

CSE343- Web Development

Labs 8, 9

Loaiy Mahmoud Fatouh

23P0419

Lab 8

1- Adding 2 posts

The screenshot shows two separate instances of the Postman application interface, each displaying a POST request to the '/posts' endpoint of a local server.

Top Instance (Postman 1):

- URL: `http://localhost:3000/posts`
- Method: `POST`
- Body tab selected, showing JSON input:

```
1 {  
2   "title": "Title 1",  
3   "content": "content of title 1"  
4 }
```
- Response status: `201 Created`
- Response body:

```
1 {  
2   "id": 2,  
3   "title": "Title 1",  
4   "content": "content of title 1",  
5   "comments": []  
6 }
```

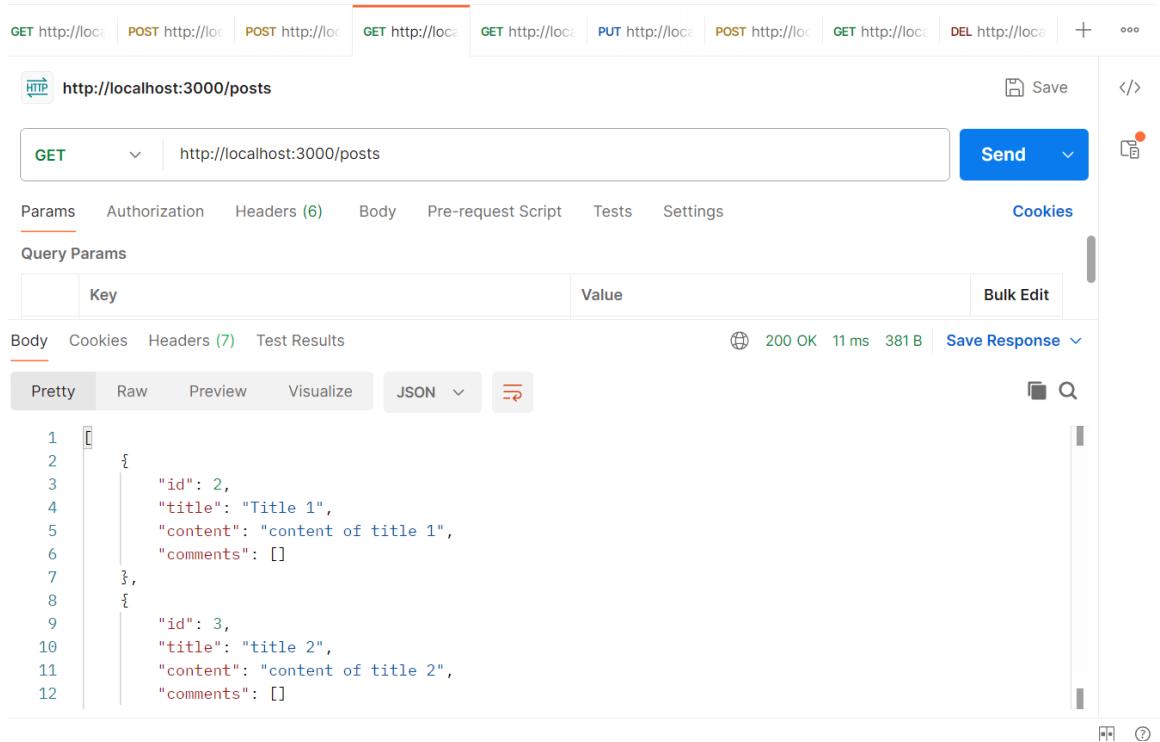
Bottom Instance (Postman 2):

- URL: `http://localhost:3000/posts`
- Method: `POST`
- Body tab selected, showing JSON input:

```
1 {  
2   "title": "title 2",  
3   "content": "content of title 2"  
4 }
```
- Response status: `201 Created`
- Response body:

```
1 {  
2   "id": 3,  
3   "title": "title 2",  
4   "content": "content of title 2",  
5   "comments": []  
6 }
```

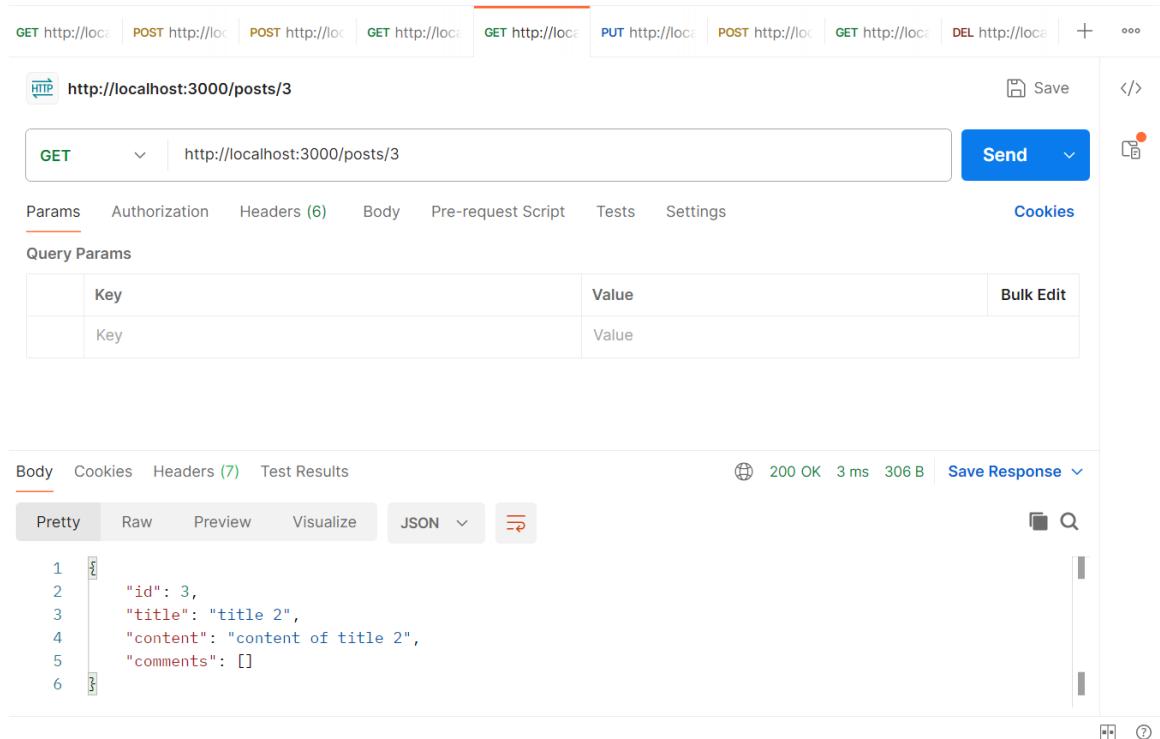
2- Getting all posts



The screenshot shows the Postman interface with a GET request to `http://localhost:3000/posts`. The response body is a JSON array containing two posts:

```
1
2 {
3     "id": 2,
4     "title": "Title 1",
5     "content": "content of title 1",
6     "comments": []
7 },
8 {
9     "id": 3,
10    "title": "title 2",
11    "content": "content of title 2",
12    "comments": []
}
```

3- Getting post by ID



The screenshot shows the Postman interface with a GET request to `http://localhost:3000/posts/3`. The response body is a single post with `id: 3`:

```
1 {
2     "id": 3,
3     "title": "title 2",
4     "content": "content of title 2",
5     "comments": []
6 }
```

4- Updating a post

The screenshot shows the Postman interface with a PUT request to `http://localhost:3000/posts/3`. The request body contains the JSON object `{"content": "content of title 2 updated"}`. The response status is 200 OK with a 5 ms duration and 314 B size. The response body is displayed in Pretty JSON format:

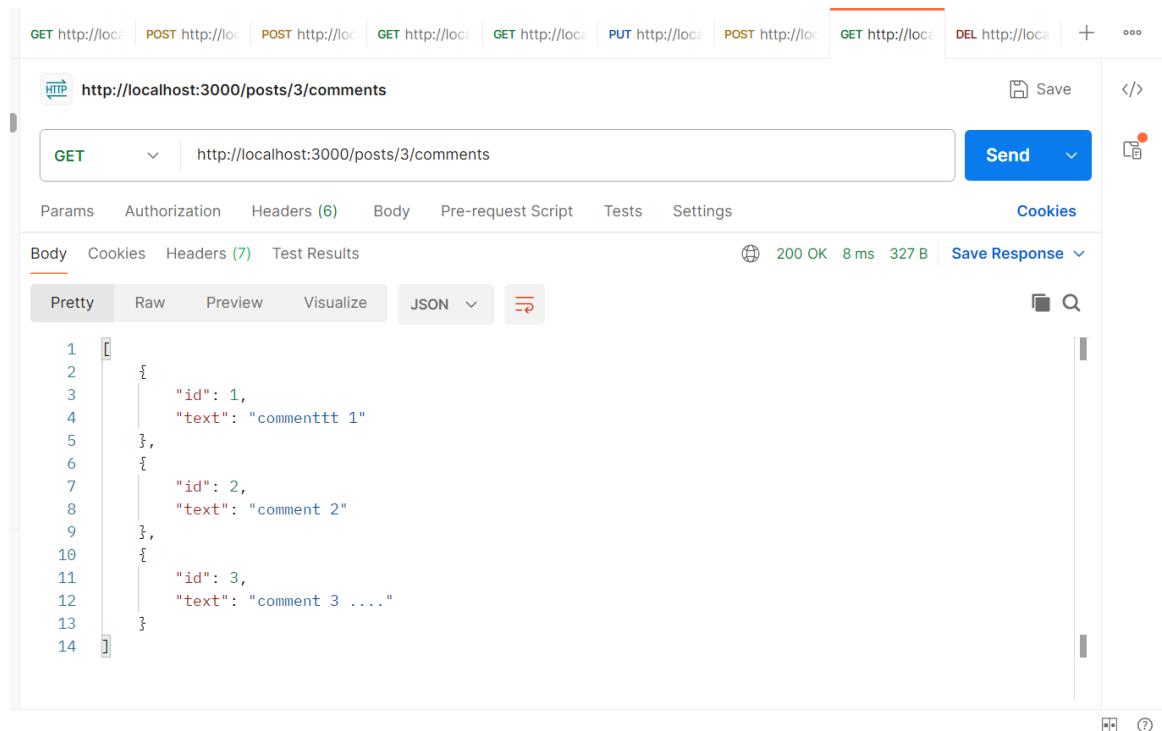
```
1 {
2   "id": 3,
3   "title": "title 2",
4   "content": "content of title 2 updated",
5   "comments": []
6 }
```

5- adding 3 comments

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/posts/3/comments`. The request body contains the JSON object `{"text": "comment 3"}`. The response status is 201 Created with a 4 ms duration and 272 B size. The response body is displayed in Pretty JSON format:

```
1 {
2   "id": 3,
3   "text": "comment 3 ...."
4 }
```

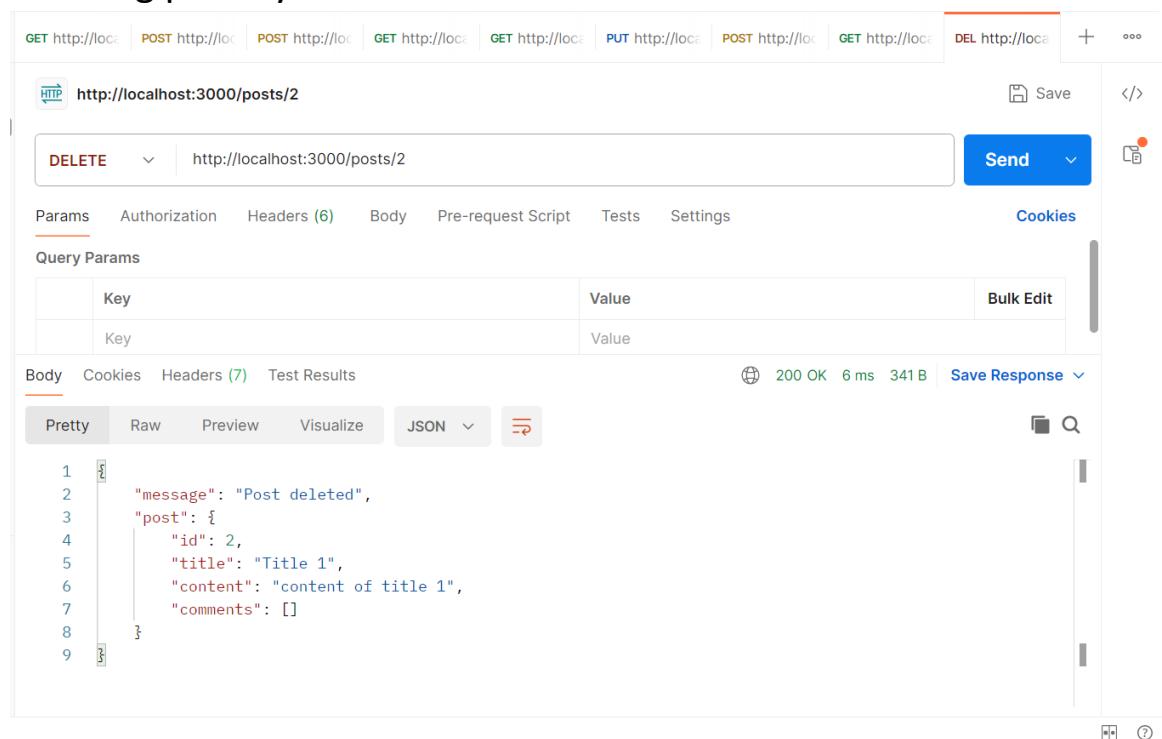
6- getting all comments



The screenshot shows the Postman interface with a GET request to `http://localhost:3000/posts/3/comments`. The response status is 200 OK with a time of 8 ms and a size of 327 B. The response body is a JSON array containing three comments:

```
1 [ { "id": 1, "text": "commenttt 1" }, { "id": 2, "text": "comment 2" }, { "id": 3, "text": "comment 3 ...." } ]
```

7- Deleting post by id



The screenshot shows the Postman interface with a DELETE request to `http://localhost:3000/posts/2`. The response status is 200 OK with a time of 6 ms and a size of 341 B. The response body is a JSON object:

```
1 { "message": "Post deleted", "post": { "id": 2, "title": "Title 1", "content": "content of title 1", "comments": [] } }
```

Lab 9

1- Posting 2 courses

The screenshot shows two separate POST requests in the Postman interface, each creating a new course in a MongoDB database.

Request 1:

- Method: POST
- URL: `http://localhost:5000/api/courses`
- Body (JSON):

```
1 "title": "Course Title 1",
2 "description": "Learning course 1",
3 "instructor": "jana tamer",
4 "price": 1200,
5 "category": "Web Development"
```

Response 1:

- Status: 201 Created
- Time: 120 ms
- Size: 510 B

Request 2:

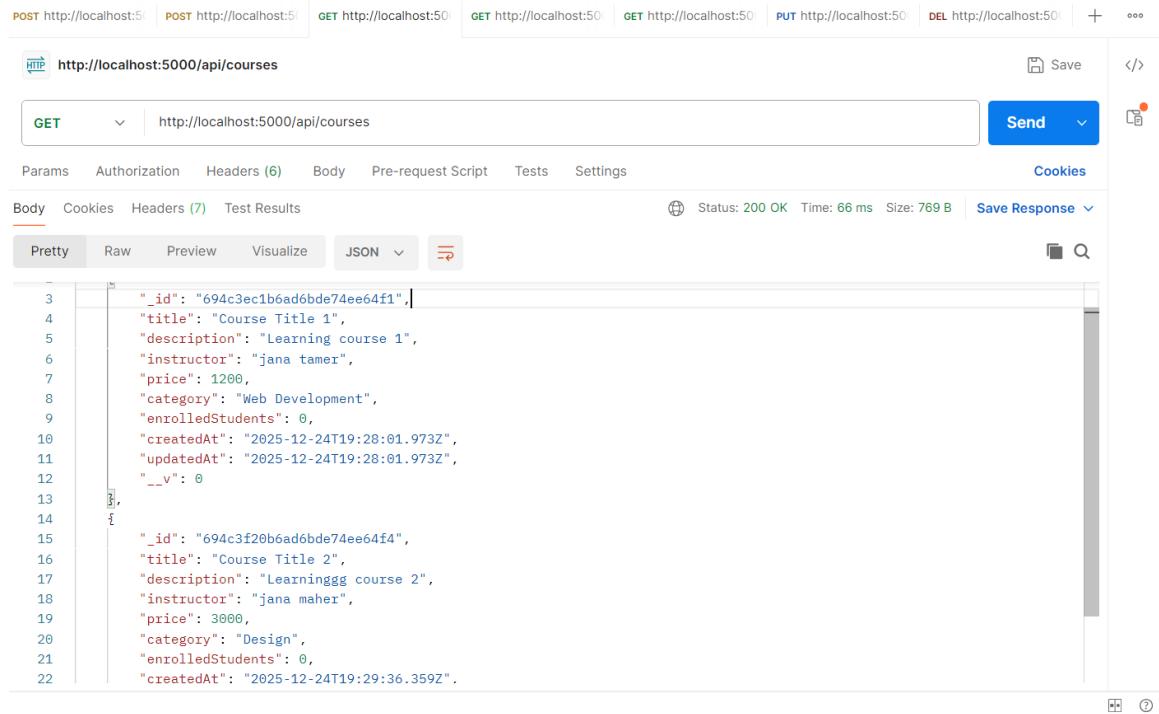
- Method: POST
- URL: `http://localhost:5000/api/courses`
- Body (JSON):

```
1 "instructor": "jana maher",
2 "price": 3000,
3 "category": "Design"
```

Response 2:

- Status: 201 Created
- Time: 141 ms
- Size: 503 B

2-Get all courses



HTTP <http://localhost:5000/api/courses>

GET <http://localhost:5000/api/courses>

Send

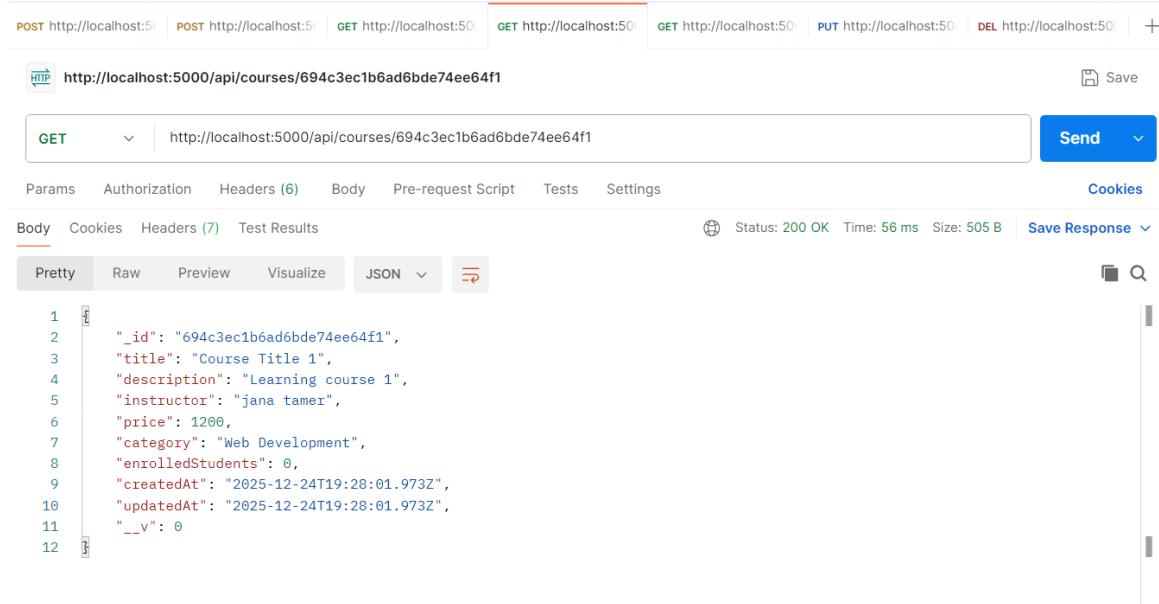
Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Status: 200 OK Time: 66 ms Size: 769 B Save Response

Pretty Raw Preview Visualize JSON

```
3   "_id": "694c3ec1b6ad6bde74ee64f1",
4     "title": "Course Title 1",
5     "description": "Learning course 1",
6     "instructor": "jana tamer",
7     "price": 1200,
8     "category": "Web Development",
9     "enrolledStudents": 0,
10    "createdAt": "2025-12-24T19:28:01.973Z",
11    "updatedAt": "2025-12-24T19:28:01.973Z",
12    "__v": 0
13  },
14  {
15    "_id": "694c3f20b6ad6bde74ee64f4",
16    "title": "Course Title 2",
17    "description": "Learninggg course 2",
18    "instructor": "jana maher",
19    "price": 3000,
20    "category": "Design",
21    "enrolledStudents": 0,
22    "createdAt": "2025-12-24T19:29:36.359Z".
```

3- Get course by ID



HTTP <http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1>

GET <http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1>

Send

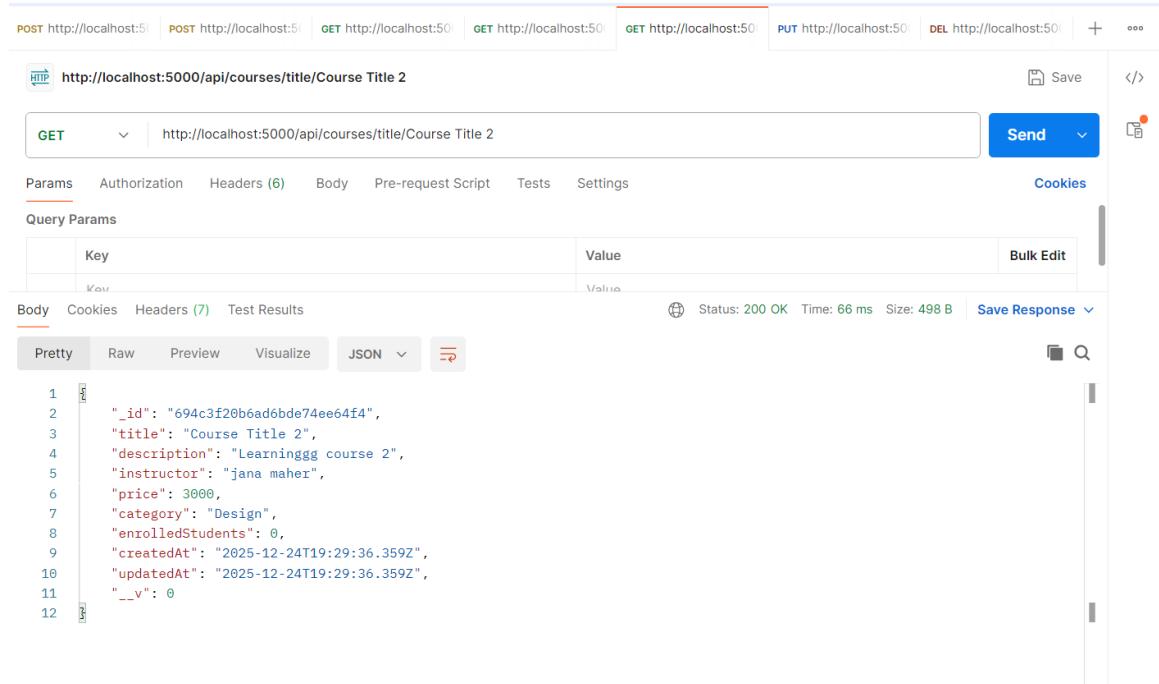
Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Status: 200 OK Time: 56 ms Size: 505 B Save Response

Pretty Raw Preview Visualize JSON

```
1   "_id": "694c3ec1b6ad6bde74ee64f1",
2     "title": "Course Title 1",
3     "description": "Learning course 1",
4     "instructor": "jana tamer",
5     "price": 1200,
6     "category": "Web Development",
7     "enrolledStudents": 0,
8     "createdAt": "2025-12-24T19:28:01.973Z",
9     "updatedAt": "2025-12-24T19:28:01.973Z",
10    "__v": 0
```

4- Get course by title



HTTP <http://localhost:5000/api/courses/title/Course Title 2>

GET <http://localhost:5000/api/courses/title/Course Title 2>

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value
Course Title	Course Title 2

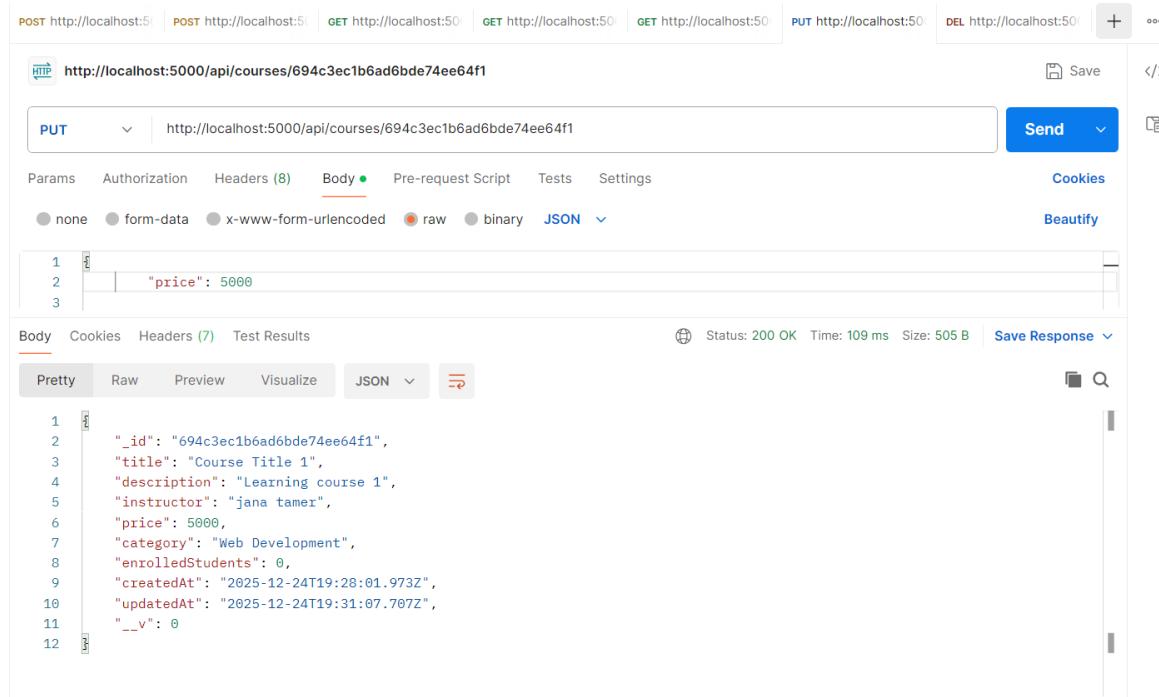
Body Cookies Headers (7) Test Results

Status: 200 OK Time: 66 ms Size: 498 B Save Response

Pretty Raw Preview Visualize JSON

```
1 _id: "694c3f20b6ad6bde74ee64f4",
2   "title": "Course Title 2",
3   "description": "Learninggg course 2",
4   "instructor": "jana maher",
5   "price": 3000,
6   "category": "Design",
7   "enrolledStudents": 0,
8   "createdAt": "2025-12-24T19:29:36.359Z",
9   "updatedAt": "2025-12-24T19:29:36.359Z",
10  "__v": 0
```

5-Update course by ID



HTTP <http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1>

PUT <http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1>

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body

none form-data x-www-form-urlencoded raw binary JSON Beautify

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 109 ms Size: 505 B Save Response

Pretty Raw Preview Visualize JSON

```
1   "price": 5000
```

```
1 _id: "694c3ec1b6ad6bde74ee64f1",
2   "title": "Course Title 1",
3   "description": "Learning course 1",
4   "instructor": "jana tamer",
5   "price": 5000,
6   "category": "Web Development",
7   "enrolledStudents": 0,
8   "createdAt": "2025-12-24T19:28:01.973Z",
9   "updatedAt": "2025-12-24T19:31:07.707Z",
10  "__v": 0
```

6- Delete course by ID

The screenshot shows the Postman interface for a DELETE request to the endpoint `http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1`. The request method is set to `DELETE`, and the URL is specified. The status bar indicates a `200 OK` response with a time of `159 ms` and a size of `276 B`. The response body is displayed in Pretty mode, showing the JSON object `{"message": "Course deleted successfully"}`.

HTTP `http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1` Save

DELETE `http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 159 ms Size: 276 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2   "message": "Course deleted successfully"
3
```