

IT4090
Cloud Computing
4th Year, 2nd Semester



Assignment 1 – Report
Lakshan wijewardana W.M.W
IT18132588

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the
Bachelor of Science Special Honors Degree in Information Technology

2021.11.07

Contents

1. Architecture	3
2. Application Logic.....	4
3. Screenshots of Docker environment	5

1. Architecture

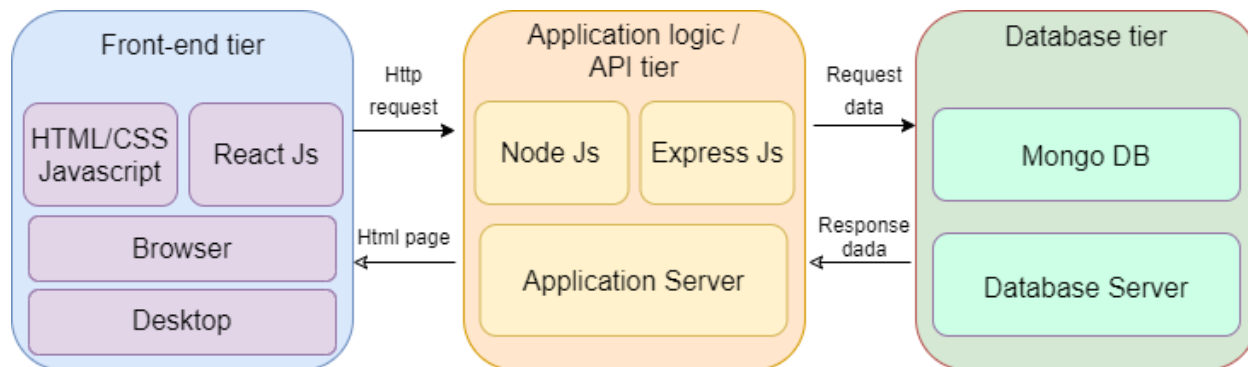


Figure 1:3 Tier Architecture

This application's architecture is built on a standard MVC model. The MVC architectural pattern is a popular paradigm for implementing this model. The logic, data, and visualization are split into three sorts of objects in the MVC paradigm, each performing its own duty. The View is in charge of the visual aspect, as well as user interaction. The Controller reacts to system and user events by instructing the Model and View to modify. The Model manipulates data, responding to information requests, and modifying its state in accordance with the Controller's instructions. Our front-end tier (View) will be developed in Javascript, HTML, and CSS, with the framework ReactJS. This is the level of architecture with which the user will interface in order to access the functionality of our application.

The Application Logic Tier (Controller)/API will be developed in NodeJS and ExpressJS, and it will represent the Application Server that will act as a communication bridge between the Front-end Tier and the Database Tier. This tier will provide HTML pages to the user's device, accept HTTP requests from the user, and answer properly.

MongoDB (Database Tier (Model)) is where we will save all of the critical data that our application need to work.

2. Application Logic

This is Student details management application. User can add students' name, address and course name and user can delete, update, view students' details. Student details are stored in MongoDB database.

3. Screenshots of Docker environment

```
6 | ports:
7 |     "3000:3000"
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
50 |
51 |
52 |
53 |
54 |
55 |
56 |
57 |
58 |
59 |
60 |
61 |
62 |
63 |
64 |
65 |
66 |
67 |
68 |
69 |
70 |
71 |
72 |
73 |
74 |
75 |
76 |
77 |
78 |
79 |
80 |
81 |
82 |
83 |
84 |
85 |
86 |
87 |
88 |
89 |
90 |
91 |
92 |
93 |
94 |
95 |
96 |
97 |
98 |
99 |
100 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\lakshan\Desktop\portfolio\docker-app> docker-compose ps

Name	Command	State	Ports
docker-app_backend-api_1	docker-entrypoint.sh node ...	Up	0.0.0.0:8001->8001/tcp
docker-app_frontend-client_1	docker-entrypoint.sh npm r ...	Up	0.0.0.0:3000->3000/tcp
docker-app_mongo_1	docker-entrypoint.sh mongod	Up	0.0.0.0:27017->27017/tcp

PS C:\Users\lakshan\Desktop\portfolio\docker-app> docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker-assignment-server	latest	1dc544e6b8e3	6 hours ago	144MB
docker-assignment-client	latest	e4237ccf989c	6 hours ago	391MB
mongo	latest	fe7d78e9381a	3 weeks ago	699MB

PS C:\Users\lakshan\Desktop\portfolio\docker-app> docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f1d78bbca0d0	docker-assignment-client	"docker-entrypoint.s..."	2 hours ago	Up About a minute	0.0.0.0:3000->3000/tcp	docker-app_frontend-client_1
c22aedefb1c9	mongo:latest	"docker-entrypoint.s..."	2 hours ago	Up About a minute	0.0.0.0:27017->27017/tcp	docker-app_mongo_1
30c0d6af2b62	docker-assignment-server	"docker-entrypoint.s..."	2 hours ago	Up About a minute	0.0.0.0:8001->8001/tcp	docker-app_backend-api_1

PS C:\Users\lakshan\Desktop\portfolio\docker-app> []

Figure 2: Docker images

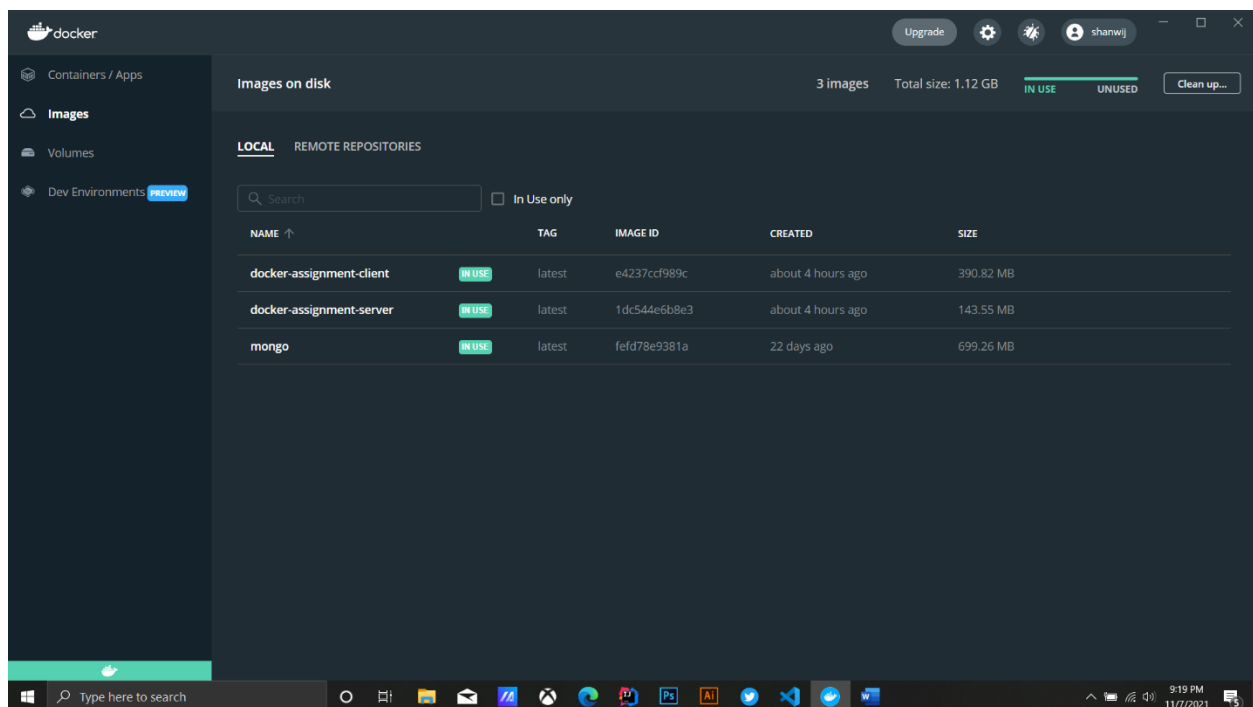


Figure 3: Docker environment

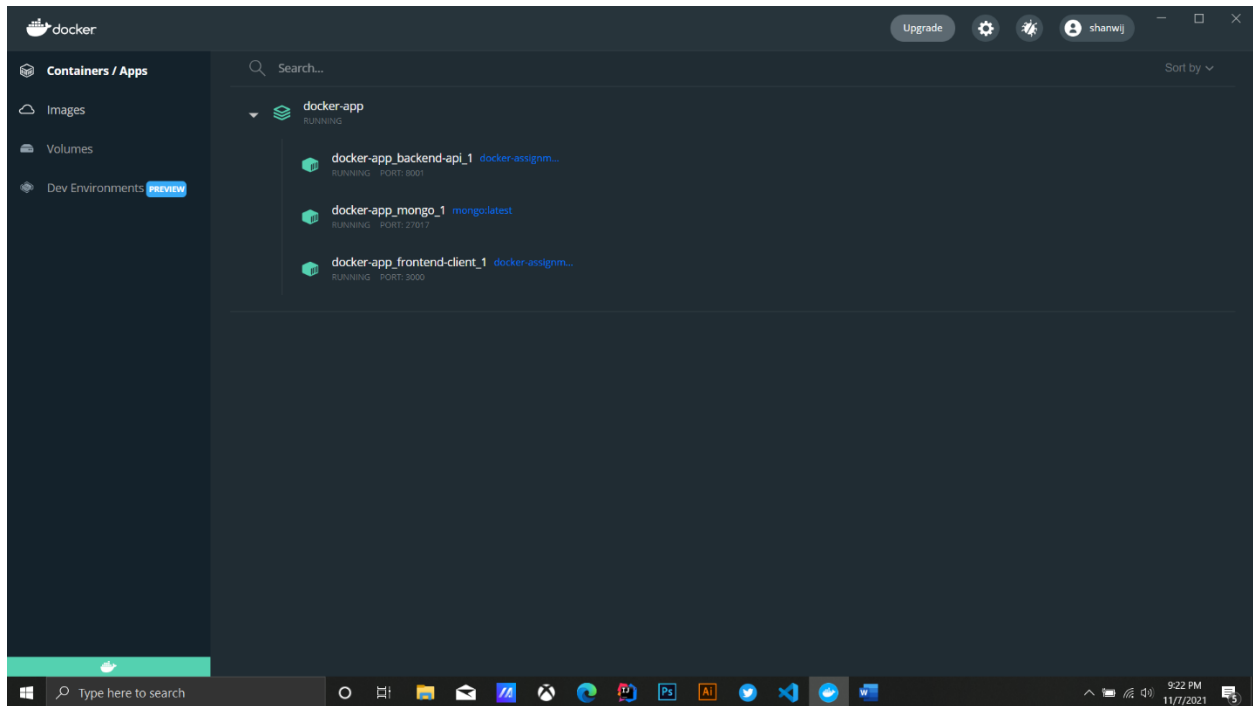


Figure 4: Docker Container

```
server > Dockerfile > ...  
1 FROM node:14-alpine  
2 WORKDIR /usr/src/app  
3  
4 COPY ./package.json ./  
5  
6 RUN npm install  
7  
8 COPY . .  
9  
10 EXPOSE 8001  
11  
12 CMD [ "node", "index.js" ]
```

Figure 5: Dockerfile - server

```
...  docker-compose.yml X
docker-compose.yml
1  version: "3"
2  services:
3    frontend-client:
4      image: docker-assignment-client
5      stdin_open: true
6      ports:
7        - "3000:3000"
8      networks:
9        - mern-app
10   backend-api:
11     image: docker-assignment-server
12     ports:
13       - "8001:8001"
14     networks:
15       - mern-app
16     depends_on:
17       - mongo
18   mongo:
19     image: mongo:latest
20     ports:
21       - "27017:27017"
22     networks:
23       - mern-app
24     volumes:
25       - ./mongo/data:/data/db
26   networks:
27     mern-app:
28       driver: bridge
29   volumes:
30     mongo-data:
31       driver: local
```

Figure 6: Docker-compose file

```
... Dockerfile X
client > Dockerfile > ...
1 FROM node:14-alpine
2
3 WORKDIR /usr/src/app
4 ENV PATH /usr/src/app/node_modules/.bin:$PATH
5 COPY ./package.json ./
6
7 RUN npm install
8
9 COPY . .
10
11 EXPOSE 3000
12
13 CMD [ "npm", "run", "start" ]
```

Figure 7: Dockerfile - Client