Mira Finance Tech Specs

Big Picture:
Anyone can create an index fund or invest in someone else's index fund using the protocol built in Move on Mira Finance.

These index funds are built as investment pools, owned and managed by resource accounts within the contract that limit updates to the fund strategy (using signer capability) and withdrawals (using a map of investors).

Everything is built on Move, so for now only tokens that can be stored in Move are compatible, as they are stored under the resource account living on-chain (on Aptos, Sui, and any other move-based chain we deploy on).

When a user wants to invest in a fund, they can invest using any compatible token. For now, this includes all tokens on Move, and in the future, it will be easy to integrate wallets on other chains, with a small contract to bridge those assets to interact with the protocol. A similar process will take place to store cross-chain assets within an investment pool.

Example:
Alice has $100 of APT in her wallet, and decides she wants to invest in Billy's Layer1 Index Fund. This fund consists of USDT, BTC, and ETH, in equal proportions. The fund charges a 1% management fee, rebalances monthly, and allows for rebalances on investment*.

She goes on our dApp, connects her wallet, and invests her funds. It is deposited into the investment pool managed by a resource account, where all of the investments sit.

*Because the fund allows for rebalances on investment, part of the gas cost of Alice's investment is spent to rebalance the portfolio immediately. If this weren't the case, her funds would sit as APT until the next rebalance, misaligning the fund. The rebalance triggers our swap optimization, where the excess token (APT) is traded for the underlying assets (USDT, BTC, and ETH).

In the investors map, Alice’s stake in the fund is recorded as a proportion of total investment. Her stake is then reduced - to a value of $99 - because Billy collects a %1 management fee. If she keeps her investment in the fund, and the value remains at $99, Billy will collect a $0.99 fee the next year, and so on.

When Alice goes to withdraw, she can only withdraw her stake of the fund, which is checked against the investor map stored in the resource account. She can choose to withdraw her stake in any token (APT, BTC, ETH, and so on), and her stake of the fund will go through the reverse process of swaps. Her profitability will mirror that of the fund, save for management fees.

How LayerZero can help:
This protocol allows funds to be built on our dApp really nicely, but we want Mira’s Index Funds to be tokenized so that we can sell them on other exchanges, not just our dApp.

Let’s say we launch an Aptos Ecosystem Fund, and call the token AEF.

In this scenario, instead of having users invest and withdraw APT/BTC/ETH…, they’ll be swapping their APT/BTC/ETH for AEF, and vice versa.

A token purchase will trigger the deposit process, and a token sale the opposite. Because the token is fully collateralized by the underlying assets, the only difference in value between fund AUM and token market cap will be Mira’s collected management fees.

We’ll be able to support direct swaps on the dApp without any liquidity or market-making. We can rely on the protocol to collect funds for a token purchase and generate funds for a token sale, reflecting the most accurate value for AEF. When the token is listed on other exchanges, we can capture any arbitrage that emerges ourselves using bots or let the market do so.

We’re excited (if it can be done) to build an OTC token with LayerZero to launch our index fund products!

Creating a fund:

- *Manager specifies parameters for their fund:*
  - pool_name: String,
  - token_allocations: vector<u64>,
    - // list of index allocation
  - token_names: vector<String>,
    - // list of index_name
  - management_fee: u64,
    - // percentage of investments that are allocated from investors to manager each year
    - // for example: management fee set at 1% - newuser invests $100; first year, 1% of new user's ownership of
    - // pool is transferred to manager (equivalent of $1.00), next year, 0.99% (equivalent of $0.99)
  - rebalancing_period: u64,
    - // how often rebalancing is scheduled (0 - 730 days)
  - minimum_contribution: u64,
    - // minimum amount an investor can contribute to the portfolio (0 - 730 days)
  - minimum_withdrawal_period: u64,
    - // minimum amount of time before an investor can withdraw their funds from the portfolio
  - referral_reward: u64,
    - // percentage of management fee earnings that will be spent on referral rewards
    - // for example: referral reward set at 10%, and management fee at 1%
    - // user recruits a new investor, who invests $100 - manager now earns 0.9% on their investment, and user 0.1%
    - // first year, manager earns $0.90, user earns $0.10, and so on
  - privacy_allocation: u8,
    - // 1, public: any user can see portfolio on leaderboard

        - // 2, private distribution: like public, except token allocation is hidden unless user has invested
        - // 3, hidden: portfolios don't show up on site, can only be viewed when special link is shared
    - gas_percentage: u64,
        - // percentage of manager's investment that is set aside for rebalancing costs
    - whitelist: Table<u64, address>,
        - // only users who are on the whitelist can invest in the portfolio
    - rebalance_on_investment: u8 // determines whether a new user investing auto-rebalances the portfolio (incorporating rebalance into their gas cost of investing)
        - // if this is on, it also means that any user can rebalance the portfolio themselves at any time through the contract.

- *Other fields are stored to manage ownership, rebalancing, and fees*
    - time_created: u64,
    - //in seconds
    - pool_address: address,
    - // address of resource signer account that owns the pool
    - manager_addr: address,
    - investors: TableWithLength<address, u64>,
    - // map of investor's address and amount
    - investor_funds: u64,
    - //total_amount
    - gas_funds: u64,
    - //amount for gas

Updating a Fund:

- *Manager can update limited parameters*
    - pool_name: vector<u8>,
    - token_names: vector<String>,

    - token_allocations: vector<u64>,
    - management_fee: u64,
        - // fee can only be decreased, not increased
    - rebalancing_period: u64,
    - minimum_contribution: u64, add commented fields in update
    - referral_reward: u64,
    - gas_percentage: u64,
    - whitelist: Table<u64, address>

Investing/Withdrawing:

- pool_name: vector<u8>,
- pool_owner: address,
- amount: u64

Other Functions:

- send_funds_to_user(sender: &signer, recipient: address, amount: u64)
    - // for peer to peer transfers
- rebalance(signer: &signer, manager: address, pool_name: vector<u8>)
    - // updates real fund allocation to match token_allocations parameter
- swap(signer: &signer, coin1: String, coin2: String, amount: u64)
    - // swap optimization to compare tcosts + speed across different AMMs
- change_gas_funds(manager: &signer, pool_name: vector<u8>, amount: u64, add_or_remove: u8)
    - // gas pool is automatically separated from manager's initial deposit when creating fund (using gas_percentage value), but if amount is overestimated or pool runs out then manager can add/remove to ensure rebalances occur
- transfer_manager(current_owner: &signer, new_owner: address, pool_name: vector<u8>)
    - // manager can transfer management privileges to someone else

- `repossess(admin: &signer, user: address, pool_name: vector<u8>)`
    - `// if manager neglects their portfolio (lets gas pool run empty and doesn’t refill for 3 rebalances, Mira admin account will automatically transfer ownership, cover gas expenses to rebalance, and adjust management fee to 1%)`