BCDV1022 - Backend Web Dev Lab Test

System Requirements

Contact: **Mike Denton**

Version #: **1.0**

1 Objective

This document contains a specification of the course lab test. It is a task where students practice skills to build a full stack web application using the backend technologies Node.js and create API endpoints with Express or Nest.js This will be integrated with an existing React UI application from Full Stack I - Frontend Web Dev.

2 Teams

This is an individual submission with one student. Students may work together and collaborate. However, code should not be directly copied and submitted from another student.

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. Express or Nest.js framework can be used for the API communication

4 Database

There is no database required for this submission. The data returned will be mock static data.

5 Node Backend Server

Build a web server that will have the following functionality:

1. Provide and expose Restful API endpoints via Express or Nest.js
2. Built logical routes using Express router or Nest.js routes

5.1. Node Custom Modules

1 Create 1 custom modules named **accounts or accountsModule**

Accounts module

- will have one public method **getAddresses**

getAddresses method:

1. There are no required parameters for this method
2. Returns list of account addresses (mock static values)
3. Export the getAddresses method so that the module will have a public method

- **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.

Implementation Notes:

- The npm package **faker-js** can be used to help generate the ethereum addresses found in mock data.
  https://www.npmjs.com/package/@faker-js/faker
  https://fakerjs.dev/api/finance.html#ethereumaddress

5.2 API Routing *(Use Express, or Express Router or Nest.js routes)*

1. Create routes for **accounts**
    a. **GET** request on route **/account/addresses** - will return a list of all the available node address
        i. The handler/callback will call **getAddresses** method from the Account module when the GET request is received

6.0 React UI Integration

- integrate the existing React Frontend UI from Fullstack I to fetch data from the backend APIs for the read only components.

6.1 Address Component

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

Blockchain Node Addresses

0xa0Ee7A142d267C1f36714E4a8F75612F20a79720
0xBcd4042DE499D14e55001CcbB24a551F3b954096
0x71bE63f3384f5fb98995898A86B02Fb2426c5788
0xFABB0ac9d68B0B445fB7357272Ff202C5651694a
0x1CBd3b2770909D4e10f157cABC84C7264073C9Ec
0xdF3e18d64BC6A983f673Ab319CCaE4f1a57C7097
0xcd3B766CCDd6AE721141F452C550Ca635964ce71
0x2546BcD3c84621e976D8185a91A922aE77ECEc30
0xbDA5747bFD65F08deb54cb465eB87D40e51B197E
0xdD2FD4581271e230360230F9337D5c0430Bf44C0
0x8626f6940E2eb28930eFb4CeF49B2d1F2C9C1199

Submission

1. The project code submission is via MS Teams via a zip file. Please include in your student id and name in the file submission ie. mike.denton.442424-lab-test-ii-zip
2. Include a README file with the project that includes the following:
    - The name and student number.
    - Instructions for installing or running the project.

Specification Percentage
Project setup and use of npm 5%
Backend Server (Node.js) 30%
Implementing API Routes *(Express, Express Router or Nest.js Routes)* 30%
React UI integration 30%
Clean Code and Clarity 5%