Quantstamp DRAFT

# Epoch Island - LP Pool

## Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

| | |
|---|---|
| Type | LP Pool |
| Timeline | 2023-10-30 through 2023-11-07 |
| Language | Solidity |
| Methods | Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review |
| Specification | Manifesto<br>Website |
| Source Code | • Moai-Labs/upside-contracts #7626a37<br>• Moai-Labs/upside-contracts #5ce8b75<br>• Moai-Labs/upside-contracts #52e6377<br>• Moai-Labs/upside-contracts #da1ea72 |
| Auditors | • Adrian Koegl Auditing Engineer<br>• Roman Rohleder Senior Auditing Engineer<br>• Poming Lee Senior Auditing Engineer |

| | |
|---|---|
| Documentation quality | Medium |
| Test quality | Medium |
| Total Findings | 14<br>Fixed: 6 Acknowledged: 6<br>Mitigated: 2 |
| High severity findings | 0 |
| Medium severity findings | 3<br>Fixed: 2 Acknowledged: 1 |
| Low severity findings | 6<br>Fixed: 3 Acknowledged: 2<br>Mitigated: 1 |
| Undetermined severity findings | 0 |
| Informational findings | 5<br>Fixed: 1 Acknowledged: 3<br>Mitigated: 1 |

## Summary of Findings

The audited contract facilitates a platform where liquidity providers ("LPs") can offer their upside tokens in exchange for downside tokens. This exchange is open to takers (any user), who can accept these offers, providing the required downside tokens to receive the upside tokens. The protocol incorporates a fee structure that benefits both the liquidity providers and the protocol itself. Additionally, there are mechanisms for reversibility; takers have the option to revert their exchange within a specific timeframe, and liquidity providers can partially cancel their offers.

While the code is generally well-written and structured, a significant risk has been identified concerning interactions with unknown ERC-20 tokens. The potential for re-entrancy attacks and compatibility issues with certain ERC-20 contracts presents a threat that could disrupt the protocol's operations.

**Update:** Following the fix review, the auditing team confirmed that all issues have been either fixed, mitigated, or acknowledged by the developer team. Furthermore, the code and documentation quality were improved by the developer team. Most critically, ELP-1 was acknowledged, considering that the front-end will whitelist tokens to be used with the contract. Furthermore, the branch coverage was reduced to 85.71% in the fix review commit.

**Update:** In the second fix review, the client resolved ELP-2 and fully resolved ELP-3. We identified the new issue ELP-14 in the code changes.

| ID | DESCRIPTION | SEVERITY | STATUS |
|---|---|---|---|
| ELP-1 | **Specially designed ERC20 tokens and malicious ERC20 tokens can cause loss of funds** | Medium | Acknowledged |
| ELP-2 | **Lack of Economic Incentive for Taker** | Medium | Fixed |
| ELP-3 | **Possible Reentrancy Attacks** | Medium | Fixed |

| ID | DESCRIPTION | SEVERITY | STATUS |
|---|---|---|---|
| ELP-4 | Insufficient User Documentation | Low | Mitigated |
| ELP-5 | Ownership Can Be Renounced | Low | Acknowledged |
| ELP-6 | Critical Role Transfer Not Following Two-Step Pattern | Low | Acknowledged |
| ELP-7 | Unlocked Pragma | Low | Fixed |
| ELP-8 | Use of unsafe ERC20 transfer functions and not checking return values | Low | Fixed |
| ELP-9 | Hard-coded values limit flexibility | Informational | Acknowledged |
| ELP-10 | Application Monitoring Can Be Improved by Emitting More Events | Informational | Mitigated |
| ELP-11 | Inefficient gas usage | Informational | Fixed |
| ELP-12 | Missing input validation | Informational | Acknowledged |
| ELP-13 | Privileged roles and ownership | Informational | Acknowledged |
| ELP-14 | Inconsistency in Token Amount Calculations for Large Input Amounts Relative to Liquidity | Low | Fixed |

# Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

**Disclaimer**

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

**Possible issues we looked for included (but are not limited to):**

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

**Methodology**

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:

1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

# Scope

**Files Included**

`EpochUpsidePoolV1.sol`

# Findings

## ELP-1

### Specially designed ERC20 tokens and malicious ERC20 tokens can cause loss of funds

Medium | Acknowledged

> **Update**
> The client provided the following explanation:
>
> > These contracts are not designed to be used with tokens that rebase or change balance over time. There will be a whitelist of tokens implemented on the frontend which enables certain verified tokens to be used with the contract.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** The protocol's current architecture does not account for the unique behaviors of various specialized ERC20 tokens, including interest-bearing, deflationary, and rebasing tokens. These tokens are characterized by their dynamic balance mechanisms, where the token balance can autonomously change over time due to factors like transaction fees, re-distribution, or balance adjustments. This aspect contradicts the protocol's underlying assumption of static token balances, potentially leading to inaccurate calculations or unintended interactions within the contract.

Additionally, the protocol's open design, which allows liquidity providers (LPs) to utilize any ERC20 token, poses a significant risk. This unrestricted approach could enable LPs to introduce malicious tokens, crafted specifically to exploit the contract's mechanics or defraud takers. Such tokens might be engineered to perform a rug-pull – abruptly devaluing or making the tokens non-transferable after a transaction – or to otherwise deceive and manipulate the takers. This vulnerability necessitates a more stringent approach to token vetting and validation within the contract, to safeguard against these deceptive strategies and ensure the integrity of transactions.

**Recommendation:** Consider a well-formulated whitelisting mechanism and carefully permit tokens.

## ELP-2 Lack of Economic Incentive for Taker

Medium | Fixed

> **Update**
> The client provided the following explanation:
>
> > We believe there is still sufficient incentive for the taker to take up a position

> **Update**
> The client identified the issue of users not being able to fully take a position due to linearly decreasing fees. To fix this issue, they implemented a fixed percentage fee. As a side effect, this also resolves this issue.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** The taker has no economic incentive to act upon desired positions immediately. The reason is that the protocol and LP decrease towards the position end. As a result, the conditions for the positions become better for the taker over time.

If a taker acts ideally in terms of game theory, they would wait with the trade as long as possible. Therefore, their ideal strategy would be to observe the mempool until they see a transaction that would prevent their transaction and front-run it. A transaction prevents the takers transaction if it reduces the available `downsideToken` amount below the desired amount.

**Recommendation:** Consider what economic incentives should be created with the fee mechanism. We recommend employing a mechanism that incentivizes trades as soon as possible to promote liveliness of the protocol.

## ELP-3 Possible Reentrancy Attacks

Medium | Fixed

**Update**
The client implemented the checks-effects-interaction pattern in all the listed functions. One inconsistency remains in the `remove()` function.

**Update**
In the second fix review, the client has implemented the CEI pattern in the `remove()` function as well.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** The following functions are violating the "Checks-Effects-Interactions"-pattern and/or do not employ any reentrancy mitigating modifers, making them susceptible to reentrancy:

1. `EpochUpsidePoolV1.supply()`.
2. `EpochUpsidePoolV1.remove()`.
3. `EpochUpsidePoolV1.take()`.
4. `EpochUpsidePoolV1.untake()`.
5. `EpochUpsidePoolV1.claimProtocolFees()`.
6. `EpochUpsidePoolV1.claimPositionFees()`.

**Recommendation:** We recommend restructuring the instructions at above mentioned locations and/or consider employing the `nonReentrant`-modifier from OpenZeppelin to prevent any potential reentrancies.

## ELP-4 Insufficient User Documentation

Low | Mitigated

**Update**
The client provided the following explanation:

> The fee mechanisms will be clearly explained in the documentation. The fee will also be displayed in the frontend to the user. NatSpec code documentation has been added to the contracts.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** We have observed some incidents of undocumented behavior that the user should be informed about:

1. The fee mechanism is undocumented and users are currently not informed how the fee changes over time.
2. While it makes sense, the documentation does not reflect that users will not receive any fees back when calling `untake()`.
3. While some NatSpec inline code documentation is provided, they do not describe function parameters and return values. Consider adding these, at least for all publicly and externally callable functions.

**Recommendation:** We recommend reflecting on whether the points above are intended behavior and documenting the behavior for users.

## ELP-5 Ownership Can Be Renounced

Low | Acknowledged

**Update**
The client provided the following explanation:

> This is intended behaviour.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** If the owner renounces their ownership, all ownable contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed.

**Recommendation:** Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function in the affected contracts. For extra security, consider using a two-step process when transferring the ownership of the contract (e.g. `Ownable2Step` from OpenZeppelin).

## ELP-6 Critical Role Transfer Not Following Two-Step Pattern

Low | Acknowledged

**Update**

The client provided the following explanation:

> We do not believe this is necessary due to multisig implementation for the Owner.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** The owner of the contracts can call `transferOwnership()` to transfer the ownership to a new address. If an uncontrollable address is accidentally provided as the new owner address, then the contract will no longer have an active owner, and functions with the `onlyOwner` modifier can no longer be executed.

**Recommendation:** Consider using OpenZeppelin's `Ownable2Step` contract to adopt a two-step ownership pattern in which the new owner must accept their position before the transfer is complete.

## ELP-7 Unlocked Pragma

Low | Fixed

**Update**

Pragma was locked to 0.8.17.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Related Issue(s):** SWC-103

**Description:** Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. The caret ( `^` ) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

**Recommendation:** For consistency and to prevent unexpected behavior in the future, we recommend removing the caret to lock the file onto a specific Solidity version.

## ELP-8 Use of unsafe ERC20 transfer functions and not checking return values

Low | Fixed

**Update**

The client provided the following explanation:

> SafeERC20 has been implemented.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** The following lines are using the 'unsafe' ERC20 interface functions ( `transfer()` ):

1. `EpochUpsidePoolV1.sol#L87` .
2. `EpochUpsidePoolV1.sol#L91` .
3. `EpochUpsidePoolV1.sol#L136` .
4. `EpochUpsidePoolV1.sol#L156` .
5. `EpochUpsidePoolV1.sol#L180` .
6. `EpochUpsidePoolV1.sol#L191` .

Further, their return values were not checked, which, depending on token, could lead to failed transactions but continued code execution, incorrect accounting within the `EpochUpsidePoolV1.sol` contract and ultimately to loss of funds for the protocol/participants.

Since some ERC20 tokens do not return a value on above mentioned functions/behave non-uniformly it is advised to use a secure wrapping interface around said functions, like for example OpenZeppelins SafeERC20, which implicitly checks for the different error cases.

**Recommendation:** Replace the `IERC20` interface with the safe variant from OpenZeppelin, SafeERC20, to ensure safe behaviour, whether or not the token returns a value or not in said transfer functions.

## ELP-9 Hard-coded values limit flexibility

Informational | Acknowledged

**Update**

The client provided the following explanation:

> These are the upper limits as part of the design.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** Two values were hard-coded into the code and, therefore, limited the flexibility:
1. The maximum LP fee that can ever be set is hard-coded into `supply()`.
2. The maximum protocol fee is hard-coded in `setProtocolFee()`.

**Recommendation:** Consider whether these values could ever change under certain circumstances. If the answer is no and those values should be fixed prior to compilation, still consider to have constant variables holding those values for readability. This doesn't inflict more gas cost as the compiler will replace the variables with the constant values.

## ELP-10
## Application Monitoring Can Be Improved by Emitting More Events

Informational | Mitigated

**Update**

The client provided the following explanation:

> Implemented suggestion for Remove event. The client extended the `Remove` event. No event was added in case an `lpPositions` element was deleted.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** For easiness of application development, it is helpful to have the most important information emitted by events at crucial state changes. The following informational content of events could be improved upon:
1. No event is emitted when `lpPositions` is deleted in L94.
2. The `Remove` event only contains the `positionId` which may lead to more complex information querying, especially when the value at that index was deleted. Consider adding the following parameters: `_upsideTokenAmount` and `_downsideTokenAmount`.

**Recommendation:** We recommend extending the `Remove` event and additionally emitting an event when an `lpPositions` element is deleted.

## ELP-11 Inefficient gas usage

Informational | Fixed

**Update**

The client turned `lpPositions` into an array to save gas.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** We could identify an incident of inefficient gas usage:

`lpPositions` is a mapping, although it is treated like an array. It is more expensive to store new values in a mapping as opposed to an array.

**Recommendation:** We recommend changing `lpPositions` to an array to save gas.

## ELP-12 Missing input validation

Informational | Acknowledged

**Update**

The client provided the following explanation:

> 1.1 _exchangeRate input is not checked and can be 0. This would simply mean the taker receives 0 tokens correct
>
> 1.2, 1.3 - Some additional user validation has been implemented.
>
> 1.4 - Acknowledged
>
> 1.5 - _feeBp can be set to 0 as per design
>
> 2.1 - Acknowledged
>
> 2.2 - _protocolFeeMaxBp can be set to 0 as per design
>
> 3.0 - Acknowledged

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. The following functions do not have a proper validation of input parameters:

1. `EpochUpsidePoolV1.supply()`:
    1. `_exchangeRate` not checked to be non-zero. Clarify if this is intended.
    2. `_startDate` and `_endDate` not checked to be non-zero.
    3. `_startDate` and `_endDate` not checked to be chronological.
    4. `_startDate` and `_endDate` not checked to have a reasonable lower bound difference between each other.
    5. `_feeBp` not checked to be non-zero. Clarify if this is intended.
2. `EpochUpsidePoolV1.setProtocolFee()`:
    1. `_protocolFeeRecipientAddress` not checked to be different from `addresss(0x0)`.
    2. `_protocolFeeMaxBp` not checked to be non-zero. Clarify if this is intended.
3. `EpochUpsidePoolV1.claimPositionFees()`: `_tokenAddresses` not checked to be the same length as `_tokenAmounts`.

**Recommendation:** Consider adding input checks accordingly.

## ELP-13 Privileged roles and ownership

Informational | Acknowledged

**Update**

The client provided the following explanation:

> Acknowledged. The only function callable by the Owner is the setProtocolFee method. We acknowledge the fee could be set higher than the user was anticipating before the users transaction executes. We have therefore added a new parameter into the "take" method which allows the user to specify the minimum number of upside tokens expected.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** Certain contracts have state variables, e.g., `owner`, which provide specific addresses with privileged roles. Such roles may pose a risk to end-users.

For a detailed list of roles per contract, see the following list:

The `EpochUpsidePoolV1.sol` contract contains the following privileged roles:

1. `_owner` (`Ownable` inheritance), as initialized during the execution of `constructor` to `msg.sender`:
    1. Renounce the role (**and thereby prevent any future calls to the following listed functions!**) by calling `renounceOwnership()`.
    2. Transfer the ownership to an arbitrary other address by calling `transferOwnership()`.
    3. Change the protocol fee (up until `7001`, 7,001%) and fee receiver address by calling `setProtocolFee()`.

A malicious or compromised owner could, in a worst-case scenario, unexpectedly change the protocol fee from zero to up to 7.001%.

**Recommendation:** Clarify the impact of these privileged actions to the end-users via publicly facing documentation.

## ELP-14 Inconsistency in Token Amount Calculations for Large Input Amounts Relative to Liquidity

Low | Fixed

**Update**

The `protocolFee` is re-calculated to ensure that the total amount of fees always equals the initial `_maxDownsideTokenIn`.

**File(s) affected:** `EpochUpsidePoolV1.sol`

**Description:** In the revised `take()` function, there is an issue concerning the calculation of token amounts when the provided `_maxDownsideTokenIn` is larger than the available liquidity. This situation triggers a reverse calculation to adjust the amount of downside tokens after fees ( `_downsideTokenBalanceAfterFees` ). However, the current implementation may lead to a scenario where the sum of `_downsideTokenBalanceAfterFees` , `protocolFee` , and `lpFee` does not equal the initial `_maxDownsideTokenIn` . This discrepancy can result in users unintentionally committing a higher amount of downside tokens than intended, especially when liquidity is low, leading to unexpected financial implications.

**Recommendation:** We suggest modifying the code to guarantee that the total amount after fees ( `_downsideTokenBalanceAfterFees` + `protocolFee` + `lpFee` ) always equals the initial `_maxDownsideTokenIn` when entering the branch where `upsideTokenAmount` exceeds available liquidity.

# Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

# Adherence to Best Practices

1. To facilitate logging it is recommended to index address parameters within events. Therefore the `indexed` keyword should be added to the (other) address parameters in:
    1. `EpochUpsidePoolV1.Take()` .
    2. `EpochUpsidePoolV1.Untake()` .
    3. `EpochUpsidePoolV1.ProtocolFeeClaimed()` .
    4. `EpochUpsidePoolV1.LPFeeClaimed()` .
    5. `EpochUpsidePoolV1.ProtocolFeeSet()` .
2. To improve readability and lower the risk of introducing errors when making code changes, it is advised to not use magic constants throughout code, but instead declare them once (as constant and commented) and use these constant variables instead. Following instances should therefore be changed accordingly:
    1. `EpochUpsidePoolV1.sol#L59` : `7501` .
    2. `EpochUpsidePoolV1.sol#L105` , `L107` and `L108` : `10**18` .
    3. `EpochUpsidePoolV1.sol#L107-L108` : `10000` .
    4. `EpochUpsidePoolV1.sol#L164` : `7001` .

# Toolset

The notes below outline the setup and steps performed in the process of this audit.

**Setup**

Tool Setup:

- Slither v0.8.3

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

# Automated Analysis

**Slither**

Slither identified 124 issues in `EpochUpsidePoolV1.sol` during the analysis of the project. However, most of them were false positives and only valid ones have been included in the report.

# Test Suite Results

We followed the README, and ran the following commands:

```
pnpm install
pnpm compile
pnpm test
```

All 140 tests have successfully passed.

**Update**: After the fix review, 141 tests successfully passed.

```
root@4c2f26cc06eb:~/repo/moai_labs-upside-contracts-7626a37-github# pnpm test

  BasicUpside Tests
Upside Token: Epoch EPOCH 18
Downside Token: USD Coin USDC 6
      (1) New USDC token balance is 100000.0
    ✔ (1) acquire 100,000 downsideTokens (1905ms)
    ✔ transfer 50,000 downsideTokens from account 0 to account 1 (484ms)
      (2) New EPOCH token balance is 101000000.0
    ✔ (2) mint 1,000,000 upside tokens into account 0 (320ms)
      (3) New EPOCH token balance is 1000000.0
    ✔ (3) mint 1,000,000 upside tokens into account 1 (202ms)
      (4) DownsideToken: 0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
      (4) UpsideToken: 0xF5aA8e3C6BA1EdF766E197a0bCD5844Fd1ed8A27
      (4) UpsideTokenAmount: 100000000000000000000000
      (4) ExchangeRate: 4.0 upside tokens per downside token
      (4) StartDate: 1687557900 2023-06-23T22:05:00.000Z
      (4) EndDate: 1695333900 2023-09-21T22:05:00.000Z
      (4) FeeBp: 500
      (4) UpsideTokenBalanceBefore: 101000000.0
      (4) upsideTokenBalanceAfter: 100000000.0
    ✔ (4) account 0 should supply 1,000,000 upside liquidity to upside contract (661ms)
    ✔ ensure protocol fee cannot be set to 71% (80ms)
    ✔ ensure non owner cannot set protocol fee (89ms)
    ✔ set protocol fee to 5% (93ms)
      (5) upsideTokenBalance: 1000000.0
      (5) downsideTokenBalance: 0.0
    ✔ (5) print information from deposit id 0 (50ms)
    ✔ ensure account 1 cannot take position, revert with NOT STARTED (675ms)
    ✔ set next block.timestamp to start date of position (55ms)
      (6) UpsideTokenBalanceBefore: 1000000.0
      (6) UpsideTokenBalanceAfter: 1025500.00116
      (6) UpsideTokenBalanceDifference: 25500.00116
      (6) DownsideTokenBalanceBefore: 50000.0
      (6) DownsideTokenBalanceAfter: 42500.0
      (6) DownsideTokenBalanceDifference: 7500.0
      (6) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 25500.00116
      (6) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 7500
    ✔ (6) account 1 should partially take position with 7,500 downside tokens (deposit id 0) (920ms)
      (7) upsideTokenBalance: 974499.99884
      (7) downsideTokenBalance: 6375.00029
    ✔ (7) print information from deposit id 0 (40ms)
    ✔ ensure total deposited downside tokens for deposit id 0 (account 1) is GTE 6375 (48ms)
      (8) Purpose: Ensure the user cannot give back MORE upside tokens to get back more downside than
what they deposited (after fees)
    ✔ (8) account 1 should NOT be able to get back 7276 downside tokens, REVERT (260ms)
    ✔ account 0 should be unable to remove any downsideToken from position, revert with POSITION NOT
ENDED (82ms)
    ✔ account 1 should be unable to remove 1 downsideToken from position, revert with NOT OWNER (75ms)
    ✔ account 0 should be unable remove 1,000,000 upsideTokens from position, REVERT (72ms)
    ✔ account 0 should remove 970,900 upsideTokens from position (90ms)
      (9) upsideTokenBalance: 3599.99884
```

```
    (9) downsideTokenBalance: 6375.00029
  ✔ (9) print information from deposit id 0
    (10) UpsideTokenBalanceBefore: 1025500.00116
    (10) UpsideTokenBalanceAfter: 1000000.0
    (10) UpsideTokenBalanceDifference: 25500.00116
    (10) DownsideTokenBalanceBefore: 42500.0
    (10) DownsideTokenBalanceAfter: 48875.00029
    (10) DownsideTokenBalanceDifference: 6375.00029
    (10) ✔ expect downsideTokenBalanceAfter-downsideTokenBalanceBefore = 6375.00029
    (10) ✔ expect upsideTokenBalanceBefore-upsideTokenBalanceAfter = 25500.00116
  ✔ (10) account 1 should return 25,500.00116 upside tokens for 6375.00029 downside tokens (1065ms)
    (11) upsideTokenBalance: 29100.0
    (11) downsideTokenBalance: 0.0
  ✔ (11) print information from deposit id 0 (65ms)
  ✔ account 0 downside fees for USDC should be equal to 374.999903
  ✔ ensure account 0 cannot claim 375 downside fees (USDC), REVERT (142ms)
    (12) totalDownsideFees: 374.999903
    (12) DownsideTokenBalanceBefore: 50000.0
    (12) DownsideTokenBalanceAfter: 50374.999903
    (12) DownsideTokenBalanceDifference: 374.999903
  ✔ (12) ensure account 0 can claim 374.999903 downside fees (USDC) (230ms)
  ✔ account 0 downside fees for USDC should be equal to 0
    (13) upsideTokenBalance: 29100.0
    (13) downsideTokenBalance: 0.0
    ✔ (13) print information from deposit id 0
    ✔ set next block.timestamp to end date of position
    ✔ ensure account 1 cannot take position (or interact), revert with POSITION ENDED (197ms)
    ✔ ensure account 1 cannot untake position (or interact), revert with POSITION ENDED (345ms)
    ✔ account 0 should be unable to remove 1 downsideTokens, REVERT (192ms)
      (14) UpsideTokenBalanceBefore: 1000000.0
      (14) UpsideTokenBalanceAfter: 1000000.0
      (14) UpsideTokenBalanceDifference: 0.0
    ✔ (14) account 0 should remove 25,500 upsideTokens from position (212ms)
      (15) upsideTokenBalance: 3600.0
      (15) downsideTokenBalance: 0.0
    ✔ (15) print information from deposit id 0 (44ms)

  BasicUpsidePartial Tests
Upside Token: Epoch EPOCH 18
Downside Token: USD Coin USDC 6
      (1) New USDC token balance is 100000.0
    ✔ (1) acquire 100,000 downsideTokens (260ms)
    ✔ transfer 50,000 downsideTokens from account 0 to account 1 (95ms)
      (2) New EPOCH token balance is 101000000.0
    ✔ (2) mint 1,000,000 upside tokens into account 0 (130ms)
      (3) DownsideToken: 0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
      (3) UpsideToken: 0xF5aA8e3C6BA1EdF766E197a0bCD5844Fd1ed8A27
      (3) UpsideTokenAmount: 1000000000000000000000000
      (3) ExchangeRate: 4.0 upside tokens per downside token
      (3) StartDate: 1687557900 2023-06-23T22:05:00.000Z
      (3) EndDate: 1695333900 2023-09-21T22:05:00.000Z
      (3) FeeBp: 150
      (3) UpsideTokenBalanceBefore: 101000000.0
      (3) upsideTokenBalanceAfter: 100000000.0
  ✔ (3) account 0 should supply 1,000,000 upside liquidity to upside contract (228ms)
      (4) upsideTokenBalance: 1000000.0
      (4) downsideTokenBalance: 0.0
  ✔ (4) print information from deposit id 0
  ✔ set next block.timestamp to start date of position (40ms)
      (5) UpsideTokenBalanceBefore: 0.0
      (5) UpsideTokenBalanceAfter: 29250.000196
      (5) UpsideTokenBalanceDifference: 29250.000196
      (5) DownsideTokenBalanceBefore: 50000.0
      (5) DownsideTokenBalanceAfter: 42500.0
      (5) DownsideTokenBalanceDifference: 7500.0
      (5) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 29250.000196000000000000
      (5) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 7500
  ✔ (5) account 1 should partially take position with 7,500 downside tokens (deposit id 0) (396ms)
      (6) upsideTokenBalance: 970749.999804
      (6) downsideTokenBalance: 7312.500049
  ✔ (6) print information from deposit id 0
  ✔ ensure total deposited downside tokens for deposit id 0 (account 1) is 7312.500049
```

```
    ✔ account 0 should remove 970,749.999708 upsideTokens from position (93ms)
        (7) UpsideTokenBalanceBefore: 29250.000196
        (7) UpsideTokenBalanceAfter: 0.0
        (7) UpsideTokenBalanceDifference: 29250.000196
        (7) DownsideTokenBalanceBefore: 42500.0
        (7) DownsideTokenBalanceAfter: 49812.500049
        (7) DownsideTokenBalanceDifference: 7312.500049
        (7) ✔ expect downsideTokenBalanceAfter-downsideTokenBalanceBefore = 7312.500049
        (7) ✔ expect upsideTokenBalanceBefore-upsideTokenBalanceAfter = 29250.000196000000000000
    ✔ (7) account 1 should return 29,250.000196000000000000 upside tokens for 7312.500049 downside tokens
(544ms)
        (8) upsideTokenBalance: 29250.000292
        (8) downsideTokenBalance: 0.0
    ✔ (8) print information from deposit id 0 (201ms)
    ✔ account 0 should be unable to remove 14,550 upsideTokens and 3637.5 downsideTokens from position,
revert with POSITION NOT ENDED (163ms)
    ✔ set next block.timestamp to end date of position (41ms)
        (9) UpsideTokenBalanceBefore: 100970749.999708
        (9) UpsideTokenBalanceAfter: 100985299.999708
        (9) UpsideTokenBalanceDifference: 14550.0
        (9) DownsideTokenBalanceBefore: 50000.0
        (9) DownsideTokenBalanceAfter: 50000.0
        (9) DownsideTokenBalanceDifference: 0.0
        (9) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 14550
        (9) ✔ expect downsideTokenBalanceAfter-downsideTokenBalanceBefore = 0
    ✔ (9) account 0 should remove 14,550 upsideTokens and 0 downsideTokens from position (287ms)
        (10) upsideTokenBalance: 14700.000292
        (10) downsideTokenBalance: 0.0
    ✔ (10) print information from deposit id 0
        (11) totalDownsideFees: 112.499971
        (11) DownsideTokenBalanceBefore: 50000.0
        (11) DownsideTokenBalanceAfter: 50100.0
        (11) DownsideTokenBalanceDifference: 100.0
    ✔ (11) ensure account 0 can claim 100 downside fees (USDC) (180ms)
        (12) totalDownsideFees: 12.499971
        (12) DownsideTokenBalanceBefore: 50100.0
        (12) DownsideTokenBalanceAfter: 50112.499956
        (12) DownsideTokenBalanceDifference: 12.499956
    ✔ (12) ensure account 0 can claim 12.499956 downside fees (USDC) (188ms)

  EdgeCases: LowerExchangeRate Tests
Upside Token: Epoch EPOCH 18
Downside Token: USD Coin USDC 6
        (1) New USDC token balance is 100000.0
    ✔ (1) acquire 100,000 downsideTokens (167ms)
    ✔ transfer 50,000 downsideTokens from account 0 to account 1 (70ms)
        (2) New EPOCH token balance is 101000000.0
    ✔ (2) mint 1,000,000 upside tokens into account 0 (119ms)
        (3) New EPOCH token balance is 1000000.0
    ✔ (3) mint 1,000,000 upside tokens into account 1 (109ms)
        (4) DownsideToken: 0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
        (4) UpsideToken: 0xF5aA8e3C6BA1EdF766E197a0bCD5844Fd1ed8A27
        (4) UpsideTokenAmount: 100000000000000000000000
        (4) ExchangeRate: 0.25 upside tokens per downside token
        (4) StartDate: 1687557900 2023-06-23T22:05:00.000Z
        (4) EndDate: 1695333900 2023-09-21T22:05:00.000Z
        (4) FeeBp: 0
        (4) UpsideTokenBalanceBefore: 101000000.0
        (4) upsideTokenBalanceAfter: 100000000.0
    ✔ (4) account 0 should supply 1,000,000 upside liquidity to upside contract (205ms)
    ✔ ensure protocol fee cannot be set to 71% (85ms)
    ✔ ensure non owner cannot set protocol fee (63ms)
    ✔ set protocol fee to 0% (61ms)
        (5) upsideTokenBalance: 1000000.0
        (5) downsideTokenBalance: 0.0
    ✔ (5) print information from deposit id 0
    ✔ ensure account 1 cannot take position, revert with NOT STARTED (117ms)
    ✔ set next block.timestamp to start date of position
        (6) UpsideTokenBalanceBefore: 1000000.0
        (6) UpsideTokenBalanceAfter: 1001875.0
        (6) UpsideTokenBalanceDifference: 1875.0
        (6) DownsideTokenBalanceBefore: 50000.0
```

```
        (6) DownsideTokenBalanceAfter: 42500.0
        (6) DownsideTokenBalanceDifference: 7500.0
        (6) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 1875
        (6) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 7500
    ✔ (6) account 1 should partially take position with 7,500 downside tokens (deposit id 0) (354ms)
        (7) upsideTokenBalance: 998125.0
        (7) downsideTokenBalance: 7500.0
    ✔ (7) print information from deposit id 0
    ✔ ensure total deposited downside tokens for deposit id 0 (account 1) is EQ 7500
        (8) Purpose: Ensure the user cannot give back MORE upside tokens to get back more downside than
what they deposited (after fees)
    ✔ (8) account 1 should NOT be able to get back 7501 downside tokens, REVERT (214ms)
    ✔ account 0 should be unable to remove any downsideToken from position, revert with POSITION NOT
ENDED (68ms)
    ✔ account 1 should be unable to remove 1 downsideToken from position, revert with NOT OWNER (69ms)
    ✔ account 0 should be unable remove 1,000,000 upsideTokens from position, REVERT (58ms)
    ✔ account 0 should remove 970,900 upsideTokens from position (75ms)
        (9) upsideTokenBalance: 27225.0
        (9) downsideTokenBalance: 7500.0
    ✔ (9) print information from deposit id 0
        (10) UpsideTokenBalanceBefore: 1001875.0
        (10) UpsideTokenBalanceAfter: 1000000.0
        (10) UpsideTokenBalanceDifference: 1875.0
        (10) DownsideTokenBalanceBefore: 42500.0
        (10) DownsideTokenBalanceAfter: 50000.0
        (10) DownsideTokenBalanceDifference: 7500.0
        (10) ✔ expect downsideTokenBalanceAfter-downsideTokenBalanceBefore = 7500
        (10) ✔ expect upsideTokenBalanceBefore-upsideTokenBalanceAfter = 1875
  ✔ (10) account 1 should return 1,875 upside tokens for 7500 downside tokens (277ms)
        (11) upsideTokenBalance: 29100.0
        (11) downsideTokenBalance: 0.0
  ✔ (11) print information from deposit id 0 (40ms)
  ✔ account 0 downside fees for USDC should be equal to 0
  ✔ ensure account 0 cannot claim 1 downside fees (USDC), REVERT (76ms)
        (12) upsideTokenBalance: 29100.0
        (12) downsideTokenBalance: 0.0
  ✔ (12) print information from deposit id 0
  ✔ set next block.timestamp to end date of position
  ✔ account 0 should be unable to remove 1 downsideTokens, REVERT (60ms)
        (13) UpsideTokenBalanceBefore: 1000000.0
        (13) UpsideTokenBalanceAfter: 1000000.0
        (13) UpsideTokenBalanceDifference: 0.0
  ✔ (13) account 0 should remove 25,500 upsideTokens from position (143ms)
        (14) upsideTokenBalance: 3600.0
        (14) downsideTokenBalance: 0.0
  ✔ (14) print information from deposit id 0

EdgeCases: TakeFullPosition Tests
Upside Token: Epoch EPOCH 18
Downside Token: USD Coin USDC 6
        (1) New USDC token balance is 100000.0
    ✔ (1) acquire 100,000 downsideTokens (169ms)
    ✔ transfer 50,000 downsideTokens from account 0 to account 1 (88ms)
        (2) New EPOCH token balance is 101000000.0
    ✔ (2) mint 1,000,000 upside tokens into account 0 (142ms)
        (3) New EPOCH token balance is 1000000.0
    ✔ (3) mint 1,000,000 upside tokens into account 1 (113ms)
        (4) DownsideToken: 0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
        (4) UpsideToken: 0xF5aA8e3C6BA1EdF766E197a0bCD5844Fd1ed8A27
        (4) UpsideTokenAmount: 100000000000000000000000
        (4) ExchangeRate: 4.0 upside tokens per downside token
        (4) StartDate: 1687557900 2023-06-23T22:05:00.000Z
        (4) EndDate: 1695333900 2023-09-21T22:05:00.000Z
        (4) FeeBp: 0
        (4) UpsideTokenBalanceBefore: 101000000.0
        (4) upsideTokenBalanceAfter: 100900000.0
    ✔ (4) account 0 should supply 100,000 upside liquidity to upside contract (269ms)
    ✔ set protocol fee to 0% (81ms)
        (5) upsideTokenBalance: 100000.0
        (5) downsideTokenBalance: 0.0
    ✔ (5) print information from deposit id 0 (38ms)
    ✔ set next block.timestamp to start date of position
```

```
    ✔ account 1 should be unable to take position 0 with 45,000 downsideTokens, REVERT (246ms)
        (6) UpsideTokenBalanceBefore: 1000000.0
        (6) UpsideTokenBalanceAfter: 1100000.0
        (6) UpsideTokenBalanceDifference: 100000.0
        (6) DownsideTokenBalanceBefore: 50000.0
        (6) DownsideTokenBalanceAfter: 25000.0
        (6) DownsideTokenBalanceDifference: 25000.0
        (6) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 100,000
        (6) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 25000
    ✔ (6) account 1 should FULLY take position with 25,000 downside tokens (deposit id 0) (325ms)
        (7) upsideTokenBalance: 0.0
        (7) downsideTokenBalance: 25000.0
    ✔ (7) print information from deposit id 0
    ✔ ensure total deposited downside tokens for deposit id 0 (account 1) is EQ 25,000 (38ms)
    ✔ account 0 should UNABLE to remove 1 upsideTokens from position, REVERT (136ms)

  ProtocolFee: DownsideUSDC6 Tests
Upside Token: Epoch EPOCH 18
Downside Token: USD Coin USDC 6
        (1) New USDC token balance is 100000.0
    ✔ (1) acquire 100,000 downsideTokens (172ms)
    ✔ transfer 50,000 downsideTokens from account 0 to account 1 (89ms)
        (2) New EPOCH token balance is 101000000.0
    ✔ (2) mint 1,000,000 upside tokens into account 0 (137ms)
    ✔ set protocol fee to 10% (52ms)
        (3) StartDate: 1687507901 2023-06-23T08:11:41.000Z
    ✔ (3) print current block timestamp (44ms)
        (4) DownsideToken: 0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
        (4) UpsideToken: 0xF5aA8e3C6BA1EdF766E197a0bCD5844Fd1ed8A27
        (4) UpsideTokenAmount: 100000000000000000000000
        (4) ExchangeRate: 4.0 upside tokens per downside token
        (4) StartDate: 1687507901 2023-06-23T08:11:41.000Z
        (4) EndDate: 1695283901 2023-09-21T08:11:41.000Z
        (4) FeeBp: 1000
        (4) UpsideTokenBalanceBefore: 101000000.0
        (4) upsideTokenBalanceAfter: 100000000.0
    ✔ (4) account 0 should supply 1,000,000 upside liquidity to upside contract (234ms)
        (5) BlockTimestamp: 1687507903 2023-06-23T08:11:43.000Z
        (5) Input Downside Tokens: 7500.0
        (5) Protocol Fee: 749.999807
        (5) LP Fee: 749.999807
    ✔ (5) display current fee to take position 0 partially with 7500 downside tokens (65ms)
    ✔ set next block.timestamp to 60% into start date of position (58ms)
        (6) BlockTimestamp: 1692173501 2023-08-16T08:11:41.000Z
        (6) Input Downside Tokens: 7500.0
        (6) Protocol Fee: 300.0
        (6) LP Fee: 300.0
    ✔ (6) display current fee to take position 0 partially with 7500 downside tokens (52ms)
        (7) UpsideTokenBalanceBefore: 0.0
        (7) UpsideTokenBalanceAfter: 27600.001544
        (7) UpsideTokenBalanceDifference: 27600.001544
        (7) DownsideTokenBalanceBefore: 50000.0
        (7) DownsideTokenBalanceAfter: 42500.0
        (7) DownsideTokenBalanceDifference: 7500.0
        (7) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore TO BE GREATER THAN OR EQUAL TO
27600
        (7) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 7500
    ✔ (7) account 1 should partially take position with 7,500 downside tokens (deposit id 0) (331ms)

  ProtocolFee: DownsideWETH18 Tests
Upside Token: Epoch EPOCH 18
Downside Token: Wrapped Ether WETH 18
        (1) New WETH token balance is 100000.0
    ✔ (1) acquire 100,000 downsideTokens (1244ms)
    ✔ transfer 50,000 downsideTokens from account 0 to account 1 (278ms)
        (2) New EPOCH token balance is 101000000.0
    ✔ (2) mint 1,000,000 upside tokens into account 0 (147ms)
    ✔ set protocol fee to 10% (52ms)
        (3) StartDate: 1687507901 2023-06-23T08:11:41.000Z
    ✔ (3) print current block timestamp
        (4) DownsideToken: 0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2
        (4) UpsideToken: 0xF5aA8e3C6BA1EdF766E197a0bCD5844Fd1ed8A27
```

```
    (4) UpsideTokenAmount: 100000000000000000000000
    (4) ExchangeRate: 4.0 upside tokens per downside token
    (4) StartDate: 1687507901 2023-06-23T08:11:41.000Z
    (4) EndDate: 1695283901 2023-09-21T08:11:41.000Z
    (4) FeeBp: 1000
    (4) UpsideTokenBalanceBefore: 101000000.0
    (4) upsideTokenBalanceAfter: 100000000.0
✔ (4) account 0 should supply 1,000,000 upside liquidity to upside contract (399ms)
    (5) BlockTimestamp: 1687507903 2023-06-23T08:11:43.000Z
    (5) Input Downside Tokens: 7500.0
    (5) Protocol Fee: 749.999807098765425
    (5) LP Fee: 749.999807098765425
✔ (5) display current fee to take position 0 partially with 7500 downside tokens (68ms)
✔ set next block.timestamp to 60% into start date of position (49ms)
    (6) BlockTimestamp: 1692173501 2023-08-16T08:11:41.000Z
    (6) Input Downside Tokens: 7500.0
    (6) Protocol Fee: 300.0
    (6) LP Fee: 300.0
✔ (6) display current fee to take position 0 partially with 7500 downside tokens (48ms)
    (7) UpsideTokenBalanceBefore: 0.0
    (7) UpsideTokenBalanceAfter: 27600.0015432098766
    (7) UpsideTokenBalanceDifference: 27600.0015432098766
    (7) DownsideTokenBalanceBefore: 50000.0
    (7) DownsideTokenBalanceAfter: 42500.0
    (7) DownsideTokenBalanceDifference: 7500.0
    (7) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore TO BE GREATER THAN OR EQUAL TO
27600
    (7) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 7500
  ✔ (7) account 1 should partially take position with 7,500 downside tokens (deposit id 0) (937ms)

  Scenarios: Scenario1 Tests
Upside Token: Epoch EPOCH 18
Downside Token: USD Coin USDC 6
    (1) New USDC token balance is 500000.0
  ✔ (1) account 0 to acquire 500,000 downsideTokens (231ms)
  ✔ transfer 50,000 downsideTokens from account 0 to account 1 (76ms)
  ✔ transfer 50,000 downsideTokens from account 0 to account 2 (391ms)
  ✔ transfer 50,000 downsideTokens from account 0 to account 3 (524ms)
    (2) New EPOCH token balance is 102500000.0
  ✔ (2) mint 2,500,000 upside tokens into account 0 (126ms)
  ✔ set protocol fee to 2% (69ms)
    (3) StartDate: 1687507903 2023-06-23T08:11:43.000Z
  ✔ (3) print current block timestamp
    (4) DownsideToken: 0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
    (4) UpsideToken: 0xF5aA8e3C6BA1EdF766E197a0bCD5844Fd1ed8A27
    (4) UpsideTokenAmount: 250000000000000000000000
    (4) ExchangeRate: 20.0 upside tokens per downside token
    (4) StartDate: 1687507903 2023-06-23T08:11:43.000Z
    (4) EndDate: 1695283903 2023-09-21T08:11:43.000Z
    (4) FeeBp: 100
    (4) UpsideTokenBalanceBefore: 102500000.0
    (4) upsideTokenBalanceAfter: 100000000.0
✔ (4) account 0 should supply 2,500,000 upside liquidity to upside contract (273ms)
    (5) BlockTimestamp: 1687507905 2023-06-23T08:11:45.000Z
    (5) Input Downside Tokens: 7500.0
    (5) Protocol Fee: 149.999961
    (5) LP Fee: 74.99998
✔ (5) display current fee to take position 0 partially with 7500 downside tokens (71ms)
    (6) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 145,500.002340000000000000
    (6) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 7500
✔ (6) account 1 should partially take position with 7,500 downside tokens (deposit id 0) (385ms)
✔ set next block.timestamp to 40% into start date of position (83ms)
    (7) BlockTimestamp: 1690618303 2023-07-29T08:11:43.000Z
    (7) Input Downside Tokens: 7500.0
    (7) Protocol Fee: 90.0
    (7) LP Fee: 45.0
✔ (7) display current fee to take position 0 partially with 7500 downside tokens (52ms)
    (8) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 147300.001180000000000000
    (8) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 7500
✔ (8) account 2 should partially take position with 7,500 downside tokens (deposit id 1) (525ms)
    (9) BlockTimestamp: 1690618305 2023-07-29T08:11:45.000Z
    (9) Input Downside Tokens: 15000.0
```

```
      (9) Protocol Fee: 179.999922
      (9) LP Fee: 89.999961
    ✔ (9) display current fee to take position 0 partially with 15,000 downside tokens (70ms)
      (10) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 294600.004660000000000000
      (10) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 15000
    ✔ (10) account 3 should partially take position with 15,000 downside tokens (deposit id 2) (533ms)
    ✔ set next block.timestamp to 80% into start date of position (61ms)
      (11) BlockTimestamp: 1693728703 2023-09-03T08:11:43.000Z
      (11) Input Downside Tokens: 7500.0
      (11) Protocol Fee: 30.0
      (11) LP Fee: 15.0
    ✔ (11) display current fee to take position 0 partially with 7500 downside tokens (55ms)
      (12) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 149,100.001180000000000000
      (12) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 7500
    ✔ (12) account 1 should partially take position with 7,500 downside tokens (deposit id 3) (338ms)
    ✔ account 2 should NOT be able to return 7,366.000059 downside tokens, REVERT (193ms)
      (13) ✔ expect downsideTokenBalanceAfter-downsideTokenBalanceBefore = 7365.000059
      (13) ✔ expect upsideTokenBalanceBefore-upsideTokenBalanceAfter = 147300.001180000000000000
    ✔ (13) account 2 should return 147,300.001180000000000000 upside tokens for 7,365.000059 downside
tokens (361ms)
    ✔ set next block.timestamp to 101% into start date of position (65ms)
    ✔ account 3 should NOT be able to return 1 downside tokens, REVERT with POSITION ENDED (181ms)
      (14) upsideTokenBalance: 1910799.99182
      (14) downsideTokenBalance: 29460.000409
    ✔ (14) print information from deposit id 0 (43ms)
      (12) ✔ expect upsideTokenBalanceAfter-upsideTokenBalanceBefore = 1,910,799.99182
      (12) ✔ expect downsideTokenBalanceBefore-downsideTokenBalanceAfter = 29460.000409
    ✔ account 0 (LP) to withdraw 1910799.99182 upside and 29460.000409 downsideTokens (367ms)
      (15) Account0 USDC Fees (downsideTokens): 224.999843
      (15) Protocol USDC Fees (downsideTokens): 449.999689
    ✔ (15) print fees earned by account 0 (USDC), print fees earned by protocol (USDC) (63ms)
      (16) ✔ expect protocolFeeRecipientAddress USDC balance to increase by 449.999689
    ✔ (16) account 0 should press button to transfer protocol fees (186ms)
      (17) Account0 USDC Fees (downsideTokens): 224.999843
      (17) Protocol USDC Fees (downsideTokens): 0.0
    ✔ (17) print fees earned by account 0 (USDC), print fees earned by protocol (USDC) (51ms)


  140 passing (1m)
```

# Code Coverage

We followed the README, and ran the following command:

```
pnpm coverage
```

The test suite achieves 100% branch coverage. We would like to note that branch coverage is not a metric for test quality and recommend mutation testing in addition.

**Update:** After the fix review, the branch coverage was reduced to 85.71 %. We recommend increasing the branch coverage to 100%.

| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines |
|---|---|---|---|---|---|
| **contracts/** | 100 | 100 | 100 | 100 | |
| EpochUpsidePoolV1.sol | 100 | 100 | 100 | 100 | |
| All files | 100 | 100 | 100 | 100 | |

**Update**: ----------|----------|----------|----------|----------|----------------|

| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines |
|---|---|---|---|---|---|
| **contracts/** | 97.73 | 85.71 | 88.89 | 97.06 | |

| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines |
|---|---|---|---|---|---|
| ITOProtocolV1.sol | 97.73 | 85.71 | 88.89 | 97.06 | 74,245 |
| All files | 97.73 | 85.71 | 88.89 | 97.06 | |

# Changelog

- 2023-11-07 - Initial report
- 2023-12-01 - Fix Review
- 2023-12-15 - Fix Review 2
- 2023-12-20 - Fix Review 3

# About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

# Quantstamp