LOTTERY DAPP

Timur Taepov
Filippo Casari
De Grandi Alessandro

AGENDA

Contract

Front End

Conclusion

The goal of the project

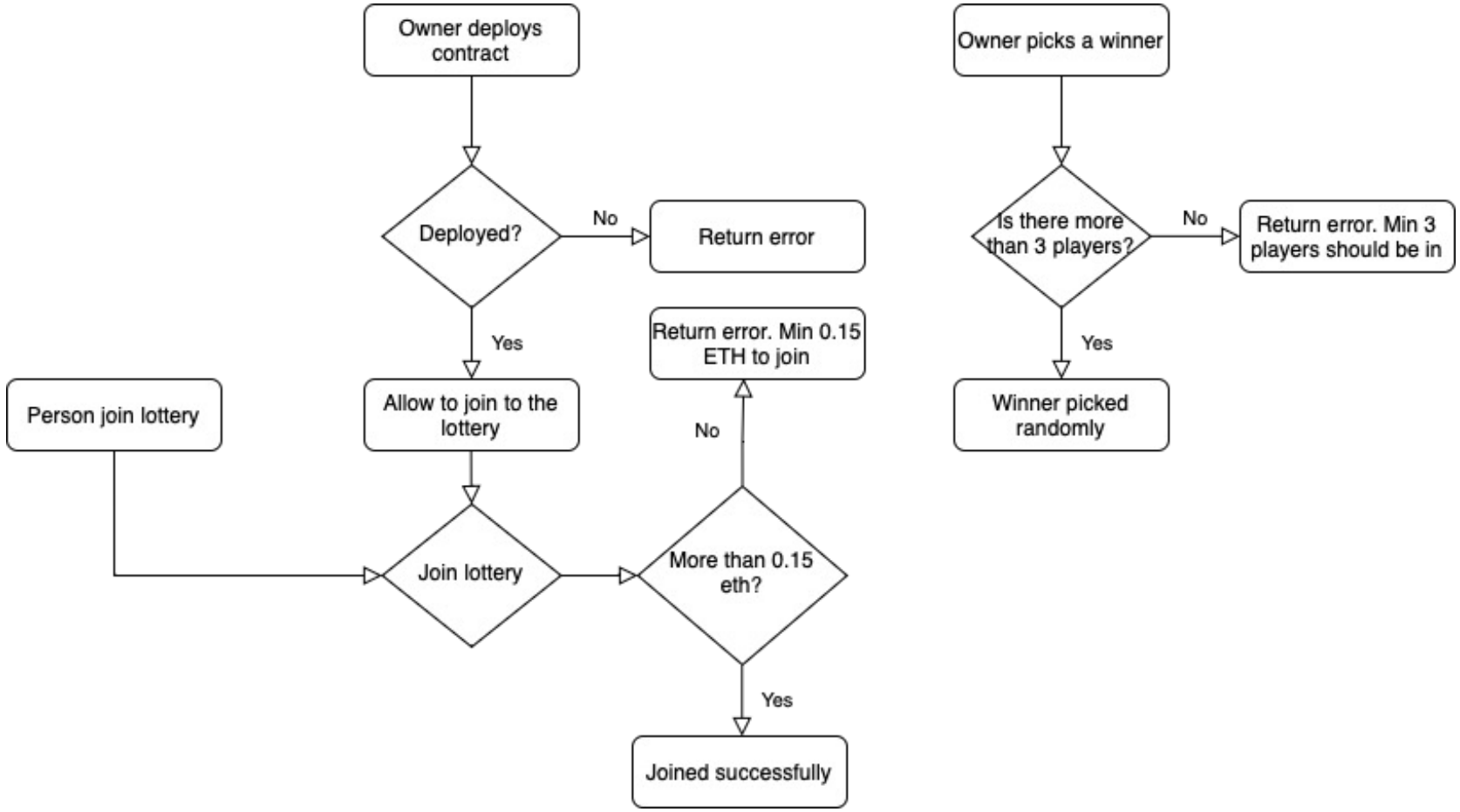
The idea is to implement a safe lottery decentralized application which works with ethereum tokens

SMART CONTRACT

Specifications of the smart contract

- Owner of the contract can select a winner
- Participants can join to the lottery with at least 0.15 ETH
- Maximum amount of lotteries which can be played is set to 100
- Minimal amount of participants in order to select winner is 3
- There is a possibility to return participant’s wallets
- Winner’s wallets are saved to the winner’s array
- Winner is selected by Keccak-256 hash algorithm which packs boss public address and timestamp and then return a random number

Smart contract flow chart



FRONT-END

- HTML HOME PAGE

Flask Web Server + Web3 + Brownie

- Flask is a micro web framework written in Python
- Web3 is a Python library for interacting with Ethereum
- Brownie is a Python-based development and testing framework for smart contracts targeting the Ethereum virtual machine.

```python
@app.route('/join_lottery', methods=['POST'])
def join_lottery():
    """an account can join the lottery

    Returns:
        str: see the html
    """
    adress = request.form['adress']
    sender = Web3.toChecksumAddress(adress)
    value = web3.toWei(0.015, "ether")
    txn = {
        "from": sender,
        "value": value,
    }
    try:
        txn_hash = contract_instance.functions.joinLottery().transact(txn
    )
        tx_receipt = web3.eth.wait_for_transaction_receipt(txn_hash)
        result = contract_instance.functions.participantsInfo().call()
        tx_dict = dict(tx_receipt)
        return render_template("joinLottery.html", account=tx_dict["from"
    ])
    except Exception as error:
        return str(error)
```

Instance of the Solidity Contract

.

```
contract_address = pathlib.Path("../utils/address.txt").read_text()
with open("../utils/abi.json") as file_a:
    abi = file_a.read()
    abi = json.loads(abi)

bytecode = web3.eth.get_code(contract_address)

contract_instance = web3.eth.contract(address=contract_address, abi=abi)
```

DEMO DAPP

Application Usage