**FS II - Backend Web Development Blockchain Explorer**

Due Tues, March 26th – 25% of Final Grade

**System Requirements**

Contact: **Mike Denton**

Date: **Mar 18th, 2024**

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**

This is a group submission, with the goal working in pairs, however individual submission is acceptable.. While students may collaborate, code should not be directly copied and submitted from another student.

**3. Backend Server**

The backend server will use Node.js. Students are allowed to use any Node packages (built-in or 3rd party from NPM) for this task. The API communication can be implemented using either Express or Nest.js framework.

**4. Database**

We will be using both NoSQL and SQL databases to persist the data. For NoSQL, we will use the cloud-hosted MongoDB on Mongo DB Cloud Atlas (<https://www.mongodb.com/cloud/atlas>). For SQL, we will use PostgreSQL, which can be hosted on cloud platforms like Heroku or AWS RDS.

**5 User Interface**

The User Interface will be built in React and will be re-used from the project completed in Full Stack I - Frontend Development.

**6 Blockchain Integration**

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

**7.0 Node Specification**

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 (alternatively Postgres can be used) to save and persist data

**7.1. Blocks - Custom Node Module**

* Completed in FS II - Backend Dev Lab Test

**7.2. Transactions - Custom Node Module**

* Will have two public functions

**Function: getTransactionHistory**

* There are no required parameters for this method
* Returns a list of transaction records retrieved from the database
* The function should be exported so that it can be imported and used in other parts of the application.

**Function: sendTransfer**

* The function requires three parameters: source, destination, and amount.
* The function should create a mock receipt using the source and destination addresses.
  + The receipt should also include the following:
    - source - the given source account address
    - destination -the given destination account address
    - amount - transfer amount
    - gasUsed - the mocked gas used
    - Receipt Hash - the mock generated has value for the receipt transaction
* The function should trigger a save to the database to persist the generated receipt transaction for future reference.
* The function should be exported so that it can be imported and used in other parts of the application.

**8.0 Database - MongoDb and Mongoose**

Utilize Mongoose middleware to create models and schemas for storing data in MongoDB.

Implement functionality to query and save transaction data in the local MongoDB database.

Construct a Transaction model and schema with the following specifications:

The schema should include the following fields:

* source: The source account address.
* destination: The destination account address.
* amount: The amount value to send.
* status: The result of the transaction (e.g., "success", "failure").
* gasUsed: The amount of gas used. This field is optional and should only be stored if the transaction was successful.
* receiptHash: The transaction hash from the receipt object. This field is optional and should only be stored if the transaction was successful.



**9.0 RESTful API - Express** ***(Alternative Nest.js routes)***

* **Blocks API Routes**
  + Completed in FS II - Backend Dev Lab Test
* **Transactions API Routes**
  + Completed in FS II - Backend Dev Lab Test

Create two api endpoints for **transactions routes**

* **GET** request on route **/transactions/history** 
  + calls the **getTransactionHistory** function from the Transactions module
  + returns a list of transaction history
* **POST** request on route **/transactions/send**
  + payload include the following details
    - source - the source account address
    - destination - the destination account address
    - amount - the transfer amount
  + calls the **sendTransfe**r function from the Transactions module
  + returns the receipt object from **sendTransfer** function

**10.0 React UI Integration**

Integrate the existing React UI from Fullstack I to fetch data from the backend APIs. Fetch or Axios should be used to make the web request from client to the server.

**10.1. Blocks & Block Details Components**

* Completed in FS II - Backend Dev Lab Test

**10.2. Transfer Component**

This component will send two requests to the backend server:

**GET** request on route **/blocks/addresses**

* (Route completed in FS II - Backend Dev Lab Test)
* use React lifecycle hook (***useEffect*** & ***useState***) or class components (***componentDidMount*** and ***setState***) to fetch the data from the backend
* Bind the source and destination addresses drop controls with the return list of addresses.

**POST** request on route **/transactions/send**

* Update the form submit event handler to send the request when the use clicks the submit/send button
* Bind the Receipt component with the receipt data returned from the server

**10.3. Transaction History Component**

**GET** request on route **/transactions/history**

* use React lifecycle hook (***useEffect*** & ***useState***) or class components (***componentDidMount*** and ***setState***) to fetch the data from the backend
* Bind the transaction history table/grid with the data returned from the server.

**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.

**Grading Specification**

| Backend Server (Node.js) | 20% |
| --- | --- |
| Implementing API Routes *(Express, Express Router or Nest.js Routes)* | 20% |
| Implementing Database *(MongoDb or Postgres)* | 20% |
| React UI integration | 20% |
| Implementation Level | 15% |
| Clean Code and Clarity | 5% |
| *Bonus Web 3 integration with virtual blockchain* | 10% |

Bonus marks can be awarded, but the total marks cannot exceed 100%