Twitter API

Instructions to run the server:

cd into the project folder, and do npm install.

You can optionally test the project before build by doing: npm test

Build the project by doing: npm run build

Then have a MongoDB server ready to go. You can set this up by having MongoDB installed on your machine, and do:

mkdir ./data
mongod --dbpath ./data

Then in a separate terminal, you can call mongo, and try to access the MongoDB server and get the database connection string.

From there, make a .env file in the root folder of the project, with the following contents:

DB_CONNECT = <Your database connection string>
PORT = <Your port number>
TOKEN_SECRET = <Your JWT secret string>

Substitute your desired values where the brackets are above.

Run the project by doing: node ./dist/src/index.js

Project Notes:

I am quite proud that I managed to build this project using TypeScript. TypeScript has its advantages of type-safety and it makes the project more scalable. It does come with the downside where development does slow down a bit to make sure the code I wrote is type-safe.

I was able to go as far as completing the first part of the assignment and the chat system portion of the second part. Unfortunately, I didn't have time to test the chat system.

The project is structured to follow the MVC paradigm. All the source code is located in the src folder. The models are located in the src/model folder, and controllers are located in

src/controllers. Routes are located in src/routes folder, and src/util contains all the helper functions that any of the other files may use.

I am quite happy with how the testing was done. I was able to elegantly test my User controllers using Mocha, Chai, and Sinon libraries. I was able to stud the function calls to the database, and stud JWT functions so that we wouldn't need to substitute a secret string. Thus I was able to properly unit test my User controller without an external database or .env file. If I had more time, I would've done integration testing with a live database.

Another thing I am happy about is that I was able to implement my chat system using socket.io. Unfortunately, I didn't have a lot of time to actually write test cases for the system, but I did my best to make sure that my code is clean.

Other Noteworthy Repositories:

Simple-C Compiler:

I am currently writing a compiler, using C++, to compile a simplified version of C to Assembly or x86-64.

Link: https://github.com/Playjasb2/Simple-C-Compiler

UofT Course Graph:

I am currently making a network graph of all the courses at UofT. I am using NextJS, React, TypeScript, and I plan to use Redux for state management.

Link: https://github.com/Playjasb2/UofT-Course-Graph/tree/NextJS

URL Shortener:

In a team, we made a URLShortner that shortens URLs, and the whole system was designed to be scalable, by using a proxy server and three web servers behind it. All the servers were implemented using Java.

Link: https://github.com/Playjasb2/URLShortener