Notes

CRUD using Mongoose and Express

- Install `express`, `nodemon` and `mongoose` .

```
//index.js

const express=require("express")

const app=express()

app.get("/",(req,res)=>{
  res.send("Welcome")
})

app.listen(4500,()=>{
  console.log("Running the server at 4500")
})
```

- Connect to the `database` now.

```
//db.js

const mongoose=require("mongoose")

const connection=mongoose.connect("mongodb://127.0.0.1:27017/masaidb")

module.exports={
  connection
}
```

- Can we establish a connection without model and schema? ⇒ `YES`
- Our server should always be running so we are not going to disconnect, we are going to keep it connected.

```
//index.js

const express=require("express")
const {connection}=require("./db")

const app=express()
app.use(express.json())
```

```
app.get("/",(req,res)=>{
  res.send("Welcome")
})

app.get("/users",(req,res)=>{
  res.send("users")
})

app.listen(4500, async ()=>{
  try{
  await connection
  console.log("Connected to the db")
  }catch(err){
    console.log("Connection to db failed")
    console.log(err)
  }
  console.log("Running the server at 4500")
})
```

- Where we are going to use it? ⇒ `app.listen()`

```
//lets have a post request to get it from user and post it in the database

app.post("/createuser",(req,res)=>{
  const data=req.body
  console.log(data)
  res.send("user has been created")
})
```

- Now we need a `schema` and `model` to capture the data from the user.

```
//db.json
//schema
const userSchema=mongoose.Schema({
  name:String,
  age:Number,
  legal:Boolean,
  city:String
  language:String
})

//model
const UserModel=mongoose.model("user",userSchema)

module.exports={
  UserModel
}
```

- Now lets `add` some data to the database.

```
//index.js
//after importing the userSchema and UserModel
app.post("/createuser",async (req,res)=>{
  const data=req.body
  const user=new UserModel(data)
  await user.save()
  res.send("user has been created")
})
```

- Now how to `get` all the data

```
app.get("/users",async (req,res)=>{
  const users=await UserModel.find()
  res.send(users)
})
```

Now we have successfully connected our database with express app.

- Use `try` `catch` to capture the error, in all the routes.
- This is a good practice actually.

Query Handling

- If you want the data of users as per the city, you can simple pass it as a `query` and then handle the query at the `backend`.
- Because we are never going to hardcode anything in backend.

```
app.get("/users",async (req,res)=>{
  const query=req.query
  const users=await UserModel.find(query)
  res.send(users)
})
```

💡 **Note:** Now you must be having a basic idea how your front end and backend work together.

Now lets `Modify` Something

```
app.patch("/editusers/:userID",async (req,res)=>{
  const userID=req.params.userID
  const payload=req.body
  try{
    const query=await UserModel.findByIdAndUpdate({_id:userID},payload)
  }catch(err){
    console.log(err)
    res.send({"err":"something went wrong"})
  }
})
```

Now lets `Delete` Something

```
app.delete("/removeuser/:userID",async (req,res)=>{
  const userID=req.params.userID
  try{
    await UserModel.findByIdAndDelete({_id:userID})
    res.send(`User with user id ${userID} has been deleted from the database`)
  }catch(err){
    console.log(err)
    res.send({"err":"something went wrong"})
  }
})
```

💡 **Homework:-** Research about status codes and how you can add them here.