**Jira Comment for User Story 2: Core Backend Infrastructure & Configuration**

**Status:** Done ✅

**Summary of Work Done:**

In this story, we brought the backend to life. The focus was on creating a runnable, observable, and configurable FastAPI service without any business logic, fulfilling all acceptance criteria.

* **Environment Setup:** A Python virtual environment (.venv) was created and activated within the /Backend directory to isolate dependencies. Core packages (fastapi[all], pydantic-settings, structlog) were installed.
* **Configuration Management:** We implemented a professional configuration setup using a .env file in the /Backend root and a Pydantic Settings model in app/core/config.py. This allows us to manage environment-specific variables cleanly.
* **FastAPI Application:** The core application was created in app/main.py. This includes the main FastAPI instance, loading configuration from the settings object.
* **Structured Logging:** A robust, production-ready logging system was implemented using structlog. It's configured to output all logs in JSON format, which is ideal for machine parsing.
* **Request ID Middleware:** A custom middleware was added to the FastAPI app. It injects a unique X-Request-ID into every incoming request and its corresponding log entries, making it incredibly easy to trace a single request's lifecycle through the system.
* **Monitoring Endpoints:** Two essential endpoints were created:
  + GET /health: Returns a 200 OK with {"status": "ok"}.
  + GET /version: Returns the current APP\_VERSION from the .env file.
* **Debugging:** We successfully diagnosed and resolved two initial startup issues:
  1. A ModuleNotFoundError was fixed by ensuring the uvicorn command was run from within the /Backend directory.
  2. A NameError was fixed by correcting the code order in main.py, ensuring the FastAPI app instance was created *before* the middleware tried to attach to it.
* **Dependency Management:** A requirements.txt file was generated from the virtual environment to lock down our dependencies.

**Deliverables:**

* **Runnable Server:** The backend can be started without errors using uvicorn app.main:app --reload from the /Backend directory.
* **Functional Endpoints:**
  + http://127.0.0.1:8000/health
  + http://127.0.0.1:8000/version
* **Structured Log Output:** The console displays JSON logs with request IDs, like this example:

JSON

{"request\_id": "4d94831b-d169-4d71-a6ce-f0875fe703c8", "method": "GET", "path": "/health", "event": "request\_started", ...}

* **Code Artifacts:** Key new files include Backend/app/main.py, Backend/app/core/config.py, and Backend/requirements.txt.

**Vibe Check:**

The backend's heart is now beating. It's not just running; it's observable and built on solid engineering principles from day one. This gives us a stable and professional platform to start building the core application features on top of.