

2020

# CAB230 Stocks API – Server Side



Scott Jacob

n9013695

01/06/2020

## Contents

Introduction .....	2
Purpose & description .....	2
Completeness and Limitations .....	2
/stocks/symbols.....	2
/stocks/{symbol}.....	2
/stocks/authed/{symbol}.....	2
/user/register .....	2
/user/login .....	2
Modules used .....	2
bcrypt.js.....	2
Technical Description .....	3
Architecture.....	3
Testing .....	3
Difficulties / Exclusions / unresolved & persistent errors.....	4
Extensions (Optional) .....	5
Installation guide .....	5
References.....	7
Appendices as you require them.....	8

### Introduction

#### Purpose & description

This server built with Express Js has been designed to provide a RESTful API for the previous Client-Side Assessment. The API front end is displayed and interacted through Swagger documentation and provides a comprehensive and clear function of the API including the endpoints available. The server and resulting API features all the available routes from the Client-Side server deployed, not limited to login, registration as well as authenticated routes for more comprehensive data retrieval.

#### Completeness and Limitations

The following routes are fully functional and implemented:

*/stocks/symbols*  
*/stocks/{symbol}*  
*/stocks/authed/{symbol}*  
*/user/register*  
*/user/login*

Modules such as Knex have been used to prevent raw SQL queries from being run against the database and will provide protection against SQL injection attacks. Other modules such as Morgan and Helmet are also used to provide verbose logging of requests to the server. The server is also served using HTTPS to prevent traffic intercepted between the server and the end user being read.

Limitations to the assessment are as follows:

- Swagger docs from home route (explained under unresolved errors)
  - o \*The documentation is implemented via redirect

#### Modules used

*bcrypt.js*

Module used in place of bcrypt, this module is written in JavaScript and provides identical functionality to the original bcrypt.

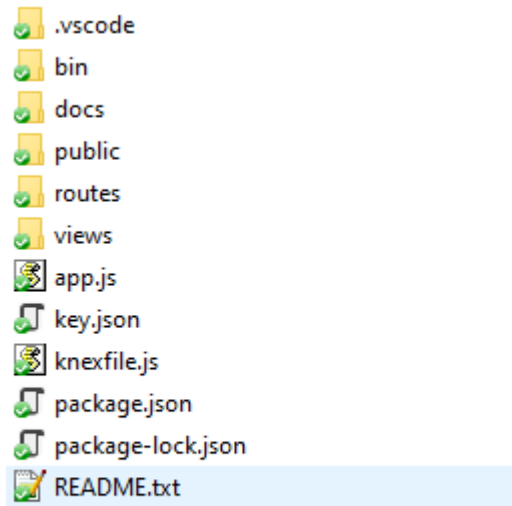
<https://www.npmjs.com/package/bcryptjs>

*No other modules used.*

### Technical Description

#### Architecture

This app was generated through the Express-generator package and all code is organized as such. Due to technical limitations (discussed later in this report) a /docs directory is being used to host and load the swagger documentation. Routing functionality is contained in the top directory within app.js and the individual routes of /stocks and /user are contained in the /routes directory within index.js and user.js respectively.



#### Testing

All tests have passed as shown in the screenshot below:

##### Test Report

Start: 2020-06-02 10:04:07

93 tests - 93 passed / 0 failed / 0 pending

/home/sjicab230/stocksapl-tests/integration.test.js

2.441s

### Difficulties / Exclusions / unresolved & persistent errors /

Major difficulties in preparing this assignment can be categorized into two main issues: syntactical and technical.

#### Technical Issues:

The swagger docs are being served from the /docs route; however, the user is redirected to this route when loading the index or '/' route. In attempting to move the swagger docs to be displayed from the index route the API would break and all attempted queries would receive a 404 error. The exact cause has not been determined.

Other persistent errors involving swagger are that from a remote machine using google chrome the swagger-ui.css file will not load. When using Firefox browser the whole site is unreachable with swaggerUIBundle returning an undefined error. Screenshots in appendices 1, 2:

The swagger documentation loads correctly on the host machine, in attempting to find a fix for this issue multiple bug reports were searched but no comprehensive solution seems to exist. Some articles refer this as a CORS error but offer limited to no solution to fixing this based on the current state of the app as well as how it is generated. i.e. other people with this issue built their app from scratch using class-based development. On testing with different devices, it seems that this is a device configuration related issue as the pages load after accepting and navigating through certificate errors on both chrome and safari on a laptop. This seems to be the case with only one device so far but multiple browsers. There does not seem to be a fix and this issue at the time of writing remains persistent.

#### Syntactical Issues:

These issues have since been resolved but were prevalent throughout the development process. One of the main issues was getting the Knex database connection to function correctly. The documentation can be strenuous to try to find the correct solution. One of which was using the distinct() method to filter returned results when sorting by industry. Some solutions used the method and appended it to the end of the statement. Others used the method to inject raw SQL for the query, after some trial and error it was found easiest to use the distinct method as you would the select method with the same parameters.

Another issue faced was the routing component for the authenticated route. Originally the route was used in the following format:

```
router.get('/stocks/authed/:symbol?from=:from&to=:to')
```

However, this naming convention failed continuously as the route for specific symbols:

```
router.get('/stocks/:symbol')
```

Would match the authenticated route and attempt to parse the query through that routes logic. This bug took considerable searching and time to fix. A solution was found in the following route:

```
router.get('/stocks/authed/:symbol')
```

This route was found to sufficiently match the intended query string with logic contained within to strip the attached queries for *from* and *to* and parse them to check validity.

## Extensions (Optional)

Future extensions for this API could include more comprehensive search queries for specialized data given a detailed enough dataset or even more niche queries server side. A larger dataset would be advantageous to any current stocks modelling rather than a snapshot period of six months however for the limitations of the assignment this was more than enough. An admin section for the API could be interesting to explore as you could edit the database through API queries such as PUT and DELETE but that is probably best left to an internal facing API and not one that is publicly exposed.

## Installation guide

Installing this application:

### Initial Installation:

Assuming you already have this application downloaded unzip the folder.

This can be done in the terminal with the command on Linux: **unzip file.zip -d destination\_folder**

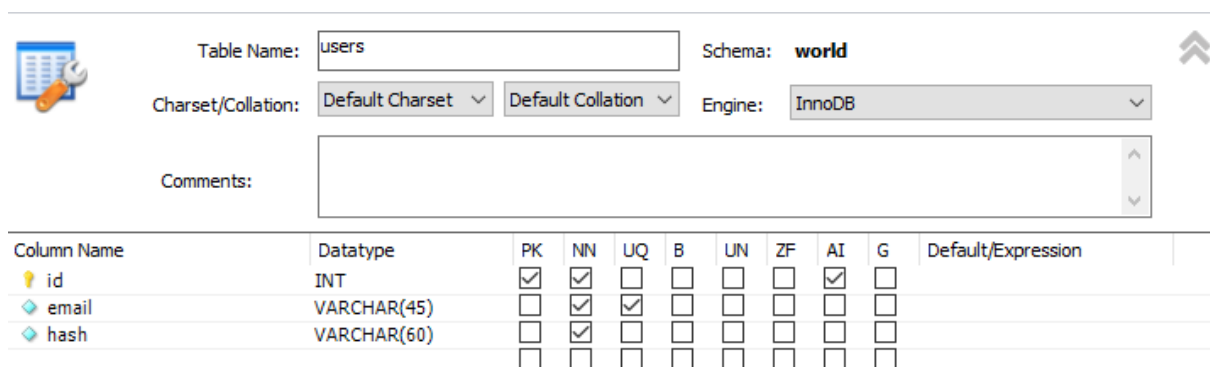
If unzip is not installed run: **sudo apt install unzip**

Change directories into **server-final** and with NodeJS installed run the command **npm install --save**

### Database:

Now before running the application we need to modify some configuration files and create and run a **MySQL** database.

- We will assume that you already have **MySQL** installed, if you do not there are plenty of guides available on how to install it on Linux.
- The application expects two tables to work:
  - o stocks
  - o users
- If you are using the available SQL script please run that, you will then need to create the users table.
  - o The users table expects the following columns and configuration:



The screenshot shows a MySQL database configuration interface. At the top, the 'Table Name' is 'users' and the 'Schema' is 'world'. Below this, 'Charset/Collation' is set to 'Default Charset' and 'Default Collation', and the 'Engine' is 'InnoDB'. A 'Comments' field is empty. Below the configuration fields is a table defining the columns for the 'users' table.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
email	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
hash	VARCHAR(60)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

\*taken from CAB230 Server side JWT Worksheet

### Connecting to the database:

Once the database is created and all configuration is done, we need to tell the application how to talk to the database.

In the root directory there is a `knexfile.js` file. Knex is used to talk and send commands to the database.

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

In the file you will see the above code. This will need to be edited with the database name, if using the script, it will be **webcomputing**, and the password to access the database.

With the default settings done you are now ready to start the server.

Run **npm start**

The server can be accessed at `//localhost:3000/`

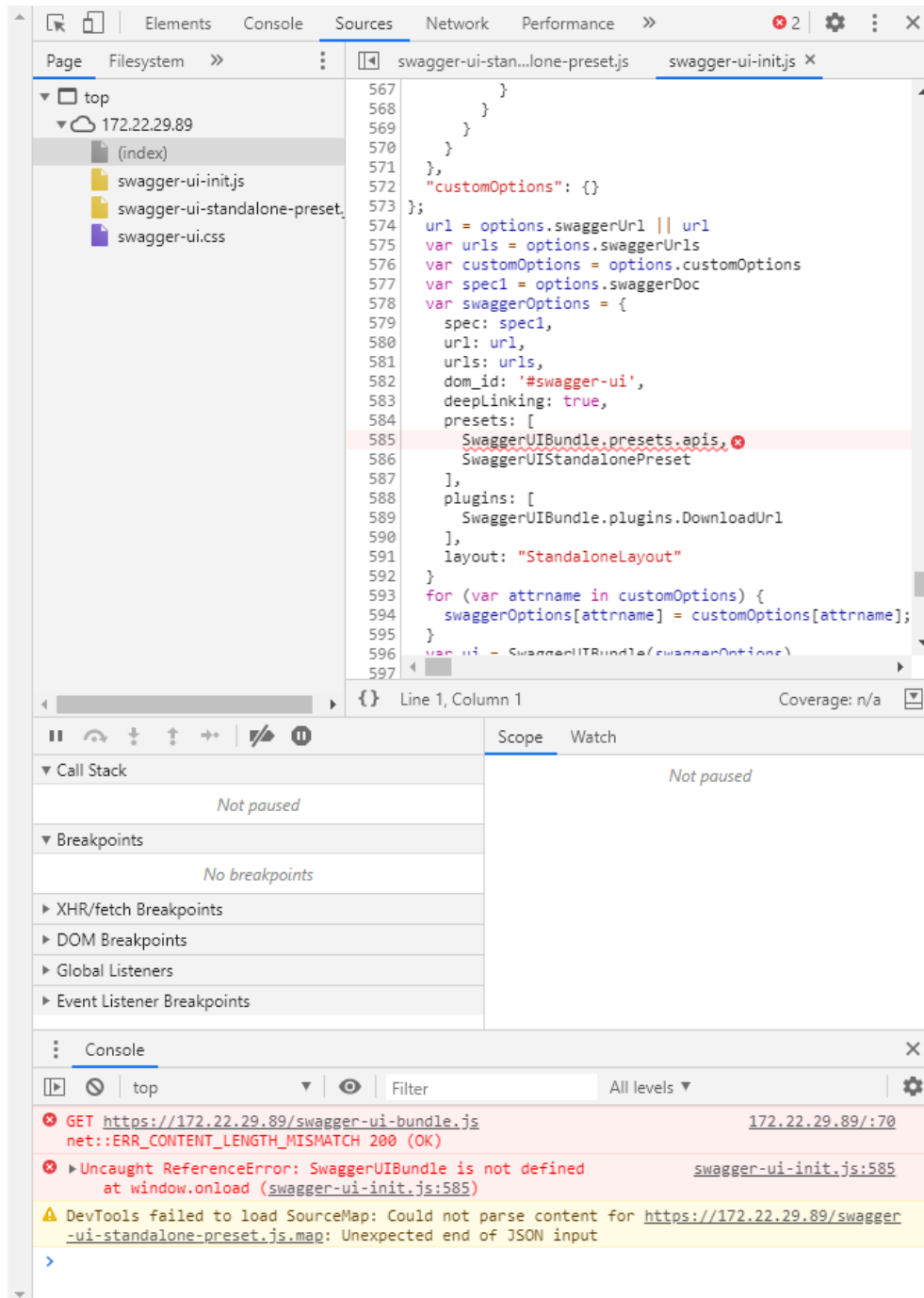
## References

N/A




## Appendices as you require them

1.



2.



## Stocks API


1.0.0

0453

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 /stocks/symbols` to allow you to retrieve data from the database - and two `POST` endpoints to manage 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.

**Servers**

<https://172.22.29.89:443> ▼

Authorize 


**Queries**

▼

[GET /stocks/symbols](#)  
Returns all available stocks, optionally filtered by industry sector.

[GET /stocks/{symbol}](#)  
Returns the latest entry for a particular stock searched by symbol (1-5 upper case letters).

[GET /stocks/authed/{symbol}](#)  
Return entries of stock searched by symbol, optionally filtered by date.

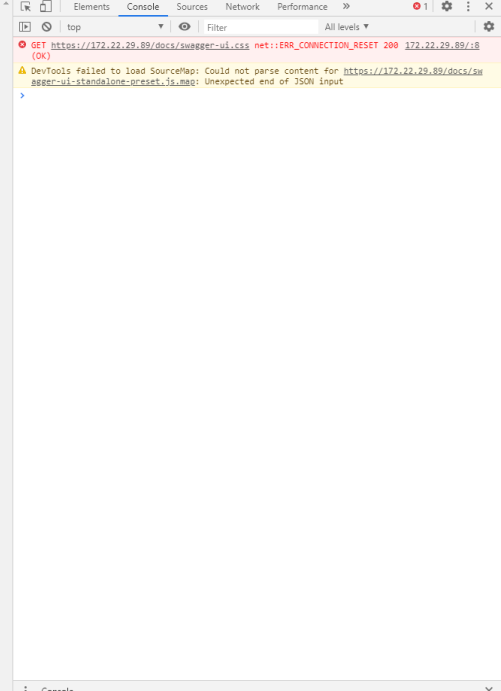


**Users**

▼

[POST /user/register](#)  
Create a new user account

[POST /user/login](#)  
Log in to existing account



Elements Console Sources Network Performance

top Filter All levels

GET <https://172.22.29.89/docs/swagger-ui.css> net::ERR\_CONNECTION\_RESET 200 172.22.29.89/18 (OK)

DevTools failed to load SourceMap: Could not parse content for <https://172.22.29.89/docs/swagger-ui-standalone-preset.js.map>: Unexpected end of JSON input

Console