DALHOUSIE UNIVERSITY

Faculty of Computer Science

CSCI 5409

Advanced Topics in Cloud Computing:

Report - Assignment 3

March 25th, 2020

Submitted by:
Satya Kumar Itekela
Dalhousie ID: B00839907
Brightspace: st798799

Table of Contents

Figures

1.Report

The CSCI 5409 model project's architecture demonstrates that a developer extracts the data from the Gotenberg dataset that contains various kinds of books in the English language that are stored in MongoDB database hosted in an EC2 instance and to communicate with the services such as catalogue, search and notes to retrieve the title, author details of the respective books that are built in docker containers hosted in EC2 instance.

The developer uses a python program to extract the data from the Gotenberg dataset from all the files from 1996 to 2020 creating different collections such as books that hold the details of the book titles and authors of multiple English books and logs collection that holds the processing time of each file. This data is stored in a MongoDB database that is schema-less and the NoSQL database stores the data in JSON format. This MongoDB database is created and hosted in EC2 instance. MongoDB database is very easy to store the data and the data retrieval is very fast compared to all other databases.

Users can use the application that is built on angular as the front end interface which contains the different functionalities that are divided into multiple modules such as search module, Note submission module, and Notes Retrieval module. These modules are implemented using different application program interfaces called as APIs that are built on nodeJs as backend services. When the user searches for any keyword, the webpage interacts with the search log API service and saves the keyword and time at the time of the search. Once the search is successful for the given keyword the search log calls for the Catalogue API service to retrieve the respective information and gathers the information from the MongoDB database and displays on the web page in the table format containing the title and author of the book. The user also have two options for saving the notes for the respective keyword and retrieve the notes regarding the keyword. The website uses Notes API for retrieving and saving the notes for the respective keyword. Three APIs build using NodeJs is hosted on the three containers in EC2 instance using docker-compose.

Docker is an open-source platform that enables the applications to develop, ship and operate easily [1]. It makes applications that are separated from the infrastructure so that the applications can be delivered fast and as portable that can run virtually [2]. It is built on the operating system and it acts like a single operating system using binaries for the containers. The Docker provides the applications to build, start and stop the containers easily on any type of operating system once it is built on any operating system. It also helps develops to develop, refractor applications and build containers for continuous development and integration [3]. These dockers can be easily implemented and running Docker on Amazon Web services provides highly reliable, scalability that can be achieved at a very low cost [3].

Each API is built on a single container using Dockerfile and all the dockers are built using docker-compose. All these services are built and placed in the docker hub [4]. The Docker hub is managed completely, secure and highly accessible. Docker hub hosts images of multiple applications that can be deployed easily and can be ported to different environments.

2.Extraction Engine

A Python program is written to extract the data such as the title and author of English books from the given Gotenberg database [5] from the year 1996 to 2020. The books list is stored in the books collection and the processing time for each file is stored in the logs collection as shown in the below diagram.

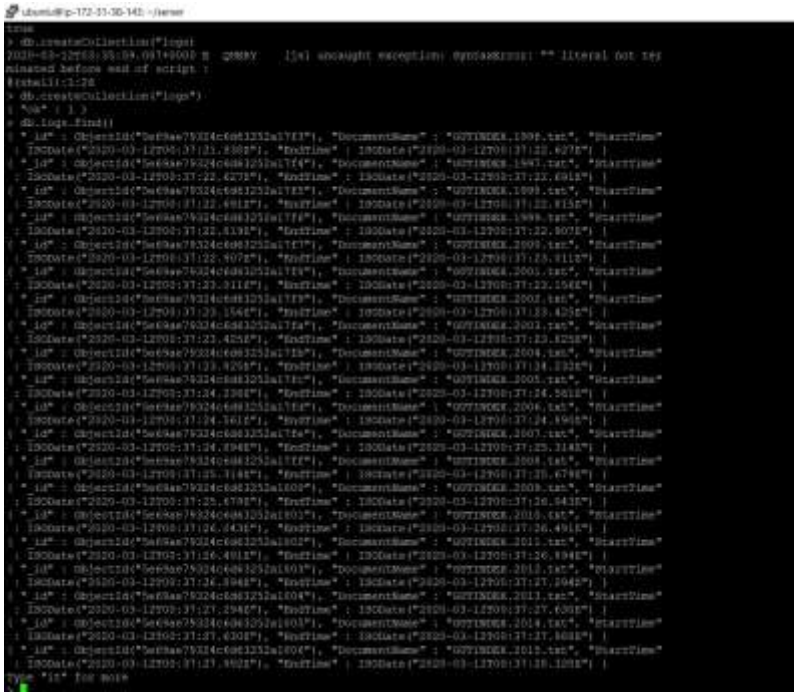

Figure 1 Logs Collection

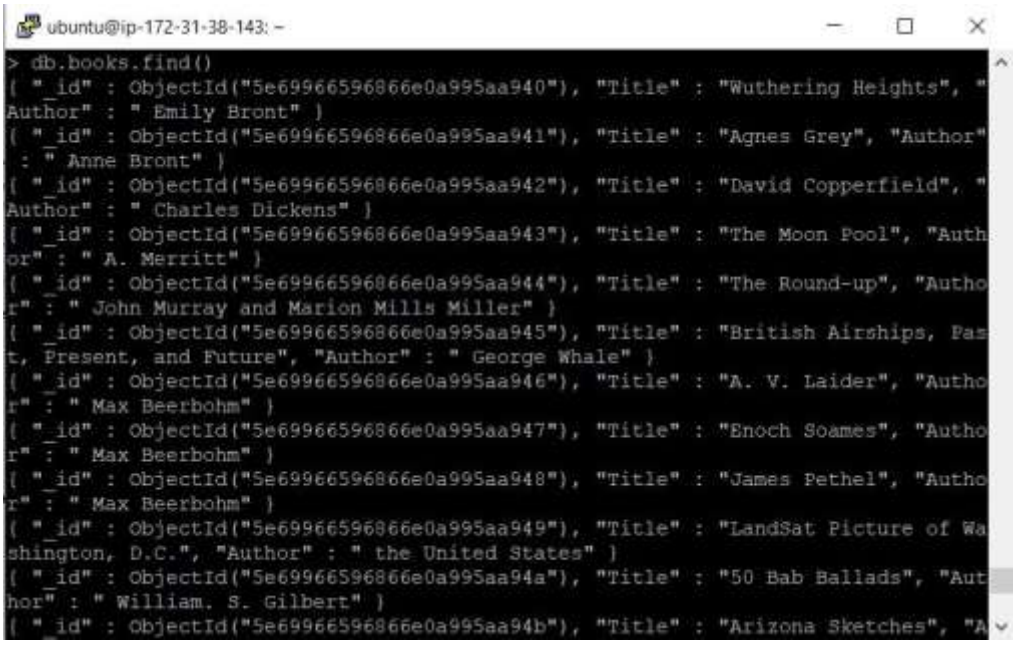

Figure 2 Books Collection

3.Webpage

An Angular application is built for the title, author search that enables users to enter a keyword to retrieve title and author for the books and write notes for the successful search and also retrieve notes in future for the specific search. When the user enters the keyword and click search, the user can able to retrieve data for the specific keyword and also can get notes for the given keyword.

Figure 3 Search Application

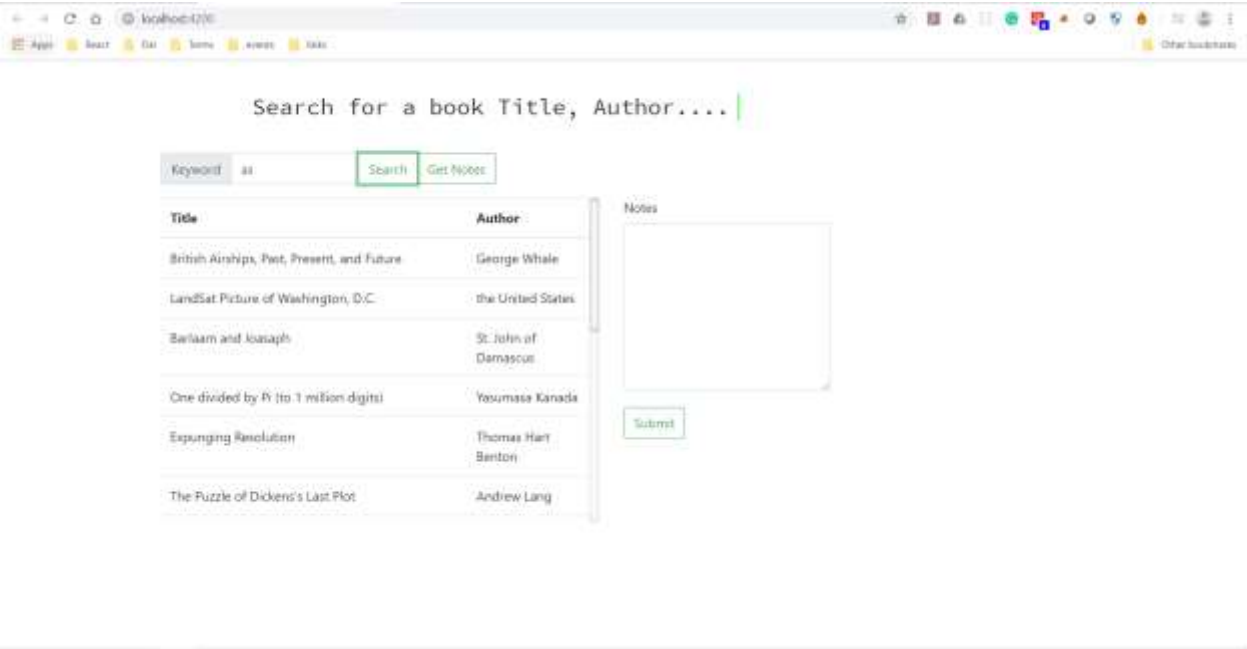

Figure 4 Books Title, Author Table

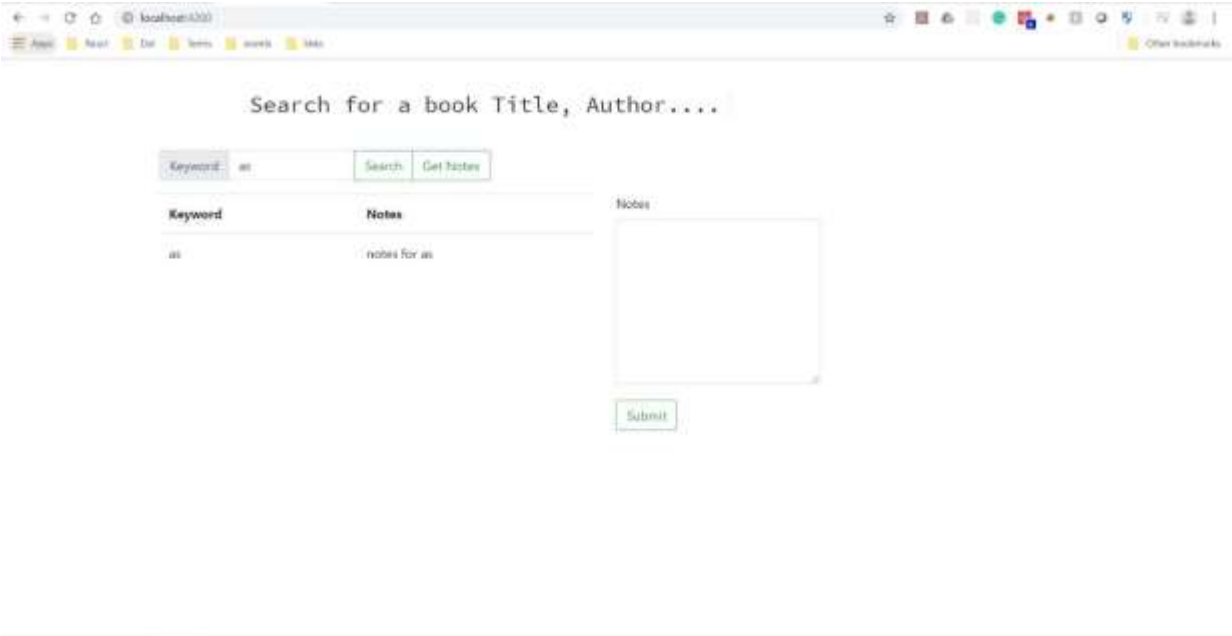

Figure 5 Notes Retrieval

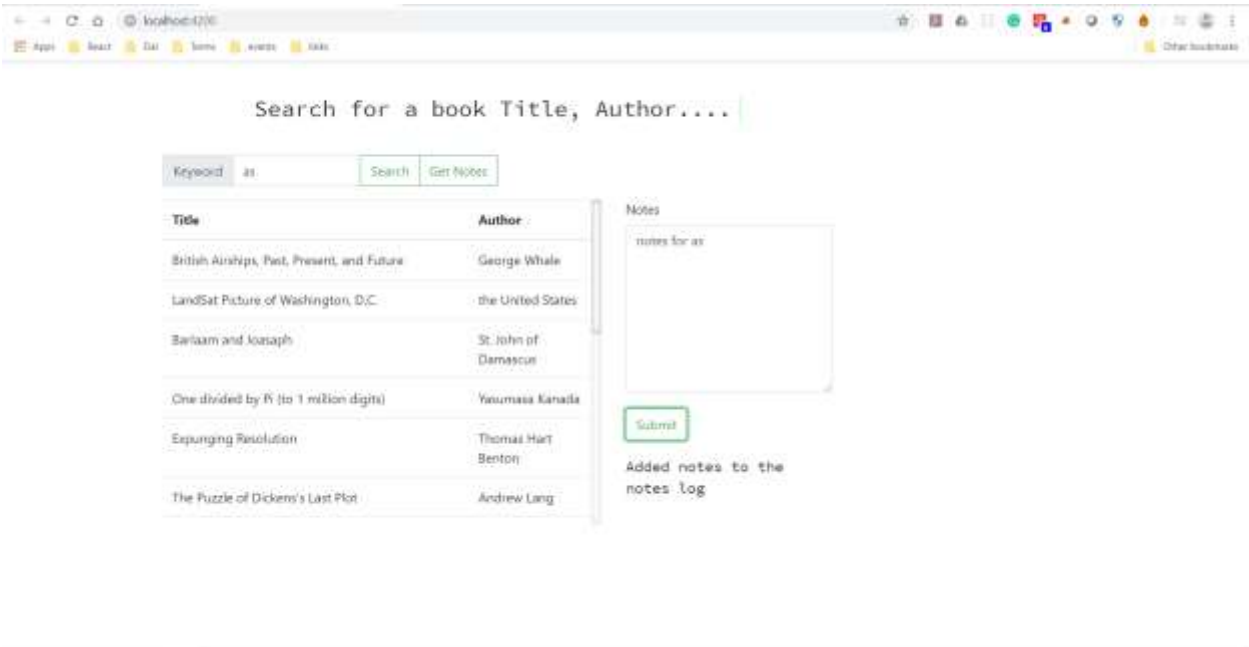

Figure 6 Notes Submission

4.Docker Container

4.1.Search log API

If a user enters the keyword and clicks the search button, the keyword and the time at the time of search are stored in the search log and also it also stores the frequency of the word searched using the post API. A docker file is written to deploy the application in the docker.

4.2.Catalogue API

If the search data is successful, the search data is displayed on the webpage using the catalogue API. When the search is successful catalogue service helps to search the keyword for each key in the MongoDB and displays the data on the webpage using the get API. A docker file is written to deploy the application in the docker.

4.3.Notes API

When the search is successful, the notes are saved for the respective keyword in the JSON format using the post API. The user can also have the ability to retrieve the data using the get API. A docker file is written to deploy the application in the docker.

After creating all the APIs the docker-compose file is written to build and deploy the application in EC2 instance.

A docker is installed in the EC2 instance for building the containers.

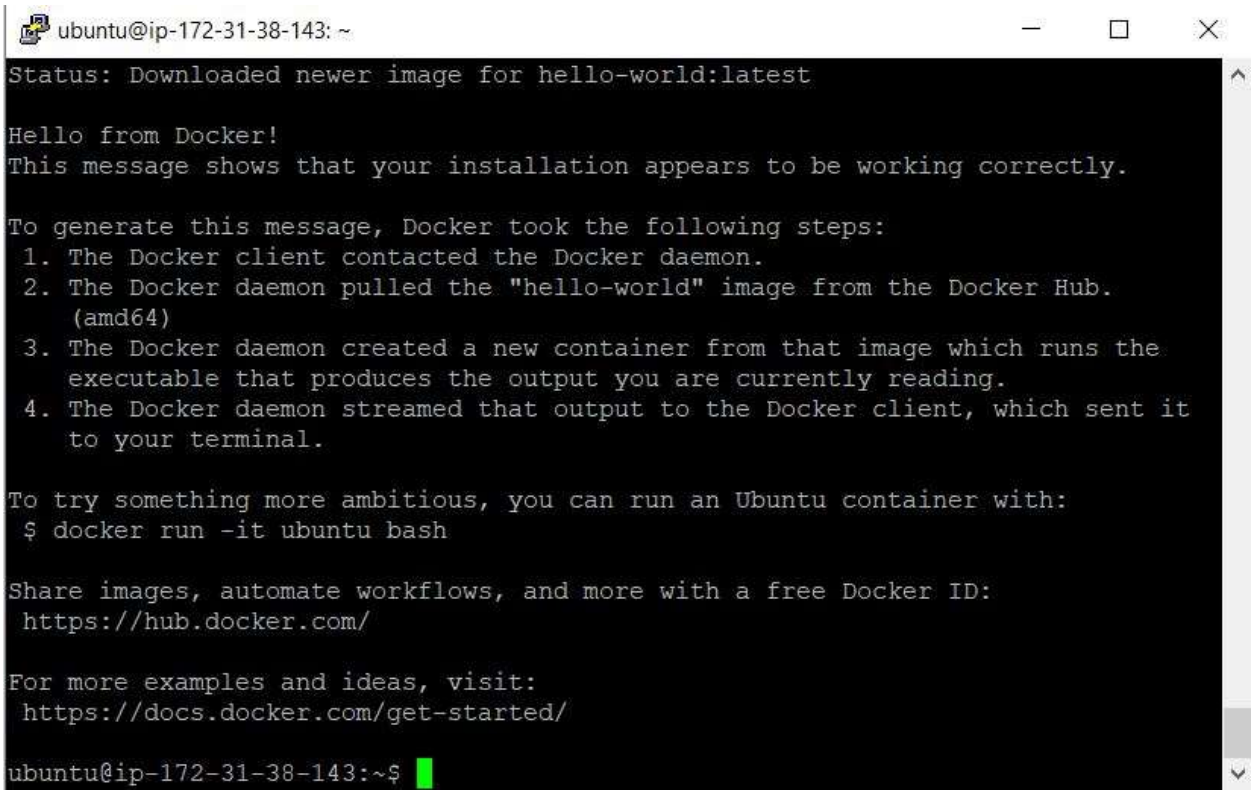

Figure 7 Docker Installation

```
ubuntu@ip-172-31-38-143: ~
 Git commit:        afacb8b7f0
 Built:             Wed Mar 11 01:25:46 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       afacb8b7f0
  Built:            Wed Mar 11 01:24:19 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
ubuntu@ip-172-31-38-143:~$
```

Figure 8 Docker Version

```
ubuntu@ip-172-31-38-143: ~/services
 ---> Running in 011b398a50d0
Removing intermediate container 011b398a50d0
 ---> 6d4e8876a126
Successfully built 6d4e8876a126
Successfully tagged services_node_1:latest
Creating services_node_3_1 ...
Creating services_node_2_1 ...
Creating services_node_1_1 ...
Creating services_node_3_1
Creating services_node_2_1
Creating services_node_3_1 ... done
Attaching to services_node_2_1, services_node_1_1, services_node_3_1
node_2_1  |
node_2_1  | > catalogue@1.0.0 start /usr/src/app
node_2_1  | > node index.js
node_2_1  |
node_1_1  |
node_1_1  | > search@1.0.0 start /usr/src/app
node_1_1  | > node index.js
node_1_1  |
node_3_1  |
node_3_1  | > notes@1.0.0 start /usr/src/app
node_3_1  | > node index.js
node_3_1  |
```

Figure 9 Docker Services

5.Deployment

All the services are deployed in different ports and deployed in different containers. When the user enters the keyword and hits the search button, it calls the search API that is running on the port number 3000 to save the search log. After the successful search, it calls the catalogue API that is running on the port number 5000 and renders the data from MongoDB. If the search is successful, the user can enter the notes and submit and retrieve the notes that is running on the port 8000. The notes log that is stored in the database is retrieved using the notes API. All the services are built on different containers and hosted on the EC2 instance using docker-compose.

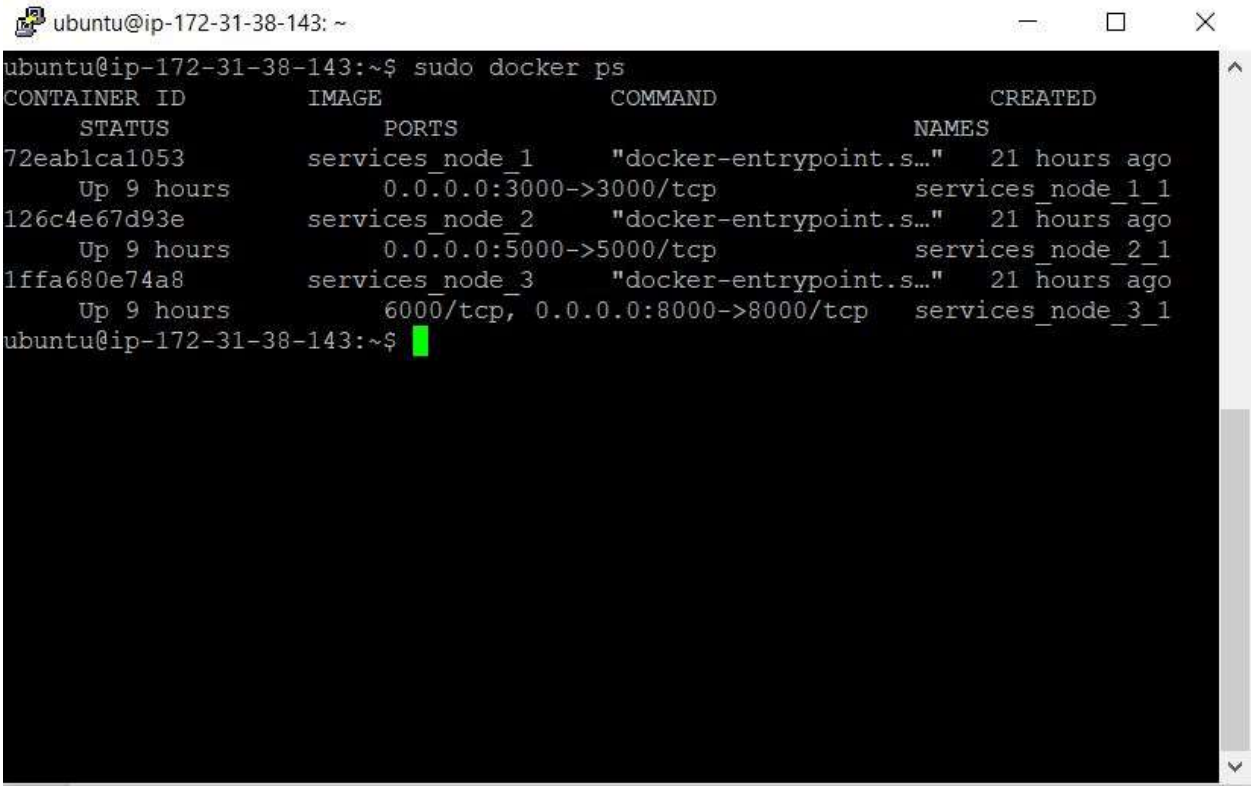

Figure 10 Docker Containers

6.Testing

1. Test Case – 1: Get the details for the entered keyword [6]

   a. Launch http://localhost:4200/ on the browser.
   b. Search for the keyword into the Search field on the top of the screen.
   c. If the keyword is successful, a title, author table is displayed. Else, the no results message is displayed.
   d. If the user does not enter the keyword and click the search button, the error message is displayed.

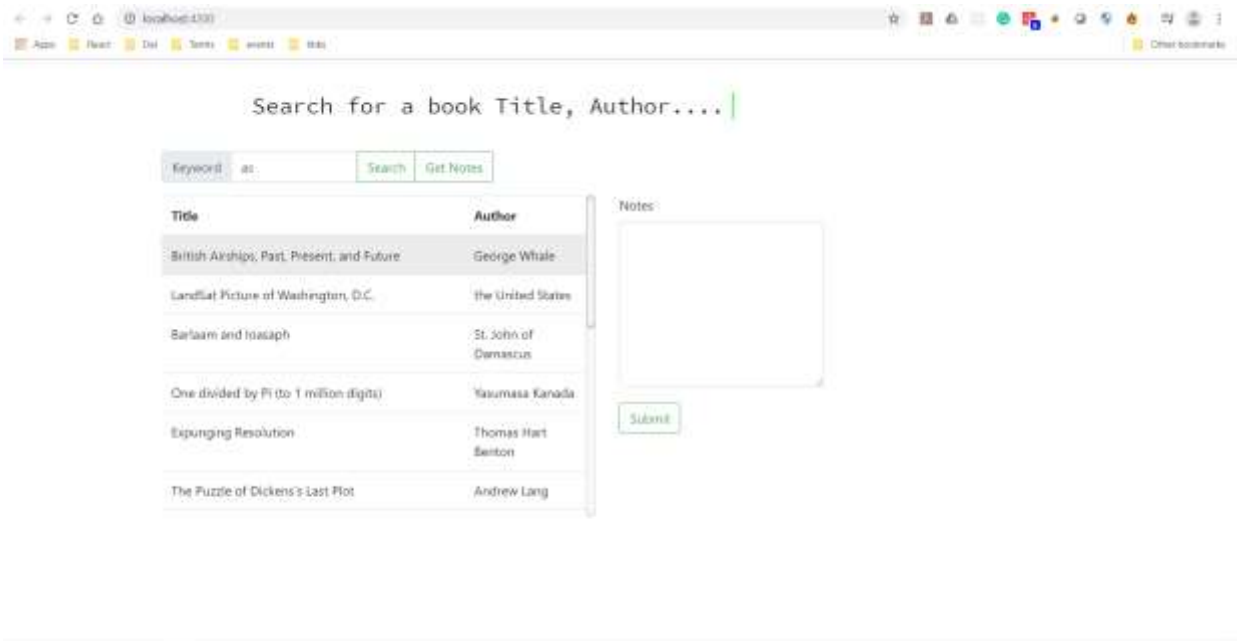

Figure 11 Success Page

Figure 12 Search - No entries

Figure 13 Error Message

2. Test Case – 2: Enter the notes and submit the notes

   a. Launch http://localhost:4200/ on the browser.
   b. Search for the keyword into the Search field on the top of the screen.
   c. If the keyword is successful, the user can able to enter the notes and submit the notes by clicking the submit button.
   d. If the notes are empty and click the submit button, the notes cannot be submitted and get an error message.

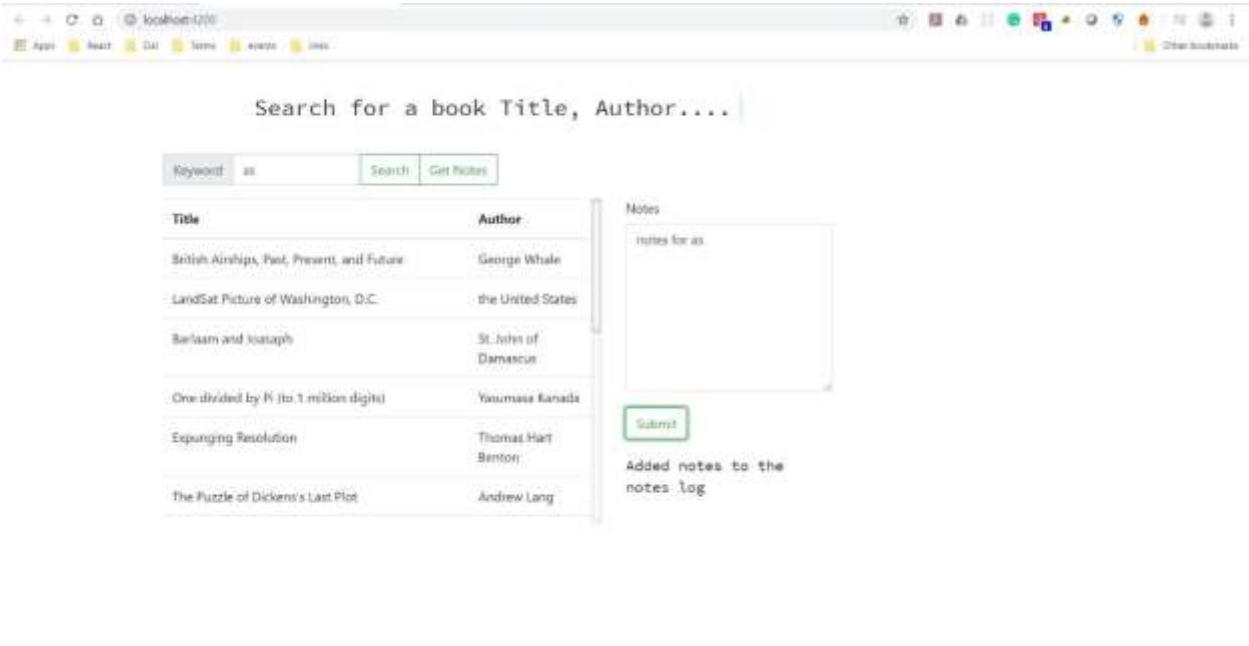

Figure 14 Notes Submission

Figure 15 No entries

Figure 16 Notes - Not Submitted

3. Test Case – 3: Enter the keyword and retrieve the notes

a. Launch http://localhost:4200/ on the browser.
b. Search for the keyword into the Search field on the top of the screen.
c. Click the button Get Notes if the keyword is successful the table is displayed with notes for the respective keyword, else error message is displayed.

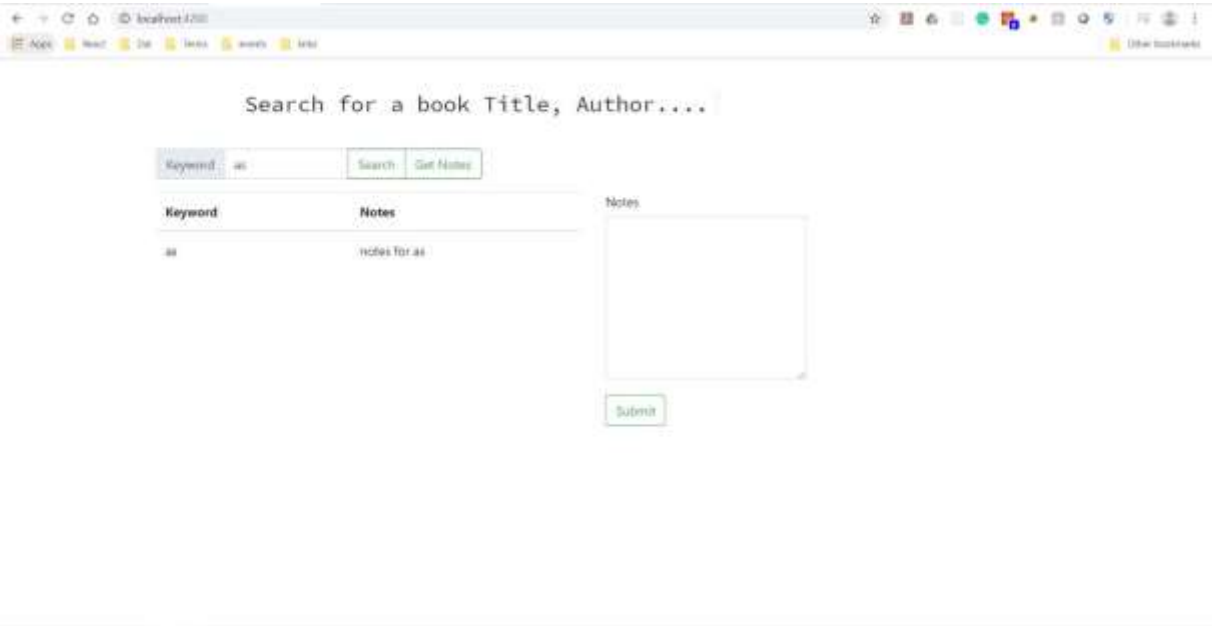

Figure 17 Notes Submission

Figure 18 Notes - No entries

4. Test Case – 4: Enter the keyword and store the search log

a. Launch http://localhost:4200/ on the browser.
b. Search for the keyword into the Search field on the top of the screen.
c. Click the button Search the keyword and the time at the time of the search and the frequency to be stored in the search log.

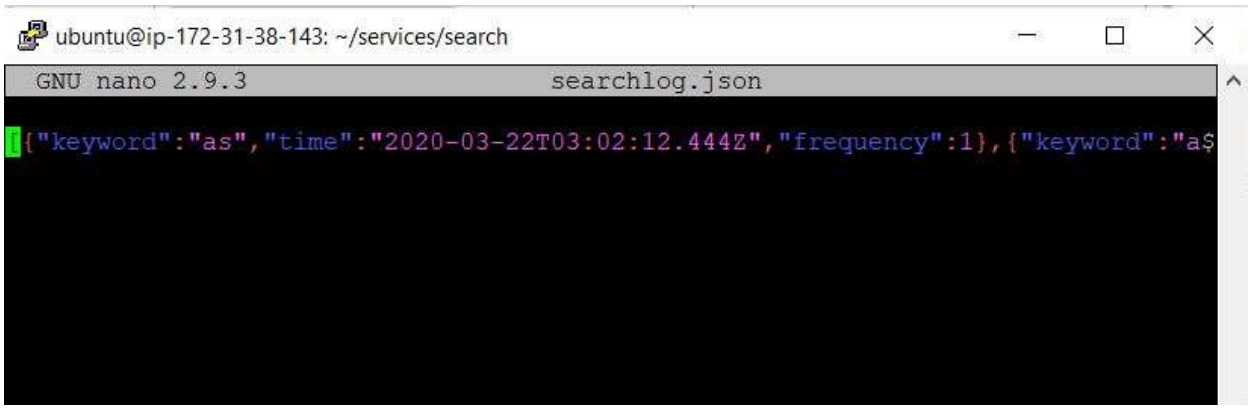

Figure 19 Search log

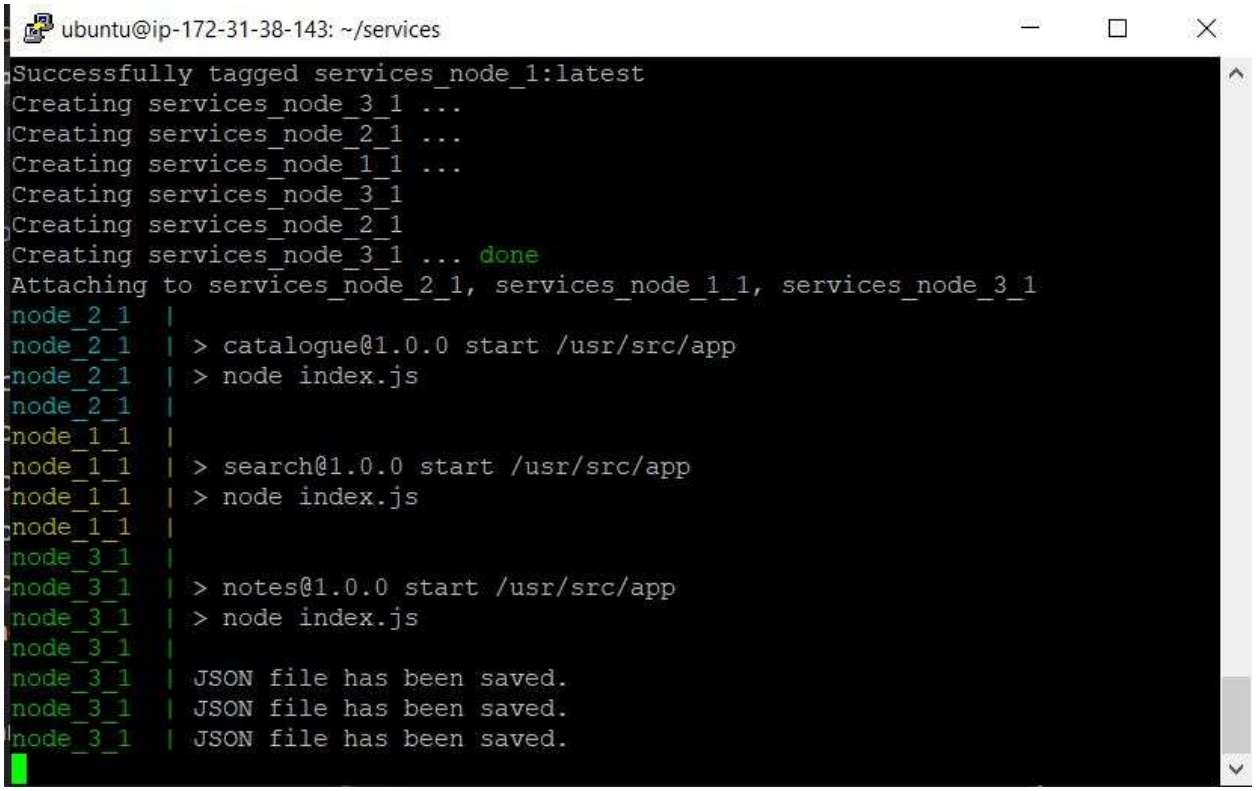

Figure 20 JSON file

7.References

[1]"Get Docker", *Docker Documentation*, 2020. [Online]. Available: https://docs.docker.com/get-docker/. [Accessed: 22- Mar- 2020].

[2] S. Vaughan-Nichols, "What is Docker and why is it so darn popular? | ZDNet", *ZDNet*, 2020. [Online]. Available: https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/. [Accessed: 22- Mar- 2020].

[3]"Top 10 Benefits of Docker - DZone DevOps", *dzone.com*, 2020. [Online]. Available: https://dzone.com/articles/top-10-benefits-of-using-docker. [Accessed: 22- Mar- 2020].

[4]"Docker Hub", *Hub.docker.com*, 2020. [Online]. Available: https://hub.docker.com/search/?type=image. [Accessed: 22- Mar- 2020].

[5]"Project Gutenberg", *Project Gutenberg*, 2020. [Online]. Available: http://www.gutenberg.org/wiki/Gutenberg:Offline_Catalogs. [Accessed: 22- Mar- 2020].

[6]"How to Write Test Cases: The Ultimate Guide with Examples", *Softwaretestinghelp.com*, 2020. [Online]. Available: https://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/. [Accessed: 22- Mar- 2020].