1. Create a folder called Dapp.
2. Inside it, open terminal, create a react app using the command **npx create-react-app blockchain_dapp**
3. Create a folder called components in the src directory. Inside it create a file called Greeting.js.

   Code for Greeting.js:

```
import { useState, useEffect } from "react";
import greetings from "../abi/greetings.json";
import Web3 from "web3";
export const Greetings = () => {
  const ethereum = window.ethereum;
  let web3 = window.web3;
  const [account, setaccount] = useState("");
  const [netId, setNetId] = useState();
  const [contract, setContract] = useState(null);
  let currentGreeting="";
  useEffect(() => {
    async function loadBlockchainData() {
      const accounts = await web3.eth.getAccounts();
      setaccount(accounts[0]);
    }

    async function loadWeb3() {
      if (ethereum) {
        web3 = new Web3(ethereum);
        await ethereum.enable();
        const x = await web3.eth.net.getId();
        setNetId(x);
        const networkData = greetings.networks[x];
        const _contract = new web3.eth.Contract(
          greetings.abi,
          networkData.address
        );
        setContract(_contract);
      } else if (web3) {
        window.web3 = new Web3(window.web3.currentProvider);
      } else {
        window.alert("Please use metamask");
      }
    }
    loadWeb3();
    loadBlockchainData();
  }, []);

    const [greeting,setGreeting] = useState("");

    const onChangeGreeting = (event) =>{
        setGreeting(event.target.value);
    }

    const handleSetGreeting = async(event) =>{
        event.preventDefault();
        if (web3 && contract && greeting) {
          try {
            await contract.methods.setGreeting(greeting).send({
from: account});

          } catch (error) {
            console.error('Error setting greeting:', error);
          }
        }
        else if(!greeting){
          alert("Enter greeting");
          window.location.reload();
        }
        else{
          alert("Error occoured !")
        }

    }

    const handleFetchGreeting = async(event) =>{
      event.preventDefault();
        if (web3 && contract) {
          try {
            currentGreeting = await
contract.methods.fetchGreeting().call();
            if(currentGreeting!="")
            {
```

```
document.getElementById("displayGreeting").innerHTML=`Current
Greeting: ${currentGreeting}`;
            }
            else{

document.getElementById("displayGreeting").innerHTML="Set Greeting
to view it !";
            }


          } catch (error) {
            console.error('Error setting greeting:', error);
          }
        }
        else{
          alert("Error occoured !")
        }
    }

  return (
    <>
      <div style={{display: "flex",flexDirection:
"column",justifyContent: "center",alignItems: "center",width:
"100vw",height: "100vh"}}>
        <p>Hello, Enter a Greeting</p>
        <div className="setGreetingContainer"
style={{display:"flex",flexDirection:"column"}}>
            <input name="greeting" onChange={onChangeGreeting}
placeholder="Enter Greeting"></input>
            <button onClick={handleSetGreeting}>Set
Greeting</button>
        </div>
        <div className="fetchGreetingContainer">
          <button onClick={handleFetchGreeting}>Fetch
Greeting</button>
        </div>
        <div id="displayGreeting"></div>
      </div>
    </>
  );
};
```

4. Modify App.js file as follows:

```
import { Greetings } from "./components/Greeting";
import React, { useEffect, useState } from "react";
import greetings from "./abi/greetings.json";
import Web3 from "web3";
function App() {

  return (
    <>
      <Greetings />
    </>
  );
}

export default App;
```

5. Create a new directory inside Greeting_Dapp folder called truffle_project.
6. Open terminal, type **cd truffle_project** .
7. Now type **truffle init**.
8. Inside the contracts folder create a new contract called greeting.sol
9. Code for greeting.sol:

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract greetings{
    string greeting;

    function setGreeting(string memory _greeting) public{
        greeting = _greeting;
    }
    function fetchGreeting() public view returns(string memory){
        return greeting;
    }
}
```

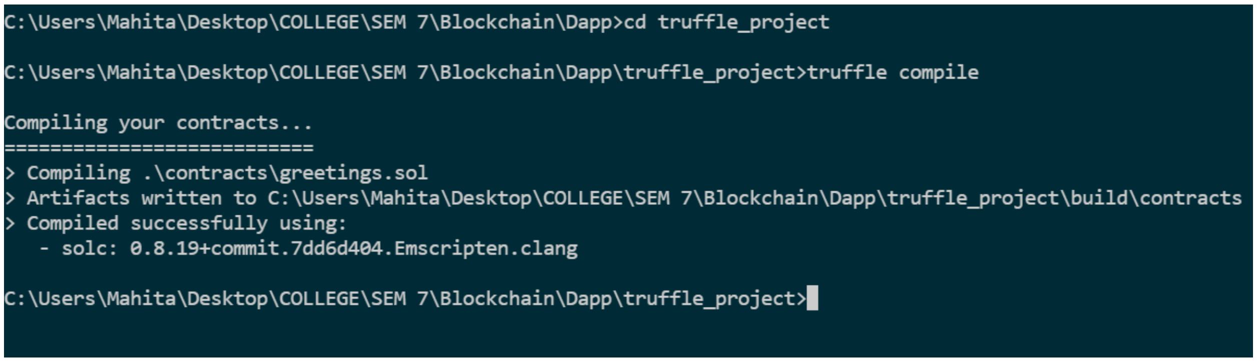
10. Inside migrations folder, create a new file called 1_deploy_greetings.js
Code for 1_deploy_greetings.js:

```
const Greetings = artifacts.require("greetings");

module.exports = function (deployer) {
  deployer.deploy(Greetings);
};
```

11. In the truffle-config.js file, uncomment the following lines and modify them as follows:

```
networks: {
    // Useful for testing. The `development` name is special - truffle uses it by default
    // if it's defined here and no other network is specified at the command line.
    // You should run a client (like ganache, geth, or parity) in a separate terminal
    // tab if you use this network and you must also set the `host`, `port` and `network_id`
    // options below to some value.
    //
      development: {
      host: "127.0.0.1",     // Localhost (default: none)
      port: 7545,            // Standard Ethereum port (default: none)
      network_id: "*",       // Any network (default: none)
     },
```

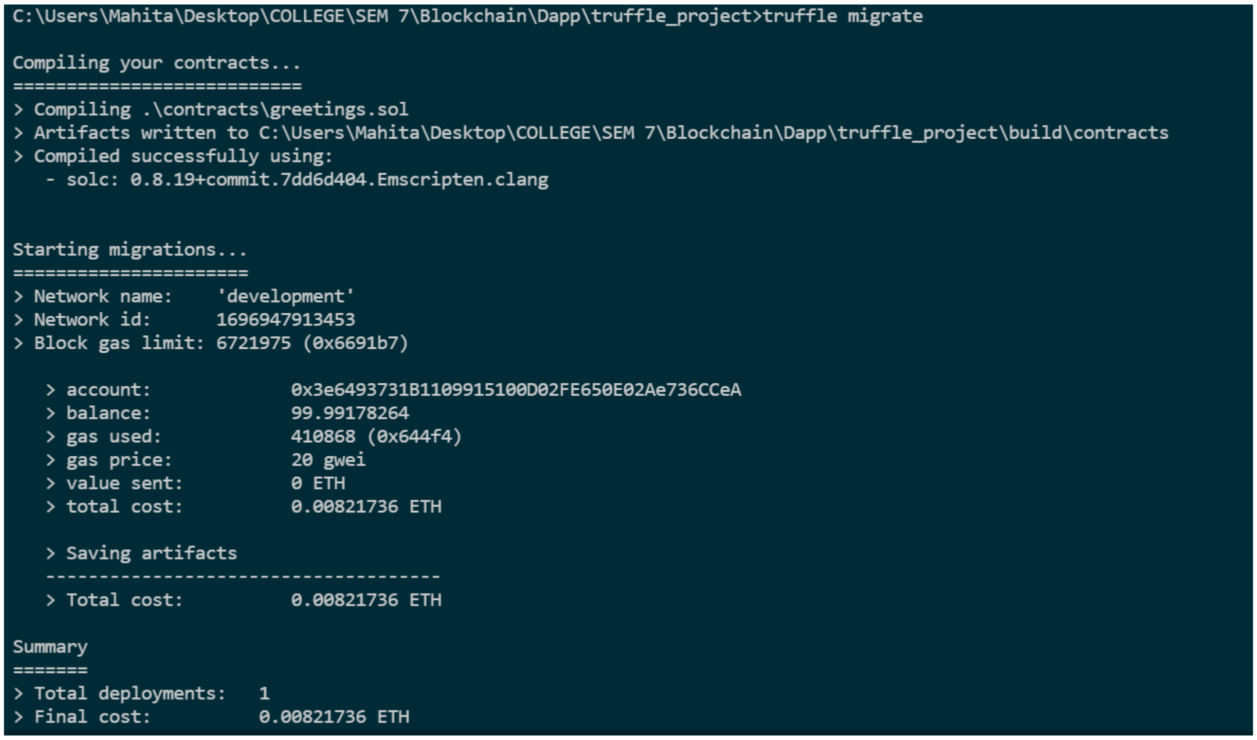
12. In terminal navigate to truffle project directory, type **truffle compile.** The contract should be compiled.



```
C:\Users\Mahita\Desktop\COLLEGE\SEM 7\Blockchain\Dapp>cd truffle_project

C:\Users\Mahita\Desktop\COLLEGE\SEM 7\Blockchain\Dapp\truffle_project>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\greetings.sol
> Artifacts written to C:\Users\Mahita\Desktop\COLLEGE\SEM 7\Blockchain\Dapp\truffle_project\build\contracts
> Compiled successfully using:
   - solc: 0.8.19+commit.7dd6d404.Emscripten.clang

C:\Users\Mahita\Desktop\COLLEGE\SEM 7\Blockchain\Dapp\truffle_project>
```

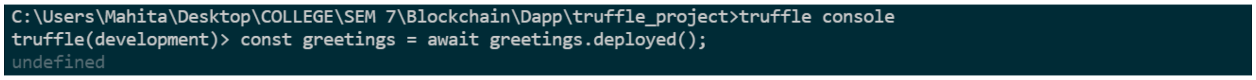
13. Open ganache, in settings add the truffle-config.js file and select the option to save and restart.

14. Enter the following command in terminal, **truffle migrate**. The contract will be deployed to the blockchain and in the ganache gui contracts tab we can see the deployed contract.
15. Inside the truffle_projet/build folder copy the greetings.json file. Create a folder in blockchain_dapp/src called abi. Paste the greetings.json file in the abi folder.

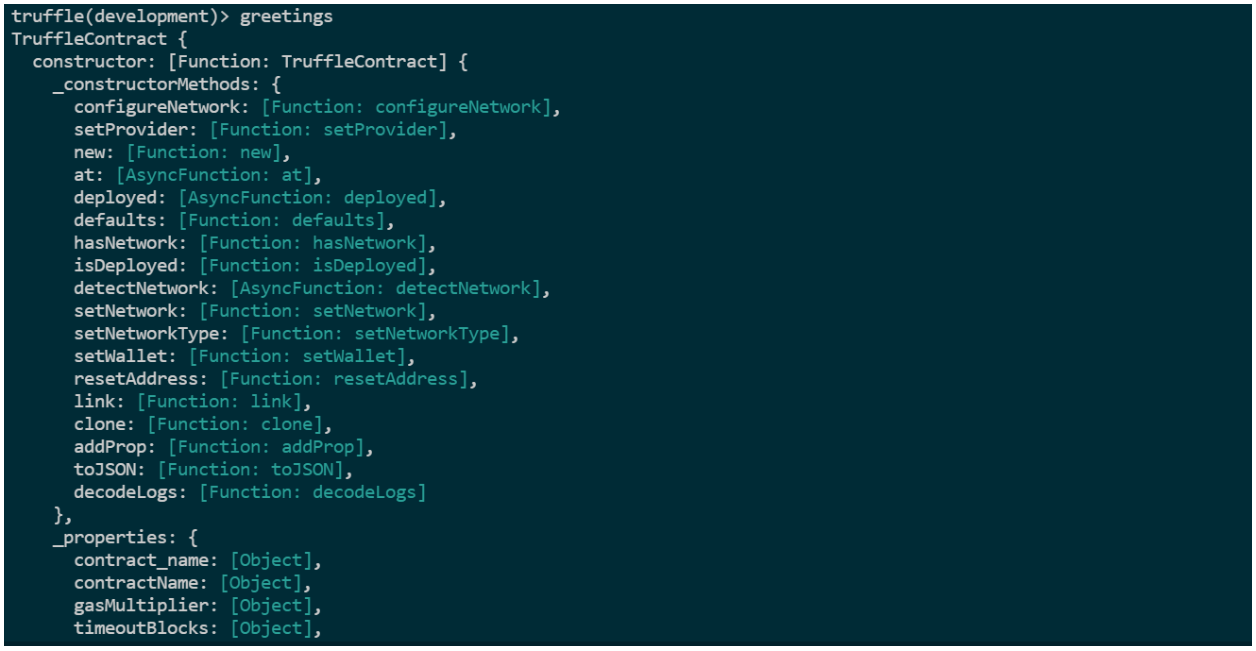


```
C:\Users\Mahita\Desktop\COLLEGE\SEM 7\Blockchain\Dapp\truffle_project>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\greetings.sol
> Artifacts written to C:\Users\Mahita\Desktop\COLLEGE\SEM 7\Blockchain\Dapp\truffle_project\build\contracts
> Compiled successfully using:
   - solc: 0.8.19+commit.7dd6d404.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      1696947913453
> Block gas limit: 6721975 (0x6691b7)

   > account:             0x3e6493731B1109915100D02FE650E02Ae736CCeA
   > balance:             99.99178264
   > gas used:            410868 (0x644f4)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00821736 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00821736 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.00821736 ETH
```

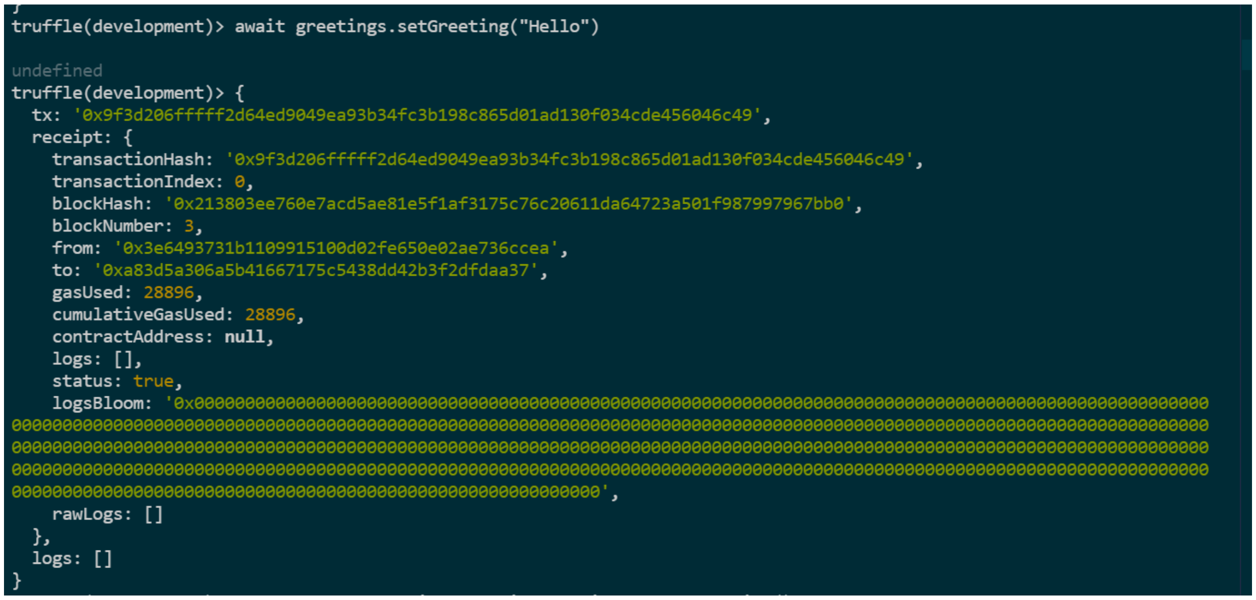
16. To interact with the blockchain, use the following command: **truffle console**
17. Type the following commands:
const greetings = await greetings.deployed();
greetings
await greetings.setGreeting("Hello")
const currentGreeting = await greetings.fetchGreeting()
currentGreeting



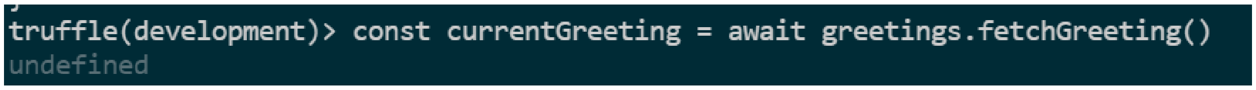
```
C:\Users\Mahita\Desktop\COLLEGE\SEM 7\Blockchain\Dapp\truffle_project>truffle console
truffle(development)> const greetings = await greetings.deployed();
undefined
```



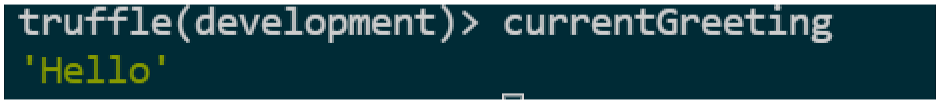
```
truffle(development)> greetings
TruffleContract {
  constructor: [Function: TruffleContract] {
    _constructorMethods: {
      configureNetwork: [Function: configureNetwork],
      setProvider: [Function: setProvider],
      new: [Function: new],
      at: [AsyncFunction: at],
      deployed: [AsyncFunction: deployed],
      defaults: [Function: defaults],
      hasNetwork: [Function: hasNetwork],
      isDeployed: [Function: isDeployed],
      detectNetwork: [AsyncFunction: detectNetwork],
      setNetwork: [Function: setNetwork],
      setNetworkType: [Function: setNetworkType],
      setWallet: [Function: setWallet],
      resetAddress: [Function: resetAddress],
      link: [Function: link],
      clone: [Function: clone],
      addProp: [Function: addProp],
      toJSON: [Function: toJSON],
      decodeLogs: [Function: decodeLogs]
    },
    _properties: {
      contract_name: [Object],
      contractName: [Object],
      gasMultiplier: [Object],
      timeoutBlocks: [Object],
```



```
truffle(development)> await greetings.setGreeting("Hello")

undefined
truffle(development)> {
  tx: '0x9f3d206fffff2d64ed9049ea93b34fc3b198c865d01ad130f034cde456046c49',
  receipt: {
    transactionHash: '0x9f3d206fffff2d64ed9049ea93b34fc3b198c865d01ad130f034cde456046c49',
    transactionIndex: 0,
    blockHash: '0x213803ee760e7acd5ae81e5f1af3175c76c20611da64723a501f987997967bb0',
    blockNumber: 3,
    from: '0x3e6493731b1109915100d02fe650e02ae736ccea',
    to: '0xa83d5a306a5b41667175c5438dd42b3f2dfdaa37',
    gasUsed: 28896,
    cumulativeGasUsed: 28896,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
```

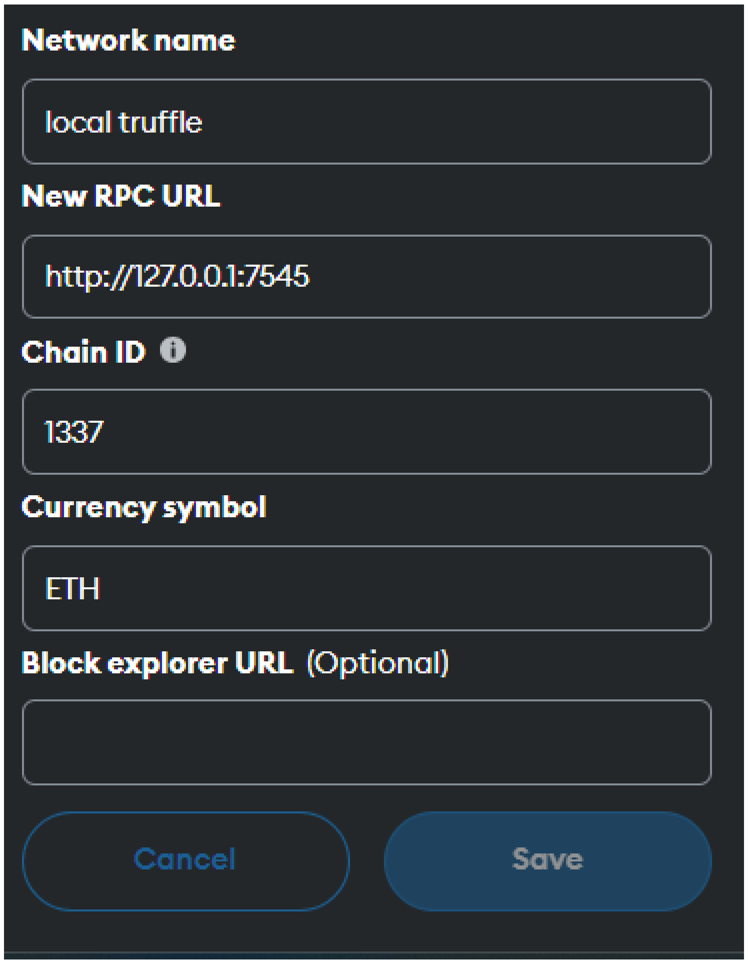


```
truffle(development)> const currentGreeting = await greetings.fetchGreeting()
undefined
```



```
truffle(development)> currentGreeting
'Hello'
```

18. Open the metamask extension, open Settings->Networks Click on Add Network and then click on Add A Network Manually and create a new network called Truffle test network with the following details:



Network name

local truffle

New RPC URL

http://127.0.0.1:7545

Chain ID

1337

Currency symbol

ETH

Block explorer URL (Optional)

19. Copy a private key of an account from Ganache GUI. In metamask, Click on Account dropdown, click on import account and paste the private key.

20. Now open new terminal, navigate to blockchain_dapp directory using command **cd blockchain_dapp.**

21. Type the command **npm start**.