



SOFTWARE ENGINEERING TAKE-HOME TASK (AI-ASSISTED DEVELOPMENT)

OBJECTIVE

Build a backend service for managing private market funds and investor commitments using AI tools to accelerate your development process.

AI USAGE EXPECTATION

We encourage and expect you to leverage AI tools such as GitHub Copilot, ChatGPT, Claude, or similar throughout this task. We know candidates will use these tools regardless, so we want to create a level playing field by encouraging everyone to use them equally. More importantly, this mirrors how our engineering team works at Titanbay, we believe AI tools are force multipliers that help engineers focus on architecture, problem-solving, and design decisions rather than boilerplate code. We're interested in seeing how you leverage these tools effectively to build quality software quickly.

REQUIREMENTS

Implement a RESTful API following the provided specification:

<https://storage.googleapis.com/interview-api-doc-funds.wearebusy.engineering/index.html>

Technical Requirements

- Database: PostgreSQL with proper schema design and relationships
- API: RESTful endpoints with JSON responses
- Validation: Input validation and proper error handling
- Language/Framework: Your choice (we work with TypeScript but use what you're strongest in)

TIME EXPECTATION

2-3 hours over 1-2 days. Don't spend more time than this - we value your time and want to see what you can accomplish efficiently with AI assistance.

EVALUATION CRITERIA

We're evaluating your ability to deliver working software efficiently while making sound engineering decisions. We care more about the quality of your implementation choices, code organization, and how you handle real-world concerns than checking boxes on a feature list. The core requirements below are table stakes, what distinguishes great submissions is thoughtful decision-making evident in the code and documentation.

Core

- Functionality: All 8 endpoints working correctly
- Data integrity: Proper database relationships and constraints
- Code quality: Clean, readable, and well-organized code
- Documentation: Clear setup instructions
- Error handling: Graceful handling of invalid requests and edge cases

Bonus

- Testing: Unit tests or integration tests
- Best practices: Following REST conventions, HTTP status codes

SUBMISSION

1. Create a public GitHub repository (or another Git provider)
2. Include a comprehensive README with:
 - Setup and run instructions
 - API endpoint documentation or reference to spec
 - Any assumptions or design decisions made
3. Ensure the application can be run locally with minimal setup
 - Include a simple way to run your application locally
 - Provide sample data or seed scripts if helpful

QUESTIONS?

If you have any questions about the requirements or need clarification, please reach out to us.

We look forward to reviewing your submission!