System for Artificial Intelligence Vulnerability Taxonomy (AIVT)

Technical Specification

Table of Contents

1 Overview ... 1

2 Architecture ... 1

2.1 System Architecture ... 1

2.1.1 Frontend ... 2

2.1.2 Backend ... 2

2.1.3 Database ... 2

2.1.4 API ... 3

2.1.5 Security ... 3

2.1.6 Hosting ... 4

3 Installation Instructions ... 4

3.1 System Configuration ... 4

3.2 System Parameters ... 4

4 Detailed Design ... 4

4.1 User Interface ... 4

4.1.1 Screens ... 4

4.2 Implementation Steps ... 6

4.2.1 Requirement Analysis: ... 6

4.2.2 Design: ... 6

4.2.3 Development: ... 6

4.2.4 Testing: ... 6

4.2.5 Deployment: ... 6

4.2.6 Maintenance: ... 6

1 OVERVIEW

The system aims to provide a comprehensive framework to address and categorize AI vulnerabilities. The application is planned to design and implement with similar functions to CVE MITRE database but specifically aiming AI vulnerabilities.

The application addresses, identifies and categorizes AI vulnerabilities that arise from AI systems in three stages of their lifecycles: development, training and deployment. The target audience is researchers, organizations that work on AI models and systems in aspects of security, reliability and fairness.

2 ARCHITECTURE

2.1 System Architecture

The web application should support the classification, search, and management of AI vulnerabilities. Key components include:

1. **Frontend**: User interface for interacting with the system.
2. **Backend**: Server-side logic for processing requests.
3. **Database**: Storage for vulnerability data.
4. **API**: Interface for communication between frontend and backend.
5. **Security**: Measures to protect the system and data.
6. **Hosting**: Infrastructure for deploying the system, including servers and networking.

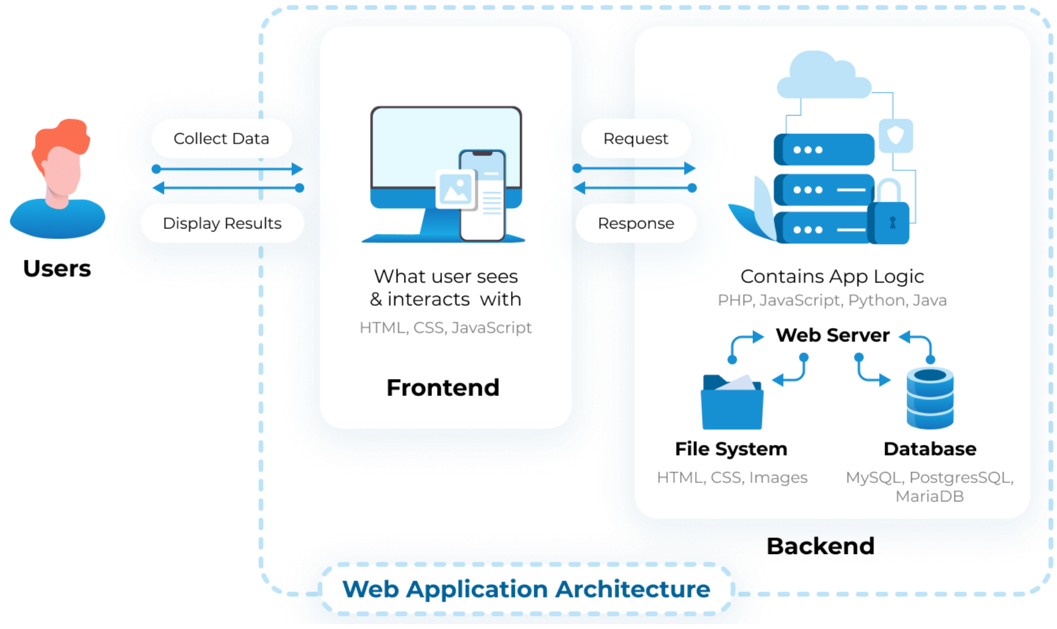


Figure 1: Web application architecture (source: https://litslink.com/blog/web-application-architecture)

2.1.1 Frontend

- **Technology**: React.js for building a dynamic user interface.
- **Components**:
    - **User Dashboard**: Display vulnerability listings, search, and filter options.
    - **Submission Form**: Form for users to submit new vulnerabilities.
    - **Detailed View**: Page to view detailed information about a specific vulnerability.
    - **Authentication**: Login and registration pages.
- **Features**:
    - Responsive design for mobile and desktop.
    - Accessibility features for inclusive design.

2.1.2 Backend

- **Technology**: Node.js with Express.js for handling server-side operations.
- **Components**:
    - **API Layer**: RESTful APIs for CRUD operations on vulnerability data.
    - **Authentication & Authorization**: JWT-based authentication.
    - **Business Logic**: Processing and validation of user inputs.
    - **Logging & Monitoring**: Tools like Winston for logging and Prometheus for monitoring.
- **Features**:
    - Data validation and sanitation.
    - Error handling and reporting.
    - Rate limiting to prevent abuse.

2.1.3 Database

- **Technology**: PostgreSQL for relational data storage.
- **Schema**:
    - **Users**: Table for storing user information.
    - **Vulnerabilities**: Table for storing vulnerability data, including fields for ID, description, severity, impact, etc.
    - **Logs**: Table for storing system logs and events.
- **Features**:
    - Indexing for fast search and retrieval.
    - Backup and recovery plans.
    - Data encryption at rest.

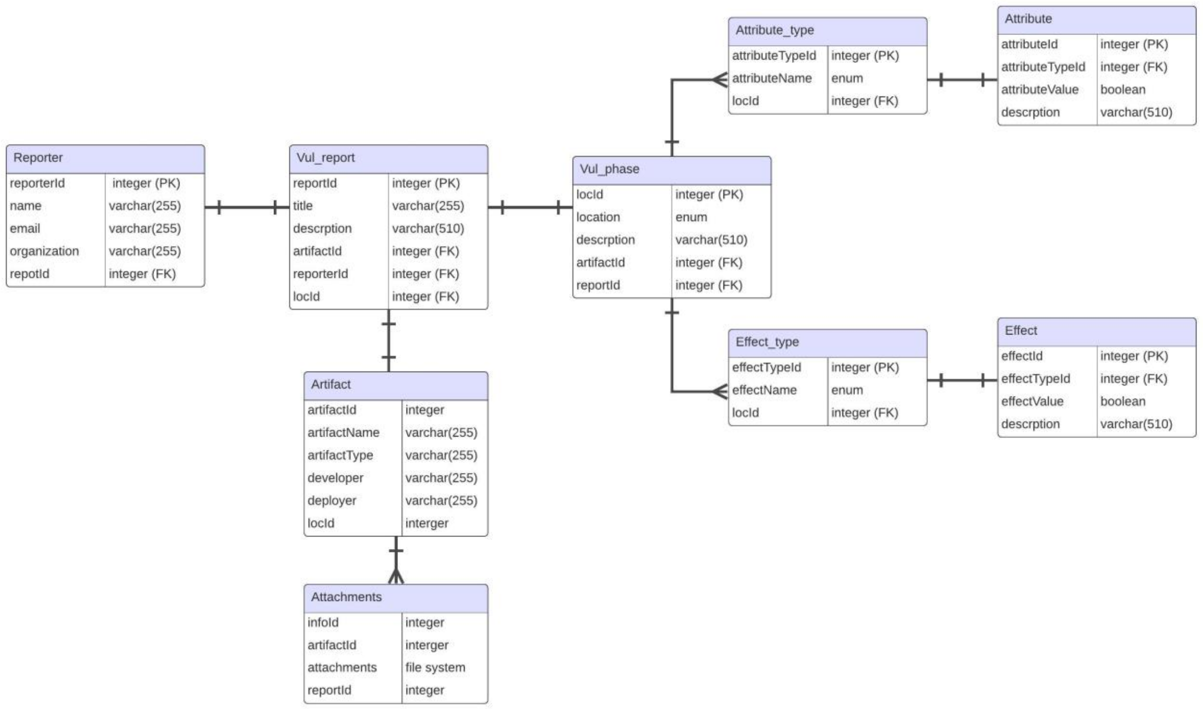


Figure 2: Entity-Relationship diagram of the DB models

2.1.4 API

- **Technology**: RESTful API.
- **Endpoints**:
  - **/attribute_type** - Endpoints for managing attribute types.
  - **/attribute** - Endpoints for managing attributes.
  - **/reporter** - Endpoints for managing reporters.
  - **/vul_report** - Endpoints for managing vulnerability reports.
  - **/vul_phase** - Endpoints for managing vulnerability phases.
  - **/artifact** - Endpoints for managing artifacts.
  - **/effect_type** - Endpoints for managing effect types.
  - **/effect** - Endpoints for managing effects.
  - **/attachments** - Endpoints for managing attachments.
  - **/search/vul_report** - Endpoint for searching vulnerability reports.

- **Features**:
  - Rate limiting.
  - API documentation using Swagger.

2.1.5 Security

- **Measures:**
  - HTTPS for secure communication.
  - Input validation to prevent SQL injection and XSS attacks.

    - Regular security audits and penetration testing.
    - Use of security headers and CSP (Content Security Policy).

2.1.6 Hosting

- **Provider**: CSC Finland.
- **Infrastructure**:
    - **Servers**: Virtual machines for hosting the backend and database.
    - **Networking**: Load balancers for distributing traffic.
    - **Storage**: Persistent storage for database and backups.
- **Features**:
    - Auto-scaling based on load.
    - Regular backups.
    - Disaster recovery plan.

3 INSTALLATION INSTRUCTIONS

3.1 System Configuration

TODO – This section will be edited later.

3.2 System Parameters

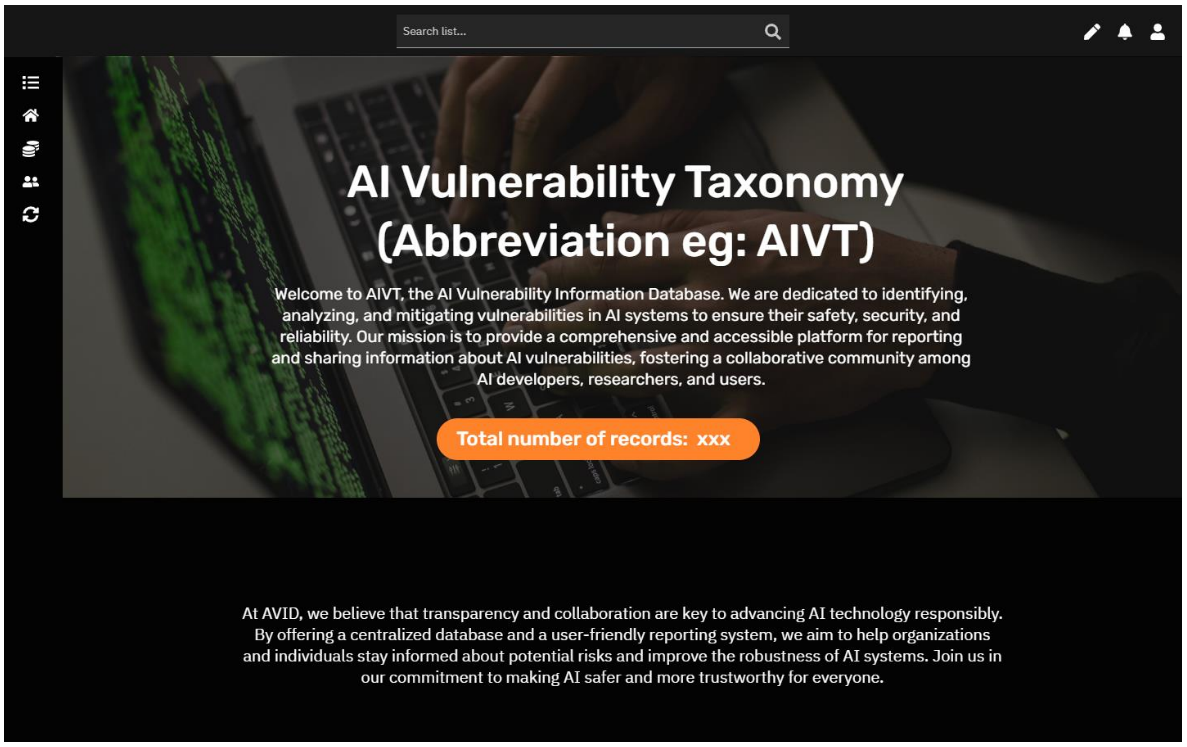
TODO – This section will be edited later.

4 DETAILED DESIGN

4.1 User Interface

4.1.1 Screens

The user interface is only planned for half of the home page until the approval.

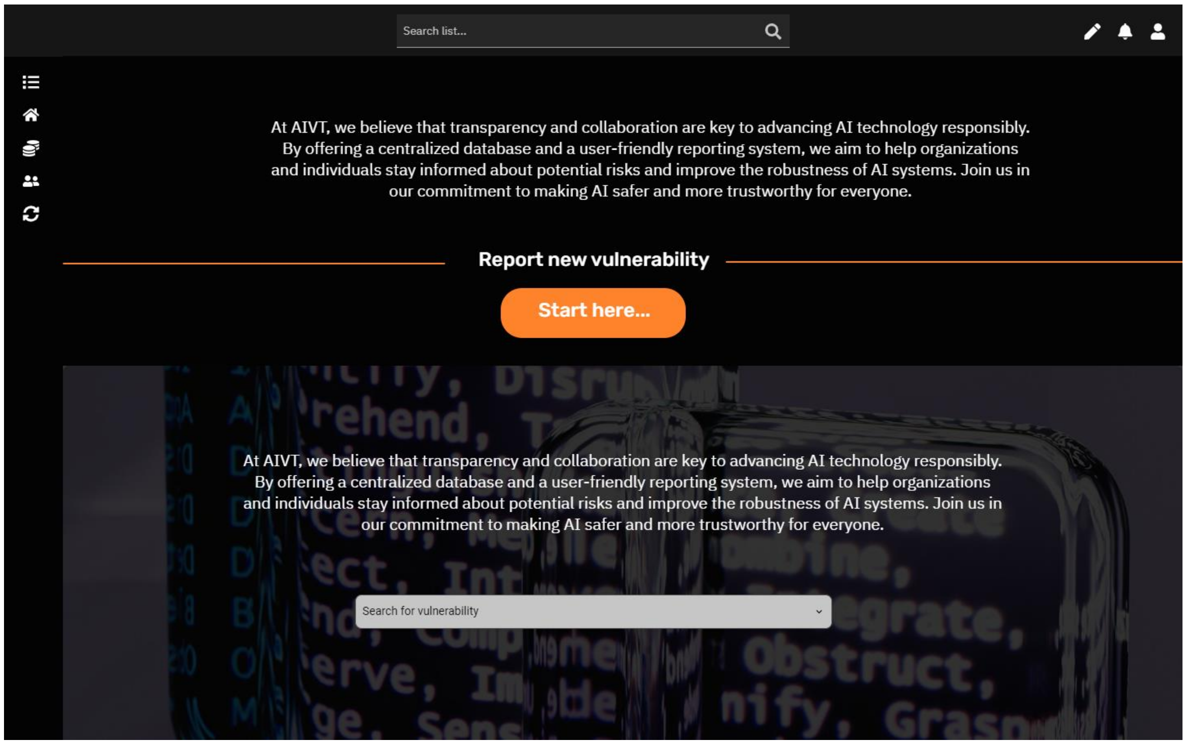

Search list...

AI Vulnerability Taxonomy (Abbreviation eg: AIVT)

Welcome to AIVT, the AI Vulnerability Information Database. We are dedicated to identifying, analyzing, and mitigating vulnerabilities in AI systems to ensure their safety, security, and reliability. Our mission is to provide a comprehensive and accessible platform for reporting and sharing information about AI vulnerabilities, fostering a collaborative community among AI developers, researchers, and users.

Total number of records: xxx

At AVID, we believe that transparency and collaboration are key to advancing AI technology responsibly. By offering a centralized database and a user-friendly reporting system, we aim to help organizations and individuals stay informed about potential risks and improve the robustness of AI systems. Join us in our commitment to making AI safer and more trustworthy for everyone.


Search list...

At AIVT, we believe that transparency and collaboration are key to advancing AI technology responsibly. By offering a centralized database and a user-friendly reporting system, we aim to help organizations and individuals stay informed about potential risks and improve the robustness of AI systems. Join us in our commitment to making AI safer and more trustworthy for everyone.

Report new vulnerability

Start here...

At AIVT, we believe that transparency and collaboration are key to advancing AI technology responsibly. By offering a centralized database and a user-friendly reporting system, we aim to help organizations and individuals stay informed about potential risks and improve the robustness of AI systems. Join us in our commitment to making AI safer and more trustworthy for everyone.

Search for vulnerability

Please note that this is half of the home page, and other pages are planned to take a similar color template and patterns. A **logo** and a **shorten name** to be confirmed before deploying the system.

4.2 Implementation Steps

4.2.1 Requirement Analysis:

- Identify all user roles and their permissions.
- Define the vulnerability data model and relationships.

4.2.2 Design:

- Create wireframes and mockups for the frontend.
- Design the database schema and API endpoints.

4.2.3 Development:

- Set up the development environment and version control.
- Develop the frontend components using React.js.
- Implement the backend logic using Node.js and Express.js.
- Design and develop the database using PostgreSQL.

4.2.4 Testing:

- Unit testing for individual components.
- Integration testing for API endpoints.
- User acceptance testing (UAT) with sample users.

4.2.5 Deployment:

- Configure the hosting environment on CSC Finland.
- Deploy the application using CI/CD pipelines.
- Monitor the application for performance and security issues.

4.2.6 Maintenance:

- Regular updates and patching.
- Continuous monitoring and logging.
- User support and feedback integration.