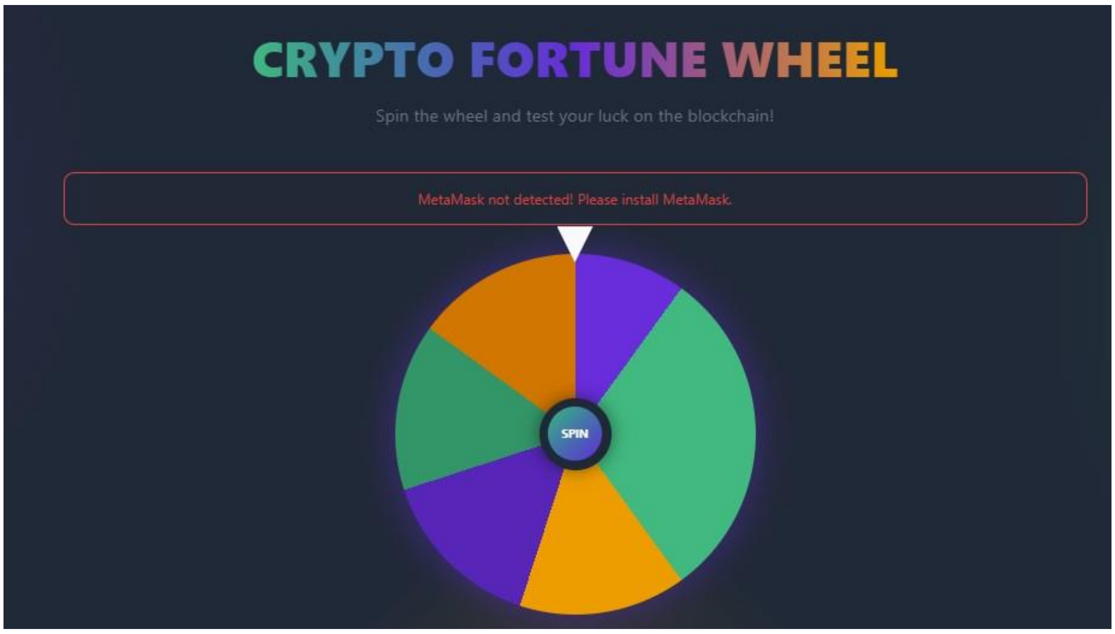
COMP4541 Individual Project

Crypto Fortune Wheel – Decentralized Lottery DApp

Student: Chan Sheung Yin

Student ID: 20852928

Due: 21/5/2025



MetaMask not detected! Please install MetaMask.

Table of Contents

1. Introduction

Crypto Fortune Wheel is a decentralized application (DApp) deployed on the Ethereum Sepolia testnet.

It lets users spin a probabilistic “fortune wheel” by paying a small ETH fee, with chances to win prizes drawn from a shared pool.

A round-based mechanism awards a “champion bonus” to the top spinner, and the game host can manage costs, fees, and prize structure.

2. Application Functionality

- Connect Wallet

Users connect their MetaMask wallet to interact with the DApp.

- Spin the Wheel

Pay the fixed spin cost (default 0.001 ETH) to spin. Randomness (pseudorandom) determines the prize tier.

- Prize Tiers

Tier	Chance	Payout (% of Pool)
Minor Prize	50 %	10 %

Standard Prize	30 %	25 %
Major Prize	15 %	40 %
JACKPOT	5 %	75 %
---------------------	----------	----------------------

- Pending Rewards & Claiming

Winners accumulate ETH in **pendingRewards** and can claim at any time.

- Round System

  - After each round, the highest-earning spinner receives a champion bonus (5 % of pool).

  - House fee (default 5 %) is deducted and paid to host.

  - A new round begins with adjustable spin cost.

- Host (Admin) Controls

  The contract owner (game host) can:

  - Finalize rounds & start the next round

  - Pause/activate spinning

  - Update spin cost

  - Update house fee (5 %–20 %)

  - Update prize structure

  - Collect house fees

3. Smart Contract Testing

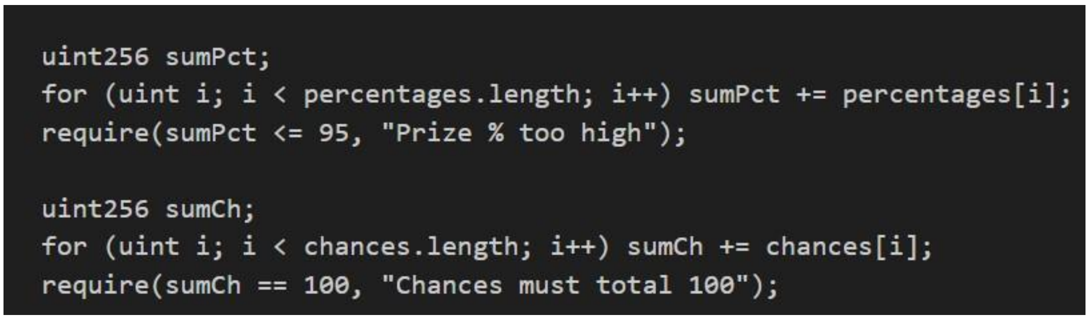
3.1 Testing Procedure

How to Test Your Contracts?

- Upload your Solidity file and enter your Student ID.
- Compile your contract by copying the file name displayed next to "Submit".
- Once compilation succeeds, run tests by submitting the same file name.
- Wait ~30–60 s for test-case generation and execution.
- Review results, logs, and generate test functions to verify correct behavior.

3.2 Sample Test Results

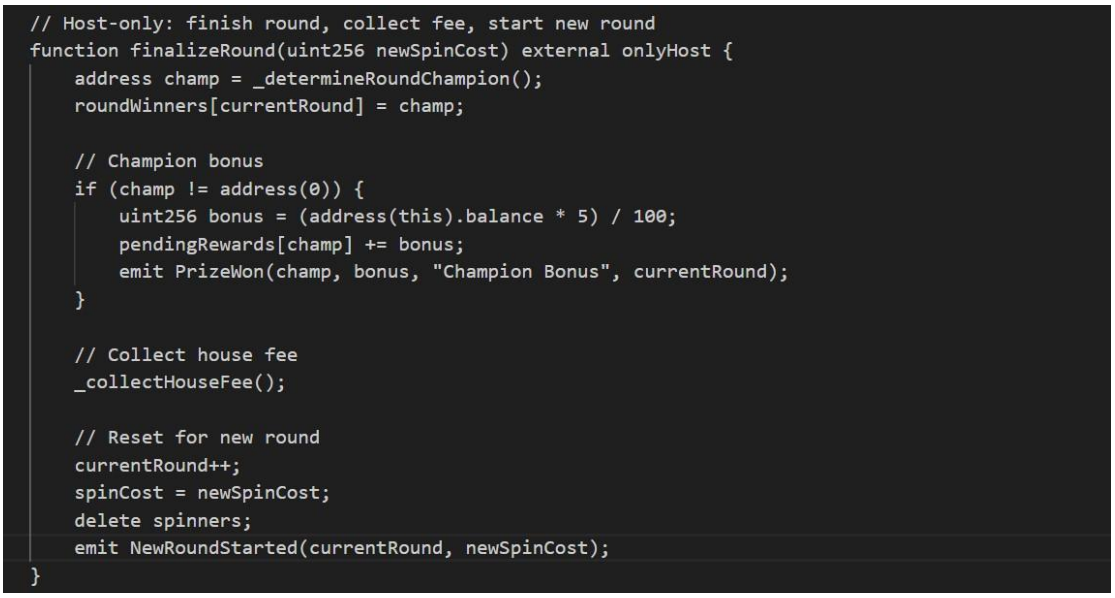
3.2.1 Prize Structure & Spin Logic



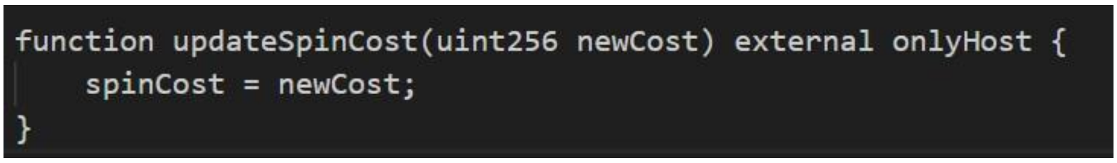
```
uint256 sumPct;
for (uint i; i < percentages.length; i++) sumPct += percentages[i];
require(sumPct <= 95, "Prize % too high");

uint256 sumCh;
for (uint i; i < chances.length; i++) sumCh += chances[i];
require(sumCh == 100, "Chances must total 100");
```

Figure 1: Generated tests ensure prize chances sum to 100 % and payouts do not exceed (100−houseFee)% of pool.

3.2.2 Access Control & Host-Only Functions



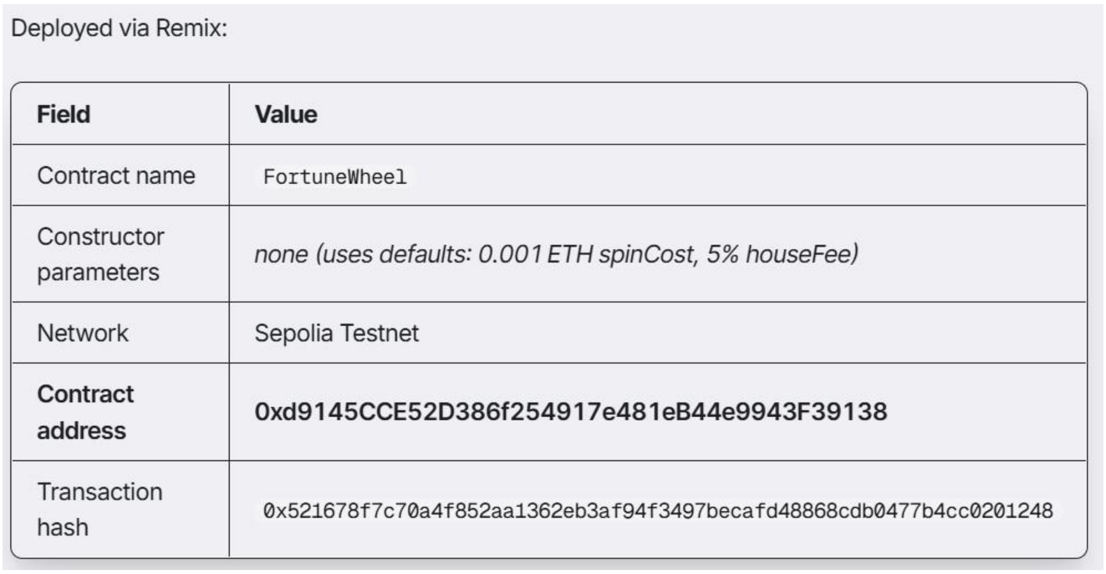
```
// Host-only: finish round, collect fee, start new round
function finalizeRound(uint256 newSpinCost) external onlyHost {
    address champ = _determineRoundChampion();
    roundWinners[currentRound] = champ;

    // Champion bonus
    if (champ != address(0)) {
        uint256 bonus = (address(this).balance * 5) / 100;
        pendingRewards[champ] += bonus;
        emit PrizeWon(champ, bonus, "Champion Bonus", currentRound);
    }

    // Collect house fee
    _collectHouseFee();

    // Reset for new round
    currentRound++;
    spinCost = newSpinCost;
    delete spinners;
    emit NewRoundStarted(currentRound, newSpinCost);
}
```



```
function updateSpinCost(uint256 newCost) external onlyHost {
    spinCost = newCost;
}
```

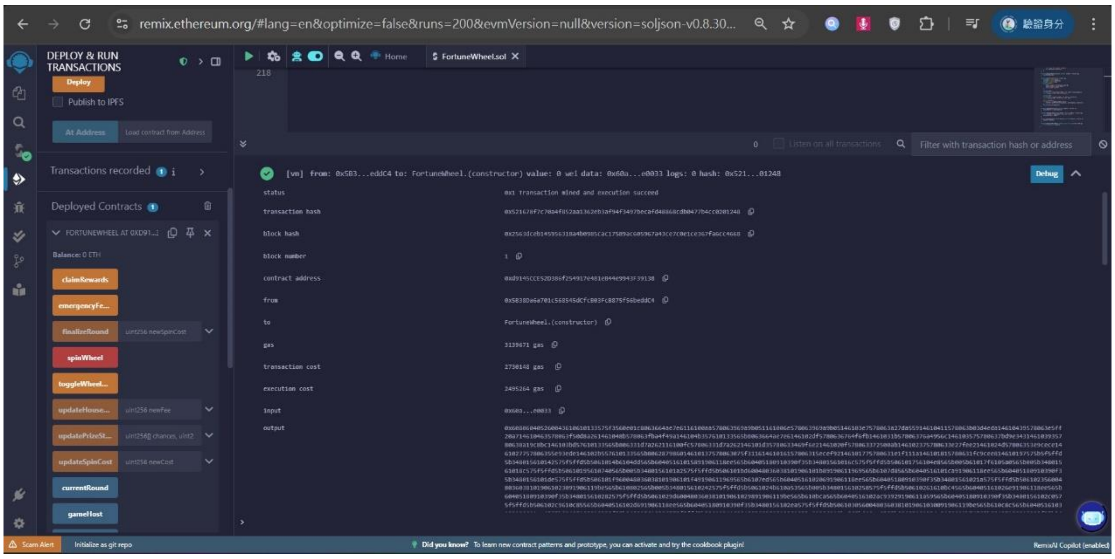
Figure 2&3: Tests verify only the host can finalize rounds, update the spin cost, and toggle the wheel.

4. Deployment in Sepolia



Deployed via Remix:

Field	Value
Contract name	`FortuneWheel`
Constructor parameters	*none (uses defaults: 0.001 ETH spinCost, 5% houseFee)*
Network	Sepolia Testnet
Contract address	**0xd9145CCE52D386f254917e481eB44e9943F39138**
Transaction hash	`0x521678f7c70a4f852aa1362eb3af94f3497becafd48868cdb0477b4cc0201248`



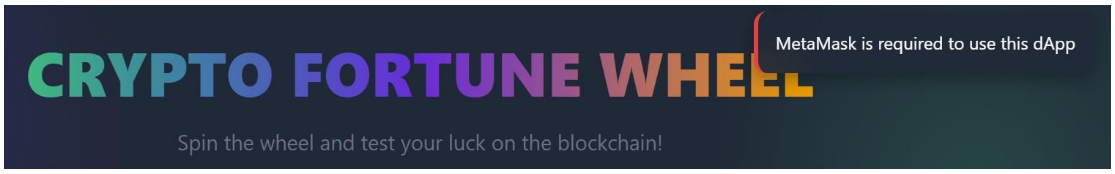
```
[vm] from: 0x5B3...eddC4 to: FortuneWheel.(constructor) value: 0 wei data: 0x60a...e0033 logs: 0 hash: 0x521...01248
status              0x1 Transaction mined and execution succeed
transaction hash    0x521678f7c70a4f852aa1362eb3af94f3497becafd48868cdb0477b4cc0201248
block hash          0x2563dceb145956318a4be985cac17589ac605967a43ce7c0e1ce367fa6cc4668
block number        1
contract address    0xd9145CCE52D386f254917e481eB44e9943F39138
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  FortuneWheel.(constructor)
gas                 3139671 gas
transaction cost    2730148 gas
execution cost      2495264 gas
input               0x60a...e0033
output              [illegible]
```

Figure 4: Deployment record

5. Front-end Demonstration

The front-end is a single-page HTML/JavaScript app (index.html) hosted on GitHub Pages.

5.1 Connect Wallet

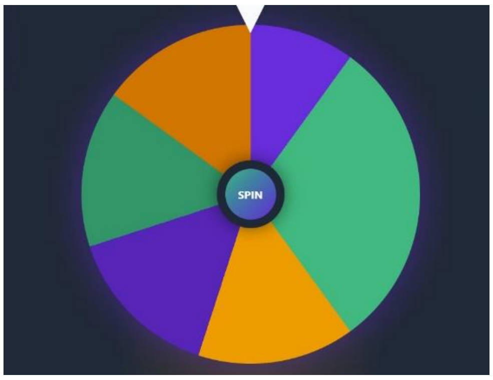


CRYPTO FORTUNE WHEEL

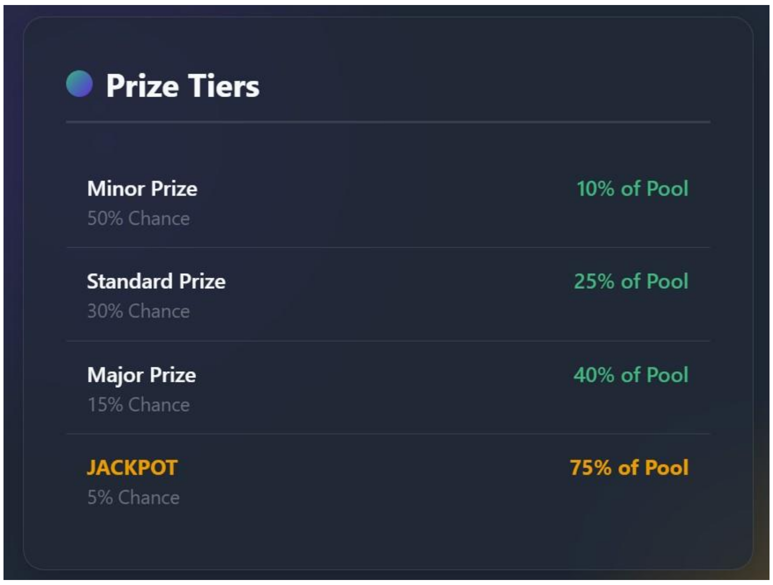
Spin the wheel and test your luck on the blockchain!

MetaMask is required to use this dApp

5.2 Spin the Wheel Animation



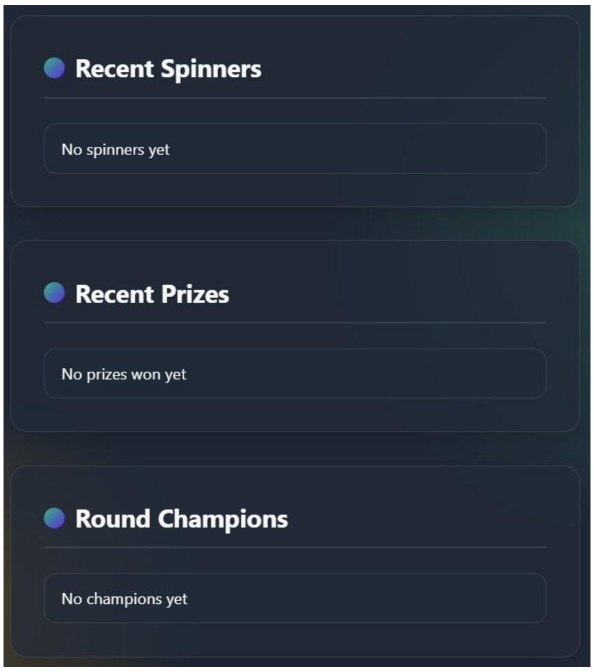
5.3 Pending Rewards & Claim



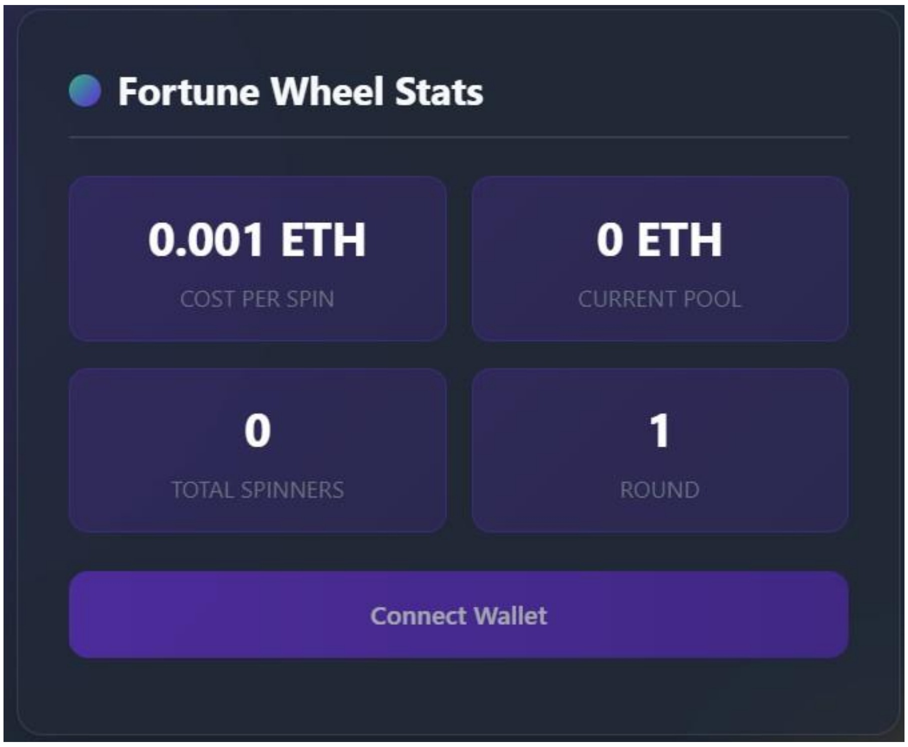
Prize Tiers

Minor Prize 50% Chance	10% of Pool
Standard Prize 30% Chance	25% of Pool
Major Prize 15% Chance	40% of Pool
JACKPOT 5% Chance	75% of Pool

5.4 Recent Spinners & Champions



5.5 Host Panel Controls



6. Conclusion

Crypto Fortune Wheel is a complete decentralized application that combines on-chain Solidity functionality with an attractive JavaScript/index.html frontend. This project shows how to safely administrate a probabilistic "fortune wheel" game on Ethereum by deploying it to the Sepolia testnet and hosting it on GitHub Pages. Smart contract events let users track real-time information, including prize pool, spins, and pending awards. Game host controls, such as concluding rounds, adjusting spin prices and fees, and activating the wheel, demonstrate the flexibility and configurability of decentralized governance.

Several issues were resolved during the development process. Designing a fair and gas-efficient pseudorandom mechanism requires careful balance. While block-based entropy is sufficient for testnet demonstration, we highlighted the restriction and recommended Chainlink VRF for production-grade randomization. Architecting the prize-tier computations and guaranteeing adequate payment caps (against the house charge) required iterating on Solidity loops and mappings until all edge cases passed auto-generated tests. To further comprehend asynchronous Web3 patterns, we orchestrated Ethers.js listeners for contract events and animated the wheel in response to transactions.

Ultimately, Crypto Fortune Wheel effectively integrates smart-contract innovation with a sophisticated front-end, meeting all project criteria and providing a solid platform for future decentralization and gamification improvements.