

PRISM Project Summary

Generated: 2026-01-19T20:09:49.289933+00:00Z

Location: /Users/amoghmerudi/PRISM

What This Project Is

- AI Repo Supervisor (PRISM) with a Next.js frontend and FastAPI-based backend services.
- Backend is split into API, AI analysis, and DB services, mounted together by a root FastAPI app.
- Includes GitHub Action-style Node wrappers to run or call the backend and post PR comments.

Architecture Overview

- Frontend: Next.js app in `frontend/ai-repo-supervisor` with Tailwind styling and mock data.
- Backend aggregator: `backend/main.py` mounts `/api`, `/ai`, `/db` services.
- Backend API service: deterministic PR analysis and demo-mode response shaping.
- Backend AI service: OpenRouter-based analysis with fallback heuristics; writes SQLite health data.
- Backend DB service: MongoDB-backed PR health history + summary endpoints, in-memory fallback.

Key User Flows

- User visits landing page and navigates to dashboard for repo health cards.
- Repo detail page shows health trend chart and recent PR analysis cards (mocked data).
- GitHub Action wrapper collects PR data, lints diff, sends to backend, and posts PR comments.

Backend Endpoints (High-Level)

- /analyze-pr: accepts PR metadata and diff, returns summary/risks/suggestions/health scores.
- /health: service status endpoint for API/AI/DB services.
- /health-history: recent health history for a repo.
- /repo-summary and /repos (DB service): summary and listing for repo health metrics.

Data Storage

- SQLite: health.db files in backend-ai and backend-api for demo/local storage.
- MongoDB: backend-db uses MongoDB Atlas when `MONGODB_URI` is set; otherwise in-memory.

Tech Stack

- Frontend: Next.js 16, React 19, TypeScript, Tailwind CSS (PostCSS v4 plugin).
- Backend: FastAPI, Uvicorn/Gunicorn, Pydantic; OpenAI client for OpenRouter.
- Node: GitHub Actions tooling via @actions/core/github, plus fetch to backend.

Project Structure

- `frontend/ai-repo-supervisor`: Next.js app (pages under `app/`).
- `backend/backend-api`: FastAPI service for PR analysis; includes analyzer module.
- `backend/backend-ai`: FastAPI service for AI analysis via OpenRouter.
- `backend/backend-db`: FastAPI service for persisting and summarizing repo health.
- `backend/main.py`: mounts all backend services into a single app.

Operational Notes

- Environment variables: OPENROUTER_API_KEY, OPENROUTER_MODEL, USE_AI, DEMO_MODE, MONITORING_ENABLED.
- GitHub Action wrappers can run locally (start FastAPI) or in CI (post PR comments).
- Multiple health.db files appear to be runtime artifacts, not core source.

Suggested Next Steps

- Add real API integration to replace mock repo data in the frontend.
- Unify backend responses between API/AI/DB services to avoid format drift.
- Add automated tests for analyzers and API endpoints.