1. npm install -g truffle
2. goto folder C:\Users\XXXX\AppData\Roaming\npm , rename truffle.cmd to truff.cmd
3. npm install -g ganache-cli
4. mkdir movieRatingApp //make a project directory
5. cd movieRatingApp
6. mkdir rating // backend folder
7. cd rating
8. truffle init //initialize a new truffle project
9. We will start by writing our smart contract called **Rating.sol**, then we will compile and interact with it by using truffle console
10. Inside contracts directory create a new file called Rating.sol with the following code:
11. pragma solidity ^0.4.17;
12. *// We have to specify what version of compiler this code will compile with*
13. contract Rating {
14. */\* mapping field below is equivalent to an associative array or hash.*
15. *\*/*
17. mapping (bytes32 => uint8) public ratingsReceived;
19. */\* We will use an array of bytes32 to store the list of movies*
20. *\*/*
22. bytes32[] public movieList;
23. */\* This is the constructor which will be called once when you*
24. *deploy the contract to the blockchain. When we deploy the contract,*
25. *we will pass an array of movies for which users will give ratings*
26. *\*/*
27. function Rating(bytes32[] movieNames) public {
28. movieList = movieNames;
29. }
30. *// This function returns the total ratings a movie has received so far*
31. function totalVotesFor(bytes32 movie) view public returns (uint8) {
32. return ratingsReceived[movie];
33. }
34. *// This function increments the vote count for the specified movie. Equivalent to upvoting*
35. function voteForMovie(bytes32 movie) public {
36. ratingsReceived[movie] += 1;
37. }
38. }
39. To deploy our newly created contract we need to add a file 2\_deploy\_contract.js file under migrations folder. After creating this file add the following code to it

var Ratings = artifacts.require("./Rating.sol");

module.exports = function(deployer) {

deployer.deploy(Ratings,['Star Wars', 'Avatar', 'Inception'], {gas: 6700000});

};

1. Inside our backend directory, open up **truffle.js** and add the following code:

module.exports = {

networks:{

development:{

host:"localhost",

port:8545,

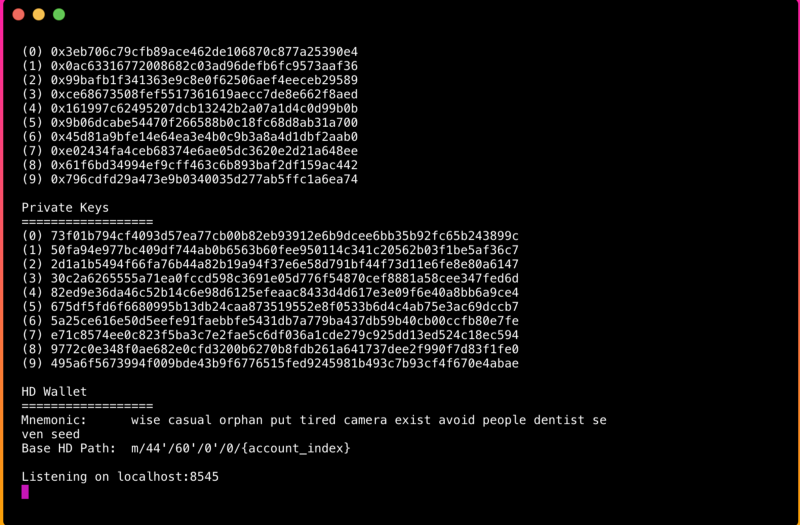
network\_id:'\*'

}

}

};

1. Run ganache-cli



1. Inside rating folder, type truffle compile

truff compile

This will create a new folder called build. This build folder contains JSON objects that will be used to deploy our smart contracts. Finally run truffle migrate

truffle migrate

As we are connected and running our testrpc on the same port we will see our smart contracts deployed successfully. Now that we have deployed our contract we can interact with it within the truffle console

truffle console

Lets see where the contract was deployed:

**Rating.address**

We will get the address where the contract was deployed in the network. Lets note down few things that we will need in order to connect this contract with our frontend. inside the truffle console, run

**JSON.stringify(Rating.abi)**

We are accessing the Application Binary Interface of our contract which is a list of functions in our contract in JSON format. This will be used to interact with our contract later.

Copy the returned contents of abi **without the single quotes** and the contract address at some place.

1. npm install -g create-react-app
2. Within our movieRatingApp folder run

create-react-app appui

1. This will create a react application within appui folder. Inside our appui folder open up package.json and **add web3 dependency** as shown:



1. Navigating into appui folder, install the new dependencies and then start the server:

npm install

npm start

1. Inside**appui/src**create a new file called EthereumSetup.js and add the following fields into it

import Web3 from 'web3';

const web3=new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

let ratingABI=[{"constant":true,"inputs":[{"name":"","type":"string"}],"name":"ratingsReceived","outputs":[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[{"name":"","type":"uint256"}],"name":"movieList","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{"inputs":[{"name":"movieNames","type":"string[]"}],"payable":false,"stateMutability":"nonpayable","type":"constructor"},{"constant":true,"inputs":[{"name":"movie","type":"string"}],"name":"totalVotesFor","outputs":[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"movie","type":"string"}],"name":"voteForMovie","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"}]

let ratingAddress='0x42b9991505950cc5ac38a6557d8efcdc40235de6';

web3.eth.defaultAccount = web3.eth.accounts[0]

const ratingContract=web3.eth.contract(ratingABI).at(ratingAddress);

export {ratingContract};

**Replace the ratingABI and ratingAddress values with previously noted values.**

1. Edit **App.js, ShowMovies.js and ShowMovies.css** files with the following code:

App.js

|  |
| --- |
| import React, { Component } from 'react'; |
| import logo from './logo.svg'; |
| import './App.css';  import { ratingContract } from "./EthereumSetup";  import {ShowMovies } from "./ShowMovies";  class App extends Component {  constructor(props){  super(props)  this.state={  movies : [{name:'Star Wars',rating:0},{name:'Avatar',rating:0},{name:'Inception',rating:0}]  }  this.handleVoting=this.handleVoting.bind(this)  }  handleVoting(movie){  ratingContract.voteForMovie(movie)  let votes=ratingContract.totalVotesFor(movie).toNumber()  this.setState({movies:this.state.movies.map(  (el)=>el.name===movie? Object.assign({},el,{rating:votes}):el    )});  }  render() {  return (  <div className="App">  <header className="App-header">  <img src={logo} className="App-logo" alt="logo" />  <h1 className="App-title">Welcome to Ethereum</h1>  </header>  <p className="App-intro">  Movie Rating Application in Ethereum and React  </p>  <div className="movie-table">  <ShowMovies movies={this.state.movies} vote={this.handleVoting}/>  </div>  </div>  );  }  }  export default App; |

ShowMovies.js

|  |
| --- |
| import React, { Component } from 'react'; |
| import './ShowMovies.css'  export class ShowMovies extends Component{  handleChange=(movie)=>{  let \_movie=movie;  this.props.vote(\_movie)  }  render(){  let movieList=this.props.movies.map((movie,i)=>  <tr key={i}>  <td onClick={this.handleChange.bind(this,movie.name)}>{movie.name}</td>  <td>{movie.rating}</td>  </tr>)  return(  <div>  <h3> Movies</h3>  <hr />  <table >  <tbody>  <tr>  <th>Movie</th>  <th>Rating</th>  </tr>  {movieList}  </tbody>  </table>  </div>  )  }  }  export default ShowMovies; |

ShowMovies.css

|  |  |
| --- | --- |
| table, th, td { |  |
|  | border: 1px solid #ccc; |
|  | margin: 0 auto; |
|  | padding:3px; |
|  | } |

1. Running blockchain explorer
   1. Goto file app.js and check the eth\_node\_url , it should point to correct port on which eth server is running
   2. Goto package.json , check the start attribute – it should have the correct port mentioned. The same port should be mentioned in the geth –rpccorsdomain “[http://localhost:8080](http://localhost:8080/)”
   3. <https://github.com/etherparty/explorer> - clone the code for explorer from following location
   4. Run the server with following command– **npm start**

Truffle console commands

Rating.at(Rating.address).movieList(0)

Rating.at(Rating.address).voteForMovie(0)

Rating.at(Rating.address).totalVotesFor(0)