Building a "moral compass" for an AI agent within a Django framework is a timely and technically challenging project. Given your background in Python/Django and interest in agentic AI security, you are perfectly positioned to bridge the gap between **high-level ethics** and **low-level code**.

In 2026, the industry is moving toward **Guardian Agent** architectures—where one AI system supervises another. You can implement this right in Django.

1. The "Guardian" Middleware (The Moral Firewall)

In Django, middleware usually handles things like authentication or security headers. You can create a custom **EthicsMiddleware** that intercepts the agent's proposed actions before they are executed.

- **Function:** When the agent generates a "plan" (a list of tool calls), the middleware sends this plan to a "Guardian" model (a smaller, highly-constrained LLM) to check for violations of a pre-defined "Constitution."
- **Security Benefit:** This creates a physical separation between the "reasoning" layer and the "enforcement" layer, preventing the agent from "talking its way out" of its restrictions.

2. Multi-Agent Consultation Dashboard

As you move into your quarter project, consider building a **Consultation Log**.

- **The Feature:** Instead of a single log entry, each decision requires a "Quorum" of two agents.
- **Django Implementation:**
    - **Model:** An AgentDecision model that stores the proposed_action, agent_1_reasoning, agent_2_critique, and final_consensus.
    - **View:** A real-time dashboard using **Django Channels** (WebSockets) so a human "overseer" can see the agents debating the ethics of an action in real-time.

3. "Red Button" Signals

In your security studies, you've looked at risk models. In Django, you can use **Signals** to trigger immediate lockouts.

- **The Logic:** If an agent attempts to access a "forbidden" model (like a UserFinancials table) more than X times, a post_save or pre-fetch signal can automatically revoke the agent's API keys and flag the account for human review.

4. Open-Source Tools for Inspiration

To jumpstart your project, look at these 2026 tools and how they integrate with Python:

Tool	Role in Your Project
Django AI Boost	Use this to give your agent a structured, "read-only" view of your Django models to prevent unauthorized data mutation.
Guardrails AI	A Python library you can wrap around your LLM calls to ensure outputs follow a specific "schema" (e.g., ensuring the AI never outputs PII).
Langfuse / Phoenix	Great for "Traces"—visualizing the step-by-step reasoning of your agent so you can audit *where* a moral lapse occurred.

Project Idea: The "Ethical Audit Trail"

For your quarter project, why not build a **Django-based AI Auditor**?

1. **Backend:** A Django API that receives "Action Requests" from an external agent (like a script using LangChain).
2. **Database:** Use the Django ORM to store every step of the agent's reasoning.
3. **The "Compass":** A set of Python classes that act as "Validators." For example, a PrivacyValidator that scans the agent's output for sensitive patterns using RegEx or specialized NLP before the Django view returns the response.

Since you've already completed the **URLs** part of the Django tutorial and are moving into **Templates**, would you like to see a basic **Django Model** structure for an "Audit Log" that could track these autonomous decisions?