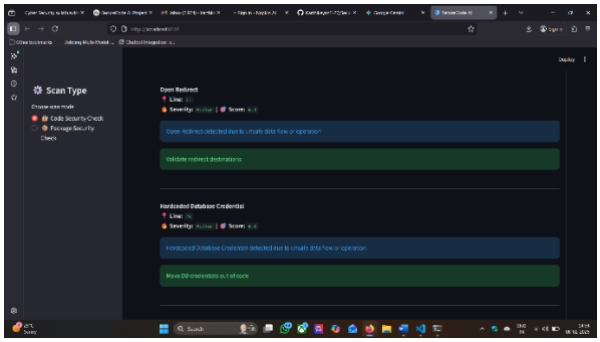
**RESULT IMAGES:**

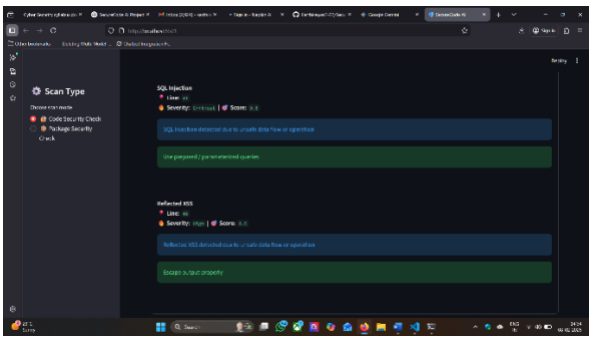

Scan Type
SecureCode AI
Unified Code & Global Package Security Scanner
Code Security Check (SAST)



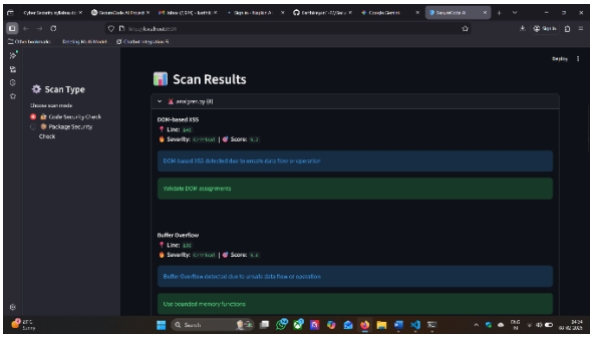

Scan Type
Scan Results

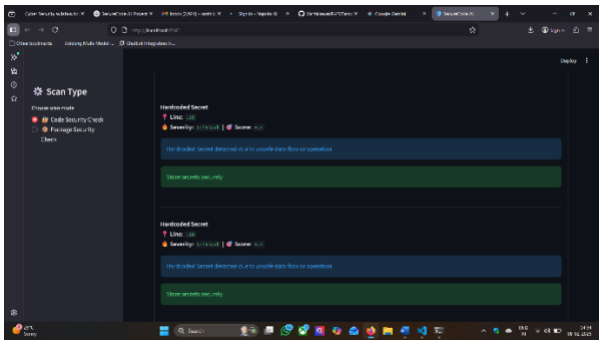

Scan Type
Scan Results

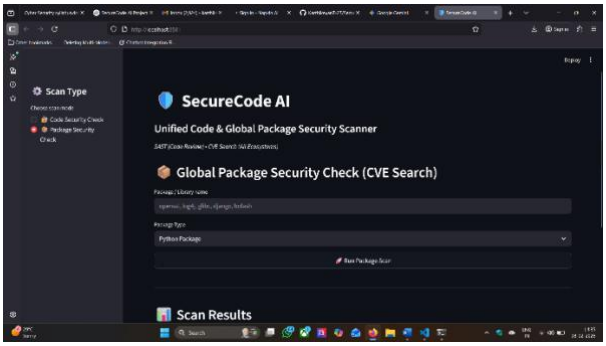

Scan Type
Open Redirect
Hardcoded Database Credential

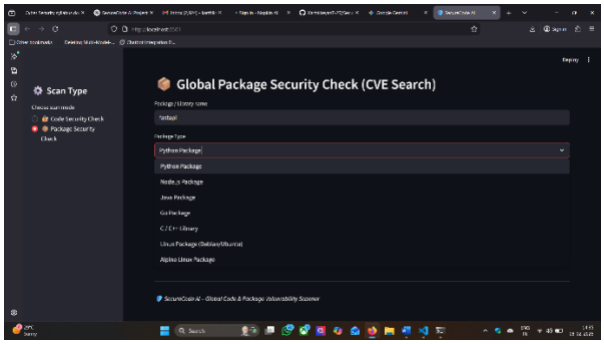

Scan Type
SQL Injection
Reflected XSS

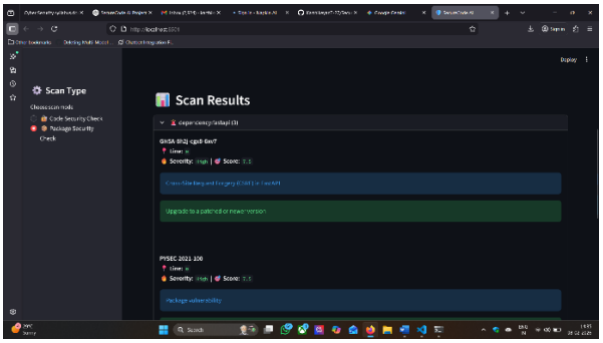

Scan Type
SecureCode AI
Unified Code & Global Package Security Scanner
Code Security Check (SAST)


Scan Type
Scan Results


Scan Type
Hardcoded Secret
Hardcoded Secret


Scan Type
SecureCode AI
Unified Code & Global Package Security Scanner
Global Package Security Check (CVE Search)
Scan Results




---

## Conclusion

SecureCode AI demonstrates the practical application of secure software engineering principles by integrating taint-based static code analysis with package security assessment. By unifying code-level and dependency-level security checks, the project provides a comprehensive and educational approach to application security analysis.

---