# Blume Liquid Staking Smart Contract Report

## Table of Contents



---

Etherscan Contract Link : https://sepolia.etherscan.io/address/0xe8d871FD30aDA433c3BE8aA09D8aE61B33A09a2D#code

## 1. Introduction

### Purpose of the Document

This report documents the development and functionalities of the **Blume Liquid Staking** smart contract. It covers the technical details, implementation, and testing results.

### Overview of Liquid Staking

Liquid staking allows users to stake their cryptocurrency tokens and receive a derivative token in return. This derivative token can be traded or used in other DeFi applications while the original tokens remain staked.

---

## 2. Contract Design

### Overview of BLS and stBLS Tokens

- **BLS Token**: The native token of the Blume Liquid Staking platform, which users stake.
- **stBLS Token**: A derivative token minted when users stake their BLS tokens, representing the staked assets.

### Contract Architecture

- The **BlumeLiquidStaking** contract inherits from OpenZeppelin's **ERC20** and **Ownable** contracts, allowing for token standards and ownership management.
- The contract manages the relationship between BLS and stBLS tokens, enabling users to stake and unstake.

---

## 3. Functionalities

### Staking Mechanism

- Users can stake their BLS tokens by calling the `stake` function, which transfers BLS tokens to the contract and mints an equivalent amount of stBLS tokens to the user's address.

### Unstaking Mechanism

- Users can unstake by calling the `unstake` function, burning the stBLS tokens and transferring back the original BLS tokens.

### Events

- `Staked`: Emitted when tokens are staked.
- `Unstaked`: Emitted when tokens are unstaked.

---

## 4. Technical Development

### Development Environment

- **Solidity Version**: 0.8.0 or above
- **IDE**: Remix, Hardhat, or Truffle
- **Network**: Ethereum test network (Rinkeby, Goerli, etc.)

### Tools and Libraries Used

- OpenZeppelin Contracts: For secure implementations of ERC20 and Ownable contracts.
- SafeERC20: For secure token transfers.

### Testing Methodology

- Unit tests were written using JavaScript or TypeScript frameworks like Mocha or Jest.
- Each function (stake, unstake) was tested for expected outcomes, including edge cases.

---

## 5. Task Testing

### Test Cases

1. **Stake Tokens**: Verify that the correct amount of stBLS tokens is minted when BLS tokens are staked.
2. **Unstake Tokens**: Ensure the stBLS tokens are burned correctly and BLS tokens are returned to the user.
3. **Revert Conditions**: Check that staking with zero amount and unstaking more than the user's balance revert correctly.

### Results

- All test cases passed successfully, confirming the correctness of the contract functionalities.

---

## 6. Conclusion

### Summary of Findings

The **Blume Liquid Staking** smart contract has been successfully implemented, allowing users to stake their BLS tokens and receive stBLS tokens in return. The contract functions as intended, with robust error handling and secure token management.