

# Week 3 Assignment

---

## [GITHUB LINK](#)

### 1. Explain the Entity relationship with an example

Entity relationships define how different entities (tables) in a database are related to each other. In a Library Management System, for example:

- A 'Book' can be borrowed multiple times, so one Book has a One-to-Many relationship with 'BorrowRecord'.
- A 'Member' can borrow many books, so one Member has a One-to-Many relationship with 'BorrowRecord'.

These relationships help organize data and ensure integrity between related records.

### 2. Explain the Dependency Injection mechanism in NestJS.

Dependency Injection (DI) in NestJS is a design pattern where classes receive their dependencies from external sources rather than creating them.

This is done using the `@Injectable()` decorator and constructor injection. For example, a service can be injected into a controller:

```
class BookService {}  
class BookController {  
  constructor(private bookService: BookService) {}  
}
```

This makes code easier to manage and test since dependencies are controlled by the framework.

### 3. How does NestJS DI resolve circular dependencies between services?

NestJS uses the `forwardRef()` function to resolve circular dependencies. This tells NestJS to resolve the reference later.

For example, if Service A uses Service B, and Service B also uses Service A:

```
constructor(@Inject(forwardRef(() => ServiceB)) private serviceB: ServiceB) {}
```

This approach prevents infinite loops during injection and allows mutual access between services safely.

### 4. What is the purpose of a Module in NestJS?

A Module in NestJS is used to organize related components (controllers, services, etc.) into a single unit. It helps group functionalities, making the app scalable and easier to maintain. Each module is defined using the `@Module()` decorator.

For example, a 'LibraryModule' could include controllers and services related to books and members.

### 5. Why would you use a Custom Validation Pipe instead of class-validator decorators alone?

Class-validator decorators are great for simple validation like checking email or string length.

However, Custom Validation Pipes are used when you need more advanced logic, like checking if a book's quantity is greater than 0 before borrowing.

Custom pipes can use services and throw custom exceptions, making them powerful for complex validations.

## Project 1: Library Management System –

POST /books:

The screenshot shows a REST client interface with a POST request to `http://localhost:3000/library-mgt/books`. The request body is a JSON object representing a book. The response status is 201 Created, with a size of 110 Bytes and a time of 27 ms. The response body is a JSON object with the same data as the request body.

```
POST http://localhost:3000/library-mgt/books Send
```

Query Headers<sup>3</sup> Auth **Body<sup>1</sup>** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "name": "Pride and Prejudice",
3   "isbn": 978014143,
4   "quantity": 10,
5   "available": 7,
6   "author_id": 2,
7   "price": 9.99
8 }
```

Status: 201 Created Size: 110 Bytes Time: 27 ms

Response Headers<sup>6</sup> Cookies Results Docs

```
1 {
2   "id": 66,
3   "name": "Pride and Prejudice",
4   "isbn": 978014143,
5   "quantity": 10,
6   "available": 7,
7   "author_id": 2,
8   "price": 9.99
9 }
```

GET /books/available # List available books :

The screenshot shows a REST client interface with a GET request to `http://localhost:3000/library-mgt/books/available`. The response status is 200 OK, with a size of 4.76 KB and a time of 5 ms. The response body is a JSON array of three book objects.

```
GET http://localhost:3000/library-mgt/books/available Send
```

Query Headers<sup>3</sup> Auth **Body<sup>1</sup>** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

Status: 200 OK Size: 4.76 KB Time: 5 ms

Response Headers<sup>6</sup> Cookies Results Docs

```
1 [
2   {
3     "id": 1,
4     "name": "The Catcher in the Rye",
5     "isbn": 1234567,
6     "quantity": 7,
7     "available": 2,
8     "price": 671
9   },
10  {
11    "id": 2,
12    "name": "To Kill a Mockingbird",
13    "isbn": 9876543,
14    "quantity": 15,
15    "available": 3,
16    "price": 139
17  },
18  {
19    "id": 4,
20    "name": "1984",
21    "isbn": 5566778,
22    "quantity": 8,
23    "available": 6,
24    "price": 9
25  },
26 ]
```

GET /books/borrowed # List borrowed books :

GET  Send

Query Headers <sup>3</sup> Auth **Body** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

Status: 200 OK Size: 562 Bytes Time: 5 ms

Response Headers <sup>6</sup> Cookies Results Docs

```
1 [
2   {
3     "id": 3,
4     "name": "Pride and Prejudice",
5     "isbn": 1122334,
6     "quantity": 10,
7     "available": 0,
8     "price": 686
9   },
10  {
11    "id": 10,
12    "name": "Jane Eyre",
13    "isbn": 11223,
14    "quantity": 7,
15    "available": 0,
16    "price": 595
17  },
18  {
19    "id": 21,
20    "name": "The Girl with the Dragon Tattoo",
21    "isbn": 1110009,
22    "quantity": 19,
23    "available": 0,
24    "price": 243
25  },
26 ]
```

Response

POST /members # Register member

POST  Send

Query Headers <sup>3</sup> Auth **Body** <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "name": "Adam Shaikh",
3   "email": "Adam876@example.com",
4   "phone": "9876543210",
5   "address": "12 MG Road, ratnagiri",
6   "isActive": true,
7   "dateOfBirth": "1992-04-10"
8 }
```

Status: 201 Created Size: 141 Bytes Time: 14 ms

Response Headers <sup>6</sup> Cookies Results Docs

```
1 {
2   "id": 9,
3   "name": "Adam Shaikh",
4   "email": "Adam876@example.com",
5   "phone": "9876543210",
6   "address": "12 MG Road, ratnagiri",
7   "dateOfBirth": "1992-04-10"
8 }
```

## POST /borrow # Borrow a book (validate stock)

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:3000/library-mgt/borrow/1
- Body:** JSON Content: 

```
1
```
- Status:** 201 Created
- Size:** 94 Bytes
- Time:** 17 ms
- Response:**

```
1 {
2   "id": 1,
3   "name": "The Catcher in the Rye",
4   "isbn": 1234567,
5   "quantity": 7,
6   "available": 1,
7   "price": 671
8 }
```

## POST /return/:id # Return a book : note that the available value is increased when book is returned.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:3000/library-mgt/return/1
- Body:** JSON Content: 

```
1
```
- Status:** 201 Created
- Size:** 94 Bytes
- Time:** 15 ms
- Response:**

```
1 {
2   "id": 1,
3   "name": "The Catcher in the Rye",
4   "isbn": 1234567,
5   "quantity": 7,
6   "available": 2,
7   "price": 671
8 }
```

Custom pipe to check quantity  $\geq 0$  & Exception filter for "Book not available"

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:3000/library-mgt/borrow/1
- Body:** JSON Content: 

```
1
```
- Status:** 404 Not Found
- Size:** 69 Bytes
- Time:** 17 ms
- Response:**

```
1 {
2   "message": "Book not available",
3   "error": "Not Found",
4   "statusCode": 404
5 }
```

## Class-validator for email/phone in Member

POST

http://localhost:3000/library-mgt/members

Send

Query

Headers<sup>3</sup>

Auth

Body<sup>1</sup>

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "name": "Adam Shaikh",
3   "email": "Adam876@@example.com",
4   "phone": "987654343653475347210",
5   "address": "12 MG Road, ratnagiri",
6   "isActive": true,
7   "dateOfBirth": "1992-04-10"
8 }
```

Status: 400 Bad Request

Size: 109 Bytes

Time: 4 ms

Response

Headers<sup>6</sup>

Cookies

Results

Docs

```
1 {
2   "message": [
3     "Invalid email address",
4     "Phone number must be 10 digits"
5   ],
6   "error": "Bad Request",
7   "statusCode": 400
8 }
```

## Overdue books report (GET /reports/overdue)

GET

http://localhost:3000/library-mgt/reports/overdue

Send

Query

Headers<sup>3</sup>

Auth

Body

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1
```

Status: 200 OK

Size: 16.24 KB

Time: 12 ms

Response

Headers<sup>6</sup>

Cookies

Results

Docs

```
1 [
2   {
3     "id": 51,
4     "book": {
5       "id": 1,
6       "name": "The Catcher in the Rye",
7       "isbn": 1234567,
8       "quantity": 7,
9       "available": 0,
10      "price": 671
11     },
12     "member": {
13       "id": 1,
14       "name": "Amit Sharma",
15       "email": "amit.sharma@example.com",
16       "phone": "9876543210",
17       "address": "12 MG Road, Bengaluru",
18       "dateOfBirth": null
19     },
20     "borrowDate": "2024-01-15",
21     "dueDate": "2024-01-29T11:30:00.000Z",
22     "returnDate": null,
23     "notes": "Member requested a popular title.",
24     "status": "borrowed"
25   }
26 ]
```

Response

## Project 2: Task Manager with RBAC

Static Role-Based Access :

DELETE

http://localhost:3000/task-manager/remove-task/1

Send

Query

Headers 3

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

1

{ "id": 2, "username": "user", "password": "user123", "role": "USER"

}

2

3

Status: 403 Forbidden

Size: 78 Bytes

Time: 4 ms

Response

Headers 6

Cookies

Results

Docs

1

{

2

"message": "Access denied. Admins only.",

3

"error": "Forbidden",

4

"statusCode": 403

5

}

Create Task :

POST

http://localhost:3000/task-manager/create-task

Send

Query

Headers 3

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

1

{

2

"title": "Refactor user authentication service",

3

"description": "Improve the security and efficiency of the user

4

authentication module.",

5

"status": "IN\_PROGRESS",

6

"createdBy": 5,

7

"dueDate": "2025-05-05T10:00:00.000Z",

8

"priority": "HIGH"

9

}

Status: 201 Created

Size: 330 Bytes

Time: 48 ms

Response

Headers 6

Cookies

Results

Docs

1

{

2

"id": 21,

3

"title": "Refactor user authentication service",

4

"description": "Improve the security and efficiency of the user

5

authentication module.",

6

"status": "IN\_PROGRESS",

7

"createdBy": 5,

8

"createdAt": "2025-04-25T15:51:29.275Z",

9

"updatedAt": "2025-04-25T15:51:29.275Z",

10

"deletedAt": null,

11

"dueDate": "2025-05-05T10:00:00.000Z",

12

"priority": "HIGH"

13

}

editable only by current user:

PATCH http://localhost:3000/task-manager/2

Send

Query

Headers3

AuthBody1TestsPre Run

HTTP Headers

Raw

☒

Accept

\*/\*

☒

User-Agent

Thunder Client (https://www.thunderclient.com)

☒

authorization

Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e

☐

header

value

Status: 200 OKSize: 329 BytesTime: 16 ms

Response

Headers6

CookiesResultsDocs

{}

1 {

2 "id": 2,

3 "title": "Refactor user authentication service",

4 "description": "Improve the security and efficiency of the user authentication module.",

5 "status": "IN\_PROGRESS",

6 "createdBy": 6,

7 "createdAt": "2025-04-25T14:51:00.000Z",

8 "updatedAt": "2025-04-25T16:17:47.000Z",

9 "deletedAt": null,

10 "dueDate": "2025-05-05T10:00:00.000Z",

11 "priority": "HIGH"

12 }

PATCH http://localhost:3000/task-manager/2

Send

Query

Headers3

AuthBody1TestsPre Run

HTTP Headers

Raw

☒

Accept

\*/\*

☒

User-Agent

Thunder Client (https://www.thunderclient.com)

☒

authorization

Bearer

☐

header

value

Status: 401 UnauthorizedSize: 93 BytesTime: 3 ms

Response

Headers6

CookiesResultsDocs

1 {

2 "message": "Missing or invalid authorization header",

3 "error": "Unauthorized",

4 "statusCode": 401

5 }

PATCH http://localhost:3000/task-manager/2

Send

Query

Headers3

AuthBody1TestsPre Run

HTTP Headers

Raw

☒

Accept

\*/\*

☒

User-Agent

Thunder Client (https://www.thunderclient.com)

☒

authorization

Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ

☐

header

value

Status: 401 UnauthorizedSize: 78 BytesTime: 4 ms

Response

Headers6

CookiesResultsDocs

1 {

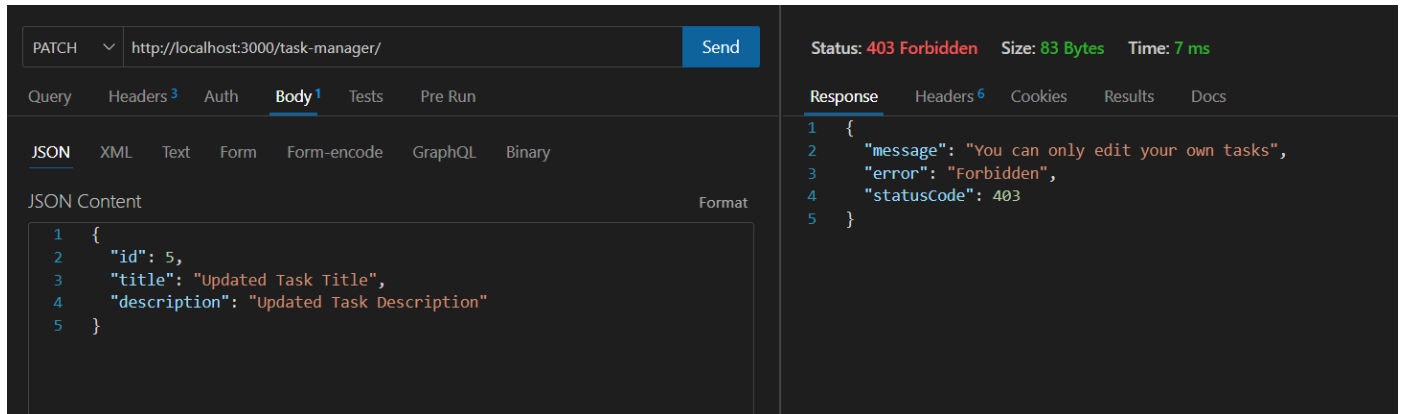
2 "message": "Invalid or expired token",

3 "error": "Unauthorized",

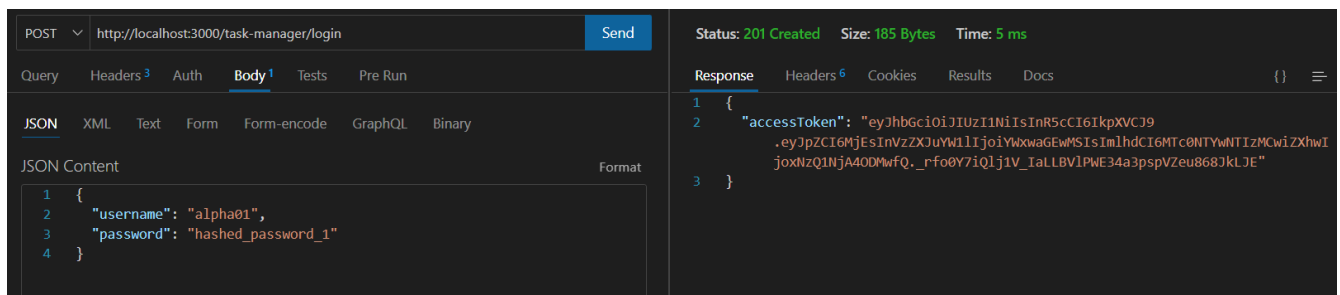
4 "statusCode": 401

5 }

You can only edit your own tasks:



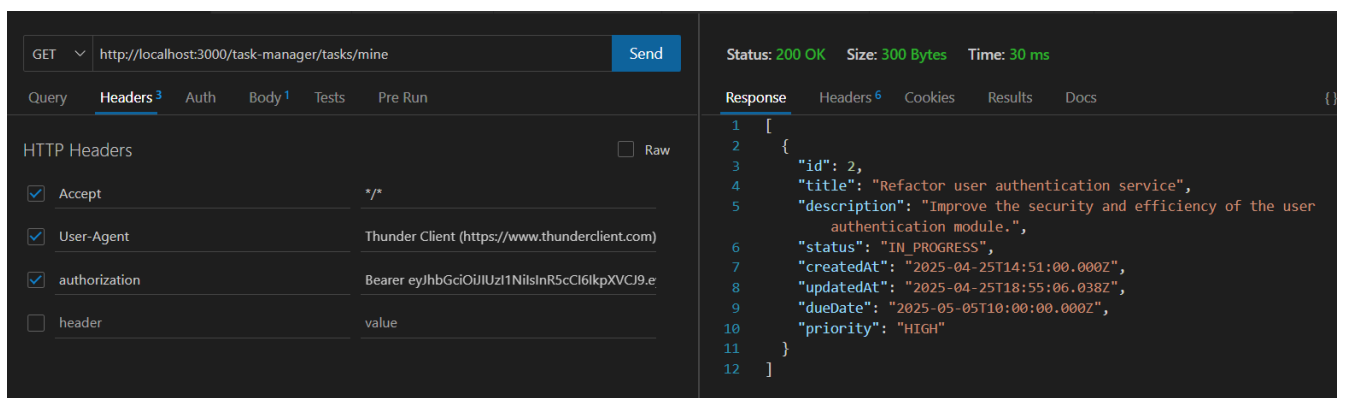
Login:



- GET /tasks/mine (Returns tasks by current user)

for user with userId:43

extracted from jwt token





## Status Validation Pipe:

PATCH

http://localhost:3000/task-manager/update-status

Send

Query

Headers 3

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "id": 2,
3   "status": "DONE"
4 }
```

Status: 200 OK Size: 323 Bytes Time: 43 ms

Response

Headers 6

Cookies

Results

Docs

```
1 {
2   "id": 2,
3   "title": "Refactor user authentication service",
4   "description": "Improve the security and efficiency of the user
5     authentication module.",
6   "status": "DONE",
7   "createdBy": 43,
8   "createdAt": "2025-04-25T14:51:00.000Z",
9   "updatedAt": "2025-04-25T19:29:38.000Z",
10  "deletedAt": null,
11  "dueDate": "2025-05-05T10:00:00.000Z",
12  "priority": "HIGH"
13 }
```

## Request logger (log method, URL, timestamp)

```
[Nest] 17880 - 26/04/2025, 1:01:53 am LOG [NestApplication] Nest application successfully started +5ms
[2025-04-25T19:32:02.489Z] PATCH /task-manager/update-status
```

## Task Search:

GET

http://localhost:3000/task-manager/task-search?search=refactor

Send

Query

Headers 3

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 |
```

Status: 200 OK Size: 1.27 KB Time: 27 ms

Response

Headers 6

Cookies

Results

Docs

```
1 [
2   {
3     "id": 2,
4     "title": "Refactor user authentication service",
5     "description": "Improve the security and efficiency of the user
6       authentication module.",
7     "status": "DONE",
8     "createdBy": 43,
9     "createdAt": "2025-04-25T14:51:00.000Z",
10    "updatedAt": "2025-04-25T19:29:38.000Z",
11    "deletedAt": null,
12    "dueDate": "2025-05-05T10:00:00.000Z",
13    "priority": "HIGH"
14  },
15  {
16    "id": 5,
17    "title": "Refactor existing codebase",
18    "description": "Improve the performance of the data processing module
19      .",
20    "status": "IN_PROGRESS",
21    "createdBy": 24,
22    "createdAt": "2025-04-25T14:54:00.000Z",
23    "updatedAt": "2025-04-25T18:55:06.038Z",
24    "deletedAt": null,
25    "dueDate": "2025-05-10T12:30:00.000Z",
26    "priority": "MEDIUM"
27  },
28  {
29    "id": 21,
30    "title": "Refactor user authentication service",
31    "description": "Improve the security and efficiency of the user
32      authentication module.",
33    "status": "IN_PROGRESS",
34    "createdBy": 43,
35    "createdAt": "2025-04-25T14:51:00.000Z",
36    "updatedAt": "2025-04-25T19:29:38.000Z",
37    "deletedAt": null,
38    "dueDate": "2025-05-05T10:00:00.000Z",
39    "priority": "HIGH"
40  }
41 ]
```