VIDMA

milestoneBased

SMART CONTRACT AUDIT

Project: milestoneBased
Date: February 27th, 2023

TABLE OF CONTENTS

SUMMARY

Vidma is pleased to present this audit report outlining our assessment of code, smart contracts, and other important audit insights and suggestions for management, developers, and users.

The audited scope included a contract that allows swapping $MILE tokens to sMILE tokens by the market price and specific entities contracts. Anyone is able to get sMILE by depositing $MILE and do reverse action. sMILE token is minting or burning based on the swap action requested. The initial conversion rate is 1-1 but may change if the $MILE rewards are added to the MILE/sMILE pool. The swapping process allows conversation some amount of ERC20 tokens to $MILE using PancakeSwap router.

There are defined entities for interactions on the platform, that were also included in the scope of the audit. User, company, and project entities for representation of the user, company, or project capital and actions. Only entity contracts may get sMILE tokens from the contract.

During the audit process, the Vidma team found several issues. A detailed summary and the current state are displayed in the table below.

Severity of the issue	Total found	Resolved	Not relevant	Unresolved
Critical	0 issues	0 issues	0 issues	0 issues
High	0 issues	0 issues	0 issues	0 issues
Medium	1 issue	1 issue	0 issue	0 issues
Low	7 issues	6 issues	1 issue	0 issues
Informational	15 issues	14 issues	1 issue	0 issues
Total	**23 issues**	**21 issues**	**2 issues**	**0 issues**

After evaluating the findings in this report and the final state after fixes, the Vidma auditors can state that the contracts are fully operational and secure. Under the given circumstances, we set the following risk level:

To set the codebase quality mark, our auditors are evaluating the initial commit given for the scope of the audit and the last commit with the fixes. This approach helps us adequately and sequentially evaluate the quality of the code. Code style, optimization of the contracts, the number of issues, and risk level of the issues are all taken into consideration. The Vidma team has developed a transparent evaluation codebase quality system presented below.

Severity of the issue	Resolved	Unresolved
Critical	1	10
High	0.8	7
Medium	0.5	5
Low	0.2	0.5
Informational	0	0.1

Please note that the points are deducted out of 100 for each and every issue on the list of findings (according to the current status of the issue). Issues marked as "not valid" are not subject to point deduction.

Codebase quality: 98.30

Evaluating the **initial commit** and the **last commit with the fixes**, Vidma audit team set the following **codebase quality** mark.

Score

Based on the **overall result of the audit** and the state of the final reviewed commit, the Vidma audit team grants the following **score**:

In addition to manual check and static analysis, the auditing team has conducted a number of integrated autotests to ensure the given codebase has an adequate performance and security level.

The test results and coverage can be found in the accompanying section of this audit report.

Please be aware that this audit does not certify the definitive reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by the Vidma audit team. If the code is still under development, we highly recommend running one more audit once the code is finalized.

SCOPE OF WORK

milestoneBased

milestoneBased is the first company to leverage a blockchain DAO and escrow smart contract capabilities, in an automated governance and milestone achievement management platform.

Within the scope of this audit, two independent auditors thoroughly investigated the given codebase and analyzed the overall security and performance of the smart contracts.

The audit was conducted from September 30, 2022 to October 25, 2022. The outcome is disclosed in this document.

The review of the fixes was conducted on October 31, 2022.

The second review of the fixes was conducted on November 4, 2022.

The additional review of the changes in the contracts was completed on February 6, 2023. No issues were detected after the latest changes made in the codebase.

The fourth review of the changes was conducted on February 27, 2023.

The scope of work for the given audit consists of the following contracts:

- Create2Factroy;
- ContractsRegistry;
- BaseEntity;
- ExpandedEntity;
- OwnerManager;
- UserEntity;
- CompanyEntity;
- ProjectEntity;
- Constants;
- ContractsRegistry;
- Create2Factory;
- EntityFactory;

- SingleSignEntityStrategy;
- WhiteList;
- SMilestoneBasedToken;
- Strategy;
- SwapMILE;

The source code was taken from the following **source**:
https://bitbucket.org/applicature/milestonebased.contracts/branch/release/v2.0

Initial commit submitted for the audit:
a3b9757d3bea50ee28740d315558dcb6878c0b2e

Last commit reviewed by the auditing team:
6b3cdb9f321208f39443a931319d76288e251640

WORKFLOW OF THE AUDITING PROCESS

Vidma audit team uses the most sophisticated and contemporary methods and well-developed techniques to ensure contracts are free of vulnerabilities and security risks. The overall workflow consists of the following phases:

Phase 1: The research phase

Research

After the Audit kick-off, our security team conducts research on the contract's logic and expected behavior of the audited contracts.

Documentation reading

Vidma auditors do a deep dive into your tech documentation with the aim of discovering all the behavior patterns of your codebase and analyzing the potential audit and testing scenarios.

The outcome

At this point, the Vidma auditors are ready to kick off the process. We set the auditing strategies and methods and are prepared to conduct the first audit part.

Phase 2: Manual part of the audit

Manual check

During the manual phase of the audit, the Vidma team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract. The initial commit as stated in the agreement is taken into consideration.

Static analysis check

Static analysis tools are used to find any other vulnerabilities in smart contracts that were missed after a manual check.

The outcome

An interim report with the list of issues.

Phase 3: Testing part of the audit

Integration tests

Within the testing part, Vidma auditors run integration tests using the Truffle or Hardhat testing framework. The test coverage and the test results are inserted in the accompanying section of this audit report.

The outcome

Second interim report with the list of new issues found during the testing part of the audit process.

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For simplicity in reviewing the findings in this report, Vidma auditors classify the findings in accordance with the severity level of the issues. (from most critical to least critical).

All issues are marked as “Resolved” or “Unresolved”, depending on if they have been fixed by project team or not. The issues with “Not Relevant” status are left on the list of findings but are not eligible for the score points deduction.

The latest commit with the fixes reviewed by the auditors is indicated in the “Scope of Work” section of the report.

The Vidma team always provides a detailed description of the issues and recommendations on how to fix them.

Classification of found issues is graded according to 6 levels of severity described below:

Critical

The issue affects the contract in such a way that funds may be lost or allocated incorrectly, or the issue could result in a significant loss.

Example: Underflow/overflow, precisions, locked funds.

High

The issue significantly affects the ability of the contract to compile or operate. These are potential security or operational issues.

Example: Compilation errors, pausing/unpausing of some functionality, a random value, recursion, the logic that can use all gas from block (too many iterations in the loop), no limitations for locking period, cooldown, arithmetic errors which can cause underflow, etc.

Medium

The issue slightly impacts the contract's ability to operate by slightly hindering its intended behavior.

Example: Absence of emergency withdrawal of funds, using assert for parameter sanitization.

Low

The issue doesn't contain operational or security risks, but are more related to optimization of the codebase.

Example: Unused variables, inappropriate function visibility (public instead of external), useless importing of SCs, misuse or disuse of constant and immutable, absent indexing of parameters in events, absent events to track important state changes, absence of getters for important variables, usage of string as a key instead of a hash, etc.

Informational

Are classified as every point that increases onboarding time and code reading, as well as the issues which have no impact on the contract's ability to operate.

Example: Code style, NatSpec, typos, license, refactoring, naming convention (or unclear naming), layout order, functions order, lack of any type of documentation.

MANUAL REPORT

Potential reentrancy issue

Medium | MM – 01 | Resolved

In the function *withdrawFromEntity()* of BaseEntity there is the possibility of reentrancy when coins are transferred.

Recommendation:

Consider handling reentrancy in the function *withdrawFromEntity()*.

Inappropriate function visibility

Low | ML – 01 | Resolved

In the audited codebase there are some functions that not called in the contract itself so their visibility can be improved from public to external:

- function *mint()* of SMilestoneBasedToken contract;
- function *addNewAddress()* of WhiteList contract;
- function *removeAddress()* of WhiteList contract;
- Function *computeAddress()* of *Create2factory()*.

Recommendation:

Consider changing visibility from public to external to safe gas usage on calling these functions.

Event indexing is missed

Low | ML – 02 | Resolved

Params event indexing is missed in the appropriate interfaces of next contracts OwnerManager, BaseEntity, SingleSignEntityStrategy, Strategy, WhiteList, EntityFactory (*AddedOwnerOfEntity()* and *RemovedOwnerOfEntity()*), SMilestoneBasedToken (event *UpdatedStrategy*). The indexed parameters for logged events give the possibility to search for these events using the indexed parameters as filters (solidity official docs)

Recommendation:

Consider adding event param indexing where it is missed.

Useless type casting

Low | ML – 03 | Resolved

In the *constructor()* of SMilestoneBasedToken param *strategy_* with type address is casting to address in the L26 what is useless.

Recommendation:

Consider avoiding useless casting to save gas.

Lack of zero value check

Low | ML – 04 | Resolved

In the function *withdrawFromEntity()* and *approve()* of BaseEntity there is no check if the amount and recipient aren`t zero value.

Recommendation:

Consider adding described checks.

Useless event emissions

Low | ML – 05 | Unresolved

In the WhiteList there is no check if the added or removed address is valid. In the first case, if it isn't in the whitelist already, in the second case if it exists in the whitelist. In these cases such addresses won't be added or removed because they are invalid, but events *AddedNewAddress* and *RemovedAddress* still will be emitted.

In the WhiteList Smart Contract *_whiteList.add()* and *_whiteList.remove()* internal calls can be wrapped in the if checking condition to revert error when the address has already been added to the whitelist or it does not exist there to avoid useless gas payments. For example:

```
if(_whiteList.add(newAddress))
revert AddressAlreadyAdded(newAddress);
if(_whiteList.remove(addressToRemove))
revert AddressDoesNotExist(addressToRemove);
```

Recommendation:

Consider adding return value check of *.add()* and *.remove()* functions from EnumerableSet to handle the correctness of event emission.

Useless revert of custom error

Low | TL – 01 | Resolved

In the *removeOwnerFromUserEntity()* function of EntityFactory Smart Contract there are *ElseEntityOwner()* error revert that will be never reverted because it is called by the UserEntity Smart Contract in the *_swapOwner()* function and will be failed earlier in the *_swapOwner()* of OwnerManager with *InvalidPrevOwnerAddress()* error.

Recommendation:

Consider removing error reverting in the *removeOwnerFromUserEntity()* or moving *super._swapOwner()* call to the end of *_swapOwner()* function of UserEntity Smart contract to cover that case.

The function selector is not checked

Low | TL – 02 | Not relevant

In the Create2Factory Smart Contract there is *setContractOwner()* function that transfers ownership from contract instance by selected contract address for msg.sender. To avoid error 'function selector was not recognized and there's no fallback function' in case when contract instance does not support *transferOwnership()* function there could be realized low-level call, for example:

```
function setContractOwner(address contractAddress) external
onlyOwner {
    bool success;
    bytes memory data = abi.encodeWithSelector(
      bytes4(keccak256("transferOwnership(address)")),
      _msgSender()
    );

    assembly {
      success := call(
        gas(),
        contractAddress,
        0,
        add(data, 32),
        mload(data),
        0,
        0
      )
    }

    if (!success) revert TransferOwnershipFailed();
  }
```

Recommendation:

Consider covering cases when *transferOwnership()* is not supported by contract instances.

Unused function parameter

Informational | MI – 01 | Resolved

In function *validateTransaction* of Strategy.sol contract function parameter amount is unused.

Recommendation:

Consider removing or commenting out the unused param.

Incorrect NatSpec

Informational | MI – 02 | Resolved

In function *swapOwner()* (and internal *_swapOwner()*) of OwnerManager and appropriate interface *IOwnerManager*, event *AddedOwner* and *RemovedOwner* is emitted. NatSpec says only about *AddedOwner* and duplicates it twice.

Recommendation:

Consider proving correct NatSpec comments.

Optimization in the ExpandedEntity initializer

Informational | MI – 03 | Not relevant

In the *__ExpandedEntity_init() BaseEntity()* initializer is called first and then checking if the owner is the UserEntity. It is possible to switch the order and first handle the checking and then initialization. It will save gas for the user in case the owner is not a user entity.

Recommendation:

Consider adding check the requirements before the initialization.

Misleading NatSpec in UserEntity

Informational | MI – 04 | Resolved

In the NatSpec comment for *initialize()* of UserEntity contract in the requirements sections there is require about the owner must be UserEntity. Based on the contract architecture and implemented logic the owner of UserEntity can be a regular user.

Recommendation:

Consider removing useless requirement.

Misleading NatSpec in SingleSignEntityStrategy

Informational | MI – 05 | Resolved

There is incorrect explanation of the return result (false or true) for the function *_isValid()*, *isValid()* of SingleSignEntityStrategy and *usedNonces()* of ISingleSignEntityStrategy.

Recommendation:

Consider correct the NatSpec description.

Duplicated functionality

Informational | MI – 06 | Resolved

There is functionality for adding and removing addresses from the whitelist in the WhiteList contract. It covered in functions *addNewAddress()*, *removeAddress()* and *addNewAddressesBatch()*, *revomeAddressesBatch()*. *addNewAddressesBatch()* duplicate implementation of *addNewAddress()* function same as *revomeAddressesBatch()* duplicate *removeAddress()*. It will be better to separate functionality for adding and removing addresses in the internal functions and call an appropriate function in appropriate functions.

Recommendation:

Consider adding an internal function to make code more readable and follow DRY pattern.

Loop optimization

Informational | MI – 07 | Resolved

There is the usage of loops in the codebase with initialization of the iterator to 0 in the first iteration. This initialization can be avoided due to the default uint256 value.

Recommendation:

Consider optimization in the loops.

Typos in the codebase

Informational | MI – 08 | Resolved

Thera are some typos in the contracts code:

- Typo in NatSpec comment, appears in UserEntity and ExpandedEntity contracts : OwnerManger;
- Error name, appears in EntityFactory, ExpandedEntity: IncorectEntityType.

Recommendation:

Consider fixing typos.

Useless IF statement

Informational | MI – 09 | Resolved

There are duplicated if statements in the *cancelWithdraw()* function of the SwapMILE contract. In lines L382 and L385 there is a check if the requested withdrawal id belongs to the correct entity but in each IF statement, a different error is reverted.

Recommendation:

Consider sticking to the one if statement to check if entity and withdraw id is appropriate:

```
!_withdrawalRequestIdsByEntity[entity].contains(withdrawalId)
```

Partial use of SafeERC20Upgradeable library

Informational | TI – 01 | Resolved

The BaseEntity Smart Contract uses SafeERC20Upgradeable that provides *safeTransfer()* for *withdrawFromEntity()* function but don't use *safeApprove()* or *safeIncreaseAllowance()* instead of *approve()* in the *approve()* function of the BaseEntity contract. It will decrease contract code to save some gas.

Recommendation:

Consider using SafeERC20Upgradeable *safeApprove()* or *safeIncreaseAllowance()* instead of checking returned value from *approve()* and reverting error to safely operate with approving tokens and save some gas value.

Requirements instead of custom errors reverting

Informational | TI – 02 | Resolved

In the WhiteList, Strategy, SMilestoneBasedToken Smart Contracts there are used requirements to revert errors but in most contracts custom revertings are used which use less gas.

Recommendation:

Consider changing requirements to the custom errors reverting of mentioned contracts.

Useless admin role

Informational | TI – 03 | Resolved

In the SMilestoneBasedToken Smart Contract there is a declared *ADMIN_ROLE* that is used instead of initialized and unused *DEFAULT_ADMIN_ROLE*.

Recommendation:

Consider removing *ADMIN_ROLE* to decrease contract code. Use *DEFAULT_ADMIN_ROLE* instead of it.

msg.sender usage

Informational | TI – 04 | Resolved

In SingleSignEntityStrategy, SMilestoneBasedToken, Create2Factory, WhiteList Smart Contracts could be used *_msgSender()* function instead of msg.sender because it is inherited from Context Smart Contract. In addition, the *_msgSender()* function call requires less gas than msg.sender.

Recommendation:

Consider changing msg.sender to *_msgSender()*.

Duplicated modifier

Informational | TI – 05 | Resolved

In the SwapMILE Smart Contract there is a declared *onlyAdmin()* modifier that is used instead of default *onlyRole()* modifier from inherited AccessControlUpgradeable Smart Contract.

Recommendation:

Consider using *onlyRole()* modifier instead of *onlyAdmin()*.

Optimization by ternary operator

Informational | TI – 06 | Resolved

In the SwapMILE Smart Contract there is _convertToSMILE() and _convertToMILE() functions that can be optimized to return result of ternary operator that will decrease using of if-else branching:

```
 function _convertToSMILE(uint256 _amount) internal view returns
(uint256) {
    return
      totalPool > 0 && sMILEPool > 0
        ? (sMILEPool * _amount) / totalPool
        : _amount;
  }

  function _convertToMILE(uint256 _amount) internal view returns
(uint256) {
    return
      totalPool > 0 && sMILEPool > 0
        ? (totalPool * _amount) / sMILEPool
        : _amount;
  }
```

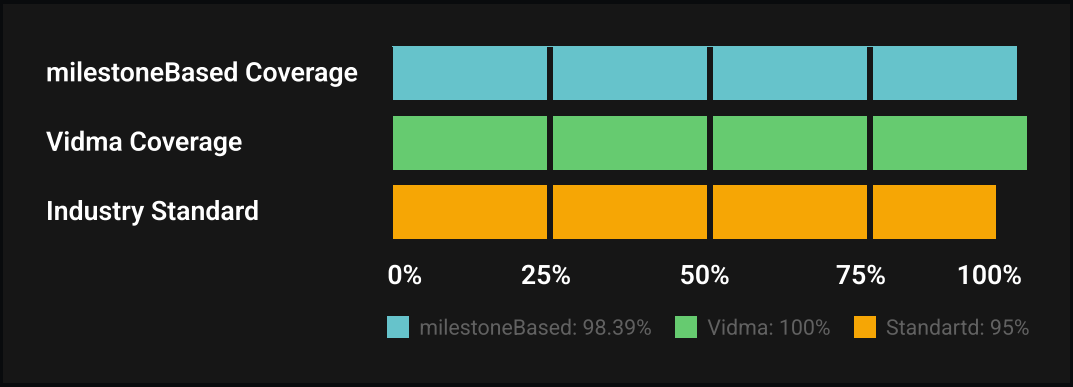
Recommendation:

Consider using ternary operators. It is also can be used in the *_calculateWithdrawFee()* function.

TEST RESULTS

To verify the security of contracts and the performance, a number of integration tests were carried out using the Hardhat testing framework.

In this section, we provide both tests written by milestoneBased and tests written by Vidma auditors.



It is important to note that Vidma auditors do not modify, edit or add tests to the existing tests provided in the milestoneBased repository. We write totally separate tests with code coverage of a minimum of 95% to meet the industry standards.

Tests written by milestoneBased

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	99.45	93.56	98.41	97.52
Constants.sol	100.00	100.00	100.00	100.00
ContractsRegistry.sol	100.00	100.00	100.00	100.00
Create2Factory.sol	100.00	100.00	100.00	100.00
EntityFactory.sol	100.00	100.00	100.00	100.00
SMilestoneBasedToken.sol	100.00	80.00	100.00	92.86
SwapMILE.sol	99.12	95.83	96.55	96.45
WhiteList.sol	100.00	77.78	100.00	100.00
contracts\entities\	100.00	100.00	100.00	100.00
CompanyEntity.sol	100.00	100.00	100.00	100.00
ProjectEntity.sol	100.00	100.00	100.00	100.00
UserEntity.sol	100.00	100.00	100.00	100.00
contracts\entities\abstract\	100.00	93.55	100.00	95.83
BaseEntity.sol	100.00	86.67	100.00	90.00

File	% Stmts	% Branch	% Funcs	% Lines
ExpandedEntity.sol	100.00	100.00	100.00	100.00
OwnerManager.sol	100.00	100.00	100.00	100.00
contracts\entities\interfaces\	100.00	100.00	100.00	100.00
IBaseEntity.sol	100.00	100.00	100.00	100.00
ICompanyEntity.sol	100.00	100.00	100.00	100.00
IExpandedEntity.sol	100.00	100.00	100.00	100.00
IOwnerManager.sol	100.00	100.00	100.00	100.00
IProjectEntity.sol	100.00	100.00	100.00	100.00
IUserEntity.sol	100.00	100.00	100.00	100.00
contracts\strategies\	82.35	80.77	88.89	80.65
SingleSignEntityStrategy.sol	100.00	100.00	100.00	100.00
Strategy.sol	50.00	37.50	66.67	45.45
contracts\strategies\interfaces\	100.00	100.00	100.00	100.00
ISingleSignEntityStrategy.sol	100.00	100.00	100.00	100.00
IStrategy.sol	100.00	100.00	100.00	100.00
IVotingStrategy.sol	100.00	100.00	100.00	100.00

File	% Stmts	% Branch	% Funcs	% Lines
All Files	98.39	92.57	97.87	95.92

Test Results

```
Contract: Create2Factory
  initialization
    ✓ should return the address of the future contract
      correctly (73ms)
  deploy
    ✓ should deploy contract via create2 opcode correctly (62ms)
    ✓ should catch event
  setContractOwner
    ✓ should transfer ownership correctly (79ms)
    ✓ should revert if created contract doesn't inherited
      Ownable contract (76ms)
    ✓ should revert if the contract isn't owner of passed
      contract (82ms)

Contract: MilestoneBased
  Constructor
    ✓ should initialize variables correctly
  Create Roadmap
    ✓ should create roadmap and voting contract by provided
      parameters correctly (123ms)
    ✓ should catch event
  setWhiteList
    ✓ should fail if sender isn't owner
    ✓ should fail if new address is zero address
    ✓ should update whiteList address correctly
  setEntityFactory
    ✓ should fail if sender isn't owner
    ✓ should fail if new address is zero address
    ✓ should update entityFactory address correctly

Contract: RoadmapMilestone
  ✓ cannot initialize twice
  ✓ should initialize correctly
```

```
set refunding contract
  ✓ cannot set if caller is not voting contract
  ✓ cannot set if address is zero
  ✓ cannot set in not Funding state (38ms)
  ✓ should set correctly
  ✓ should emit event on set
set voting contract
  ✓ cannot set if caller is not voting contract
  ✓ cannot set if address is zero
  ✓ cannot set in not Funding state (84ms)
  ✓ should set correctly
  ✓ should emit event on set
add milestone
  ✓ cannot add if caller is not voting contract
  ✓ cannot add if in not Funding state (49ms)
  ✓ cannot add if roadmap balance is less than required
    for the milestone (67ms)
  ✓ cannot add if milestone already exists (78ms)
  ✓ cannot add if start end date is 0
  ✓ cannot add if start end is later than end date
  ✓ cannot add already started milestone
  ✓ should add correctly (62ms)
  ✓ should add multi milestones correctly (118ms)
  ✓ should emit event on add (38ms)
update milestone
  ✓ cannot update if caller is not voting contract
  ✓ cannot update if in not Funding state (43ms)
  ✓ cannot update non-existent milestone
  if milestone not started yet
    ✓ cannot update if start end date is 0
    ✓ cannot update if new amount greater than amount of funded
      tokens
    ✓ should update correctly (79ms)
    ✓ should emit event on update
  if milestone started
    ✓ cannot update if start date different then original
    ✓ cannot update if end date < now
    ✓ cannot update already started milestone if new amount -
      withdrawn amount > funded tokens amount (90ms)
    ✓ cannot update already started milestone if new amount <
      withdrawn amount
```

```
      ✓ should update several milestones with withdrawn tokens
        for the correctly (229ms)
      ✓ should update correctly (71ms)
      ✓ should emit event on update
  remove milestone
    ✓ cannot remove if caller is not voting contract
    ✓ cannot remove non existent milestone
    ✓ cannot remove if in not Funding state
    ✓ cannot remove already started milestone
    ✓ should update locked funds value correctly after
      remove milestone
    ✓ should update locked funds value correctly after remove
      milestone with withdrawn tokens (57ms)
    ✓ should remove correctly
    ✓ should emit event on remove
  update milestone voting status
    ✓ cannot update if caller is not voting contract
    ✓ cannot update if in not Funding state
    ✓ cannot update if milestone does not exist
    ✓ cannot update back to Active
    ✓ cannot update back to Suspended in Finished Status
    ✓ cannot update to Finished in Suspended Status
    ✓ cannot update to Suspended in Suspended Status
    ✓ cannot update to Finished in Finished Status
    ✓ cannot update before start date
    ✓ should update to Suspended correctly
    ✓ should update to Finished correctly
    ✓ should emit event on update
  update roadmap state
    ✓ cannot update if caller is not voting contract
    ✓ hould update to refunding state correctly (50ms)
    ✓ cannot update back to funding state (39ms)

Contract: SMilestoneBasedToken
  constructor
    ✓ should grant roles to deployer correctly
  DEFAULT_ADMIN_ROLE
    ✓ should grant DEFAULT_ADMIN_ROLE correctly
  mint
    ✓ should revert if caller does not have permission to mint
    ✓ should mint tokens correctly
  transfer
```

```
    ✓ should revert if transfer from and to invalid address (64ms)
    ✓ should transfer tokens correctly (47ms)
  updating Registry
    ✓ should revert if sender isn't owner
    ✓ should update strategy of sMILE in registry correctly (47ms)
    ✓ should update whiteList in registry correctly (75ms)
  setRegistry
    ✓ should revert if new address is zero address
    ✓ should update registry correctly
    ✓ should catch event

Contract: SingleSignVotingStrategy
  constructor
    ✓ should deploy with correct trusted signer address
    ✓ should deploy with correct signature generation resource url
  functions:
    setTrustedSigner
      ✓ should revert if caller is not owner
      ✓ should revert if trusted signer with such an address is
        already setted
      ✓ should set trusted signer correctly
      ✓ should catch event
    setUrl
      ✓ should revert if caller is not owner
      ✓ should revert if this url is already set
      ✓ should set url correctly
      ✓ should catch event
    url
      ✓ should return url correctly
    isValid
      ✓ should return true with a valid signature (40ms)
      ✓ should return false with an invalid signature
      ✓ should revert if argumentsU256 not one
      ✓ should revert if argumentsB32 not two

Contract: MilestoneBasedVesting
  Constructor
    ✓ Must fail if passed zero address of token
    ✓ Must set up correctly (45ms)
  addTokensForVesting
    ✓ Must fail if sender isn't owner
    ✓ Must fail if passed amount equal to zero
```

```
    ✓ Must fail if sender hasn't approved tokens for vesting
      contract
    ✓ Must add tokens for vesting to vesting contract (51ms)
  createVesting
    ✓ Must fail if sender isn't owner
    ✓ Must fail if passed zero address of beneficiary
    ✓ Must fail if amount of vesting equal to 0
    ✓ Must fail if passed incorrect vesting period
    ✓ Must fail if passed incorrect vesting period
    ✓ Must fail if amount of vesting bigger then allocation
    ✓ Must create vesting correctly
  createVestingBatch
    ✓ Must fail if sender isn't owner
    ✓ Must fail if passed arrays have different length
    ✓ Must create vesting multiple times for different users
    ✓ Must create vestings with different type for one user
  emergencyWithdraw
    ✓ Must fail if sender isn't owner
    ✓ Must withdraw correct amount of tokens from vesting
      contract (60ms)
    ✓ Must fail if there is no available tokens to withdraw
  testing vesting types calculation and withdraw
    ✓ Withdraw function must fail if user doesn't have vestings
    ✓ All vesting must return 0 if vestings haven't started
      yet (43ms)
    ✓ Vesting type "Marketing" must give small part of vested
      tokens after creation of vesting
    ✓ Vestings with types "Founder" and "Rewards" must return 0
      while lock period
    ✓ Vesting with type "Rewards" must return part of tokens after
      lock period
    ✓ Must return correct amount of withdrawable tokens for user
      with multiple vestings after withdraw (85ms)
    ✓ Vesting type "Founder" must return correct amount of
      tokens (210ms)
    ✓ Vestings must return all vested tokens after vesting's
      end (188ms)
  revokeVestingOfUser
    ✓ Must fail if user's vesting is irrevocable (102ms)
    ✓ Must revoke vesting correctly
    ✓ Must fail if vesting has been already revoked
```

```
    ✓ Must revoke correct amount of user's tokens if user has
      claimed tokens (56ms)
  burn
    ✓ Must fail if sender doesn't have enougth tokens to burn (46ms)
    ✓ Must burn tokens correctly
  burnFrom
    ✓ Must fail if sender doesn't have enougth allowance
    ✓ Must fail if owner doesn't have enougth tokens
    ✓ Must burn tokens from another user correctly

Contract: Voting
  functions:
    [editing functionality]
      ✓ should revert if caller is not a voting
      ✓ should change properties of the contract correctly (184ms)
      ✓ should catch events
    addProposal
      ✓ should add new proposal correctly
      ✓ should revert if arrays have different length
      ✓ should revert if subarrays have different length
      ✓ should revert if targets array is empty
      ✓ should catch event
    vote
      ✓ hould votes in proposal correctly (60ms)
      ✓ should revote correctly (45ms)
      ✓ should catch event
      ✓ should revert with invalid signature (58ms)
      ✓ should revert with invalid optionId (46ms)
      ✓ cannot vote with the same options (45ms)
    cancelVote
      ✓ should cancel vote correctly (39ms)
      ✓ should revert if vote not exist
      ✓ should revert if vote not in proposal time interval
      ✓ should catch event
    execute
      ✓ should execute an option in a proposal correctly (107ms)
      ✓ should catch event
      ✓ should revert if proposal does not exist
      ✓ should revert if not maximum power option
      ✓ should revert if option already executed (38ms)
      ✓ should revert if minConsensusVotingPower greater than
        total voting power in this proposal (115ms)
```

```
      ✓ should revert if minConsensusVotersCount greater than total
        voters in this proposal (101ms)
      ✓ should revert with invalid callTargets (118ms)
    getOptionCount
      ✓ should return option count correctly
      ✓ should revert if proposal does not exist
    getOptions
      ✓ should return options correctly (51ms)
      ✓ should revert if proposal does not exist
    proposalsCount
      ✓ should return proposals count correctly
    proposalExists
      ✓ should return true if proposal exist
      ✓ should return false if proposal not exist
    proposalTimeInterval
      ✓ should return LockBeforeVoting time interval correctly
      ✓ should return Voting time interval correctly
      ✓ should return LockBeforeExecution time interval correctly
      ✓ should return Execution time interval correctly
      ✓ should return AfterExecution time interval correctly
    maxVotingPowerOption
      ✓ should return information about option with a maximum
        voting power for proposal correctly

Contract: WhiteList
  constructor
    ✓ should grant roles to deployer correctly
  addNewAddress
    ✓ should revert if caller is not owner or MB contract
    ✓ should add new address to white list correctly
    ✓ should catch event
  addNewAddressesBatch
    ✓ should add new addresses to white list correctly
    ✓ should catch event
  removeAddress
    ✓ should revert if caller is not owner or MB contract
    ✓ should remove address from white list correctly
    ✓ should catch event
  removeAddressesBatch
    ✓ should remove addresses from white list correctly
    ✓ should catch event
  getAllAddresses
```

```
    ✓ should return empty array
    ✓ should return all addresses correctly
    ✓ should return first part of all addresses correctly
    ✓ should return one address with offset(length - 1)
    ✓ should return empty array if offset is greater or equal
      array length
    ✓ should return empty array if limit is 0
  isValidAddress
    ✓ should return true if contract has such address, and false
      if doesn't

Contract: ContractRegistry
  initialize
    ✓ should correct set owner after initialize
    ✓ should revert if try initialize second time
  registerContract
    should revert if
      ✓ call from not Owner
      ✓ try register zero address
    should success
      register first key and emit event
        ✓ check corect registration
        ✓ check emit event
      register second key and emit event
        ✓ check corect registration
        ✓ check emit event
        ✓ check first key not rewrite
      update contract address after call second time with
      registered key
        ✓ check corect registration
        ✓ check emit event
  unregisterContract
    should revert if
      ✓ call from not Owner
      ✓ try call unregister for not registtered key
    should success
      unregister first key and emit event
        ✓ check corect unregistration
        ✓ check emit event
        ✓ check second key not unregister
      unregister second key and emit event
        ✓ check corect unregistration
```

```
      ✓ check emit event
  registerContracts
    should revert if
      ✓ call from not Owner
      ✓ try register zero address
      ✓ send array with different length
    should success
      register all keys and emit event
        ✓ check emit event
        should correct
          ✓ check registration of key: 0
          ✓ check registration of key: 1
          ✓ check registration of key: 2
  isRegistered
    ✓ Should return TRUE if key registered
    ✓ Should return FALSE if key not registered
  register
    ✓ Should return correct address by key
    ✓ Should return zero address if key not registered
  getContractByKey
    ✓ Should return correct address by key
    ✓ Should revert if key not registered
  getContractsByKeys
    ✓ Should return correct addresses by keys
    ✓ Should return zero array if send empty keys array
    ✓ Should revert if one from keys not registered

Contract: BaseEntity
  initialize
    ✓ should correct set owner after initialize
    ✓ should correct set contractRegistry address after initialize
    ✓ should revert if try initialize second time
    ✓ should revert if call from not initialize function
    ✓ should revert if try provide invalid owner address (75ms)
    ✓ should revert if try provide invalid factory address (38ms)
  functions
    withdrawFromEntity
      should fail
        ✓ if caller not owner
        ✓ if not enought erc20 tokens on contract
        ✓ if not enought native coins on contract
      should success
```

```
      transfer (100% - 1 token) erc20 from contract to recipient
      and emit event
        ✓ correct change balances
        ✓ correct emit event
      transfer all erc20 from contract to recipient and emit
      event
        ✓ correct change balances
        ✓ correct emit event
      transfer (100% - 0.1 ether) coin from contract to
      recipient and emit event
        ✓ correct change balances
        ✓ correct emit event
      transfer all native coins from contract to recipient
      and emit event
        ✓ correct change balances
        ✓ correct emit event
  approve
    should fail
      ✓ if caller not owner
      ✓ if inner approve return false
    should success approve tokens and emit event
      ✓ correct change allowance
      ✓ correct emit event
    should success approve tokens to less amount and emit event
      ✓ correct change allowance
      ✓ correct emit event
  increaseAllowance
    should fail
      ✓ if caller not owner
      ✓ if caller pass zero address as spender
      ✓ if caller pass zero amount
    should success increaseAllowance tokens and emit event
      ✓ correct change allowance
      ✓ correct emit event
  decreaseAllowance
    should fail
      ✓ if caller not owner
      ✓ if caller pass zero address as spender
      ✓ if caller pass zero amount
      ✓ if caller try to decrease more than approved
    should success decreaseAllowance tokens and emit event
      ✓ correct change allowance
```

```
        ✓ correct emit event
    _getEntityFactory
      ✓ Should corectly return actual entityFactory from registry

Contract: CompanyEntity
  initialize
    ✓ should correct set owner after initialize
    ✓ should correct set contractsRegistry address after initialize
    ✓ should revert if try initialize second time
    ✓ should revert if try provide not UserEntity address as
      owner (47ms)

Contract: ExpandedEntity
  initialize
    ✓ should correct set owner after initialize (39ms)
    ✓ should correct set contractsRegistry address after initialize
    ✓ should revert if try initialize second time
    ✓ should revert if call from not initialize function
    ✓ should revert if try provide not UserEntity address as
      owner (58ms)
  functions
    isOwner
      should success
        ✓ return TRUE if user is owner of userEntity
        ✓ return TRUE if userEntity is owner of ExpandedEntity
        ✓ return FALSE if user not owner
        ✓ return FALSE if provided another userEntity which
          not owner
    swapOwner
      should fail if
        ✓ caller not one from owners
        ✓ new owner is not UserEntity address
      should success
        corectly change old userEntity owner to new userEntity
        and emits event
          ✓ should correct emits events
          ✓ should correct return new owner
          ✓ should correct remove owner status from old owner
          ✓ should correct set owner status for new owner

Contract: OwnerManager
  initialize
    ✓ should correct set owner after initialize
```

```
    ✓ should correct emit event
    ✓ should revert if try initialize second time
    ✓ should revert if call from not initialize function
    ✓ should revert if try provide invalid address as
      owner (48ms)
  functions
    getOwners
      should success
        ✓ return array with actual owner
        ✓ return array if owners is more then one
    isOwner
      should success
        ✓ return TRUE if user is owner
        ✓ return FALSE if user not owner
        ✓ return FALSE if provided address is _SENTINEL_OWNERS
    swapOwner
      should fail if
        ✓ caller not one from owners
        ✓ new owner is active owner in contract
        ✓ old owner is invalid address
        ✓ new owner is invalid address
        ✓ previus owner is not previous for old owner
      should success
        corectly change old owner to new and emits event
          ✓ should correct emits events
          ✓ should correct return new owner
          ✓ should correct remove owner status from old owner
          ✓ should correct set owner status for new owner
          ✓ old owner shouldn't have rights to call swapOwner
            again
          ✓ getFirstOwner should return new owner
        corectly change with a lot of owners
          ✓ corectly change with a lot of owners (70ms)

Contract: ProjectEntity
  initialize
    ✓ should correct set owner after initialize
    ✓ should correct set contractsRegistry address after initialize
    ✓ should revert if try initialize second time
    ✓ should revert if try provide not UserEntity address as
      owner (45ms)
```

```
Contract: ProjectEntity
  initialize
    ✓ should correct set owner after initialize
    ✓ should correct set contractsRegistry address after initialize
    ✓ should revert if try initialize second time
  swapOwner
    should fail if
      ✓ caller not one from owners
    should success
      corectly change old userEntity owner to new userEntity and
      emits event
        ✓ MPORTANT: should correct change owner in EntityFactory
        ✓ should correct emits events
        ✓ should correct return new owner
        ✓ should correct remove owner status from old owner
        ✓ should correct set owner status for new owner

Contract: EntityFactory
  initialize
    ✓ should correct set owner after initialize
    ✓ should correct set contractregistry address after initialize
    ✓ should correct set upgradeableBeacon addresses by EntityTypes
    ✓ should revert if try initialize second time
    ✓ should revert if provide zero address to any parameters (65ms)
  updateUpgradeableBeacon
    should fail
      ✓ if caller is not owner
      ✓ if provide zero address
      ✓ if provide incorrect entity type
    should success update beacon and emit event
      ✓ UserEntity
      ✓ ProjectEntity
      ✓ CompanyEntity
  createEntity
    should fail
      ✓ if provide incorrect entityType
      ✓ if provide incorrect signature (51ms)
    create UserEntity
      should fail
        ✓ if user have exists userEntity (119ms)
      should success create userEntity and emit event
        ✓ correct emit event
```

```
    ✓ correct add entity address to whitelist
    ✓ correct register entity with correct type
    ✓ correct set initial owner
    Check user entity on correct deploy
      ✓ should correct setup owner
      ✓ should correct setup contractRegistry
      ✓ should correct setup beacon of Proxy
  should success create userEntity for another user and
  emit event
    ✓ correct emit event
    ✓ correct add entity address to whitelist
    ✓ correct register entity with correct type
    ✓ correct set initial owner
    Check user entity on correct deploy
      ✓ should correct setup owner
      ✓ should correct setup contractRegistry
      ✓ should correct setup beacon of Proxy
create CompanyEntity
  should fail
    ✓ if user haven't exists userEntity (54ms)
  should success create CompanyEntity and emit event
    ✓ correct emit event
    ✓ correct add entity address to whitelist
    ✓ correct register entity with correct type
    ✓ correct set initial owner as UserEntity from owner
    Check user entity on correct deploy
      ✓ should correct setup owner
      ✓ should correct setup contractRegistry
      ✓ should correct setup beacon of Proxy
    should success create second CompanyEntity for one user
      ✓ check thats it's full new entity
  should success create CompanyEntity for another user
    ✓ correct emit event
    ✓ correct add entity address to whitelist
    ✓ correct register entity with correct type
    ✓ correct set initial owner as UserEntity from owner
    Check user entity on correct deploy
      ✓ should correct setup owner
      ✓ should correct setup contractRegistry
      ✓ should correct setup beacon of Proxy
    should success create second CompanyEntity for one user
      ✓ check thats it's full new entity
```

```
create ProjectEntity
  should fail
    ✓ if user haven't exists userEntity (52ms)
  should success create ProjectEntity and emit event
    ✓ correct emit event
    ✓ correct add entity address to whitelist
    ✓ correct register entity with correct type
    ✓ correct set initial owner as UserEntity from owner
    Check user entity on correct deploy
      ✓ should correct setup owner
      ✓ should correct setup contractRegistry
      ✓ should correct setup beacon of Proxy
    should success create second ProjectEntity for one user
      ✓ check thats it's full new entity
  should success create ProjectEntity for another user
    ✓ correct emit event
    ✓ correct add entity address to whitelist
    ✓ correct register entity with correct type
    ✓ correct set initial owner as UserEntity from owner
    Check user entity on correct deploy
      ✓ should correct setup owner
      ✓ should correct setup contractRegistry
      ✓ should correct setup beacon of Proxy
    should success create second ProjectEntity for one user
      ✓ check thats it's full new entity
isOwnerOfUserEntity
  ✓ should return FALSE if user not have userEntity
  ✓ should return TRUE if user have userEntity
addOwnerToUserEntity
  should fail
    ✓ if caller is not registered userEntity
    ✓ if entity already exists for new owner
  should success add new owner for userEntity and emit event
    ✓ should correct check userEntity by new owner
    ✓ corect emit event
removeOwnerFromUserEntity
  should fail
    ✓ if caller is not registered userEntity
  should success remove owner from userEntity and emit event
    ✓ should correct check userEntity by old owner
    ✓ corect emit event
```

```
Contract: SingleSignEntityStrategy
  initialize
    ✓ should correct set owner after initialize
    ✓ should correct set contractsRegistry contract
    ✓ should correct set trustedSigner
    ✓ should revert if try deploy with zero trusted signer addres
    ✓ should revert if try deploy with zero entity factory addres
  functions
    setTrustedSigner
      ✓ should revert if caller is not owner
      ✓ should revert if trusted signer try set zero address
      ✓ should set trusted signer correctly and emit event
    isValid
      ✓ should return FALSE if nonce is used
      ✓ should return FALSE if signature expired
      ✓ should return FALSE if signer is not equal trusted signer
      ✓ should return TRUE if signer is actual trusted signer
      should return FALSE if any from param was changed
        ✓ if changed deadline
        ✓ if changed signer
        ✓ if changed id (40ms)
        ✓ if changed nonce
        ✓ if changed entityFactory
        ✓ if changed entity owner
        ✓ if changed entityType
        ✓ if changed v
        ✓ if changed r
        ✓ if changed s
    useSignature
      should revert
        ✓ should revert if caller is not entityFactory
        ✓ should revert if signature is invalid
      should correctly use signature
        ✓ should corecrty use signature and set used nonce
        ✓ Should corectly set nonce is used for user
        ✓ should revert if used nonce which was use before

Contract: SwapMile
  initialize
    ✓ should revert when one of required addresses is zero (76ms)
    ✓ should revert when cooldownPeriod or withdrawPeriod equal
      to 0 (61ms)
```

```
  ✓ should revert when cooldownPeriod is greater than
    withdrawPeriod or equal (71ms)
  ✓ should revert if the contract already initialized
staking
  ✓ Should revert if passed entity isn't entity (99ms)
  ✓ Should revert if sender isn't address of passed entity (90ms)
  ✓ Should revert if amount of token to stake equal to Zero (126ms)
  ✓ should revert to stake if user haven't approved enough
    tokens to stake (106ms)
  ✓ should revert to stake if user doesn't have enough tokens
    to stake (128ms)
  ✓ should stake from sender wallet (265ms)
  ✓ should stake from userEntity (273ms)
  ✓ should stake from passed entity (266ms)
requestWithdraw
  ✓ Should revert if passed address isn't entity
  ✓ Should revert if sender isn't Owner of passed entity
  ✓ Should revert if passed zero amount of tokens
  ✓ Should revert if entiti's balance of sMILE tokens less
    than required amount to withdraw
  ✓ Should create request withdraw correctly (51ms)
cancel withdraw
  ✓ Should revert if passed address isn't entity
  ✓ Should revert if sender isn't owner of passed entity
  ✓ Should revert if passed withdrawal Id is incorrect
  ✓ should cancel correctly (65ms)
withdraw
  ✓ Should revert if passed address isn't entity
  ✓ Should revert if sender isn't owner of passed entity
  ✓ Should revert if request does not exist
  ✓ Should revert if passed request was created by not passed
    entity
  ✓ Should revert if entity doesn't have enough sMILE tokens (88ms)
  ✓ should cancel withdraw request after withdraw period (58ms)
  ✓ should withdraw correctly after cooldown period (103ms)
fee calculation check
  ✓ should get fee after immidiatly withdraw (56ms)
  ✓ should get fee after 1/5 cooldown period (62ms)
  ✓ should get fee after 1/2 cooldown period (64ms)
  ✓ should get fee after 3/5 cooldown period (62ms)
  ✓ should get fee after cooldown period (61ms)
set cooldown and withdraw period
```

```
    ✓ should revert if sender isn't admin (91ms)
    ✓ should revert if new value is zero
    ✓ should revert to set cooldownPeriod if it greater or
      equal to withdrawPeriod
    ✓ should revert to set withdrawPeriod if it smaller or equal
      to cooldownPeriod
    ✓ should update cooldownPeriod correctly
    ✓ should update withdrawPeriod correctly
  addRewards
    ✓ should revert if passed zero amount of tokens
    ✓ should revert if sender doesn't have enogth MILE tokens
    ✓ should revert if sender doesn't have enogth MILE tokens
    ✓ should add rewards correctly (64ms)
  withdrawUnusedFunds
    ✓ should revert if sender isn't admin (54ms)
    ✓ should revert if passed zero amount of tokens
    ✓ should revert if passed zero address
    ✓ should revert of owner tries withdraw MILE tokens from pool
    ✓ should withdraw unused MILE (46ms)
    ✓ should withdraw unused BNB (55ms)
    ✓ should withdraw unused ERC20 token (48ms)
  getRequestIdsByEntity & getRequestsByEntity
    ✓ should return empty array if entity doesn't have any
      withdrawal requests
    ✓ should return empty array due to incorrect limit
    ✓ should return empty array due to incorrect offset
    ✓ should return correct array of withdrawal requests (48ms)
    ✓ should return correct array of withdrawal request with
      offset and limit
  get available sMILE to withdraw
    ✓ should return 0 if entity doesn't have sMILE tokens (46ms)
    ✓ should return amount of stake tokens
    ✓ must change amount of withdrawable tokens after creating
      withdraw request (42ms)

  474 passing (23s)
```

Tests written by Vidma auditors

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	100.00	100.00	100.00
Constants.sol	100.00	100.00	100.00	100.00
ContractsRegistry.sol	100.00	100.00	100.00	100.00
Create2Factory.sol	100.00	100.00	100.00	100.00
EntityFactory.sol	100.00	100.00	100.00	100.00
SMilestoneBasedToken.sol	100.00	100.00	100.00	100.00
SwapMILE.sol	100.00	100.00	100.00	100.00
WhiteList.sol	100.00	100.00	100.00	100.00
contracts\entities\	100.00	100.00	100.00	100.00
CompanyEntity.sol	100.00	100.00	100.00	100.00
ProjectEntity.sol	100.00	100.00	100.00	100.00
UserEntity.sol	100.00	100.00	100.00	100.00
contracts\entities\abstract\	100.00	100.00	100.00	100.00
BaseEntity.sol	100.00	100.00	100.00	100.00

File	% Stmts	% Branch	% Funcs	% Lines
ExpandedEntity.sol	100.00	100.00	100.00	100.00
OwnerManager.sol	100.00	100.00	100.00	100.00
contracts\entities\interfaces\	100.00	100.00	100.00	100.00
IBaseEntity.sol	100.00	100.00	100.00	100.00
ICompanyEntity.sol	100.00	100.00	100.00	100.00
IExpandedEntity.sol	100.00	100.00	100.00	100.00
IOwnerManager.sol	100.00	100.00	100.00	100.00
IProjectEntity.sol	100.00	100.00	100.00	100.00
IUserEntity.sol	100.00	100.00	100.00	100.00
contracts\strategies\	100.00	100.00	100.00	100.00
SingleSignEntityStrategy.sol	100.00	100.00	100.00	100.00
Strategy.sol	100.00	100.00	100.00	100.00
contracts\strategies\interfaces\	100.00	100.00	100.00	100.00
ISingleSignEntityStrategy.sol	100.00	100.00	100.00	100.00
IStrategy.sol	100.00	100.00	100.00	100.00
IVotingStrategy.sol	100.00	100.00	100.00	100.00

File	% Stmts	% Branch	% Funcs	% Lines
All Files	100.00	100.00	100.00	100.00

Test Results

```
Contract: ContractsRegistry
  Deployment / Initialization
    ✓ should revert if register has already initialized before
    ✓ check owner
  Functions
    registerContract
      ✓ should revert if caller is not the owner
      ✓ should revert if contract address argument is a zero
        address
      ✓ should register contract address correctly (53ms)
    registerContracts
      ✓ should revert if caller is not the owner
      ✓ should revert if arrays of keys and addresses have not
        equal lenghts
      ✓ should register contract addresses correctly (54ms)
    unregisterContract
      ✓ should revert if caller is not the owner
      ✓ should revert if contract address was not registered
      ✓ should unregister contract address correctly (54ms)
    isRegistered
      ✓ should return false value if key is not registered
      ✓ should return true value if key is registered (45ms)
    getContractByKey
      ✓ should revert if key is not registered
      ✓ should return true value if key is registered
    getContractsByKeys
      ✓ should revert if at least one of keys is not registered
      ✓ should return contract addresses by keys correctly (53ms)

Contract: Create2Factory
  Functions
    deploy
```

```
      ✓ should revert if caller is not the owner
      ✓ should deploy contract correctly by the bytecode (59ms)
    setContractOwner
      ✓ should revert if caller is not the owner
      ✓ should set ownership of the deployed contract
        correctly (60ms)
    computeAddress
      ✓ should return computed address of contract by the bytecode

Contract: BaseEntity
  Deployment / Initialization
    ✓ should revert if contract has already initialized
    ✓ should revert if contract is not initialized
    ✓ should revert if contracts registry argument has zero
      address (41ms)
    ✓ check owner address
    ✓ check registry address
  Functions
    ✓ _getEntityFactory
    withdrawFromEntity
      ✓ should revert if caller is not the owner
      ✓ should revert if recipient argument is zero address
      ✓ should revert if amount argument is zero
      ✓ should withdraw tokens from entity correctly (47ms)
      ✓ should withdraw ether from entity correctly
      ✓ should revert reentrancy call (62ms)
    approve
      ✓ should revert if caller is not the owner
      ✓ should revert if recipient argument is a zero address
      ✓ should approve tokens from entity correctly
    increaseAllowance
      ✓ should revert if caller is not the owner
      ✓ should revert if recipient argument is a zero address
      ✓ should revert if amount argument is zero
      ✓ should increase allowance from entity correctly (42ms)
    decreaseAllowance
      ✓ should revert if caller is not the owner
      ✓ should revert if recipient argument is a zero address
      ✓ should revert if amount argument is zero
      ✓ should decrease allowance from entity correctly (63ms)

Contract: CompanyEntity
```

```
  Deployment / Initialization
    ✓ should revert if contract has already initialized
    ✓ should revert if owner argument is not user entity (51ms)
    ✓ check owner address
    ✓ check registry address
  Functions
    _swapOwner
      ✓ should revert if caller is not the owner
      ✓ should swap owner correctly (45ms)
    _isOwner
    ✓ should return true value if argument owner is owner of
      user entity
    ✓ should return true value if argument owner is owner
      of expended entity
    ✓ should return false value if argument owner is not owner
      of user or expended entity

Contract: ExpandedEntity
  Deployment / Initialization
    ✓ should revert if contract has already initialized
    ✓ should revert if contract is not initialized
    ✓ should revert if owner argument is not user entity (49ms)
    ✓ check owner address
    ✓ check registry address
  Functions
    _swapOwner
      ✓ should revert if caller is not the owner
      ✓ should swap owner correctly (48ms)
    _isOwner
      ✓ should return true value if argument owner is owner of
        user entity
      ✓ should return true value if argument owner is owner
        of expended entity
      ✓ should return false value if argument owner is not owner
        of user or expended entity

Contract: OwnerManager
  Deployment / Initialization
    ✓ should revert if it's second initialization
    ✓ should revert if contract is not initialized
    ✓ should revert if owner address is not valid (50ms)
    ✓ check owner address
```

```
    ✓ should correct emit event
  Functions
    getOwners
      ✓ should get array of owner addresses correctly
      ✓ should get array of owner addresses after adding new ones
        correctly (39ms)
    isOwner
      ✓ should return true value (46ms)
      ✓ should return false value
    swapOwner
      ✓ should revert if caller is not the owner
      ✓ should revert if new owner is already owner
      ✓ should revert if old owner argument has invalid address
      ✓ should revert if new owner argument has invalid address
      ✓ should revert if previus owner argument is not previous
        to old owner argument
      ✓ should swap owner correctly
    getFirstOwner
      ✓ should return first owner address
      ✓ should return first owner address after swap
  Test Cases
    ✓ multiple owner swapping (107ms)

Contract: ProjectEntity
  Deployment / Initialization
    ✓ should revert if contract has already initialized
    ✓ should revert if owner argument is not user entity (50ms)
    ✓ check owner address
    ✓ check registry address
  Functions
    _swapOwner
      ✓ should revert if caller is not the owner
      ✓ should swap owner correctly (48ms)
    _isOwner
      ✓ should return true value if argument owner is owner of
        user entity
      ✓ should return true value if argument owner is owner
        of expended entity
      ✓ should return false value if argument owner is not owner
        of user or expended entity

Contract: UserEntity
```

```
  Deployment / Initialization
    ✓ should revert if contract has already initialized
    ✓ check owner address
    ✓ check registry address
  Functions
    _swapOwner
      ✓ should revert if caller is not the owner
      ✓ should swap owner correctly (48ms)

Contract: EntityFactory
  Deployment / Initialization
    ✓ should revert if contract has already initialized
    ✓ should revert if at least one of the arguments is
      zero address (62ms)
    ✓ check owner address
    ✓ check entities addresses
    ✓ check register address
  Functions
    updateUpgradeableBeacon
      ✓ should revert if caller is not the owner
      ✓ should revert if proxy address is zero
      ✓ should revert if entity type is not selected
      ✓ should update upgradeable beacon correctly
    createEntity
      ✓ should revert if entity type argument is not provided
      ✓ should revert if signature is not valid for the user (62ms)
      ✓ should revert if user tried create user entity twice (136ms)
      ✓ should revert if user tried create company entity but
        he has no created user entity
      ✓ should create entity correctly (132ms)
      ✓ should create company/project entity correctly (273ms)
    isOwnerOfUserEntity
      ✓ should return true value if user argument is owner of
        userEntity (70ms)
      ✓ should return true value if user argument is not the owner
        of userEntity
    addOwnerToUserEntity
      ✓ should revert if caller is not user entity
      ✓ should revert if new owner argument already has user
        entity (109ms)
      ✓ should add owner to user entity correctly
    removeOwnerFromUserEntity
```

```
      ✓ should revert if caller is not user entity
      ✓ should remove owner from user entity correctly (40ms)

Contract: SMilestoneBasedToken
  Deployment / Initialization
    ✓ should revert if strategy argument address is zero address
    ✓ check default admin role role
    ✓ check register address
  Functions
    setRegistry
      ✓ should revert if caller is not admin (55ms)
      ✓ should revert if register argument address is zero address
      ✓ should set register correctly
    mint
      ✓ should revert if caller is not admin or swapper
      ✓ should mint correctly if caller is admin
      ✓ should mint correctly if caller is swapper
    _beforeTokenTransfer
      ✓ should revert if caller and receiver is not whitelisted
      ✓ should transfer correctly (45ms)

Contract: SingleSignEntityStrategy
  Deployment / Initialization
    ✓ should revert if trusted signer argument is a zero address
    ✓ should revert if contract register argument is a zero address
    ✓ check trusted signer address
    ✓ check owner address
    ✓ check register address
  Functions
    setTrustedSigner
      ✓ should revert if caller is not the owner
      ✓ should revert if signer address is a zero address
      ✓ should set trusted signer correctly
    useSignature
      ✓ should revert if caller is not the factory
      ✓ should revert if signature is invalid (45ms)
      ✓ should use signature correctly (44ms)
    isValid
      ✓ should return true value if signature is valid to the user
      ✓ should return false value if deadline is expired
      ✓ should return false value if nonce was already used
        for the user (48ms)
```

```
      ✓ should return false value if signature is not valid for
        the user

Contract: Strategy
  Deployment / Initialization
    ✓ should revert if whitelist argument address is zero address
    ✓ check register address
  Functions
    setRegistry
      ✓ should revert if caller is not the owner
      ✓ should revert if register argument address is zero address
      ✓ should set register correctly
    validateTransaction
      ✓ should revert if both addresses is not whitelisted
      ✓ should validate transaction correctly

Contract: SwapMILE
  Deployment / Initialization
    ✓ should revert if contract has already initialized
    ✓ should revert if admin or contract registry argument
      address is zero address (228ms)
    ✓ should revert if cooldown or withdraw period value is
      zero (77ms)
    ✓ should revert if withdraw period will be less or equal
      to cooldown period (62ms)
    ✓ check default admin role role
    ✓ check min fee
    ✓ check max fee
    ✓ check total
    ✓ check divider
    ✓ check register address
    ✓ check cooldown period
    ✓ check withdraw period
  Functions
    setCoolDownPeriod
      ✓ should revert if caller is not the admin (52ms)
      ✓ should revert if new value of cooldown period is zero
      ✓ should revert if new value of cooldown period will be
        more (or equal) than withdraw period
      ✓ should cooldown period correctly
    setWithdrawPeriod
      ✓ should revert if caller is not the admin (49ms)
```

```
    ✓ should revert if new value of withdraw period is zero
    ✓ should revert if new value of withdraw period will be
      less (or equal) than cooldown period
    ✓ should cooldown period correctly
  stake
    ✓ should revert if entity argument is not the entity
    ✓ should revert if caller is not the owner of entity
    ✓ should revert if staked amount is zero (48ms)
    ✓ should stake correctly (57ms)
    ✓ should stake correctly if user entity will be sender (58ms)
    ✓ should stake correctly if passed entity will be
      sender (68ms)
  stakeTo
    ✓ should revert if entity argument is not the entity
    ✓ should revert if entityTo argument is not the entity
    ✓ should revert if caller is not the owner of entity
    ✓ should revert if staked amount is zero (41ms)
    ✓ should stake correctly (63ms)
    ✓ should stake correctly if user entity will be sender (71ms)
    ✓ should stake correctly if passed entity will be sender (71ms)
  swapStake
    ✓ should revert if entity argument is not the entity
    ✓ should revert if caller is not the owner of entity
    ✓ should revert if staked amount is zero (50ms)
    ✓ should revert if token path is invalid (46ms)
    ✓ should revert if the last token is not the MILE token (39ms)
    ✓ should stake correctly (102ms)
    ✓ should stake correctly if user entity will be sender (114ms)
    ✓ should stake correctly if passed entity will be
      sender (125ms)
  swapStakeTo
    ✓ should revert if entity argument is not the entity
    ✓ should revert if entityTo argument is not the entity
    ✓ should revert if caller is not the owner of entity
    ✓ should revert if staked amount is zero (44ms)
    ✓ should stake correctly (117ms)
    ✓ should stake correctly if user entity will be sender (125ms)
    ✓ should stake correctly if passed entity will be
      sender (126ms)
  requestWithdraw
    ✓ should revert if entity argument is not the entity
    ✓ should revert if caller is not the owner of entity
```

```
    ✓ should revert if amount argument is zero
    ✓ should revert if caller has insufficient balance
    ✓ should revert if caller has insufficient balance
    ✓ should revert if withdrawal request already exists (45ms)
    ✓ should create withdrawal request correctly (51ms)
  withdraw
    ✓ should revert if entity argument is not the entity
    ✓ should revert if caller is not the owner of entity
    ✓ should revert if withdrawal request is not exist (45ms)
    ✓ should revert if entity argument has no selected id
    ✓ should cancel withdrawal request if withdraw period has
      ended (56ms)
    ✓ should withdraw correctly (80ms)
    ✓ should withdraw correctly before cooldown end with
      bigger fee (77ms)
  cancelWithdraw
    ✓ should revert if entity argument is not the entity
    ✓ should revert if caller is not the owner of entity
    ✓ should revert if withdrawal request is not exist
    ✓ should revert if entity argument has no selected id
    ✓ should revert if withdrawal request has already
      canceled (50ms)
    ✓ should cancel withdrawal request correctly (42ms)
  addRewards
    ✓ should revert if amount argument is zero
    ✓ should add rewards correctly (41ms)
  withdrawUnusedFunds
    ✓ should revert if caller is not the admin (50ms)
    ✓ should revert if amount argument is zero
    ✓ should revert if recipient has zero address
    ✓ should revert if contracts has insufficient balance
      of MILE tokens
    ✓ should withdraw native coins correctly
    ✓ should fail reentrancy attack (48ms)
    ✓ should withdraw MILE tokens correctly (45ms)
    ✓ should withdraw other tokens correctly (67ms)
  getRequestIdsByEntity
    ✓ should return withdrawal requests ids array correctly
  getRequestsByEntity
    ✓ should return empty array if entity has no withdrawal
      requests
```

```
      ✓ should return empty array if limit or offset statements
        are invalid
      ✓ should return withdrawal requests array correctly
      ✓ should return few withdrawal requests array correctly
    getAvailableSMILEToWithdraw
      ✓ should return 0 if it has no available sMILE tokens (43ms)
      ✓ should return available amount correctly (45ms)
    getLiquidityAmount
      ✓ should return total pool and sMile pool amount correctly

Contract: WhiteList
  Deployment / Initialization
    ✓ check admin role
    ✓ check factory role
  Functions
    addNewAddressesBatch
      ✓ should revert if caller is not the factory or admin
      ✓ should add addresses to whitelist correctly
    removeAddressesBatch
      ✓ should revert if caller is not the factory or admin
      ✓ should remove addresses from whitelist correctly (42ms)
    getWhitelistedAddresses
      ✓ should return empty array if offset argument is bigger
        than count of whitelisted addresses
      ✓ should return all addresses from whitelist
      ✓ should return only the first address
    isValidAddress
      ✓ should return true value if addresses are registered
      ✓ should return false value if addresses are
        unregistered (38ms)
    addNewAddress
      ✓ should revert if caller is not the factory or admin
      ✓ should add addresses to whitelist correctly
      ✓ should not add address to whitelist if it is already
        added (38ms)
    removeAddress
      ✓ should revert if caller is not the factory or admin
      ✓ should remove address from whitelist correctly
      ✓ should not remove address from whitelist if it does
        not exist there (51ms)

254 passing (12s)
```

We are delighted to have a chance to work with the milestoneBased team and contribute to your company's success by reviewing and certifying the security of your smart contracts.

The statements made in this document should be interpreted neither as investment or legal advice, nor should its authors be held accountable for decisions made based on this document.

Vidma is a security audit company helping crypto companies ensure their code and products operate safely and as intended, enabling founders to sleep soundly at night. We specialize in auditing DeFi protocols, layer one protocols, and marketplace solutions. Our team consists of experienced and internationally trained specialists. Our company is based in Ukraine, known for its strong engineering, cryptography, and cybersecurity culture.

Website: **vidma.io**
Email: **security@vidma.io**

VIDMA