**🧪 Technical Challenge – Fund Administration API**

**Context**

You are joining a team responsible for building a backend API for a Fund Administration platform. This platform manages investment funds, investors, and their financial transactions (subscriptions and redemptions).

Your task is to implement a clean, scalable, and secure **ASP.NET Core Web API** using modern development practices and the **Repository Pattern**.

**🧩 Business Domain**

The API should support:

* **Funds**: Investment vehicles with basic metadata.
* **Investors**: Individuals investing in funds.
* **Transactions**: Financial operations linked to investors and funds.

**🗃️ Entity Definitions**

**Fund**

* FundId: Unique identifier
* FundName: (e.g. Global Ventures Fund)
* CurrencyCode: (e.g., USD, EUR)
* LaunchDate: Date the fund was launched

**Investor**

* InvestorId: Unique identifier
* FullName: Investor's full name
* Email: Investor's email address
* FundId: Foreign key referencing a Fund

**Transaction**

* TransactionId: Unique identifier
* InvestorId: Foreign key referencing an Investor
* Type (enum): Subscription or Redemption
* Amount (decimal): Must be positive
* TransactionDate: Date of the transaction

**🛠️ Requirements**

**Core Functionality**

* CRUD operations for Funds and Investors
* POST endpoint to register Transactions
* GET endpoints to:
  + Retrieve all transactions for a specific investor
  + Get total subscribed and redeemed amounts per fund

**Architecture**

* Use the Repository Pattern
* Apply DTOs
* Use Dependency Injection

**Validation & Error Handling**

* Validate input data (e.g., positive transaction amounts)
* Global exception handling middleware
* Standardized error responses using ProblemDetails (RFC 7807)

**Documentation**

* Swagger/OpenAPI with XML comments

**Security**

* JWT Authentication (mocked or real)
* HTTPS enforcement

**Testing**

* Unit tests for at least one service and one repository
* Use mocking for data access abstractions

**🐳 Bonus Features (Optional)**

* API versioning
* Health check endpoint
* Integration of Serilog for structured logging

**📊 Reporting Endpoint (Bonus)**

Create an endpoint that returns:

* Net investment per fund (subscriptions minus redemptions)
* Number of investors per fund

**🧪 Evaluation Criteria**

Your submission will be evaluated based on:

* Code clarity and organization
* Use of modern .NET development practices
* API design and REST principles
* Test coverage and maintainability
* Security and validation
* Documentation and usability

**🚀 Getting Started**

You may use any tools or libraries you’re comfortable with. Once completed, please submit your solution via GitHub or as a downloadable archive.