mongosh

show dbs

use dbName

db.collectionName.find()

Indexing

db.collectionName.createIndex({name:"text"})

Read about indexing: mongodb uses multi key indexes

```
students> db.students.createIndex({name:"text"})
name_text
students> db.students.find({$text:{$search:"aniket"}})
[
  {
```

```
students> db.students.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { _fts: 'text', _ftsx: 1 },
    name: 'name_text',
    weights: { name: 1 },
    default_language: 'english',
    language_override: 'language',
    textIndexVersion: 3
  }
]
students>
```

Aggregation

```
// Aggregation

Aggregation is an operation used to process data and return computed. we can perform count, sum, average, and match
```

Aggregation has 3 ways to perform pipe

```
cars> db.cars.aggregate([{$group:{_id:"$name"}}])
[ { _id: 'dzire' }, { _id: 'abc' }, { _id: 'Alto' } ]
cars>
```

Filtering data

```
cars> db.cars.aggregate([{$match:{fuel:"diesel"}}])
[
  {
    _id: ObjectId("646e26173bca17203d12c9c7"),
    name: 'i10',
    type: 'hatchback',
    price: '900000',
    fuel: 'diesel'
  },
  {
    _id: ObjectId("646e261e3bca17203d12c9c8"),
    name: 'i20',
    type: 'hatchback',
    price: '1200000',
    fuel: 'diesel'
  }
]
```

```
cars> db.cars.aggregate([{$match:{fuel:"diesel"}},{$group:{_id:"$type",totalPrice:{$sum:"$price"}}}])
[ { _id: 'hatchback', totalPrice: 2200000 } ]
```

Aggregation is pending , moving to mongoose

Mongoose is ODM – Object Data Modelling

Helps to create schema , structure of the document

Create a folder

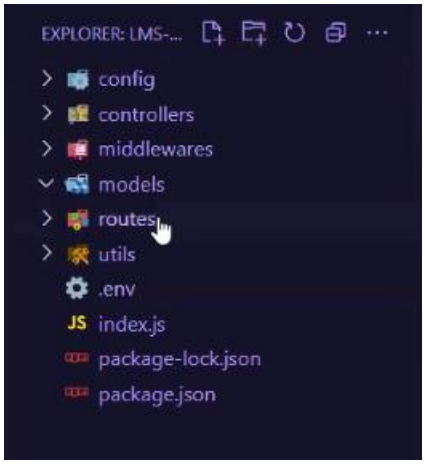
npm init

npm i express mongoose body-parser dotenv

npm i nodemon

```
package.json   JS index.js   Extension: Better Comments   .env
JS index.js > app.get("/") callback
1  import * as dotenv from "dotenv";  3.3k (gzipped: 1.6k)
2
3  import express from "express";
4
5  /**
6   * TODO All Configuration
7   */
8
9  dotenv.config();
10
11 // all Initialization
12
13 const app = express();
14 const PORT = process.env.PORT || 4000;
15
16 app.get("/", (req, res) => {
17   res.status(200).send("Hello from node js application");
18 });
19
20 app.listen(PORT, () => {
21   console.log(`Server is listening at http://localhost:${PORT}/`);
22 });
23
```

Folder structure:



Public folder we use when we have images to store or using template engine

Create .env file

```
.env
1  PORT=5000
2  MONGODB_URL="mongodb://localhost:27017/lmsnew"
```

Inside config folder create a file dbConnect.js

config > JS dbConnect.js > dbConnect > conn

```
import mongoose from "mongoose";  846.5k (gzipped: 228.7k)

export const dbConnect = async () => {
  try {
    const conn = await mongoose.connect(process.env.MONGODB_URL);
    console.log("Database Connect Successfully! ");
  } catch (error) {
    console.log(error);
  }
};

```

Create a Schema for User

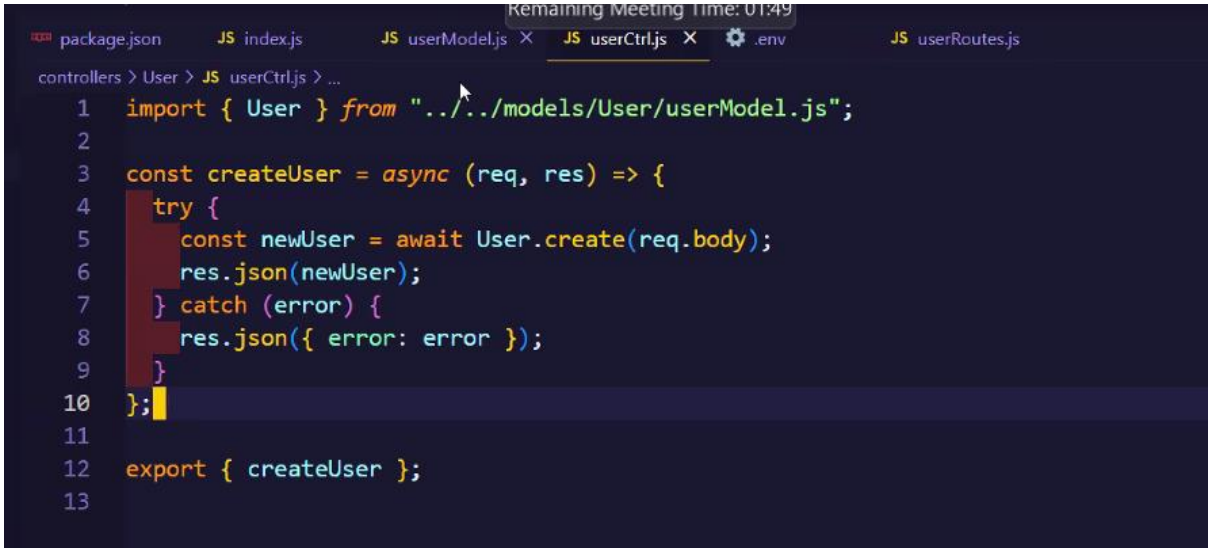
package.json JS index.js JS userModel.js ×

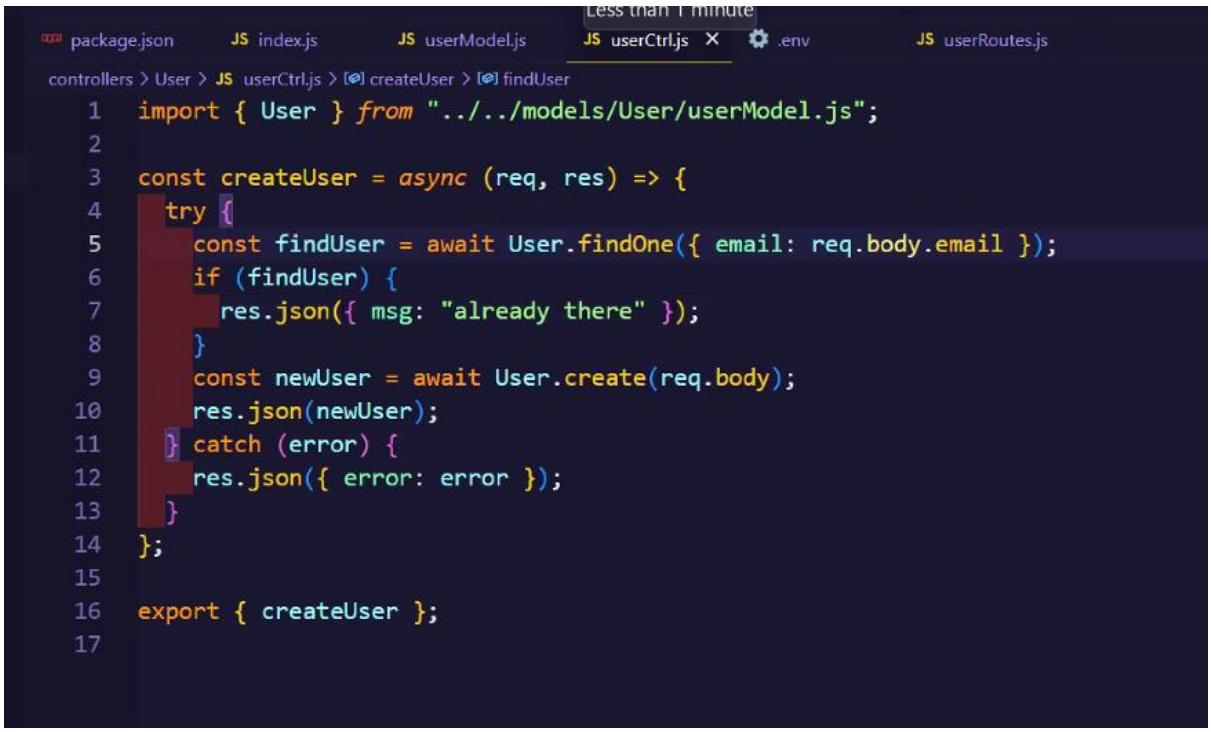
models > User > JS userModel.js > User

```
const userSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: (true, "Name is Required"),
      minlength: 3,
      maxlength: 30,
    },
    email: {
      type: String,
      required: (true, "Email is Required"),
    },
  },
  { timestamps: true }
);

export const User = mongoose.model("User", userSchema);
```

1/3


Remaining Meeting Time: 01:49
package.json
index.js
userModel.js
userCtrl.js
.env
userRoutes.js
controllers > User > userCtrl.js > ...
import { User } from "../../models/User/userModel.js";
const createUser = async (req, res) => {
try {
const newUser = await User.create(req.body);
res.json(newUser);
} catch (error) {
res.json({ error: error });
}
};
export { createUser };


Less than 1 minute
package.json
index.js
userModel.js
userCtrl.js
.env
userRoutes.js
controllers > User > userCtrl.js > createUser > findUser
import { User } from "../../models/User/userModel.js";
const createUser = async (req, res) => {
try {
const findUser = await User.findOne({ email: req.body.email });
if (findUser) {
res.json({ msg: "already there" });
}
const newUser = await User.create(req.body);
res.json(newUser);
} catch (error) {
res.json({ error: error });
}
};
export { createUser };