BCDV1022 - Backend Web Development
Blockchain Explorer

Due Tues, Nov 21th – 30% of Final Grade

System Requirements

Contact: **Mike Denton**

Date: **Nov 19th, 2023**

Version #: **1.0**

1 Objective

This document contains a specification of the course assignment. It is a task where students practice skills to build a full stack web application using the backend technologies Node.js and MongoDb and React offered by the MERN Stack. This task will also include, working together in a group developing the project, plan, manage and coordinate development activities, to be done effectively to a deadline.

2 Teams

Teams will consist of one to two students (with two being the target number).

3 Backend Server

The backed server will include Node.js. Any Node packages built-in or 3rd party found on the NPM registry can be used.

4 Database

We will be using NOSQL Database to persist the data. We will be using the cloud hosted MongoDB on Mongo DB Cloud Atlas https://www.mongodb.com/cloud/atlas

5 User Interface

The User Interface will be based on the React application built in FullStack I.

6 Specification

We will be integrating the frontend technologies of React and backend technologies of Node.js, Express (alternatively Nest.js) and Mongoose.

7 Blockchain Integration

There is no blockchain integration with truffle or hardhat required for this project. This will be covered in Fullstack III.

8 Node Backend Server

Build a web server that will have the following functionality:

1. Provide and expose Restful API endpoints via Express
2. Built logical routes using Express router
3. Manage connection to Mongo Db via Mongoose to save and persist data

8.1. Node Custom Modules

1 Create 2 custom modules named **accounts** and **transactions**

Accounts module

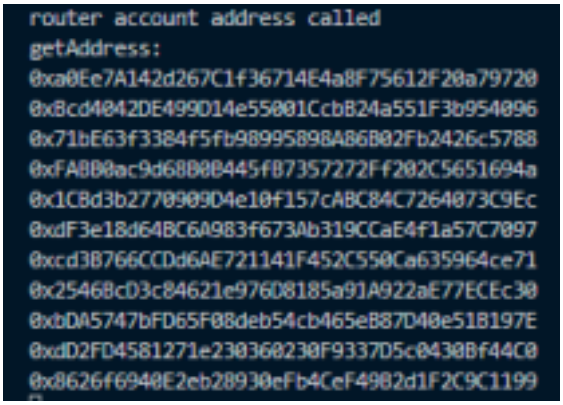
- will have two public methods **getAddresses** and **getBalance**

getAddresses *method: (Lab Test - Completed)*

1. There are no required parameters for this method
2. Returns list of account addresses

- **Note**: This method can return the list of mock addresses from Full Stack I. We are moving the mock/static values from the React component to the Node.js module.

Sample output from console when call getAddress node module



getBalance method

1. one parameter address - the account address to retrieve the balance
2. returns the balance and address of the account

- **Note**: This method can return the list of mock/static balance and address from Full Stack I. We are moving the mock/static values from the React component to the Node.js module.

Sample output from console when call getBalance node module

```
router GET:account/balance called:
getBalance module called..
account: 0xf39Fd6e51aad88F6F4ce6aB8827279cfffFb92266: balance: 98765432100
```

Transaction module

- will have one public method **getTransactionHistory**

- **getTransactionHistory** will have the following requirements
- create a mongoose connection to retrieve and return all of the transaction history documents -

- **sendTransaction**
    - will have three parameters
        - source - the source account address
        - destination - the destination account address
        - value - the amount value to send
        -
    - Returns a receipt static/mock object.

8.2 Mongoose and MongoDb

Build models and schemas with mongoose middleware to store data in mongoDb

1. Query data in local mongo database
2. Build **Transaction** model and schema to retrieve **transaction** history
    a. schema should include the following fields:
        i. **source** - a source account address
        ii. **destination** - a destination account address
        iii. **amount** - amount value to send
        iv. **status** - result of the transaction
        v. **gasUsed** - the amount of gas used, this is an optional field and will only be stored if transaction was successful
        vi. **receiptHash** - the transaction hash from the receipt object. this is an optional field and will only be stored if the transaction was successful

Note: Use the and run the following script in node to seed your mongodb with transaction history. Continue building the mongoose schema to include the following fields.

https://drive.google.com/file/d/1LBGgfYNmXBi2rqL2McFf9b8Onye1GRWr/view?usp=share_link


```
_id: ObjectId("62428623209adfad7175feb2")
source: "0xdd2fd4581271e230360230f9337d5c0430bf44c0"
destination: "0x8626f6940e2eb28930efb4cef49b2d1f2c9c1199"
amount: "250"
status: "SUCCESS"
gasUsed: "21000"
receiptHash: "0x39ae9270a0f2edac19f51d1e9b42380d73440f142d1d1709fb3806c97539c6eb"
createdAt: 2022-03-29T04:08:03.172+00:00
updatedAt: 2022-03-29T04:08:03.172+00:00
__v: 0
```

```
_id: ObjectId("62428634209adfad7175feb4")
source: "0xdd2fd4581271e230360230f9337d5c0430bf44c0"
destination: "0x8626f6940e2eb28930efb4cef49b2d1f2c9c1199"
amount: "5000"
status: "SUCCESS"
gasUsed: "21000"
receiptHash: "0xdecea8161a7602952960d9bd3f2e7b9fe9255c28856b479c00ed3f3a65e853e4"
createdAt: 2022-03-29T04:08:20.496+00:00
updatedAt: 2022-03-29T04:08:20.496+00:00
__v: 0
```


8.2 Express and Router *(Alternative Next.js routes)*

1. Create routes for **accounts**
    a. ***GET*** *request on route* ***/account/addresses*** *- will return a list of all the available node address from hardhat (no parameters required)*
        i. calls ***getAddresses*** *method from the Account module*
    b. **GET** request on route **/account/balance** - will return a balance of the account given the **address** as a query string parameter
        i. calls **getBalance** method from Account module

2. Create routes for **transaction**
    a. **GET** request on route **/transaction/history** - will return a list of transaction history
        i. calls the **getTransactionHistory** method from the Transaction module

    a. **POST** request on route **/transaction/send**

i. calls the **sendTransaction** method in the transaction custom module

9.0 React UI Integration

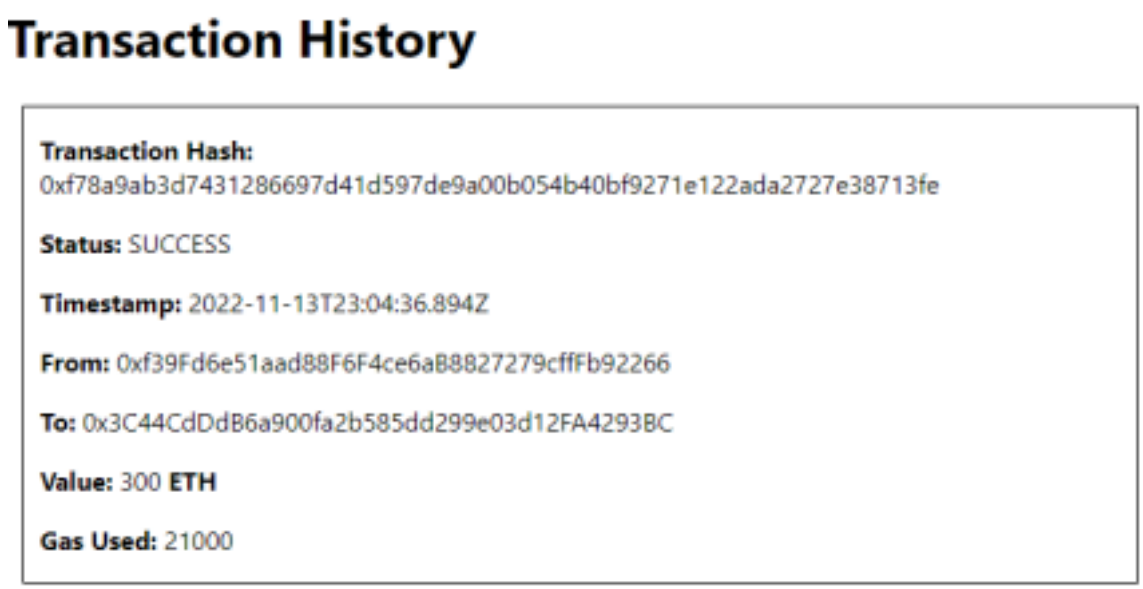
- integrate the existing React UI from Fullstack I to fetch data from the backend APIs for the read only components. The form input screens including POST requests are out of scope for this project.

9.1 Blockchain Node Address Component *(Lab Test - Completed)*

- use **fetch** or **axios** to send a **GET** request to the route **/account/addresses** - use hooks (***useEffect*** & ***useState***) or class components (***componentDidMount*** and ***setState***) to trigger the **GET** request and update state with the readonly data - replaced the existing mock data from Fullstack I and render the data returned from the web GET request

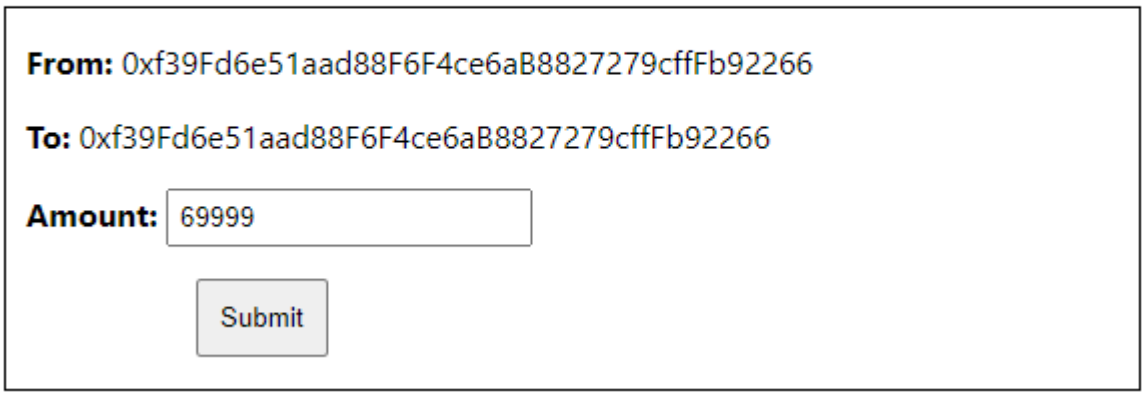
9.2 Transaction History Component

- use fetch or axios to send a GET request to the route **/transaction/history** - use hooks (***useEffect*** & ***useState***) or class components (***componentDidMount*** and ***setState***) to trigger the **GET** request and update state with the readonly data - replaced the existing mock data from Fullstack I and render the data returned from the web GET request



9.3 Transfer/Send Ether

Transfer



Component Requirements

- **Developer Note:** The **default node address** for all balance and transaction requests can be one hard-coded account address value or from a target block ie. Last Block.

- This component will contain an interactive form and will handle the submission of the form via the user click action on the submit button.

- This component will send a **POST request** to the Express API route **transaction/send.** The data payload will include the source, destination addresses as well as the amount

The component will show the **Transfer Receipt (Section 9.4)**, if the **POST request** was successful. Otherwise, the receipt should not be shown

9.4 Wallet Component

- use fetch or axios to send a GET request to the route **/account/balance** - use hooks (***useEffect*** & ***useState***) or class components (***componentDidMount*** and ***setState***) to trigger the **GET** request and update state with the readonly data - replaced the existing mock data from Fullstack I and render the data returned from the web GET request

My Wallet

Address: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

Balance: 98765432100 ETH

Bonus (Optional)

Continue with the blockchain integration using web3.js or ether.js to connect and integrate with a virtual blockchain ie. Test RPC, Ganache, Hardhat, Truffle

Submission

1. The project code submission is via MS Teams via a zip file. Please include in your student id and group member names in READ.ME file
2. Include a README file with the project that includes the following:
    - The names and student number of all the members of the team.
    - Instructions for installing or running the project.

Specification	Percentage
Backend Server (Node.js) 25%	**25%**
Implementing Express & Router *(Next.js Routes)* 25%	**25%**
Implementing Mongoose 20%	**20%**
React UI integration 25%	**25%**
Clean Code and Clarity 5%	**5%**
Bonus Mark - Blockchain Integration	**15%**