Project for Full Stack Developer Role at Madeline & Co.

SiteSage — Automated SEO Performance Analyzer

Overview

Build a **production-grade web platform** that analyzes website URLs for SEO and performance quality.
The system should crawl a given URL, extract key metadata (title, meta description, images, heading tags, etc.), evaluate basic performance metrics (load time, accessibility, best practices), and generate a structured SEO report.
An **AI-powered component** should convert raw metrics into human-readable insights and optimization suggestions.
Both the **frontend and backend** must be fully functional, modular, tested, containerized, and deployed with an automated CI/CD pipeline.
This project assesses end-to-end engineering capability — architecture, development discipline, automation, and pragmatic use of AI.

Tech Stack

- **Frontend:** Next.js, React.js, TailwindCSS
- **Backend:** FastAPI (Python), LangChain (for AI orchestration)
- **Database:** PostgreSQL with **Alembic** for migrations
- **Storage:** AWS S3 (or mock equivalent for static report files)
- **Vector DB (optional):** Qdrant (for AI embeddings if used)
- **Containerization:** Docker + Docker Compose
- **CI/CD:** GitHub Actions (preferred)
- **Testing:** Pytest / Jest with automated pipeline execution
- **Deployment:** Render / Railway / AWS / Vercel (both frontend and backend required)

Core Features (Required)

1. URL Audit & Data Collection

- User submits one or more website URLs via the frontend.
- Backend asynchronously crawls and extracts:
  - Title, meta tags, H1/H2 structure
  - Image alt tags
  - Page speed / accessibility metrics (via Python libraries or APIs like Lighthouse CI, requests-html, or aiohttp)
- All results stored in PostgreSQL.

2. SEO Analysis & Scoring

- Backend computes an SEO score based on extracted attributes.
- Includes basic checks such as missing alt tags, broken links, duplicate titles, and missing meta descriptions.

3. AI Insight Generation

- Use an LLM (via LangChain or direct API) to:
  - Summarize site quality in 2–3 paragraphs
  - Suggest 3–5 actionable improvements
- Minimal prompt engineering required.

4. Report Generation & Delivery

- Return report in structured **JSON** via API.
- Optional endpoint to **download PDF report** (bonus).
- Include SEO score, metrics, and AI-generated text.

5. Frontend Dashboard

- Modern dashboard UI with:
  - Form to submit URLs
  - Display of reports in tabular and card view
  - Responsive design
  - Loading states and proper API error handling

6. CI/CD, Testing & Deployment

- **CI/CD Pipeline:**
  - Run linting, unit tests, and integration tests on every push.
  - Auto-deploy to production once tests pass.
- **Automated Tests:**
  - Unit + integration tests for both backend and frontend.
  - AI-assisted testing tools allowed.
- **Dockerization:**
  - Both frontend and backend must run in containers using Docker Compose.

Bonus / Optional Features

- Multi-URL batch analysis with summary comparison.
- Historical report tracking (trends over time).
- Authentication and role-based access control.
- Enhanced PDF styling (charts, tables, colored sections).
- Custom AI prompt configuration in UI.

Evaluation Criteria

Category	**Description**	**Weightage**
Frontend	Responsive UI, modern design, clean component architecture, state management, and API integration.	**40%**
Backend	Modular service architecture, clean API design, Alembic migrations, AI integration, database structure, and performance.	**40%**
DevOps / Testing / CI-CD	Docker setup, test automation, CI pipeline, deployment quality, and project documentation.	**20%**
Bonus (within above)	Excellent code readability, advanced testing, polished deployment.	—

Breakdown inside Backend (40%):

- 10% Code quality & modularity
- 15% AI orchestration & insight logic
- 15% Functionality & reliability

Breakdown inside Frontend (40%):

- 10% Code structure & readability
- 30% Functionality, UX, responsiveness

Submission Guidelines

1. **Timeline:** 3 days from assignment date.
2. **Deliverables:**

- Deployed application (live URL for both frontend & backend).
- GitHub repository (public) containing:
  - Full code
  - Dockerfile(s) + Docker Compose
  - CI/CD workflow configuration
  - Unit and integration test scripts
  - Detailed **README.md** with setup instructions and API documentation
- Swagger / OpenAPI docs generated via FastAPI.

1. **Data:** Candidates must gather their own URLs for testing (no pre-provided data).
2. **Evaluation:** Manual inspection of repository, CI logs, deployment, and running application.

Deliverables Checklist

- Functional FastAPI backend with modular design
- Next.js + Tailwind responsive frontend
- Deployed production build (frontend + backend)
- Dockerized environment with Compose
- CI/CD pipeline with automated tests
- Alembic migration setup for PostgreSQL
- AI summarization integrated into reports
- Clear README + Swagger documentation