Main Component

1. Backend Services

The backend processes transactions, interacts with the blockchain, and ensures off-chain storage.

Key Responsibilities:

* Processing Investments & Redemptions:
* Input validation (e.g., ensure usdAmount > 0 and shares > 0).
* Call smart contract methods (invest, redeem) via Web3/Ethers.js library.
* Handle blockchain transaction failures (e.g., gas estimation errors).
* Tracking Fund Metrics:
* Retrieve fund metrics (e.g., totalAssetValue, sharesSupply, sharePrice) directly from the smart contract using polling or event listeners.
* Cache the results in Redis for performance optimization.
* Record Transaction History:
* Store every blockchain interaction (Investment, Redemption) in a PostgreSQL database.
* Record additional metadata (e.g., timestamps and investor details) for compliance and reporting.

Backend Components

* Blockchain Interface: Middleware for reliable blockchain communication.
* Scheduler: Sync fund metrics periodically in case of event listener failure.

2. Blockchain Layer

The smart contract provided in the requirements (IFundToken) is the core of the fund.

All state (e.g., investments, redemptions, fund metrics) is recorded immutably on-chain.

Key Functions:

* invest(address investor, uint256 usdAmount)

Investors send funds to purchase shares.

Emits Investment event on successful investment.

* redeem(address investor, uint256 shares)

Investors receive USD equivalent for requested shares.

Emits Redemption event.

3. Off-Chain Data Storage

An off-chain PostgreSQL database is used to store historical and transactional data.

Table:

Transaction

id: Auto-incremented.

type: investment or redemption.

investor\_id: Wallet address (foreign key).

usd\_amount: USD invested/redeemed.

shares: Number of shares involved.

timestamp: Time of the transaction.

4. Cache (Redis)

- Purpose:

Store frequently accessed metrics (e.g., fund value, share price) to reduce blockchain interaction latency.

- Cache Strategy:

* Cache smart contract outputs for metrics like getFundMetrics() or getSharePrice.
* Cache TTL (Time-to-Live) set to 1-5 minutes to ensure freshness.

Basic Data Models

* Transactions

id: Auto-incremented.

type: investment or redemption.

investor\_id: Wallet address (foreign key).

usd\_amount: USD invested/redeemed.

shares: Number of shares involved.

timestamp: Time of the transaction.

* FundMetrics

totalAssetValue: Total Asset Value while investing and redeem

sharesSupply: Total Shares supply while investing and redeem

lastUpdateTime: Last time you updated

How to handle Blockchain interactions

There are 2 components to handle blockchain interactions.

Smart contract and Backend Service

* Transaction Management

Send Transactions

Handle gas estimation and retries

* Query Management:

Fetch fund metrics and share price

* Error Handling

Handle RPC errors, transaction failures, and timeouts

* Event Listening

Listen to Investment and Redemption events for real-time updates

Example Sequence Diagram

User -> Backend: Investment Request

Backend -> Blockchain: Call invest()

Blockchain -> Backend: Emit Investment Event

Backend -> User: Confirmation

User -> Backend: Redemption Request

Backend -> Blockchain: Call redeem()

Blockchain -> Backend: Emit Redemption Event

Backend -> User: Confirmation

User -> Backend: Fund Metrics Request

Backend -> Blockchain: Call getFundMetrics()

Blockchain -> Backend: Return Metrics

Backend -> User: Display Metrics