**/\*Report**

**Members:-**

**Deepak Barwal :- 101531741**

**Dapp :- Gold Digger**

**About the project :-**

The Lottery Draw App is a decentralized application built on the Solana blockchain. It lets users to buy lottery tickets and check results and interact with the lottery system easily. The dapp is designed to be more secure and transparent which makes the lottery process straightforward and fair for everyuser because of its decentralized property.

**Tech Stack:**

**Frontside:**

* **React.js**
* **TypeScript**
* **TailwindCSS**

**Backend:**

* **Solidity**
* **Thirdweb**

**Blockchain:**

* **Solana**

**Architecture:**

**A diagram of a company

Description automatically generated**

**Get Started:**

**Prerequisites:**

* Create a MetaMask account: Install the MetaMask browser extension and create an account to interact with the blockchain.
* Sign up for a Thirdweb account: Register on the Thirdweb platform to manage and deploy your smart contracts.
* Install Node.js: Ensure Node.js is installed on computer for running JavaScript code and managing packages.

**Environmental variables:**

To run this project, you will need to add the following environment variable to your .env file

- NEXT\_PUBLIC\_LOTTERY\_CONTRACT\_ADDRESS

**Installations:**

1. Node.js and npm

Node.js: Make sure Node.js installed, which includes npm (Node Package Manager).

**2. Solana CLI**

**Purpose:** The Solana Command Line Interface (CLI) is needed to interact with the Solana blockchain.

Installation:

* + Open Command Prompt and Run the command:

[ sh -c "$(curl -sSfL <https://release.solana.com/v1.11.14/install>)" ]

**3. Solana Web3.js (for interacting with the Solana blockchain in the frontend)**

* **Purpose:** Solana Web3.js is a library for connecting DApp frontend to the Solana blockchain.
* **Installation:**
  + Add it to your project by running:

“npm install @solana/web3.js”

**4. TypeScript**

* **Purpose:** TypeScript is used for writing error-free code.
* **Installation:**
  + Add it to project by running:

“npm install typescript --save-dev”

* + Initialize a TypeScript configuration file:

“npx tsc –init”

**5. TailwindCSS**

* **Purpose:** TailwindCSS is used for styling the DApp.
* **Installation:**
  + Follow these commands to install and configure TailwindCSS:

“npm install -D tailwindcss postcss autoprefixer”

“npx tailwindcss init -p”

**6. MetaMask**

* **Purpose:** MetaMask is a browser extension wallet used for interacting with the blockchain.
* **Installation:**
  + Install from the MetaMask website .

**Analysis:**

**1. Integration with Solana Blockchain**

* **Easy:**
  + **Using Solana Tools:** Tools and libraries from Solana make it simpler to connect your app to the blockchain.
  + **Testing on Devnet:** You can test everything on Solana’s test network without risking real money.
* **Hard:**
  + **Debugging Issues:** Fixing problems in your smart contracts was tricky.
  + **Handling Transactions:** Managing transaction fees and wallet interactions was complicated.

**2. Smart Contract Development**

* **Easy:**
  + **Using Frameworks:** Frameworks like Anchor help make smart contract development easier by providing useful tools.
  + **Following Patterns:** Using proven methods for lottery contracts can simplify the process.
* **Hard:**
  + **Contract Complexity:** Writing secure and bug-free smart contracts involves detailed work.
  + **Testing and Deployment:** Testing on test networks and deploying to the mainnet was challenging.

**3. React Frontend Development**

* **Easy:**
  + **React Libraries:** React and related libraries help build the user interface and connect to the blockchain.
  + **UI Components:** Creating UI elements with React is straightforward with the help of component libraries.
* **Hard:**
  + **Handling Transactions:** Managing blockchain transactions smoothly on the frontend was complex.
  + **State Management:** Keeping track of blockchain interactions and errors requires careful planning.

**4. Security and Testing**

* **Easy:**
  + **Automated Tools:** Tools helped automatically find common issues early on.
  + **Best Practices:** Following standard security and testing practices simplifies the process.
* **Hard:**
  + **Ensuring Security:** Making sure your smart contracts and frontend are secure requires thorough testing and checks.
  + **End-to-End Testing:** Full testing of the app, including how all parts work together, and was complex and time-consuming.

**What Worked Well:**

* Solana Blockchain: Choosing the Solana blockchain was a good move because it’s fast and has low transaction fees, making your lottery DApp efficient and affordable for users.
* Smart Contracts with Truffle: Using Truffle to develop your smart contracts gave you a strong, reliable framework for building, testing, and deploying your lottery features.
* Fair Lottery System: The decentralized lottery system you created, where winners are chosen by smart contracts, ensured that the process was fair and transparent, which is the most important part for user trust.
* Easy-to-Use Interface: Making the user interface simple and user-friendly helped make your DApp accessible to a wider audience, even those who aren’t familiar with web applications.
* Security Focus: We made sure the smart contracts were safe and had no flaws, which is really important for keeping the DApp and users secure.

User Interface:

A screenshot of a game

Description automatically generated