**Name : Nasir Abbas**

**Reg: 20Pwbcs0745**

**Group Member: Baidar Ahmad**

**Group Mem Reg: 20Pwbcs0733**

**Class: 8th A**

**Date: 6/4/2024**

**Assignment : 02**

**Subject: Distributed Computing (BlockChain)**

**Problem Statement**

Create a smart contract that allows users to vote for candidates in an election. The contract

should allow for the following functionalities:

1. Adding candidates.

2. Casting votes.

3. Viewing the results of the election.

**Solving Problem**

**Part of Installation**

1. First I am download the node js and install it.
2. Second I am install truffle with the command of **“npm install -g truffle”** using cmd.
3. I have already installed the “metamask” but I am not using metamask because I am

Using Vs Code I can directly configure to ganache with little bit configuration in the portion of

Networking I will explain these steps letter.

1. I have already installed the ganache.

**Initialize a Truffle Project**

1. First I am create a Directory with the name of VotingApp
2. I open this Directory within Vs Code Editor
3. I open the terminal within Vs code type the command “Truffle init”
4. It give me some file
5. Contracts
6. Migrations
7. Test
8. Truffle-config.js

I can write our contracts within Contracts directory, this will be important for compilation for smart contracts.

I can write our deployment codes in the Migration director, this will be important for deployment for smart contracts.

All test of our project are store in the Test director**.**

Whenever I want to configure compiler version and configuration with

Local networking I will use “truffle-config.js file”

**Creating Smart Contracts**

1. First Create a new file name it “Voting.sol”
2. Defined the license
3. Defined the version of Compiler
4. Created a “Voting Contract”
5. Created a Structure name of Candidate within Structure I declare the
6. unit id,
7. String name,
8. Uint VoteCount

Why I use the unit data type because it can store only positive number.

**Variables**

1. I created a variable name it “owner” with data type of address.

Why I created this variable because it can restrict the smart contract like owner can add the candidate. Within this variable only owner address will be store

1. I created a candidatesCount variable with data type of unit. The purpose of this variable is to keep track of the total number of candidates that have been added to the election.

When new candidate will be added it can increment by 1, that ensure that every candidate will be uniquely identified.

1. Mapping state variable named “candidates” with “Candidate Structure”. This mapping is used to store and mange candidates in the election.

The mapping associates each candidates ID with “Candidate Structure”. The Candidate Structure contains information about the candidate such as their ID, name, vote count.

We can retrieved information of candidates and store information of candidates with key of ID.

1. Mapping state variable named “voter” with “ voter address”. This mapping is used to keep track of which addressed have voted in the election. The Boolean value indicates whether the address has already voted or not.

**Modifer**

1. I created a modifier named it onlyOwner is used only be called by the contract owner.

Inside the modifier the “require function” is used to check if the caller of the function (msg.sender) is the same as the owner of the contract . if this condition is not meet the transaction is reverted and the message “only owner can call this function” is returned.

**Constructor**

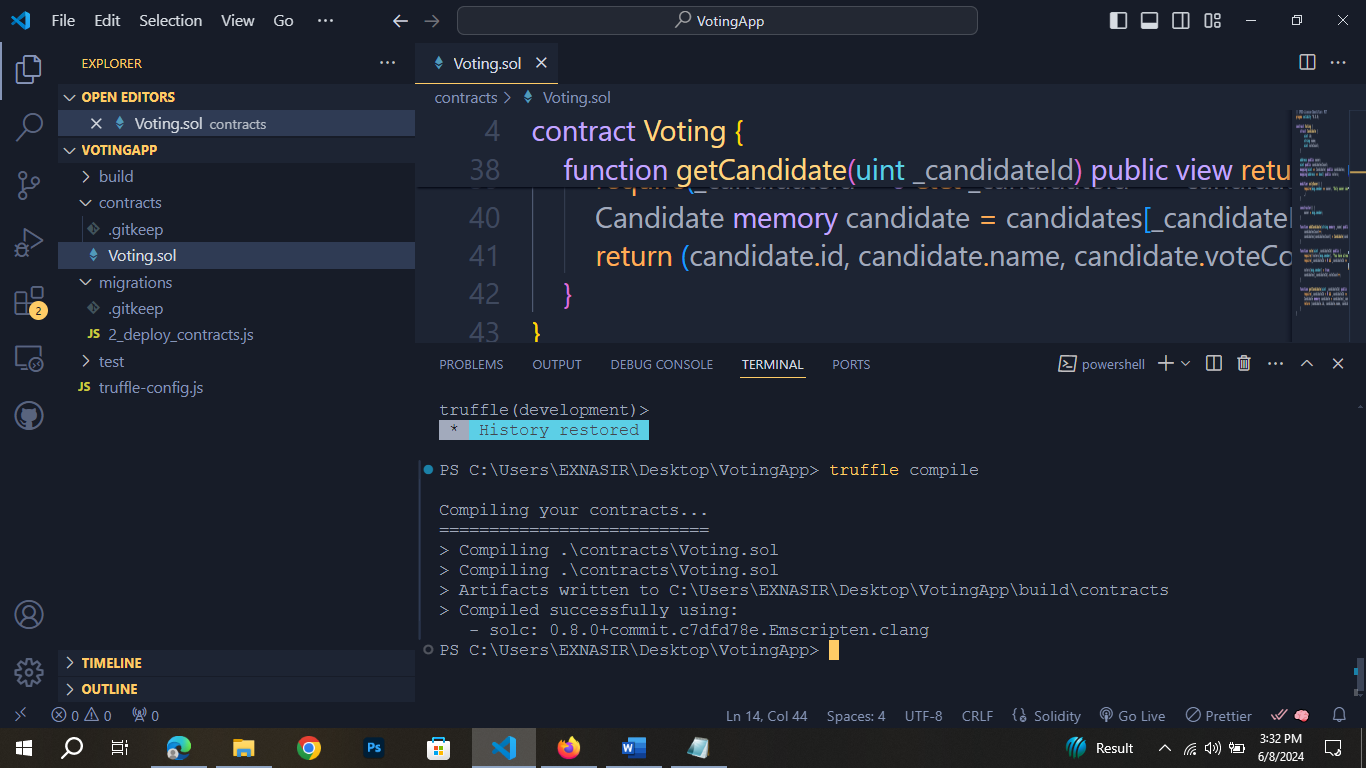
1. I created the Constructor, It can execute only once during the execution of smart contracts, In this contract is to assign the ownership of the contract to deployer. The contract ensure that the deployers address is stored in the owner variable.

**Functions**

1. I created a AddCandidates function, The main objective of that function is to only owner can allow to add new candidate. It is declared as public, meaning it can be called by any external entity. However it is restricted by the “onlyOwner” modifier which means only contract owner can actually execute it.  
   1. candidateCount: This is the Id of the new candidate. It is a unique identifier that is incremented each time candidate is added  
   2. Candidate(candidatesCount, \_name, 0): The 0 means that represent the initial vote count for the new candidate.
2. The Vote function in the smart contract allows registered users to cast their vote for a specific candidate. The function is declare as public, meaning it can be called by any external entity. It can also be checking if the candidate is already voted or not, It can check the validity of candidate id with some constraints, It can also update the or increment the candidateCount whenever marking the voter as having voted.
3. The getCandidate function allow users to retrieve information about a specific candidate. The Function is declare as public View, which means that it can be called by any external entity and does not modify the state of the contract. It will be used for to retrieved information of candidates like name, id. The main primary purpose of the this function is to provide a way for users to query and retrieve detailed about a specific candidate using their candidate ID.

**Compiling Smart Contract:**

I compiled the smart contract with the help of truffle command “truffle compile”. It can run all contracts in the “contracts” directory.

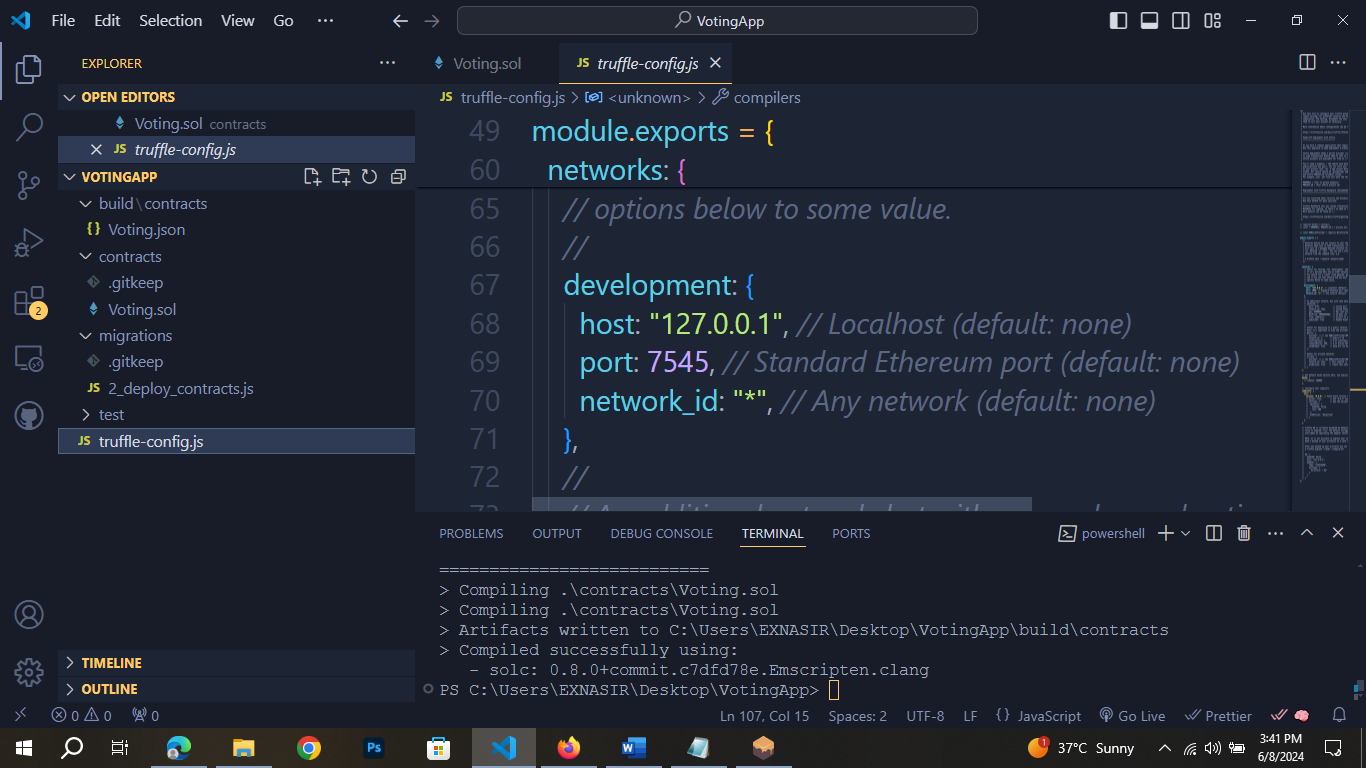


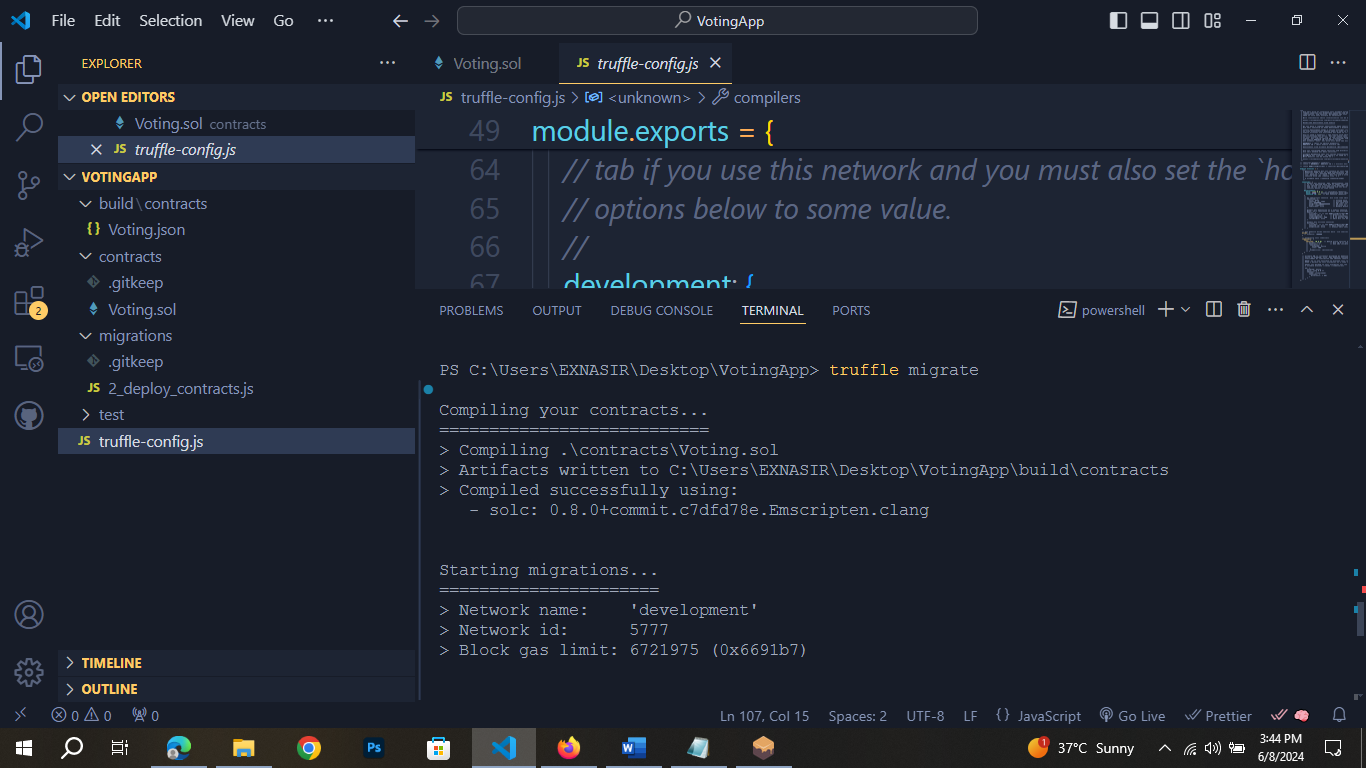
After compilation it can generate the new directory in which is “build”. In this directory one file contains which is name “Voting.json”, it can store all transactions in the form of json format.

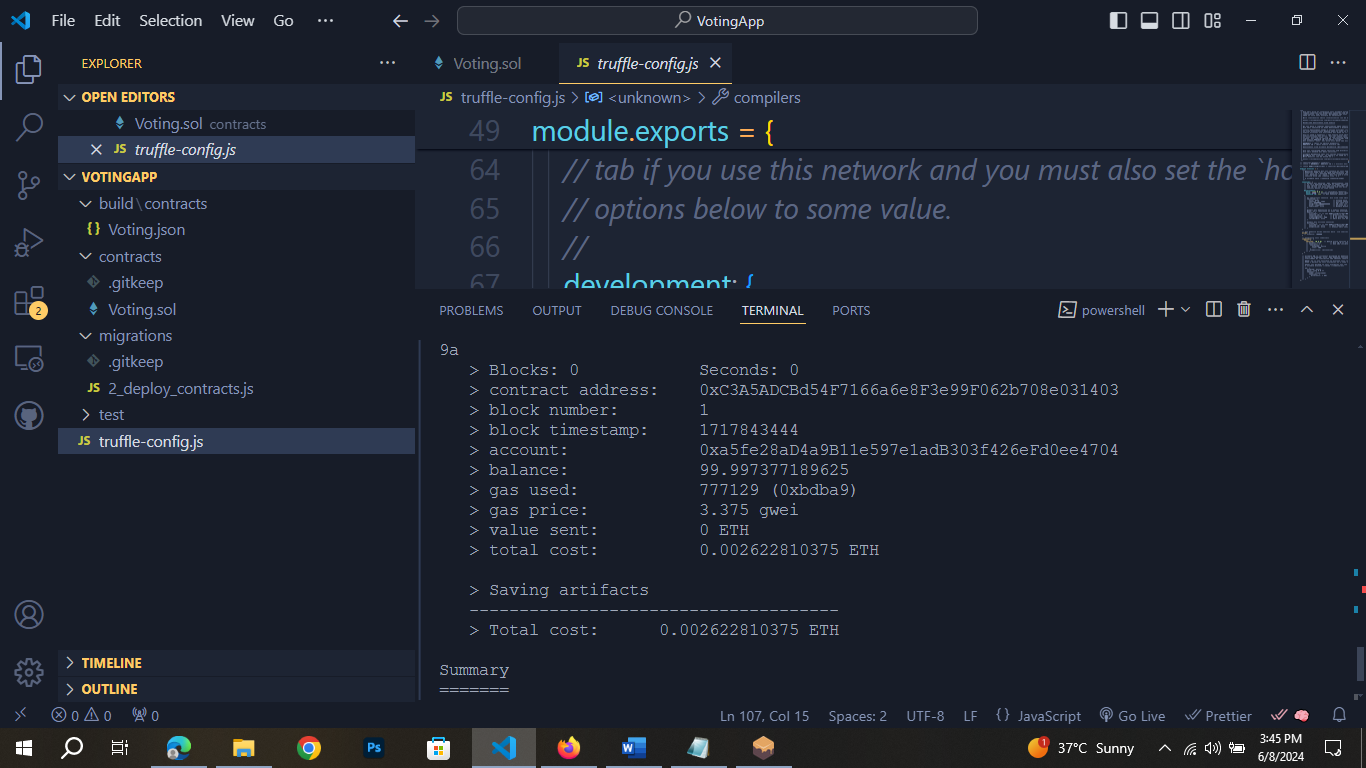
**Deploy Smart Contract**

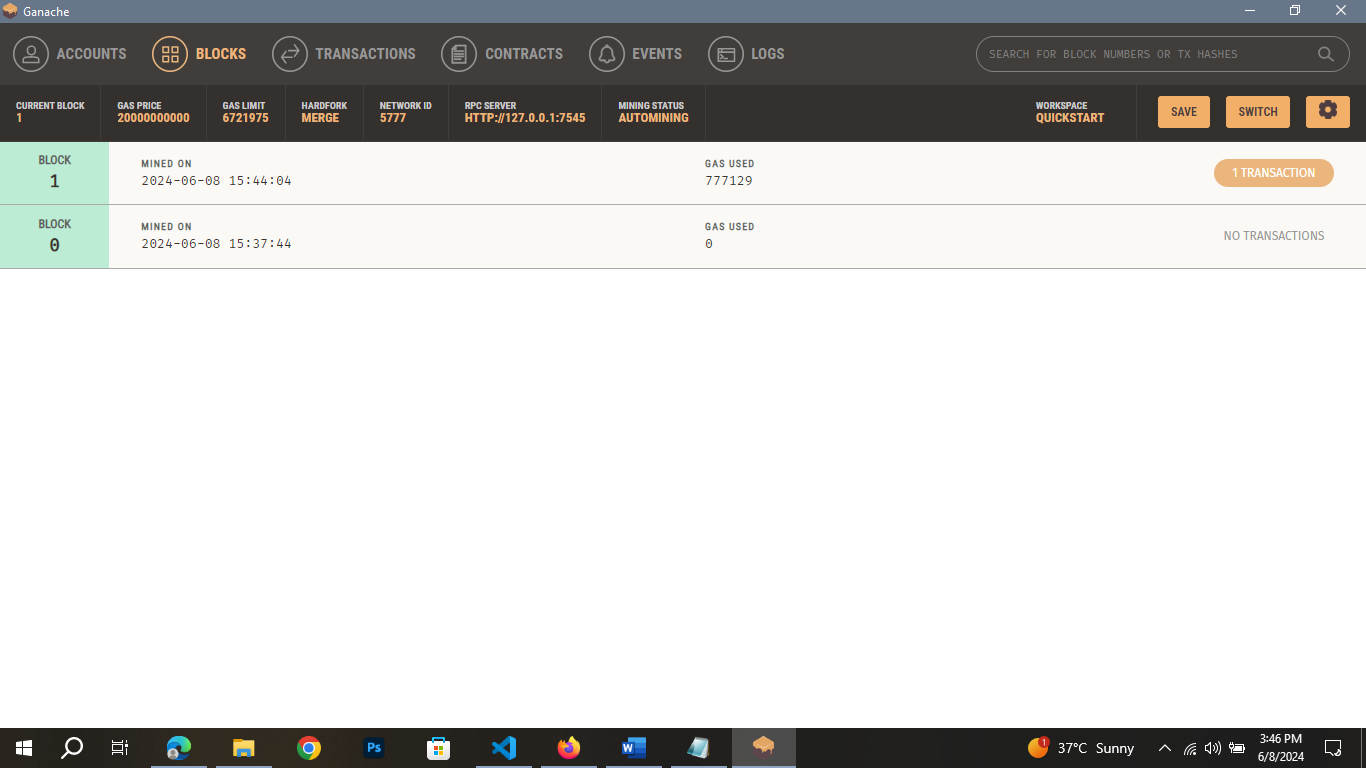
1. For deployment smart contracts, I need to run ganache.
2. And also write a deployment code in the “Migration” directory,
3. Create a new file in the Migration directory name it “2\_deploy\_contracts.js”
4. Connecting “Ganache” to truffle, I need some configuration for this purpose

Go to the file of “truffle-config.js” and scroll down see the option of “network”

Within network you can see the “development” option and set the http, and port   
number.

1. Now deployed the contract with the command of “truffle migrate”. It can run the Migrations directory.

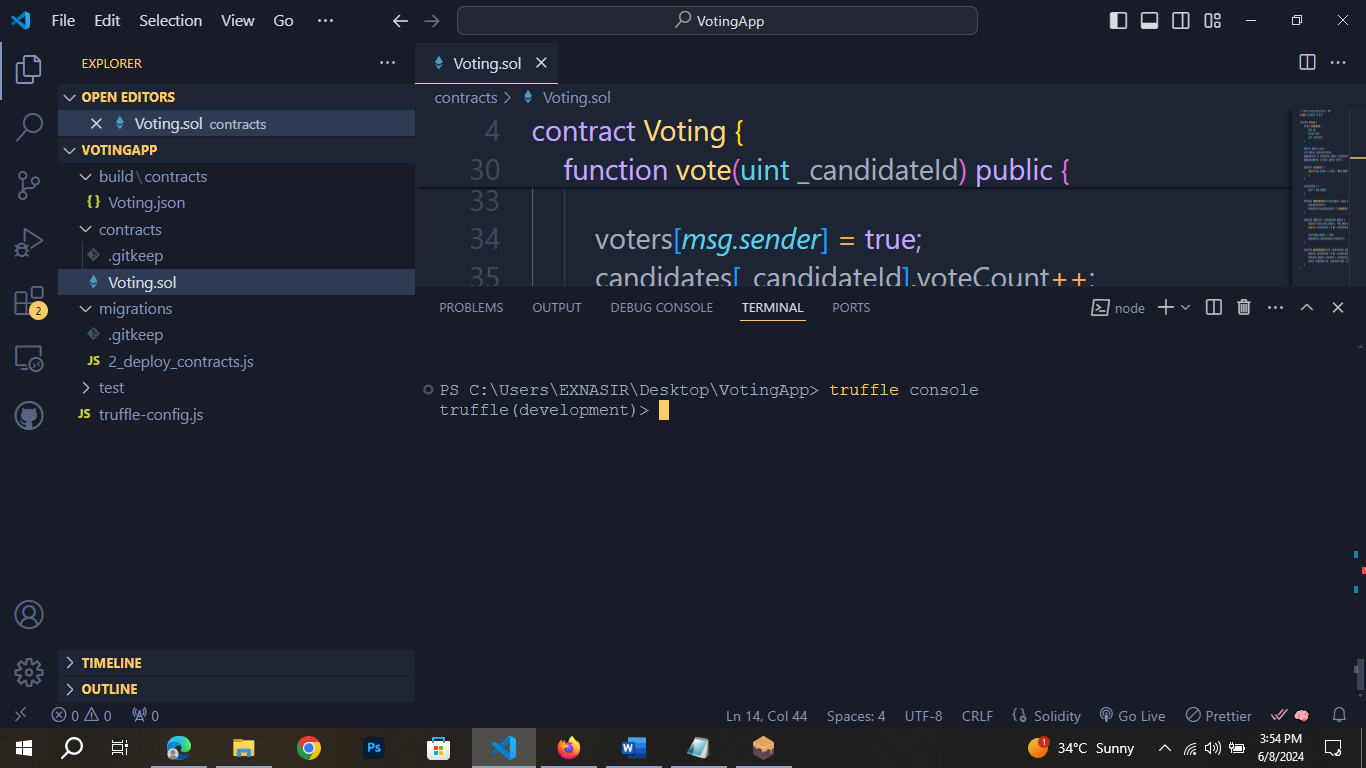
****

****

As you can see It’s working

**Testing Smart Contract**

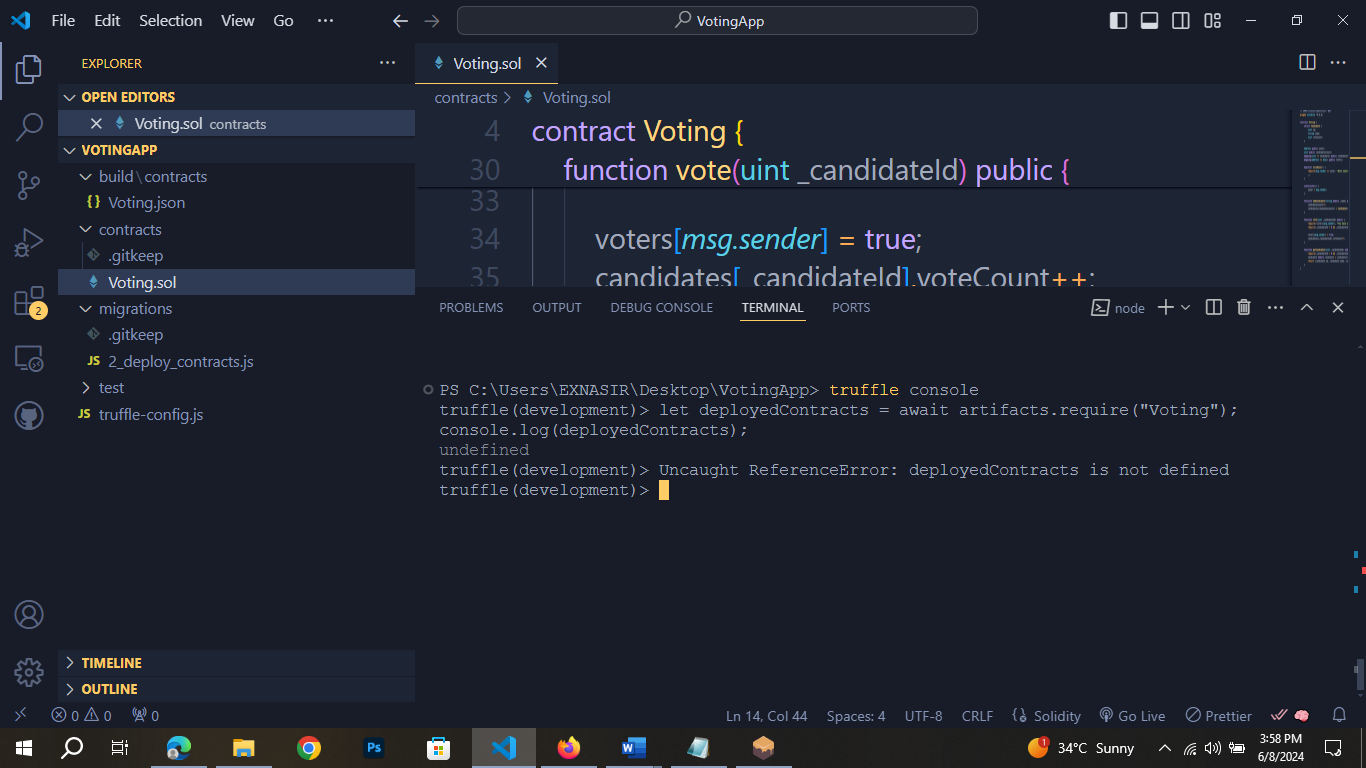
For testing smart contract using the environment of truffle with command of “truffle console”.



1. **Check All Deployed Contract**

I run this line of code in the terminal:  
  
let deployedContracts = await artifacts.require("Voting");

console.log(deployedContracts);

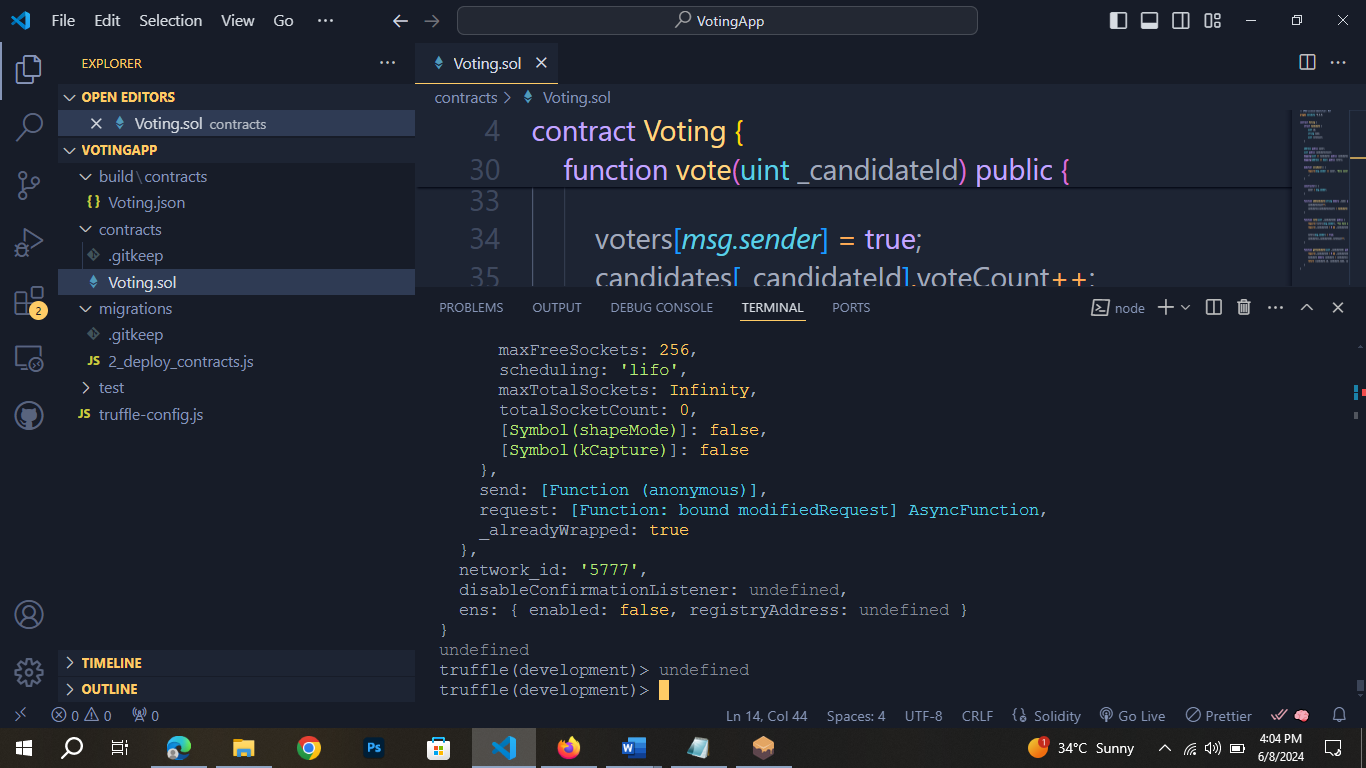


It say “undefined ” but it will work await

1. **Get Deployed Contract Instance**

const Voting = await artifacts.require("Voting").deployed();

console.log(Voting);

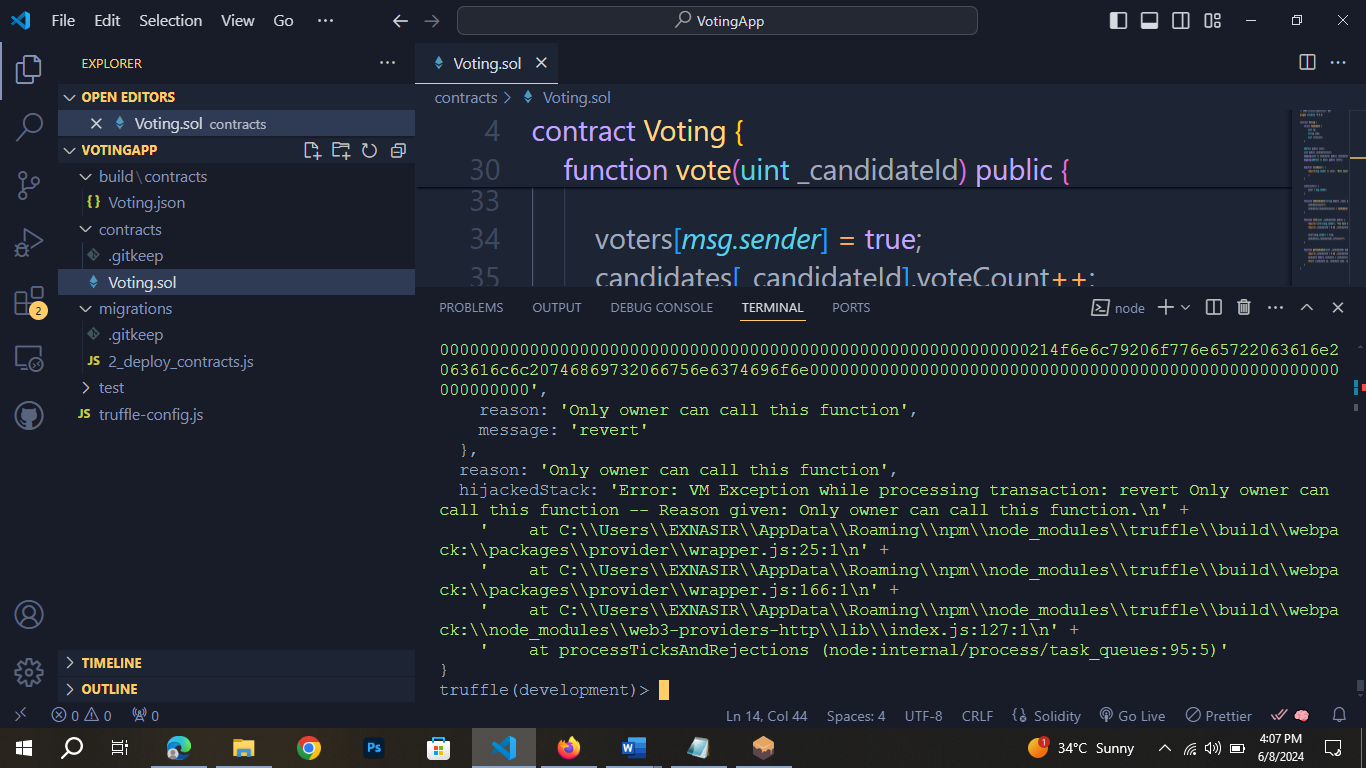


It’s gives me too text but I mansion this text only

1. **Add Candidate**

await Voting.addCandidate("Nasir", {from: web3.eth.accounts[0]});

await Voting.addCandidate("Abbas", {from: web3.eth.accounts[0]})

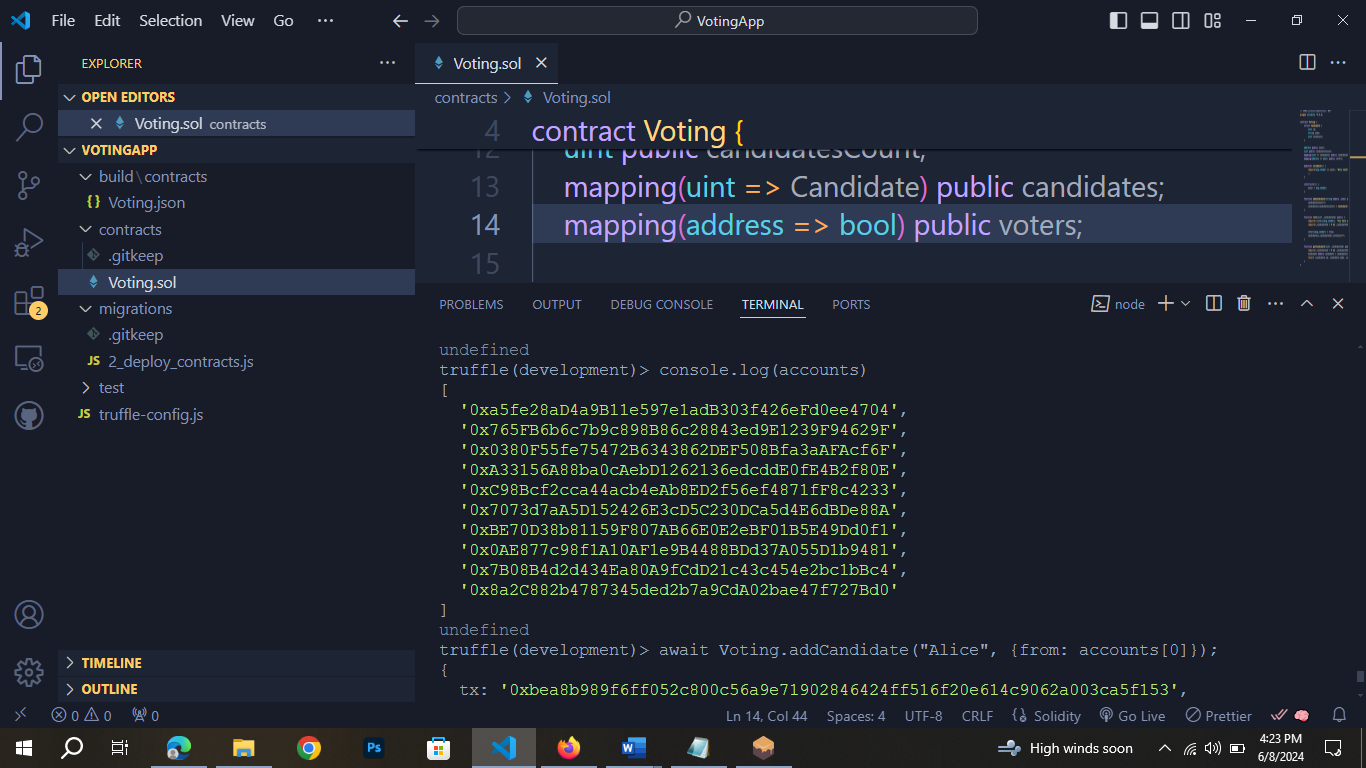


I am not using the owner address, it give me message to cannot add candidate and message is revert.

1. **Let’s see available account**

const accounts = await web3.eth.getAccounts();

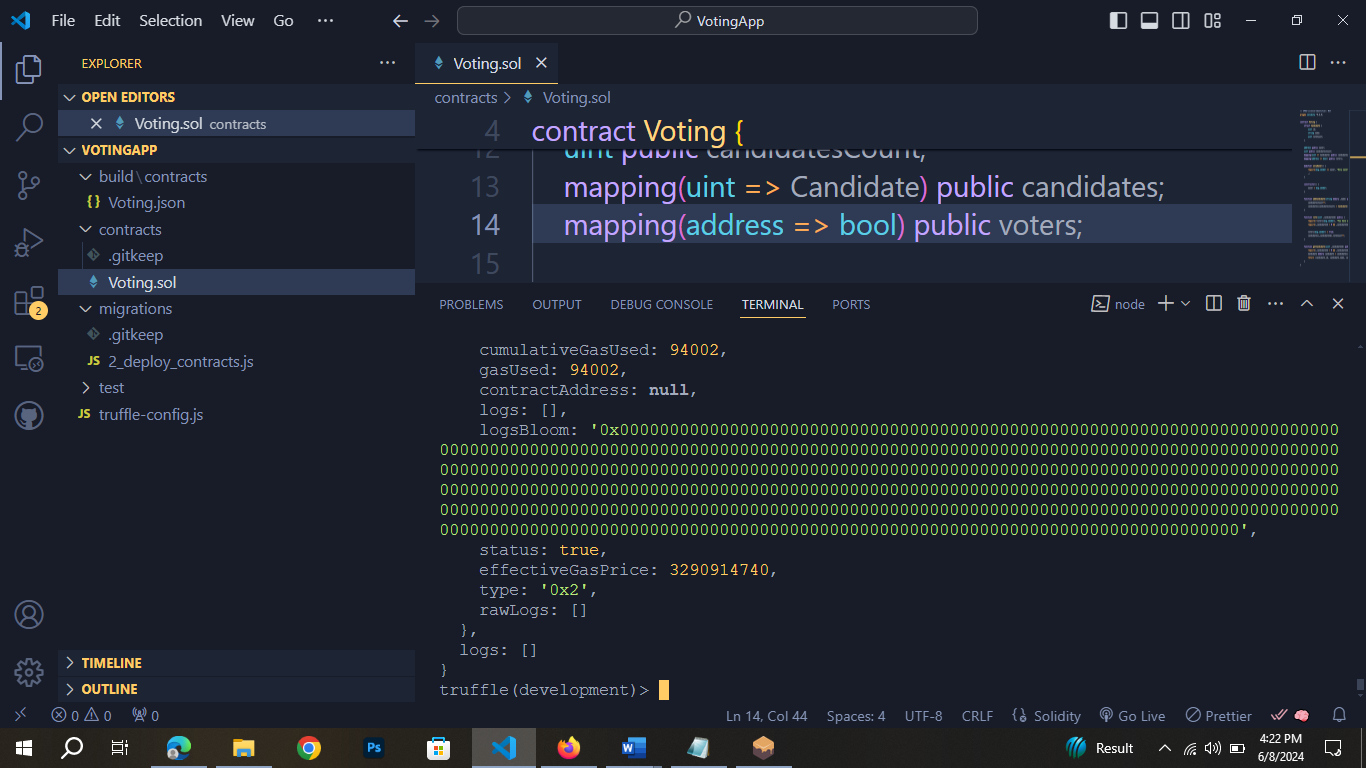
console.log(“Available Accounts: ${accounts}”);



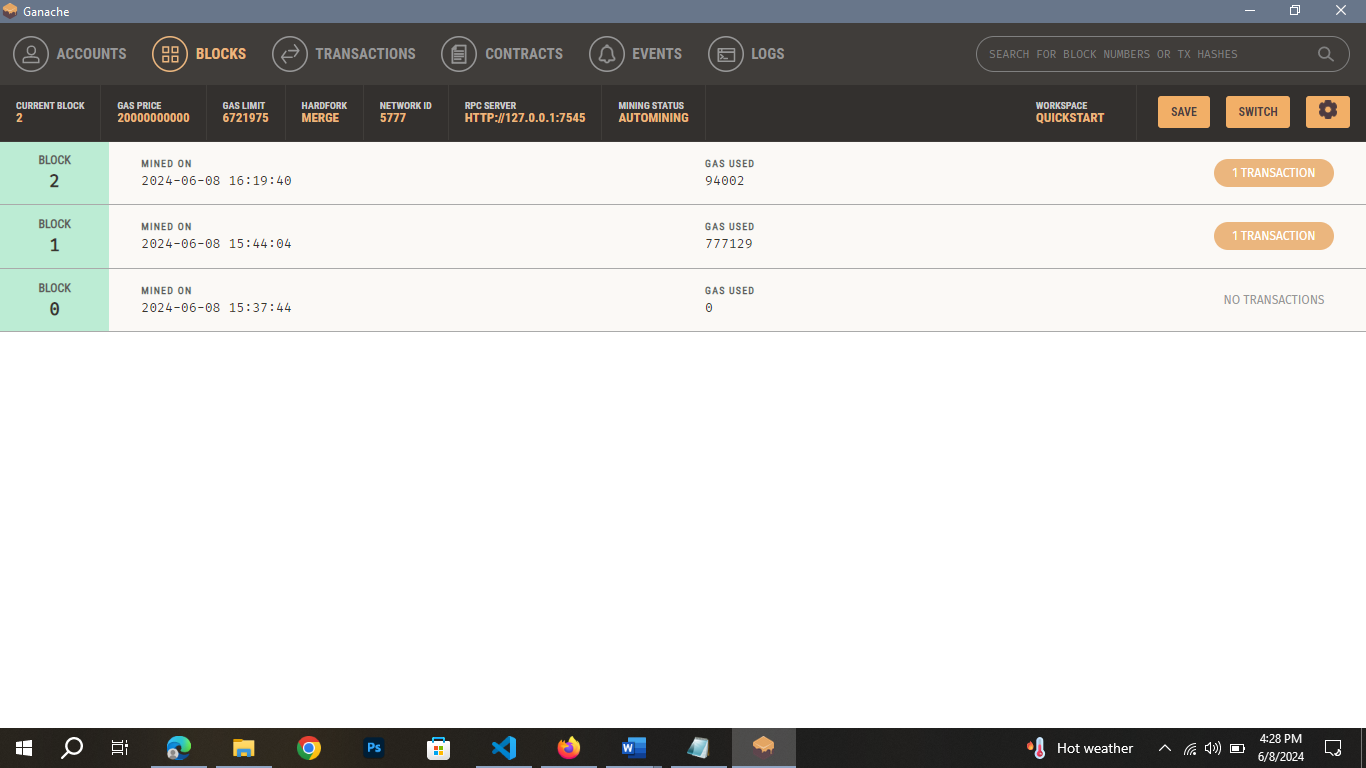
This is Ganache Account addresses.

1. **I am chose the first one is the owner address,**

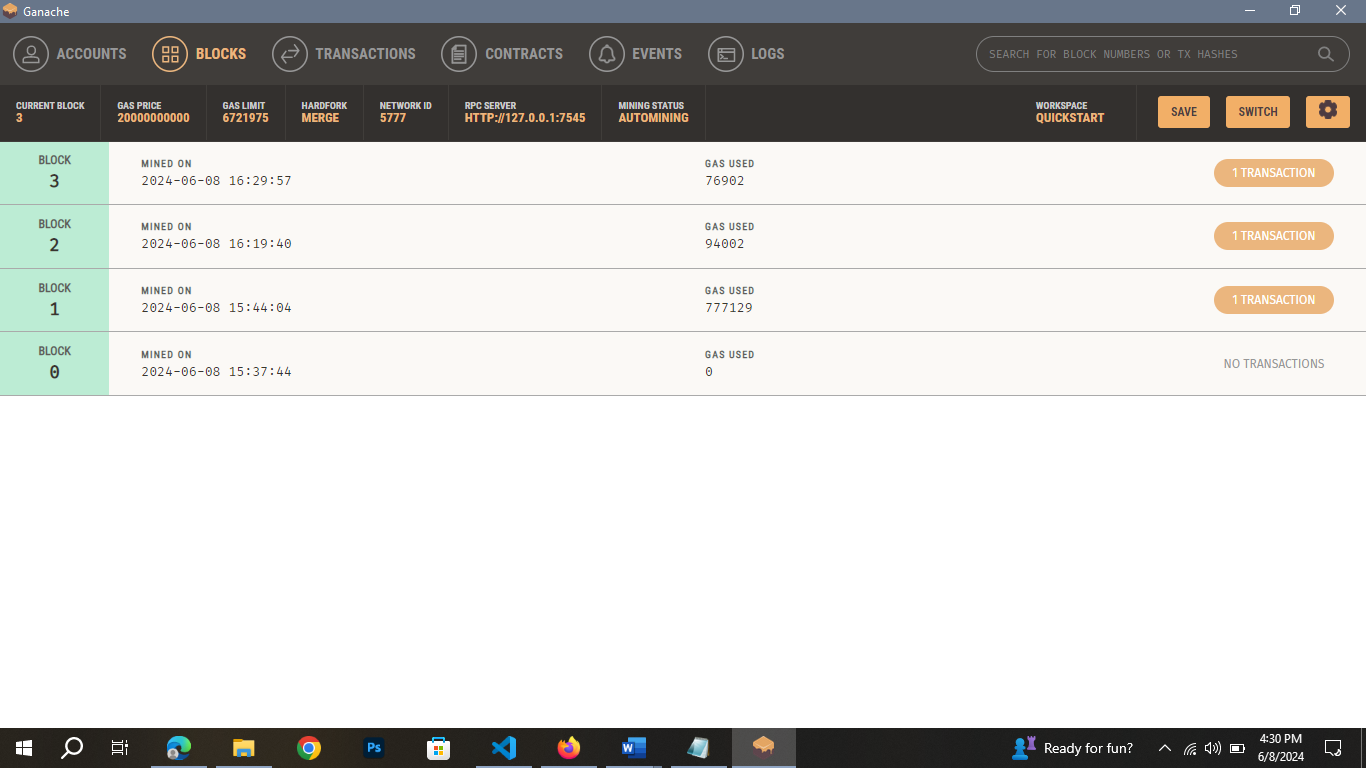
await Voting.addCandidate("Alice", {from: accounts[0]});



Now I can see the Ganache

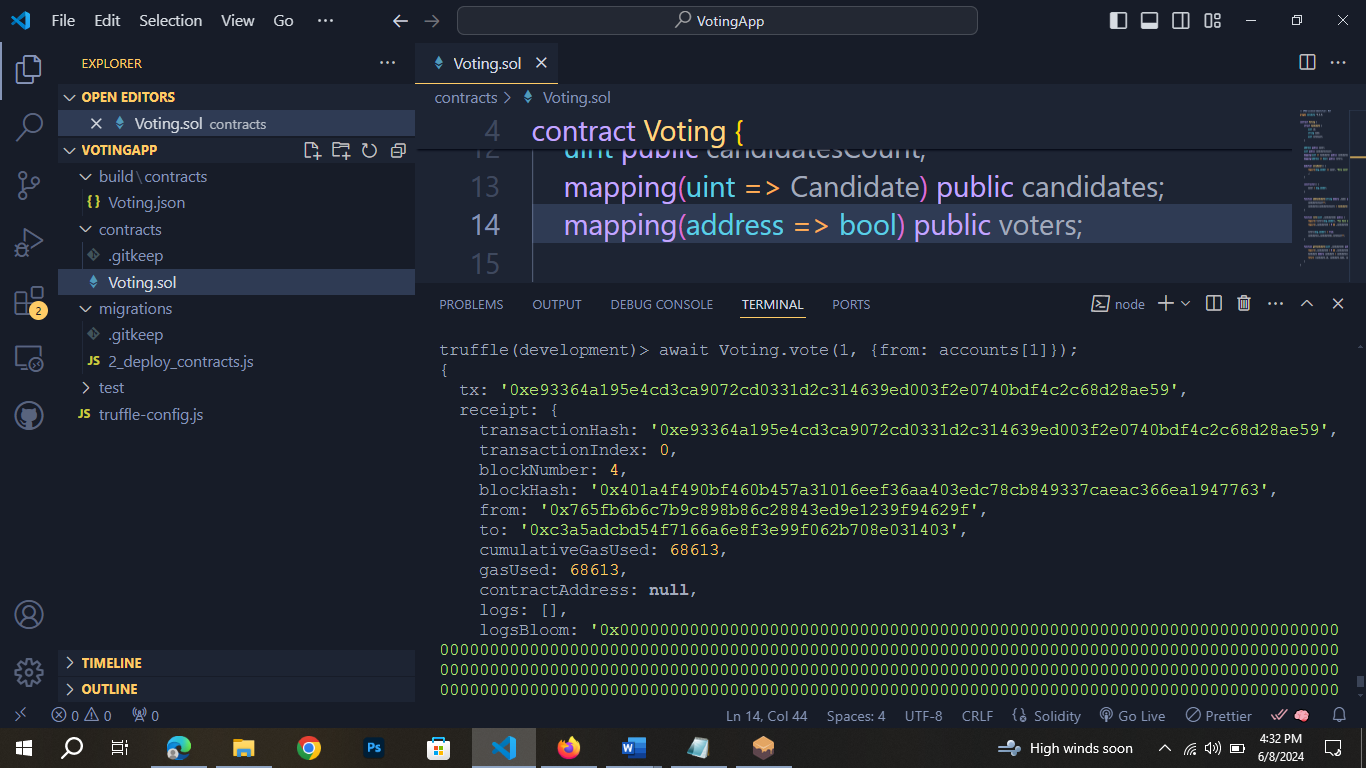


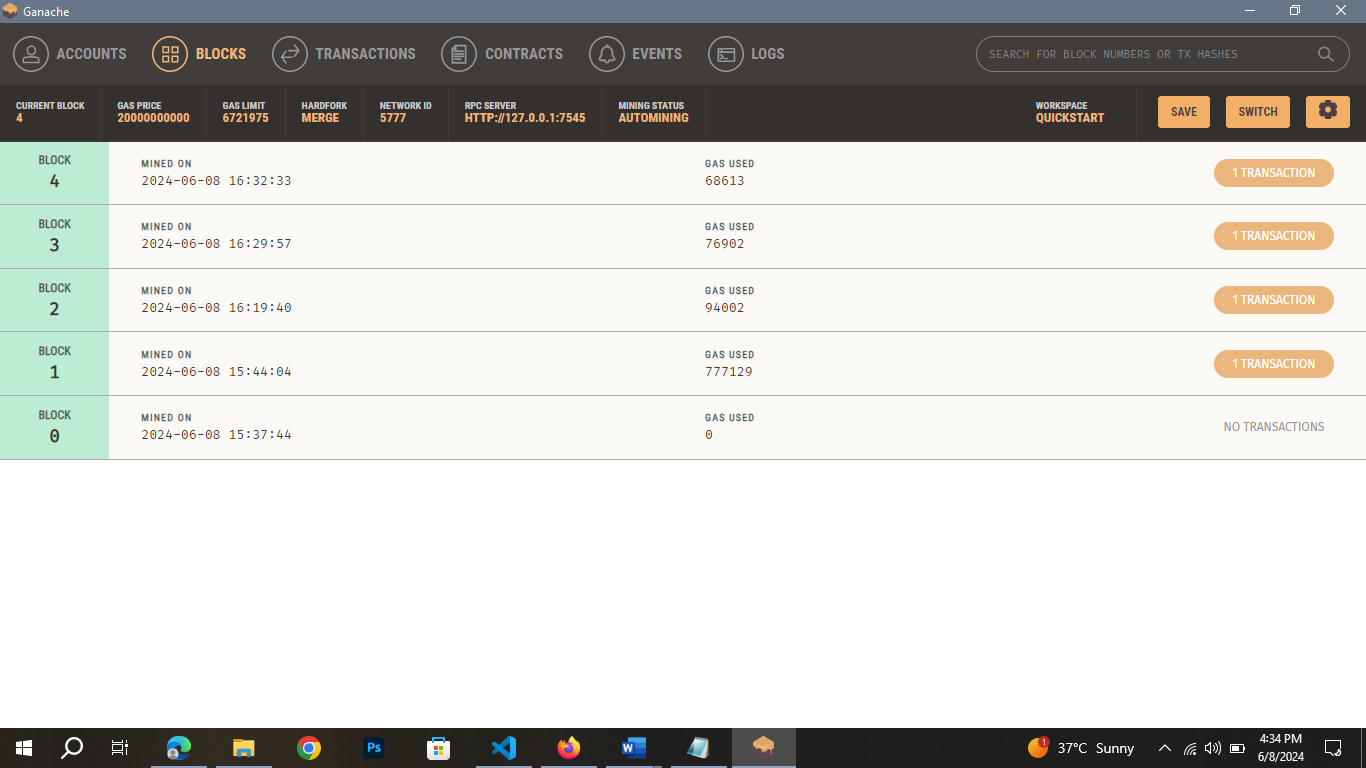
As you can see block are added. I am adding one more candidate.



1. **Votes for Candidates**

await Voting.vote(1, {from: accounts[1]});

****

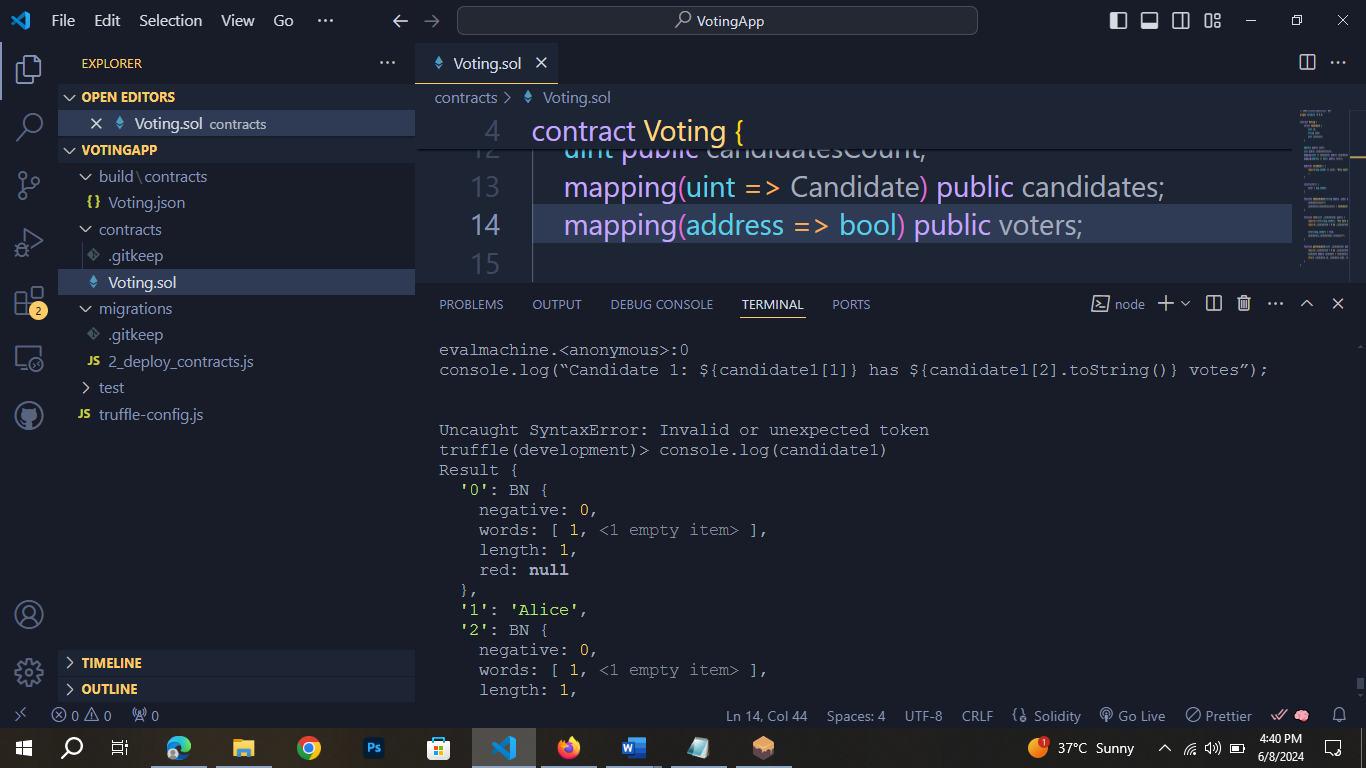
****

I voted.

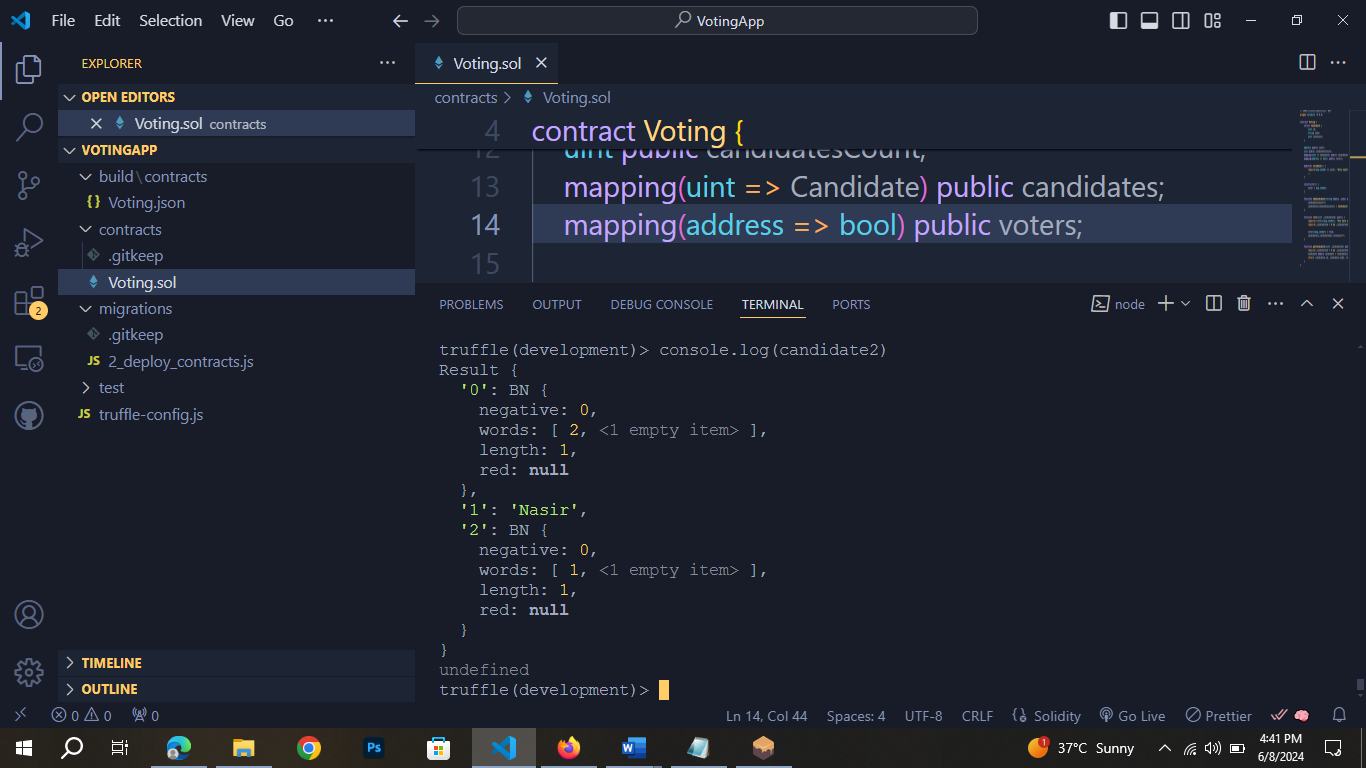
1. **View Results**

let candidate1 = await Voting.getCandidate(1);

console.log(candidate1);



This is second candidate



If I give invalid id it give me error

