



CSCI 5409 Advanced Topics in Cloud Computing

Assignment-3

March 25, 2020

Submitted By

Vamsi Gamidi

B00834696

VM709690

Job-1: Report

Extraction:

Initially, I have downloaded the files from the given website [7] and extracted the book names and author names by using a python script. I have also introduced a 5-minute delay for processing each file.

Database:

By using the python script, I have saved the book names and author names in a mongodb database in EC2 instance [3]. I have also stored the start_time and end_time for processing each file in a separate collection.

API's and Containers:

By following the given architecture, I have created three API's for search functionality, notes retrieval functionality and save notes functionality. When user enters a keyword into search bar and clicks on search button, server running on port 5000 will be listening to it and receives the keyword. The keyword will be used to query for records in the mongodb. Upon successful retrieval of records for the keyword, it will be stored in the log file and returned to the controller as a response which will be displayed in a tabular format.

When the user enters notes for a keyword, the notes along with the keyword will be received by the server as a request. Server will store the notes and keyword in the notes.json file. Similarly, if the user enters a keyword and clicks on retrieve notes button, server will receive the keyword as input and searches in the notes.json file matching the records with the keyword. All the matching notes will be sent as a response to the controller and will be displayed in a tabular format.

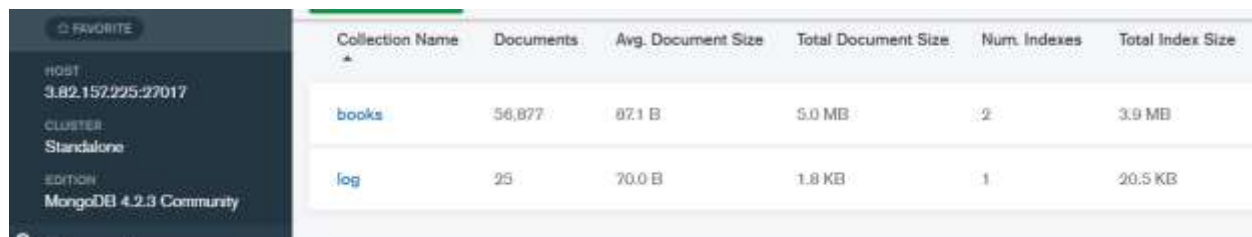
I have written a Dockerfile and package.json file which hold the configuration required for the application to run [4]. The following are the contents of the Dockerfile:

- FROM node:10.16.3
- WORKDIR /usr/src/app
- COPY package*.json ./
- RUN npm install
- COPY . .
- EXPOSE 5000
- CMD ["node", "server.js"]

By using the above files, I have created a docker image and pushed it to docker hub. I have pulled the image from dockerhub and used it in EC2 instance to deploy the application [4].

Job-2: Extraction Engine

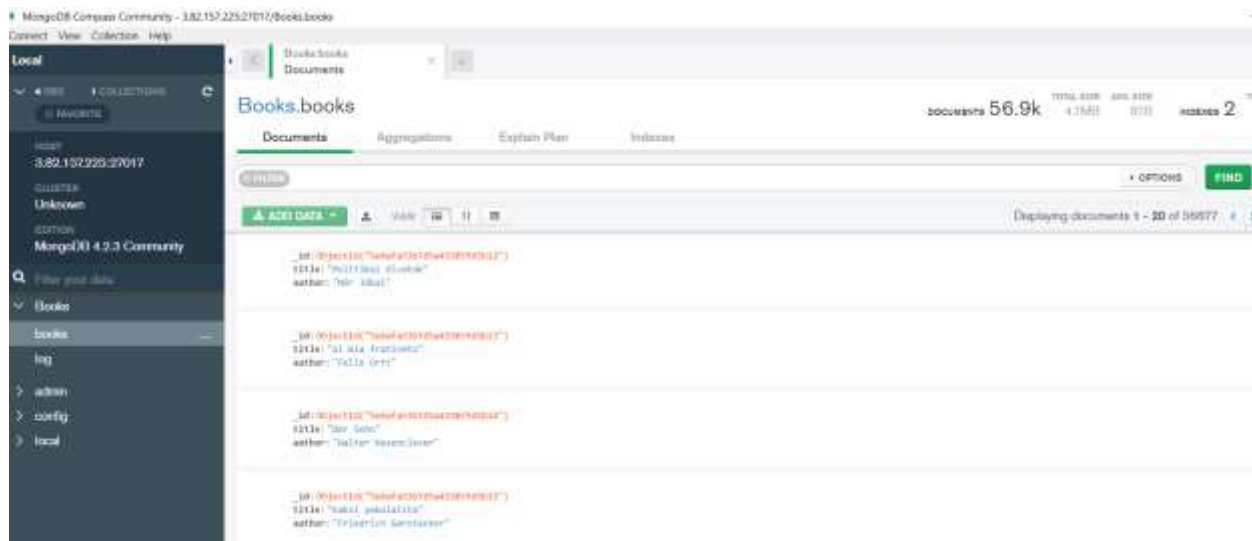
To extract the book and author details, I have downloaded the GUTINDEX files from Gutenberg.org [7] and I have written a python script to extract the book title and book author. After extracting those details, I have inserted the book details into a mongodb collection 'books' in AWS instance with public ip '3.82.157.225'. I have introduced a 5-minute delay for processing each file and stored the start_time, end_time, book_title in another mongodb collection 'log'. Both the collections are stored in the mongodb database 'Books'.



The screenshot shows the MongoDB Compass interface. On the left, a sidebar displays the connection details: Host 3.82.157.225:27017, Cluster Standalone, and Edition MongoDB 4.2.3 Community. The main panel shows a table of collections in the 'Books' database. The table has columns: Collection Name, Documents, Avg. Document Size, Total Document Size, Num. Indexes, and Total Index Size. Two collections are listed: 'books' with 56,877 documents, an average size of 87.1 B, a total size of 5.0 MB, 2 indexes, and a total index size of 3.9 MB; and 'log' with 25 documents, an average size of 70.0 B, a total size of 1.8 KB, 1 index, and a total index size of 20.5 KB.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
books	56,877	87.1 B	5.0 MB	2	3.9 MB
log	25	70.0 B	1.8 KB	1	20.5 KB

Figure 1: books and log collections in the Database



The screenshot shows the MongoDB Compass interface with the 'books' collection selected. The left sidebar shows the database 'Books' and the collection 'books'. The main panel displays the 'books' collection with a 'Documents' tab selected. The top right shows statistics: documents 56.9k, total size 4.7MB, avg. size 87.1B, and indexes 2. Below the statistics, there is a search bar and a 'FIND' button. The main area displays a list of documents. Each document is shown as a JSON object with fields: _id, title, and author. The first document has title 'Pittiers d'Europe' and author 'John Hall'. The second document has title 'Al via frangente' and author 'Gillo Gatti'. The third document has title 'Der Sohn' and author 'Walter Hasenclever'. The fourth document has title 'Kauz geballt' and author 'Friedrich Schiller'.

Document
<pre>{ "_id": "Pittiers d'Europe", "title": "Pittiers d'Europe", "author": "John Hall" }</pre>
<pre>{ "_id": "Al via frangente", "title": "Al via frangente", "author": "Gillo Gatti" }</pre>
<pre>{ "_id": "Der Sohn", "title": "Der Sohn", "author": "Walter Hasenclever" }</pre>
<pre>{ "_id": "Kauz geballt", "title": "Kauz geballt", "author": "Friedrich Schiller" }</pre>

Figure 2: books collection in Database

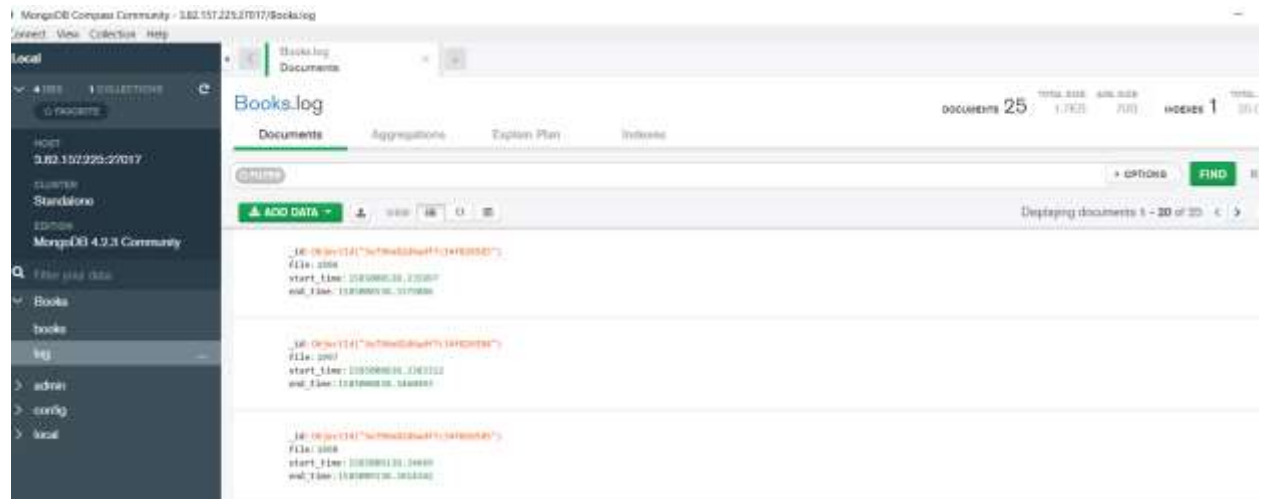


Figure 3: log collection with start_time and end_time

Job-3: Web Page

I have created a web application using MEAN stack i.e. Node JS and Express JS for backend, Angular JS for frontend and MongoDB as database. The webpage contains a search bar, search button and retrieve notes button initially (figure 4). User can enter a keyword in the search bar and can click the search button or retrieve notes button. If the user clicks on the search button, the keyword will be searched in the database, all the records matching the keyword will be retrieved and shown in a tabular format in the webpage (figure 5). After every successful retrieval of data from database, all the entries will be saved into log.json file in the jsonfiles folder. User can also add notes in the text area for a specific keyword. We can also retrieve the saved notes by clicking on retrieve notes button (figure 6). All the added notes will be saved to notes.json file in the json files folder.

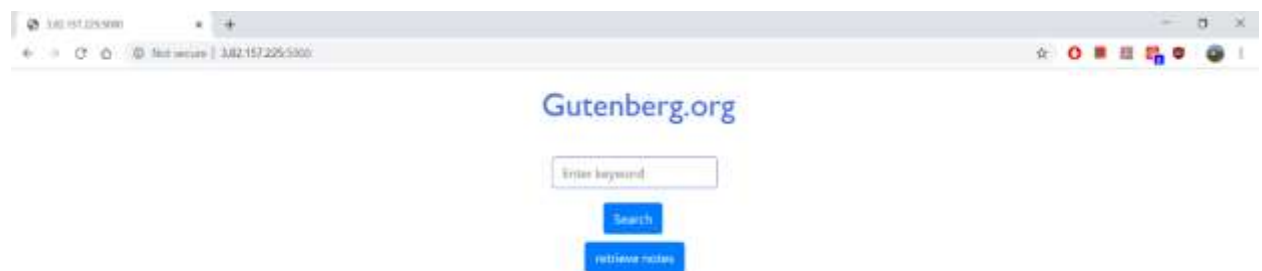


Figure 4: Home Page

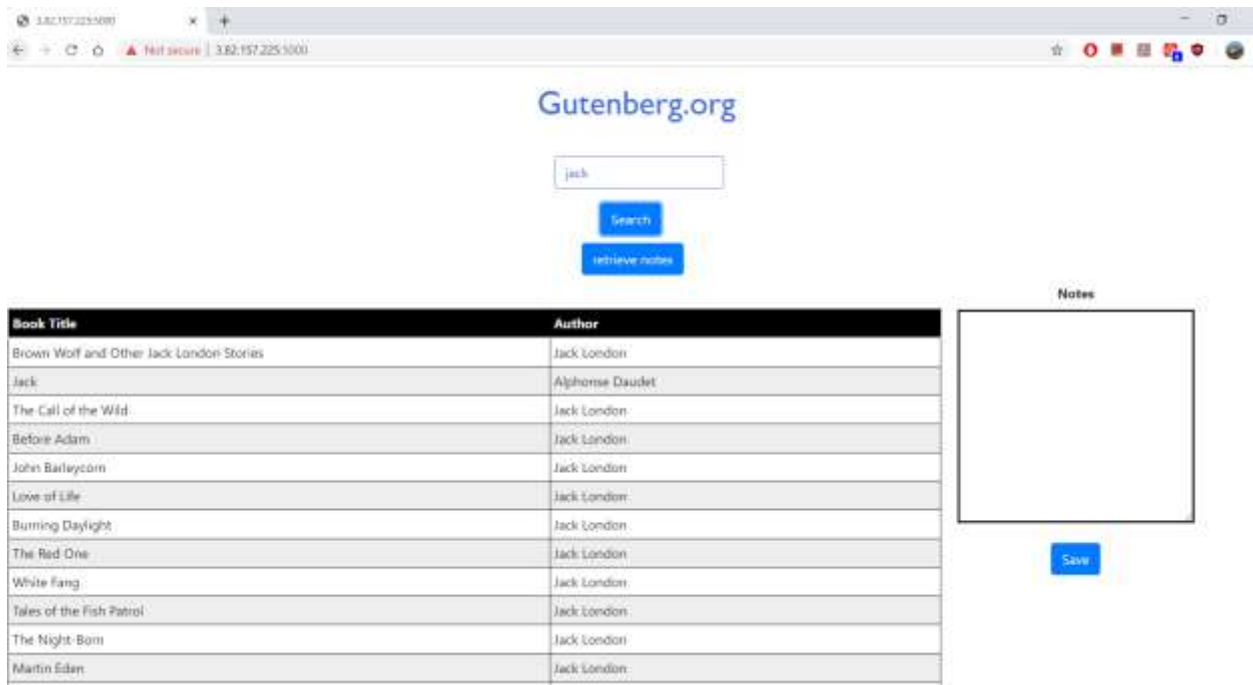


Figure 5: Search Results

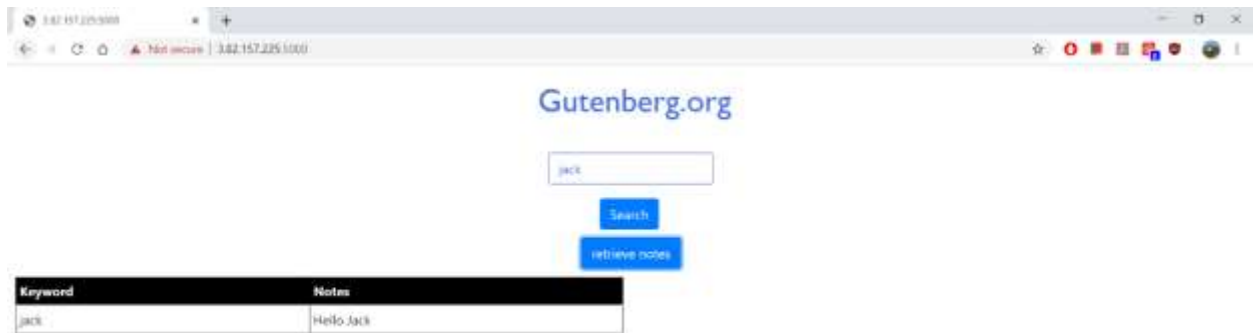
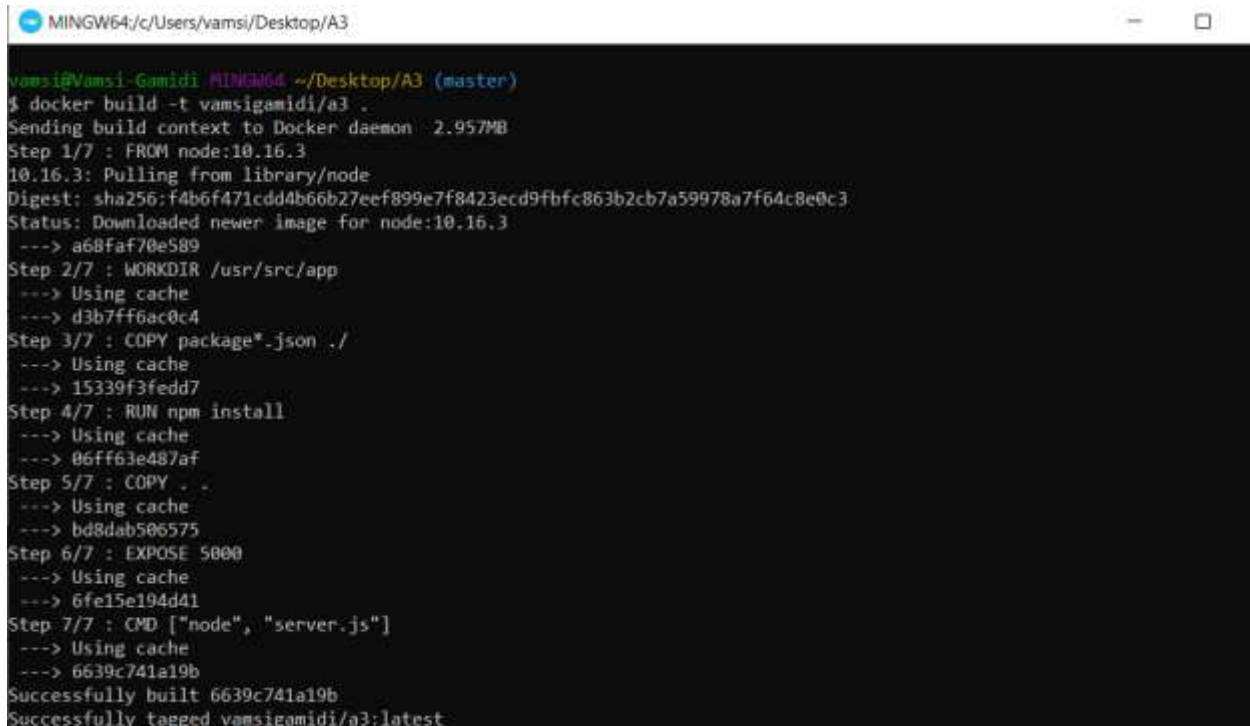


Figure 6: Notes Retrieval

Job-4: Docker Containers

I have created a Dockerfile which has all the configurations related to the application. After installing docker and virtual box, I have created docker image (figure 7) and pushed the image to docker hub (figure 9).



```
MINGW64:/c:/Users/vamsi/Desktop/A3
vamsi@Vamsi-Gamidi MINGW64 ~/Desktop/A3 (master)
$ docker build -t vamsigamidi/a3 .
Sending build context to Docker daemon 2.957MB
Step 1/7 : FROM node:10.16.3
10.16.3: Pulling from library/node
Digest: sha256:f4b6f471cdd4b66b27eef899e7f8423ecd9fbfc863b2cb7a59978a7f64c8e0c3
Status: Downloaded newer image for node:10.16.3
--> a68faf70e589
Step 2/7 : WORKDIR /usr/src/app
--> Using cache
--> d3b7ff6ac0c4
Step 3/7 : COPY package*.json ./
--> Using cache
--> 15339f3fedd7
Step 4/7 : RUN npm install
--> Using cache
--> 06ff63e487af
Step 5/7 : COPY . .
--> Using cache
--> bd8dab506575
Step 6/7 : EXPOSE 5000
--> Using cache
--> 6fe15e194d41
Step 7/7 : CMD ["node", "server.js"]
--> Using cache
--> 6639c741a19b
Successfully built 6639c741a19b
Successfully tagged vamsigamidi/a3:latest
```

Figure 7: Docker build image

In figure 8, I have created docker container with id 47df97323935 and image vamsigamidi/a3 which is ready to be pushed to docker hub [6].

```
MINGW64/c/Users/vamsi/Desktop/A3
vamsi@Vamsi-Gamidi MINGW64 ~/Desktop/A3 (master)
$ docker run -p 5000:5000 -d vamsigamidi/a3
47df973239351e90b45b80ce8b1508ea26e479bfd30a2b6ded74a70106df2c69

vamsi@Vamsi-Gamidi MINGW64 ~/Desktop/A3 (master)
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
47df97323935        vamsigamidi/a3     "docker-entrypoint.s..." 6 seconds ago       Up 5 seconds       0.0.0.0:5000->5000/tcp
204bde236608        vamsigamidi/a3     "docker-entrypoint.s..." 6 minutes ago       Up 6 minutes       5000/tcp, 0.0.0.0:49160->8080/tcp
vamsi@Vamsi-Gamidi MINGW64 ~/Desktop/A3 (master)
$
```

Figure 8: Docker run

```
MINGW64/c/Users/vamsi/Desktop/A3
vamsi@Vamsi-Gamidi MINGW64 ~/Desktop/A3 (master)
$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: vamsigamidi18
Password:
WARNING! Your password will be stored unencrypted in C:\Users\vamsi\.docker\config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-helper
Login Succeeded

vamsi@Vamsi-Gamidi MINGW64 ~/Desktop/A3 (master)
$ docker tag vamsigamidi/a3 vamsigamidi18/a3:v1
vamsi@Vamsi-Gamidi MINGW64 ~/Desktop/A3 (master)
$ docker push vamsigamidi18/a3:v1
The push refers to repository [docker.io/vamsigamidi18/a3]
cb0c5043f08: Pushed
624f709fc881: Pushed
4ef45882cbe8: Pushed from library/node
dca4df481ec3: Pushed from library/node
672a7e5537e1: Pushed from library/node
4870070910e: Pushed from library/node
070d1238a77: Pushed from library/node
v1: digest: sha256:241710c1b09cc3109a1794a11be7abff49380f01b425ec6ce81ab4cdae4 size: 3051
```

Figure 9: Docker Push

In figure 9, I have logged into dockerhub account and pushed the image to dockerhub [4].



Figure 10: Docker Hub

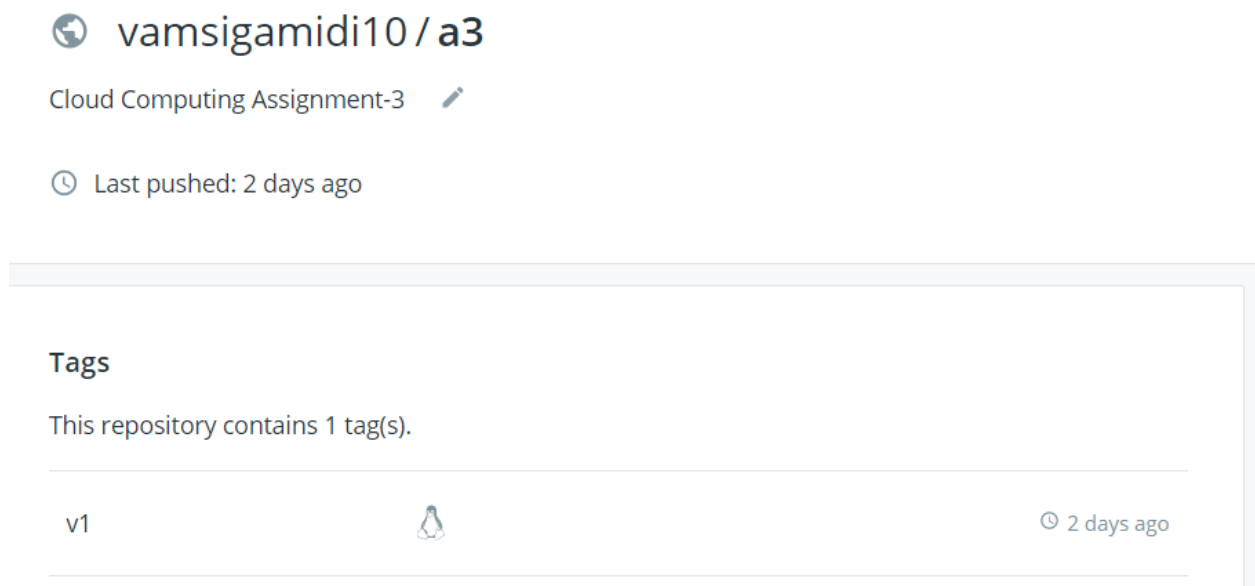


Figure 11: Docker hub tag

Job-5: Deployment

To deploy the application in ec2, I have installed docker in my ec2 instance and logged Docker Hub and pulled the image I have pushed earlier (figure 13) [4].



Figure 12: EC2 Instance

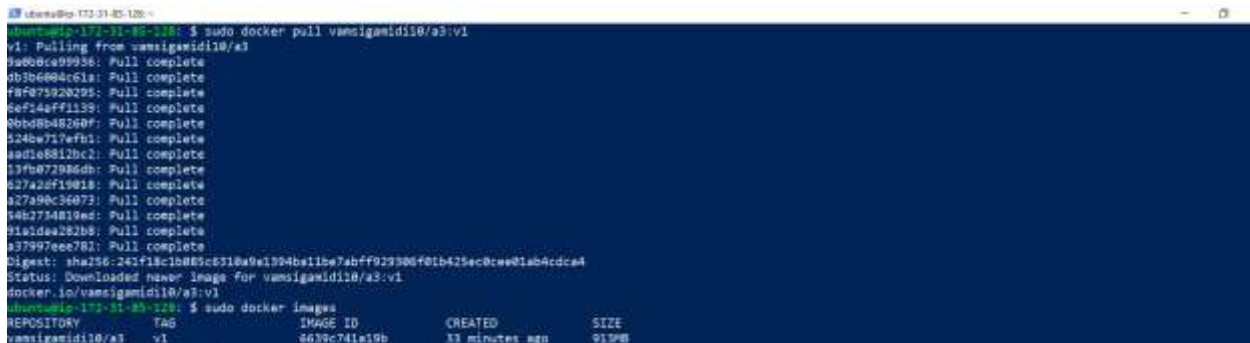


Figure 13: Docker Pull

I have executed the docker run command to create the container which can be accessed using the port 5000 (figure 14).



Figure 14: Docker Run



Figure 15: Docker PS

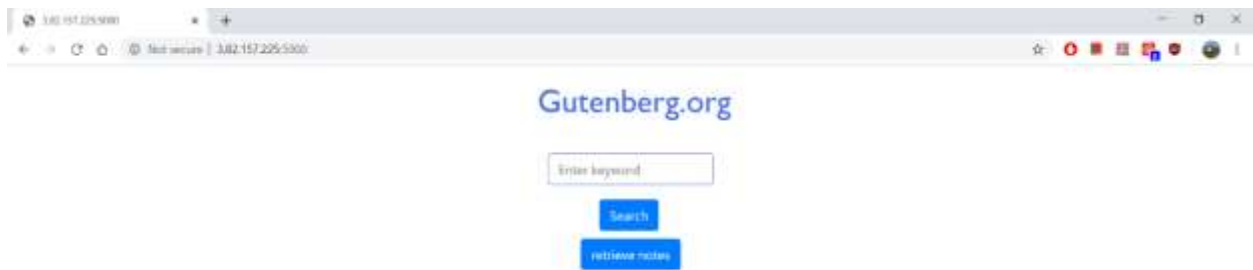


Figure 16: Application Deployed

In figure 16, we can see the application deployed and can be accessed by using the public ip of ec2 instance and port number 5000.

Job-6: Testing

Test Case 1: The basic functionality of this application is to take a keyword as input and display the records matched in the database in a tabular format. I have searched for the word 'jack' and displayed the records.

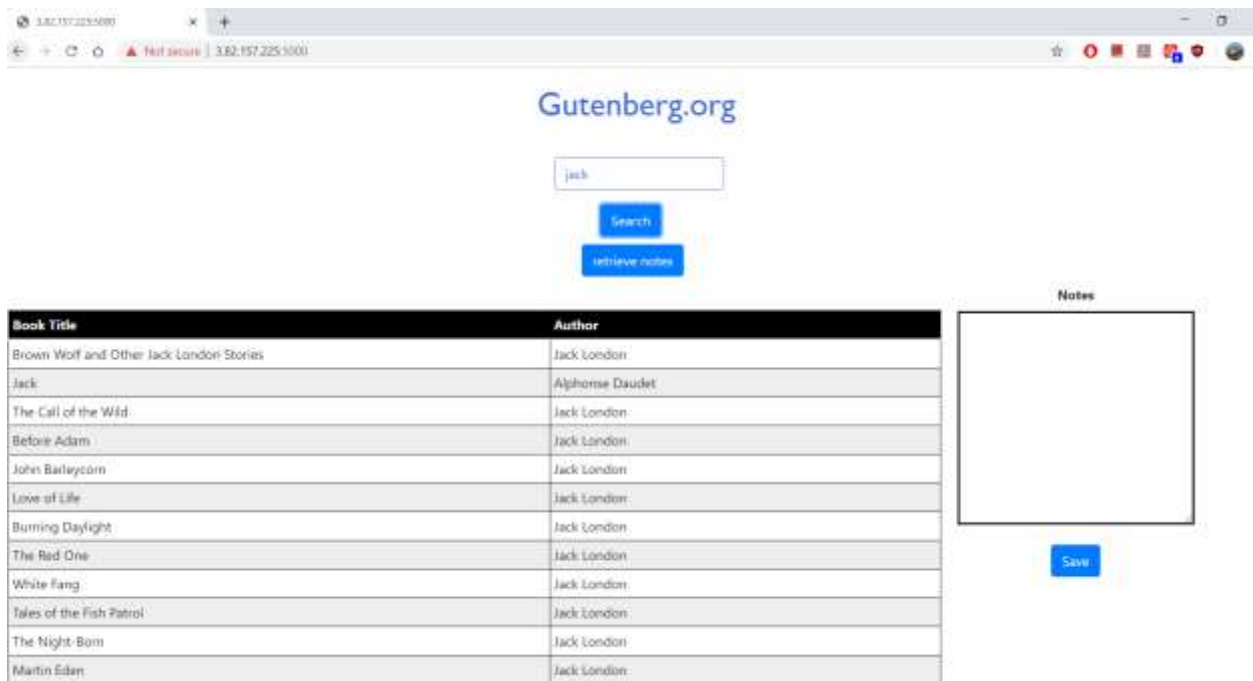


Figure 17: Search Results for keyword 'jack'

Test Case 2: For a successful keyword match, user should be able to save notes into notes.json file and should be able to retrieve them.

Notes

The most popular book of jack
london is white fang.

Figure 18: Taking notes as input

← → ↻ 🔍 Not secure | 3.82.157.225:5000

Gutenberg.org

Keyword	Notes
jack	Hello Jack
jack	The most popular book of jack london is white fang.

Figure 19: Retrieved notes

```
root@55d573d89d46:/usr/src/app/json_files# tail -f log.json
[{"keyword": "james", "count": 4, "timestamp": [1584132178480, 1584132326181, 1584132554819, 1584739278314]}, {"keyword": "sasa", "count": 1, "timestamp": [1584132477974]}, {"keyword": "hilo", "count": 1, "timestamp": [1584132491825, 1584132497943]}, {"keyword": "john den", "count": 17, "timestamp": [1584737539888, 1584738276994, 1584818428396, 1584818432609, 1584832897910, 1584833579829, 1584835729381, 1584835734049, 1584835817509, 1584836185870, 1584836188244, 1584836381722, 1584836468204, 1584836588250, 1584837508398, 1584847687459, 15848547780874]}, {"keyword": "john", "count": 10, "timestamp": [1584758495299, 1584758661843, 1584818078887, 1584819595192, 1584832289668, 1584832831707, 1584836838837, 1584836290441, 1584836426759, 1584836429712]}, {"keyword": "john den", "count": 2, "timestamp": [1584820077755, 1584832544372]}, {"keyword": "jack", "count": 4, "timestamp": [1584832871116, 1584833000955, 1584836001047, 1584836188935]}, {"keyword": "jack", "count": 1, "timestamp": [1584837578588]}, {"keyword": "jack reacher", "count": 1, "timestamp": [1584837590612]}, {"keyword": "donald", "count": 1, "timestamp": [1584837676722, 1584837732555]}]C
root@55d573d89d46:/usr/src/app/json_files# tail -f notes.json
{"notes": [{"keyword": "james", "note": "Hello james"}, {"keyword": "hello", "note": "fgdfgdfg"}, {"keyword": "john", "note": "Hello John"}, {"keyword": "donald", "note": "Hello donald"}, {"keyword": "john den", "note": "siksiks"}, {"keyword": "john den", "note": "Hey John"}]}C
```

Figure 20: Notes and Log files

Test Case 3: Searching for keywords and notes with no matches in the database and json file should clearly display a message to the user stating so.

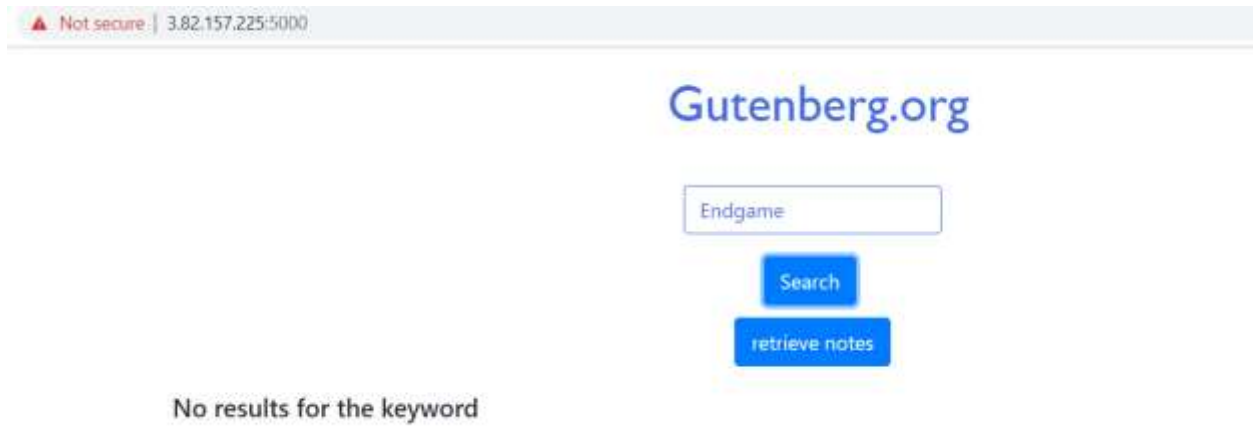


Figure 21: No results found for the given keyword

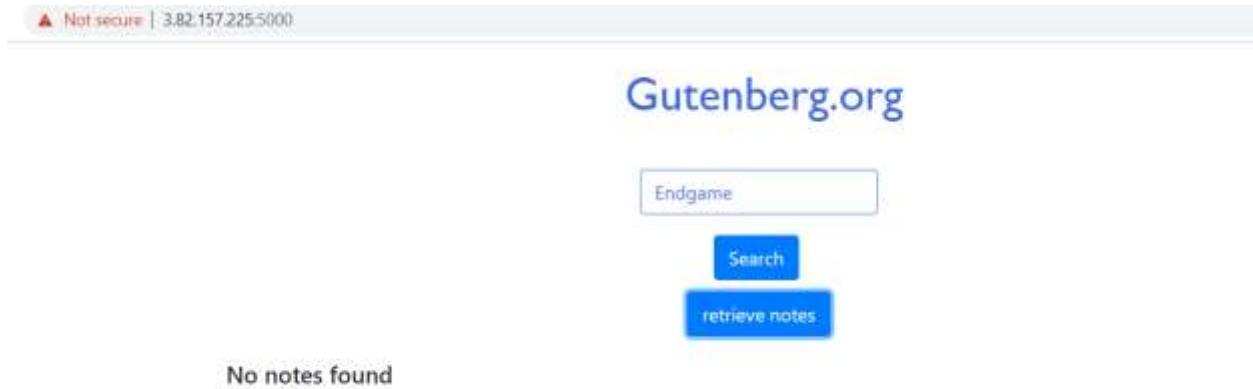


Figure 22: No notes found for the keyword

References

[1] Mithilesh's 'Setting and connecting to a remote mongodb database' [Online]

Available: <https://medium.com/founding-ithaka/setting-up-and-connecting-to-a-remote-mongodb-database-5df754a4da89> [Accessed on March 10, 2020]

[2] Robert walter's 'Getting started with python and mongodb' [Online]

Available: <https://www.mongodb.com/blog/post/getting-started-with-python-and-mongodb> [Accessed on March 10, 2020]

[3] Mongodb's documentation on 'Installing mongodb on Ubuntu' [Online]

Available: <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/> [Accessed on March 10, 2020]

[4] Docker's documentation on 'Build and run your image' [Online]

Available: <https://docs.docker.com/get-started/part2/> [Accessed on March 12, 2020]

[5] Samuel James's 'Docker build: beginners guide' [Online]

Available: <https://stackify.com/docker-build-a-beginners-guide-to-building-docker-images/> [Accessed on March 12, 2020]

[6] Node JS's documentation on 'Dockerizing a node JS web app' [Online]

Available: <https://nodejs.org/en/docs/guides/nodejs-docker-webapp/> [Accessed on March 13, 2020]

[7] Gutenberg:Offline Catalogs [Online]

Available: http://www.gutenberg.org/wiki/Gutenberg:Offline_Catalogs [Accessed on March 10, 2020]