# Server-Side Signed Txn (Part 3 of 4)

**4) Initialize and Configure ExpressJs Server**

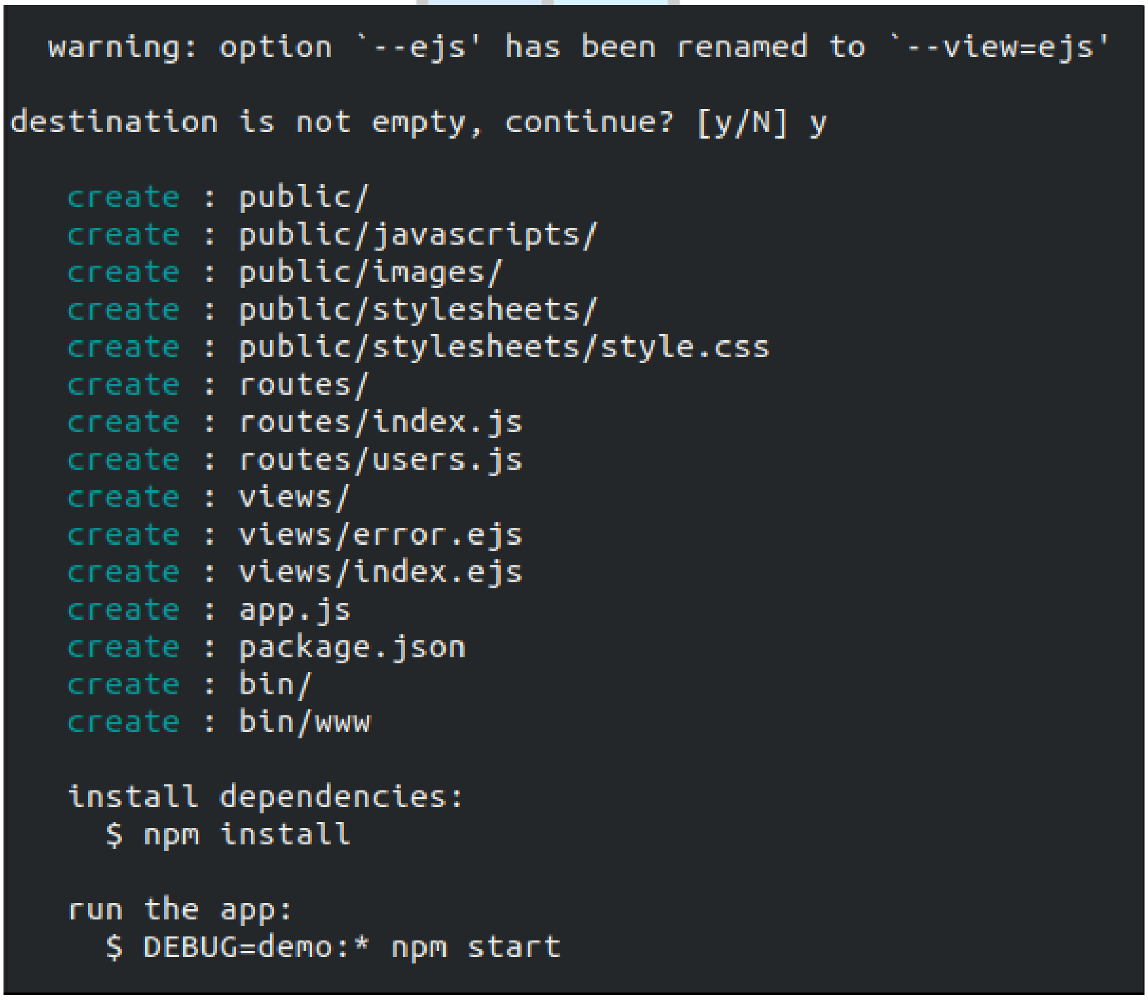
Now, we have successfully compiled and deployed smart contract in Ropsten network.

1) Initialize ExpressJS project structure in directory

   Open terminal in the project directory and run following command

```
$ express --view = ejs
```

   ___Truffle Output____



```
  warning: option `--ejs' has been renamed to `--view=ejs'

destination is not empty, continue? [y/N] y

   create : public/
   create : public/javascripts/
   create : public/images/
   create : public/stylesheets/
   create : public/stylesheets/style.css
   create : routes/
   create : routes/index.js
   create : routes/users.js
   create : views/
   create : views/error.ejs
   create : views/index.ejs
   create : app.js
   create : package.json
   create : bin/
   create : bin/www

   install dependencies:
     $ npm install

   run the app:
     $ DEBUG=demo:* npm start
```

# Server-Side Signed Txn (Part 3 of 4)

2) Install Dependencies

To install the basic Dependencies required by the ExpressJS project run below command.

```
>> npm install
```

Also We need some additional Dependencies to run our project.

```
>> npm install path --save
```

```
>> npm install web3@1.0.0-beta.48 --save
```

```
>> npm install ethereumjs-tx --save
```

After the installation you will see all the dependencies included in the **package.js** file as follows

```
{
 "name": "auction-server-side-txn",
 "version": "0.0.0",
 "private": true,
 "scripts": {
   "start": "node ./bin/www"
 },
 "dependencies": {
   "cookie-parser": "~1.4.3",
   "debug": "~2.6.9",
   "ejs": "~2.5.7",
   "ethereumjs-tx": "^1.3.7",
   "express": "~4.16.0",
   "http-errors": "~1.6.2",
   "morgan": "~1.9.0",
   "path": "^0.12.7",
   "web3": "^1.0.0-beta.48"
 }
}
```

# Server-Side Signed Txn (Part 3 of 4)

**5) Writing Code**

Follow the below steps to complete the Front-End, Web3 Integration and Transaction signing.

1. Copy the html code to the **views/index.ejs.**

2. Copy the below JavaScript code to **routes/index.js**.

```
var express = require('express');
var router = express.Router();

router.get('/', function (req, res, next) {
  res.render('index');
});

router.get('/populate', function (req, res, next) {
  MyContract.methods.getAuctionDetails().call().then(function
(aucDet) {
    MyContract.methods.auction_status().call().then(function (state)
{
      MyContract.methods.Mycar().call().then(function (carDet) {
        var highestEther = web3.utils.fromWei(aucDet[1] , "ether");
        if (state == 1)
          aucStatus = "Live"
        else if (state == 0)
          aucStatus = "Completed"
        var result = [carDet[0], carDet[1], carDet[2], aucDet[0],
highestEther, aucDet[2], aucDet[3], aucStatus];
        res.send(result);
      })
    })
  })
});

router.post('/view', function (req, res, next) {
  var bidderAddress = req.body.vw_publicAddr;
  MyContract.methods.bids(bidderAddress).call().then(function (data)
{
    data = web3.utils.fromWei(data , "ether");
    res.send(data);
  })
});

router.post('/bidnow', function (req, res, next) {
var bidderAddress = req.body.bid_publicAddr;
  var privateKey = req.body.bid_privateKey;
  var bidVal = String(req.body.bid_bidValue);
  var methodCall = MyContract.methods.bid();
  sendTransaction(methodCall, bidVal, bidderAddress, privatKey,
function (responce) {
```

# Server-Side Signed Txn (Part 3 of 4)

```
     if (responce == true)
      res.send('bidding Successful...');
    else
      res.send('bidding failed... Check Console for error...');
  });
});

router.post('/getAmount', function (req, res, next) {
  var ownerAddress = req.body.finish_publicAddr;
  var privatKey = req.body.finish_privateKey;
  var methodCall = MyContract.methods.withdraw();
  sendTransaction(methodCall, null, ownerAddress, privatKey, function
(responce) {
    if (responce == true)
      res.send('Withdrawal Successfull...');
    else
      res.send('Failed to withdraw... Only Bidders can withdraw bid
value...');
  });
});

router.post('/withdraw', function (req, res, next) {
  var bidderAddress = req.body.wtdw_publicAddr;
  var bidderKey = req.body.wtdw_privateKey;
  var methodCall = MyContract.methods.getAmount();
  sendTransaction(methodCall, null, bidderAddress, bidderKey,
function (responce) {
    if (responce == true)
      res.send('Withdrawal Successfull...');
    else
      res.send('Failed to withdraw... Only Auctioneer can withdraw
bid value...');
  });
});
module.exports = router;
```