Cloud Computing &
Cloud Computing Concepts
Coursework

Cloud Software as a Service:
Authenticated Online Auction System and Test Application

ccbay

Mark Lewis
13181409

MSc Data Science (PT)
April 2022

Google Cloud and Docker Installation:

I built a cloud VM called **ccbay-docker** using the standard ubuntu HD, located in Iowa and using 25GB hard drive.

```
mlewis11@ccbay-docker:~$ sudo docker –version
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

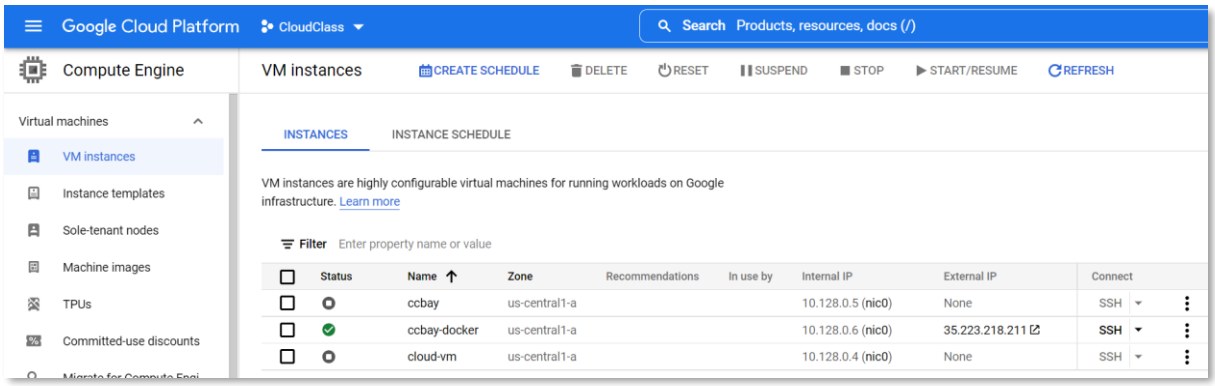


Fig 1. The Cloud VM implementation running with ccbay-docker

With the cloud vm created, I then pushed my app from Visual studio using the command line and Git to an already made GitHub repository:

Github:
https://github.com/m4rk-lewis/ccbay.git

ccbay-github-token:
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

From the google cloud SHH command line, I cloned the github repository to the vm, ran it and tested the application was ruinning by hitting the external IP via google chrome as shown in fig 2.

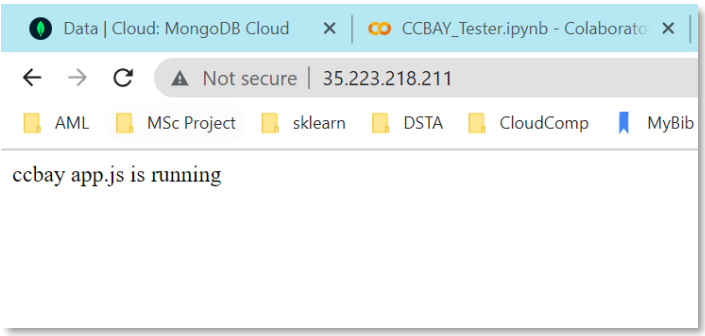


Fig 2. Basic testing of the functioning of the cloud server

Folder Structure:

Folder structure for the project was quite simple. I tried to keep the data flow as simple as possible. Everything is routed from the app.js file using the middleware. Schemas are in the models and functions are in the routers.

I have used the oAuth v2 protocol for user authorisation to prevent non-users from entering or modifying app. In some cases, there are also restrictions applied to what users can see. Viewing uploaded items for example is private for each user and viewing them will only show items uploaded by the currently logged in user. To list these items for every user to see, one must post the item in an auction using the item _id so that only one auction can be posted per item.

Multiple versions of the same item can be uploaded. This is by design to reflect the fact that some users may have numerous identical items and want to sell all of them simultaneously. In this case, the user would simply link an auction to each item _id.

ccbay

- Models:
  - **Auction.js**
  - **Bid.js**
  - **Item.js**
  - **User.js**
- Routes:
  - **auctions.js**
  - **auth.js**
  - **bids.js**
  - **items.js**
- Validations:
  - **Validation.js**
- **.env**
- **App.js**
- **commands.md**
- **verifyToken.js**

When a new auction is posted, the auction is generated, and it also outputs the item data that is pulled from the database relating to that item. This is just a final error check to make sure the auction is linked to the correct item.

MongoDB Database Design:

I kept the MongoDB implementation very simple in just four collections and placed all bids for all auctions into a pool, rather than nesting stored bids within the auction collection. Only the best bid is used to modify the auction along with the user _id for the successful bidder. MongoDB is fast, so there are negligible delays in in this kind of implementation. This means that AuctionID needs to be added to the bid by the bidder.

POST Register New User:

API endpoints (local and cloud):
localhost:3000/api/user/register
35.223.218.211/api/user/register

Body:

```
{
    "username": "nick",
    "email": "nick@gmail.com",
    "password": "123123"
}
```

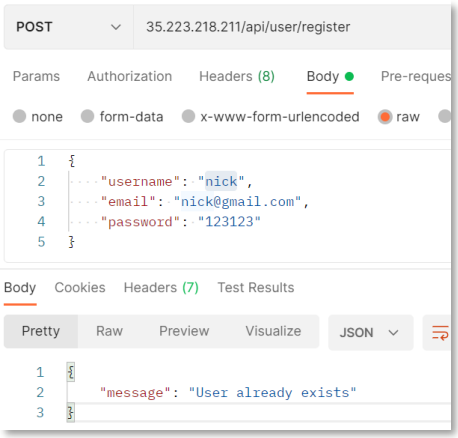
Auth-Token:
Not needed

Output (if user does not exist in DB):


```
{
    "username": "nick",
    "email": "nick@gmail.com",
    "password": "$2a$05$Zslt8gBZiZ.dYbIJ9uy7kuZ1qURXx5KgOMKUC6Xuozd/Dh8BAYC3G",
    "_id": "62659a58c923c02c88e63127",
    "date": "2022-04-24T18:43:36.900Z",
    "__v": 0
}
```


Output (if user exists in DB):

```
{
    "message": "User already exists"
}
```



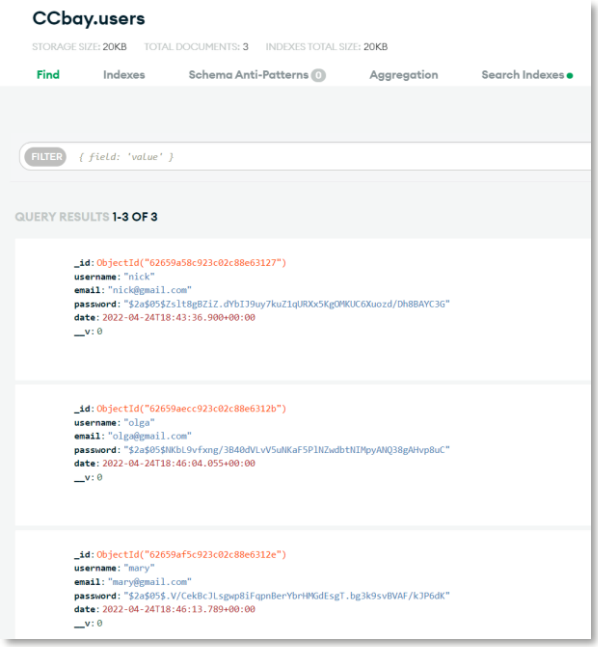


Fig 3 and 4. Output from Postman testing and the data stored on a Mongo DB database

POST User Login

API endpoints (local and cloud):
localhost:3000/api/user/login
35.223.218.211/api/user/login

Body:

```
{
    "email": "nick@gmail.com",
    "password": "123123"
}
```

Auth-Token:
Not needed

Output (if username and pass is correct):

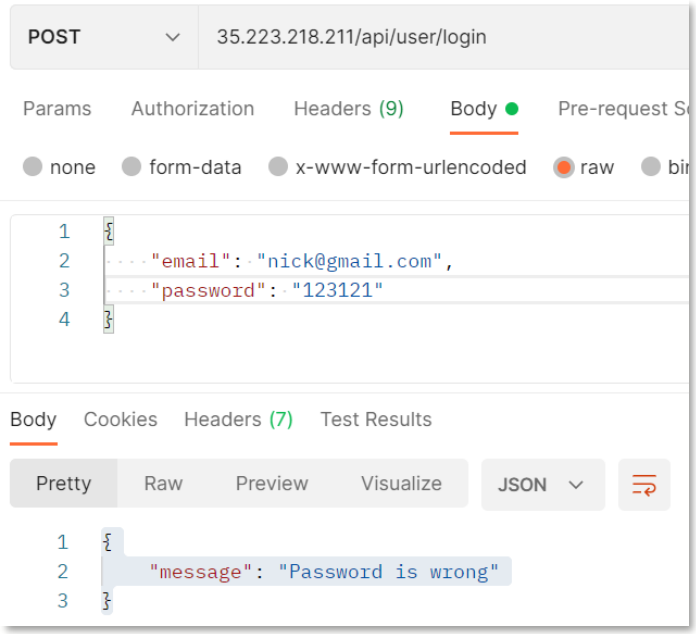

```
{
    "auth-
token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjY1OWE1OGM5MjNjMDJjODhlN
jMxMjciLCJpYXQiOjE2NTA4MjYwOTR9.wKmxdY0kDpMqL9Tn2gigeK7w0FDpqqKAjW0qLSul2XY"
}
```


Output (if pass is not enough digits):

```
{
    "message": "\"password\" length must be at least 6 characters long"
}
```

Output (if pass is wrong):

```
{
    "message": "Password is wrong"
}
```



POST New Item

API endpoints (local and cloud):
localhost:3000/api/item/
35.223.218.211/api/item/post/

Body:
```
{
    "item_title": "Nick's Macbook Pro for sale",
    "item_used": true,
    "item_description": "i'm gonna be honest, I dropped it pretty hard",
    "item_location": "London"
}
```

Auth-Token:
Auth-token from user login

Response (if auth token is correct):
```
{
    "item_title": "Nick's Macbook Pro for sale",
    "item_timestamp": "2022-04-24T18:55:55.294Z",
    "item_used": true,
    "item_description": "i'm gonna be honest, I dropped it pretty hard",
    "item_owner_user_id": "62659a58c923c02c88e63127",
    "item_location": "London",
    "_id": "62659d3bc923c02c88e63134",
    "__v": 0
}
```

Response (if auth token is wrong):
```
{
    "message": "Invalid token"
}
```

Response (if auth token is missing):
```
{
    "message": "Access denied"
}
```

GET all Items (only items uploaded by logged in user)

API endpoints (local and cloud):
localhost:3000/api/item/
35.223.218.211/api/item/

Body:

Auth-Token:
Auth-token from user login

Response (if auth token is correct, item id is correct, and item not already auctioned):


```
[
    {
        "_id": "62659d3bc923c02c88e63134",
        "item_title": "Nick's Macbook Pro for sale",
        "item_timestamp": "2022-04-24T18:55:55.294Z",
        "item_used": true,
        "item_description": "i'm gonna be honest, I dropped it pretty hard",
        "item_owner_user_id": "62659a58c923c02c88e63127",
        "item_location": "London",
        "__v": 0
    }
]
```


Response (if auth token is wrong):


```
{
    "message": "Invalid token"
}
```


Response (if auth token is missing):


```
{
    "message": "Access denied"
}
```

POST New Auction (linking to uploaded item only)

API endpoints (local and cloud):
localhost:3000/api/auction/post/
35.223.218.211/api/auction/post/

Body:
```
{
    "auction_item_id": "6264737d662e5a5f63161ccc",
    "auction_expiration_date": "2022-05-01T12:36:06.591Z",
    "auction_live_status": true
}
```

Auth-Token:
Needed from login

Response (if auth token is correct, item id is correct, and item not already auctioned):
```
{
    "item_info": [
        {
            "_id": "62659d3bc923c02c88e63134",
            "item_title": "Nick's Macbook Pro for sale",
            "item_timestamp": "2022-04-24T18:55:55.294Z",
            "item_used": true,
            "item_description": "i'm gonna be honest, I dropped it pretty hard",
            "item_owner_user_id": "62659a58c923c02c88e63127",
            "item_location": "London",
            "__v": 0
        }
    ],
    "auction_details": {
        "auction_item_title": "Nick's Macbook Pro for sale",
        "auction_item_id": "62659d3bc923c02c88e63134",
        "auction_best_bid": 0,
        "auction_best_bidder_id": "No Biders Yet",
        "auction_live_status": true,
        "auction_expiration_date": "2022-04-29T12:36:06.591Z",
        "_id": "62659eccc923c02c88e6313d",
        "__v": 0
    }
}
```
Response (if auction_item_id is wrong):
```
{
    "message": "Item does not exist. Please Post new item"
}
```
Response (if item is already auctioned):
```
{
    "message": "Auction already exists"
}
```

Response (if auth token is wrong):

```
{
    "message": "Invalid token"
}
```

Response (if auth token is missing):

```
{
    "message": "Access denied"
}
```

POST 35.223.218.211/api/auction/post/

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1  {
2      "auction_item_id": "62659d3bc923c02c88e63134",
3      "auction_expiration_date": "2022-04-29T12:36:06.591Z",
4      "auction_live_status": true
5  }
```

ody Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1  {
2      "item_info": [
3          {
4              "_id": "62659d3bc923c02c88e63134",
5              "item_title": "Nick's Macbook Pro for sale",
6              "item_timestamp": "2022-04-24T18:55:55.294Z",
7              "item_used": true,
8              "item_description": "i'm gonna be honest, I dropped it pretty hard",
9              "item_owner_user_id": "62659a58c923c02c88e63127",
10             "item_location": "London",
11             "__v": 0
12         }
13     ],
14     "auction_details": {
15         "auction_item_title": "Nick's Macbook Pro for sale",
16         "auction_item_id": "62659d3bc923c02c88e63134",
17         "auction_best_bid": 0,
18         "auction_best_bidder_id": "No Biders Yet",
19         "auction_live_status": true,
20         "auction_expiration_date": "2022-04-29T12:36:06.591Z",
21         "_id": "62659eccc923c02c88e6313d",
22         "__v": 0
23     }
24 }
```

GET SINGLE AUCTION by auction_item_id (shows time remaining):

API endpoints (local and cloud):
localhost:3000/api/auction/
35.223.218.211/api/auction/

Body:
```
{
    "auction_item_id": "62659d3bc923c02c88e63134"
}
```

Auth-Token:
Needed from user login

Response (if auth token is correct):
```
{
_id: new ObjectId("62659eccc923c02c88e6313d"),
auction_item_title: "Nick's Macbook Pro for sale",
auction_item_id: '62659d3bc923c02c88e63134',
auction_best_bid: 0,
auction_best_bidder_id: 'No Biders Yet',
auction_live_status: true,
auction_expiration_date: 2022-04-29T12:36:06.591Z,
__v: 0
}{
_id: new ObjectId("62659d3bc923c02c88e63134"),
item_title: "Nick's Macbook Pro for sale",
item_timestamp: 2022-04-24T18:55:55.294Z,
item_used: true,
item_description: "i'm gonna be honest, I dropped it pretty hard",
item_owner_user_id: '62659a58c923c02c88e63127',
item_location: 'London',
__v: 0
}Auction Time Remaining: 113 hours, 22 minutes, 50 seconds
```

Response (if auth token is wrong):
```
{
    "message": "Invalid token"
}
```

Response (if auth token is missing):
```
{
    "message": "Access denied"
}
```

Response (if auth token is correct but auction has ended):


```
{
_id: new ObjectId("62659eccc923c02c88e6313d"),
auction_item_title: "Nick's Macbook Pro for sale",
auction_item_id: '62659d3bc923c02c88e63134',
auction_best_bid: 77.99,
auction_best_bidder_id: '62659af5c923c02c88e6312e',
auction_live_status: true,
auction_expiration_date: 2022-03-29T12:36:06.591Z,
__v: 0
}{
_id: new ObjectId("62659d3bc923c02c88e63134"),
item_title: "Nick's Macbook Pro for sale",
item_timestamp: 2022-04-24T18:55:55.294Z,
item_used: true,
item_description: "i'm gonna be honest, I dropped it pretty hard",
item_owner_user_id: '62659a58c923c02c88e63127',
item_location: 'London',
__v: 0
}Auction has ended
```

PATCH post bid

API endpoints (local and cloud):
localhost:3000/api/Bid/new
35.223.218.211/api/Bid/new

Body:
```
{
    "auction_item_id": "62659d3bc923c02c88e63134",
    "bid_new_bid": 77.99
}
```

Auth-Token:
Needed from user login

Response (if auth token is correct):
```
{
    "updateBidByID": {
        "acknowledged": true,
        "modifiedCount": 1,
        "upsertedId": null,
        "upsertedCount": 0,
        "matchedCount": 1
    },
    "savedBids": {
        "auction_id": "62659d3bc923c02c88e63134",
        "bid_new_bid": 77.99,
        "bid_bidder_id": "62659af5c923c02c88e6312e",
        "bid_timetamp": "2022-04-24T19:28:10.482Z",
        "_id": "6265a4ca0c5aae8e80674344",
        "__v": 0
    }
}
```

Response (if auth token is wrong):
```
{
    "message": "Invalid token"
}
```
Response (if auth token is missing):
```
{
    "message": "Access denied"
}
```
Response (logged in user is the owner of the item listed):
```
{
    "message": "You cannot bid on your own auction"
}
```
Response (if not owner but bid is smaller than current best bid):
```
{
    "message": "Bid must be larger than current bid"
}
```

GET timestamped bid history using auction_item_id

API endpoints (local and cloud):
localhost:3000/api/Bid/
35.223.218.211/api/bid/
Body:

```
{
    "auction_item_id": "62659d3bc923c02c88e63134",
    "bid_new_bid": 77.99
}
```

Auth-Token:
Needed from user login

Response (if auth token is correct):

```
{
    "updateBidByID": {
        "acknowledged": true,
        "modifiedCount": 1,
        "upsertedId": null,
        "upsertedCount": 0,
        "matchedCount": 1
    },
    "savedBids": {
        "auction_id": "62659d3bc923c02c88e63134",
        "bid_new_bid": 77.99,
        "bid_bidder_id": "62659af5c923c02c88e6312e",
        "bid_timetamp": "2022-04-24T19:28:10.482Z",
        "_id": "6265a4ca0c5aae8e80674344",
        "__v": 0
    }
}
```

Response (if auth token is wrong):

```
{
    "message": "Invalid token"
}
```

Response (if auth token is missing):

```
{
    "message": "Access denied"
}
```

Response (logged in user is the owner of the item listed):

```
{
    "message": "You cannot bid on your own auction"
}
```

Response (if not owner but bid is smaller than current best bid):

```
{
    "message": "Bid must be larger than current bid"
}
```

Response (everything is correct, but auction has finished):

```
{
    "message": "Auction has completed"
}
```

GET 35.223.218.211/api/bid/

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "auction_id": "62659d3bc923c02c88e63134"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6265a4a50c5aae8e8067432f",
        "auction_id": "62659d3bc923c02c88e63134",
        "bid_new_bid": 39.99,
        "bid_bidder_id": "62659af5c923c02c88e6312e",
        "bid_timetamp": "2022-04-24T19:27:33.736Z",
        "__v": 0
    },
    {
        "_id": "6265a4c10c5aae8e80674336",
        "auction_id": "62659d3bc923c02c88e63134",
        "bid_new_bid": 43.99,
        "bid_bidder_id": "62659af5c923c02c88e6312e",
        "bid_timetamp": "2022-04-24T19:28:01.248Z",
        "__v": 0
    },
    {
        "_id": "6265a4c50c5aae8e8067433d",
        "auction_id": "62659d3bc923c02c88e63134",
        "bid_new_bid": 49.99,
        "bid_bidder_id": "62659af5c923c02c88e6312e",
        "bid_timetamp": "2022-04-24T19:28:05.979Z",
        "__v": 0
    },
    {
        "_id": "6265a4ca0c5aae8e80674344",
        "auction_id": "62659d3bc923c02c88e63134",
        "bid_new_bid": 77.99,
        "bid_bidder_id": "62659af5c923c02c88e6312e",
        "bid_timetamp": "2022-04-24T19:28:10.482Z",
        "__v": 0
    }
```

DELETE auction using auction id

API endpoints (local and cloud):
localhost:3000/api/auction/delete
35.223.218.211/api/auction/delete`

Body:
```
{
    "auction_id": "62659eccc923c02c88e6313d"
}
```

Auth-Token:
Needed from user login

Response (if auth token is correct and auction_id correct):
```
{
    "acknowledged": true,
    "deletedCount": 1
}
```

Response (if auth token is correct and auction_id wrong):
```
{
    "acknowledged": true,
    "deletedCount": 0
}
```

Response (if auth token is wrong):
```
{
    "message": "Invalid token"
}
```

Response (if auth token is missing):
```
{
    "message": "Access denied"
}
```

DELETE item using item id

API endpoints (local and cloud):
localhost:3000/api/item/delete
35.223.218.211/api/item/delete

Body:
```
{
    "item_id": "62654494af27b5ae92319407"
}
```

Auth-Token:
Needed from user login

Response (if auth token is correct and item_id correct):
```
{
    "acknowledged": true,
    "deletedCount": 1
}
```

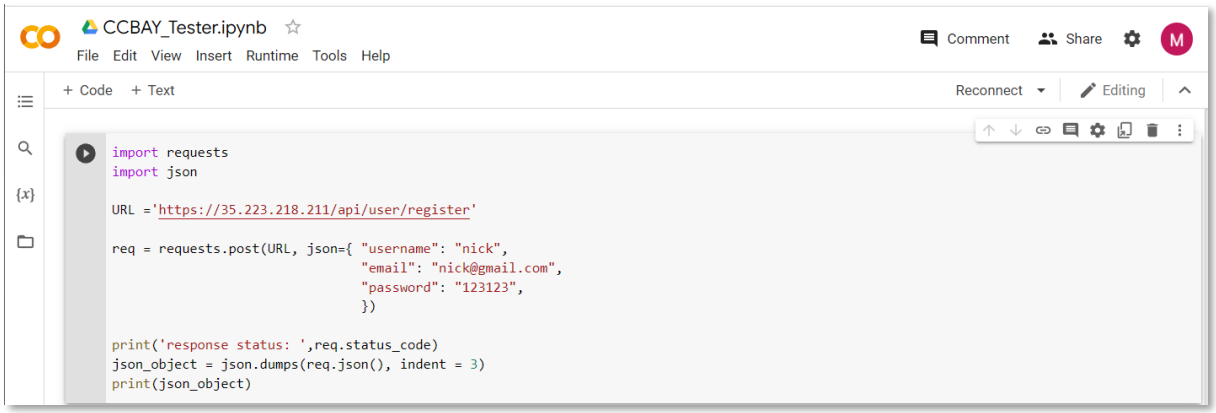
Response (if auth token is correct and item_id wrong):
```
{
    "acknowledged": true,
    "deletedCount": 0
}
```

Response (if auth token is wrong):
```
{
    "message": "Invalid token"
}
```

Response (if auth token is missing):
```
{
    "message": "Access denied"
}
```

External Testing Solution:

I have included a basic python testing program, coded within Google Colab and connect to the API via the requests and json imported packages. The outputs from the web requests shows the status and the response, so a status of 200 demonstrates that the request was successful.

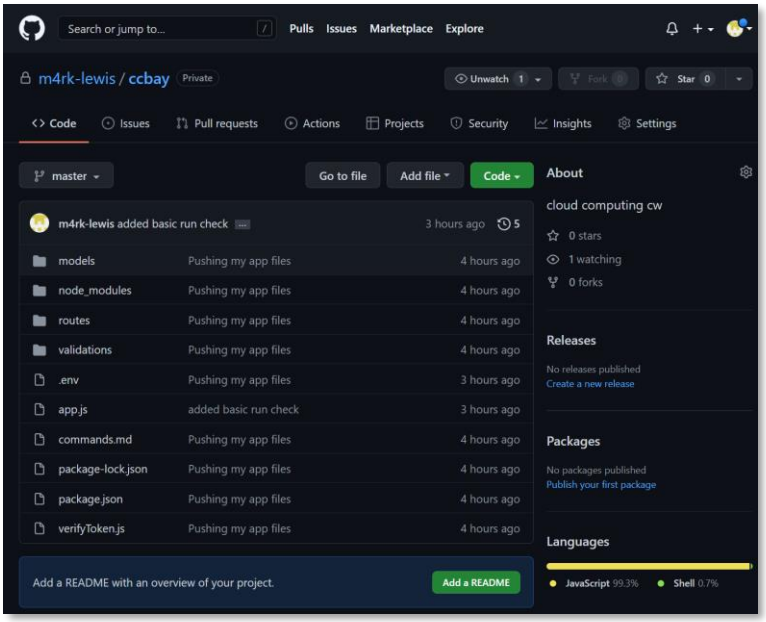


Conclusion:

I will admit, I ran out of time to implement all the features that I wished to implement. It would have been nice to add a web interact that ran all the commands etc. so that it looked more like eBay. There were some aspects that I struggled with and did not quite resolve. I would have liked each time EVERY auction was listed, for it to have shown time remaining without needing to look at the detailed single action view. I would also have liked to implement the ability to only view live auctions and only view closed auctions using filter on the mongo dB selection queries.

Output code for docker image building and cloning from GitHub:

git remote add origin https://m4rk.lewis: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@github.com/m4rk-lewis/ccbay.git
git clone --branch master https://m4rk.lewis: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@github.com/m4rk-lewis/ccbay.git
docker image build -t ccbay-image:2 .
6fe4b7bc8171efe017f101081337d9290e3dda5a3618e0e802fcc640aa355f21



References:

[1] https://vitux.com/how-to-make-a-user-an-administrator-in-ubuntu/
[2] https://linuxize.com/post/how-to-install-node-js-on-ubuntu-18.04/
[3] https://www.ionos.co.uk/digitalguide/websites/web-development/nodejs-for-a-website-with-apache-on-ubuntu/
[4] https://www.geeksforgeeks.org/unit-testing-of-node-js-application/
[5] https://stackoverflow.com/questions/25250551/how-to-generate-timestamp-unix-epoch-format-nodejs