2020

CAB230 Stocks API – Server Side

CAB230

Stocks API – Server Side Application

Max O’Dell

N10336516

4/23/2020

Contents

Introduction

Purpose & description

The subject of this report is the Stocks API, an API that exposes several endpoints that enable its users to work with a database of stock entries. The database in question contains a number of different stocks of companies pertaining to a variety of industries and is used by the API to provide a vast range of functionality.

Completeness and Limitations

The opinion is held that this API is developed at a very high level with little-to-no obvious shortcomings. All endpoints have been supported, all obvious errors have been handled, all response headers and statuses have been implemented and the API reacts to any issues gracefully.

/stocks/symbols – fully functional

/stocks/{symbol} – fully functional

/stocks/authed/{symbol} – fully functional

/user/register – fully functional

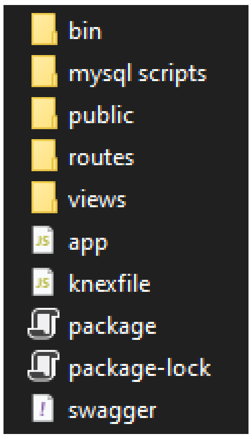
/user/login – fully functional

Modules used

No additional modules used

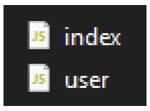
Technical Description

Architecture



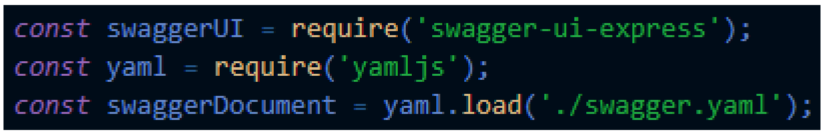
bin
mysql scripts
public
routes
views
app
knexfile
package
package-lock
swagger

The architecture of my application is very basic, with minimal directories. The knex options file and the swagger document are both on the top-level directory to keep things simple and accessible, and the mySQL scripts are included in a folder.

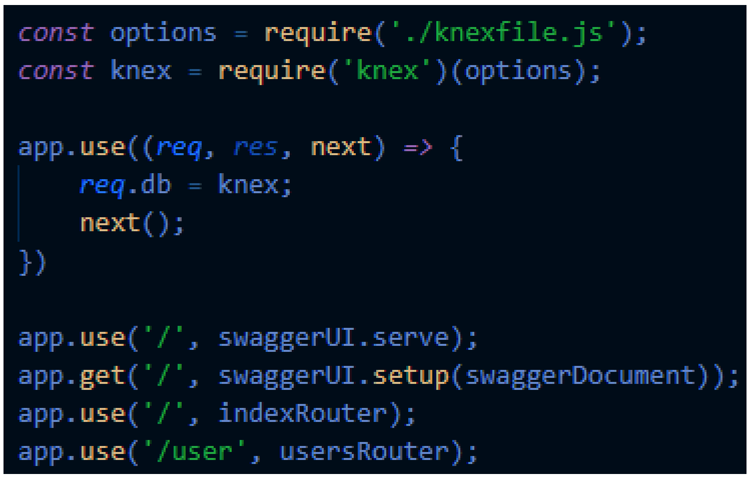


index
user

The routes directory stays unmodified, as do the rest of the directories. The Express Generator imposed the split of processes and directories that I would have ultimately gone with off my own accord, and so I elected to keep this untouched.



```
const swaggerUI = require('swagger-ui-express');
const yaml = require('yamljs');
const swaggerDocument = yaml.load('./swagger.yaml');
```



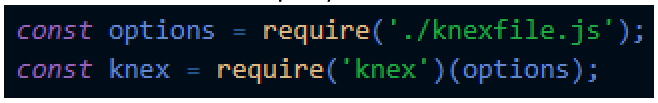
```
const options = require('./knexfile.js');
const knex = require('knex')(options);

app.use((req, res, next) => {
    req.db = knex;
    next();
})

app.use('/', swaggerUI.serve);
app.get('/', swaggerUI.setup(swaggerDocument));
app.use('/', indexRouter);
app.use('/user', usersRouter);
```

As is seen, the main app,js file delegates the roles of the routers, serves the swagger docs as the landing screen for the user and establishes the connection to the mySQL database.

Security

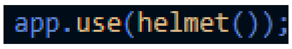
All security prerequisites have been incorporated.

- ✓ Use of `knex` or other query builder without raw SQL



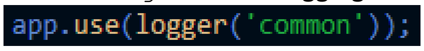
```
const options = require('./knexfile.js');
const knex = require('knex')(options);
```

- ✓ Use of `helmet` with at least the default settings enabled



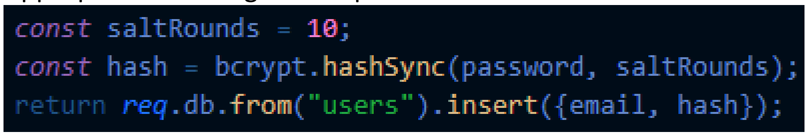
```
app.use(helmet());
```

- ✓ Use of `morgan` with a logging level similar to that used in the pracs.



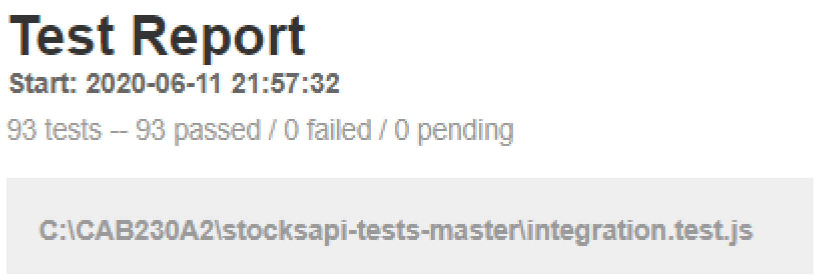
```
app.use(logger('common'));
```

- ✓ Appropriate handling of user passwords as described in the JWT Server-Side worksheet.



```
const saltRounds = 10;
const hash = bcrypt.hashSync(password, saltRounds);
return req.db.from("users").insert({email, hash});
```

Testing



Test Report

Start: 2020-06-11 21:57:32

93 tests -- 93 passed / 0 failed / 0 pending

C:\CAB230A2\stocksapi-tests-master\integration.test.js

Difficulties / Exclusions / unresolved & persistent errors /

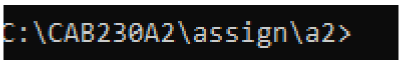
There are no outstanding bugs in this API, all features have been implemented successfully. There is no functionality that was not included in the final version of this application and there were no real roadblocks in development – it all ran fairly smoothly.

Installation guide

Tell us how to install the application – this should be at the same sort of level as most github sites.

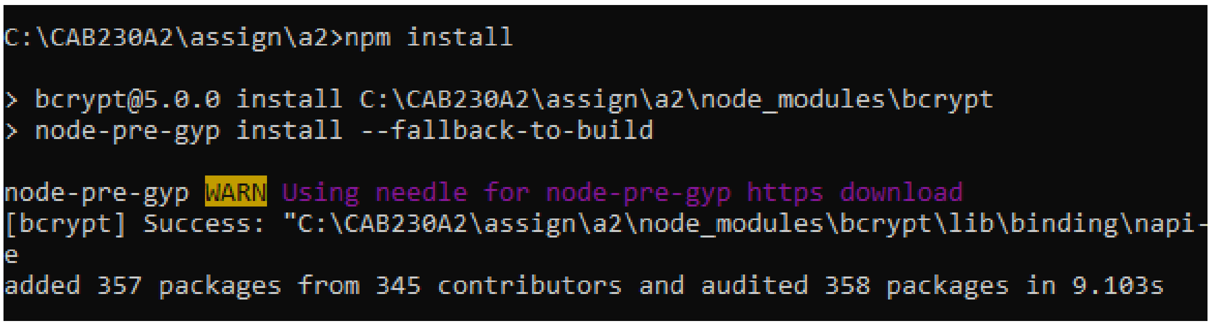
Use screenshots as needed.

Installing the application starts with a terminal window in the top-level directory, my personal version looking like this:



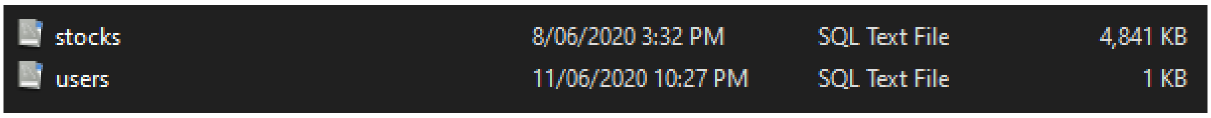
```
C:\CAB230A2\assign\a2>
```

From there, *npm install* is required to download all package dependencies into the project file and enable all of the critical features and files.



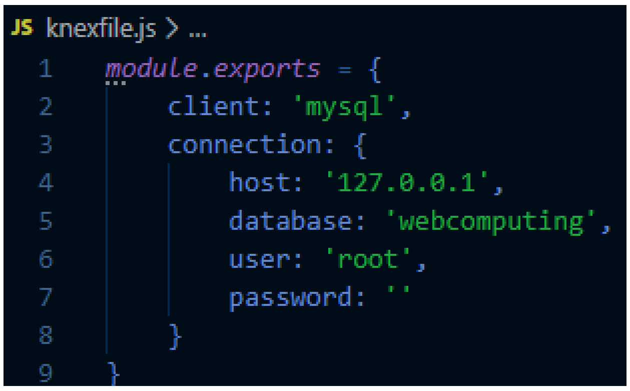
```
C:\CAB230A2\assign\a2>npm install

> bcrypt@5.0.0 install C:\CAB230A2\assign\a2\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

node-pre-gyp WARN Using needle for node-pre-gyp https download
[bcrypt] Success: "C:\CAB230A2\assign\a2\node_modules\bcrypt\lib\binding\napi-
e
added 357 packages from 345 contributors and audited 358 packages in 9.103s
```

Next to be set up is the mySQL database. Inside the project directory is a folder named *mysql scripts*.



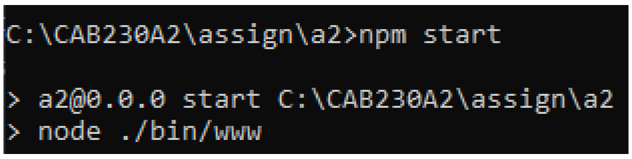
```
stocks    8/06/2020 3:32 PM     SQL Text File    4,841 KB
users     11/06/2020 10:27 PM   SQL Text File        1 KB
```

Both of these scripts are to be executed within mySQL workbench to create the two tables – *stocks* and *users.* Executing the stocks script first creates the *webcomputing* schema that both tables can then be created within. *knexfile.js* is responsible for establishing the connection to mySQL and should be edited to include the user's own details for connection.



```
knexfile.js > ...
module.exports = {
    client: 'mysql',
    connection: {
        host: '127.0.0.1',
        database: 'webcomputing',
        user: 'root',
        password: ''
    }
}
```

After mySQL set up, the last step is to start the server. Typing *npm start* into the terminal window will launch the server, serving on https://localhost.



```
C:\CAB230A2\assign\a2>npm start

> a2@0.0.0 start C:\CAB230A2\assign\a2
> node ./bin/www
```