


## Tutorial Week 7

### Nguyen Dinh Hoang Son - GCS220616

```
todoListApi > models > todoListModel.js > ...
You 21 minutes ago | 1 author (You)
1 const mongoose = require('mongoose');
2 const Schema = mongoose.Schema;
3
4 const TaskSchema = new Schema({
5   name: {
6     type: String,
7     required: 'Kindly enter the name of the task'
8   },
9   Created_date: {
10    type: Date,
11    default: Date.now
12  },
13  status: {
14    type: [{
15      type: String,
16      enum: ['pending', 'ongoing', 'completed']
17    }],
18    default: ['pending']
19  }
20 });
21
22 module.exports = mongoose.model('Tasks', TaskSchema);
23
```



```
18   default: ['pending']
19 }
20 });
21
22 module.exports = mongoose.model('Tasks', TaskSchema);
23
```


PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SQL CONSOLE

```
> todolistapi@1.0.0 start
> nodeemon server.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
todo list RESTful API server started on: 3000
```

PS C:\Users\jimmy\Greenwich> git branch -M main # Rename the branch to 'main'

```
>> git push -u origin main
>>
Enter passphrase for key '/c/Users/jimmy/.ssh/id_ed25519':
Enumerating objects: 58762, done.
Counting objects: 100% (58762/58762), done.
Delta compression using up to 16 threads
Compressing objects: 100% (33941/33941), done.
Writing objects: 98% (58030/58762), 1.81 GiB | 2.20 MiB/s
```



package.json todoListController.js todoListModel.js todoListRoutes.js Extension Postcode Postcode X Postcode server.js main.py

GET http://localhost:3000/tasks Send

Params Authorization Headers (4) Body Options Code

none form-data x-www-form-urlencoded raw binary GraphQL

| KEY  | VALUE      | DESCRIPTION |
|------|------------|-------------|
| name | Coursework | Description |
| Key  | Value      | Description |

Body Headers JSON Status: 200 OK Duration: 78 ms

```
1 {
2   "_id": "671a0da98e07cf7a3234fd56",
3   "name": "Coursework",
4   "status": [
5     "pending"
6   ],
7   "Created_date": "2024-10-24T09:04:41.985Z",
8   "__v": 0
9 }
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SQL CONSOLE

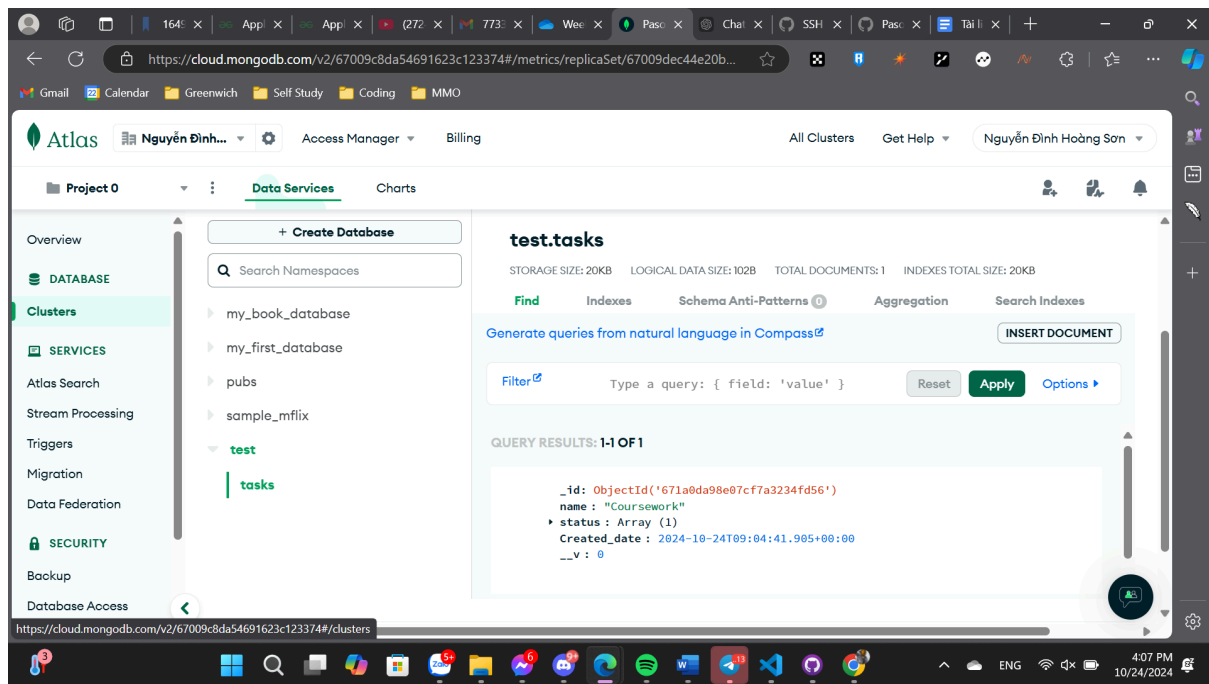
```
> todolistapi@1.0.0 start
> nodeemon server.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
todo list RESTful API server started on: 3000
```

PS C:\Users\jimmy\Greenwich> git branch -M main # Rename the branch to 'main'

```
>> git push -u origin main
>>
Enter passphrase for key '/c/Users/jimmy/.ssh/id_ed25519':
Enumerating objects: 58762, done.
Counting objects: 100% (58762/58762), done.
Delta compression using up to 16 threads
Compressing objects: 100% (33941/33941), done.
Writing objects: 98% (58030/58762), 1.90 GiB | 2.02 MiB/s
```

0 Connect Go Live Background Colorize: 0 variables Colorize



For this tutorial, I worked on building a RESTful API, following the instructions from the lecture slides starting from slide 23. The primary focus was to create and connect an API to a MongoDB database using Node.js and Express.js.

First, I started by defining the model in Mongoose for my database. This involved creating a schema that would describe the structure of the documents stored in MongoDB. I had to make sure I set up all necessary fields and added validation rules.

Next, I set up an Express server and made sure it was running on port 3000 using nodemon. Running the server with nodemon made it easier to track changes and restart automatically whenever I updated my code. I faced a bit of a challenge here due to a port conflict, but I resolved it by either freeing up the port or changing it to a different one.

After setting up the server, I used Postman to test the API endpoints and confirm that the connection to MongoDB worked as intended. I verified that the API could successfully retrieve, create, update, and delete records.

Finally, I accessed my MongoDB instance to ensure data was being stored properly. Overall, this tutorial helped me better understand how to implement a REST API and troubleshoot common server and database issues, improving my familiarity with backend development tools.