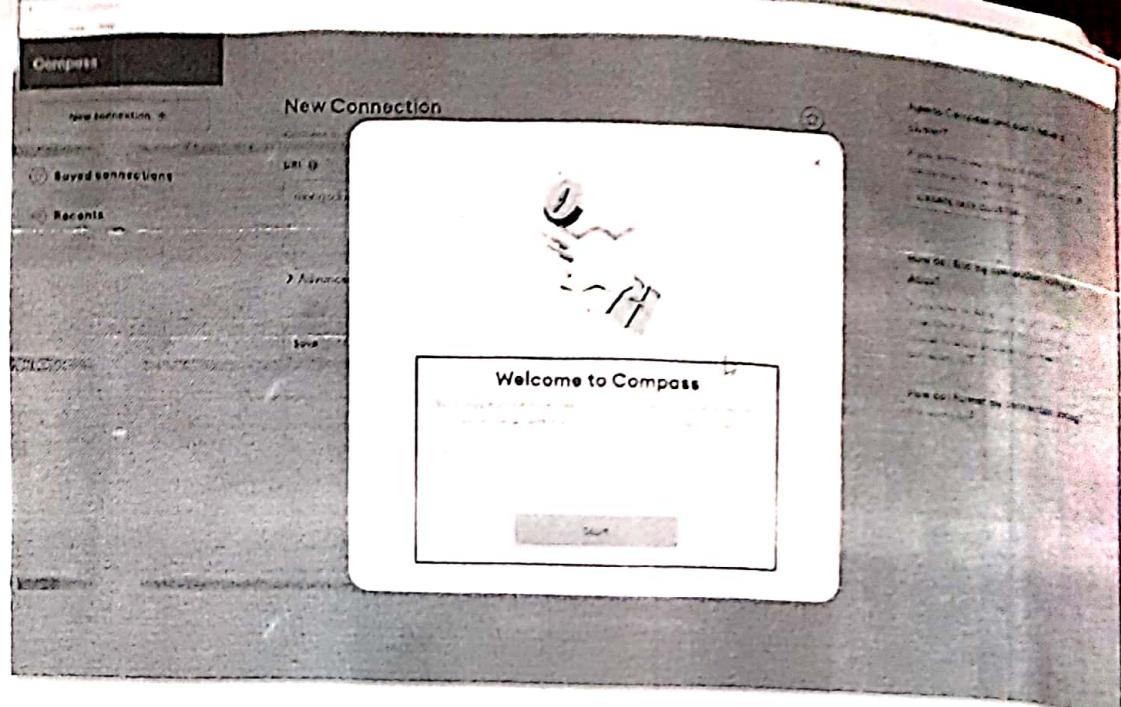


INDEX

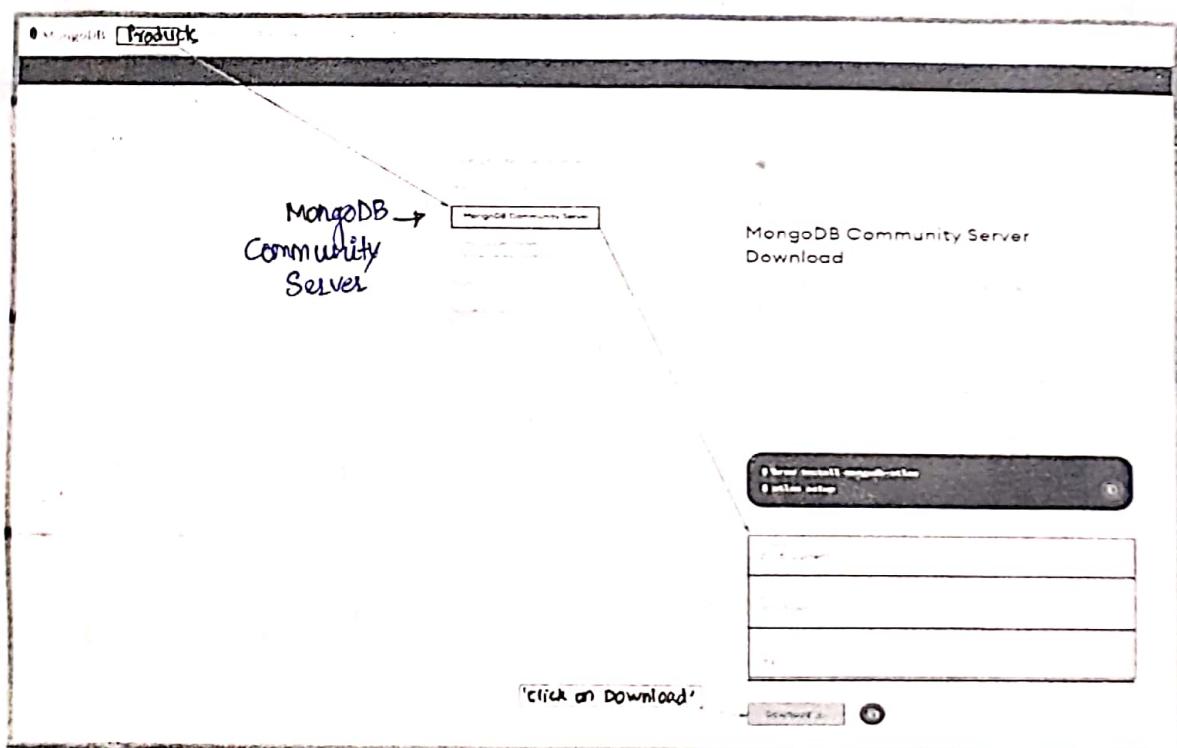
Step.01:

- Go to MongoCompass
- Click on Next & select 'compass'



Step.02:

- Under 'Products' Select 'MongoDB Community Server' → Download for the 'Windows' & Download it.



- ### Step.03:
- To check the version of (mongo-db, use command prompt. ↴shell)

- Put 'mongo' cmd, & all the details of MongoDB Server.

```
C:\Users\Vimal.Dagax mongo
MongoDB shell version: v4.4.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("fdb58706-ab40-4ae6-8bc8-f2da1019d453") }
MongoDB server version: 4.4.5
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
    https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
    https://community.mongodb.com
...
The server generated these startup warnings when booting:
2021-04-28T11:32:41.533+05:30: Access control is not enabled for the database. Read and
data and configuration is unrestricted
```

SKILL ACTIVITY NO: 1

Date: 02/02/2023

(To be filled by the Student)

Title:

Creating a Database by setting MongoDB Atlas Cloud and installing MongoDB Compass.

1. What is the purpose of this activity? (Explain in 3 – 4 lines)

The purpose of this activity are as follows:- (To understand

- (i) What are Nosql databases? Why we need them?
- (ii) Implementation of Nosql databases, open-source Nosql db's.
- (iii) (What, Why, How) of Mongodb Nosql database.
- (iv) Various features of Atlas cloud & Mongodb as a service.
- (v) Installation of MongoDB Compass on our local system.
- (vi) Query Engine & Document-type data storage models.
 - why they are better than SQL?
 - what's the Real-world usecase of them?

Algo's working behind the documentdb's CRUD operation.

2. Step performed in this activity (Explain in 5 – 6 lines)

Steps performed in this activity are as follows:-

- Step(I) Installing MongoDB server. Open your browser & browser "MongoDB download" → URL: <https://www.mongodb.com/download/community>
- Step(II) Click on "Mongodb Community Server Download".
- Step(III) Under Products, go to MongoDB Comm. & select Server
- Step(IV) Click on "Download" & simultaneously download Mongodb shell & MongoDB compass.
- Step(V) Downloading MongoDB compass: go to following URL: <https://www.mongodb.com/products/compass>
- Step(VI) After downloading both server & compass:-
→ Double-click on ".msi" Installer → Next → choose "Custom"
→ Under Custom Installation → Select "MongoDB as a service"
→ Run it as Network Service User.
→ Then, click on "Next"
- Step(VII) Also, Edit the Environment variables → So, Paste the binaries path → i.e. C:\Program files\mongodb\server\4.4\bin

3. What resources / materials / equipments / tools did you use for this activity ?

1. Online Resources :

1. ↳ MongoDB Community Server V4.4.5
2. (<https://www.mongodb.com/tarball/community>)
3. ↳ MongoDB official documentation (<https://www.mongodb.com/docs/atlas>)
4. ↳ MongoDB Compass documentation (<https://www.mongodb.com/download-center/compass>)

5. Equipments: PC; Intel(R) Core(TM) i5-6400@2.70GHz

LENOVO_MT_30_AS_BU

LENOVO_FM_ThinkStation_P320

• Microsoft Windows 10 Pro 10.0.19044 Build

: Greenshot Screenshot Tool

7. Materials: ↳ Lecture PPT on Installation of MongoDB Community server.

4. What skills did you acquire ?

1. WSH abt NoSQL databases
2. WSH abt Document-type databases
3. How, queries evalua" plan, algo's created by developers of MongoDB?
- 4.
5. Real-world usecases of MongoDB NoSQL database
6. Installation of "MongoDB Compass" & "MongoDB server Community server"
- 7.
8. Database creation on MongoDB Server & connecting with MongoDB Compass.

5. Time taken to complete the activity ? 02:00 (hours)

Wajid ✓

Signature of Student

(To be filled by Faculty)

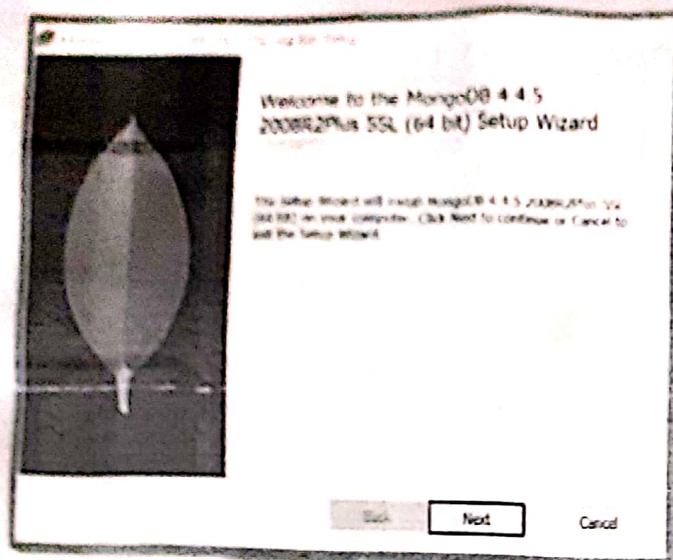
S. No.	Skills / Competencies	(Achieved / Not Achieved) (YES / NO)
01.	Installation of MongoDB Community Server <u>(VG.O.5)</u>	
02.	Installation of MongoDB Compass in our local PC. <u>(VG.O)</u>	
03.	(Why, what, How) regarding to NoSQL databases	
04.	Provisioning the MongoDB Compass in the cloud using <u>ATLAS</u> ,	
05.	Realworld usecases of MongoDB database as DOCUMENT BASED Database.	

Remarks

Total marks _____ out of 10.

Sign of Faculty

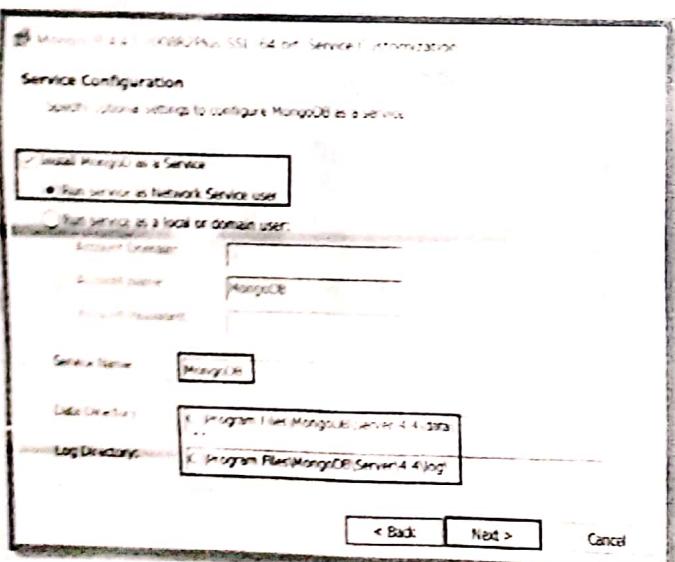
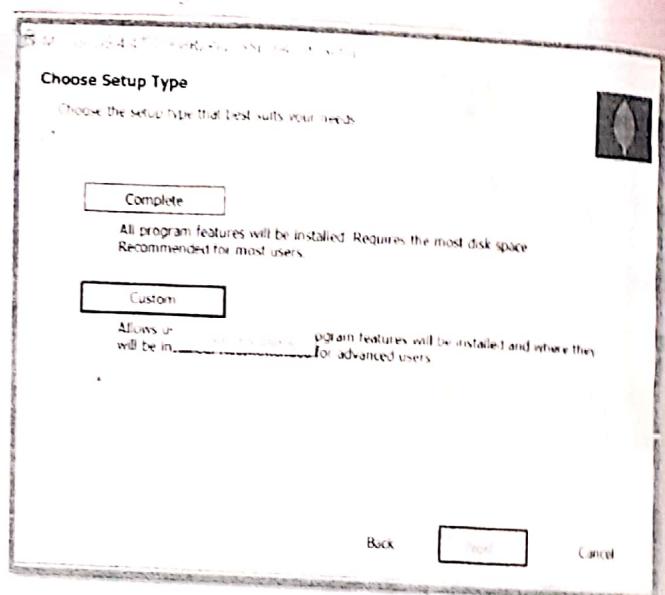
Date:



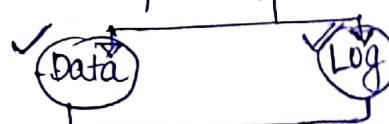
Step-04: After downloading setup, double-click on MongoDB Installer (V4.4.5).
 ↳ Click on 'Next' Button.

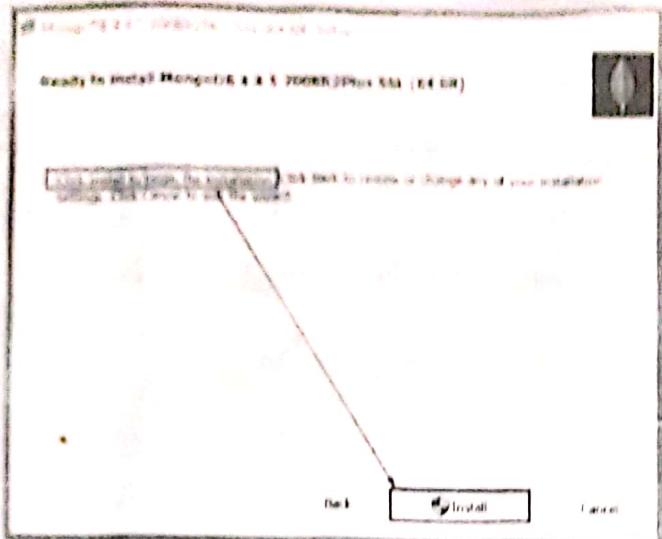
Step-05: Now select 'custom' setup type,

↳ Here, we can easily customize MongoDB community server as a Thread (process) and using cloud paradigm of 'MongoDB as a service'.



Step-06: Under 'Service Configuration',
 ↳ Select to
 a) Install MongoDB as a Service (mongod)
 ↳ Run service as Network Service User
 b) Select 'Service Name' == MongoDB
 c) Provide desired path of directory:-

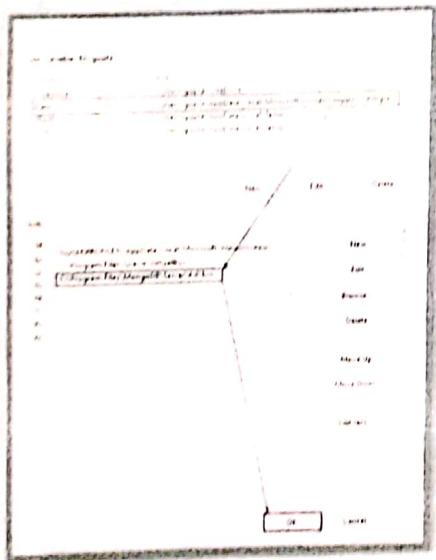
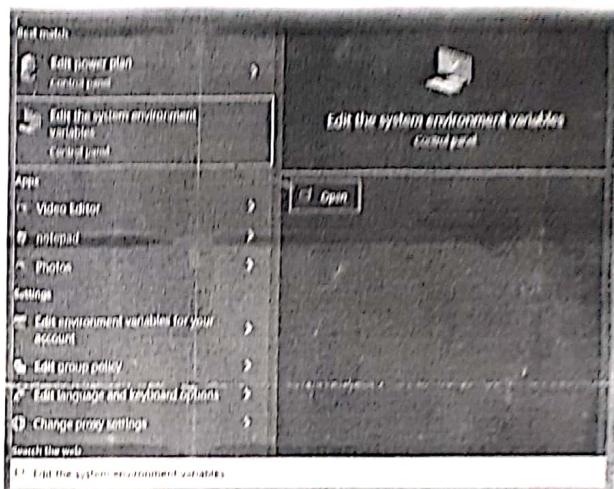




Step-07:, Click on the MongoDB 4.4.3 2008R2}
 ↳ (Twice) → And it will
 start installing the server.
 (by entering User, Account Control)
 (prompting) → Give the permission)

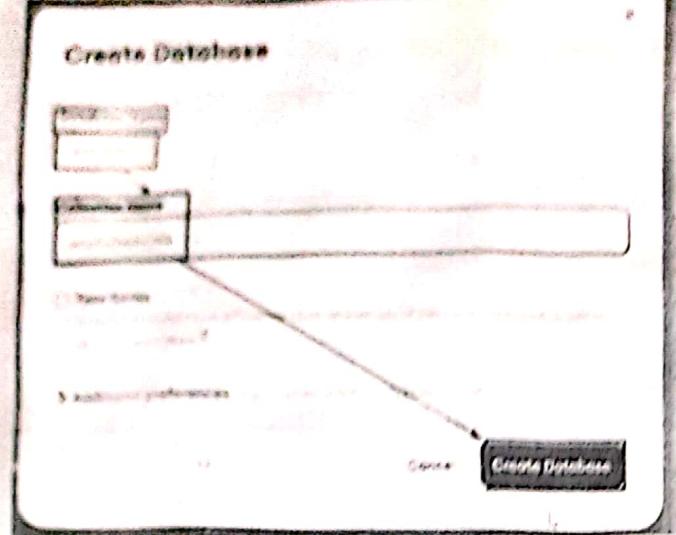
Step-08: Now, to use mongo shell command from normal user cmd prompt :-

- Search for Edit environmental variables, & open it.
- After opening it, Search for "System Environmental Variables".



Step-09:

- Under "System Environmental Variables",
- Click on "New" button.
 - Enter the "binaries" → "bin" path of MongoDB Server (Community)
i.e.
 - Then click on "OK" button.
Due to which Environmental variable.



- Step.01: CRUD Operation: CREATE
- In this screenshot, we are creating a database 'employee' \Rightarrow DATABASE_NAME
 - Also, providing COLLECTION_NAME = 'employeedetails'
 - Finally, click on 'Create Database'

Step.02: In this screenshot, the employee.employeedetails database & collection Both are (Created) successfully.
And we will insert the data now.

```

db=employee
use employee
db.employeedetails.insertOne({firstname:'Aayu', lastname:'Gupta', batch:'B2'})
{
  acknowledged:true,
  insertedId:5e097597c929a41c026a7f0
}
employee>
  
```

Step.03: In this screenshot Using Mongoshell,

- Select the db command: use employee
- For INSERTING
 - first data or Row in our database.

Use cmd:
db.employeedetails.
insertOne({ \Rightarrow
 firstname:'A'
 lastname:'G'
 batch:'B')

SKILL ACTIVITY NO: 2
(To be filled by the Student)

Date: 18/02/2023

Title:

Implementing CRUD operation : Illustration of CREATE & READ command of MongoDB.

1. What is the purpose of this activity ? (Explain in 3 – 4 lines)

The purpose of this activity are as follows:-

- (i) To understand what are CRUD Operations in a NoSQL database?
- (ii) To understand how these CRUD operations performed are different from the operations performed in SQL databases?
- (iii) To understand & implement the CREATE & READ Operations in a NoSQL database i.e. MongoDB Community Server (V4.4.5.)
- (iv) Real-World application → via creating a Database, collection & Inserting data fields (= i.e. rows/Tuples) in DocumentBasedDB.

2. Step performed in this activity (Explain in 5 – 6 lines)

Steps performed in this activity are as follows:-

- Step(i): Open MongoDB Compass. i.e. { if installed locally, → then in PC use MongoDB compass Installer }
- Step(ii): Click on the button → "Create Database"
- Step(iii): After the dialogue popup → Enter following details such as {
→ Database Name : employee
→ Collection Name : employeeDetails }
- Step(iv): Then, as per our database schema → we create our DB → {
employee
↳ employeeDetails }
- Step(v): After, successfully completing Step 4 → we will CREATE the document (table)
and insert our first record. using Mongo Shell { → already installed }
- Step(vi): Then, we execute the CRUD i.e. CREATE and READ

3. What resources / materials / equipments / tools did you use for this activity ?

1. Online Resources
 - ↳ MongoDB Community Server v4.4.5 (<https://www.mongodb.com/tarball/community>)
 - 2. ↳ MongoDB Official documentation (<https://www.mongodb.com/docs/atlas/>)
 - 3. ↳ MongoDB Compass download.
 - ↳ (<https://www.mongodb.com/download-center/compass>)
5. Equipments: PC: Intel (R) Core(TM) i5-6400@2.70GHz
LENOVO_MT_30_AS_BU
LENOVO_FM_ThinkStation_P310
Microsoft Windows 10 Pro 10.0.19044 Build
= Greenshot Screenshot
7. Materials:
 - ↳ lecture PPT for Installation of MongoDB
- 8.

4. What skills did you acquire ?

1. W5HH obt CREATE & READ operations
2. Query Evaluation Engine of MongoDB Database
3. Implementation of MongoDB Compass on Local System.
4. Implementation of MongoDB shell
5. Comparison between CREATE and READ of [NoSQL database & SQL database]
- 6.
- 7.
- 8.

5. Time taken to complete the activity ? 02:00 (hours)



Signature of Student

(To be filled by Faculty)

S. No.	Skills / Competencies	(Achieved / Not Achieved) (YES / NO)
O1.	Performing CRUD operations (i.e. CREATE and READ) operations on the MongoDB community server v6.0.	
O2.	Implementing and creating MongoDB Compass on local PC via inserting the records/docs in the DB	
O3.	Also, Implementing MongoDB shell & executing the commands.	

Remarks

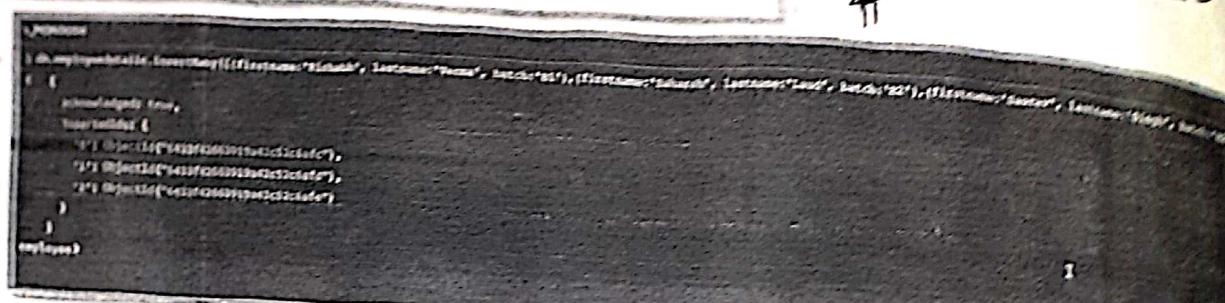
Total marks _____ out of 10.

Sign of Faculty

Step 04: In this screenshot, we can see Datafields & rows/tuples, i.e. data has been inserted into the database.

Format of DB-data: JSON

Step 05: Here, we are using MongoShell, to insert Many records at a given time.



```
db.employee.insertMany([{"EmployeeID": "E1001", "EmployeeName": "Rakesh", "Batch": "S1"}, {"EmployeeID": "E102", "EmployeeName": "Karan", "Batch": "S2"}, {"EmployeeID": "E103", "EmployeeName": "Gaurav", "Batch": "S3"}, {"EmployeeID": "E104", "EmployeeName": "Rahul", "Batch": "S4"}])
```

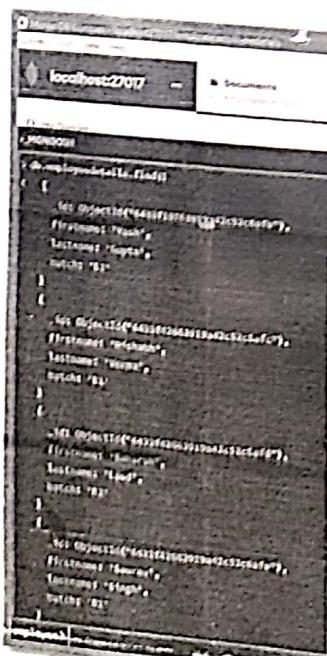
Step 06: Here, we are just changing the view of database.

Means: → displaying documents in a tabular view as provided by MongoDB compass.

Step 07: In this screenshot, performing READ operation using the inbuilt function of MongoDB i.e. .find().

After executing the command in MongoShell,

we will get all the (4 Objects = 4 Docs = 4 Records)



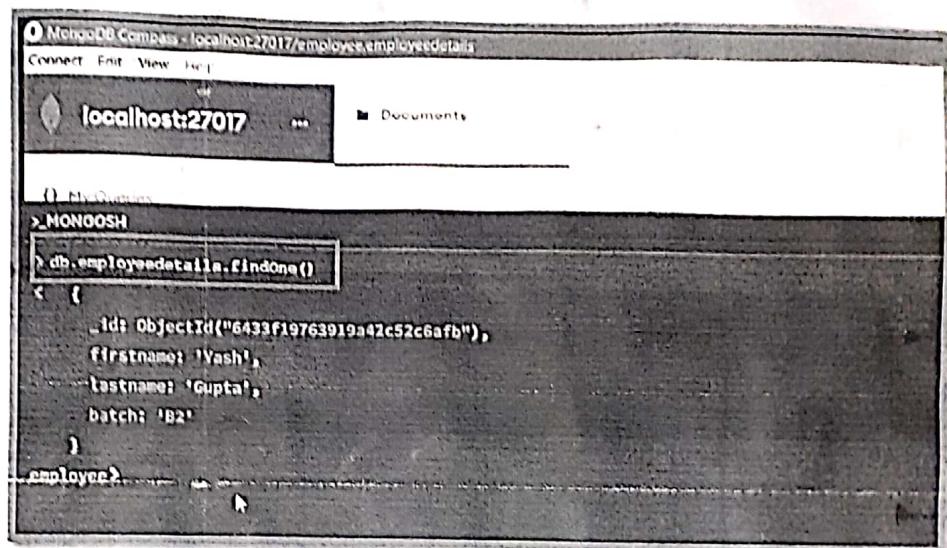
```
localHost:27017 - Documents
```

```
db.employee.find()
[{"EmployeeID": "E1001", "EmployeeName": "Rakesh", "Batch": "S1"}, {"EmployeeID": "E102", "EmployeeName": "Karan", "Batch": "S2"}, {"EmployeeID": "E103", "EmployeeName": "Gaurav", "Batch": "S3"}, {"EmployeeID": "E104", "EmployeeName": "Rahul", "Batch": "S4"}]
```

Step-08: Here, we are querying to find all those Records=Rows which have 'BATCH:B2'
 (batch)
 $\{ \text{batch: } \underline{\text{B2}} \} \rightarrow$ & as an output we get all those Records.

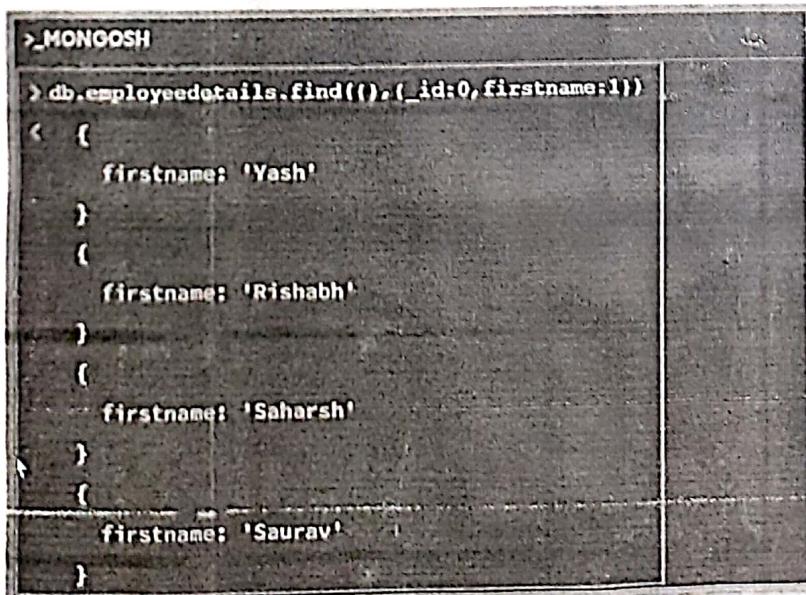
Step-09: In this screenshot, we are using Inbuilt functionality of function of MongoDB i.e. findOne()

L.O/P: As an output, we will get the FIRST { -Id:0 } Record of the Table.



The screenshot shows the MongoDB Compass interface. The query bar contains the command: `> db.employeedetails.findOne()`. The results pane displays a single document:

```
_id: ObjectId("6433f19763919a42c52c6afb"),
  firstname: 'Yash',
  lastname: 'Gupta',
  batch: 'B2'
```



The mongo shell command is: `> db.employeedetails.find({}, {_id:0, firstname:1})`. The output shows multiple documents, each with the `firstname` field and no `_id`:

```
< {
  firstname: 'Yash'
}
< {
  firstname: 'Rishabh'
}
< {
  firstname: 'Saharsh'
}
< {
  firstname: 'Saurav'
}
```

Step-10: In this screenshot, we are querying the database to find all those (Records / Rows/Tuples) where

$\begin{cases} \rightarrow \text{-id:0,} \\ \rightarrow \text{firstname:1} \end{cases} \rightarrow B$

L as an output, we are getting all the firstname: 1, using (.find)

Step-01: In this screenshot, as our database employee is already created. Also, the data tuples are stored.

Now, we want to update SINGLE RECORD

Therefore, mongoDB cloud shell provide a function: `updateOne()`

} Here, enter the data which we want to update.

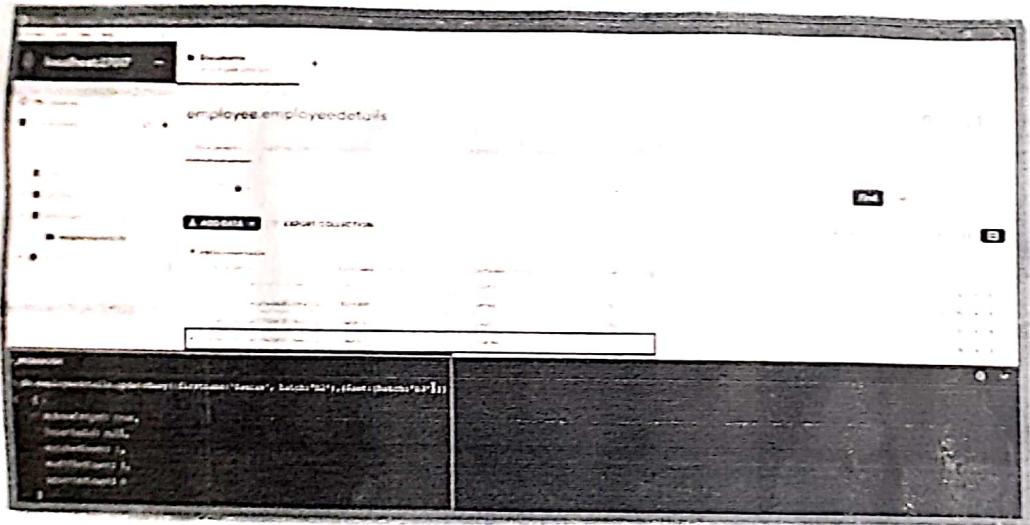
↳ `updateOne()`. func^{n°}:

CRUD

↳ UPDATE operation

↳ but only on ONE Record.

i.e. (updating the details of One Record)



Step-02:

In this screenshot, we are updating

↳ Many data fields (row) tuples in one (single command)

Mongo-DB has provided one Inbuilt function i.e. `updateMany({ })`

CRUD

(UPDATE operation)

Here, we are updating all the record where 'firstname: Savar'

8 (And)

'batch: B2'

There set the 'batch: B3' & update it in the database

SKILL ACTIVITY NO: 3
(To be filled by the Student)

Date: 15/03/2023

Title:

Implementing CRUD Operation : Illustration of UPDATE & DELETE command of MongoDB

1. What is the purpose of this activity ? (Explain in 3 – 4 lines)

The purpose of this activity are as follows:-

- (i) To understand what are CRUD operations in MongoDB {document based DB}.
- (ii) To understand how these CRUD Operations perform as compared to SQL database? {Complexity wise analysis} using perf-schema db given in MongoDB.
- (iii) To understand and implement the UPDATE & DELETE operations in a NOSQL database i.e. MongoDB Community Server v4.4.5.
- (iv) Real-world application: → via creating & updating & deleting the rows/tuples in MongoDB database engine.

2. Step performed in this activity (Explain in 5 – 6 lines)

Steps performed in this activity are as follows:-

Step(i) From previous skill activity we already launched the DB Server, created the collection name. Now, we are using CRUD operation i.e. UPDATE & DELETE

Step(ii) For UPDATE Operation in Database ↗ use (\$set) func.
↳ (a) Execute .updateOne() & as argument
↳ (b) Execute .updateMany() & as argument ↗ use (\$set) func.

Step(iii) For DELETE Operation in Database. ↗ use (\$set) func.
↳ (a) Execute .deleteOne() & as argument
↳ (b) Execute .deleteMany() & as argument ↗ use (\$set) func.

3. What resources / materials / equipments / tools did you use for this activity?

1. Online Resources:

1. ↳ MongoDB Community Server v4.4.5
(<https://www.mongodb.com/trial/download/community>)
2. ↳ MongoDB Official Documentation
(<https://www.mongodb.com/docs/atlas>)
3. ↳ MongoDB Compass Download
(<https://www.mongodb.com/download-center/compass>)

5. Equipments:

5. PC Intel iR-Core(TM) i5-6400@2.70GHz
6. LENOVO MT-30 AC-BU
6. LENOVO FM ThinkStation P310
6. Microsoft Windows 10 Pro 10.0.19044.1521
6. Greenshot → Screenshot application.

7. Materials:

7. ↳ Lecture PPT for CRUD operations

4. What skills did you acquire?

1. W5HH abt UPDATE & DELETE operations.

2. ↳ How to execute it in MongoDB?

3. Implementation of queries on the MongoDB compass.

4. Algorithms behind the Inbuilt function provided by MongoDB such as
• updateOne()
• updateMany()

5. (Query Evaluation Engine of MongoDB at the RuntimE of servEY.)

6.

7. Reading the MongoDB ATLAS dashboard to understand the Cluster metrics.

8. Understanding how JSON format schema is used for argument.

RAM No. of Hits Miss

5. Time taken to complete the activity?

02:00

(hours)

✓ Gupta ✓

(To be filled by Faculty)

S. No.	Skills / Competencies	(Achieved / Not Achieved) (YES / NO)
01.	Performing CRUD operations (i.e. UPDATE & DELETE) operations on the MongoDB Community Server (v6.0)	
02.	Implementing and data Updation, Deletion of entries in our database	
03.	Implementation of inbuilt functions provided by MongoDB. i.e. {updateOne(), updateMany(), deleteOne(), deleteMany()}	
04.	Real world usecases of MongoDB	

Remarks

Total marks _____ out of 10.

Sign of Faculty

Date:

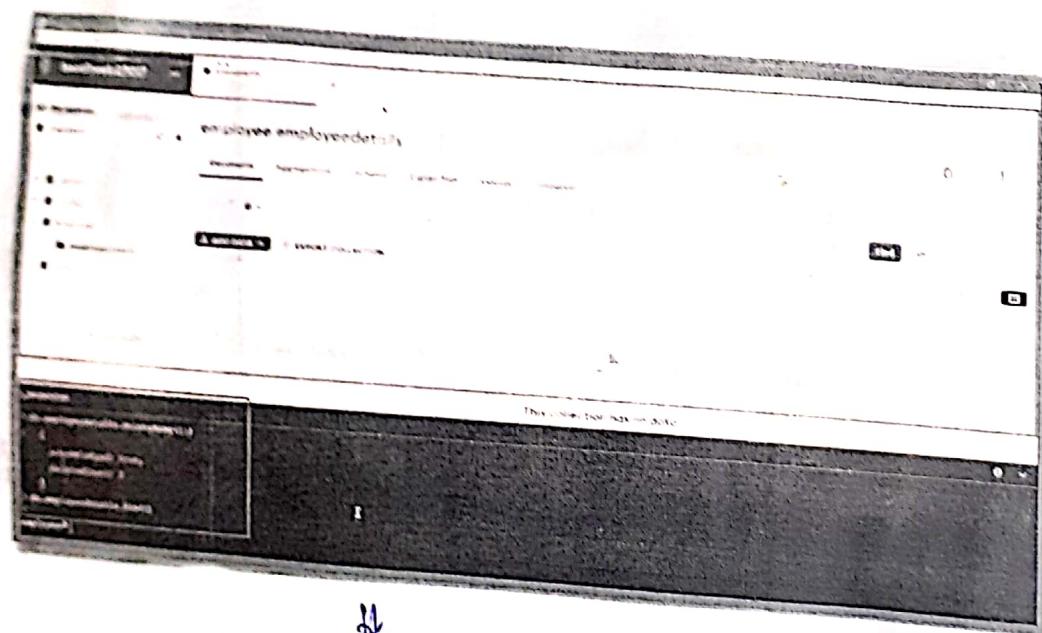
Step 03: In this screenshot, we are performing one of the CRUD operation i.e.

DELETE

using Inbuilt function of MongoDB database i.e.

.deleteOne()

Now, Here we are deleting that datafield
WHERE firstname: Saurav



Now,

Now, Here in this screenshot.

(i) We execute 'MongoShell'

→ then we Run '.deleteMany()' function.

(ii) Entire database {employee} & (employeeDetails)

Step 04: In this screenshot,

we are performing DELETE

operation & DELETING Many Records of the MongoDB database (i.e. employeedetails)

at the same point of time.

.deleteMany({ })

MONGODB CRUD Operations

- (i) CREATE : Create or Insert operations add new documents to a collection. If the collection doesn't currently exist, Insert operations will create the operation.

Eg:

```
db.employeedetails.insertOne()  
{  
    first name: "Saurav",  
    last name: "Singh",  
    batch: "B20".  
}
```

MongoDB provides the following methods to insert documents into a collection:

↳ db.collection.insertOne()
↳ db.collection.insertMany()

In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on a level of single document.

- (ii) READ: Read operations retrieve documents from a collection, i.e. a query a collection for documents.

Eg:

```
db.employeedetails.find();
```

MongoDB provides the following methods to read documents from a collection:

↳ db.collection.find()

You can specify query filters or criteria that identify that documents that return.

- (iii) UPDATE:

Update operations modify existing

in a

MongoDB provides the following methods to update documents of a collection:

↳ db.collection.updateOne()

↳ db.collection.updateMany()

↳ db.collection.replaceOne()

In MongoDB, update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

- (iv) DELETE:

Delete Operations remove documents from a collection.

Eg:

```
db.employeedetails.deleteOne()  
db.employeedetails.Many()
```

MongoDB provides the following methods to delete documents of a collection:-

↳ db.collection.deleteOne()

↳ db.collection.deleteMany()

In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

CODE REPRESENTING MONGODB

```

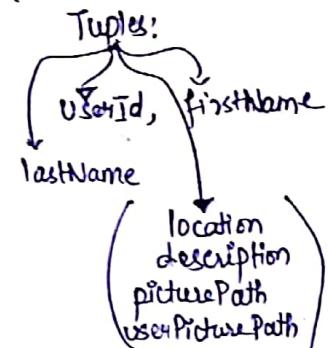
File Edit Selection View Go Run Terminal Help
Post.js - Post.js (1 tab)
OPEN EDITORS
MERN SLIDES
  include
  client
  server
  controllers
  index.js
  middleware
  models
    Post.js
      User.js
      node_modules
      public
      routes
    env
    index.js
      package-lock.json
      package.json
      README.md
      .gitignore
    OUTLINE
    TIMELINE
Live Share
Post.js
1 import mongoose from "mongoose";
2
3 const postSchema = mongoose.Schema({
4   userId: {
5     type: String,
6     required: true,
7   },
8   firstName: {
9     type: String,
10    required: true,
11   },
12   lastName: {
13     type: String,
14     required: true,
15   },
16   location: {
17     type: String,
18     required: true,
19   },
20   description: {
21     type: String,
22     required: true,
23   },
24   picturePath: {
25     type: String,
26     required: true,
27   },
28   userPicturePath: {
29     type: String,
30     required: true,
31   },
32   likes: {
33     type: Map,
34     of: Boolean,
35   },
36 });
  
```

→ FileName: Post.js

→ Code Explanation:

↳ Creating (FORM) (a) schema for our project.

(b) Rows: ≡ Docs



```

File Edit Selection View Go Run Terminal Help
Post.js - Post.js (1 tab)
OPEN EDITORS
MERN SLIDES
  include
  client
  server
  controllers
  index.js
  middleware
  models
    Post.js
      User.js
      node_modules
      public
      routes
    env
    index.js
      package-lock.json
      package.json
      README.md
      .gitignore
    OUTLINE
    TIMELINE
Live Share
Post.js
14   type: String,
15   required: true,
16 },
17   location: String,
18   description: String,
19   picturePath: String,
20   userPicturePath: String,
21   likes: {
22     type: Map,
23     of: Boolean,
24   },
25   comments: {
26     type: Array,
27     default: [],
28   },
29   timestamps: true
30 },
31 },
32 const Post = mongoose.model("Post", postSchema);
33
34 export default Post;
  
```

→ file Name: Post.js

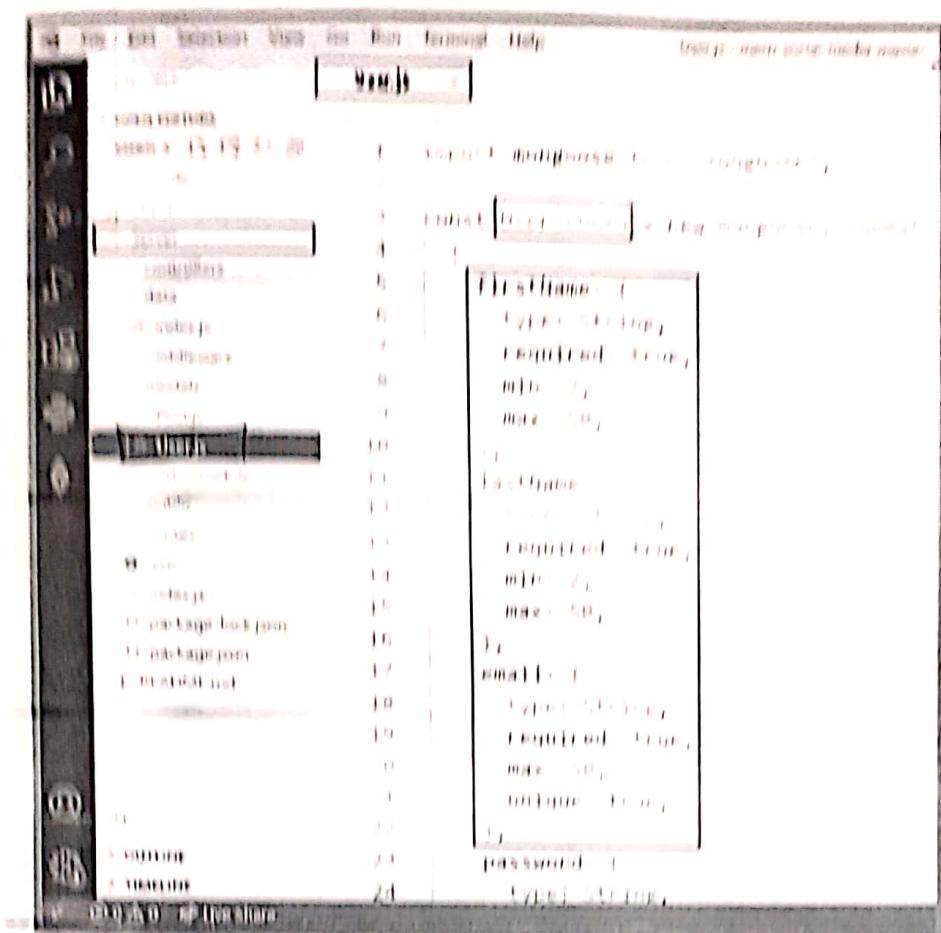
→ Code Explanation:

Here, we are storing various data-fields such as:

location:] string,
description:] string

comments ; likes

Type: Array Type: Map



1001.15
Plate No. 1

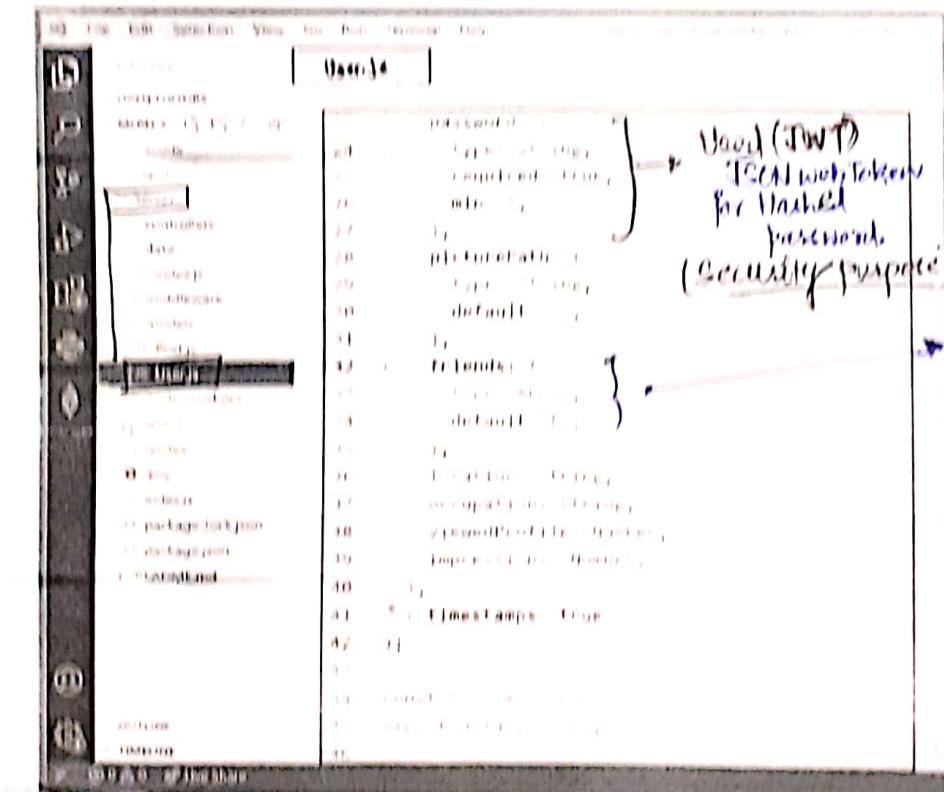
→ Code-Dekomposition

→ In this exercise
file, we are creating a database.

```

graph TD
    data["data"] --> username["username"]
    data --> password["password"]
    data --> picturePath["picturePath"]
    data --> friends["friends"]
    friends --> affinestamps["affinestamps"]

```



->FileName: D:\ang

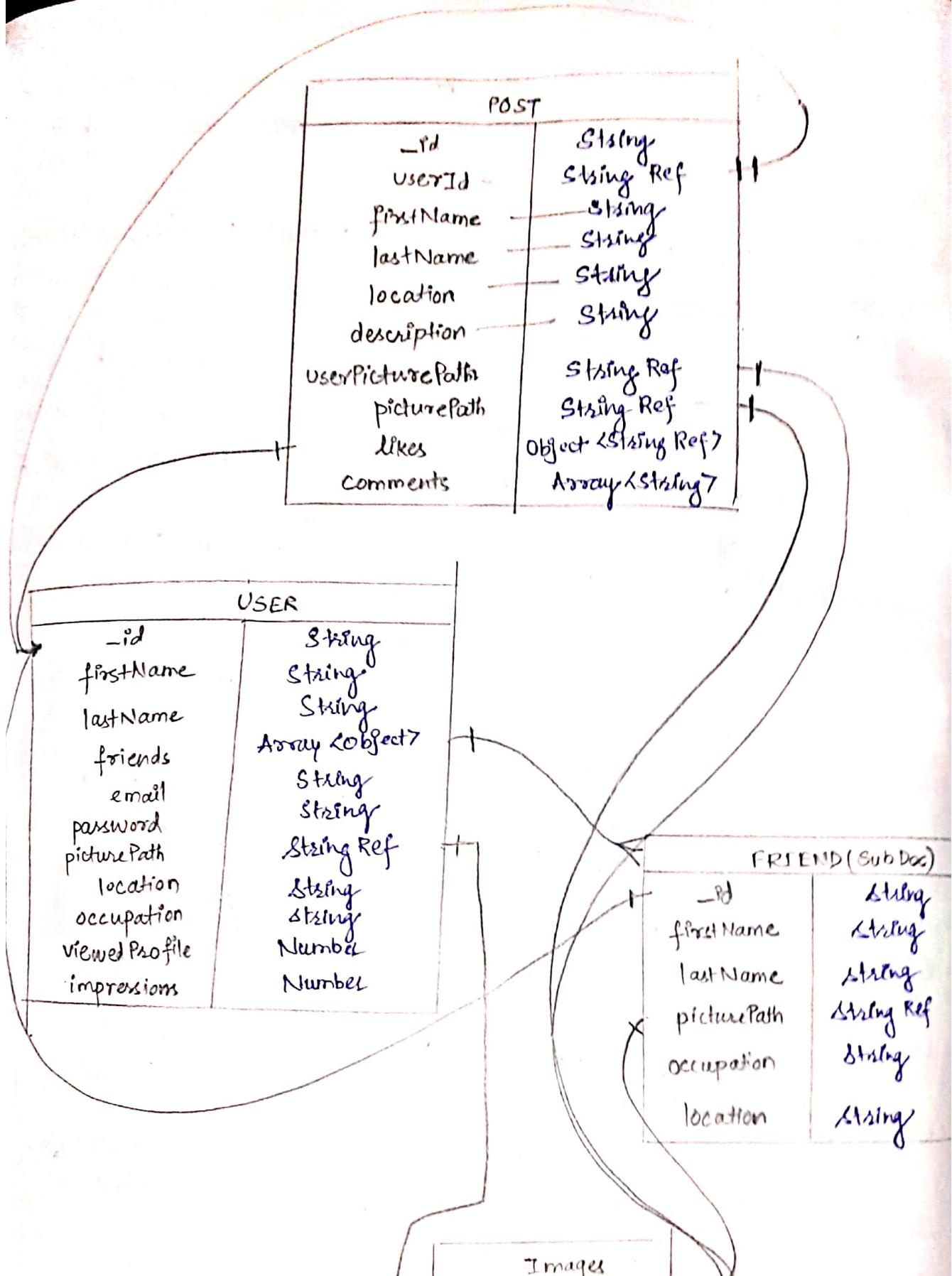
→ Code Explanations

We are exploring

"friends" feature

→ Every minute to
Facebook, Twitter?

So the datatype is "Array"



DATABASE SCHEMA ER DIAGRAM UML

SKILL ACTIVITY NO: 4
(To be filled by the Student)

Date: 23/03/2023

Title:

Designing a NoSQL Database for a Project

1. What is the purpose of this activity? (Explain in 3 – 4 lines)

The purpose of this activity are as follows:-

- (i) To understand the need of database - [which we need SQL databases vs NoSQL - finding the tradeoff between the SQL and NoSQL database.]
 - Eg: Properties of Document Based DB.
- (ii) To understand in NoSQL database, which one is best for our project we can.
- (iii) Now designing the database schema, what will the structure of our database.
 - ↳ How many fields are required?
 - ↳ What will be the minimum size (byte/bitsize) of various fields & also their datatype.
- (iv) Future scope of our database schema
 - Optimization
 - Master-Slave Cluster Architecture
 - for better manageability of database.

2. Step performed in this activity (Explain in 5 – 6 lines)

Steps performed in this activity are as follows:-

Step(i): Firstly, understand all the requirements of database design i.e (schema). Then design it and program it using Javascript.

Step(ii): Open VSCode. Then create directory structure.

Step(iii) → As, the database is using POST method, therefore, we will

Step(iv) → Now, we will use 'Mongoose' Mongo-DB (SDK library) & implement the schema using Javascript code.

userId
dtype: String

firstName
dtype: String

lastName
dtype: String

likes
dtype: Map

3. What resources / materials / equipments / tools did you use for this activity ?

1. Online Resources:

1. ↳ MongoDB Community Server (<https://mongodb.com/tsh/download/community>)
2. ↳ MongoDB Official Documentation (<https://mongodb.com/docs/atlas>)
3. ↳ MongoDB JS SDK download. (<https://mongodb.com/download-center/compass>)

Dependencies:

react: '11.10.6';
style: '11.10.6';
non-minified: '5.11.16';
redux-dom: '18.2.0'

5. Equipments: Laptop: NB QBRJST, Intel(R) Core i5-8265U CPU @ 1.60GHz 1.80GHz
Memory: 8.00 GB (3.98 GB)

7. Materials:

1. ↳ VS Code Editor
2. ↳ github.com/YGBAT5D
3. ↳ db-diagrams.net

Node: (v18.15.0)
Express: (v4.8.12)

All of them downloaded

4. What skills did you acquire ?

1. WSMH about (ER) Diagram & (VML) diagram
- 2.
3. Realworld Implementation & designing of Database Schema in UML website.
- 4.
5. What, why & where for Data Modelling
6. Understanding about NoSQL database schema design
7. Difference b/w NoSQL document based schema & SQL Based Schema (design)
8. Defining, Planning of Datatypes of columns in MongoDB

5. Time taken to complete the activity ? 05:00 (hours)

Yash ✓

Signature of Student

(To be filled by Faculty)

S. No.	Skills / Competencies	(Achieved / Not Achieved) (YES / NO)
01.	Designing a NoSQL database using MongoDB Atlas. (Cloud Version of MongoDB)	
02.	Data Modelling and ER Diagrams ↳ Handmade Shortest I page of SK-04.	
03.	React Redux file folder architecture Understanding & Implementation.	
04.	Understanding of Redux Concept & Implementation.	
05.	Understanding of Frontend User Interface. & Implementation.)	

Remarks

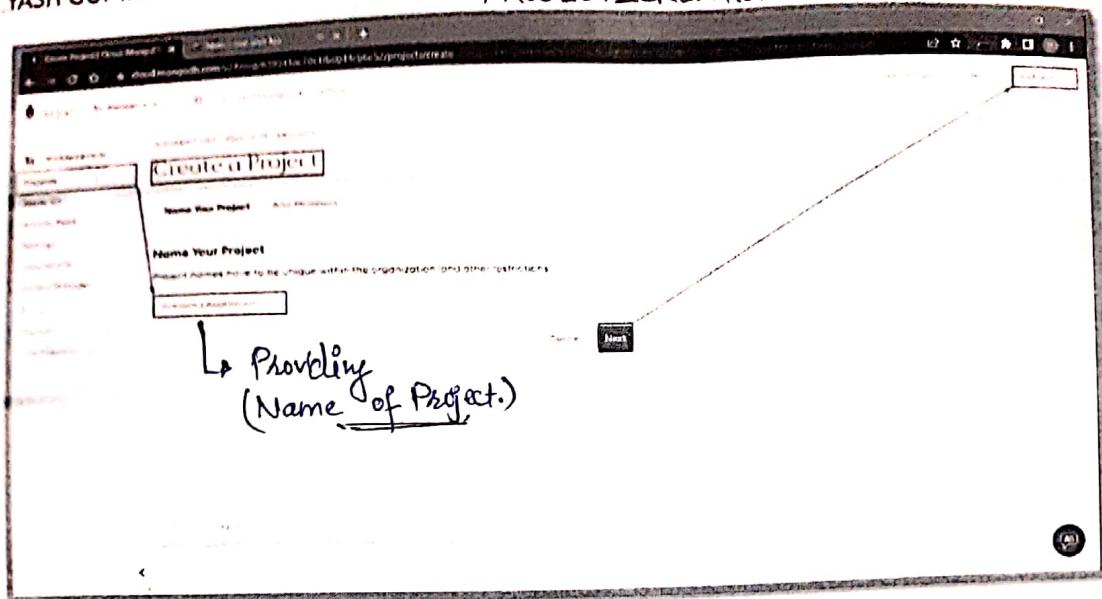
Total marks _____ out of 10.

Sign of Faculty

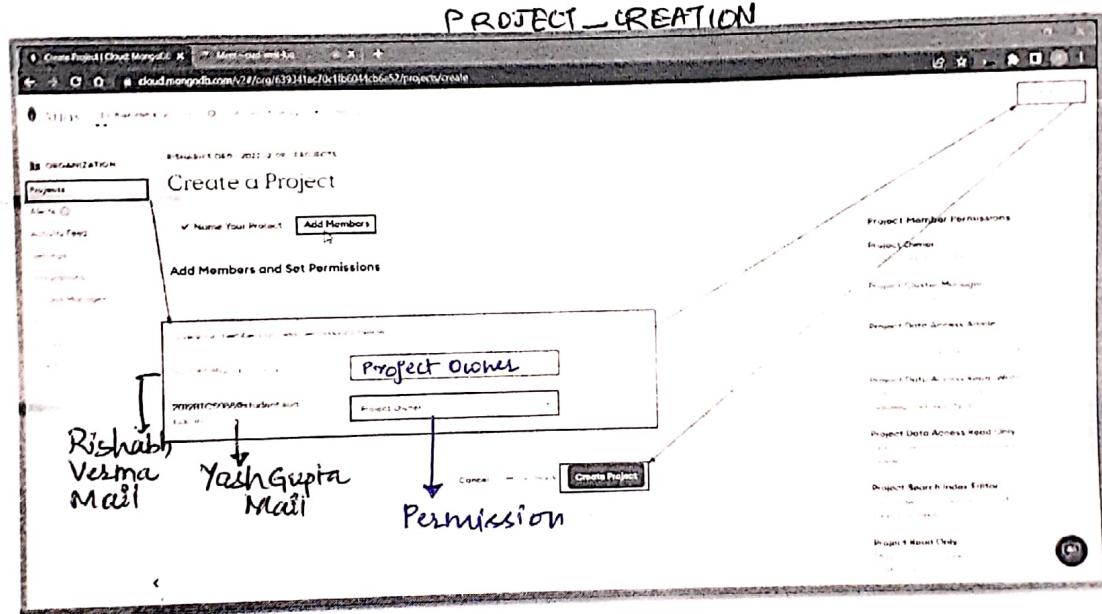
Date:

PROJECT_CREATION

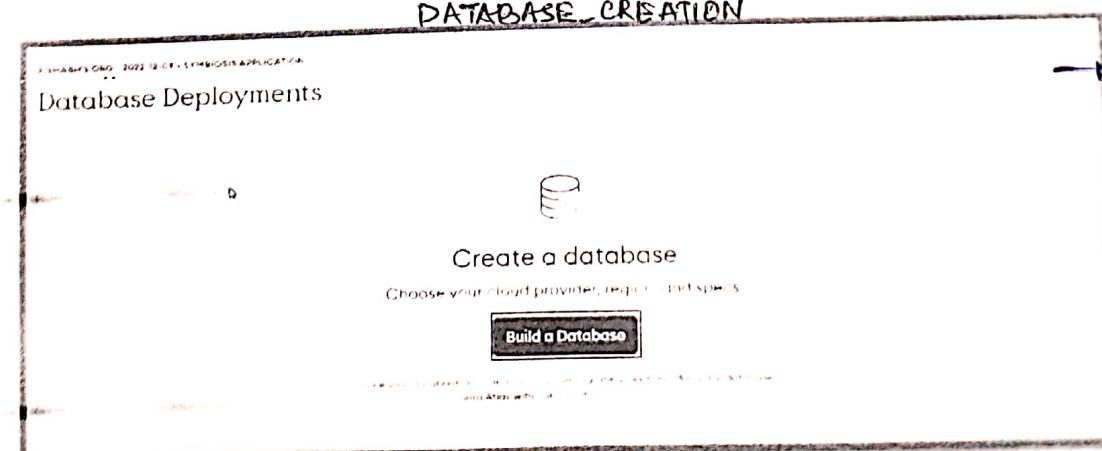
①



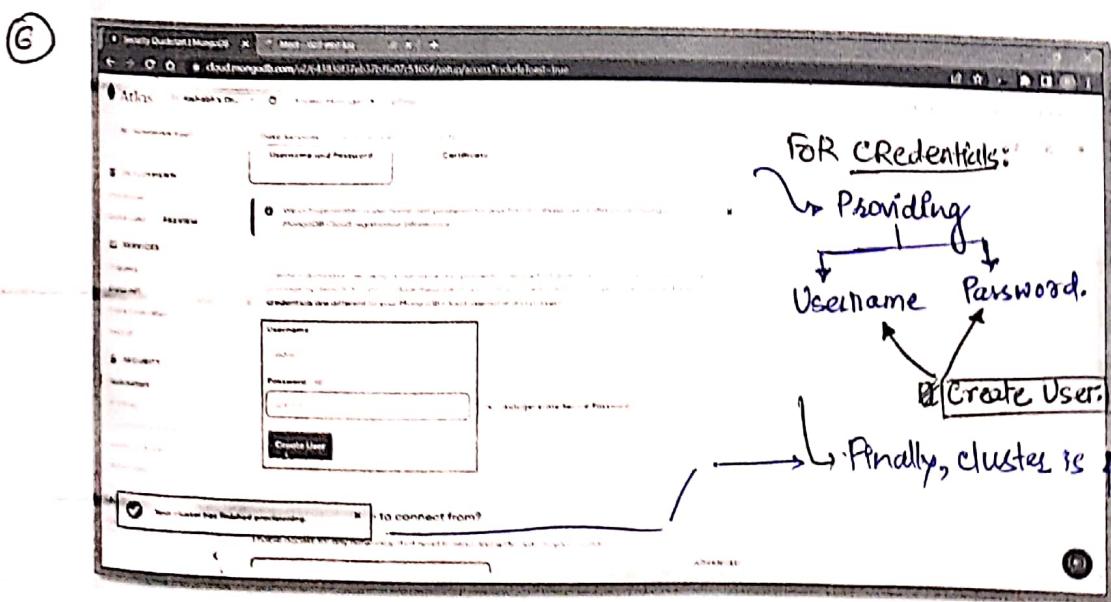
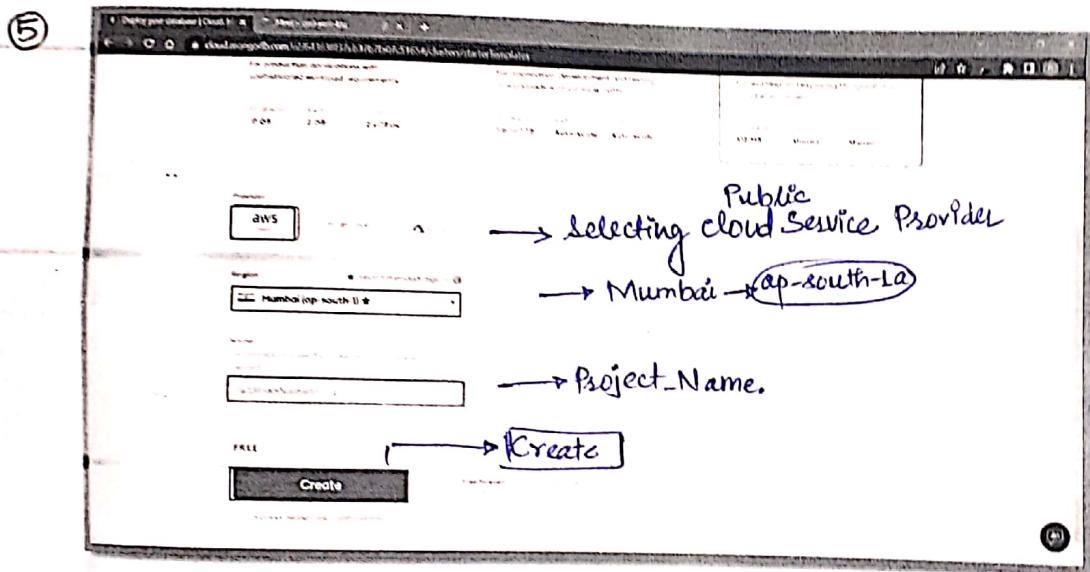
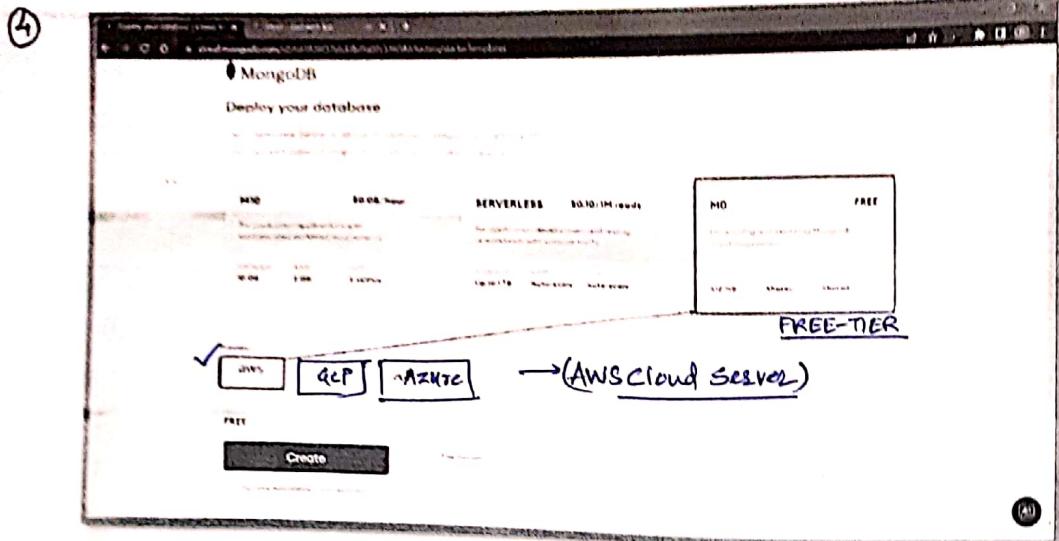
②

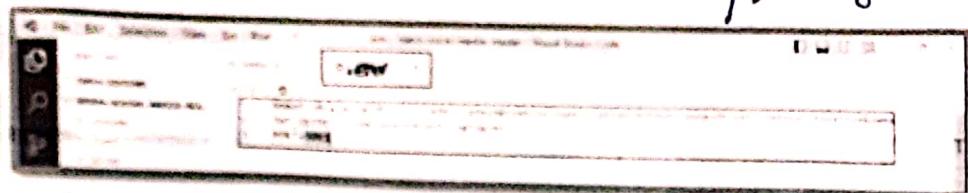


③

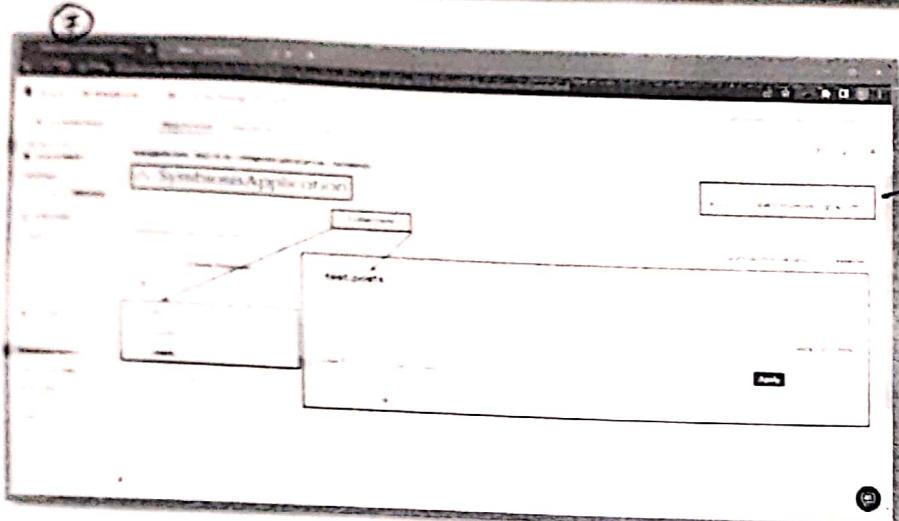


For deployment of database server
→ We are using [FREE-TIER]

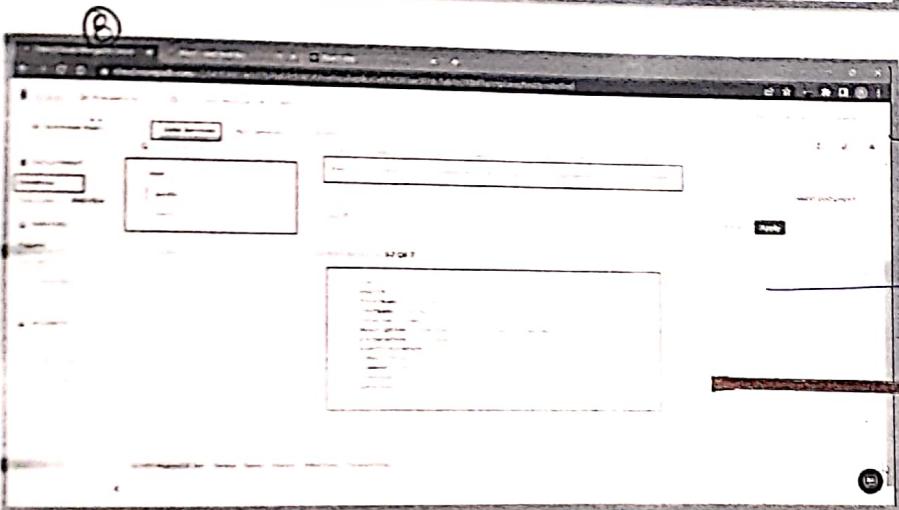




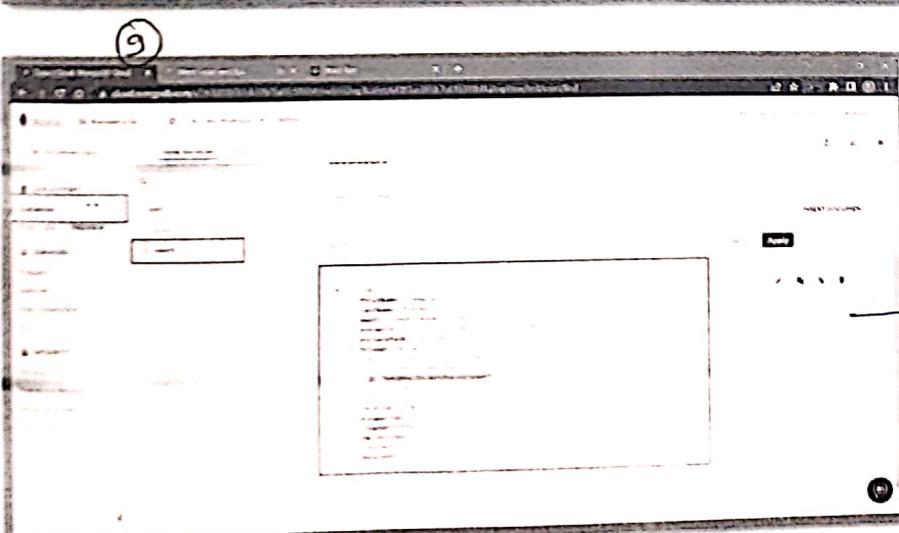
→ Setting up the URL where our cluster is deployed
→ Also, adding Username & password for authentication.
→ Post: In which server is running.



→ AWS Cloud Region
(Number ap-south-1)
where our server is deployed.



→ All the data which generated by Userform in Experimental Data is inserted in order to check about CRUD operations.
→ POST.js page
↳ Post in our

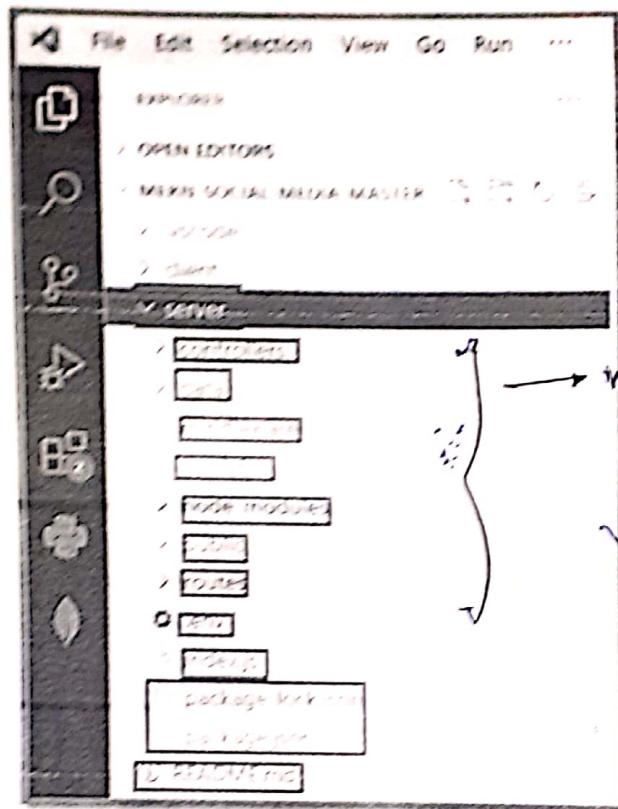


→ All the data which is credential Based
↳ Email
Password

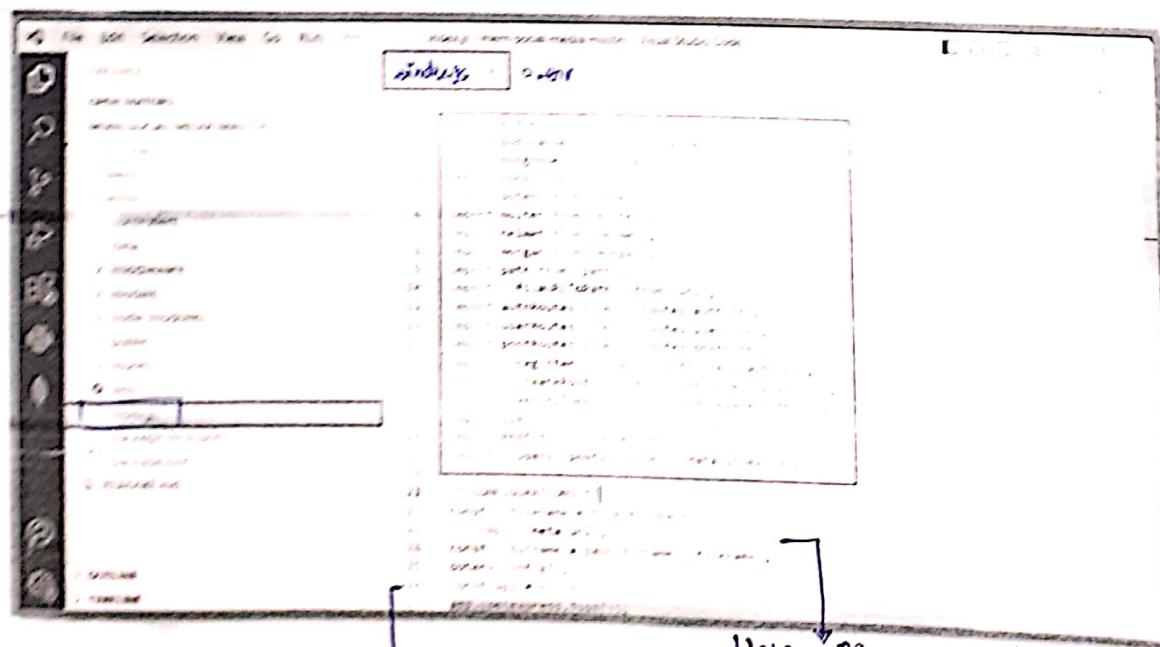
VASH GUPTA

SAR_05

Codebase: Server Side



various
directories
created as per
JS coding standards
in order to manage.



Line 26! (app is started
via 'Express' sever)

Here, filename & directoryname

are created.

```

1 // This is a simple express server
2
3 const express = require('express');
4 const cors = require('cors');
5 const mongoose = require('mongoose');
6 const path = require('path');
7
8 const app = express();
9
10 // Set static folder
11 app.use(express.static(path.join(__dirname, 'build')));
12
13 // Set cors
14 app.use(cors());
15
16 // Set mongoose
17 mongoose.connect('mongodb://127.0.0.1:27017/socialmedia', {
18   useNewUrlParser: true,
19   useUnifiedTopology: true,
20   useCreateIndex: true,
21   useFindAndModify: false
22 });
23
24 // Set port
25 const PORT = process.env.PORT || 3001;
26
27 // Add routes
28 app.use('/api', require('./routes/api'));
29
30 // Error handling
31 app.use((err, req, res, next) => {
32   console.error(err.message);
33   if (err.name === 'MongoError' && err.code === 11000) {
34     res.status(400).send(`User ${err.value.username} already exists`);
35   } else {
36     res.status(500).send(`Something went wrong! ${err.message}`);
37   }
38 });
39
40 // Start server
41 app.listen(PORT, () => console.log(`Server successfully up and running on port ${PORT}`));
42
43 module.exports = app;

```

→ Declaring (Routes with files) → ReactJS Routes

→ Implementation of CORS
→ FILE_STORAGE

Code explanation: Here, the Mongoose → Fun To connect with it, we are inserting Many users → posts

```

1 // This is a simple express server
2
3 const express = require('express');
4 const cors = require('cors');
5 const mongoose = require('mongoose');
6 const path = require('path');
7
8 const app = express();
9
10 // Set static folder
11 app.use(express.static(path.join(__dirname, 'build')));
12
13 // Set cors
14 app.use(cors());
15
16 // Set mongoose
17 mongoose.connect('mongodb://127.0.0.1:27017/socialmedia', {
18   useNewUrlParser: true,
19   useUnifiedTopology: true,
20   useCreateIndex: true,
21   useFindAndModify: false
22 });
23
24 // Set port
25 const PORT = process.env.PORT || 3001;
26
27 // Add routes
28 app.use('/api', require('./routes/api'));
29
30 // Error handling
31 app.use((err, req, res, next) => {
32   console.error(err.message);
33   if (err.name === 'MongoError' && err.code === 11000) {
34     res.status(400).send(`User ${err.value.username} already exists`);
35   } else {
36     res.status(500).send(`Something went wrong! ${err.message}`);
37   }
38 });
39
40 // Start server
41 app.listen(PORT, () => console.log(`Server successfully up and running on port ${PORT}`));
42
43 module.exports = app;

```

→ Here, we are booting up the NODEjs server in port 3001.
Is Running OR NOT?

→ Node Package Manager is started!

→ code for the
Tweet/comment posted by user
in His/Her profile

→ Lock & feel of the User's profile is ended.

```
File Edit Selection View Go Run authjs mem-social-media-main visual Studio Code

posts.js
1 import Post from "../models/Post.js";
2 import User from "../models/User.js";
3
4 /* CREATE */
5 export const createPost = async (req, res) => {
6   try {
7     const { userId, description, picture } = req.body;
8     const user = await User.findById(userId);
9     const newPost = new Post({
10       userId,
11       firstname: user.firstname,
12       lastName: user.lastName,
13       location: user.location,
14       description,
15       userPicturePath: user.picturePath,
16       picturePath,
17       likes: [],
18       comments: []
19     });
20     await newPost.save();
21
22     const post = await Post.findById(postId);
23     res.status(201).json(post);
24   } catch (err) {
25     res.status(400).json({ message: err.message });
26   }
27 }

/* READ */
28 export const getPosts = async (req, res) => {
29   try {
30     const posts = await Post.find();
31     res.status(200).json(posts);
32   } catch (err) {
33     res.status(400).json({ message: err.message });
34   }
35 }

users.js
1 import User from "../models/User.js";
2
3 /* READ */
4 export const getAllUsers = async (req, res) => {
5   try {
6     const users = await User.find();
7     res.status(200).json(users);
8   } catch (err) {
9     res.status(400).json({ message: err.message });
10    }
11  }
12
13 /* CREATE */
14 export const createUser = async (req, res) => {
15   try {
16     const { userId, firstname, lastName, location, description, userPicturePath, picturePath } = req.body;
17     const user = new User({
18       userId,
19       firstname,
20       lastName,
21       location,
22       description,
23       userPicturePath,
24       picturePath
25     });
26     const newUser = await user.save();
27     res.status(201).json(newUser);
28   } catch (err) {
29     res.status(400).json({ message: err.message });
30   }
31 }

/* READ */
32 export const getUser = async (req, res) => {
33   try {
34     const user = await User.findById(req.params.userId);
35     res.status(200).json(user);
36   } catch (err) {
37     res.status(400).json({ message: err.message });
38   }
39 }

/* UPDATE */
40 export const updateUser = async (req, res) => {
41   try {
42     const user = await User.findById(req.params.userId);
43     const updatedUser = {
44       ...user._doc,
45       ...req.body
46     };
47     const updateUser = await user.updateOne(updatedUser);
48     res.status(200).json(updateUser);
49   } catch (err) {
50     res.status(400).json({ message: err.message });
51   }
52 }

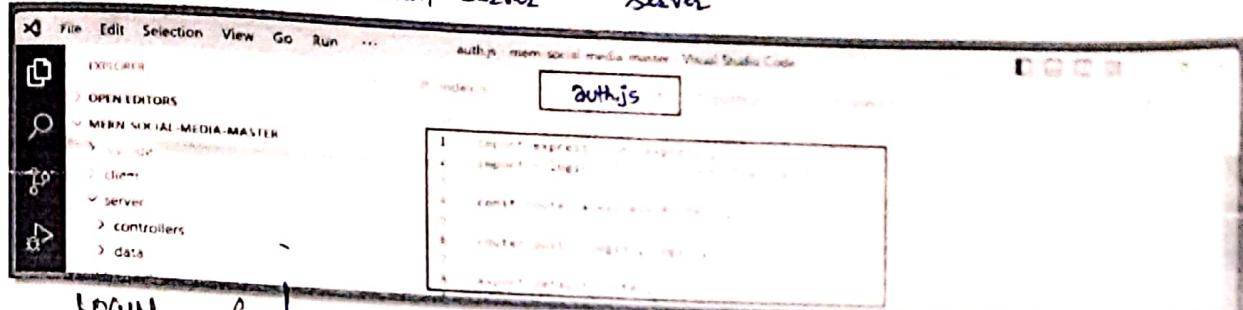
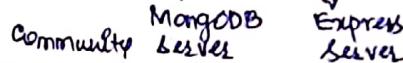
/* DELETE */
53 export const deleteUser = async (req, res) => {
54   try {
55     const user = await User.findByIdAndDelete(req.params.userId);
56     res.status(200).json(user);
57   } catch (err) {
58     res.status(400).json({ message: err.message });
59   }
60 }

library: bcrypt, passport
authjs
```

2019BTCS088

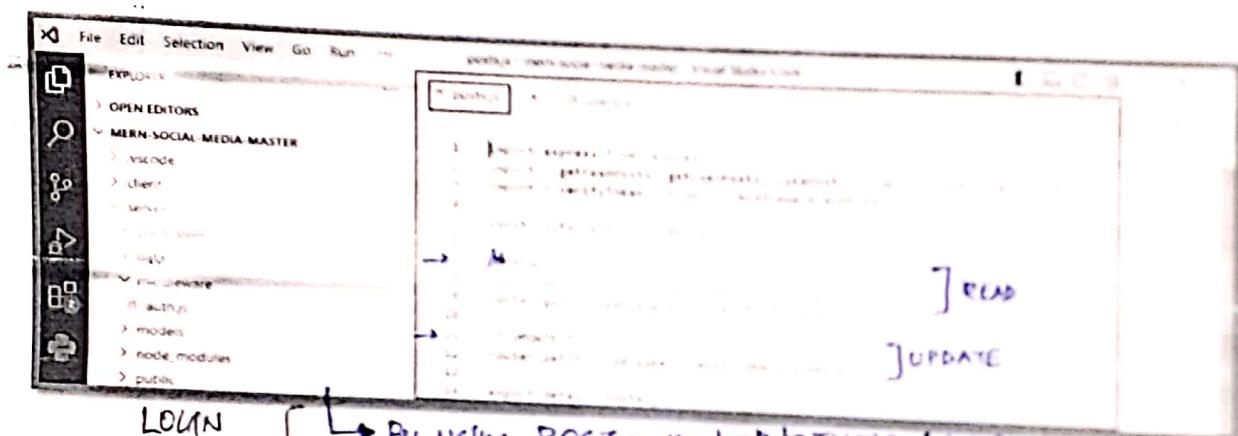
YASH GUPTA

AUTHENTICATION PAGES in (Tavascrip)



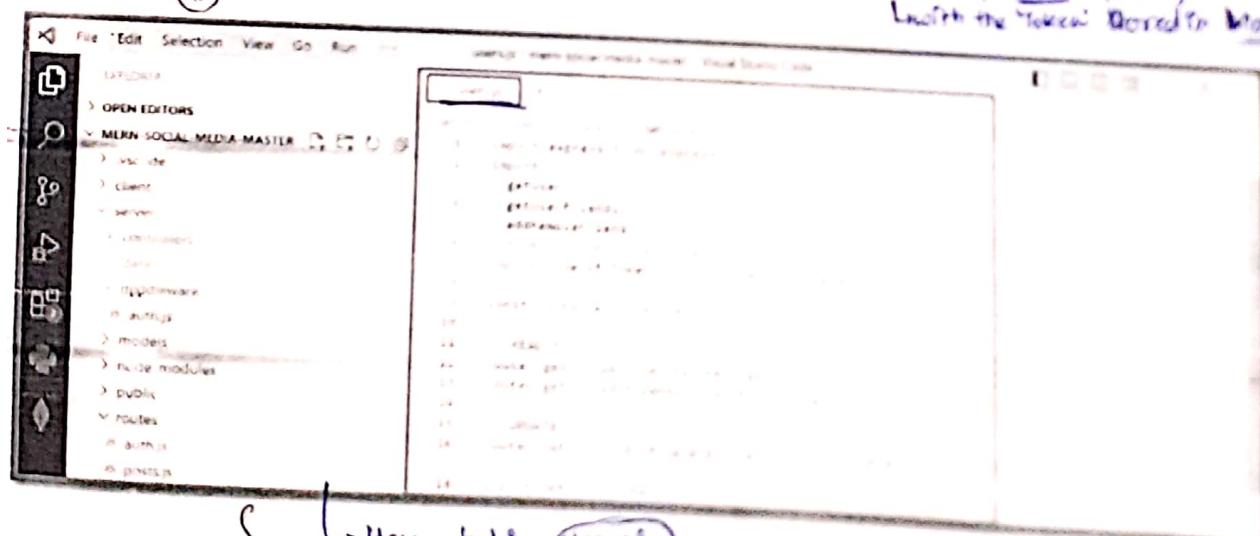
LOGIN Authentication Process

→ We are initiating the connection by using Express sever with Login page



LOIN
Authentication
Process
(ii)

- By using, POST method
 - to INSERT (data)
 - & take it as input from user
 - And verification of "Token" {JWT}
 - With the "Token" stored to MongoDB



↳ Here, taking `Users.js`,

→ GetUser } → These two methods are called for
→ GetUserfriends } displaying the details.

SKILL ACTIVITY NO: 5

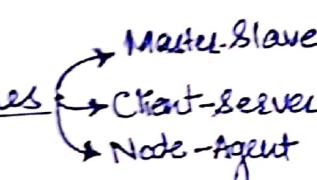
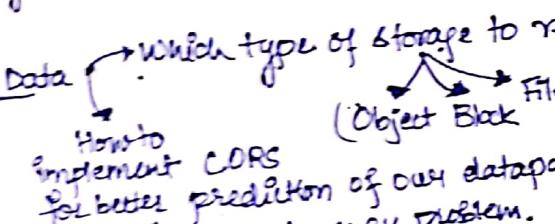
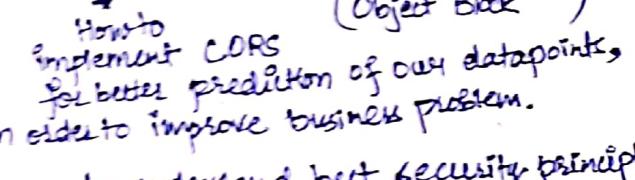
(To be filled by the Student)

Date: 19/04/2023

Title: Implementation of a Database Management Project using MongoDB as Backend.

1. What is the purpose of this activity? (Explain in 3 - 4 lines)

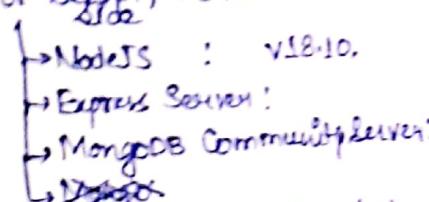
The purpose of this activity are as follows:-

- (i) To understand about various types of Software architectures  In which one fulfills the requirement in our Project.
- (ii) To understand how to deploy our Database Server, on which Public Cloud Service Provider, which region to deploy. Also, to ensure Maximum security, as data is MOST CRUCIAL.
- (iii) To understand better on Storage of Data  which type of storage to refer?
Therefore, we deploy MongoDB in AWS Cloud.
- (iv) For Authentication & Authorization:
↳ Using Email, Password  Note to understand best security principles acc. to our need & cost trade-off. We used which one tool/library?

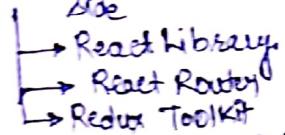
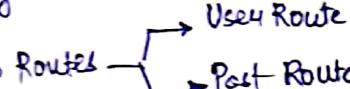
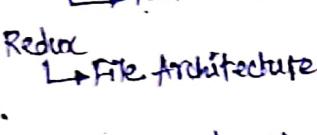
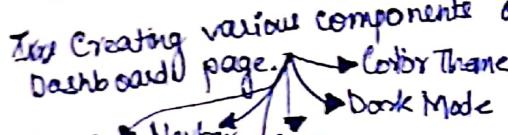
2. Step performed in this activity (Explain in 5 - 6 lines)

Steps performed in this activity are as follows:-

SERVER SIDE

- Step(I) For server, we need to install 
→ NodeJS : v18.10.
→ Express Server.
→ MongoDB Community Server.
→ Docker
- Step(II) Then start creating the data modelling & ERDiagrams which provides flow of database schema.
- Step(III) Created credentials of NodeJS server for authentication.
- Step(IV) Adding Backend data & taking demo (if it's working properly)

CLIENT-SIDE

- Step(V): For Client, we need to install 
→ React Library
→ React Router
→ Redux Toolkit
- Step(VI): Using the credentials provided by NodeJS server & under controllers inserting the logical business logic.
- Step(VII): Setting up Routes 
→ Use4 Route
→ Post Route
- Step(VIII): Setting React Router & React Redux 
→ File Architecture
- Step(IX): For Creating various components of Dashboard page. 
→ Color Theme
→ Dark Mode
- ↳ Homepage, Navbar, Styling & Setup, Widgets, Register, Login Pages & Form.

3. What resources / materials / equipments / tools did you use for this activity ?

1. Online Resources

1. [MongoDB Community QA](https://www.mongodb.com/community/qa/)
2. [MongoDB Official Doc](https://www.mongodb.com/docs/atlas/community/)
3. [MongoDB JS SDK Download](https://www.mongodb.com/docs/atlas/)
4. [MongoDB Compass](https://www.mongodb.com/download-center/compass)

5. Equipments: Laptop: NBQBRJST, Intel (R) Core™ i7

- PS-8265U CPU @ 1.60 GHz
- VS Code Editor

6. Materials: [github.com/YG3A5D]

8.

Node(v18.15.0)
Express: (v4.8.12)

DEPENDENCIES:

node: "16.10.0"
yarn: "1.19.6"
mongodb: "5.11.16"
aws-sdk: "3.16.0"
body-parser: "1.0.0"

4. What skills did you acquire ?

1. Understanding of Client-Server Architecture.
↳ How it works &
How to implement?
- 2.
3. WFMH about Node, Express, MongoDB
4. Testing: Understanding about
CTLC & implementing
Unit Testing in our
Project.
(Software
Testing Life
Cycle)

5. Middleware architecture implementation
using Node.js & Express.js
6. For Authentication & authorization
↳ Use of bcrypt & JWT
7. Database : CORS application.
↳ Launched our dbcluster on AWS cloud
Mumbai ap-south1 region & hoodie to
implement "CORS"
- 8.

5. Time taken to complete the activity ? 202:00 (hours)

Yogesh

Signature of Student

(To be filled by Faculty)

S. No.	Skills / Competencies	(Achieved / Not Achieved) (YES / NO)
01.	Installation of NodeJS Server & deploying it on local PC at port 3001.	
02.	Installing multiple dependencies required in project for frontend & Backend both i.e. React + React-dom, etc.	
03.	Using <u>bcrypt</u> and <u>JWT</u> for hashing our User password & following best practices of Authorization & Authentication.	
04.	Understanding of CORS <small>(Cross Origin Resource Sharing)</small> used in AWS S3 bucket.	
05.	Unit Testing performed for all the React components in <u>[Front-side, Page, Web]</u> <u>Interfaces</u> .	

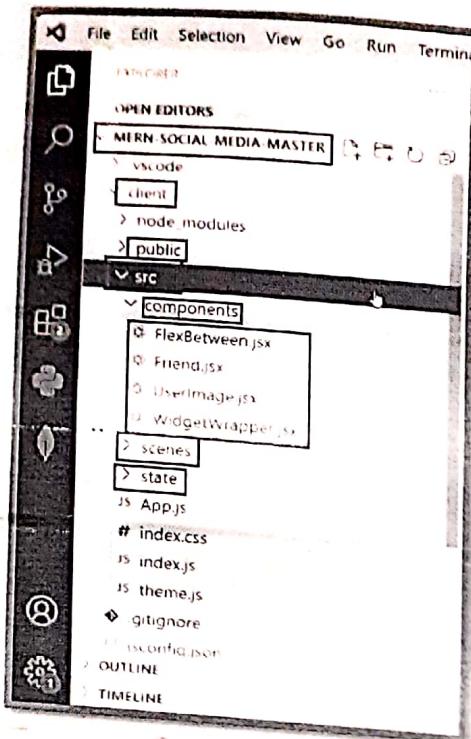
Remarks

Total marks _____ out of 10.

Sign of Faculty

Date:

Codebase: Client-Side



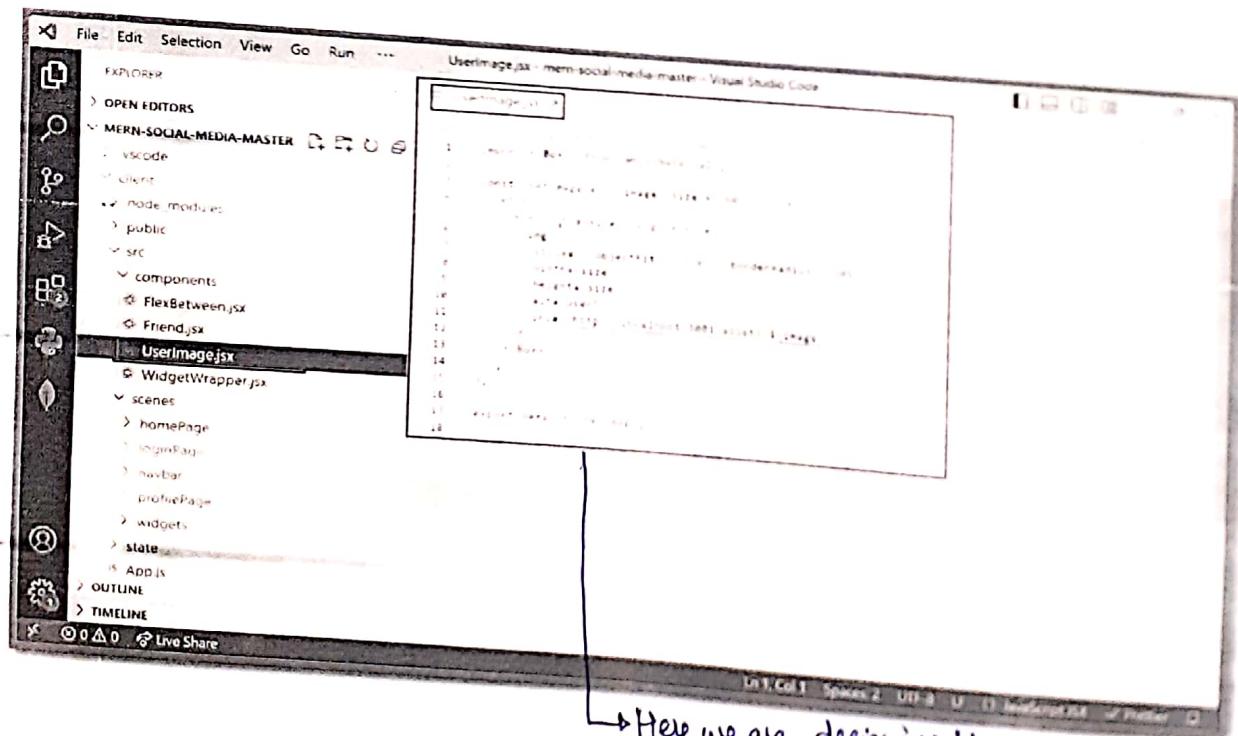
→ Screenshot showing (Client) directory tree

↳ Installed all dependencies & their version under node_modules directory.

→ Under 'src' directory,

↳ Created React-Components used in our project

- FlexBetween.jsx
- Friend.jsx
- UserImage.jsx
- WidgetWrapper.jsx



→ Here, we are designing how User's Image will display by defining the 'width' & 'height'.

Here, we created the Rule logic
if any user add another user
making friend

```
Friend.jsx - main/social-media/master - Visual Studio Code
```

```
1 // Importing required libraries for making API calls
2 import axios from 'axios';
3 import { useState, useEffect } from 'react';
4 import { Link, Navigate } from 'react-router-dom';
5 import { FlexBetween, FlexCenter, FlexRow } from 'ui';
6 import UserImage from 'ui/UserImage';
7
8 const Friend = ({ friendId, name, subtitle, interstitialLink }) => {
9   const dispatch = useDispatch();
10  const navigate = useNavigate();
11  const token = useSelector(state => state.user);
12  const tokenUser = useSelector(state => state.token);
13  const friend = useSelector(state => state.user.friend);
14
15  const palette = usePalette();
16  const paletteLight = palette.paletteLight;
17  const paletteDark = palette.paletteDark;
18  const paletteColor = palette.paletteColor;
19  const paletteText = palette.paletteText;
20
21  const [friendData, setFriendData] = useState(null);
22  const [error, setError] = useState(null);
23
24  const fetchFriend = async () => {
25    const response = await fetch(
26      `https://api.socialmedia.com/v1/friend/${friendId}`,
27      {
28        method: 'PATCH',
29        headers: {
30          Authorization: `Bearer ${tokenUser.access_token}`
31        }
32      }
33    );
34    const data = await response.json();
35    setFriendData(data);
36  };
37
38  useEffect(() => {
39    fetchFriend();
40  }, []);
41
42  return (
43    <FlexBetween>
44      <FlexCenter>
45        <UserImage user={friend}></UserImage>
46        <div>
47          <h3>{name}</h3>
48          <p>{subtitle}</p>
49          <button onClick={() => dispatch(navigate(interstitialLink))}>View Profile</button>
50        </div>
51      </FlexCenter>
52      <FlexCenter>
53        <button onClick={() => dispatch(navigate('/'))}>Logout</button>
54      </FlexCenter>
55    </FlexBetween>
56  );
57}
```

This (.JSON) file contains list of all dependencies used in our project.

These are the dependencies with their version.

→ This file contains the login Page form code.

```

File Edit Selection View Go Run Explorer Form.jsx MERN-SOCIAL-MASTER Visual Studio Code

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
    
```

The screenshot shows the Visual Studio Code interface with the file 'Form.jsx' open in the center editor. The code is a functional component that handles file uploads. It uses the 'useState' hook to manage state, including 'values' and 'onSubmitProps'. It creates a 'FormData' object, appends files from 'picturePaths' to it, and then sends a POST request to 'http://localhost:9001/api/auth/login' with the method 'fetch'.

→ This file contains code of the DASHBOARD page.
where all the activities related to User gets displayed.

```

File Edit Selection View Go Run Explorer Index.jsx MERN-SOCIAL-MASTER Visual Studio Code

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
    
```

The screenshot shows the Visual Studio Code interface with the file 'Index.jsx' open in the center editor. The code is a functional component that imports various components like 'Report', 'ReportList', 'UserList', 'UserForm', 'UserList', 'UserForm', 'AdvertWidget', and 'AdvertWidget'. It then defines a function 'getUsers' which fetches user data from 'http://localhost:9001/api/users'. The component then maps over the fetched data to render 'UserList' and 'UserForm' components for each user.

OUTPUT of Symbiosis Application