

Birkbeck, University of London  
BSc Computing

Undergraduate level

Cloud Computing Concepts

MiniWall RESTful API report

Author: Alisena Mudaber,  
Student ID: **13184798**

## **1- Setup:**

In this section, a brief description of the project setup will be given. This project uses various libraries to complete its development. The following libraries are used in this project:

- Nodemon: used for easily re-running the project
- Mongoose: used for connecting to the mongoDB database
- Express: used for rapidly creating web apps
- Dotenv: used for environment variables
- Body-parser: used for parsing json data
- Bcryptjs: to encrypt / decrypt passwords
- Joi: used for data validation
- Jsonwebtoken: used for generating auth-token when users login.

The project is divided into 3 folders, containing the models, routes, and validation files. The entry point of the project is app.js.

## **2- Database models:**

The models represent the how data is structured in the database. The post model has a user field which represents which user is posting it, a title, description and date. Also, a like field which counts the number of likes the post has received. The user schema has a username, email, and password. The like schema takes in a post and user to identify which post is liked by which user. The comment schema consists of a comment, a user for knowing who posted it and post for what post does it belong to.

## **3- Routes:**

In this project there are four routes. Post, comment, user, and like. In the post endpoint users, can create, update, read, and delete posts by making calls to appropriate endpoints. In the comment route, users can create, read, update, and delete comments on specific posts. In the user route, the user can register and login. Lastly, the like route is used for liking a specific post.

## **4- RESTful API endpoints:**

- Localhost:3000/api/user/register
  - o Registers users with body json.
- Localhost:3000/api/user/login
  - o Logins users and generates auth-token
- Localhost:3000/api/post
  - o It either posts a post or reads all posts
- Localhost:3000/api/post/{postId}
  - o Gets, updates or deletes a single post by ID
- Localhost:3000/api/comment
  - o Posts a comment or gets all comments
- Localhost:3000/api/userId

- Gets all comments for a single user
- Localhost:3000/api/commentId
  - Either gets, updates or deletes a comment based on commentId
- Localhost:3000/api/like
  - Likes a specific post when the post id is provided in the request body
- Localhost:3000/api/like/postId
  - Gets the number of likes for a specific post

## 5- Test cases:

**TC1:**

The screenshot shows the Postman application interface. A POST request is being made to `localhost:3000/api/user/register`. The request body is set to `JSON (application/json)` and contains the following JSON data:

```

1 < [
2   "username": "Olga",
3   "email": "olga@gmail.com",
4   "password": "1234567"
5 ]

```

The response status is `200 OK`, with a time of `248 ms` and a size of `428 B`. The response body is also displayed in JSON format:

```

1 < [
2   {
3     "username": "Olga",
4     "email": "olga@gmail.com",
5     "password": "$2b$05$g5poX/25o.zuPAICkxN2H0nGUA36rF2xSN99.UjgeIaoCR8I8jjv6",
6     "_id": "639646cb9f79e4fa26485ec7",
7     "Date": "2022-12-11T21:08:27.226Z",
8     "__v": 0
9   }
10 ]

```

My Workspace

localhost:3000/api/user/register

POST localhost:3000/api/user/register

Body (11) JSON (application/json)

```

1+ {
2   "username": "Nick",
3   "email": "Nick@mail.com",
4   "password": "1234567"
5 }

```

Status: 200 OK Time: 128 ms Size: 428 B

My Workspace

localhost:3000/api/user/register

POST localhost:3000/api/user/register

Body (11) JSON (application/json)

```

1+ {
2   "username": "Mary",
3   "email": "Mary@mail.com",
4   "password": "1234567"
5 }

```

Status: 200 OK Time: 93 ms Size: 428 B

**TC2:**

The screenshot shows the Postman application interface. At the top, there's a header bar with 'My Workspace' and 'Sign In' buttons. Below the header is a navigation bar with tabs for 'POST Login', 'PATCH update current...', 'POST Signup', 'DELETE delete current us...', 'GET Get All Users', '[CONFLICT] GET gx...', 'GET localhost:30...', 'POST localhost:30...', 'POST localhost:30...', and 'POST localhost:30...'. The main area is titled 'localhost:3000/api/user/login'. It shows a 'POST' request to 'localhost:3000/api/user/login'. The 'Body' tab is selected, displaying the following JSON payload:

```
1 + {  
2   "email": "Old@gmail.com",  
3   "password": "1234567"  
4 }
```

Below the body, the 'Test Results' section shows a successful response with status 200 OK, time 37 ms, and size 565 B. The response body contains an 'auth-token' key with a long string value.

This screenshot is nearly identical to the one above, showing the same Postman interface and request details. The only difference is the email address in the payload, which has been changed to 'Nick@gmail.com'.

```
1 + {  
2   "email": "Nick@gmail.com",  
3   "password": "1234567"  
4 }
```

The 'Test Results' section again shows a successful response with status 200 OK, time 44 ms, and size 565 B, with the same 'auth-token' value as the previous request.

**TC3:**

localhost:3000/api/user/login

POST Body (JSON)

```
{
  "email": "Mary@mail.com",
  "password": "123456"
}
```

Status: 200 OK Time: 31 ms Size: 565 B

**TC4:**

localhost:3000/api/post

POST Headers (3)

KEY	VALUE	DESCRIPTION
Content-Type	application/json	
auth-token	eyJhbGciOiUzI1NlslnR5cCl6IkpxVCJ9eyJfaWQiOii2Mzk2NzlD1mNzllNGZHMyYR0DVhY2QlLCjpxXQl0jE2NzA30TMyOTB9.X9bb7hQ6iwNeezezn1Cuc-BYrjXWQHhgviIozQrc	
Key	Value	Description

Status: 401 Unauthorized Time: 9 ms Size: 272 B

TC4:

My Workspace | Invite | Sign In

History Collections APIs **BETA**

Save Responses Clear all

Today

**POST** localhost:3000/api/post

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/post

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/user/register

**POST** localhost:3000/api/user/register

**POST** localhost:3000/api/user/register

**POST** localhost:3000/api/comment

**POST** localhost:3000/api/comment

**GET** localhost:3000/api/like/63961

**GET** 480e51e5e553abaeafb6

**POST** localhost:3000/api/like

**POST** localhost:3000/api/like

**POST** localhost:3000/api/post

localhost:3000/api/post

POST Params Authorization Headers (12) Body Pre-request Script Tests Cookies Code Comments (0) Beautify

```
1 ~ {
  2   "user": "63964c6ca9f79e4fa264850c7",
  3   "title": "My first post",
  4   "description": "I am writing my first post",
  5 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 57 ms Size: 423 B Download

Pretty Raw Preview JSON

```
1 ~ {
  2   "user": "63964c6ca9f79e4fa264850c7",
  3   "title": "My first post",
  4   "description": "I am writing my first post",
  5   "likes": 0,
  6   "id": "639649789f79e4fa26485ad3",
  7   "date": "2022-12-11T21:19:52.894Z",
  8   "__v": 0
  9 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 57 ms Size: 423 B Download

Pretty Raw Preview JSON

## TC5:

My Workspace | Invite | Sign In

History Collections APIs **BETA**

Save Responses Clear all

Today

**POST** localhost:3000/api/post

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/post

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/post

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/user/login

**POST** localhost:3000/api/user/register

**POST** localhost:3000/api/user/register

**POST** localhost:3000/api/user/register

**POST** localhost:3000/api/comment

**POST** localhost:3000/api/comment

**GET** localhost:3000/api/like/63961

**GET** 480e51e5e553abaeafb6

**POST** localhost:3000/api/like

**POST** localhost:3000/api/like

**POST** localhost:3000/api/post

localhost:3000/api/post

POST Params Authorization Headers (13) Body Pre-request Script Tests Cookies Code Comments (0) Beautify

```
1 ~ {
  2   "user": "639647449f79e4fa264850ca",
  3   "title": "My first post",
  4   "description": "My name is Nick",
  5 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 52 ms Size: 412 B Download

Pretty Raw Preview JSON

```
1 ~ {
  2   "user": "639647449f79e4fa264850ca",
  3   "title": "My first post",
  4   "description": "My name is Nick",
  5   "likes": 0,
  6   "id": "639649789f79e4fa26485ad6",
  7   "date": "2022-12-11T21:23:00.874Z",
  8   "__v": 0
  9 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 52 ms Size: 412 B Download

Pretty Raw Preview JSON

## TC6:

The screenshot shows the Postman application interface. The top navigation bar includes 'New', 'Import', 'Runner', 'My Workspace', 'Invite', and a user icon. The left sidebar has sections for 'History' (with a 'Save Responses' toggle), 'Collections', 'APIs (BETA)', and a 'Today' section listing various API requests. The main workspace shows a 'POST' request to 'localhost:3000/api/post'. The 'Body' tab is selected, showing a JSON payload:

```
1 - {  
2   "user": "639647ed9f79e4fa26485acd",  
3   "title": "My name is Mary",  
4   "description": "My name is Mary"  
5 }
```

The 'Body' tab also includes tabs for 'Pretty', 'Raw', 'Preview', and 'JSON'. Below the preview area, the response details are shown: Status: 200 OK, Time: 75 ms, Size: 414 B, and a 'Download' link. The bottom of the screen features a toolbar with icons for search, file operations, and help.

**TC7:**

TC8:

**TC9:**

**TC10:**

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'New', 'Import', 'Runner', and other icons. The title bar says 'My Workspace' with an 'Invite' button. Below the title bar, there's a toolbar with filters and a 'History' section. The main area displays a collection named 'Dev: Natours'. The collection contains several API endpoints:

- POST Login**: Returns a token (e.g., eyJhbGciOiJzIiNisnR5C6lkpXvC9...).
- PATCH update**: Returns a token (e.g., eyJhbGciOiJzIiNisnR5C6lkpXvC9...).
- POST Signup**: Returns a token (e.g., eyJhbGciOiJzIiNisnR5C6lkpXvC9...).
- DEL delete**: Returns a token (e.g., eyJhbGciOiJzIiNisnR5C6lkpXvC9...).
- GET GetAll**: Returns a JSON array of posts. One post is highlighted with a checkmark in the 'auth-token' row.
- GET loc**: Returns a token (e.g., eyJhbGciOiJzIiNisnR5C6lkpXvC9...).
- POST loc**: Returns a token (e.g., eyJhbGciOiJzIiNisnR5C6lkpXvC9...).
- GET loc**: Returns a token (e.g., eyJhbGciOiJzIiNisnR5C6lkpXvC9...).

The 'Temporary Headers' section shows a single entry: 'Key' (Content-Type) with 'Value' (application/json). The 'Body' section shows the raw JSON response for the highlighted post:

```
1+ [
2+   {
3+     "_id": "63964979f79e4fa26485ad3",
4+     "user": "639646d9f79e4fa26485e7c",
5+     "title": "My first post",
6+     "description": "I am writing my first post",
7+     "likes": 0,
8+     "date": "2022-12-11T21:19:52.894Z",
9+     "__v": 0
10+   },
11+   {
12+     "_id": "63964549f79e4fa26485ad6",
13+     "user": "639647a49f79e4fa26485aco",
14+     "title": "My first post",
15+     "description": "My name is Nick",
16+     "likes": 0,
17+     "date": "2022-12-11T21:23:00.874Z",
18+     "__v": 0
19+   },
20+   {
21+     "_id": "63964d9f79e4fa26485cd",
22+     "user": "639647e9f79e4fa26485cd",
23+     "title": "My name is Mory",
24+     "description": "My name is Mory",
25+     "likes": 0,
26+     "date": "2022-12-11T21:37:35.134Z",
27+     "__v": 0
28+   }
29 ]
```

The status bar at the bottom indicates 'Status: 200 OK', 'Time: 29 ms', and 'Size: 782 B'. There are also 'Send', 'Save', and 'Download' buttons.

**TC11:**

TC12:

The screenshot shows the Postman interface with a successful API call. The URL is `localhost:3000/api/like`. The request method is `POST`. The body is set to `JSON (application/json)` and contains the following JSON:

```
1 ~ {  
2   "post": "63964d9fcd8c845d9cbcf35",  
3   "user": "639647ed9f79e4fa26485ac7",  
4 }
```

The response status is `200 OK`, time `83 ms`, and size `414 B`.

The screenshot shows the Postman interface with a successful API call. The URL is `localhost:3000/api/like`. The request method is `POST`. The body is set to `JSON (application/json)` and contains the following JSON:

```
1 ~ {  
2   "post": "63964d9fcd8c845d9cbcf35",  
3   "user": "639646cc99f79e4fa26485ac7",  
4 }
```

The response status is `200 OK`, time `129 ms`, and size `414 B`.

**TC13:**

My Workspace Dev: Natours

localhost:3000/api/like

POST localhost:3000/api/like

```

1+ {
2   "post": "63964d9fed0bc845d9cbcf35",
3   "user": "639647e09f79e4fa26485acd"
4 }

```

Status: 400 Bad Request Time: 65 ms Size: 282 B

## TC14:

My Workspace Dev: Natours

localhost:3000/api/like/63964d9fed0bc845d9cbcf35

GET localhost:3000/api/like/63964d9fed0bc845d9cbcf35

```

1+ {
2   "_id": "63964d9fed0bc845d9cbcf35",
3   "likes": 2
4 }

```

Status: 200 OK Time: 40 ms Size: 279 B

## TC15:

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'My Workspace' and 'Invite' buttons. Below the bar, a search field and filter dropdown are present. The main area has tabs for 'History', 'Collections', and 'APIs'. A 'POST' request is selected, targeting 'localhost:3000/api/post'. The 'Headers' tab shows 'Content-Type: application/json' and 'auth-token' fields with various token values. The 'Body' tab displays a JSON array of posts. The first post has an '\_id' of '63964d9fed08c845d9cbcf35', 'user' of '639647ed9f79e4fa26485ec0', 'title' of 'My name is Mary', 'description' of 'My name is Mary', 'likes' of 2, 'date' of '2022-12-11T21:37:35.134Z', and '\_v' of 0. The second post has an '\_id' of '63964d9f979ed9f79e4fa26485ad3', 'user' of '639647ed9f79e4fa26485ec7', 'title' of 'first post', 'description' of 'I am writing my first post', 'likes' of 0, 'date' of '2022-12-11T21:19:52.894Z', and '\_v' of 0. The third post has an '\_id' of '63964d9f79ed9f79e4fa26485ad6', 'user' of '639647ed9f79e4fa26485ec0', 'title' of 'My first post', 'description' of 'My name is Nick', 'likes' of 0, 'date' of '2022-12-11T21:23:00.874Z', and '\_v' of 0. The fourth post has an '\_id' of '63964d9f79ed9f79e4fa26485acd', 'user' of '639647ed9f79e4fa26485ad8', 'title' of 'post', 'description' of ' ', 'likes' of 0, 'date' of '2022-12-11T21:23:00.874Z', and '\_v' of 0. The fifth post has an '\_id' of '63964d9f79ed9f79e4fa26485ad9', 'user' of '639647ed9f79e4fa26485ad0', 'title' of ' ', 'description' of ' ', 'likes' of 0, 'date' of '2022-12-11T21:23:00.874Z', and '\_v' of 0. The bottom right corner shows status: 200 OK, time: 33 ms, size: 782 B.

```

1 - [
2   {
3     "_id": "63964d9fed08c845d9cbcf35",
4     "user": "639647ed9f79e4fa26485ec0",
5     "title": "My name is Mary",
6     "description": "My name is Mary",
7     "likes": 2,
8     "date": "2022-12-11T21:37:35.134Z",
9     "_v": 0
10   },
11   {
12     "_id": "63964d9f979ed9f79e4fa26485ad3",
13     "user": "639647ed9f79e4fa26485ec7",
14     "title": "first post",
15     "description": "I am writing my first post",
16     "likes": 0,
17     "date": "2022-12-11T21:19:52.894Z",
18     "_v": 0
19   },
20   {
21     "_id": "63964d9f79ed9f79e4fa26485ad6",
22     "user": "639647ed9f79e4fa26485ec0",
23     "title": "My first post",
24     "description": "My name is Nick",
25     "likes": 0,
26     "date": "2022-12-11T21:23:00.874Z",
27     "_v": 0
28   },
29 ]

```

## References:

medium.com, How to increment a number value in mongoose, available online at:  
<https://medium.com/@salonimalhotra1ind/how-to-increment-a-number-value-in-mongoose-785066ba09d8>

Last accessed: 11/12/2022