AI Subtitles MVP

Senior TypeScript Developer Assessment

Fluent AI • Maximum 3 Hours

MANDATORY REQUIREMENT

This assignment MUST use the Effect framework. Solutions not implementing Effect will be automatically rejected, regardless of functionality.

See Annex A for Effect code examples and patterns.

1 Project Overview

You will develop a TypeScript CLI application that generates AI-powered subtitles from YouTube videos using external speech-to-text APIs. This challenge evaluates your ability to architect type-safe, maintainable solutions using functional programming principles.

Career Opportunity: Success in this assessment leads to a senior position developing our multi-platform subtitle generation system, enabling millions of learners to access YouTube and Netflix content with accurate AI-generated subtitles in their target languages.

2 Technical Specification

2.1 Required Technology Stack

- **Runtime:** Bun (latest stable version)
- **Language:** TypeScript (strict mode enabled)
- **Framework:** Effect (Effect Documentation) (Effect Tutorial)
- **Speech-to-Text:** External API of your choice (OpenAI Whisper API, AssemblyAI, Google Cloud Speech-to-Text, etc.)

2.2 Interface Definition

CLI Interface:
`bun run dev <youtube-url>`
Output Format: JSON array conforming to the SubtitleToken interface

```
type SubtitleToken = {
  id: number;
  value: string;
  startTimeMs: number;
  endTimeMs: number;
  score: number; // Confidence score from API
```

```
};

type SubtitleResult = Array<SubtitleToken>;
```

Listing 1: Required TypeScript Interfaces

3 Implementation Requirements

3.1 Core Functionality

1. **YouTube Audio Processing**
   - Extract audio stream from YouTube URL
   - Convert to format compatible with chosen API
   - Handle various video qualities and formats
2. **Effect-based Architecture**
   - Implement services using `Effect.Service`
   - Use `Context` and `Layer` for dependency injection
   - Handle configuration with `Config` module
   - Implement proper error handling with tagged errors
3. **API Integration**
   - Process audio through external speech-to-text service
   - Handle rate limiting, retries, and network failures
   - Transform API responses to required format
4. **CLI Implementation**
   - Accept YouTube URL as command-line argument
   - Validate input URL format
   - Output properly formatted JSON
   - Implement comprehensive error reporting

3.2 Effect Framework Requirements

Mandatory Effect Patterns:

- Use `Effect.gen` for async operations
- Implement services with dependency injection
- Create proper `Layer` composition
- Handle errors using tagged error types
- Use `Schema` for data validation
- Implement `Config` for environment variables

4 Deliverables

1. **Source Code:** Complete TypeScript implementation using Effect
2. **README.md:** Comprehensive documentation including:
   - Project setup and installation steps
   - Environment configuration (API keys, etc.)
   - CLI usage examples with sample outputs
   - Architecture overview explaining Effect usage
   - Troubleshooting guide for common issues
3. **Configuration:** Proper `tsconfig.json` and `package.json`
4. **Tests:** Basic test coverage demonstrating Effect testing patterns

5 Bonus Considerations

- Advanced Effect patterns (custom operators, resource management)
- Comprehensive error recovery strategies
- Performance optimizations for large video files
- Elegant handling of API rate limits with exponential backoff
- Clean separation of concerns in service architecture
- Innovative use of Effect's concurrency features

Ready to showcase your Effect expertise?

We look forward to reviewing your implementation.