MPEC Programming Test

Mathematical Proof Explanatory Chain

Fullstack TypeScript Developer Position

May 25, 2025

Welcome to the MPEC Programming Challenge

This is a 1-day coding challenge designed to assess your fullstack TypeScript development skills. You will be implementing a simplified version of our Mathematical Proof Explanatory Chain (MPEC) system that processes mathematical content and creates knowledge graphs.

1 Overview

The MPEC system is an AI-assisted mathematical proof analysis tool that:

- Processes mathematical course content in LaTeX format
- Extracts knowledge graphs from mathematical proofs
- Creates explanatory chains for mathematical calculations
- Visualizes mathematical reasoning as interactive graphs

2 Your Mission

You will implement a web application with the following workflow:

1. **Course Pattern Extraction**: Process LaTeX mathematical course content to extract a knowledge graph pattern
2. **Example Analysis**: Apply the extracted pattern to a math example to create an explanatory chain
3. **Test Question Solving**: Use the pattern and example to solve new test questions with step-by-step explanations

3 Technology Requirements

Technology Stack

- **Backend**: NestJS with TypeScript
- **Frontend**: Next.js with TypeScript
- **Database**: In-memory storage (no external database required)
- **AI Integration**: Mock OpenAI responses (no API key needed)
- **Visualization**: Any responsive graph library of your choice

4 Core Functionality

4.1 Backend API Endpoints

Implement three REST API endpoints:

1. `POST /api/extract-course-pattern`
2. `POST /api/apply-pattern-to-example`
3. `POST /api/solve-test-question`

Each endpoint should return structured knowledge graphs with entities and relations representing mathematical concepts and their relationships.

4.2 Frontend Interface

Create a responsive web application with:

- Input areas for LaTeX mathematical content
- Processing buttons to trigger the three-step workflow
- Interactive graph visualization of results
- Step-by-step explanatory chains display
- Error handling and loading states

5 Sample Mathematical Content

Code Example

Course Content Example (LaTeX):

```
\section{Recursive Definition}
For non-negative integers \(a\) and \(b\):
\[
    a + b = \begin{cases}
        a                 & \text{if } b = 0 \\
        (a + (b - 1)) + 1 & \text{if } b > 0
    \end{cases}
\]
```

Code Example

Example Problem (LaTeX):

```
\section{Example: \( 3 + 2 \)}
\[
    \begin{aligned}
        3 + 2 & = (3 + 1) + 1 \\
              & = ((3 + 0) + 1) + 1 \\
              & = (3 + 1) + 1 \\
              & = 4 + 1 = 5
    \end{aligned}
\]
```

6 Expected Deliverables

1. **Complete Source Code**: Both backend and frontend implementations
2. **README.md**: Clear setup and run instructions
3. **Working Demo**: Application that runs locally
4. **Documentation**: API documentation and component descriptions
5. **Tests**: Unit tests for core functionality

7 Evaluation Criteria

Your submission will be evaluated on:

- **Functionality (30%)**: All required features work correctly
- **Code Quality (25%)**: Clean, maintainable, well-structured TypeScript code
- **UI/UX Design (20%)**: Responsive, intuitive, and visually appealing interface
- **Graph Visualization (15%)**: Creative and effective graph representation
- **Technical Implementation (10%)**: Error handling, performance, testing

8 Time Allocation Suggestion

- Backend API Development: 3-4 hours
- Frontend Implementation: 3-4 hours
- Graph Visualization: 2-3 hours
- Testing and Documentation: 1-2 hours
- Integration and Polish: 1 hour

9 Getting Started

1. Create a new NestJS project for the backend
2. Create a new Next.js project for the frontend
3. Review the provided mock data and examples
4. Implement the core functionality step by step
5. Focus on creating a working demo first, then polish

Important Notes

- **No OpenAI API Key Required**: Use the provided mock responses
- **No Database Setup**: Use in-memory storage for simplicity
- **Focus on Core Features**: Implement the main workflow first
- **Be Creative**: Show your skills in graph visualization and UI design

10 Bonus Opportunities

Impress us with:

- Advanced graph visualization features (animations, interactions)
- Additional mathematical operations (multiplication, subtraction)
- Exceptional code quality and testing
- Performance optimizations
- Accessibility considerations

11 Submission Instructions

1. Create a Git repository with your complete solution
2. Include a comprehensive README.md with setup instructions
3. Ensure both backend and frontend can run simultaneously
4. Provide a brief explanation of your design decisions
5. Submit the repository link or compressed archive

Good luck! We're excited to see your implementation.

For questions or clarifications, please contact the hiring team.