MONGOOSE (MongoDB)

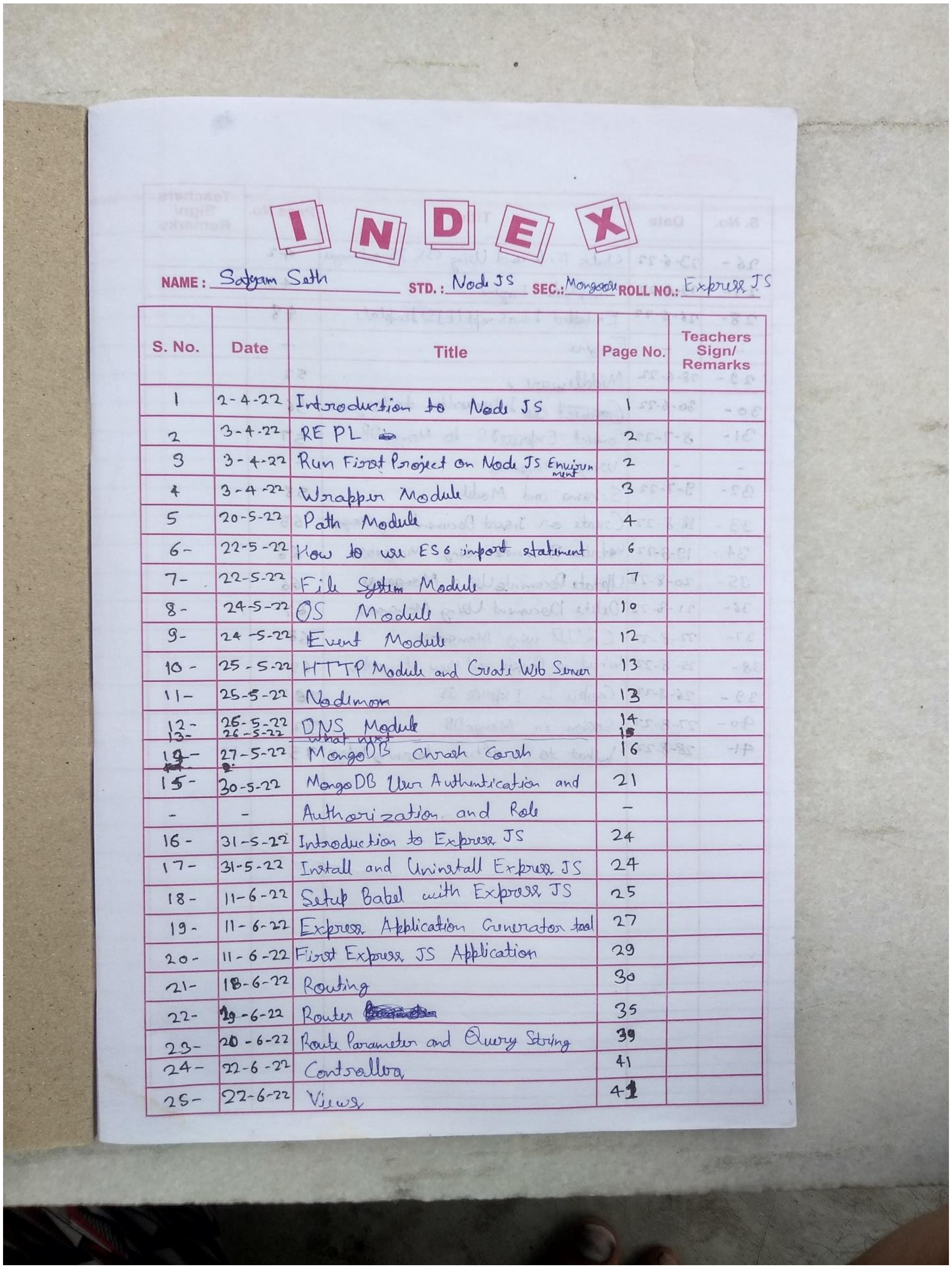
GEEKYSHOWS YOUTUBE CHANNEL LEARNING NOTES

Source Code - https://github.com/satyam-seth-learnings/mongodb_learning/tree/main/Geekyshows

YouTube Links -

1. https://youtu.be/02Y1Bciz8jk
2. https://youtu.be/JMPAzAgTfXU
3. https://youtu.be/uT-34VZxx5w

SATYAM SETH
[COMPANY NAME] | [COMPANY ADDRESS]



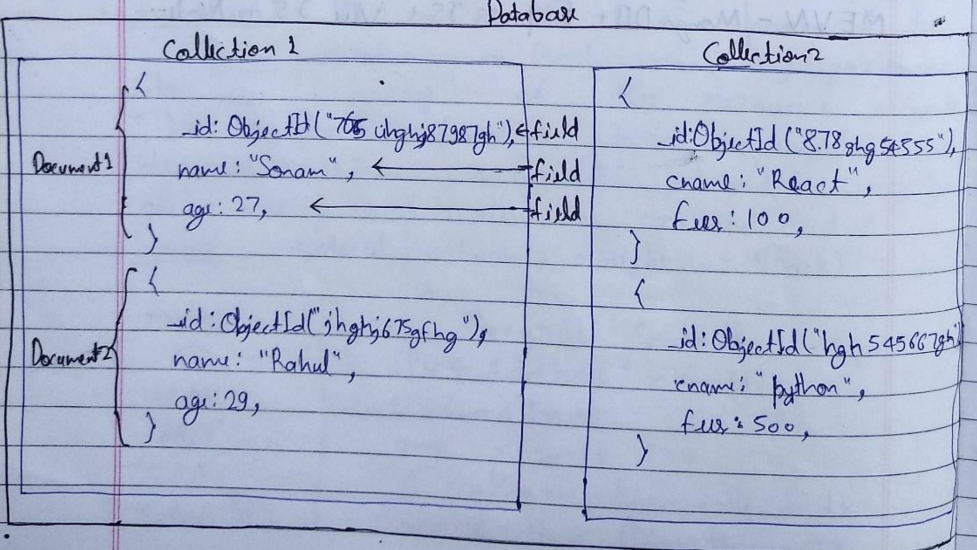
INDEX

NAME : Satyam Seth STD. : Node JS SEC.: Mongoose ROLL NO.: Express JS

S. No.	Date	Title	Page No.	Teachers Sign/ Remarks
1	2-4-22	Introduction to Node JS	1	
2	3-4-22	REPL	2	
3	3-4-22	Run First Project on Node JS Environment	2	
4	3-4-22	Wrapper Module	3	
5	20-5-22	Path Module	4	
6-	22-5-22	How to use ES6 import statement	6	
7-	22-5-22	File System Module	7	
8-	24-5-22	OS Module	10	
9-	24-5-22	Event Module	12	
10-	25-5-22	HTTP Module and Create Web Server	13	
11-	25-5-22	Nodemon	13	
12-	25-5-22	DNS Module	14	
13-	26-5-22	What next	15	
14-	27-5-22	MongoDB Crash Course	16	
15-	30-5-22	MongoDB User Authentication and	21	
-	-	Authorization and Role	-	
16-	31-5-22	Introduction to Express JS	24	
17-	31-5-22	Install and Uninstall Express JS	24	
18-	11-6-22	Setup Babel with Express JS	25	
19-	11-6-22	Express Application Generator tool	27	
20-	11-6-22	First Express JS Application	29	
21-	18-6-22	Routing	30	
22-	19-6-22	Router	35	
23-	20-6-22	Route Parameter and Query String	39	
24-	22-6-22	Controllers	41	
25-	22-6-22	Views	42	

Introduction to MongoDB- MongoDB is a document database designed for ease of development and scaling. It is one of most powerful NoSQL system and database. Being a NoSQL means that it does not use the usual rows and columns. This database uses a document storage format called BSON which is a binary style of JSON documents.

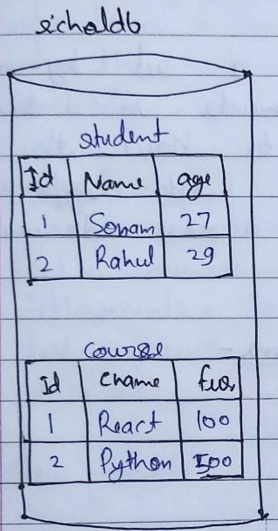
Ex-
```
{
    _id: ObjectId("6ab2665jhej7"),
    name: "Sonam",
    age: 27,
    hobbies: ['Dancing', 'Reading'],
    city: "Ranchi",
    islogin: true.
}
```



```
{
    _id: ObjectId("~~765~~ uhghj87987gh"),
    name: "Sonam",
    age: 27,
}
{
    _id: ObjectId("jhghj675gfhg"),
    name: "Rahul",
    age: 29,
}
```

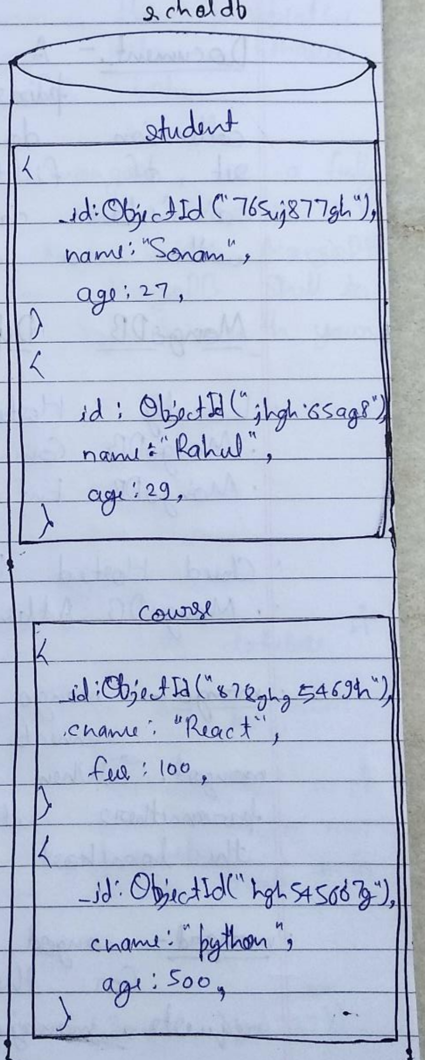
```
{
    _id: ObjectId("878ghg54555"),
    cname: "React",
    fees: 100,
}
{
    _id: ObjectId("hgh545667gh"),
    cname: "python",
    fees: 500,
}
```

MySQL	MongoDB
DataBase	DataBase
Table	Collection
Raw	Document
Column	Field



Id	Name	age
1	Sonam	27
2	Rahul	29

Id	cname	fee
1	React	100
2	Python	500



```
{
  _id: ObjectId("765uj877gh"),
  name: "Sonam",
  age: 27,
}
{
  _id: ObjectId("jhgh'65ag8"),
  name: "Rahul",
  age: 29,
}
```

```
{
  _id: ObjectId("878ghg5469h"),
  cname: "React",
  fee: 100,
}
{
  _id: ObjectId("hgh545067g"),
  cname: "python",
  age: 500,
}
```

Database - In MongoDB, databases hold one or more collections.

Collection - MongoDB stores documents in collections are analogous to tables in relational database.

Document - A document is a set key-value pairs. The documents in a single collection do not need to have the same set of fields and the data type for a field can differ across documents within a collection.

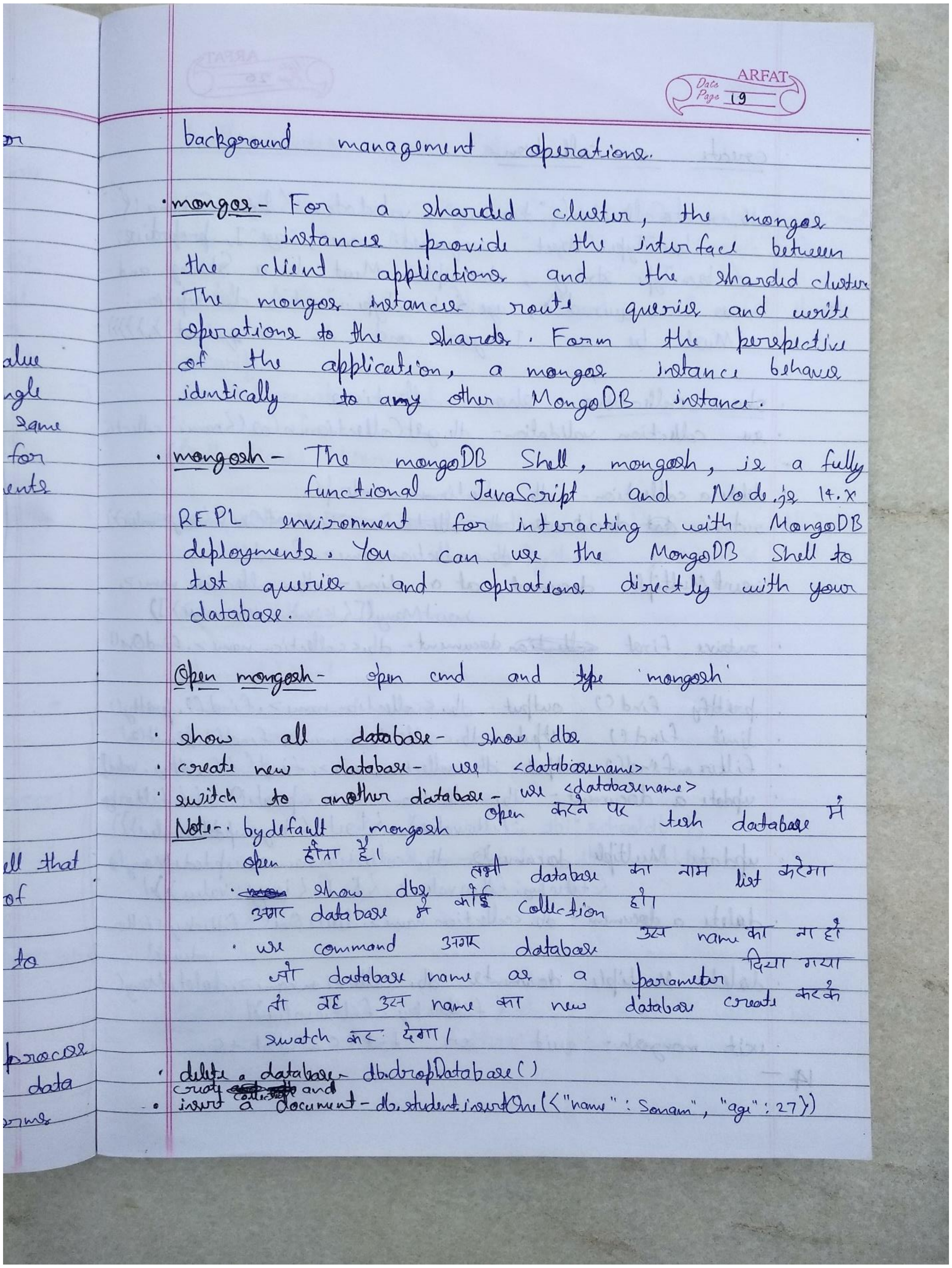
MongoDB Deployment Options -

- Locally Hosted Deployments -
- MongoDB Community
- MongoDB Enterprise Advance.

- Cloud Hosted Depolyments -
- MongoDB Atlas.

- mongo - mongo is the command-line shell that connects to a specific instance of mongod. When you run mongo with no paramethers it defaults to connecting to the localhost on port 27017.

- mongod - mongod is the primary daemon process for the MongoDB system. It handles data requests, manages data access, and performs



background management operations.

- **mongos** - For a sharded cluster, the mongos instances provide the interfac between the client applications and the sharded cluster. The mongos instance route queries and write operations to the shards. Form the perspective of the application, a mongos instance behave identically to any other MongoDB instance.

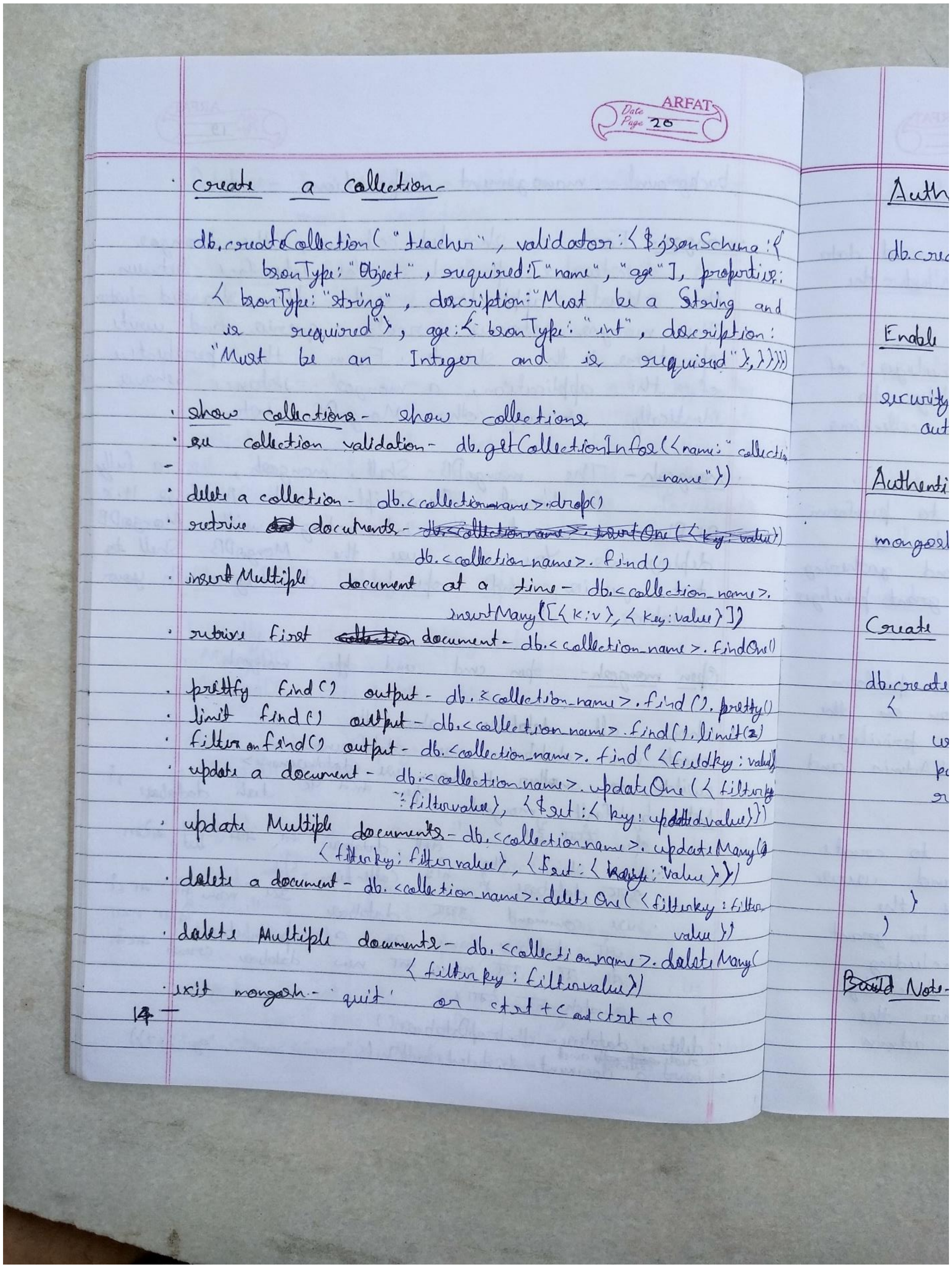
- **mongosh** - The mongoDB Shell, mongosh, is a fully functional JavaScript and Node.js 14.x REPL environment for interacting with MongoDB deployments. You can use the MongoDB Shell to test queries and operations directly with your database.

Open mongosh - open cmd and type 'mongosh'

- show all database - show dbs
- create new database - use <databasename>
- switch to another database - use <databasename>

Note - · bydefault mongosh open करने पर test database में open होता है।
- ~~[illegible]~~ show dbs सभी database का नाम list करेगा अगर database में कोई collection हो।
- use command अगर database उस name का न हो जो database name as a parameter दिया गया तो वह उस name का new database create करके swatch कर देगा।

- delete a database - db.dropDatabase()
- create ~~[illegible]~~ collection and insert a document - db.student.insertOne({"name" : Sonam", "age" : 27})

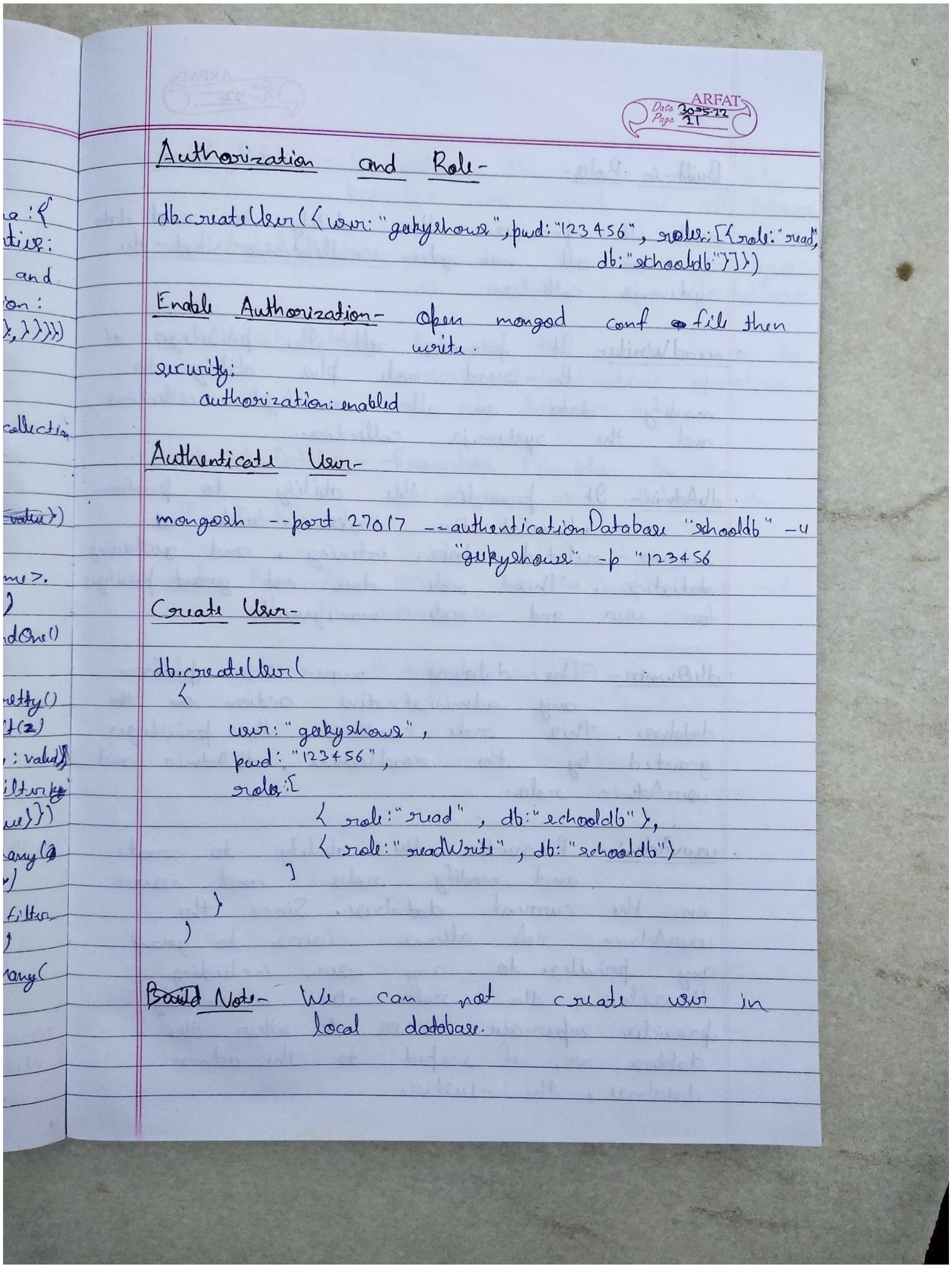


· <u>create a collection</u>-

db.createCollection("teacher", validator: <$jsonSchema: {
 bsonType: "Object", required: ["name", "age"], properties:
 < bsonType: "string", description: "Must be a String and
 is required" >, age: < bsonType: "int", description:
 "Must be an Integer and is required" >, >}}))

· <u>show collections</u>- show collections
· see collection validation- db.getCollectionInfos(<name: "collection
-   name">)
· delete a collection- db.<collectionname>.drop()
· retrive ~~[illegible]~~ documents- ~~db.<collection name>. insertOne(<key: value>)~~
                              db.<collection_name>. find()
· insert Multiple document at a time- db.<collection_name>.
                              insertMany([<k:v>, <key:value>])
· retrive first ~~collection~~ document- db.<collection_name>. findOne()

· prettfy find() output- db.<collection_name>. find(). pretty()
· limit find() output- db.<collection_name>. find(). limit(2)
· filter on find() output- db.<collection_name>. find(<fieldkey: value>)
· update a document- db.<collection name>. updateOne(< filterkey
                 : filtervalue>, <$set: < key: updatedvalue>})
· update Multiple documents- db.<collection name>. updateMany(
          < filterkey: filtervalue>, <$set: < ~~key~~: Value >})
· delete a document- db. <collection_name>. deleteOne(<filterkey : filter
                                                   value >)
· delete Multiple documents- db. <collection_name>. deleteMany(
                    < filter key : filtervalue>)
· exit mongosh- 'quit' or ctrl + c and ctrl + c
14 -



Authorization and Role-

```
db.createUser({user:"geekyshows", pwd:"123456", roles:[{role:"read",
                                    db:"schooldb"}]})
```

Enable Authorization- open mongod conf ~~o~~ file then write.

```
security:
    authorization: enabled
```

Authenticate User-

```
mongosh --port 27017 --authenticationDatabase "schooldb" -u
                        "geekyshows" -p "123456"
```

Create User-

```
db.createUser(
  {
        user: "geekyshows",
        pwd: "123456",
        roles:[
                { role:"read" , db:"schooldb" },
                { role:"readWrite" , db:"schooldb" }
              ]
        }
  )
```

~~Build~~ **Note-** We can not create user in local database.

Built-in Roles-

- read- It provides the ability to read data on all non-system collections and the system.js collection.

- readWrite- It provides all the privileges of the read role plus ability to modify data on all non-system collections and the system.js collection.

- dbAdmin- It provides the ability to perform administrative tasks such as schema-related tasks, indexing, and gathering statistics. This role does not grant privileges for user and role management.

- dbOwner- The database owner can perform any administrative action on the database. This role combines the privileges granted by the readWrite, dbAdmin and userAdmin roles.

- userAdmin- Provides the ability to create and modify roles and users on the current database. Since the userAdmin role allows users to grant any privilege to any user, including themselves, the role also indirectly provides superuser access to either the database or, if scoped to the admin database, the cluster.

- readAnyDatabase - Provides the same read-only privileges as read on all databases.

- readWriteAnyDatabase - Provides the same privileges as readWrite on all databases.

- userAdminAnyDatabase - Provides the same access to user administration operations as userAdmin on all databases.

- dbAdminAnyDatabase - ~~Provides~~ Provides the same privileges as dbAdmin on all databases.

- root - Provides access to the operations and all the resources.

Delete User - db.dropUser("user name")

15—