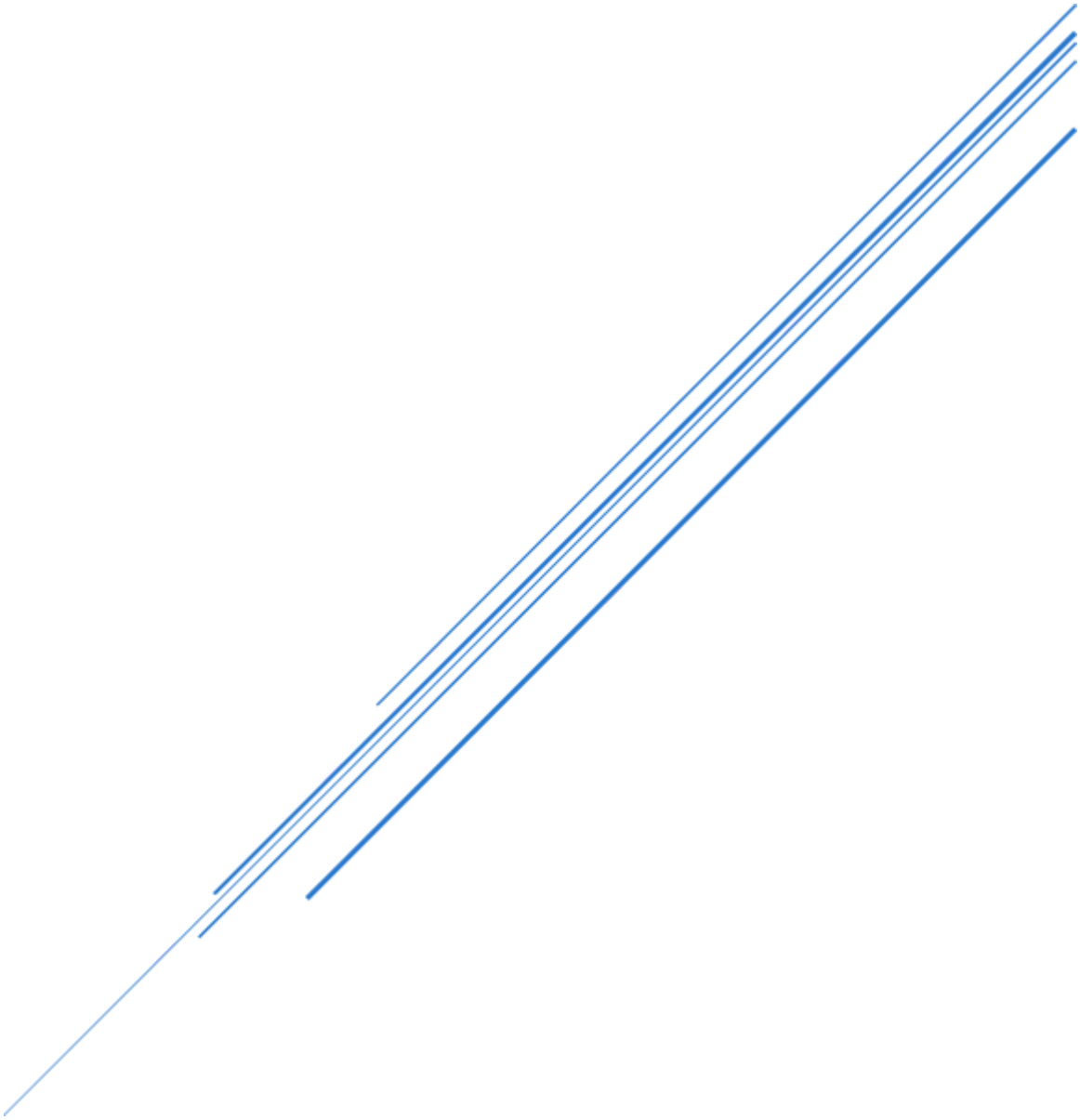
DECENTRALIZED RAFFLE SYSTEM

CS-411 Blockchain



Hamza Hasan Ellahie 2021197
Muhammad Zulfiqar Ali 2021493

Quiz 3: Deployment Instructions

---

1. Functional Prototype

1.1 Project Overview

Our project is a Lottery DApp that uses a smart contract (`Lottery.sol`) on an Ethereum-like blockchain. Participants can:

- **Enter** the lottery by sending 0.1 ETH.
- **Pick a winner** (restricted to the contract's manager/owner).
- **Claim** the prize if they are the randomly selected winner.

The code structure includes:

1. **Hardhat** project for compiling and deploying the smart contract.
2. **React** front-end for user interaction via MetaMask.
3. **Ethers.js v6** for contract interaction.

1.2 User Flows

1. **Enter the Lottery**
    - User connects MetaMask.
    - Clicks **Enter Lottery** on the *Home* page.
    - Sends 0.1 ETH to the contract.
2. **Pick Winner**
    - Only the contract owner can pick a winner.
    - The winner is chosen pseudorandomly using `block.prevrandao`, `block.timestamp`, and the number of players.
3. **Claim Prize**
    - If the lottery is complete and the connected wallet is the winner's address, the user can claim the balance of the contract.

1.3 Screens

- **Home (Lottery Page)**
    - Displays a button to **Enter** (if the lottery is not complete).
    - Displays a button to **Claim Prize** if the user is the winner and the lottery is complete.
    - Otherwise, displays "You are not the winner."
- **PickWinner (Result Page)**
    - If the user is the manager, shows a button to **Pick Winner**.

- If the lottery is complete, shows the winner's address.
- Otherwise, displays "You are not the owner" if a non-owner attempts access.

---

2. Deployment and Setup Instructions

Below is a step-by-step guide for setting up and deploying this DApp.

2.1 Prerequisites

- **Node.js v16+**
- **Hardhat** (already included in `devDependencies`)
- **MetaMask** extension in your browser (for testing locally or on a testnet)
- **Git** (to clone the repository)

2.2 Cloning the Repository

1. **Fork** the repository on GitHub (for your own workspace).
2. **Clone** the fork locally:

```
git clone https://github.com/<your-username>/lottery-dapp.git
cd lottery-dapp
```

2.3 Installing Dependencies

From the project root, install all required packages:

```
npm install
```

This installs:

- **React** + dependencies (via `react-scripts`).
- **Ethers v6**.
- **Hardhat** and Hardhat Toolbox.

2.4 Configuring Hardhat

In your `hardhat.config.js`, you will see something like:

```
require("@nomicfoundation/hardhat-toolbox");

module.exports = {
  solidity: "0.8.0",
  networks: {
    hardhat: {
      chainId: 31337,
      accounts: {
```

```
        // Example configuration
        mnemonic: "test test test test test test test test test test test
junk",
        // count: 20 // optional
      }
    }
    // add config for testnet if needed
  }
};
```

You can modify the mnemonic or other network parameters as needed.

2.5 Local Deployment

1. **Run a local Hardhat node** (in a separate terminal):

   ```
   npx hardhat node
   ```

2. **Deploy the contract** to your local Hardhat network:

   ```
   npx hardhat run scripts/deploy.js --network localhost
   ```

3. **Record the contract address** printed to the console (e.g., `Lottery deployed to: 0x...`).
   - Update the address in `src/constants.js` (`contractAddress`).

2.6 Front-End Configuration

1. **Update** `constants.js` with the deployed contract address:

   ```
   const contractAddress = "0x5FbDB2315678afecb367f032d93F642f64180aa3";
   // ...
   ```

2. **Start the React app** in development mode:

   ```
   npm start
   ```

3. **Open** http://localhost:3000 in your browser.

2.7 MetaMask Setup

1. In MetaMask, **import** the private keys of wallets displayed by your `showKeys.js` (only in a development/test environment).

2. **Switch** MetaMask to the Hardhat network (localhost:8545) so that it connects to your local node and loads up with Test Ethereum.

3.8 Test the DApp

1. **Go to the Home page**:
    - Click **Enter Lottery**. Check that 0.1 ETH is requested.
    - Confirm the transaction in MetaMask.
2. **Manager**:
    - In the Hardhat console or from an address that matches `manager`, pick a winner on the `/PickWinner` page.
    - Ensure only the manager can see the "Pick Winner" button.
3. **Winner**:
    - The winning address should see "Claim Prize" on the Home page.
    - The winning address is displayed on the `/PickWinner` page for all users.

Final Thoughts

With these steps, we have presented a **fully functional prototype** of our Lottery DApp. We have also hosted our annotated code on **GitHub** (including a thorough `README.md`), and the above instructions fulfill the **deployment and setup** documentation requirement.