

AI ENTERPRISE CONFIGURATION & CODE AUDITOR

Author: © Thorsten Bylicki

Company: © BYLICKILABS

App ID / Name: ai-enterprise-configuration-code-auditor

Version: 1.0.0

1. Purpose and Business Value

The AI Enterprise Configuration & Code Auditor is a server-side analysis application for security, risk and configuration validation of software projects uploaded as ZIP archives.

The application is designed for enterprise workflows and provides:

- traceable analysis results (findings and score)
- persistent scan history (SQLite via SQLAlchemy)
- strict separation of application layers (UI, scanner, storage)
- full bilingual support (German and English)

2. Feature Set (Current State)

- ZIP project scanning via web user interface
- Calculation of a risk score and risk level
(LOW, MEDIUM, HIGH, CRITICAL)
- Detailed findings per file (file name, severity, message)

- Persistent storage of all scans in SQLite
- Multi-language support via i18n
- Server-side rendering using FastAPI and Jinja2
(no fragile SPA or client-side HTML parsing logic)

3. Architecture (High-Level)

The application follows a pure server-side rendering approach.

Components:

- FastAPI:
Web server and API endpoints
- Jinja2:
Server-side rendered user interface templates
- SQLAlchemy:
ORM for SQLite (optionally PostgreSQL later)
- Scanner module:
Extracts ZIP archives, analyzes files and produces structured findings
- Repository layer:
Central persistence layer for all database access

Important:

The user interface is fully rendered by the server.

JavaScript is used only for UI interactions

(e.g. language switching, modals).

4. Project Structure

Example project structure:

ai-enterprise-auditor/

app/

__init__.py

main.py FastAPI entry point

config.py Application metadata and default configuration

i18n.py Server-side translation helper

analysis/

__init__.py

scanner.py ZIP analysis and security scans

storage/

__init__.py

db.py SQLAlchemy engine, session and base

models.py ORM models (scans, findings, audits)

repository.py Persistence layer

ui/

templates/

index.html Main dashboard (server-side rendered)

static/

css/

style.css

js/

i18n.js

app.js

data/

app.db SQLite database (auto-generated)

logs/ Optional log files

requirements.txt

README.txt

5. Requirements / Dependencies

The application requires the following Python packages:

- fastapi
- uvicorn[standard]
- jinja2
- sqlalchemy
- python-multipart (for ZIP uploads)

6. Installation (Windows – recommended)

1. Open project directory

cd C:\Users\...\Desktop\ai-enterprise-auditor

2. Create virtual environment

```
python -m venv .venv
```

3. Activate virtual environment

```
.venv\Scripts\activate
```

4. Install dependencies

```
pip install -r requirements.txt
```

7. Installation (Linux / macOS)

```
cd ai-enterprise-auditor  
python3 -m venv .venv  
source .venv/bin/activate  
pip install -r requirements.txt
```

8. Start / Execution

The application must always be started from the project root directory:

```
python -m app.main
```

After startup, you will typically see:

Uvicorn running on http://0.0.0.0:8000

Open in browser:

`http://127.0.0.1:8000`

9. First Scan (Web UI)

1. Select a ZIP file
2. Choose scan profile (BASE, OWASP, CIS)
3. Use the API token in development mode:
`admin-dev-token`
4. Start scan

The scan result is rendered server-side.

10. Database (SQLite)

The SQLite database is automatically created on first startup.

Default path:

`data/app.db`

Reset during development:

1. Stop the application (CTRL + C)
2. Delete the database file
`data/app.db`
3. Restart the application
`python -m app.main`

11. Configuration

The database URL can be configured via environment variable.

Example:

`DB_URL=sqlite:///./data/app.db`

Optional later:

`DB_URL=postgresql+psycopg2://user:pass@localhost:5432/auditor`

12. Troubleshooting

Problem:

`ModuleNotFoundError: fastapi`

Solution:

- Activate virtual environment
- Run `pip install -r requirements.txt`

Problem:

`No module named 'app'`

Solution:

- Start the application correctly:
`python -m app.main`

Problem:

CSS or JavaScript not loading

Solution:

- Check requests in the terminal
- Perform a hard browser reload:
CTRL + SHIFT + R

Problem:

"File is not a zip file"

Solution:

- Ensure that a valid ZIP archive is uploaded

13. Roadmap (Recommended Next Steps)

- Scan history in the UI (list and detail view)
- Severity filtering and search
- JSON export and PDF reporting
- Production activation of the role model
(Admin, Auditor, Viewer)

14. Hash Stability Notice for GitHub Repositories

Hash values remain identical only when the repository is obtained

using git clone.

When using GitHub's "Download ZIP" function, GitHub automatically creates a new root directory, for example:

<repository-name>-main

<repository-name>_main

The "main" suffix is assigned automatically and modifies the original root directory name.

Technical background:

- The root directory name is part of the file path
- File paths are included in recursive hash calculations
- Changing the root directory inevitably results in different hashes

Recommendation:

- Obtain the repository via git clone
- Validate hashes based solely on defined file contents or manifests

ZIP downloads should not be used as a basis for integrity validation.

15. License

BYLICKILABS – Internal / Enterprise Usage