PROJECT REPORT

TITLE:

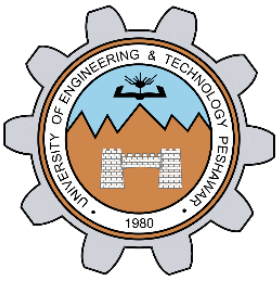
Building DAPP-on Tron Blockchain Using React & Solidity

BY:

SYED HASNAIN SHAH
23PWCSE2330
3rd SEMESTER

SUBMITTED TO:

Engr. Sumayyea Salahuddin



FALL 2024
CSE208L Object Oriented Programming Lab

DEPARTMENT OF COMPUTER SYSTEM ENGINEERING
UNIVERSITY OF ENGINEERING & TECHNOLOGY PESHAWAR

Building DAPP-on Tron Blockchain Using React & Solidity

Steps For Building DAPP:

- Install nodejs, React & Set it for development.
- Start Creating Project (game) in react and finalize it.
- Create a Tron-Link Wallet Account First get some Shasta Test-network TRX.
- Now go to www.tronide.io & link it with Tron-Link Wallet then set it up for Development then start writing the smart contract according to your Game (app).
- Once It is completed Then Select Environment (Injected Tron-Web) & deploy Smart Contract.
- You Will Get **Contract Address & ABI KEY** Which you will save it with your self it will be needed.
- Now go to project folder type cmd "**npm install tronweb**"
- Now Go to your Project Folder, Create a files names **contract.js & tronWeb.js**, which will interact with your Smart contract deployed on Tron Blockchain & The Contract address & ABI key will be needed there.
- Now When everything is ready go to **app.netlify.com** Create account there & for that it would need 'Dist' folder from our Project, for that type cmd **'npm run build'** it will create Dist Folder than upload it to website and it will create a live website for the app.
- Now go to Telegram.com and Create a Bot there and set it as a web-app bot it will ask for url, you will copy the url from the live website and paste it there, and do some setting after that your bot is ready.

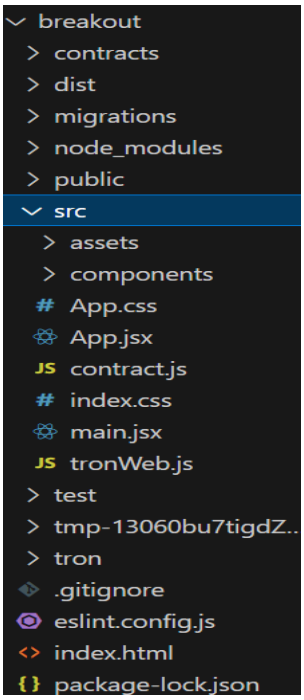
IMPLEMENTATION, ERRORS & PROBLEMS

- Did a Guided Project on Coursera **" Write & Test Smart Contracts "**
  https://coursera.org/verify/CCSOY8OAPCJY
  During This Project I learned how to setup, create & deploy Smart Contract (BASICs)
- Research about how to create DAPP on Tron Blockchain checked different websites blogs, videos, github projects continuously for a week, got different tutorial's but it was Of no use as they would create it differently, but after that I took a lot of help from CHATGPT as it was help full more than any other things it guides me step by step what to do which cmd's etc.
- I started Creating App using React Old version as it was my first time to work in react, I got continuously errors for 4 days I was trying to fix just errors & errors after that I contacted my friend who had experience in React he told me to do it in React Vite, so after that I start it in react vite mostly errors were not there, but there were errors which I took help from google & chatgpt to solve them.

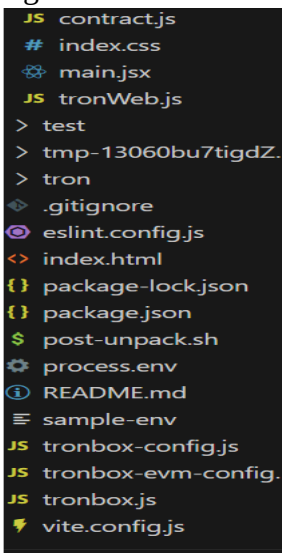
- After creating project(app) started working on Smart contract which I deployed, now here is the big challenge for me to connect App and Smart contract to make it DAPP, here I got puzzled & started searching youtube, google & chatgpt , I was trying through every possibility I could see and it finally interacted but I can't host the interacted DAPP on a Live because it was not compatible with tronweb besides that I had hosted that app when it was not interacting with smart contract it went smoothly.

EVERY STEP IN DETAIL:

- By Using **npx create-react-app breakout** cmd here we get initials Files.



- By Using **npm install tronweb** we get all Tron files.



- Now Creating the Game (**BREAKOUT**) using React-Vite, Here is the Code & Working of **Ball Component**.

```jsx
import React from 'react';
//Ball is a functional React component.,
const Ball = ({ position }) => {
  // Check ball position
  console.log('Rendering Ball at:', position);
  // /Prints the position to the browser's console whenever the component is rendered or re-rendered
  return (
    <div
      style={{
        width: '20px',
        height: '20px',
        backgroundColor: 'red',
        borderRadius: '50%',
        //Ensures the ball can be placed anywhere on the page based on the top and left values.
        position: 'absolute',
        //These dynamically set the ball's vertical and horizontal position based on the position prop.
        top: `${position.y}px`,
        left: `${position.x}px`,
      }}
    />
  );
};

export default Ball;
```

- After that we Created **Bricks Component**.

```jsx
import React from 'react';
const Bricks = ({ bricks }) => {
  // Check brick state
  console.log('Rendering Bricks:', bricks);
  return (
    <>
      {bricks.map((brick, index) =>
        !brick.destroyed ? (
          <div
            key={index}
            style={{
              width: '50px',
              height: '20px',
              backgroundColor: 'blue',
              position: 'absolute',
              top: `${brick.y}px`,
              left: `${brick.x}px`,
            }}
          />
        ) : null
      )}
    </>
  );
};

export default Bricks;
```

- Now Creating the **Paddle Component.**

```jsx
import React, { useEffect } from 'react';
const Paddle = ({ position, movePaddle }) => {
  console.log('Rendering Paddle at:', position); // Check paddle position

  useEffect(() => {
    const handleKeyDown = (e) => {
      //handleKeyDown: A function that listens for the keydown event and calls
      // movePaddle with // 'left' or 'right' depending on the key pressed.
      if (e.key === 'ArrowLeft') movePaddle('left');
      if (e.key === 'ArrowRight') movePaddle('right');
    };
    //window.addEventListener: Adds the keydown event listener when the component mounts.
    window.addEventListener('keydown', handleKeyDown);
    //window.removeEventListener: Removes the keydown event listener
    //  when the component unmounts to prevent memory leaks.
    return () => window.removeEventListener('keydown', handleKeyDown);
  }, [movePaddle]);
  return (
    <div
      style={{
        width: '200px', // Updated paddle width
        height: '10px',
        backgroundColor: 'black',
        position: 'absolute',
        bottom: '0',
        left: `${position}px`,
      }}
    />
  );
};

export default Paddle;
```

- Now Creating the **GameBoard component** in which we will import the all the other components & define game logics, functions of importing contract which will be.

```jsx
import { useEffect, useState, useRef } from 'react';
import Paddle from './Paddle';
import Ball from './Ball';
import Bricks from './Bricks';
import { getContract } from '../contract'; // Ensure this function returns the loaded contract
const GameBoard = ({ telegramUsername }) => {
  const boardRef = useRef(null);
  const [touchStartX, setTouchStartX] = useState(0); // Track the initial touch position
  const [touchMoveX, setTouchMoveX] = useState(0);   // Track the touch movement position

  // Function to generate bricks for a level
  const createBricks = (level) => {
    const rows = 5 + level; // Add more rows for each level
    const cols = 10;
    const brickWidth = 50;
    const brickHeight = 20;

    const bricksArray = [];
    for (let row = 0; row < rows; row++) {
      for (let col = 0; col < cols; col++) {
        bricksArray.push({
          x: col * brickWidth + 10,
          y: row * brickHeight + 40, // Add space to position bricks below the score/level
          destroyed: false,
        });
      }
    }
    return bricksArray;
  };
```

Figure 1 In This Code, I Imported different Components, functions from React Library and Contract from our Contract.js, & Logic of How rows & columns will be created In game.

```
// State variables
const [ballPosition, setBallPosition] = useState({ x: 300, y: 400 });
const [ballVelocity, setBallVelocity] = useState({ dx: 5, dy: -5 });
const [paddlePosition, setPaddlePosition] = useState(250);
const [bricks, setBricks] = useState(createBricks(1)); // Start with level 1
const [score, setScore] = useState(0);
const [level, setLevel] = useState(1);
const [userId, setUserId] = useState(null);
const [tokenPoints, setTokenPoints] = useState(0); // Track token points

// Initialize the user
useEffect(() => {
  const initializeUser = async () => {
    const uniqueId = `user_${telegramUsername}`;
    setUserId(uniqueId);

    try {
      const contract = await getContract();

      // Fetch the player's game points and token points
      const [userGamePoints, userTokenPoints] = await contract.getPlayerStats(uniqueId).call();
      setScore(userGamePoints || 0); // Initialize score with game points
      setTokenPoints(userTokenPoints || 0); // Initialize token points
    } catch (error) {
      console.error('Error initializing user:', error);
    }
  };
```

Figure 2 In This Part of Code, I defined different variables for Ball,Bricks,Score,Level,User ID & Created a function that will calculate points , score & get Telegram Username of a Player.

```
useEffect(() => {
  initializeUser();
}, [telegramUsername]);

// Convert points to tokens
const convertPointsToTokens = async () => {
  try {
    console.log('Fetching contract...');
    const contract = await getContract();
    console.log('Contract fetched successfully:', contract);

    console.log('Attempting to call convertPointsToTokens...');
    await contract.methods.convertPointsToTokens().send();
    console.log('convertPointsToTokens called successfully!');
    // Fetch updated stats after conversion
    const stats = await contract.methods.getPlayerStats(window.tronWeb.defaultAddress.base58).call();
    console.log('Updated player stats:', stats);

    // Update state with the fetched stats
    setTokenPoints(parseInt(stats.tokenPoints, 10)); // Update tokens
    setScore(parseInt(stats.gamePoints, 10)); // Update game points (if needed)

    alert('Points converted to tokens!');
  } catch (error) {
    console.error('Error converting points:', error);
    alert('There was an error converting points. Check the console for details.');
  }
};
```

Figure 3 In This part of code, I am defining a function which converts Score Points to Token Points, It Calls the Contract.js file which is linked to Tron Blockchain Shasta Testnet , When it is called it requires a specific condition i.e points >100 than it is executed and Transaction occurs on Blockchain Which I will be sharing Later.

```
useEffect(() => {
  // Run ball movement in intervals
  const interval = setInterval(() => {
    moveBall();
  }, 16); // 60 FPS
  return () => clearInterval(interval);
}, [ballPosition, ballVelocity, bricks]);
//Function to move the ball
const moveBall = () => {
  let { x, y } = ballPosition;
  let { dx, dy } = ballVelocity;
  // Ball collision with walls
  if (x <= 0 || x >= 600) dx = -dx;
  if (y <= 0) dy = -dy;
  // Ball collision with paddle
  if (y >= 580 && x >= paddlePosition && x <= paddlePosition + 200) {
    dy = -dy;
  }
  // Ball collision with bricks
  setBricks((prevBricks) =>
    prevBricks.map((brick) => {
      if (
        !brick.destroyed &&
        x >= brick.x &&
        x <= brick.x + 50 &&
        y >= brick.y &&
        y <= brick.y + 20
      ) {
        dy = -dy;
        setScore((prevScore) => prevScore + 10); // Increase score
        addGamePoints(10); // Call the smart contract function to add points
        return { ...brick, destroyed: true };
      }
      return brick;
    })
  );
```

Figure 4 In This Part of code, I am defining logic of a ball and bricks when it breaks the brick & speed of the ball & score when ball breaks the brick

```
  // Update ball position
  setBallPosition({ x: x + dx, y: y + dy });
  setBallVelocity({ dx, dy });

  // Check if all bricks are destroyed and go to next level
  if (bricks.every((brick) => brick.destroyed)) {
    handleNextLevel();
  }

  // Ball falls below the paddle
  if (y >= 600) {
    resetGame();
  }
};

// Function to move the paddle
const movePaddle = (direction) => {
  const paddleSpeed = 4 * 20; // Increase paddle speed by 4x
  if (direction === 'left' && paddlePosition > 0) {
    setPaddlePosition((prev) => prev - paddleSpeed);
  }
  // Adjust to prevent paddle from going off-screen
  if (direction === 'right' && paddlePosition < 400) {
    setPaddlePosition((prev) => prev + paddleSpeed);
  }
};

// Function to handle next level
const handleNextLevel = () => {
  setLevel((prevLevel) => prevLevel + 1);
  setBallPosition({ x: 300, y: 400 }); // Reset ball position
  setBricks(createBricks(level + 1)); // Create new level of bricks
};
```

Figure 5 In this Part of Code, I am defining the logic of Ball & paddle position that wheny the ball should bounce back

```
// Function to reset the game
const resetGame = () => {
  alert('Game Over!');
  setScore(0); // Reset score
  setLevel(1); // Reset level
  setBallPosition({ x: 300, y: 400 }); // Reset ball position
  setBallVelocity({ dx: 5, dy: -5 }); // Reset ball speed to initial speed
  setBricks(createBricks(1)); // Reset bricks for level 1
};

// Add points to the contract
const addGamePoints = async (points) => {
  try {
    const contract = await getContract();
    await contract.methods.addGamePoints(userId, points).send();
    alert(`Added ${points} points!`);
  } catch (error) {
    console.error('Error adding game points:', error);
  }
};
// Touch event handling
const handleTouchStart = (e) => {
  setTouchStartX(e.touches[0].clientX); // Record the initial touch position
};

const handleTouchMove = (e) => {
  const touchX = e.touches[0].clientX;
  const deltaX = touchX - touchStartX;
  const newPaddlePosition = paddlePosition + deltaX;

  // Ensure the paddle stays within the bounds of the game area
  if (newPaddlePosition >= 0 && newPaddlePosition <= 400) {
    setPaddlePosition(newPaddlePosition);
  }
  setTouchStartX(touchX); // Update touch start position
};
```

Figure 6 This code will execute a backend transaction Calling Contract.js file and interacting with Smart Contract Deployed on Blockchain converting Points to Token for that it will pass parameters to backend.

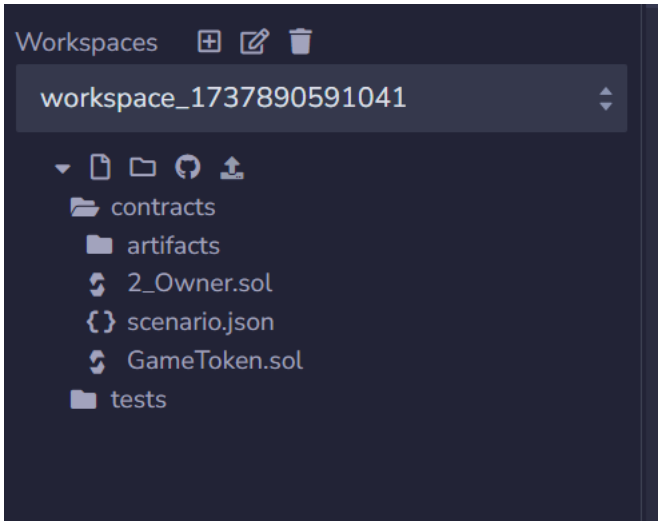
```
return (
  <div
    ref={boardRef}
    style={{
      width: '600px',
      height: '600px',
      backgroundColor: 'lightblue',
      position: 'relative',
      overflow: 'hidden',
      margin: 'auto',
    }}
    onTouchStart={handleTouchStart}
    onTouchMove={handleTouchMove}
  >
    {/* Score and Level Display */}
    <h2 style={{ textAlign: 'center', position: 'absolute', top: '1px', width: '100%',
        fontSize: '20px', color: 'black' }}>
      Username: {telegramUsername} | Score: {score} | Level: {level}
    </h2>

    {/* Ball, Paddle, and Bricks */}
    <Ball position={ballPosition} />
    <Paddle position={paddlePosition} movePaddle={movePaddle} />
    <Bricks bricks={bricks} />
```

Figure 7 In This part of code, UI is defined & the app should work on screen touch.

```
222          {/* Convert Points Button */}
223          <div style={{
224            position: 'absolute',
225            top: '80%',
226            right: '10px',
227            backgroundColor: '#4CAF50',
228            color: 'white',
229            padding: '10px 20px',
230            border: 'none',
231            borderRadius: '5px',
232            cursor: 'pointer',
233          }} onClick={convertPointsToTokens}>
234            Convert Score to Tokens
235          </div>
236
237          {/* Display Token Points */}
238          <div style={{
239            position: 'absolute',
240            top: '90%',
241            right: '10px',
242            fontSize: '16px',
243            color: 'black',
244          }}>
245        {/* Tokens : {tokenPoints} */}
246          </div>
247        </div>
248      );
249    };
250
251    export default GameBoard;
```

Figure 8 It's 2nd part for UI

- **Creating Smart Contract & Deploying it**, for that go to https://www.tronide.io/ & it will ask to connect **TronLink Wallet** (You Must have installed Extension of Wallet from chrome webstore) and click on create **New** Workspace, Create a File inside **Contracts** Folder by name **GameToken.sol** the rest files are default.

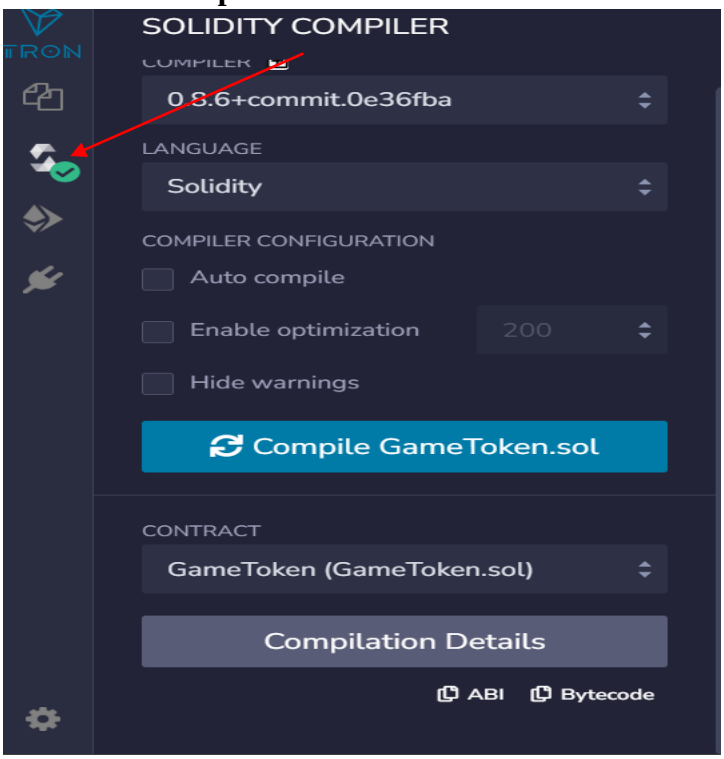


Than inside that write the smart contract that what should it do, in our case it is this,

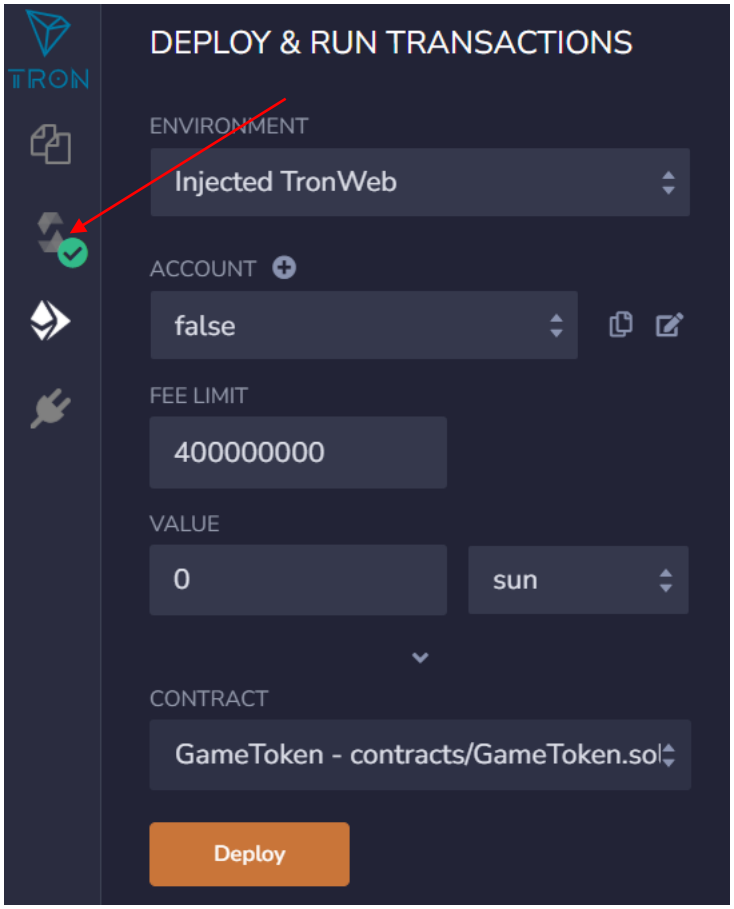
```
contract GameToken {
    uint256 public constant CONVERSION_RATE = 100;
    mapping(address => uint256) public gamePoints;
    mapping(address => uint256) public tokenPoints;

    event PointsConverted(address indexed user, uint256 gamePointsUsed, uint256 tokenPointsAwarded);

    function addGamePoints(address user, uint256 points) public {
        require(points > 0, "Points must be greater than zero");
        gamePoints[user] += points;
    }

    function convertPointsToTokens() public {
        uint256 userGamePoints = gamePoints[msg.sender];
        require(userGamePoints >= CONVERSION_RATE, "Not enough game points to convert");

        uint256 tokenPointsToAdd = userGamePoints / CONVERSION_RATE;
        uint256 remainingGamePoints = userGamePoints % CONVERSION_RATE;

        gamePoints[msg.sender] = remainingGamePoints;
        tokenPoints[msg.sender] += tokenPointsToAdd;

        emit PointsConverted(msg.sender, userGamePoints - remainingGamePoints, tokenPointsToAdd);
    }

    function getPlayerStats(address user) public view returns (uint256, uint256) {
        return (gamePoints[user], tokenPoints[user]);
    }
}
```

Figure 9 In this code, I had create a contract GameToken, Contract is equal to Class in C++, inside Contract we had different Variables defined i.e gamepoint, token point, after that we had added a function which converts earned score in game to Token (this transaction would take place on blockchain) as you can see below in code, msg.sender it send a transaction to specific wallet which will be connected to app,

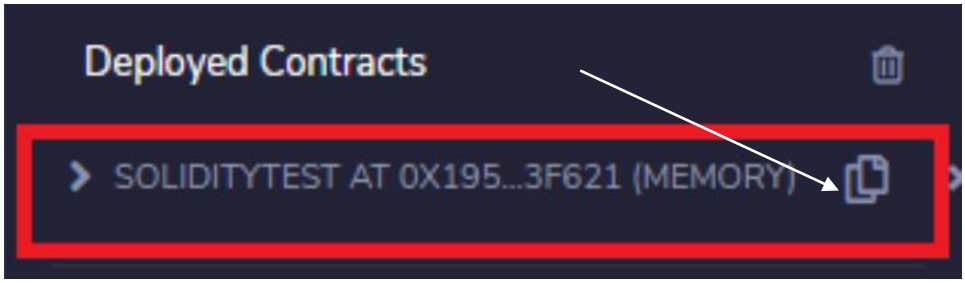
- Now it's Time to Deploy the Smart Contract. First click on **Solidity Compiler** on left side 2nd option and Click on **Compile GameToken.sol**



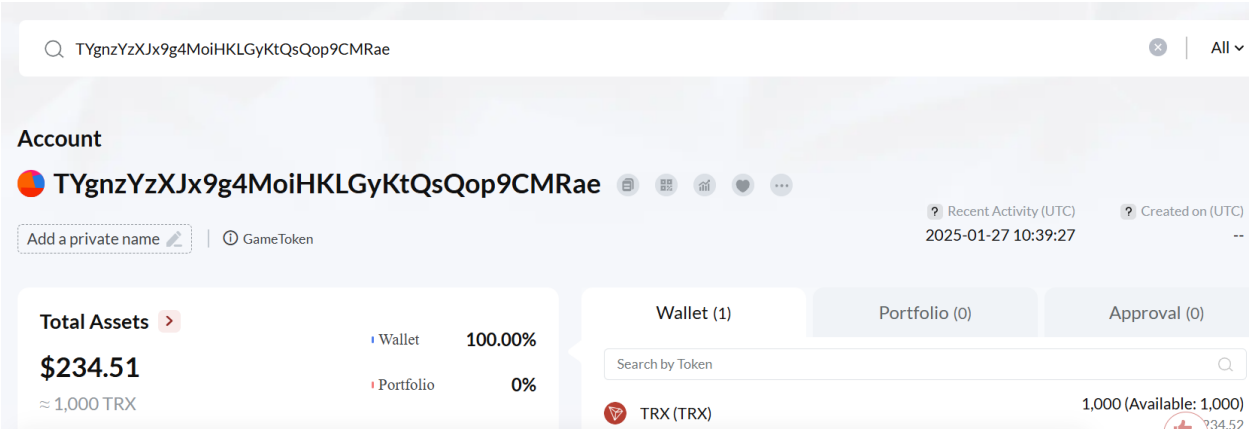
- After Compilation, Click on **Deploy & Run Transaction** 3rd option on left side, select the Environment **Injected TronWeb** & rest of setting default as it is, than click on Deploy.



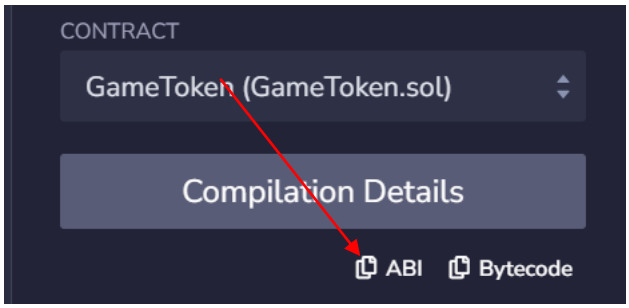
- Once Deployed you will see Address like this, than copy that address and save it.



- Now Let's Check it is deployed or not, go to https://shasta.tronscan.org/#/ paste the copied **contract address** there you will get interface like this it means your contract was deployed properly. For TRX to be deposited in your wallet go to https://shasta.tronex.io/ & enter your wallet address which you will get in your wallet, enter it and you will get 2k TRX **Note**: The TRX in the Total assets are not given as default, you can transfer it from your tronlink wallet.



- Also when you deploy the Smart Contract than go to **Solidity Compiler** and there will be option **ABI** copy it and save it, it will be needed later.

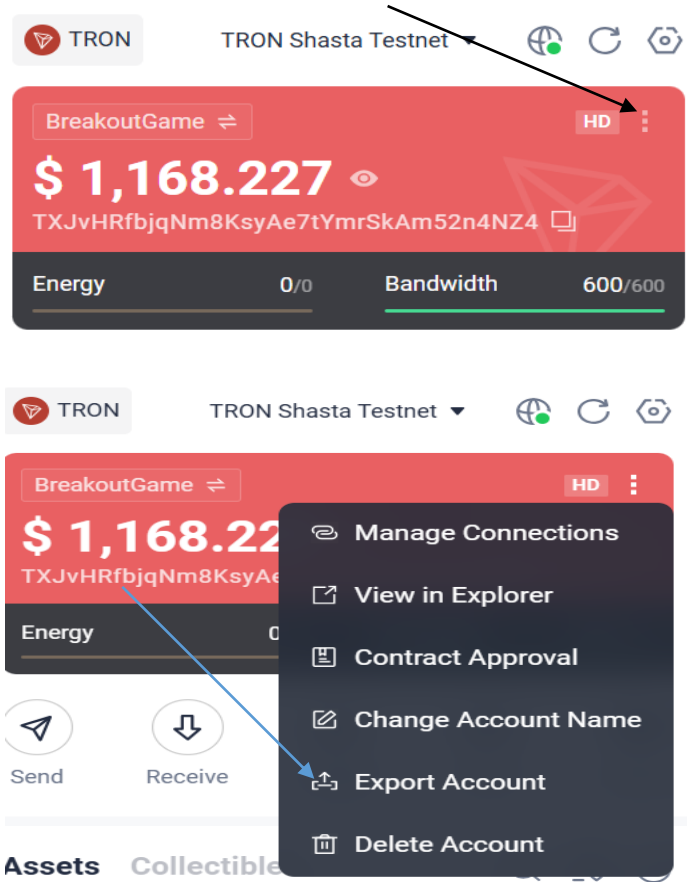


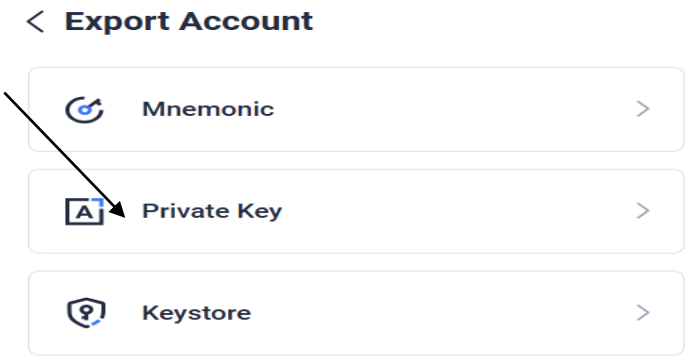
- Now, I have To call Contract to our react app , for that I had created a file's **contract.js** & **tronWeb.js**, here is the code of **tronWeb.js**.

```
import TronWeb from 'tronweb';
const tronWeb = new TronWeb({
  fullHost: 'https://api.shasta.trongrid.io', // Use the appropriate testnet or mainnet URL
  privateKey: 'e5fd333b64edd0d3603686b088257196b47dba0bf0f0c6bd13ed7d9f9df2d23a'
  // Use your wallet's private key for signing
});

export default tronWeb;
```

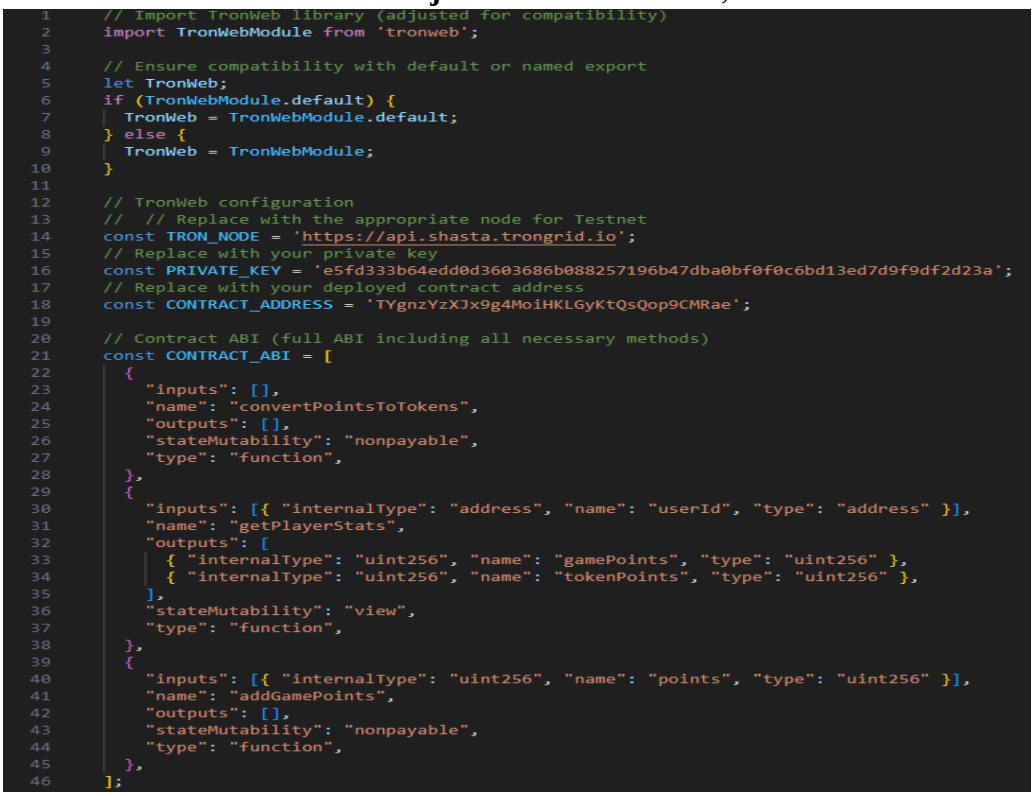
Figure 10 The code takes users wallet address which is called through Private Key, the fullhost is a public node of shasta Testnetwork which is availible on their website. I had also imported the tronweb from tronweb library as we installed it earlier using npm install tronweb.

- We can get **Private key** from Wallet, go to chrome extension of TronLink wallet, click on the **3 dots**, then click on **Export Account** after that click on **private key** enter your wallet password and copy **private key**.





- Now let's write the code for **contract.js** as mentioned earlier, here is the code.



```
// Import TronWeb library (adjusted for compatibility)
import TronWebModule from 'tronweb';

// Ensure compatibility with default or named export
let TronWeb;
if (TronWebModule.default) {
  TronWeb = TronWebModule.default;
} else {
  TronWeb = TronWebModule;
}

// TronWeb configuration
//  // Replace with the appropriate node for Testnet
const TRON_NODE = 'https://api.shasta.trongrid.io';
// Replace with your private key
const PRIVATE_KEY = 'e5fd333b64edd0d3603686b088257196b47dba0bf0f0c6bd13ed7d9f9df2d23a';
// Replace with your deployed contract address
const CONTRACT_ADDRESS = 'TYgnzYzXJx9g4MoiHKLGyKtQsQop9CMRae';

// Contract ABI (full ABI including all necessary methods)
const CONTRACT_ABI = [
  {
    "inputs": [],
    "name": "convertPointsToTokens",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function",
  },
  {
    "inputs": [{ "internalType": "address", "name": "userId", "type": "address" }],
    "name": "getPlayerStats",
    "outputs": [
      { "internalType": "uint256", "name": "gamePoints", "type": "uint256" },
      { "internalType": "uint256", "name": "tokenPoints", "type": "uint256" },
    ],
    "stateMutability": "view",
    "type": "function",
  },
  {
    "inputs": [{ "internalType": "uint256", "name": "points", "type": "uint256" }],
    "name": "addGamePoints",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function",
  },
];
```

*Figure 11 In this part of code I defined that to which node of Tron network it will connect, Given the private key, contract address of Smart Contract deployed on Shasta Testnet After that the **ABI** is defined which i copied earlier when we deployed Smart Contract. ABI is the converted from of Smart contract written in Solidity,*

```
// Initialize TronWeb instance
const tronWeb = window.tronWeb && window.tronWeb.defaultAddress.base58
  ? window.tronWeb // Use injected TronWeb instance if available (e.g., TronLink)
  : new TronWeb({
      fullHost: TRON_NODE,
      privateKey: PRIVATE_KEY,
    });

// Function to get the contract instance
export const getContract = async () => {
  try {
    if (!CONTRACT_ABI || !CONTRACT_ADDRESS) {
      throw new Error('Contract ABI or address is missing.');
    }
    // Fetch and return the contract instance
    const contract = await tronWeb.contract(CONTRACT_ABI, CONTRACT_ADDRESS);
    return contract;
  } catch (error) {
    console.error('Error loading contract:', error);
    throw error;
  }
};

// Example function: Add game points
export const addGamePoints = async (points) => {
  try {
    const contract = await getContract();
    await contract.methods.addGamePoints(points).send();
    console.log('Game points added successfully!');
  } catch (error) {
    console.error('Error adding game points:', error);
    throw error;
  }
};

// Example function: Convert points to tokens
export const convertPointsToTokens = async () => {
  try {
    const contract = await getContract();
    await contract.methods.convertPointsToTokens().send();
    console.log('Points converted to tokens!');
  } catch (error) {
    console.error('Error converting points to tokens:', error);
    throw error;
  }
};
```

Figure 12 This part of code defines that if anything i.e ABI or contract address is missing it will pop up error & Two functions are defined, the 1st send request to app and ask for how many points does the player had, so according to it it adds the points & 2nd one converts points to token.

```
// Example function: Get player stats
export const getPlayerStats = async (userId) => {
  try {
    const contract = await getContract();
    const stats = await contract.methods.getPlayerStats(userId).call();
    return stats; // { gamePoints, tokenPoints }
  } catch (error) {
    console.error('Error fetching player stats:', error);
    throw error;
  }
};
```

Figure 13 In this part of code , I am getting player userid

- Now calling all other files to **App.jsx.**

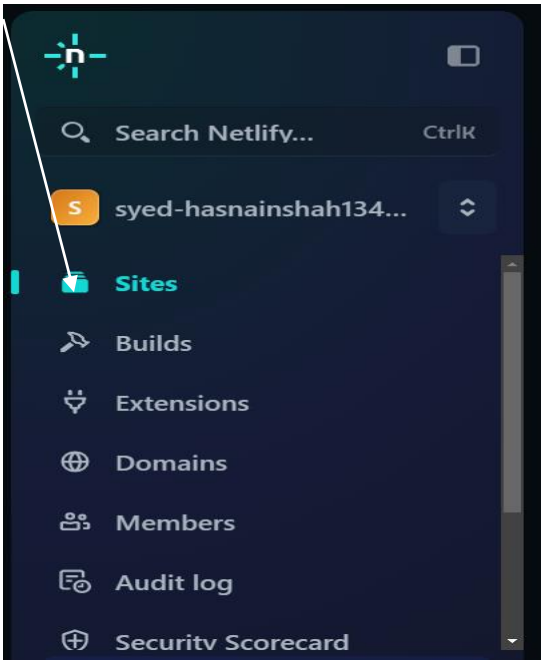
```jsx
import React from 'react';
import GameBoard from './components/GameBoard';
import './App.css'; // Ensure this file exists
import './index.css';
import './App.css';


function App() {
  return (
    <div className="App">
      <h1><b>Breakout Game By SYED HASNAIN SHAH </b> </h1>
      <GameBoard />
    </div>
  );
}

export default App;
```

- Now importing **App.jsx** to **main.jsx**.

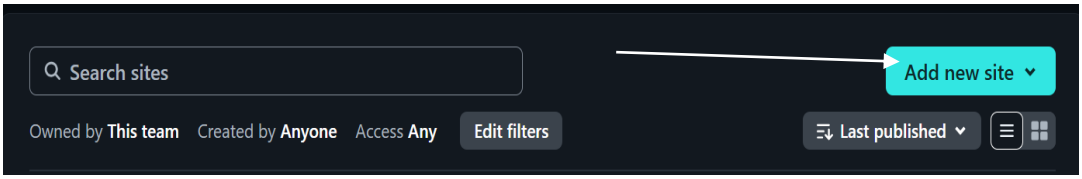
```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css'; // Ensure this file exists
import App from './App';
import './index.css'; // Optional: Add global styles
import './index.css';
import './App.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

*Figure 14 The **ReactDom** shows us the code in browser, it creates local host.*

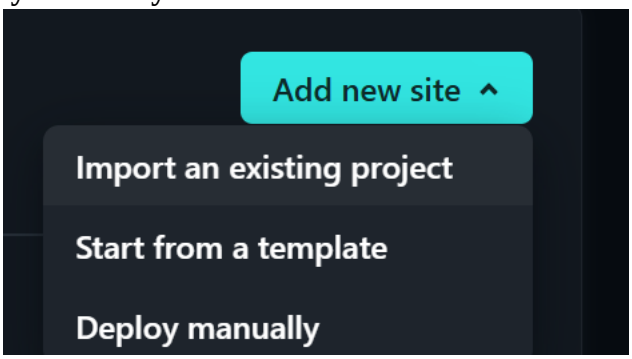
- Our **DAPP** is ready now, Let's host it on **Telegram** using **app.netlify.com**, first create account in it, next go to **sites** page



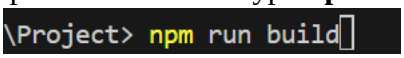
- After that click on **Add new site**



- Then Click on **Deploy Manually**

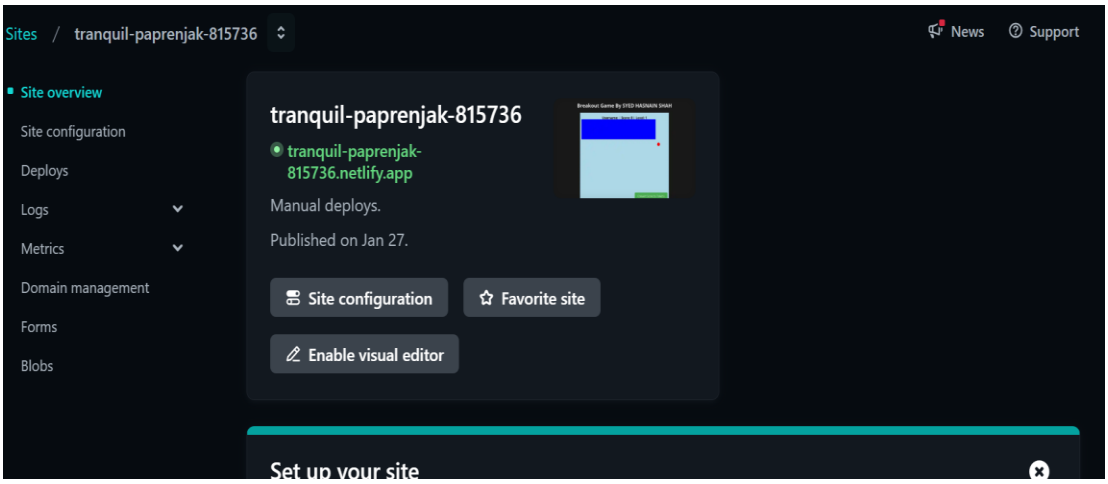


- Now Go to your project and open terminal and Type **npm run build**

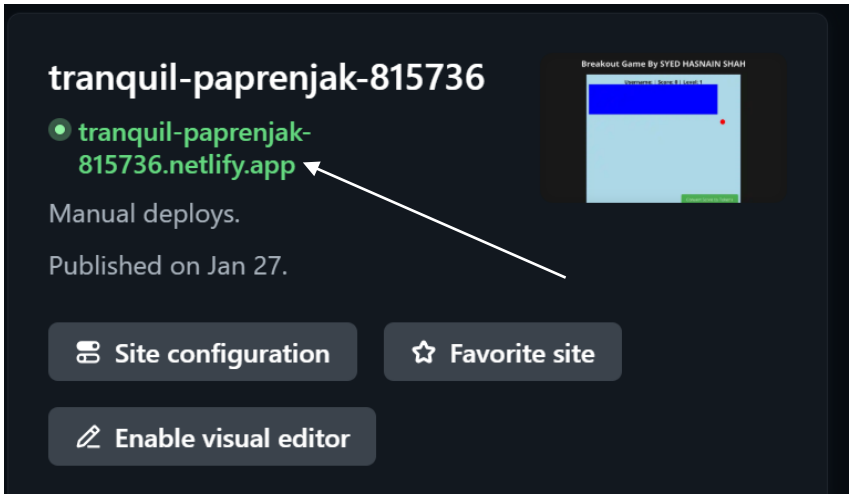


- Now it will create a folder name dist in your project folder.

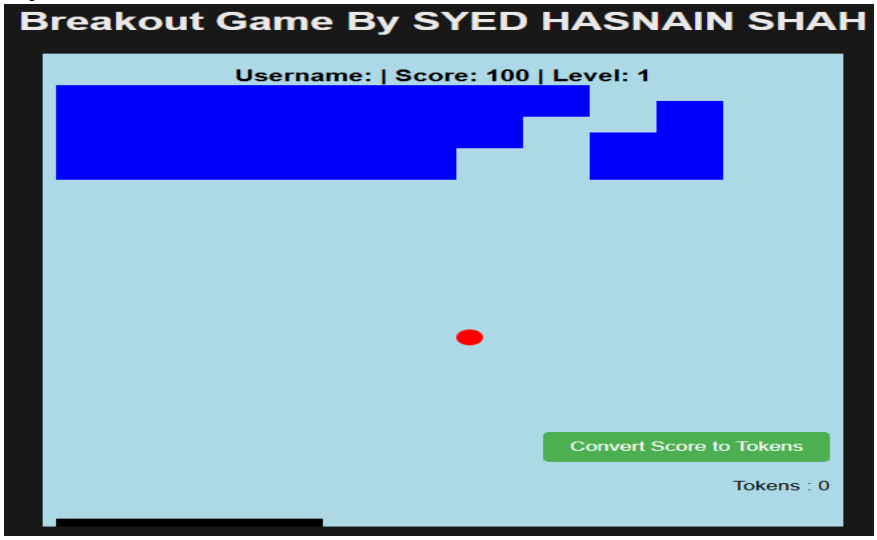
- Now upload this **dist** folder to **app.netlify.com** by option deploy manually, after that you will see a interface like this



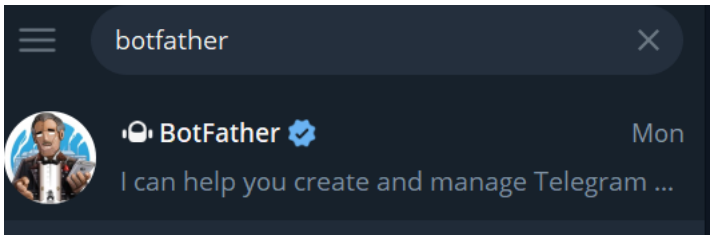
- Then click on this link.



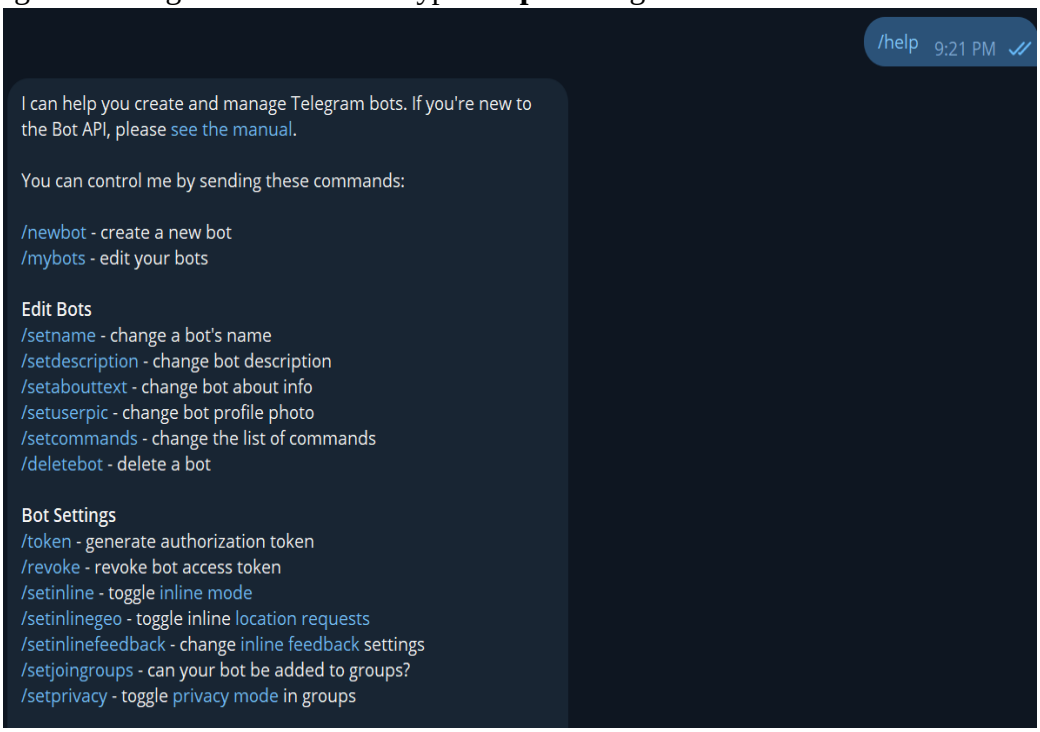
- It will direct you to Live **DAPP**

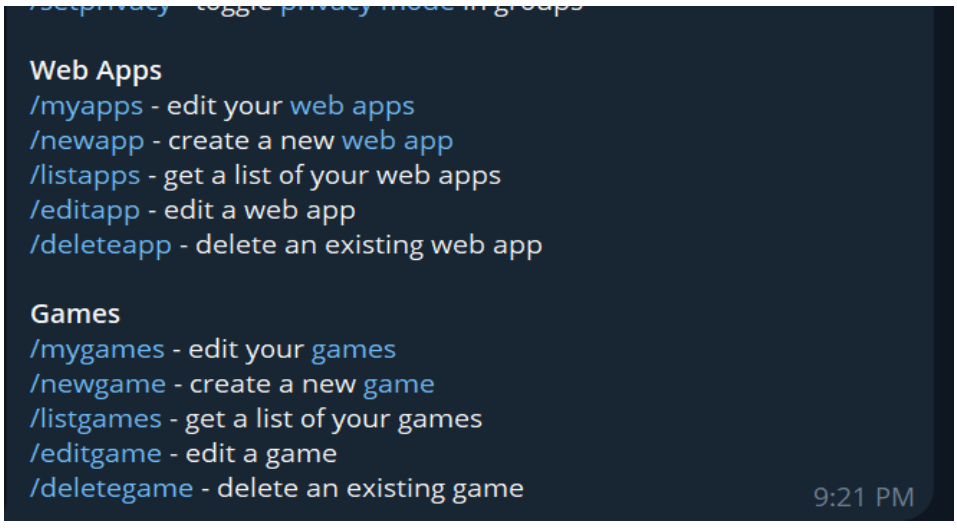


- Now go to **Telegram** & serach for **botfather**

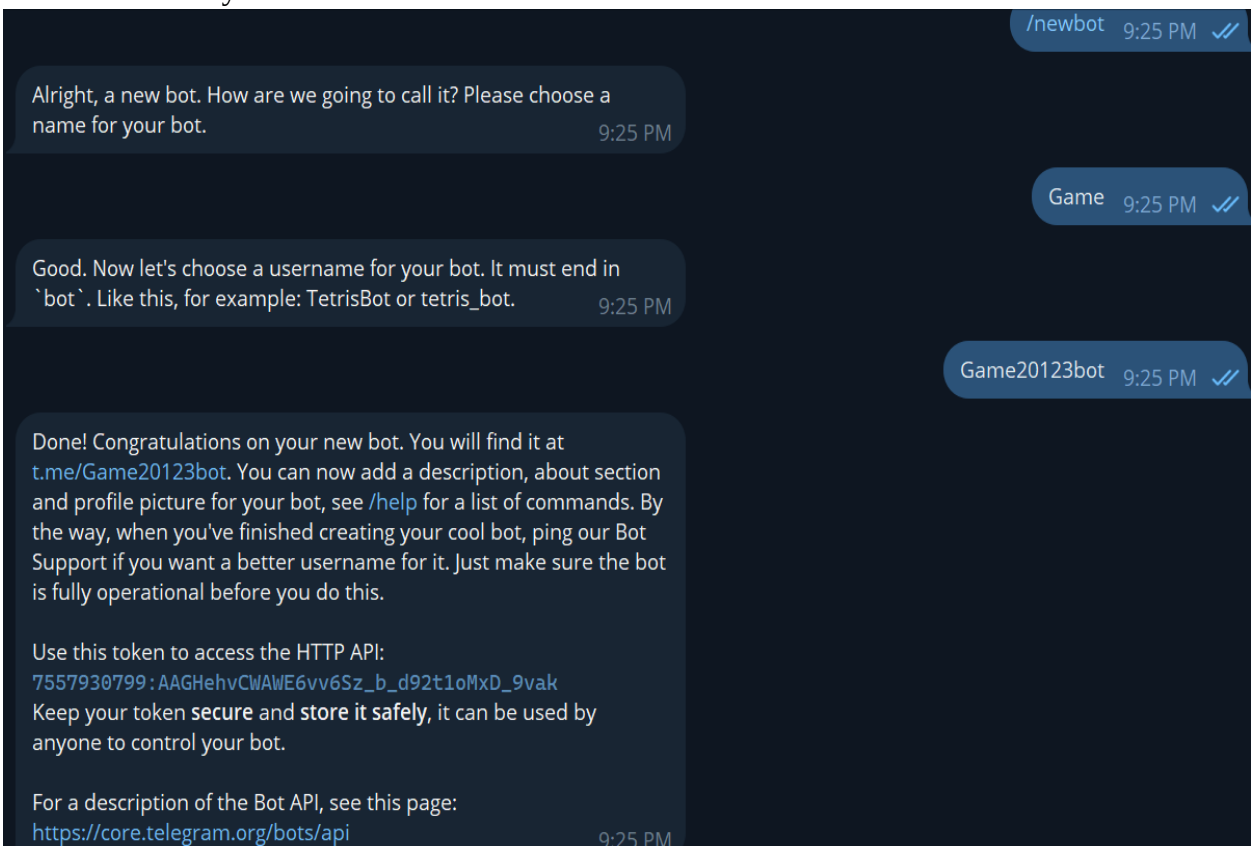


- Now go to message of **botfather** & type **/help** it will generate all cmds.

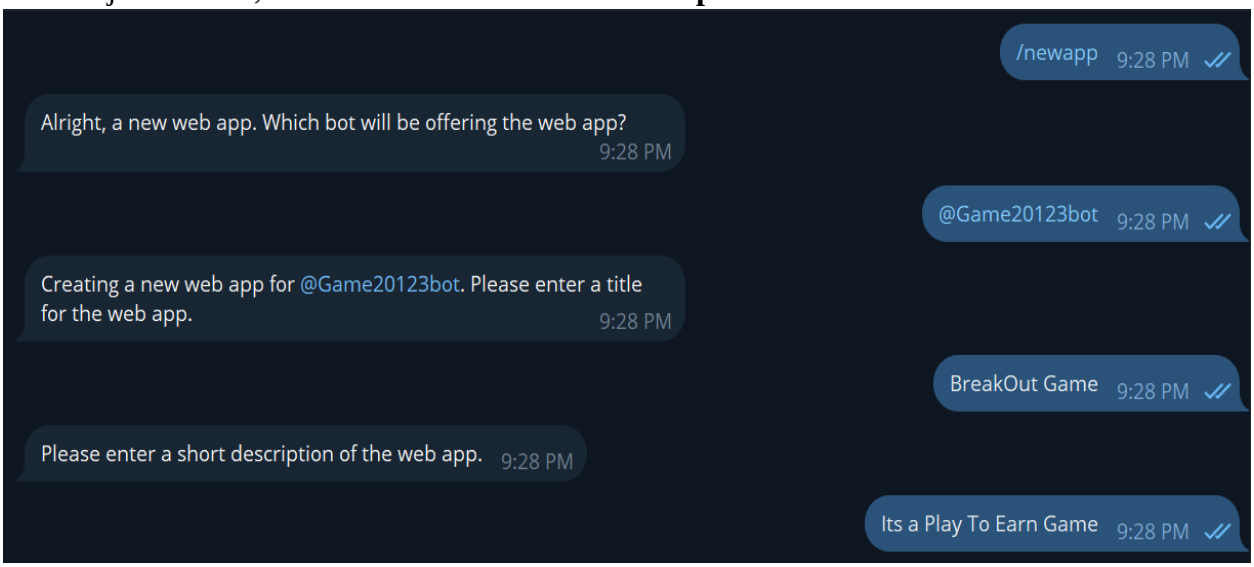




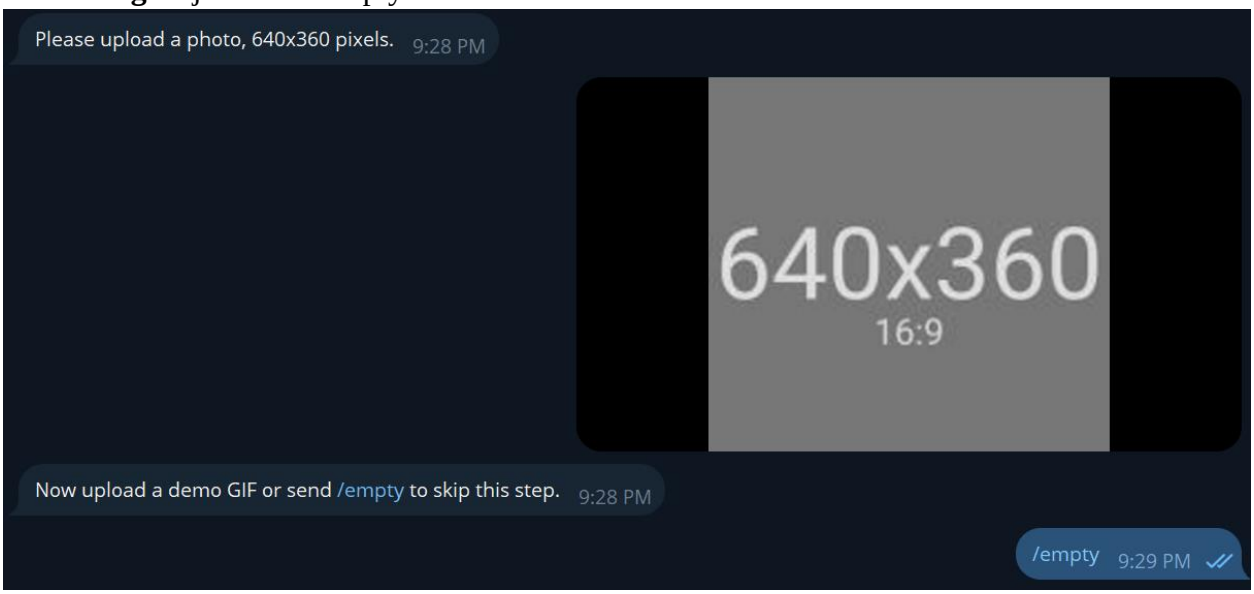
- Now type **/newbot**, after that it will ask you to **name Bot**, hen **Bot username** now your bot is successfully created.



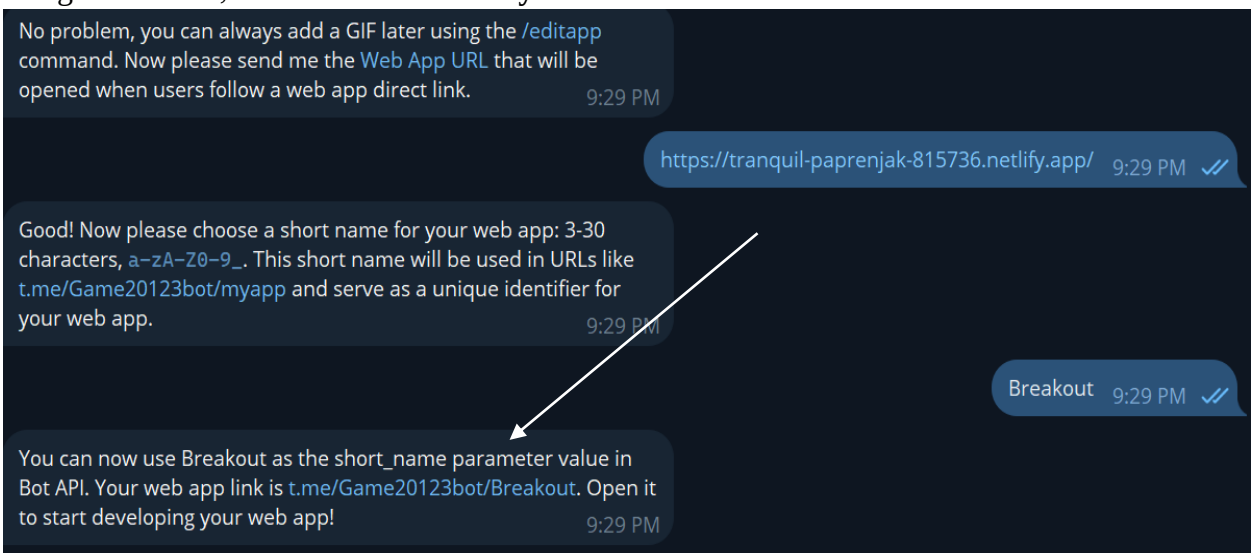
- Now type **/newapp** then it will ask which bot will be offering the **web app** select the **bot** which I just created, then it will ask for a **short description.**



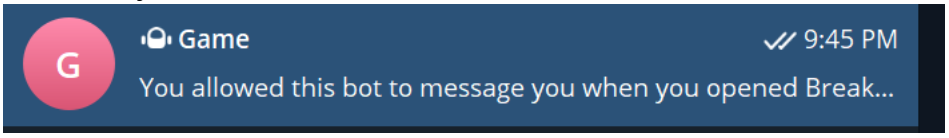
- Now it will ask for a photo for **logo**, I just uploaded a **random** picture, after that it asked for **demo gif** I just left it empty.



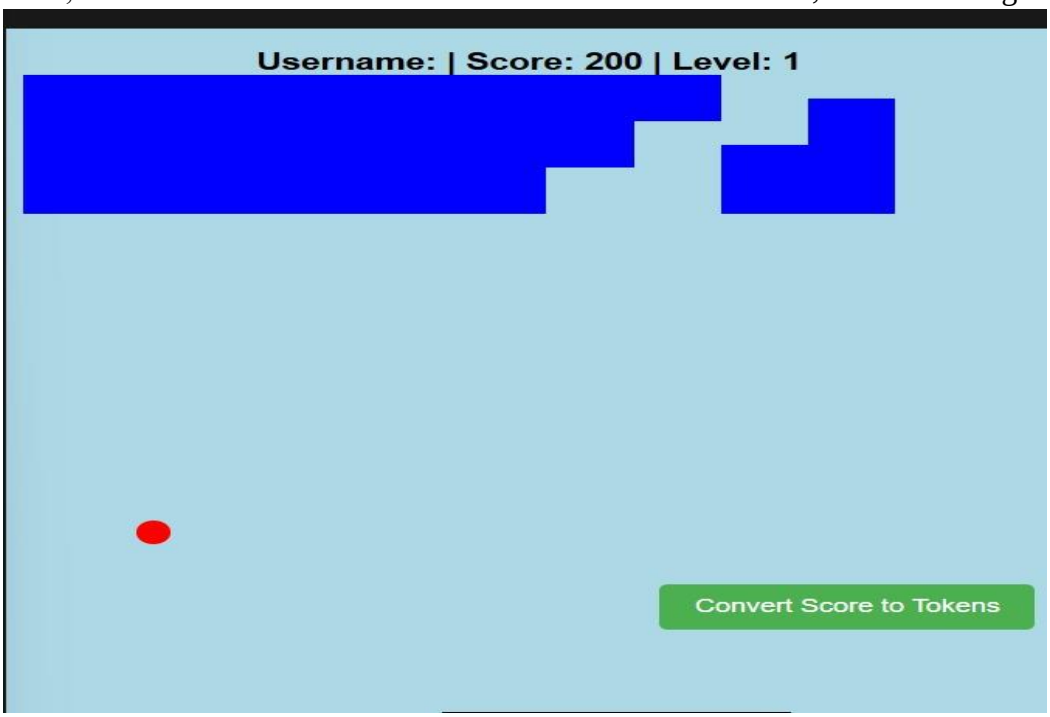
- Now it asks for **WEB URL**, I just copied the URL which we got from **app.netlify.com** and give it to bot, Now **Our Bot is ready**.



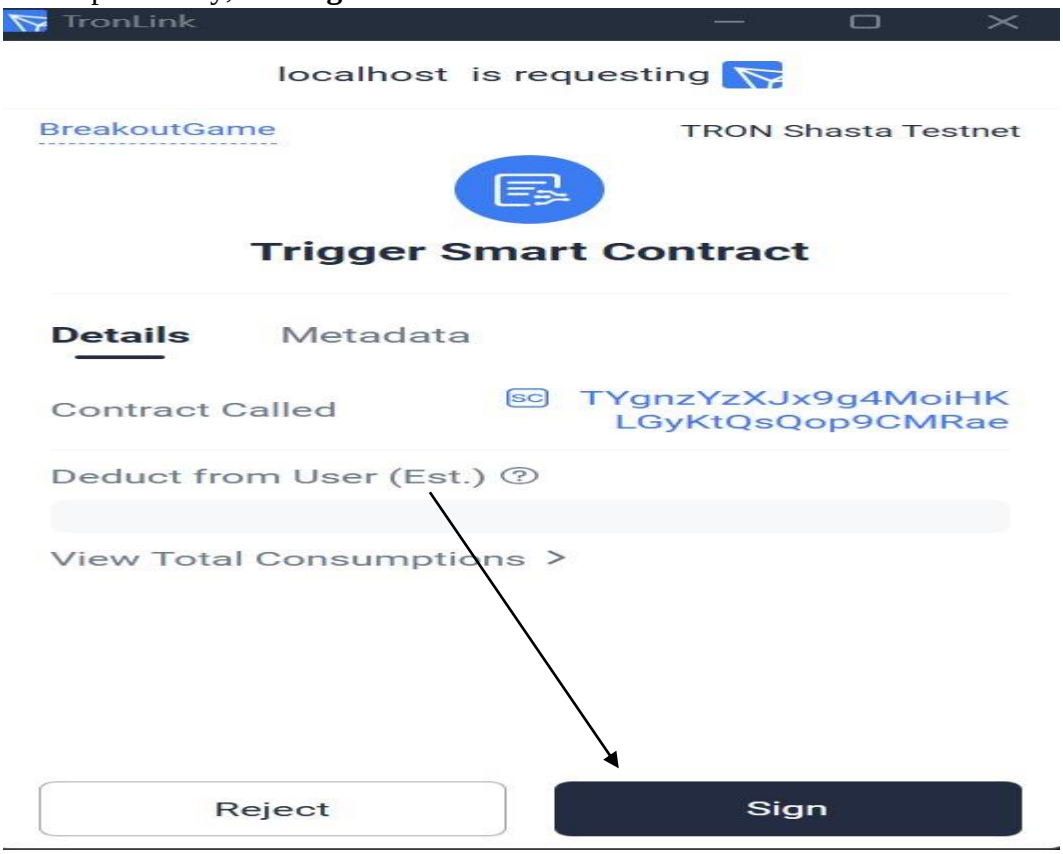
- Now, click on the **link** in the upper picture it will create a bot message like this and our bot is ready now



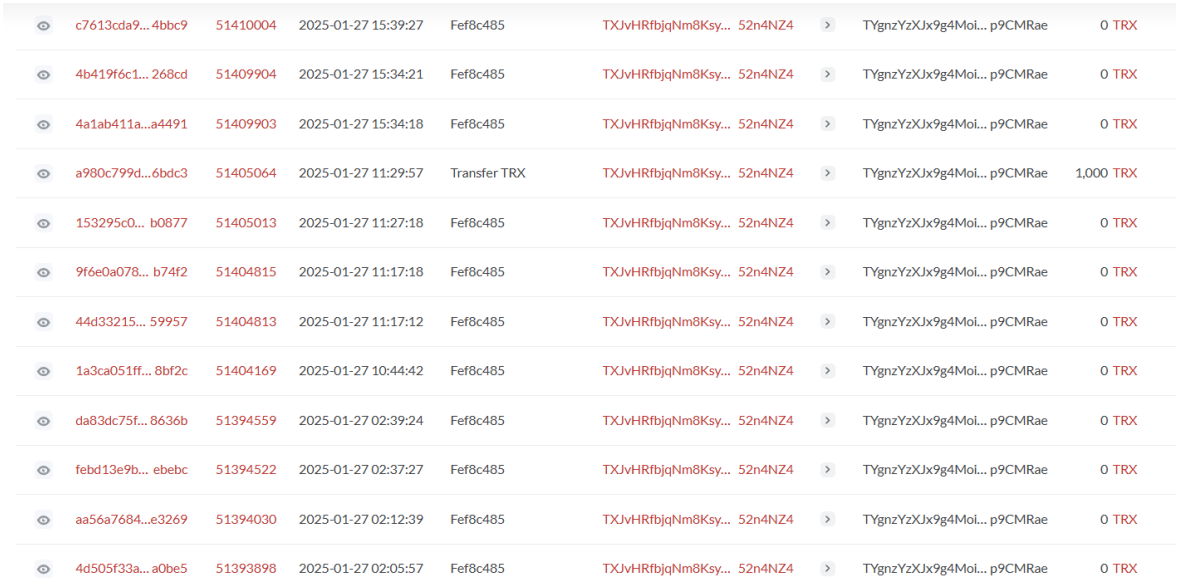
- Now, let's test the Game & backend transaction on block chain, now start the game.



- The game is working properly now let's test the **Convert Score to Tokens function**, when I clicked on **Convert Score To Tokens** it opens authorization of **tronlink** wallet which I linked previously, then **sign the contract**.



- After **signing the contract** the transaction will **occur on blockchain** and it will **convert points to tokens**, here are the transaction which occurred when I was testing it, you can check transactions at https://shasta.tronscan.org/#/address/TYgnzYzXJx9g4MoiHKLGyKtQsQop9CMRae



c7613cda9... 4bbc9	51410004	2025-01-27 15:39:27	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
4b419f6c1... 268cd	51409904	2025-01-27 15:34:21	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
4a1ab411a...a4491	51409903	2025-01-27 15:34:18	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
a980c799d...6bdc3	51405064	2025-01-27 11:29:57	Transfer TRX	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	1,000 TRX
153295c0... b0877	51405013	2025-01-27 11:27:18	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
9f6e0a078... b74f2	51404815	2025-01-27 11:17:18	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
44d33215... 59957	51404813	2025-01-27 11:17:12	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
1a3ca051ff... 8bf2c	51404169	2025-01-27 10:44:42	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
da83dc75f... 8636b	51394559	2025-01-27 02:39:24	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
febd13e9b... ebebc	51394522	2025-01-27 02:37:27	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
aa56a7684...e3269	51394030	2025-01-27 02:12:39	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX
4d505f33a... a0be5	51393898	2025-01-27 02:05:57	Fef8c485	TXJvHRfbjqNm8Ksy...	52n4NZ4	TYgnzYzXJx9g4Moi...	p9CMRae	0 TRX

CONCLUSION

- This was my Object Oriented Programming Lab Project in which I learned a lot of new things as it was unique, because of this project my grip on block chain is increased as it will help me a lot in future.