

CSE343- Web Development

Labs 8, 9

Loaiy Mahmoud Fatouh

23P0419

Lab 8

1- Adding 2 posts

The image displays two screenshots of the Postman application, illustrating the process of adding two posts to a REST API.

Top Screenshot:

- Request:** A POST request to `http://localhost:3000/posts` with a JSON body:

```
{  "title": "Title 1",  "content": "content of title 1"}
```
- Response:** The response is a 201 Created status with a 9 ms response time and 311 B of data. The JSON body is:

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

Bottom Screenshot:

- Request:** A POST request to `http://localhost:3000/posts` with a JSON body:

```
{  "title": "title 2",  "content": "content of title 2"}
```
- Response:** The response is a 201 Created status with a 13 ms response time and 311 B of data. The JSON body is:

```
{  "id": 3,  "title": "title 2",  "content": "content of title 2",  "comments": []}
```

2- Getting all posts

GET http://localhost:3000/posts

Send

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

Query Params

Key	Value	Bulk Edit
-----	-------	-----------

Body Cookies Headers (7) Test Results 200 OK 11 ms 381 B Save Response

Pretty Raw Preview Visualize JSON

```
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": []
13  }
```

3- Getting post by ID

GET http://localhost:3000/posts/3

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 200 OK 3 ms 306 B Save Response

Pretty Raw Preview Visualize JSON

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

4- Updating a post

The screenshot shows a REST client interface with a list of requests at the top. The selected request is a PUT to `http://localhost:3000/posts/3`. The request body is a JSON object: `{ "content": "content of title 2 updated" }`. The response is a 200 OK status with a 5 ms response time and 314 B of data. The response body is displayed in a pretty-printed JSON format: `{ "id": 3, "title": "title 2", "content": "content of title 2 updated", "comments": [] }`.

```
PUT http://localhost:3000/posts/3
```

```
{ "content": "content of title 2 updated" }
```

200 OK 5 ms 314 B

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

5- adding 3 comments

The screenshot shows a REST client interface with a list of requests at the top. The selected request is a POST to `http://localhost:3000/posts/3/comments`. The request body is a JSON object: `{ "text": "comment 3" }`. The response is a 201 Created status with a 4 ms response time and 272 B of data. The response body is displayed in a pretty-printed JSON format: `{ "id": 3, "text": "comment 3" }`.

```
POST http://localhost:3000/posts/3/comments
```

```
{ "text": "comment 3 ...." }
```

201 Created 4 ms 272 B

```
{ "id": 3, "text": "comment 3 ...." }
```

6- getting all comments

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

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

7- Deleting post by id

The screenshot shows a Postman interface with a DELETE request to `http://localhost:3000/posts/2`. The response is a 200 OK status with a response time of 6 ms and a body size of 341 B. The response body is displayed in JSON format, showing a message and the details of the deleted post:

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

Lab 9

1- Posting 2 courses

POST http://localhost:5000/api/courses

Send

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

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

```
1 {
2   "title": "Course Title 1",
3   "description": "Learning course 1",
4   "instructor": "jana tamer",
5   "price": 1200,
6   "category": "Web Development"
7 }
```

Body Cookies Headers (7) Test Results 201 Created 120 ms 510 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "title": "Course Title 1",
3   "description": "Learning course 1",
4   "instructor": "jana tamer".
5 }
```

POST http://localhost:5000/api/courses...

Send

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

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

```
4   "instructor": "jana maher",
5   "price": 3000,
6   "category": "Design"
7 }
```

Body Cookies Headers (7) Test Results 201 Created 141 ms 503 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "title": "Course Title 2",
3   "description": "Learninggg course 2",
4   "instructor": "jana maher",
5   "price": 3000,
6   "category": "Design",
7   "enrolledStudents": 0,
8   "_id": "694c3f20b6ad6bde74ee64f4",
9   "createdAt": "2025-12-24T10:29:36.350Z"
10 }
```

2-Get all courses

POST http://localhost:5000/api/courses GET http://localhost:5000/api/courses PUT http://localhost:5000/api/courses DEL http://localhost:5000/api/courses

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

GET http://localhost:5000/api/courses Send

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

Body Cookies Headers (7) Test Results Status: 200 OK Time: 66 ms Size: 769 B Save Response

Pretty Raw Preview Visualize JSON

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

3- Get course by ID

POST http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1 GET http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1 PUT http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1 DEL http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1

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

GET http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1 Send

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

Body Cookies Headers (7) Test Results Status: 200 OK Time: 56 ms Size: 505 B Save Response

Pretty Raw Preview Visualize JSON

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

4- Get course by title

The screenshot shows the Postman interface for a GET request. The URL bar contains `http://localhost:5000/api/courses/title/Course Title 2`. The method dropdown is set to **GET**. The **Send** button is visible. Below the URL bar, the **Params** tab is active, showing an empty table with columns **Key**, **Value**, and **Bulk Edit**. The **Body** tab is also visible. The **Test Results** section shows a status of **200 OK**, a time of **66 ms**, and a size of **498 B**. The **Save Response** button is present. The **Body** tab is selected, and the response is displayed in **JSON** format, showing a course object with fields like `_id`, `title`, `description`, `instructor`, `price`, `category`, `enrolledStudents`, `createdAt`, `updatedAt`, and `__v`.



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

5-Update course by ID

The screenshot shows the Postman interface for a PUT request. The URL bar contains `http://localhost:5000/api/courses/694c3ec1b6ad6bde74ee64f1`. The method dropdown is set to **PUT**. The **Send** button is visible. Below the URL bar, the **Params** tab is active, showing an empty table with columns **Key**, **Value**, and **Bulk Edit**. The **Body** tab is also visible. The **Test Results** section shows a status of **200 OK**, a time of **109 ms**, and a size of **505 B**. The **Save Response** button is present. The **Body** tab is selected, and the request body is displayed in **JSON** format, showing a course object with fields like `_id`, `title`, `description`, `instructor`, `price`, `category`, `enrolledStudents`, `createdAt`, `updatedAt`, and `__v`.

```
1 {
2   "price": 5000
3 }
```


6- Delete course by ID

 <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

3

"message": "Course deleted successfully"