Virtual Assistant for Developer Dependency and Code Vulnerability

1. **Use Case**

   A virtual assistant that helps developers with package dependencies and vulnerability scanning of their Python Projects. The virtual assistant basically answers questions like “is this safe to ship?” , “For Django, list all advisories affecting my version, earliest patched version, and links.”, etc. The virtual assistant does this by scanning the dependency manifest (e.g., requirements.txt, pip freeze, poetry.lock) and checking known vulnerabilities (through tool calling).The recommended outputs include a set of safe pins and recommendation based on the vulnerability scan of the project.

2. **Data (accessed through tools)**

   1. OSV API : A distributed vulnerability database for Open Source Projects.
   2. PyPI JSON Warehouse API : provides releases, info.requires_python, and distribution URLs (wheels vs sdist) and project URLs.
   3. Standards/libs : enforce deterministic version handling via PEP 440 and dependency specification via PEP 508 using the packaging library.
   4. Baseline comparators (pip-audit CLI, OSV Scanner).

3. **LLMs**

   1. Phi-3.5-mini-instruct-GGUF
   2. Llama-2-13B-GGUF

4. **Candidate User queries**

   1. Scan my requirements.txt for vulns; propose minimal safe pins for Python 3.11.
   2. Is urllib3==1.25.8 vulnerable? Show CVEs, CVSS, and first fixed with a pip command.
   3. Given requests>=2.28,<2.31, pick the highest safe version for Python 3.11 and explain.
   4. Which deps lack wheels for linux/arm64?
   5. Identify vulnerabilities and the top-level packages pulling them in.

6. Show only issues added/updated in the last release.
7. Generate pip commands to remediate everything with CVSS ≥ 7.0.
8. Urllib3 fix conflicts with requests<2.31—explain and give two alternatives.
9. Create a constraints.txt with the first non-vulnerable versions for all packages in requirements.in.
10. Does pydantic 1.10.12 have advisories? List OSV IDs, CVEs, fixed-in.
11. Do my pins satisfy requires_python for Python 3.12? Mark incompatible ones.
12. For upgrades (e.g., FastAPI 0.92 → 0.110), include release notes links.
13. Export the scan as JSON and a Markdown summary.
14. Give me a GitHub Actions step to run pip-audit on every push.
15. Scan this poetry.lock and recommend safe, compatible upgrades.
16. Apply policy: no pre-releases, min CVSS 5.0—what pins do you recommend?
17. Here's my pip freeze; count Critical/High/Medium/Low and top 5 risky packages.
18. For Django, list advisories affecting my version, earliest patched version, and links.
19. Will upgrading numpy to 2.x break pandas 1.2 based on declared requirements? Suggest a path.
20. Scan this CycloneDX SBOM and output the top 10 risky components with proposed fixes.