Guardian API — Project Outline (V1, Multi-Model, Multi-Label Moderation System)

1. Project Summary

Guardian API is a multi-model, multi-label content moderation system built to demonstrate real production engineering, AI model serving, and developer experience design. It provides a single REST API that analyzes text for harmful behavior using four coordinated models:

- **Sexism Classifier** (custom LASSO model trained on your ~40k tweet dataset)
- **General Toxicity Transformer** (lightweight HuggingFace model)
- **Rule-Based Heuristics Engine** (slur detection, threat patterns, profanity lists, etc.)
- **Combined Ensemble Model** (aggregates outputs, weights scores, determines severity)

The result is a professional, production-style moderation API suitable for portfolio, demos, and real-world usage.

---

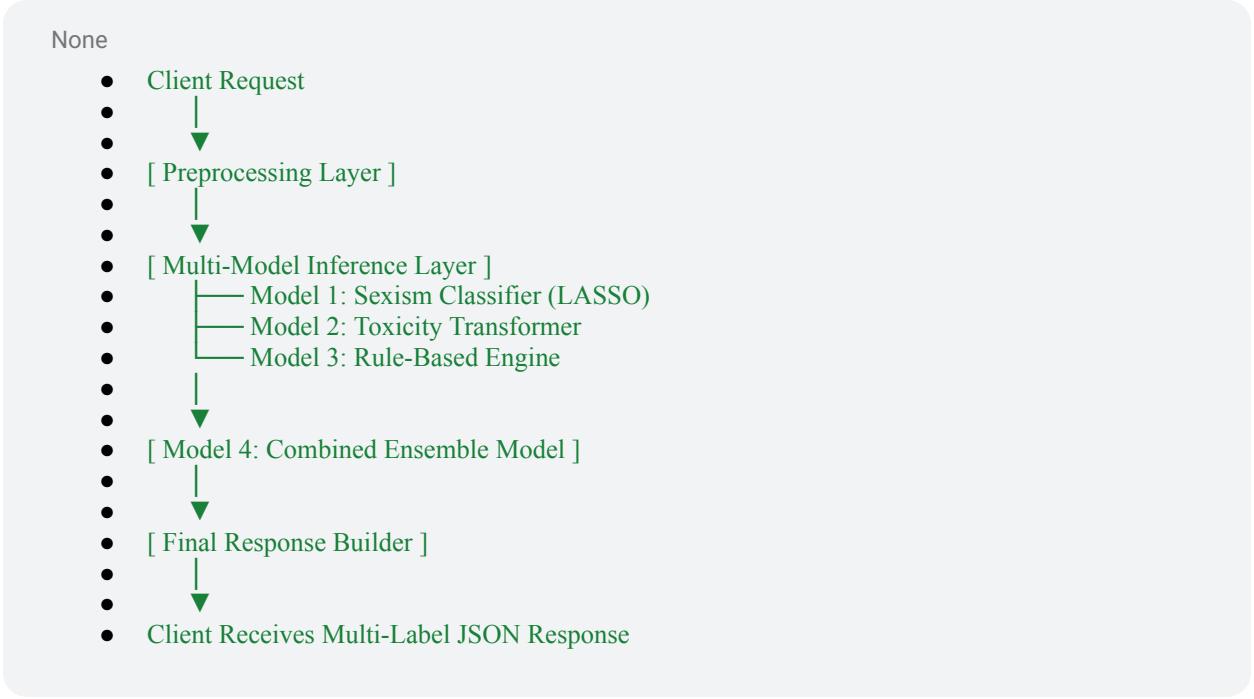
2. Core Objectives

- Provide a multi-label moderation service with rich analysis
- Showcase ability to build production systems, not just models
- Deliver an API with developer-friendly design, including docs, SDKs, and examples
- Demonstrate model serving, ensemble logic, and scalable architecture
- Keep operating costs minimal while maintaining strong feature depth
- Build a platform that can expand to additional moderation domains (v2 and beyond)

---

3. High-Level Architecture

3.1 System Architecture Overview



3.2 Components

- **Backend:** FastAPI, Python, Uvicorn
- **Models:**
    - LASSO classifier + vectorizer
    - DistilRoBERTa toxicity model
    - Custom rule engine
- **Data Store:** Upstash Redis for rate limits, SQLite or small analytics store
- **Frontend:** React + TypeScript + Tailwind + shadcn/ui
- **Docs:** Mintlify or Docusaurus
- **Hosting:** Fly.io or Render (backend), Cloudflare Pages (frontend)
- **CI/CD:** GitHub Actions

---

4. Model Breakdown

Model 1 — Sexism Classifier

- Custom LASSO model trained on ~40k labeled tweets
- Vectorizer + classifier pipeline
- Outputs:
    - probability of sexism
    - severity (derived via threshold ranges)
    - model version metadata

Model 2 — General Toxicity Transformer

- Small HF transformer (e.g. unitary/unbiased-toxic-roberta)
- Outputs multi-label scores:
    - toxicity
    - insult
    - threat
    - profanity
    - identity attack

Model 3 — Rule-Based Heuristics Engine

- Slur dictionary
- Threat phrase patterns
- Self-harm phrase list
- Profanity lists
- All-caps yelling detection
- Repeated character toxicity
- Regex-based phrase checks
- Outputs binary flags

Model 4 — Combined Ensemble Model

- Not an ML model

- Handles:
    - weighted score fusion
    - label consolidation
    - severity computation
    - conflict resolution (rules override low scores)
    - final harmfulness score
    - summary + category classification

---

5. API Features

5.1 Endpoints

- **POST /v1/moderate/text** – Accepts single text or batch, returns multi-model, multi-label analysis
- **GET /v1/models** – Returns loaded models and version metadata
- **GET /v1/health** – Health and readiness check
- **POST /v1/moderate/batch** – CSV or JSON list moderation

5.2 Response Structure

```
None
```


```
{
 "text": "...",
 "labels": {
  "sexism": {
   "score": 0.82,
   "severity": "moderate",
   "model_version": "sexism_lasso_v1"
  },
  "toxicity": {
   "overall": 0.74,
   "insult": 0.63,
   "threat": 0.12,
   "identity_attack": 0.41,
   "profanity": 0.58,
   "model_version": "toxic_roberta_v1"
  },
  "rules": {
   "slur_detected": true,
   "self_harm_flag": false,
```

```
      "profanity_flag": true,
      "caps_abuse": false,
      "model_version": "rules_v1"
    }
  },
  "ensemble": {
    "summary": "likely harmful",
    "primary_issue": "sexism",
    "score": 0.81
  },
  "meta": {
    "processing_time_ms": 24,
    "models_used": ["sexism_lasso_v1", "toxic_roberta_v1", "rules_v1"]
  }
}
```


---

6. Implementation Plan

Phase 1 — Infrastructure Foundations

- Create FastAPI project structure
- Implement preprocessing pipeline
- Load vectorizer + sexism model
- Load toxicity transformer
- Implement rule engine

Phase 2 — Ensemble & Output

- Implement ensemble logic
- Build severity + harmfulness scoring
- Construct unified JSON response
- Add error handling, logging, rate limiting

Phase 3 — Developer Experience

- Generate OpenAPI docs
- Build Mintlify doc site

- Create Python + JS SDKs
- Add usage examples

Phase 4 — Playground UI

- Build React + TypeScript interface
- Real-time moderation calls
- Visualization of model scores
- Code snippet widgets

Phase 5 — Analytics + Production Polish

- Add usage logging
- Add dashboard charts
- Implement API key system (optional v1.1)
- Build benchmarks page

---

7. Folder Structure (Recommended)

```
None
```

- guardian-api/
- backend/
- app/
- main.py
- config.py
- routers/
- moderate.py
- health.py
- models.py
- core/
- preprocessing.py
- ensemble.py
- utils.py
- models/
- sexism/
- vectorizer.pkl
- classifier.pkl
- toxicity/
- hf_model/
- rules/

```
        rules.json
      schemas/
        request.py
        response.py
    tests/
    Dockerfile

  frontend/
    src/
    public/
    components/
    pages/

  docs/
    mintlify/
    config.yml
    pages/

  sdk/
    python/
    javascript/

  benchmarks/
  scripts/
```

---

8. Key Design Principles

- Modular model loading
- Low-cost inference-first selection of models
- Transparent metadata and versioning
- Extensible architecture for future domains
- Rule engine ensures real-world reliability
- Clean developer experience
- Minimal external dependencies

---

9. V1 Deliverables Checklist

Backend

- FastAPI service
- Sexism model inference
- Toxicity model inference
- Rule engine implementation
- Ensemble fusion logic
- Multi-label response
- Rate limiting via Upstash
- Logging

Frontend

- Moderation playground
- Results visualization
- Dark/light mode
- Code example widget

Documentation

- API Reference
- Quickstart
- SDK guides
- Model cards (v1)
- Benchmarks

SDKs

- Python client
- JS/TS client

---

10. Future Extensions (V2+)

- Multilingual support
- Additional harm categories (self-harm, hate speech, cyberbullying)
- Conversation-level context
- Image moderation
- Model distillation
- A/B testing for model upgrades
- Billing + API key dashboard
-