2020

CAB230 Stocks API – Server Side

Stocks API 1.0.0 OAS3

This API has been created to support assignment work in the QUT Web Computing units for 2020. It exposes a small number of REST endpoints which implement CRUD operations on a database containing a snapshot of publicly available stock price data. The database includes entries - over a limited time frame - for a selected number of companies listed on our example Stock Exchange. There are three Query endpoints - `GET` s to allow you to retrieve data from the database - and two `POST` endpoints to manager User registration and login. Each of these endpoints is fairly straight forward and their usage is documented below. Note that the `/stocks` and `/stocks/authed` endpoints are similar in principle, but the latter endpoint offers additional functionality - the ability to select data based on a date range - that is available only to authenticated users. ***Note:*** All non-path query parameters are *optional* and *must* be lower case.

CAB230

Stocks API – Server Side Application

Do Viet Hoang

N10329935

4/6/2020

Contents

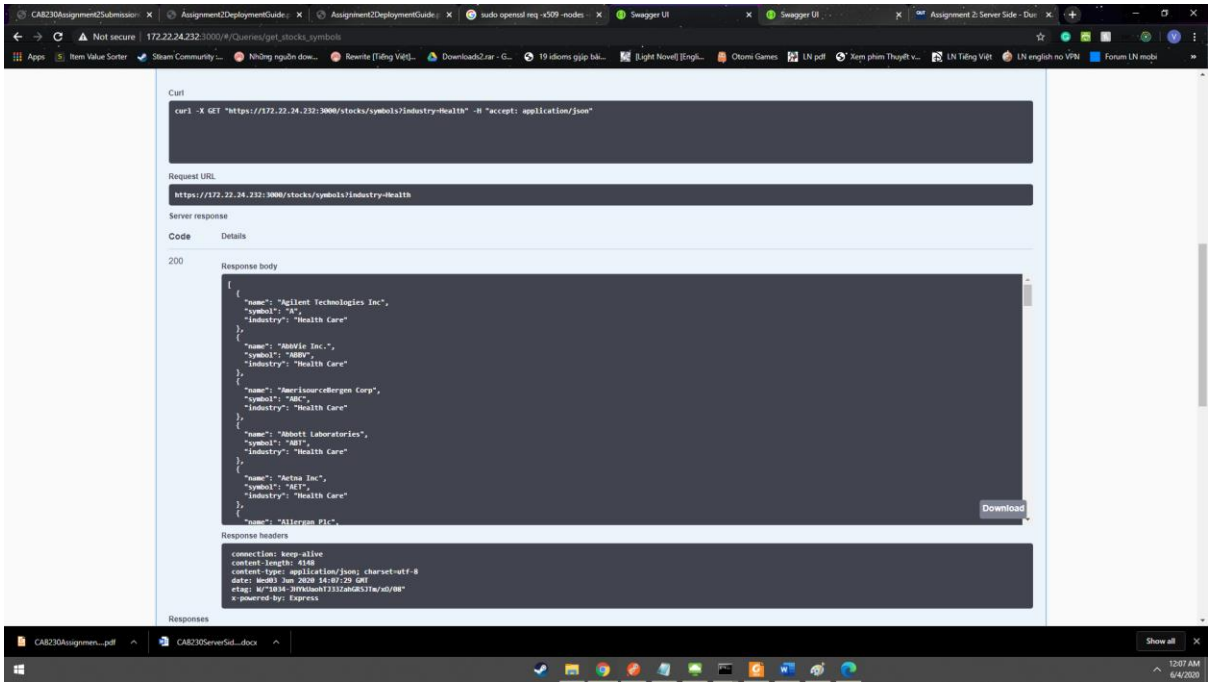
This template is adapted from one created for a more elaborate application. The original author spends most of his professional life talking to clients and producing architecture and services reports. You may find this a bit more elaborate than you are used to, but it is there to help you get a better mark

This report will probably be around 5 pages or so including screenshots

Introduction

Purpose & description

The Stocks API contains the database about stock price history of different companies in several industries. It offers three GET endpoints to help users retrieve desired information about companies, and two POST endpoints to help users create and login their account.



Completeness and Limitations

You may find it helpful to work with the list of endpoints below, telling us of any limitations. If the endpoint is fully functional, just say ‘fully functional’ after the route:

/stocks/symbols – fully functional

/stocks/{symbol} – fully functional

/stocks/authed/{symbol} – fully functional

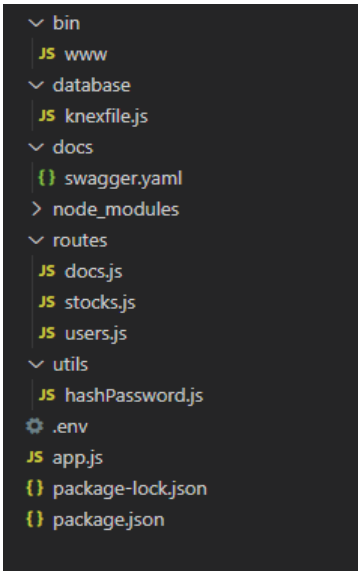
/user/register – fully functional

/user/login – fully functional

Modules used

No additional modules used

Technical Description



Architecture

The application has 5 directories:

/bin: define and setup server

/docs: define and setup Swagger document

/routes: define the routes

/utils: handle password hashing

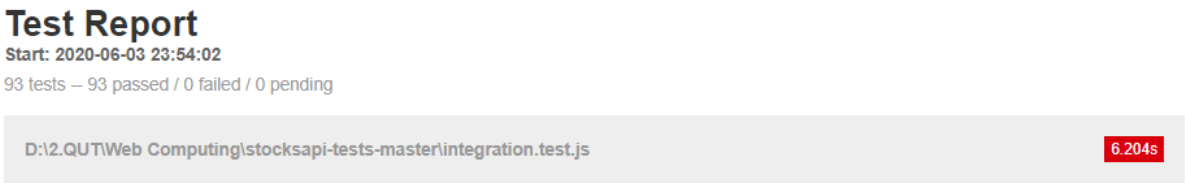
/database: setup knex options

The modules are divided in separate folder based on their functionality.

Security

- Knex is implemented to avoid the use of raw SQL
- Helmet is implemented with the default settings enable
- Morgan is implemented with a logging level similar to that used in the pracs
- Users password is appropriately handled

Testing



Difficulties / Exclusions / unresolved & persistent errors /

The server works properly without major bugs.

Installation guide

Open the .env file to change the port and database setting to suit the system

```
PORT=3000
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=root
DB_DATABASE=webcomputing
DB_PORT=3306
SECRET_KEY=secret_key
```

To run the server, first install the dependencies using command: npm install

Then start the server using command: npm start

Now, the server should run on http://localhost:3000