Technical Interview Exercise: AI-Enhanced Drug Information Publishing Platform

Time Commitment

Take-home portion: ~8 hours (one work day)
Follow-up interview: 1 hour code walkthrough and technical discussion

Objective

Build an **AI-enhanced drug information publishing platform** using Next.js and NestJS that processes FDA drug labels and creates SEO-optimized content pages. This exercise evaluates your:

- **AI integration for content enhancement** and optimization in production applications
- **Full-stack development skills** with Next.js frontend and NestJS backend
- **SEO optimization expertise** for high-traffic publishing platforms
- **Code quality and architecture** decisions for scalable healthcare content systems

You'll build a **working application** that demonstrates production-ready AI integration.

Background

PrescriberPoint needs to automatically generate high-quality drug information pages from FDA label data. The system should use AI to enhance content for healthcare professionals while maintaining medical accuracy and achieving excellent SEO performance.

Real-world context: Publishers need to create thousands of drug pages efficiently while ensuring content quality, SEO performance, and regulatory compliance.

Requirements

Core Functionality

1. Drug Label Processing (Straightforward JSON Processing)

- **Process provided FDA label JSON data**

- **Extract key information**: drug name, indications, contraindications, dosing, warnings, manufacturer
- **Structure content** for web presentation
- **Handle missing or incomplete data** gracefully

2. AI-Powered Content Enhancement (The Real Challenge)

- **Generate SEO-optimized titles and meta descriptions** using AI
- **Create provider-friendly explanations** of medical conditions and drug uses
- **Generate related content suggestions** (similar drugs, conditions, etc.)
- **Enhance content readability** while maintaining medical accuracy
- **Create structured FAQ sections** from label information
- **Reimagine the interface** do you want to see a page per drug or is there a better way to find content and compare

3. Publishing Platform Features

- **High-performance drug information pages** optimized for SEO (meet or exceed Core Web Vitals standards)
- **Server-side rendering** for optimal search engine indexing
- **Responsive design** for healthcare professionals
- **Content search and filtering** functionality

Technical Requirements

Full-Stack Implementation

- **Frontend**: Next.js 14+ with TypeScript, SEO optimization
- **Backend**: NestJS with TypeScript, robust API design. Data processing can be done via Python
- **AI Integration**: Use production AI APIs (OpenAI, Claude, etc.)
- **MCP Integration**: Implement Model Context Protocol interface for AI tool access
- **Database**: Store processed drug data and AI-generated content
- **Docker**: Containerized deployment with `docker-compose`

Code Quality Standards

- **Clean architecture**: Separation of concerns, maintainable code structure
- **Error handling**: Robust handling of AI API failures and data issues
- **Testing**: Unit/integration tests for core functionality
- **Performance**: Efficient content processing and caching strategies

Specific Deliverables

1. Working Application

Core Workflow:

```
FDA Label JSON → Data Processing → AI Content Enhancement → SEO-Optimized Pages
```

Key Features to Implement:

- Process drug label JSON into structured content
- AI-powered content enhancement (titles, descriptions, explanations)
- SEO-optimized drug information pages
- Search and navigation functionality

2. AI Integration Strategy

Production AI Implementation:

- **Which AI service** did you choose and why? (Cost, reliability, medical accuracy)
- **How do you handle AI reliability** in production? (Retries, fallbacks, validation)
- **What prompting strategies** work best for medical content generation?
- **How do you ensure content accuracy** and prevent AI hallucinations?

3. Next.js SEO Optimization

Technical Implementation:

- **Server-side rendering** strategy for drug pages
- **Meta tags and structured data** implementation
- **Core Web Vitals optimization** approach
- **URL structure and internal linking** strategy

4. NestJS Backend Architecture

API Design:

- **Clean API structure** for drug data and AI content
- **MCP server implementation** to expose drug data as AI tools
- **Caching strategies** for performance
- **Error handling** for AI service failures
- **Data validation** and sanitization

Implementation Guidelines

AI Content Enhancement Requirements

- Generate compelling, medically accurate page titles
- Create SEO-optimized meta descriptions under 160 characters
- Explain complex medical terms in provider-friendly language
- Generate related content suggestions based on drug categories
- Create FAQ sections answering common patient questions
- Maintain medical accuracy while improving readability

SEO Optimization Requirements

- Implement proper heading structure (H1, H2, H3) for content hierarchy
- Generate structured data markup for drug information
- Optimize page load times with efficient data fetching
- Create semantic URLs (e.g., `/drugs/taltz-ixekizumab`)
- Implement proper internal linking between related drugs

Performance Requirements

- Server-side render drug pages for SEO
- Implement caching for processed content
- Handle AI API rate limits and failures gracefully
- Optimize database queries for content retrieval
- Minimize client-side JavaScript for better Core Web Vitals

Technical Constraints

- **MCP Service**: Use production APIs with MCP protocol for AI tool access
- **Frontend**: Next.js 14+ with TypeScript, focus on SEO and performance
- **Backend**: NestJS with proper API design, MCP server, and validation
- **Deployment**: Must run with `docker-compose up`
- **Data**: You'll be provided with processed FDA label JSON files
- **Testing**: Include meaningful tests for core functionality

Evaluation Criteria

- **AI Integration:** Production-ready AI workflows with proper error handling and validation
- **Next.js Expertise:** SEO optimization, SSR implementation, performance optimization
- **NestJS Architecture:** Clean API design, proper validation, error handling
- **Code Quality:** Maintainable architecture, proper testing, documentation

- **Publishing Platform Understanding:** SEO best practices, content structure, performance optimization

Deliverables

1. GitHub Repository

- **Complete source code** with clear project structure
- **Comprehensive README** with setup instructions
- **Docker configuration** that works with `docker-compose up`
- **Tests** with reasonable coverage

2. README Must Include

- **Quick start guide** (should work in under 5 minutes)
- **AI integration decisions** and rationale
- **Architecture overview** with key technical decisions
- **SEO optimization approach** and implementation details
- **Performance considerations** and caching strategies
- **Known limitations** and potential improvements

3. Working Demo

- **Multiple drug information pages** with AI-enhanced content
-
- **Search functionality** across drugs and conditions
- **SEO-optimized markup** visible in page source

Interview Format

Code Walkthrough

- **Demo the working application** and key features
- **Explain AI integration** decisions and prompt engineering
- **Walk through architecture** and key components
- **Discuss SEO implementation** and performance optimizations

Technical Deep-Dive

- **AI reliability and error handling** implementation
- **Next.js SSR and SEO optimization** techniques
- **NestJS API design** and data validation approaches
- **Performance optimization** and caching strategies

Leadership Discussion

- **How would you lead a team** building this type of system?
- **What would you prioritize** for production deployment?
- **How would you ensure** content accuracy and compliance?

Success Indicators

A strong submission will:

- **Actually work** when we run `docker-compose up`
- **Demonstrate sophisticated AI integration** with proper error handling
- **Show deep Next.js SEO expertise** in implementation
- **Include clean NestJS architecture** with proper validation
- **Balance innovation with reliability** in technical decisions
- **Show understanding** of healthcare content requirements

Submission

- **GitHub repository link** submitted 24 hours before interview
- **Include live demo** if deployed (optional, but impressive)
- **Prepare to run locally** during the interview if needed