

COMP4541

Decentralized Lottery App

Spring 2024-25

Name: PARK, Siyoon

SID: 20739700

Submission Date: May 20, 2025

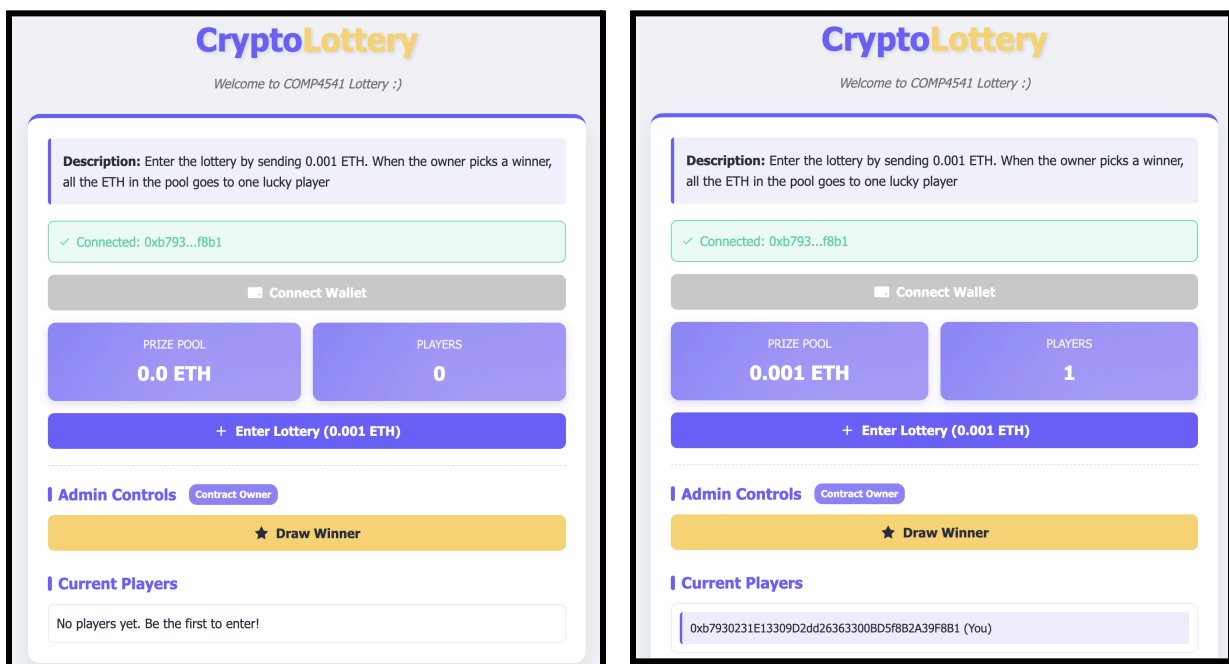
Contents

Overview.....	1
App functionalities.....	2
Additional Testing Process.....	4

Overview

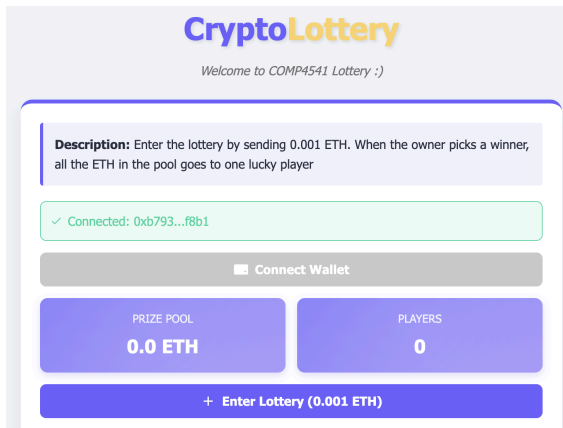
Website address: <https://winm00.github.io/lottery/>

This app is a 'lottery app', where users can enter by sending 0.001 ETH. The winner will be selected at random and be granted prize money.

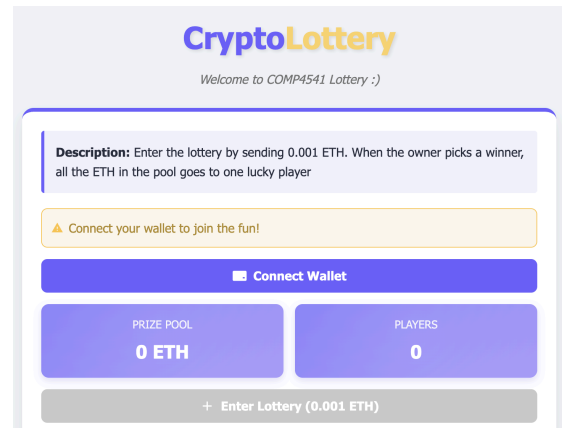


App functionalities

Wallet connection: Users can connect their Metamask wallet to the application. The application will send a request to Metamask. Metamask will ask the user for approval. If successful, a message will be displayed on the website confirming the connection. After the first time, users will no longer have to connect manually as the app will check automatically for pre-existing Metamask connection.

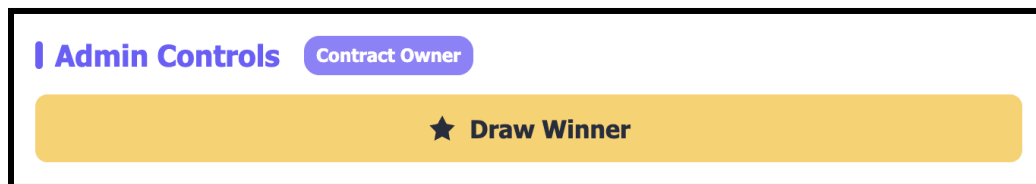


Successful connection

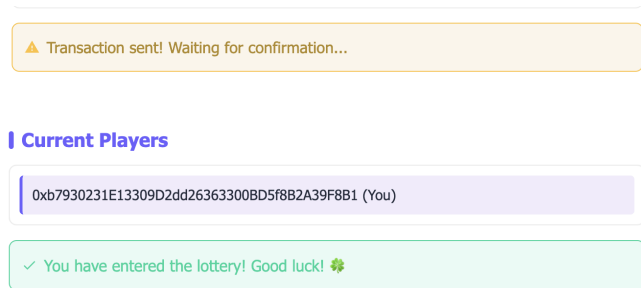
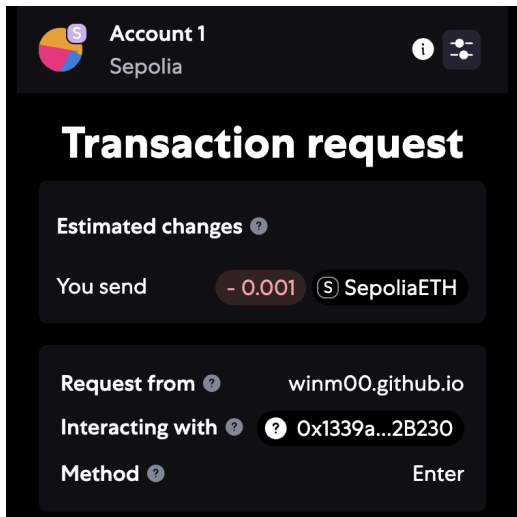


*No wallet detected
("Enter Lottery" button is disabled.)*

Owner verification: The application will call `contract.owner()` to retrieve the contract owner address and compare the return value with the user's address. If there is a match, that means the user is the owner of the contract, so it will display the "Draw Winner" option, which is reserved for the owner.



Entering the lottery: After the wallet has been connected, users can click on the 'enter lottery' button. The app will ensure that the user has no previous record of entering the lottery. It will call `contract.enter` and users will be able to initiate the transaction on metamask. A "Transaction sent" message will be displayed. It will display a success message once the transaction is confirmed.



Updating changes: After a new entry, the prize pool value will increase and the UI will update to show the new prize money and the number of players. For the user, they will no longer be able to enter the lottery. The “enter lottery” button will be disabled.

Before:



After:



Retrieving current players: The application will retrieve the list of current players and display their wallet addresses (as shown above)

Winner selection: For the owner who has been verified, they will be able to draw a random winner. A random number will be generated by taking the block timestamp, difficulty, and player address. The prize money will be transferred to the address of the winner.

| Current Players

No players yet. Be the first to enter!

✓ Winner has been picked! Check your wallet for winnings. 🎉

Possible concerns around security: so far, the app is taking into account factors such as block timestamp, difficulty, and player address. This can be a bit predictable to miners if this app was to be implemented in a larger setting. To address this, more sources of randomness could be added later when expanding the app.

Additional Testing Process

In addition, the platform provided by the course was used to test the app (<https://mycontract.fun/>). The following are the results.

Test Result:

Using contract tester version 0.8.0

(1/3) 🌀 Compiling contract: lottery_20739700_1747715222.sol

[✓ PASS] Compilation

(2/3) 📄 Generating test case for: lottery_20739700_1747715222.sol

[✓ PASS] Read contract

[i INFO] Contract file already exists in destination, skipping adjust.

[⚠ WARNING] No </think> tag found in input

[✓ PASS] Test case generation

[✓ PASS] Write test contract file

(3/3) 🌀 Running tests in lottery_20739700_1747715222.t.sol

(🌀 Attempt 1/3)

[DEBUG] STDOUT

Compiling 1 files with Solc 0.8.28

Solc 0.8.28 finished in 1.18s

Compiler run successful with warnings:

Warning (8417): Since the VM version paris, "difficulty" was replaced by "prevrandao", which now returns a random number based on the beacon chain.

--> src/lottery_20739700_1747715222.sol:47:68:

```
|  
47 |     return uint256(keccak256(abi.encodePacked(block.timestamp, block.difficulty, players)));  
|                                     ^^^^^^^^^^^^^^^^^^^
```

Warning (8417): Since the VM version paris, "difficulty" was replaced by "prevrandao", which now returns a random number based on the beacon chain.

--> test/lottery_20739700_1747715222.t.sol:151:69:

```
|  
151 |     bytes32 hash1 = keccak256(abi.encodePacked(block.timestamp, block.difficulty,  
lottery.getPlayers()));  
|                                     ^^^^^^^^^^^^^^^^^^^
```

Warning (8417): Since the VM version paris, "difficulty" was replaced by "prevrandao", which now returns a random number based on the beacon chain.

--> test/lottery_20739700_1747715222.t.sol:158:69:

```
|  
158 |     bytes32 hash2 = keccak256(abi.encodePacked(block.timestamp, block.difficulty,  
lottery.getPlayers()));  
|                                     ^^^^^^^^^^^^^^^^^^^
```

Warning (2018): Function state mutability can be restricted to view

--> test/lottery_20739700_1747715222.t.sol:131:5:

```
|  
131 |     function test_constructor_initialization() public {  
|     ^ (Relevant source part starts here and spans across multiple lines).
```

Ran 7 tests for test/lottery_20739700_1747715222.t.sol:LotteryTest

[PASS] test_constructor_initialization() (gas: 24599)

Logs:

Lottery deployed with studentId: 12345

Running test_constructor_initialization

Traces:

[24599] LotteryTest::test_constructor_initialization()

├─ [0] console::log("Running test_constructor_initialization") [staticcall]

└─ ← [Stop]

├─ [2751] Lottery::owner() [staticcall]

└─ ← [Return] LotteryTest: [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496]

├─ [0] VM::assertEq(LotteryTest: [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496],

LotteryTest: [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496], "Owner should be set to deployer address") [staticcall]

└─ ← [Return]

├─ [2642] Lottery::studentId() [staticcall]

└─ ← [Return] 12345 [1.234e4]

```
└─ [0] VM::assertEq(12345 [1.234e4], 12345 [1.234e4], "studentId should be correctly stored upon deployment") [staticcall]
  └─ ← [Return]
    └─ ← [Return]
```

[PASS] test_enterLottery_success() (gas: 76024)

Logs:

Lottery deployed with studentId: 12345

Running test_enterLottery_success

Traces:

```
[76024] LotteryTest::test_enterLottery_success()
└─ [0] console::log("Running test_enterLottery_success") [staticcall]
  └─ ← [Stop]
    └─ [47334] Lottery::enter{value: 10000000000000000}()
      └─ emit LotteryEntered(player: LotteryTest:
[0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496], amount: 10000000000000000 [1e15])
        └─ ← [Return]
          └─ [1933] Lottery::getPlayers() [staticcall]
            └─ ← [Return] [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496]
              └─ [0] VM::assertEq(1, 1, "Player count should be 1 after one successful entry") [staticcall]
                └─ ← [Return]
                  └─ [0] VM::assertEq(LotteryTest: [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496],
LotteryTest: [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496], "The entered player's address
should match the caller") [staticcall]
                    └─ ← [Return]
                      └─ ← [Return]
```

[PASS] test_enterLottery_wrongFee() (gas: 22426)

Logs:

Lottery deployed with studentId: 12345

Running test_enterLottery_wrongFee

Traces:

```
[22426] LotteryTest::test_enterLottery_wrongFee()
└─ [0] console::log("Running test_enterLottery_wrongFee") [staticcall]
  └─ ← [Stop]
    └─ [0] VM::expectRevert(custom error 0xf28dceb3: Entry fee is 0.001 ETH)
      └─ ← [Return]
        └─ [718] Lottery::enter{value: 20000000000000000}()
          └─ ← [Revert] revert: Entry fee is 0.001 ETH
            └─ ← [Return]
```

[PASS] test_pickWinner_noPlayers() (gas: 19904)

Logs:

Lottery deployed with studentId: 12345

Running test_pickWinner_noPlayers

Traces:

```
[19904] LotteryTest::test_pickWinner_noPlayers()
```

```

├── [0] console::log("Running test_pickWinner_noPlayers") [staticcall]
│   └── ← [Stop]
├── [0] VM::expectRevert(custom error 0xf28dceb3: No players in the lottery)
│   └── ← [Return]
├── [5239] Lottery::pickWinner()
│   └── ← [Revert] revert: No players in the lottery
└── ← [Return]

```

[PASS] test_pickWinner_onlyOwner() (gas: 78219)

Logs:

Lottery deployed with studentId: 12345

Running test_pickWinner_onlyOwner

Traces:

```

[78219] LotteryTest::test_pickWinner_onlyOwner()
├── [0] console::log("Running test_pickWinner_onlyOwner") [staticcall]
│   └── ← [Stop]
├── [0] VM::deal(0x00000000000000000000000000000000bEEF, 1000000000000000000
[1e18])
│   └── ← [Return]
├── [0] VM::prank(0x00000000000000000000000000000000bEEF)
│   └── ← [Return]
├── [47334] Lottery::enter{value: 10000000000000000}()
│   └── emit LotteryEntered(player: 0x00000000000000000000000000000000bEEF, amount:
10000000000000000 [1e15])
│       └── ← [Return]
├── [0] VM::prank(0x00000000000000000000000000000000bEEF)
│   └── ← [Return]
├── [0] VM::expectRevert(custom error 0xf28dceb3: %Only the owner can call this function)
│   └── ← [Return]
├── [2926] Lottery::pickWinner()
│   └── ← [Revert] revert: Only the owner can call this function
└── ← [Return]

```

[PASS] test_pickWinner_success() (gas: 149618)

Logs:

Lottery deployed with studentId: 12345

Running test_pickWinner_success

Number of players before picking winner: 3

Contract balance after picking winner: 0

Traces:

```

[192288] LotteryTest::test_pickWinner_success()
├── [0] console::log("Running test_pickWinner_success") [staticcall]
│   └── ← [Stop]
├── [0] VM::deal(ECRecover: [0x0000000000000000000000000000000000000000000000000000000000000001],
10000000000000000000000000 [1e18])
│   └── ← [Return]
├── [0] VM::deal(SHA-256: [0x0000000000000000000000000000000000000000000000000000000000000002],
10000000000000000000000000 [1e18])

```



```

└─ ← [Return]
└─ [0] VM::deal(RIPEMD-160: [0x0000000000000000000000000000000000000000000000000000000000000000],
1000000000000000000 [1e18])
└─ ← [Return]
└─ [0] VM::prank(ECRecover: [0x0000000000000000000000000000000000000000000000000000000000000001])
└─ ← [Return]
└─ [47334] Lottery::enter{value: 10000000000000000}()
└─ emit LotteryEntered(player: ECRecover:
[0x0000000000000000000000000000000000000000000000000000000000000001], amount: 1000000000000000 [1e15])
└─ ← [Return]
└─ [0] VM::prank(SHA-256: [0x0000000000000000000000000000000000000000000000000000000000000002])
└─ ← [Return]
└─ [25434] Lottery::enter{value: 10000000000000000}()
└─ emit LotteryEntered(player: SHA-256:
[0x0000000000000000000000000000000000000000000000000000000000000002], amount: 1000000000000000 [1e15])
└─ ← [Return]
└─ [0] VM::prank(RIPEMD-160: [0x0000000000000000000000000000000000000000000000000000000000000003])
└─ ← [Return]
└─ [25434] Lottery::enter{value: 10000000000000000}()
└─ emit LotteryEntered(player: RIPEMD-160:
[0x0000000000000000000000000000000000000000000000000000000000000003], amount: 1000000000000000 [1e15])
└─ ← [Return]
└─ [3441] Lottery::getPlayers() [staticcall]
└─ ← [Return] [0x0000000000000000000000000000000000000000000000000000000000000001,
0x0000000000000000000000000000000000000000000000000000000000000002,
0x0000000000000000000000000000000000000000000000000000000000000003]
└─ [0] console::log("Number of players before picking winner:", 3) [staticcall]
└─ ← [Stop]
└─ [0] VM::assertEq(3, 3, "There should be 3 players before picking a winner") [staticcall]
└─ ← [Return]
└─ [0] VM::assertEq(30000000000000000 [3e15], 30000000000000000 [3e15], "Contract balance
should be equal to total entry fees") [staticcall]
└─ ← [Return]
└─ [19224] Lottery::pickWinner()
└─ [60] PRECOMPILEs::sha256{value: 30000000000000000} (0x)
└─ ← [Return]
0xe3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
└─ emit WinnerPicked(winner: SHA-256:
[0x0000000000000000000000000000000000000000000000000000000000000002], prize: 3000000000000000 [3e15])
└─ ← [Return]
└─ [1179] Lottery::getPlayers() [staticcall]
└─ ← [Return] []
└─ [0] VM::assertEq(0, 0, "Players array should be empty after picking a winner") [staticcall]
└─ ← [Return]
└─ [0] VM::assertTrue(true, "One of the players should have received the prize amount") [staticcall]
└─ ← [Return]
└─ [0] console::log("Contract balance after picking winner:", 0) [staticcall]
└─ ← [Stop]
└─ [0] VM::assertEq(0, 0, "Contract balance should be 0 after prize payout") [staticcall]
└─ ← [Return]

```

└─ ← [Return]

[PASS] test_randomness_influence() (gas: 80792)

Logs:

Lottery deployed with studentId: 12345

Running test_randomness_influence

Traces:

[80792] LotteryTest::test_randomness_influence()

└─ [0] console::log("Running test_randomness_influence") [staticcall]

└─ ← [Stop]

└─ [47334] Lottery::enter{value: 1000000000000000}()

└─ emit LotteryEntered(player: LotteryTest:
[0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496], amount: 1000000000000000 [1e15])

└─ ← [Return]

└─ [1933] Lottery::getPlayers() [staticcall]

└─ ← [Return] [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496]

└─ [0] VM::warp(101)

└─ ← [Return]

└─ [1933] Lottery::getPlayers() [staticcall]

└─ ← [Return] [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496]

└─ [0] VM::assertTrue(true, "Random values should differ with changed block parameters")
[staticcall]


└─ ← [Return]

└─ ← [Return]

Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 1.94ms (1.19ms CPU time)

Ran 1 test suite in 16.48ms (1.94ms CPU time): 7 tests passed, 0 failed, 0 skipped (7 total tests)

 Start Cleaning

 [PASS] Moved test file to 'finished'

There are test cases that are failing at times. However, this seems to be due to factors irrelevant to the functionality app - such as expected event name mismatch.