CookBook Connect - Backend Developer Challenge

Overview

You'll build a recipe sharing platform where users can upload their recipes, discover new dishes by searching with ingredients they have at home, and connect with other cooking enthusiasts. This project tests your ability to work with modern backend technologies including GraphQL APIs, complex database relationships, real-time features, and AI integrations.

The platform allows users to upload recipes with detailed ingredients and instructions, search for recipes based on available ingredients, get AI-powered suggestions to improve their recipes, follow other users to see their latest creations in real-time, and engage through ratings and comments.

Implementation Stages

Stage 1: Environment Setup

Set up the development environment

Requirements:

- Create NestJS application with GraphQL (Apollo Server)
- Set up PostgreSQL and Elasticsearch using Docker
- Configure Prisma ORM for database operations
- Basic project structure with proper TypeScript configuration

Deliverables:

- Working NestJS GraphQL server
- Docker containers running PostgreSQL and Elasticsearch
- Prisma schema file configured
- Basic health check endpoints

Stage 2: Core CRUD Operations

Build the fundamental data operations

Database Tables Required:

- `users` - User profiles and authentication data
- `recipes` - Recipe information and metadata

- `ingredients` - Recipe ingredients with quantities
- `instructions` - Step-by-step cooking instructions
- `ratings` - User ratings for recipes (1-5 stars)
- `comments` - User comments on recipes
- `follows` - User following relationships

GraphQL Operations to Implement:

- User management (register, login, profile updates)
- Recipe CRUD (create, read, update, delete recipes)
- Ingredient management within recipes
- Rating and commenting system
- User following/follower functionality

Complex Query Requirements:

- Get recipes with average ratings and comment counts
- Find users with most followed recipes
- Get recipe recommendations based on user's rating history
- List recipes by multiple ingredient matches
- User feed showing recipes from followed users

Success Criteria:

- All CRUD operations working via GraphQL
- Complex queries efficiently retrieving related data
- Proper error handling and validation
- Database relationships correctly implemented

Stage 3: Search & Discovery

Implement Elasticsearch for recipe search

Requirements:

- Index all recipes in Elasticsearch with ingredients and metadata
- Implement ingredient-based search functionality
- Add filtering by cuisine, difficulty, cooking time
- Create auto-complete for ingredient suggestions

Key Features:

- Full-text search across recipe titles and descriptions
- "Cook with what I have" - match recipes by available ingredients
- Advanced filtering and sorting options
- Search result ranking by relevance and ratings

Success Criteria:

- Fast search responses (under 100ms)
- Accurate ingredient matching
- Proper search result ranking
- Search analytics tracking

Stage 4: Real-time Updates

Add live features using Redis/Kafka or WebSocket

Requirements:

- Implement real-time notifications for new ratings and comments
- Live updates when followed users post new recipes
- Real-time recipe recommendation updates
- Activity feeds with live data

Technology Options:

- Redis Pub/Sub for lightweight real-time features
- Kafka for more robust event streaming
- GraphQL Subscriptions for real-time API
- WebSocket connections for instant updates

Success Criteria:

- Users see live updates without page refresh
- Efficient real-time data delivery
- Proper error handling for connection issues
- Scalable real-time architecture

Stage 5: AI Enhancement

Integrate AI for recipe suggestions

Requirements:

- Integrate OpenAI or similar AI service
- Generate recipe improvement suggestions
- Provide ingredient substitution recommendations
- Create cooking tips and technique suggestions

Features:

- Analyze existing recipes and suggest improvements
- Recommend ingredient substitutions for dietary restrictions
- Generate cooking tips based on recipe complexity
- Suggest wine pairings or side dishes

Success Criteria:

- AI suggestions are relevant and helpful
- Proper error handling for AI service failures
- Reasonable response times (under 5 seconds)
- Cost-effective API usage with caching

Technical Requirements

Core Technologies

- **Backend:** NestJS with TypeScript
- **API:** GraphQL with Apollo Server
- **Database:** PostgreSQL with Prisma ORM
- **Search:** Elasticsearch
- **Real-time:** Redis/Kafka or WebSocket
- **AI:** OpenAI API or similar service

Performance Expectations

- GraphQL queries respond within 200ms
- Search queries complete within 100ms
- Database operations optimized with proper indexing

- Support concurrent users efficiently

Architecture Standards

- Clean code with proper TypeScript typing
- Separation of concerns (services, resolvers, repositories)
- Proper error handling and validation
- Scalable folder structure
- Environment-based configuration

Evaluation Criteria

Technical Implementation (70%)

- Code quality and TypeScript usage
- Database design and query optimization
- GraphQL schema design and resolver efficiency
- Integration quality with external services
- Error handling and edge case management

Problem Solving (20%)

- Approach to complex query requirements
- Real-time feature implementation strategy
- AI integration and optimization
- Performance optimization techniques

Documentation (10%)

- Clear setup instructions
- API documentation quality
- Code comments and explanations
- Architecture decision documentation

Deliverables

Required

- Complete NestJS application with all stages implemented
- README with setup and run instructions

- GraphQL schema documentation
- Docker configuration for easy deployment
- Database migration files

Bonus

- Unit tests for core business logic
- Performance benchmarking results
- Creative feature additions
- Optimization strategies documentation

Note: Prioritize completing working features over adding extra functionality. We want to see clean, functional code that solves the requirements effectively.