

Project Design Phase-I

Solution Architecture

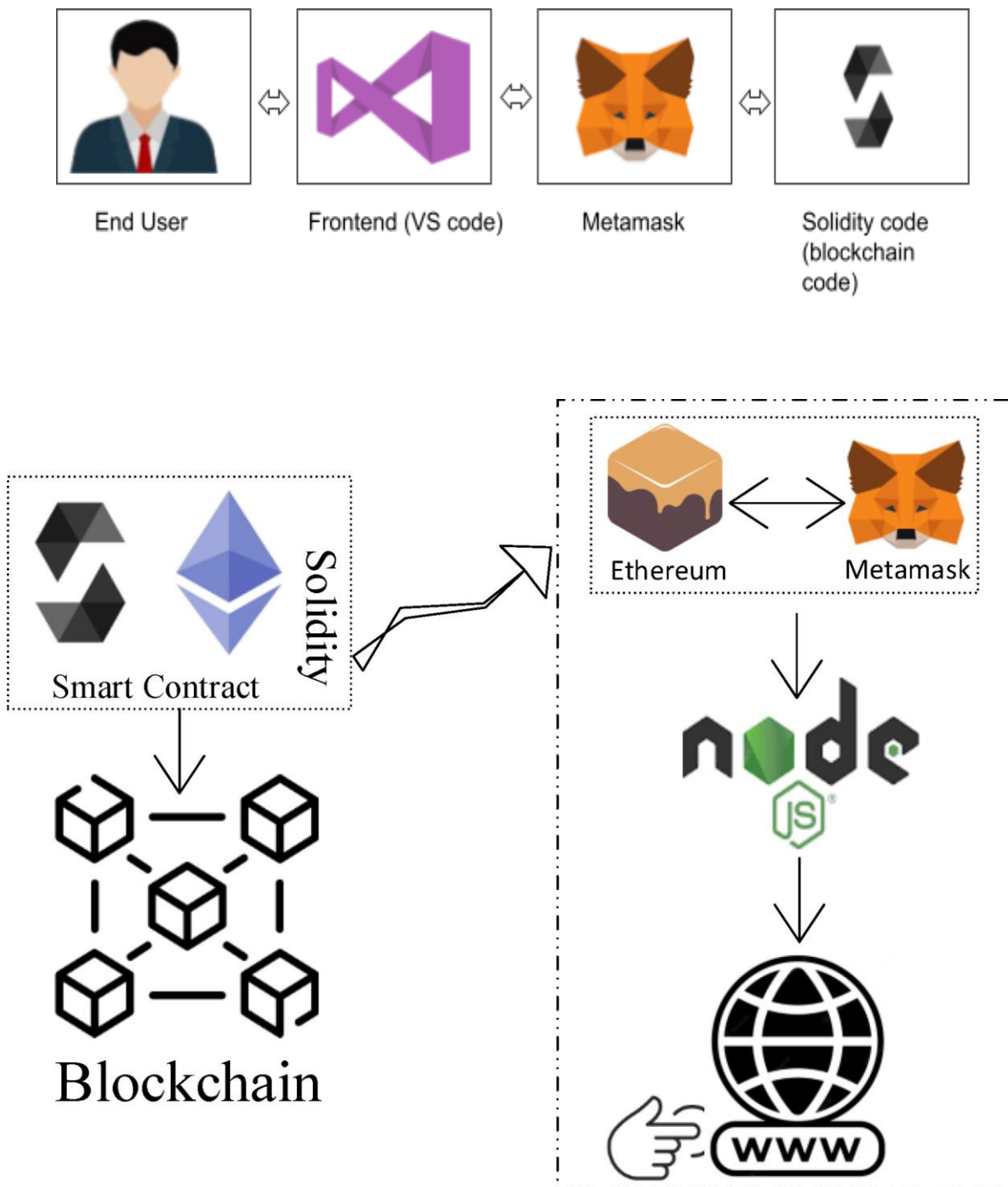
Date	19 October 2023
NM id	D4A1B4223C958ADA97449AF234539CF5
Project Name	Electronic Voting System
Maximum Marks	4 Marks

Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Example - Solution Architecture Diagram:



Prerequisite

1 download node.js : [Node.js](#)

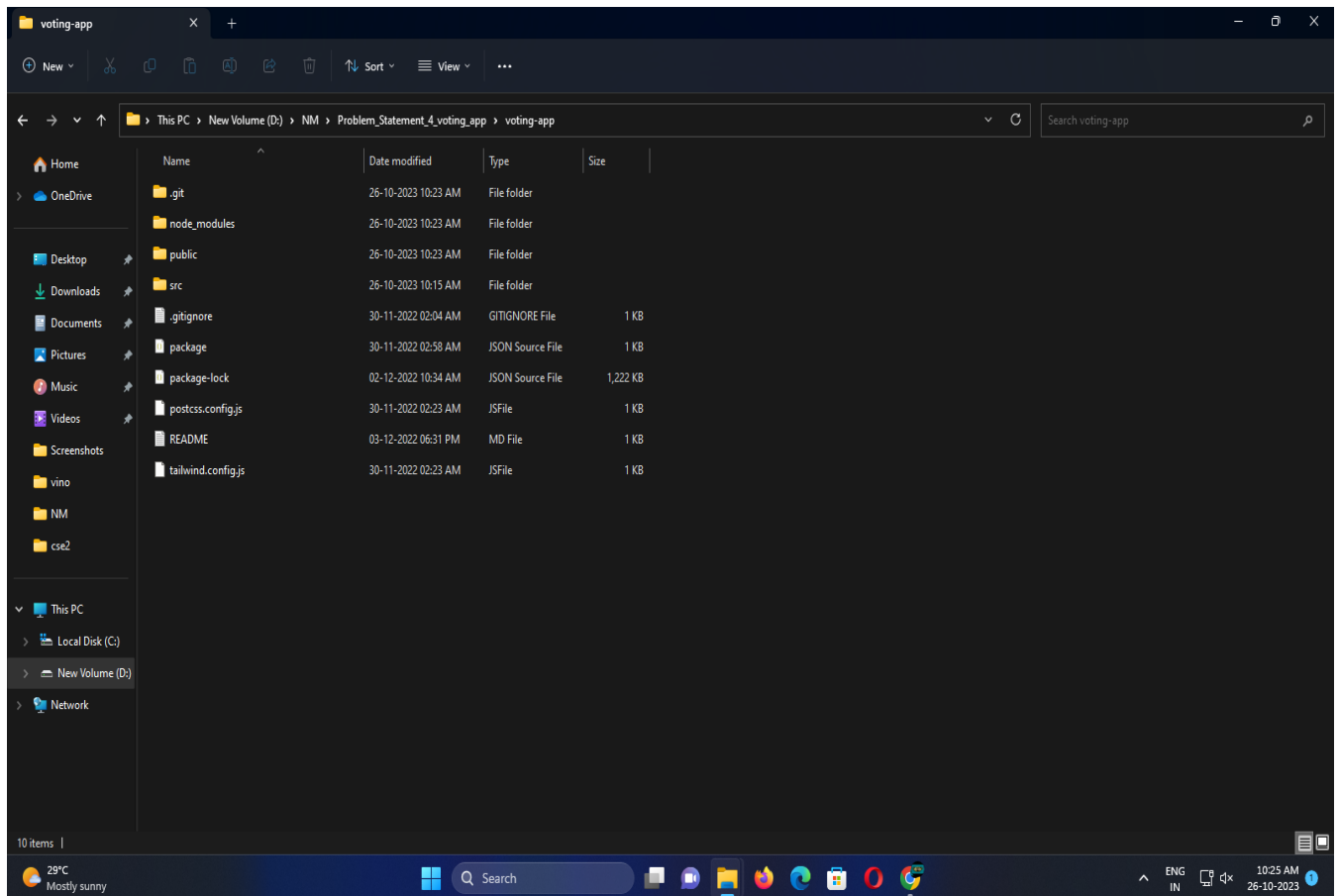
2 download vs code: [Li4nk](#)

3 download metamask : <https://metamask.io/>

Steps to complete the project

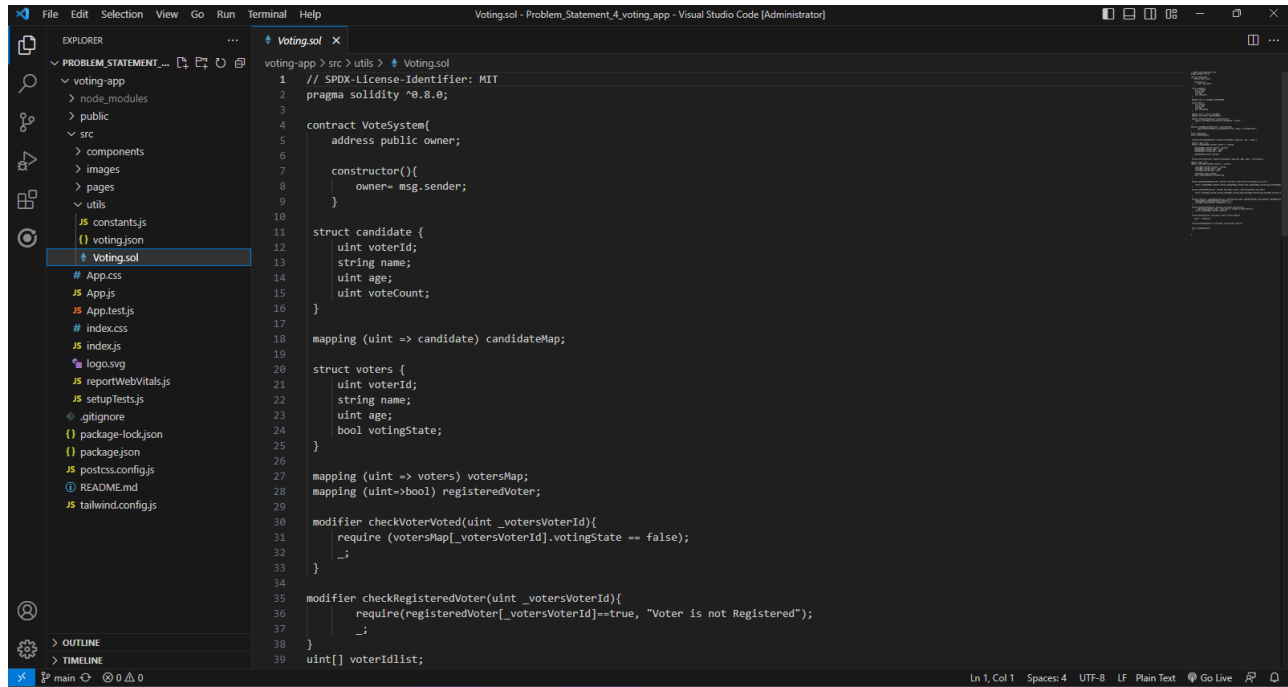
Step 1:-

1. Open the Zip file and download the zip
file.Extract all zip files



Step 2 :

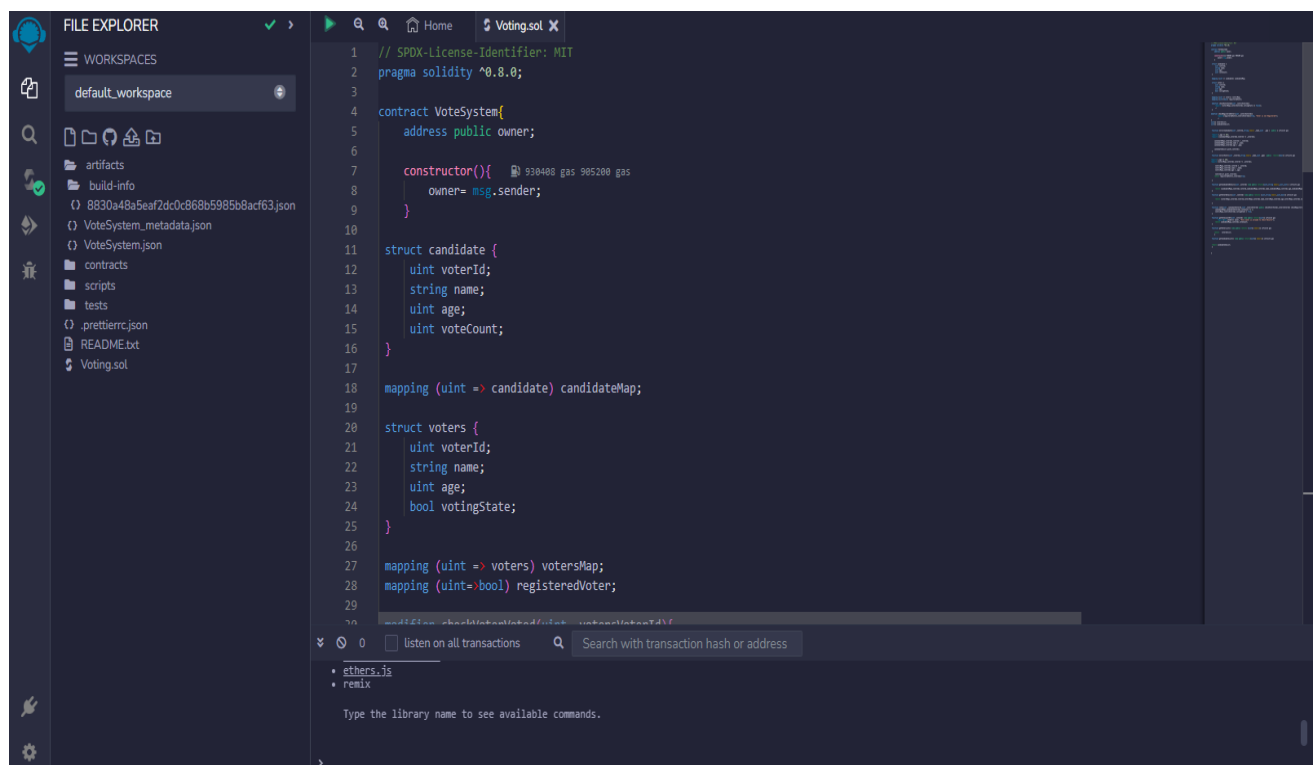
1. Open vs code in the left top select open folder. Select extracted file and open .
2. Select the projectname.sol file and copy the code.



The screenshot shows the Visual Studio Code interface. In the Explorer view on the left, the file 'Voting.sol' is selected under the 'src' directory. The Editor view on the right displays the Solidity code for 'Voting.sol'. The code includes a SPDX license header, a pragma statement for Solidity 0.8.0, and a contract named 'VoteSystem'. The contract has a public owner, a constructor that sets the owner to the sender, and a 'candidate' struct with fields for voterId, name, age, and voteCount. It also includes mappings for 'candidateMap', 'votersMap', and a 'registeredVoter' variable. Two modifier functions, 'checkVoterVoted' and 'checkRegisteredVoter', are defined to enforce voting rules.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract VoteSystem{
5     address public owner;
6
7     constructor(){
8         owner= msg.sender;
9     }
10
11     struct candidate {
12         uint voterId;
13         string name;
14         uint age;
15         uint voteCount;
16     }
17
18     mapping (uint => candidate) candidateMap;
19
20     struct voters {
21         uint voterId;
22         string name;
23         uint age;
24         bool votingState;
25     }
26
27     mapping (uint => voters) votersMap;
28     mapping (uint=>bool) registeredVoter;
29
30     modifier checkVoterVoted(uint _votersVoterId){
31         require (votersMap[_votersVoterId].votingState == false);
32         _;
33     }
34
35     modifier checkRegisteredVoter(uint _votersVoterId){
36         require(registeredVoter[_votersVoterId]==true, "Voter is not Registered");
37         _;
38     }
39     uint[] voterIdList;
```

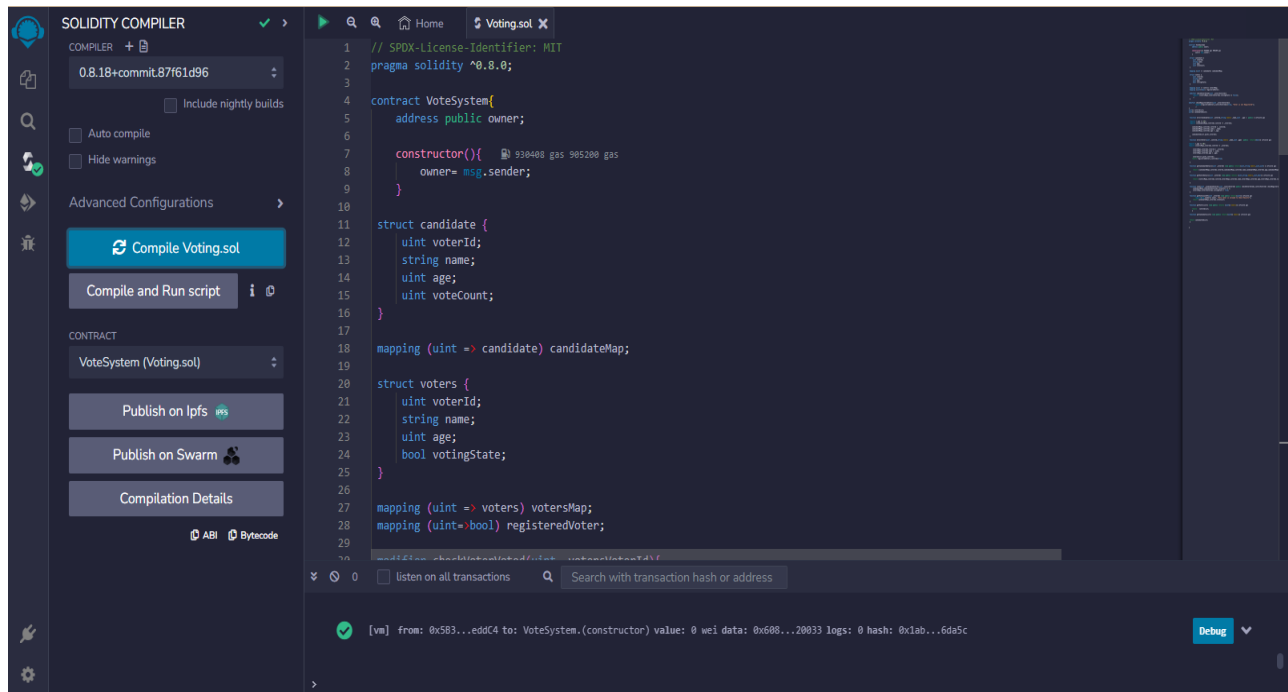
3. Open the remix ide platform and create a new file by giving the name of voting.sol and paste the code which you copied from vs code.



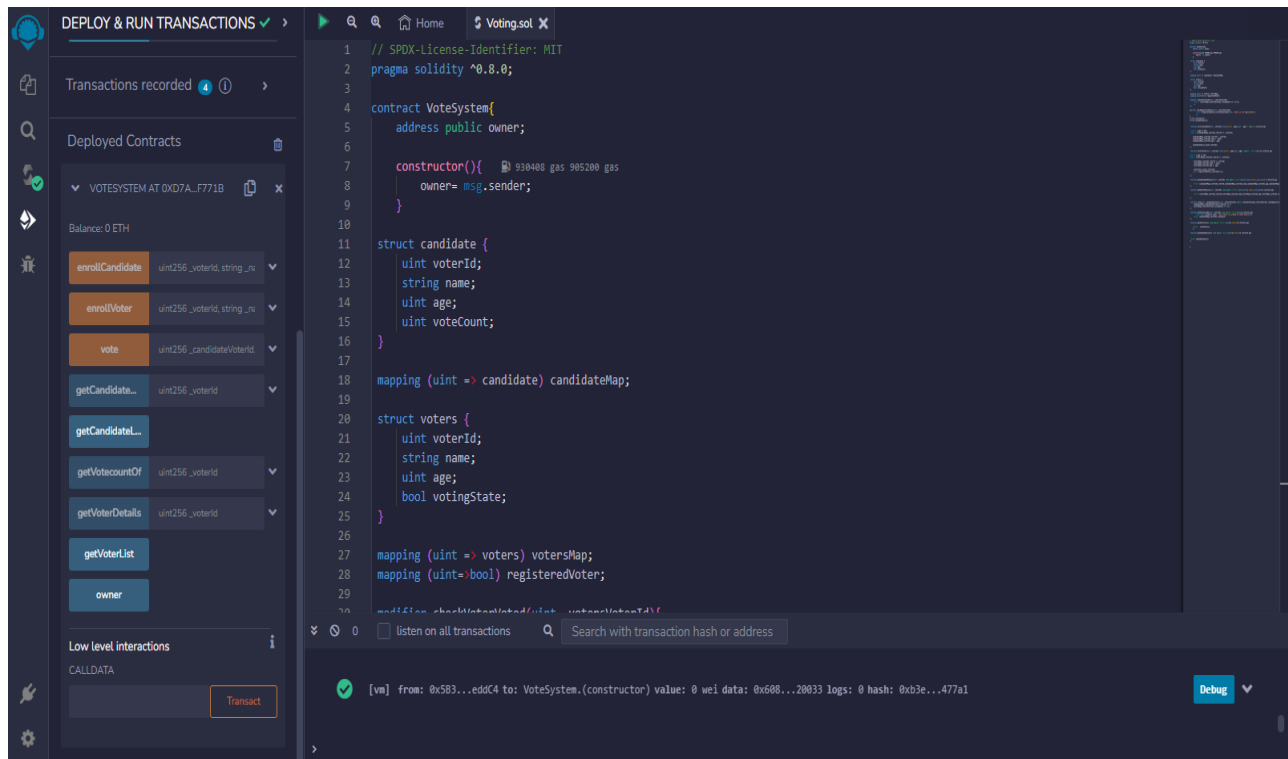
The screenshot shows the Remix IDE interface. In the File Explorer on the left, the file 'Voting.sol' is listed under the 'contracts' directory. The editor on the right displays the Solidity code for 'Voting.sol', which is identical to the code shown in the previous screenshot. The code includes a SPDX license header, a pragma statement for Solidity 0.8.0, and a contract named 'VoteSystem'. The contract has a public owner, a constructor that sets the owner to the sender, and a 'candidate' struct with fields for voterId, name, age, and voteCount. It also includes mappings for 'candidateMap', 'votersMap', and a 'registeredVoter' variable. Two modifier functions, 'checkVoterVoted' and 'checkRegisteredVoter', are defined to enforce voting rules.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract VoteSystem{
5     address public owner;
6
7     constructor(){
8         owner= msg.sender;
9     }
10
11     struct candidate {
12         uint voterId;
13         string name;
14         uint age;
15         uint voteCount;
16     }
17
18     mapping (uint => candidate) candidateMap;
19
20     struct voters {
21         uint voterId;
22         string name;
23         uint age;
24         bool votingState;
25     }
26
27     mapping (uint => voters) votersMap;
28     mapping (uint=>bool) registeredVoter;
29
30     modifier checkVoterVoted(uint _votersVoterId){
31         require (votersMap[_votersVoterId].votingState == false);
32         _;
33     }
34
35     modifier checkRegisteredVoter(uint _votersVoterId){
36         require(registeredVoter[_votersVoterId]==true, "Voter is not Registered");
37         _;
38     }
39     uint[] voterIdList;
```

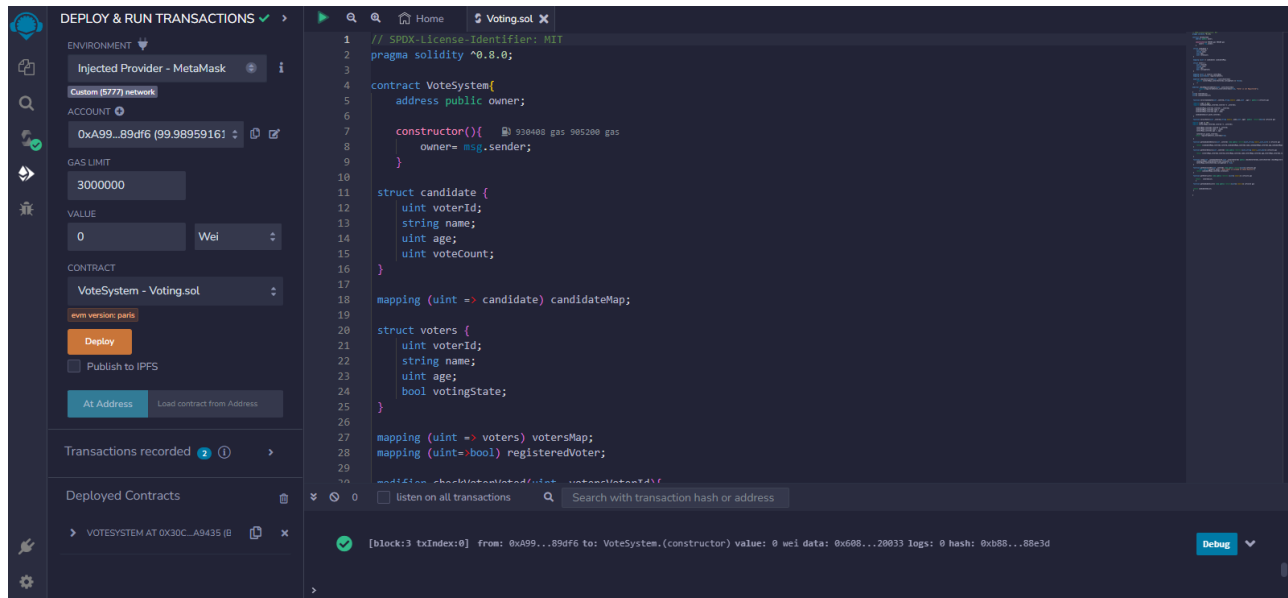
4. Click on solidity compiler and click compile the voting.sol



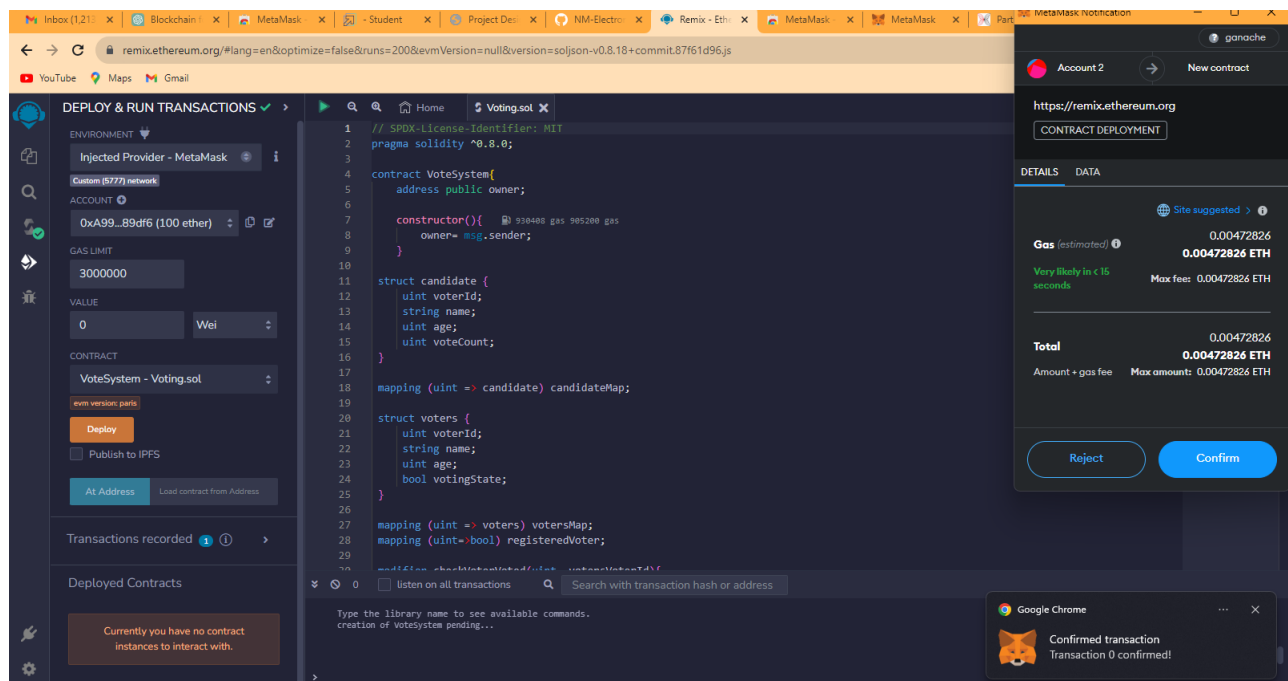
5. Deploy the smart contract by clicking on the deploy and run transaction.



6. select injected provider - MetaMask. In environment



7. Click on deploy. Automatically MetaMask will open and give confirmation. You will get a pop up click on ok.

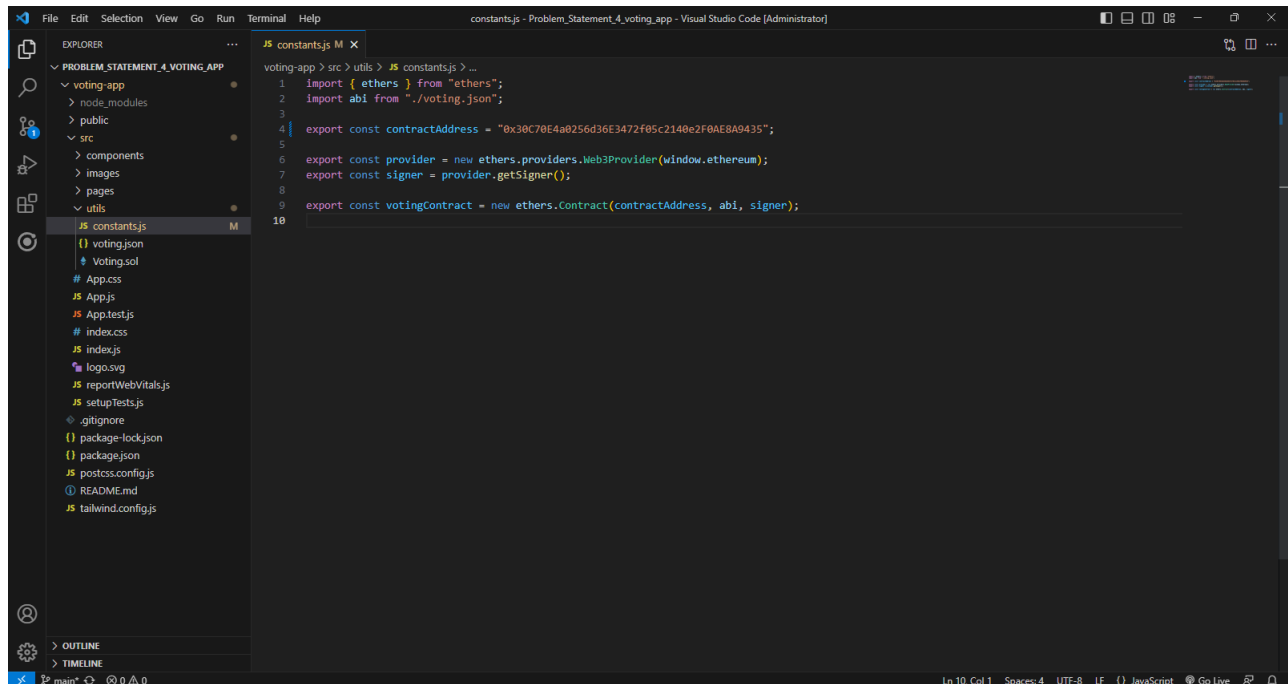


8. In the Deployed contract you can see one address copy the address.

0x30C70E4a0256d36E3472f05c2140e2F0AE8A9435

9. Open vs code and search for the connector.js. In contract.js you can paste the address at the bottom of the code. In export const address.

10. Save the code.



Step 3:

open file explorer

1. Open the extracted file and click on the folder.

2. Open src, and search for utiles.

3 . You can see the frontend files. Select all the things at the top in the search bar by clicking alt+ A. Search for cmd

4. Open cmd enter commands

npm install

npm bootstrap

npm start

```
Administrator: Windows Powe
D:\NM\Problem_Statement_4_voting_app\voting-app\src\utils>npm start

> voting-app@0.1.0 start
> react-scripts start

Browserslist: caniuse-lite is outdated. Please run:
  npx update-browserslist-db@latest
  Why you should do it regularly: https://github.com/browserslist/update-db#readme
(node:8908) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is de
precated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:8908) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is
deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view voting-app in the browser.

Local:      http://localhost:3000
On Your Network:  http://10.10.5.244:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
|
```

5. It will install all the packages and after completing it will open {LOCALHOST IP ADDRESS} copy the address and open it to chrome so you can see the frontend of yourproject.

