

Guide to Generating and Editing To-Do Lists and Titles

Scribe 

- 1 Navigate to <https://please.in:8181/docs#>

- 2 Click "/generate_new_token/"

misc

GET /generate_new_token/ Generate New Token

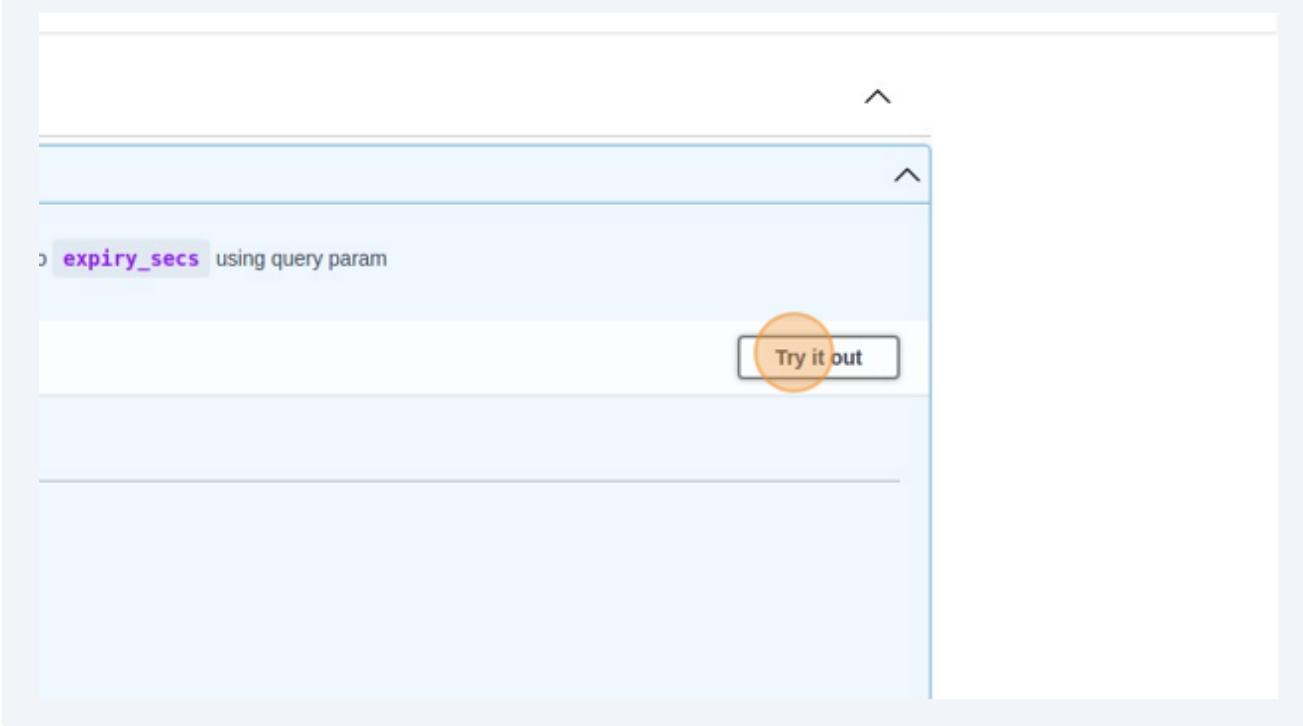
GET /login/ Users

users

GET /users/ Users

POST /users/ Users

- 3 Click "Try it out"



- 4 Click the "pass_phase" field.

A screenshot of a "Parameters" section from an API documentation tool. The section title is "Parameters". Below it is a table with two rows. The first row has "Name" and "Description" columns. The second row contains "expiry_secs" with the description "expiry_secs integer (query)". The third row contains "pass_phase" with the description "pass_phase string (query)". The "pass_phase" input field is highlighted with a thick orange circle. At the bottom of the screen, there is a blue bar with some white text that is mostly obscured by a dark overlay. A small circular icon with a blue and red gradient is visible on the left side.

- 5 Type "123456"

- 6 Click "Execute"

The screenshot shows a configuration interface for a REST API endpoint. On the left, there is a sidebar with a circular icon containing a blue and red gradient. The main area has a light gray header bar. Below it, there are two sections: 'Parameters' and 'Responses'.
Parameters Section:
A table with columns 'Name' and 'Description'. It contains two rows:

- Parameter: expiry_secs, Type: integer, Description: expiry_secs, Value: 123456
- Parameter: pass_phase, Type: string, Description: (query), Value: 123456

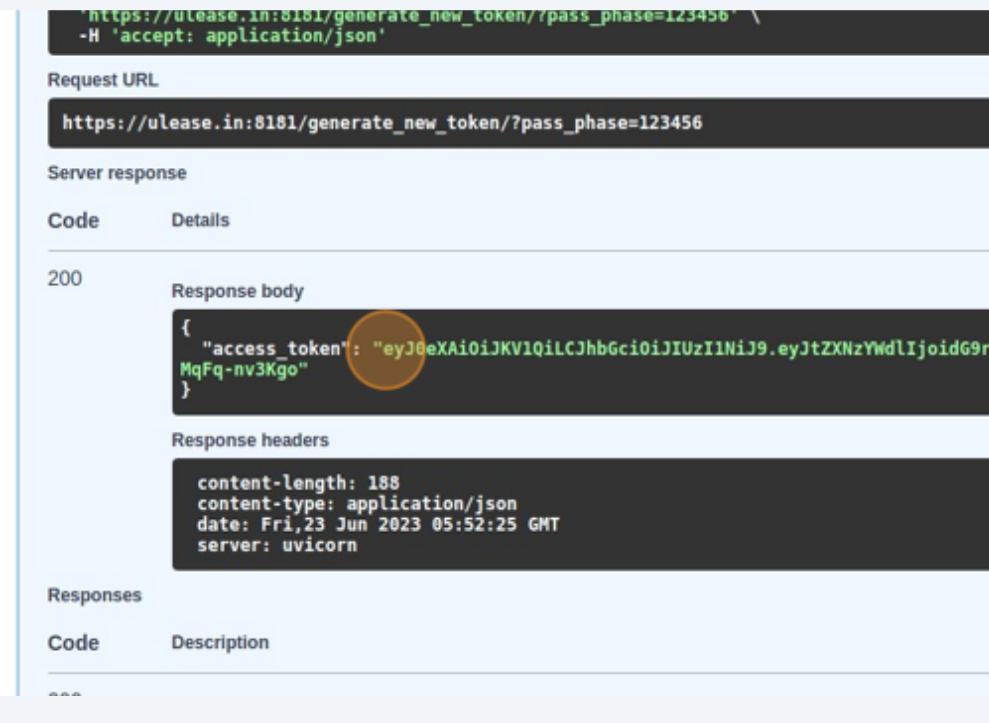
Responses Section:
A table with columns 'Code' and 'Description'. It contains one row:

- Code: 200

A large blue button labeled 'Execute' is centered below the parameters section. A circular highlight is placed over the 'Execute' button.

7

Click ""eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJtZXNzYWdlIjoidG9rZW4gZ2VuZXJhdGVkIGZvcIB0ZXN0aW5nIHB1cnBvc2UiLCJleHAIoJE2OTUyNzU1OTI9.VjDaNME3SI92tau2G..."



```
'https://ulease.in:8181/generate_new_token/?pass_phase=123456' \
-H 'accept: application/json'
```

Request URL
`https://ulease.in:8181/generate_new_token/?pass_phase=123456`

Server response

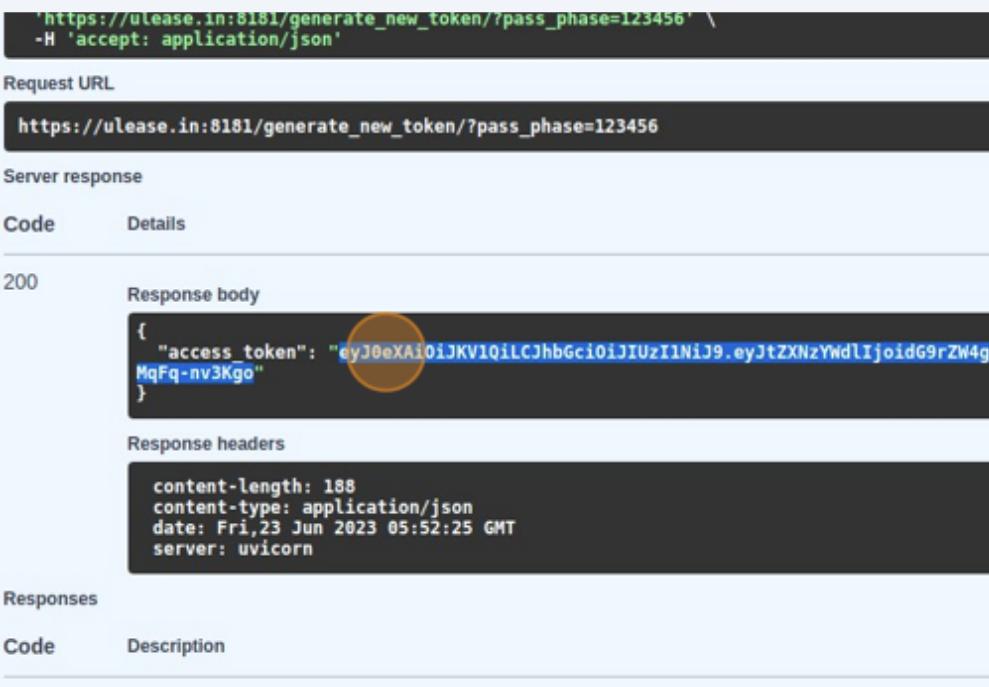
Code	Details
200	Response body <pre>{ "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJtZXNzYWdlIjoidG9rMqFq-nv3Kgo" }</pre> Response headers <pre>content-length: 188 content-type: application/json date: Fri,23 Jun 2023 05:52:25 GMT server: uvicorn</pre>

Responses

Code	Description
...	

8

Right-click ""eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJtZXNzYWdlIjoidG9rZW4gZ2VuZXJhdGVkIGZvcIB0ZXN0aW5nIHB1cnBvc2UiLCJleHAIoJE2OTUyNzU1OTI9.VjDaNME3SI92tau2G..."



```
'https://ulease.in:8181/generate_new_token/?pass_phase=123456' \
-H 'accept: application/json'
```

Request URL
`https://ulease.in:8181/generate_new_token/?pass_phase=123456`

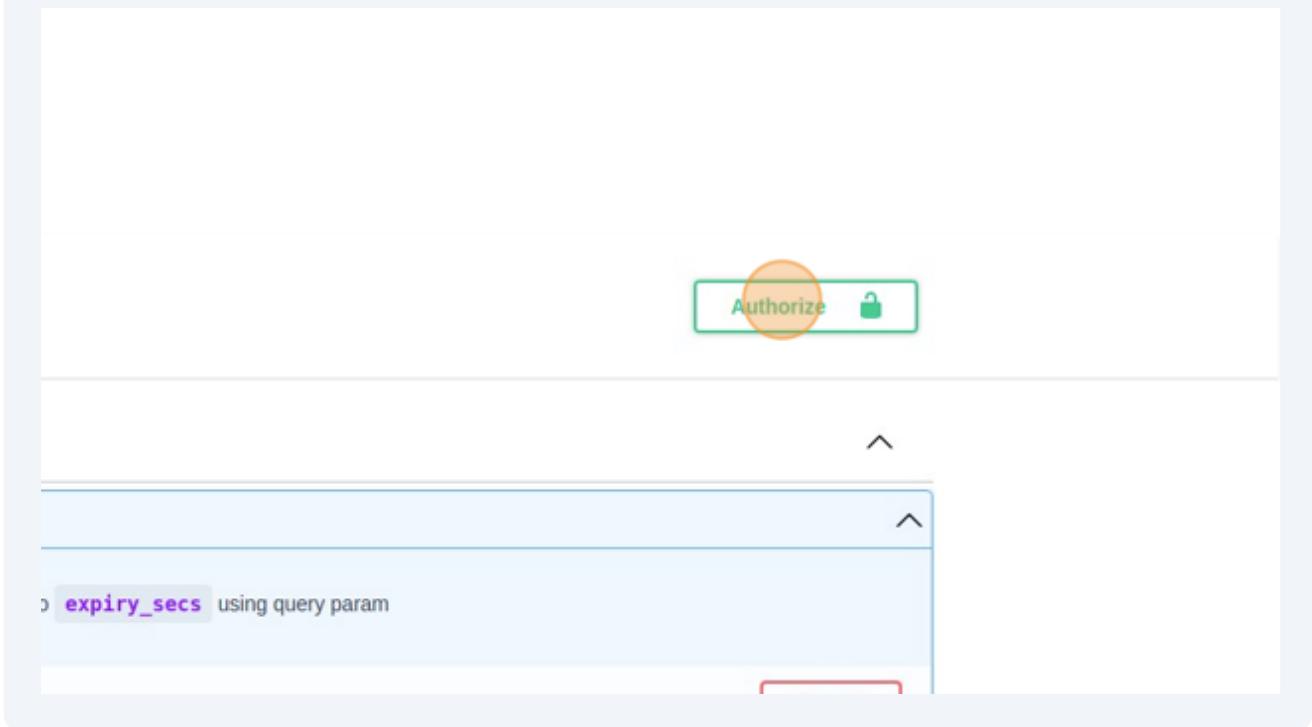
Server response

Code	Details
200	Response body <pre>{ "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJtZXNzYWdlIjoidG9rZW4gMqFq-nv3Kgo" }</pre> Response headers <pre>content-length: 188 content-type: application/json date: Fri,23 Jun 2023 05:52:25 GMT server: uvicorn</pre>

Responses

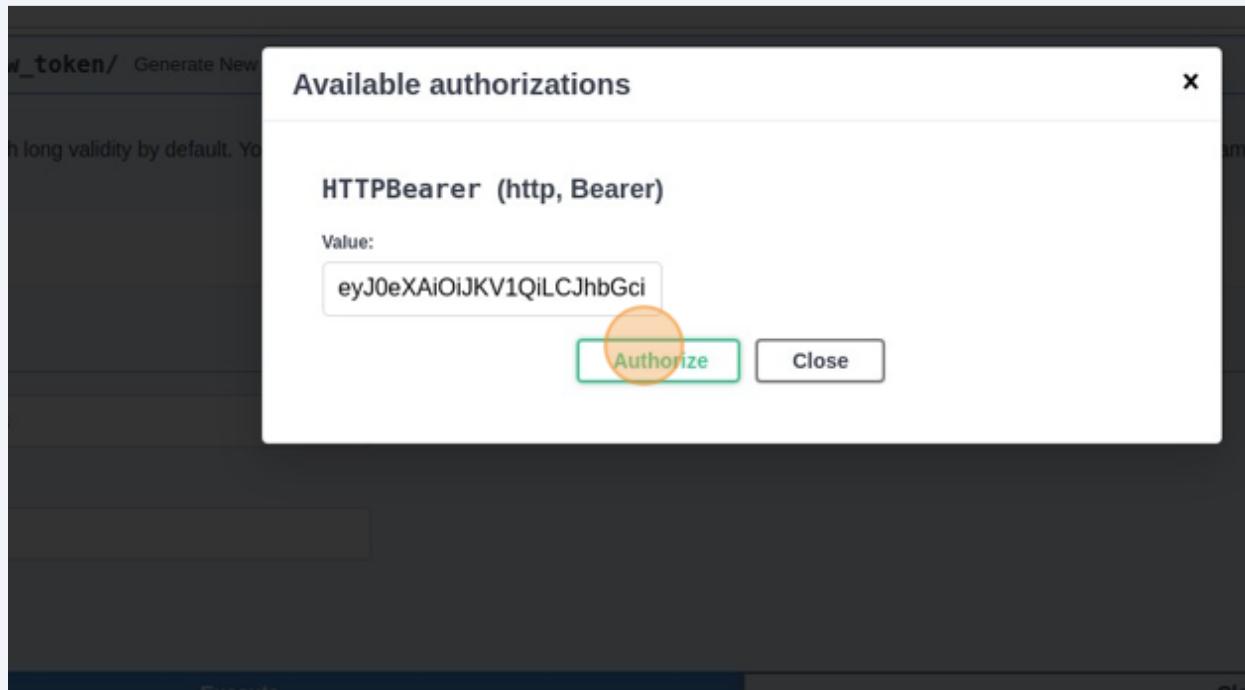
Code	Description
...	

9 Click "Authorize"

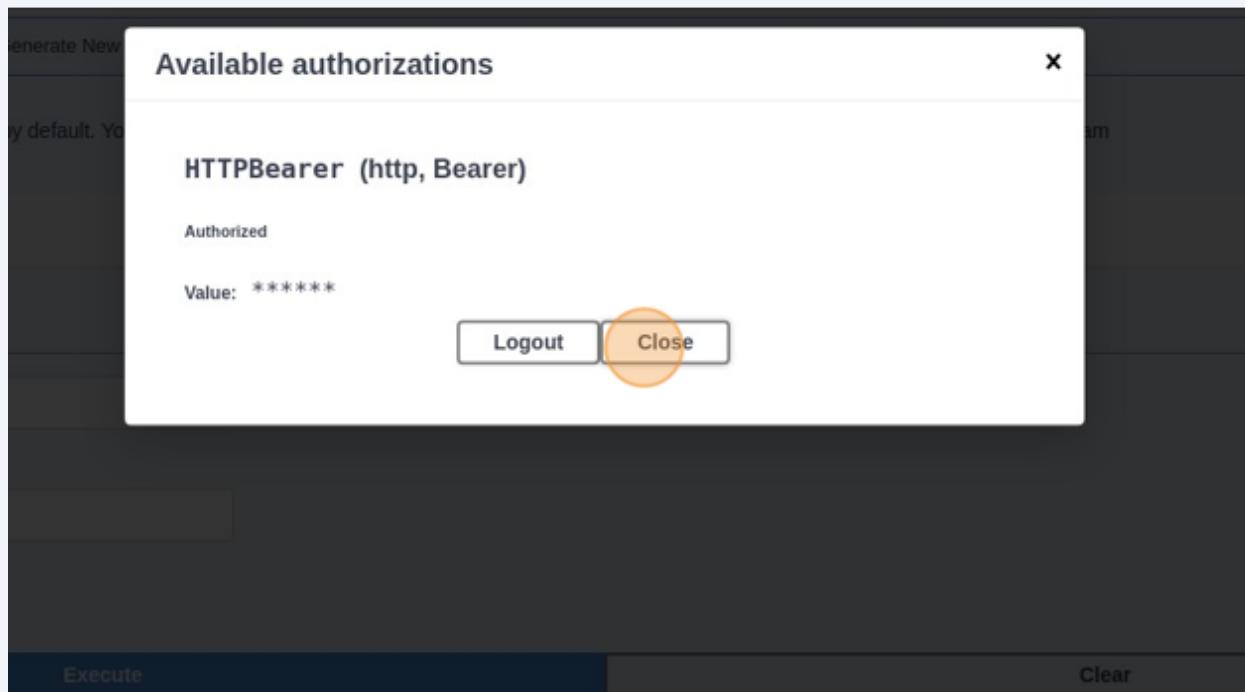


10 Press **CTRL + V**

11 Click "Authorize"



12 Click "Close"



13 Click here.

The screenshot shows a section of a Swagger UI interface. At the top left, there is a link labeled "openapi_json". Below it, the word "misc" is displayed. Under "misc", there is a blue button labeled "GET" followed by the endpoint "/generate_new_token/" and the description "Generate New Token". A large orange circle highlights the endpoint text. Below this, a detailed description states: "Generates an access token with long validity by default. You can also generate with custom expiry by passing value in seconds to `expiry_secs`". A "Parameters" section follows, containing a table with two columns: "Name" and "Description".

14 Click "Generate New Token"

This screenshot is identical to the one above, showing the "misc" section and the "Generate New Token" endpoint. A large orange circle highlights the endpoint text again. The detailed description and the "Parameters" table are also present.

15 Click "Users"

The screenshot shows a list of API endpoints for the 'users' resource:

- GET /generate_new_token/** Generate New Token
- GET /login/** Users
- users**

 - GET /users/** Users
 - POST /users/** Users (highlighted with an orange circle)
 - GET /users/{record_id}** Users
 - DELETE /users/{record_id}** Users
 - PATCH /users/{record_id}** Users

- todo**

16 Click "{
"user_name": "user_name",
"mail_id": "mail@gmail.com",
"password": "1234"
}"

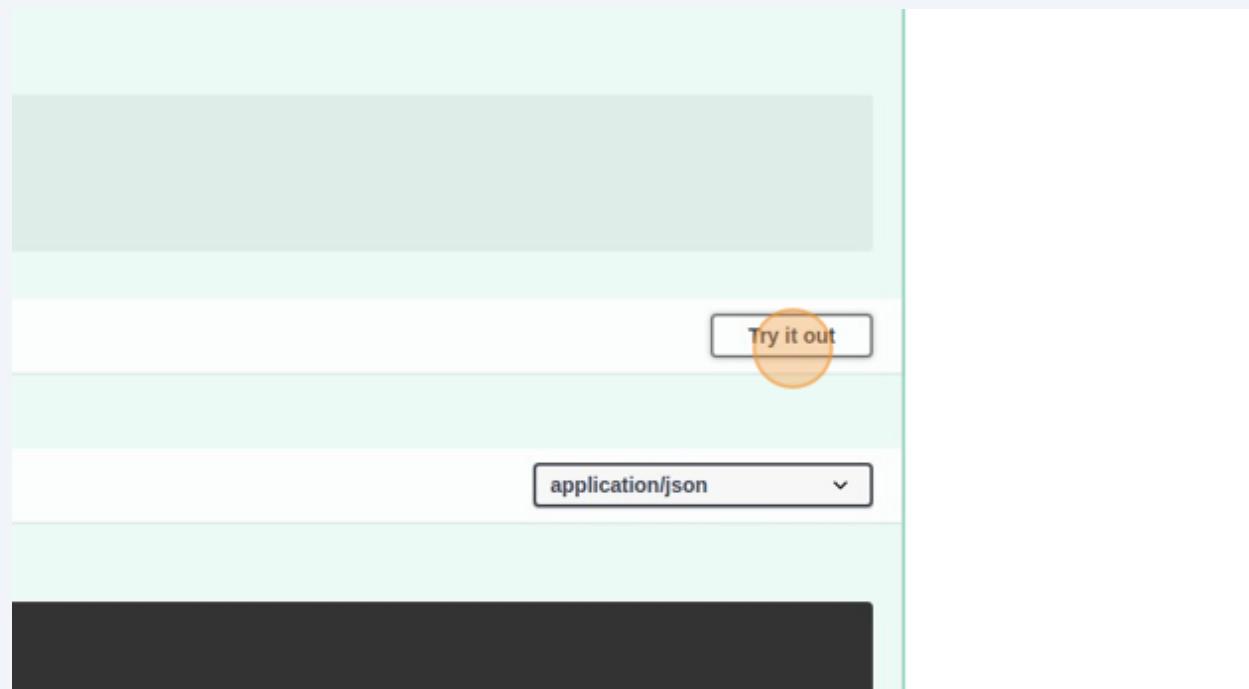
The screenshot shows the details for the **POST /users/** endpoint:

- POST /users/** Users
- Data is expected in json like
 - Copy JSON and Execute to post records to Database
- ```
{
 "user_name": "user_name",
 "mail_id": "mail@gmail.com",
 "password": "1234"
}
```

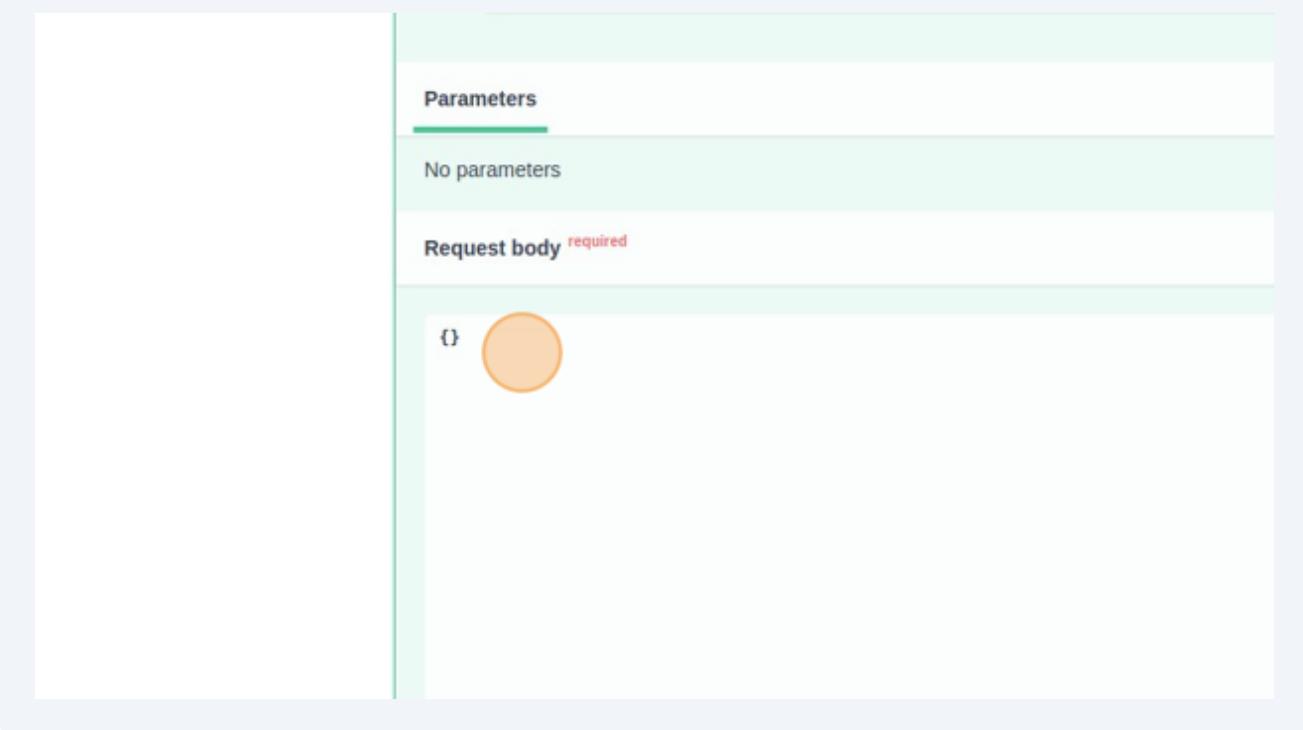
(The JSON object is highlighted with an orange circle.)
- Parameters**
  - No parameters
- Request body** required
  - Example Value | Schema

**17** Press **CTRL + C**

**18** Click "Try it out"

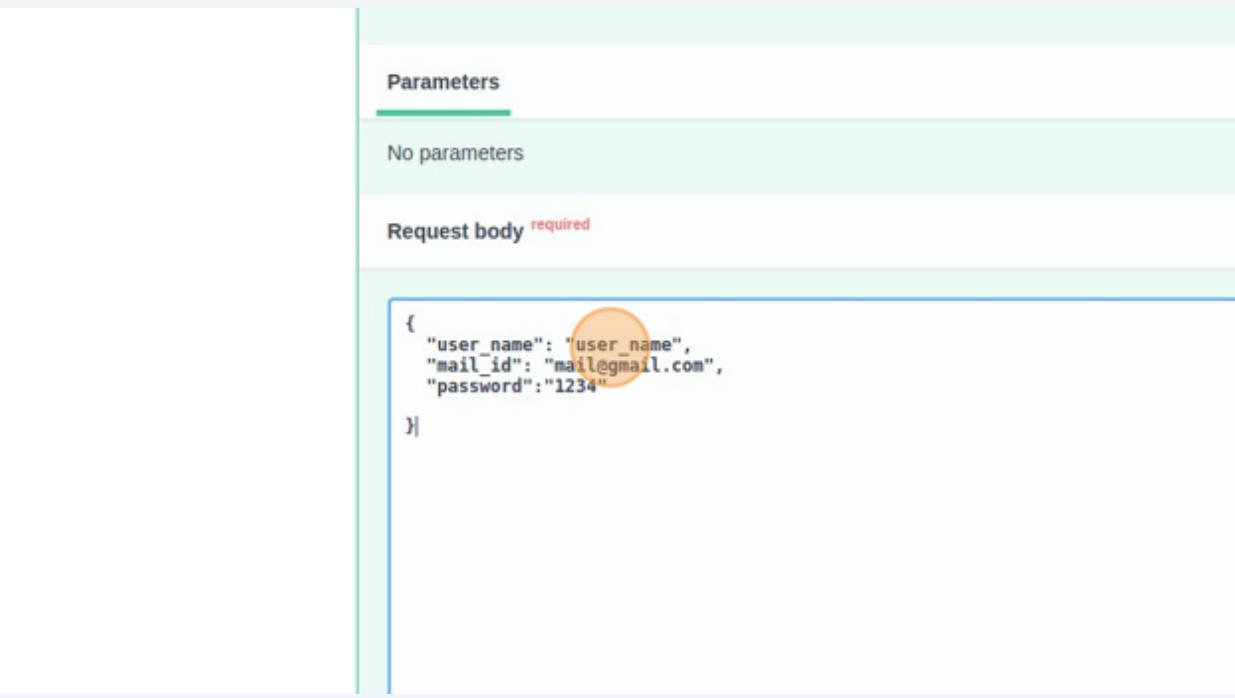


**19** Double-click this field.



**20** Press **CTRL + V**

**21** Double-click this field.



The screenshot shows a JSON editor interface. At the top, there's a green header bar with the word "Parameters". Below it, a light green section says "No parameters". Underneath that is another green header bar labeled "Request body" with a red "required" tag. The main content area contains a JSON object:

```
{
 "user_name": "user_name",
 "mail_id": "mail@gmail.com",
 "password": "1234"
}
```

The "user\_name" field is highlighted with a red circle.

**22** Type "shameem"

**23** Double-click this field.

The screenshot shows a JSON editor interface. At the top, there's a section labeled "Parameters" with the sub-section "No parameters". Below this is a section labeled "Request body" with the word "required" in red. The request body contains the following JSON code:

```
{
 "user_name": "shameem",
 "mail_id": "mail@gmail.com",
 "password": "1234"
}
```

The word "password" is highlighted with a yellow oval, indicating it is the field to be double-clicked.

**24** Type "shameem"

**25** Click "Execute"

The screenshot shows a user interface for executing a REST API call. At the top, there is a large, empty input field. Below it is a blue button labeled "Execute" with an orange circle highlighting the central part where the cursor would click. Underneath the button, the word "Responses" is displayed. A table follows, with columns for "Code" and "Description". One row shows "200" and "Successful Response". At the bottom of the table, the text "Media type" is visible.

| Code | Description         |
|------|---------------------|
| 200  | Successful Response |

**26** Click "POST /users/ Users"

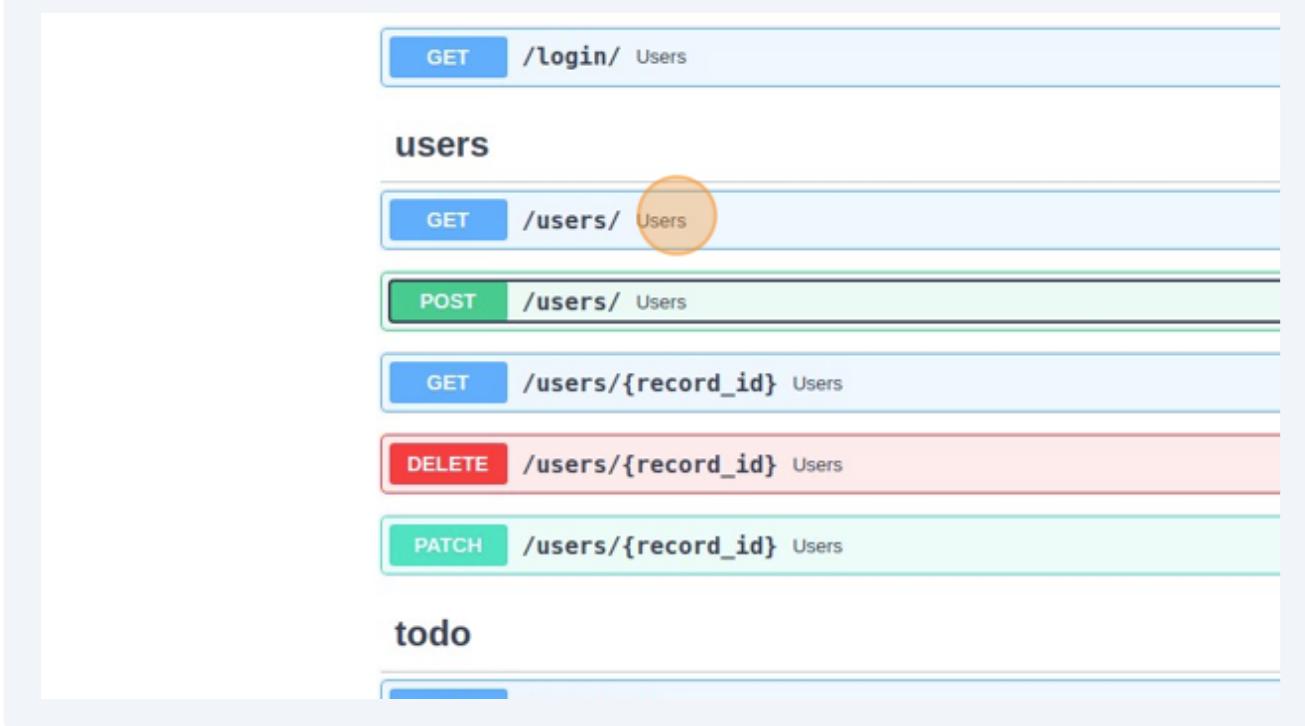
The screenshot shows a REST API interface with a main title "users". Below it, there are three buttons: a blue "GET" button, a green "POST" button, and a blue "GET" button for "/users/". The "POST" button is highlighted with an orange circle. Below the buttons, a list of instructions and sample JSON data is provided. At the bottom, there is a section titled "Parameters".

• Data is expected in json like  
• Copy JSON and Execute to post records to Database

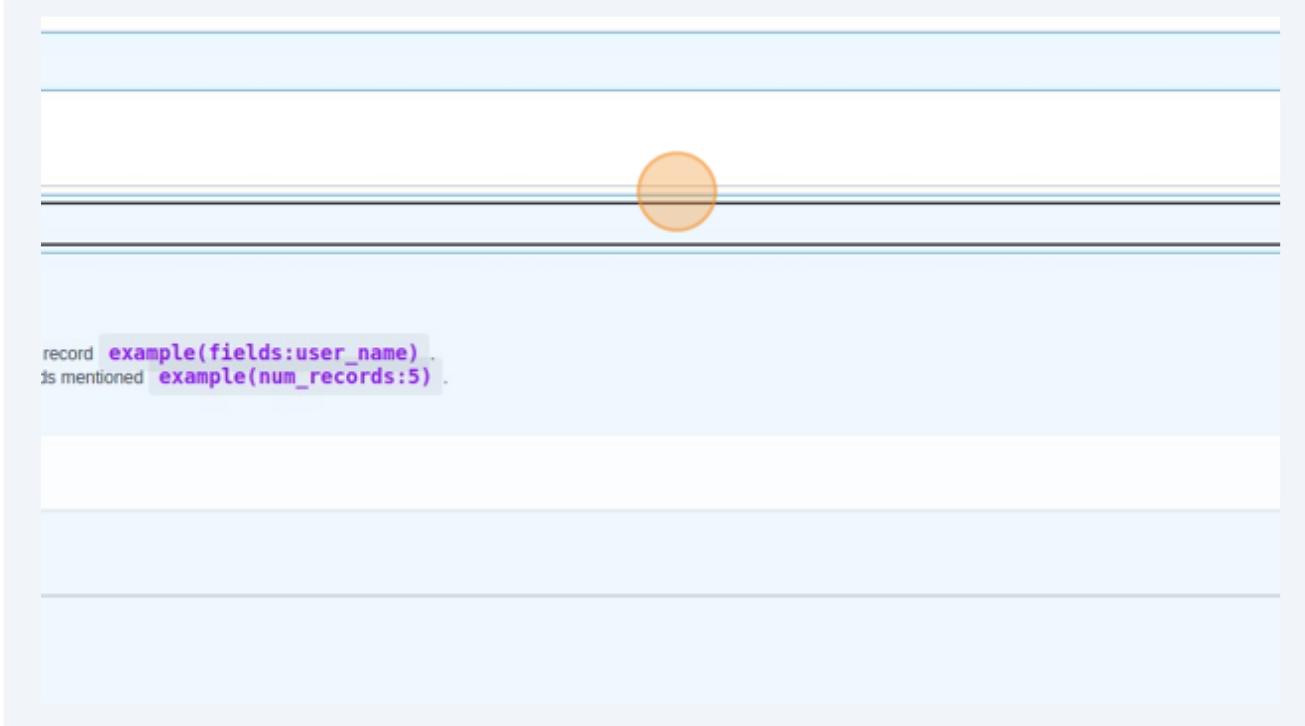
```
{
 "user_name": "user_name",
 "mail_id": "mail@gmail.com",
 "password": "1234"
}
```

Parameters

27 Click "Users"



28 Click here.



29

Click "GET  
/users/  
Users"



I record `example(fields:user_name)`.  
rds mentioned `example(num_records:5)`.

30

Click "GET  
/login/  
Users"

### misc

`GET /generate_new_token/` Generate New Token

`GET /login/` Users



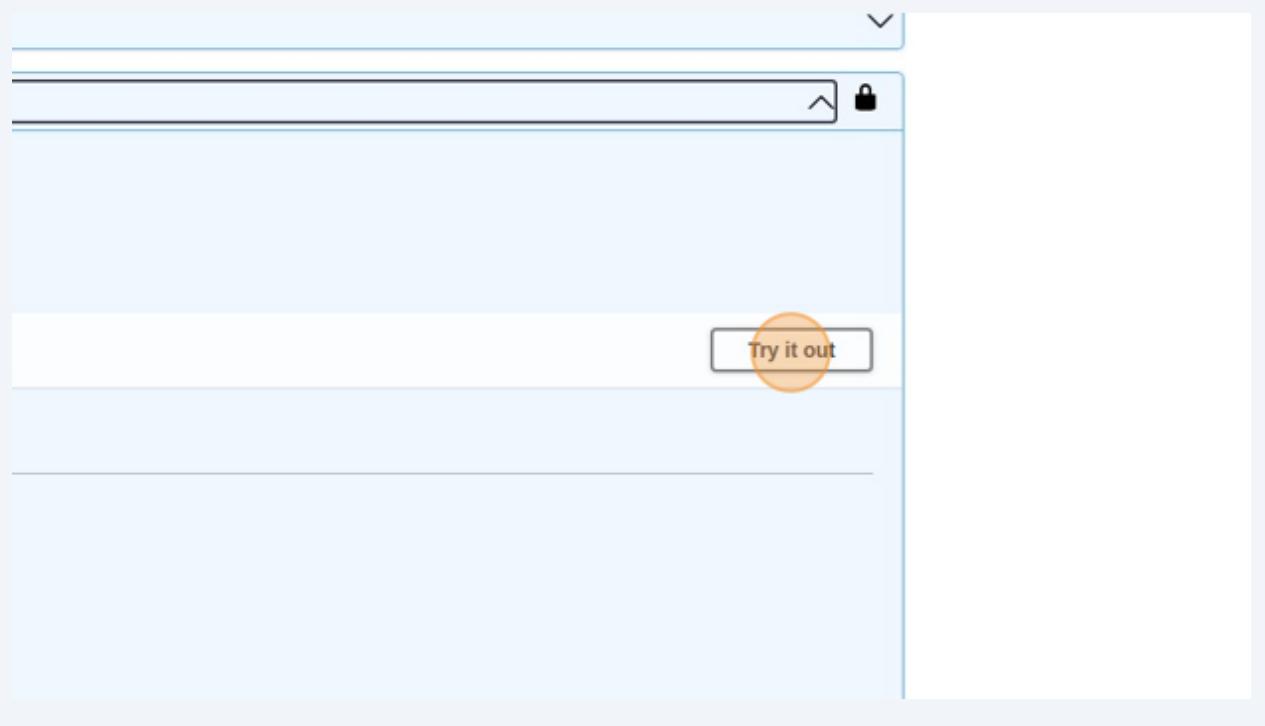
### users

`GET /users/` Users

`POST /users/` Users

`GET /users/{record_id}/` Users

31 Click "Try it out"



32 Click the "user\_mail" field.

A screenshot of a Swagger API documentation page. On the left, there is a sidebar with sections like "Parameters" and "Responses". The main area shows a table for "Parameters". The first row has columns for "Name" and "Description". The "Name" column contains "user\_mail \* required" and the "Description" column contains "user\_mail". This row is highlighted with an orange circle around the "user\_mail" input field. The second row has columns for "Name" and "Description". The "Name" column contains "user\_password \* required" and the "Description" column contains "user\_password". A blue button at the bottom is partially visible.

**33** Type "shameem@gmail.com"

**34** Click the "user\_password" field.

| Parameters                                    |                                            |
|-----------------------------------------------|--------------------------------------------|
| Name                                          | Description                                |
| user_mail * required<br>string<br>(query)     | shameem@gmail.com                          |
| user_password * required<br>string<br>(query) | <input type="text" value="user_password"/> |

| Responses                              |             |
|----------------------------------------|-------------|
| Code                                   | Description |
| <input type="button" value="Execute"/> |             |

**35** Type "1234"

**36** Click "Execute"

| Name                                                 | Description       |
|------------------------------------------------------|-------------------|
| <b>user_mail</b> * required<br>string<br>(query)     | shameem@gmail.com |
| <b>user_password</b> * required<br>string<br>(query) | 1234              |

Execute

**Responses**

| Code | Description         |
|------|---------------------|
| 200  | Successful Response |

**37** Click here.

**misc**

|     |                      |                    |
|-----|----------------------|--------------------|
| GET | /generate_new_token/ | Generate New Token |
|-----|----------------------|--------------------|

|     |         |       |
|-----|---------|-------|
| GET | /login/ | Users |
|-----|---------|-------|

Parameters:

- **fields**: This retrieves only the field name given from all record `example(fields:user_name)`.
- **num\_records**: This retrieves only the number of records mentioned `example(num_records:5)`.

**Parameters**

| Name                        | Description |
|-----------------------------|-------------|
| <b>user_mail</b> * required |             |

38 Click "Users"

## misc

GET /generate\_new\_token/ Generate New Token

GET /login/ Users



Parameters:

- fields: This retrieves only the field name given from all record `example(fields:user_name)`.
- num\_records: This retrieves only the number of records mentioned `example(num_records:5)`.

### Parameters

| Name | Description |
|------|-------------|
|------|-------------|

|           |            |
|-----------|------------|
| user mail | * required |
|-----------|------------|

39 Click "/add\_remove\_todo/"

## todo

GET /todo/ Users

GET /todo/{record\_id} Users

DELETE /todo/{record\_id} Users

POST /add\_remove\_todo/ Users

PATCH /edit\_todo\_title/{record\_id} Users

## code

GET /download\_code/ Files

**40** Click "Users"

The screenshot shows a list of API endpoints under the 'users' category. The 'GET /users/' endpoint is highlighted with an orange circle. The other endpoints listed are POST /users/, GET /users/{record\_id}, DELETE /users/{record\_id}, and PATCH /users/{record\_id}.

| users  |                          |
|--------|--------------------------|
| GET    | /users/ Users            |
| POST   | /users/ Users            |
| GET    | /users/{record_id} Users |
| DELETE | /users/{record_id} Users |
| PATCH  | /users/{record_id} Users |

**41** Click "Try it out"

The screenshot shows a browser window with a URL bar containing 'https://...'. Below the URL bar is a large input field. At the bottom of the page, there is a 'Try it out' button, which is highlighted with an orange circle.

## 42 Click "Execute"

| Name                               | Description                                    |
|------------------------------------|------------------------------------------------|
| fields<br>array[string]<br>(query) | <input type="button" value="Add string item"/> |
| num_records<br>integer<br>(query)  | <input type="text" value="num_records"/>       |

**Responses**

| Code | Description         |
|------|---------------------|
| 200  | Successful Response |

## 43 Click ""user\_name""

```
{
 "success": true,
 "total_count": 2,
 "count": 2,
 "records": [
 {
 "id": 4,
 "user_name": "shameem",
 "mail_id": "shameem@gmail.com",
 "password": "1234",
 "created_at": "2023-06-23T05:53:43.844609",
 "updated_at": "2023-06-23T05:53:43.844609"
 },
 {
 "id": 3,
 "user_name": "user_name",
 "mail_id": "mail@gmail.com",
 "password": "1234",
 "created_at": "2023-06-23T05:52:50.042753",
 "updated_at": "2023-06-23T05:52:50.042753"
 }
]
}
```

### Response headers

```
content-length: 379
content-type: application/json
date: Fri, 23 Jun 2023 05:52:25 GMT
server: uvicorn
```

Responses

**44** Click ":"

<https://ulease.in:8181/users/>

Server response

| Code | Details       |
|------|---------------|
| 200  | Response body |

```
{
 "success": true,
 "total_count": 2,
 "count": 2,
 "records": [
 {
 "id": 4,
 "user_name": "shameem",
 "mail_id": "shameem@gmail.com",
 "password": "1234",
 "created_at": "2023-06-23T05:53:43.844609",
 "updated_at": "2023-06-23T05:53:43.844609"
 },
 {
 "id": 3,
 "user_name": "user_name",
 "mail_id": "mail@gmail.com",
 "password": "1234",
 "created_at": "2023-06-23T05:52:50.042753",
 "updated_at": "2023-06-23T05:52:50.042753"
 }
]
}
```

**45** Right-click "4"

<https://ulease.in:8181/users/>

Server response

| Code | Details       |
|------|---------------|
| 200  | Response body |

```
{
 "success": true,
 "total_count": 2,
 "count": 2,
 "records": [
 {
 "id": 4,
 "user_name": "shameem",
 "mail_id": "shameem@gmail.com",
 "password": "1234",
 "created_at": "2023-06-23T05:53:43.844609",
 "updated_at": "2023-06-23T05:53:43.844609"
 },
 {
 "id": 3,
 "user_name": "user_name",
 "mail_id": "mail@gmail.com",
 "password": "1234",
 "created_at": "2023-06-23T05:52:50.042753",
 "updated_at": "2023-06-23T05:52:50.042753"
 }
]
}
```

46 Click "Try it out"



47 Click the "user\_id" field.

A screenshot of a "Try it out" interface for a POST request. The request body is shown as JSON: {"todoitem": "apple", "method" : "ADD/DELETE" - default ADD}. Below the request body, there is a table titled "Parameters". The table has two columns: "Name" and "Description". There are four rows in the table. The first row corresponds to the "user\_id" field in the JSON, with the value "user\_id" in the "Description" column. This input field is highlighted with a thick orange circle. The other three rows correspond to the "title", "todoitem", and "method" fields in the JSON, with their respective values "title", "todoitem", and "method" in the "Description" column.

48 Press **CTRL + V**

49

Click "{  
"user\_id": 1,  
"title": "shopping list",  
"todoitem": "apple",  
"method": "ADD/DELETE" - default ADD  
}"

The screenshot shows a JSON editor interface. At the top, there is a button labeled "Copy JSON and Execute to post records to Database". Below it is a code block containing a JSON object:

```
{
 "user_id": 1,
 "title": "shopping list",
 "todoitem": "apple",
 "method": "ADD/DELETE" - default ADD
}
```

Below the code block is a section titled "Parameters" with a green underline. It contains three entries:

| Name                                     | Description |
|------------------------------------------|-------------|
| user_id * required<br>integer<br>(query) | 4           |
| title * required<br>string<br>(query)    | title       |
| todoitem * required<br>string            | todoitem    |

**50** Click "{  
"user\_id": 1,  
"title": "shopping list",  
"todoitem": "apple",  
"method" : "ADD/DELETE" - default ADD  
}"

The screenshot shows a JSON editor interface. At the top, there is a button labeled "Copy JSON and Execute to post records to Database". Below it is a code block containing a JSON object:

```
{
 "user_id": 1,
 "title": "shopping list",
 "todoitem": "apple",
 "method" : "ADD/DELETE" - default ADD
}
```

Below the code block is a section titled "Parameters" with a green header bar. It contains three entries:

| Name                                     | Description |
|------------------------------------------|-------------|
| user_id * required<br>integer<br>(query) | 4           |
| title * required<br>string<br>(query)    | title       |
| todoitem * required<br>string            | todoitem    |

**51** Press **CTRL + C**

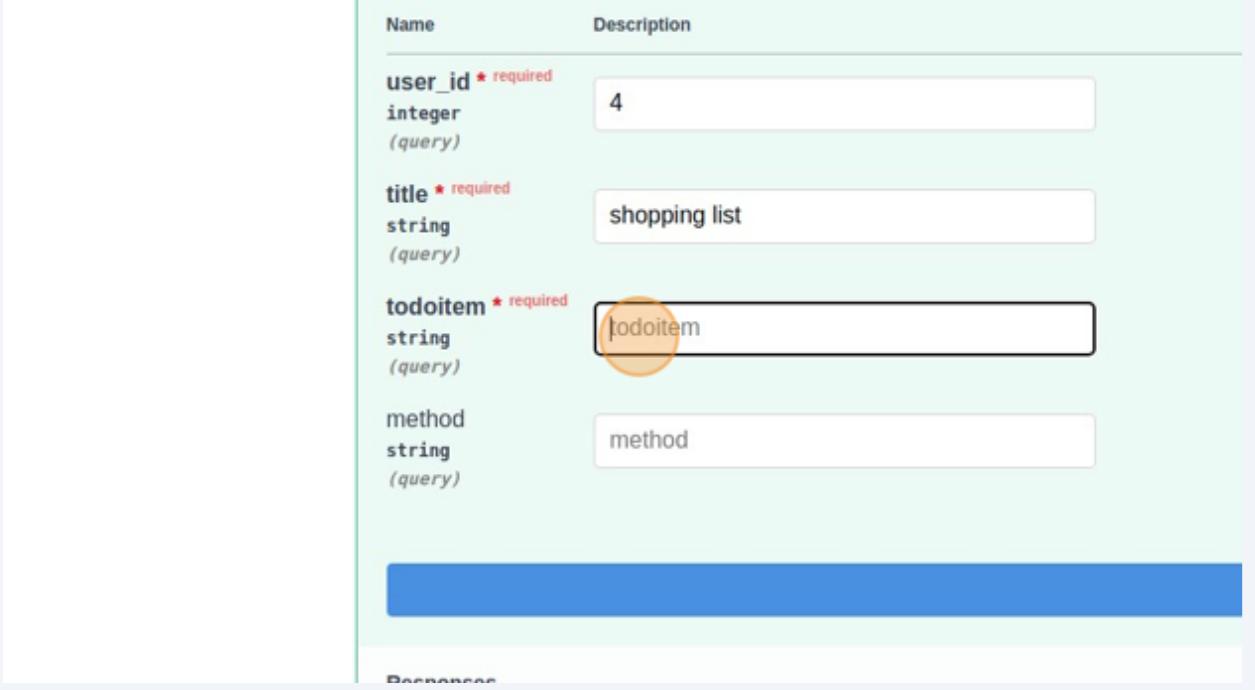
**52** Click the "title" field.

**53** Press **CTRL + V**

- 54** Click the "todoitem" field.

| Name                                     | Description   |
|------------------------------------------|---------------|
| user_id * required<br>integer<br>(query) | 4             |
| title * required<br>string<br>(query)    | shopping list |
| todoitem * required<br>string<br>(query) | todoitem      |
| method<br>string<br>(query)              | method        |

**Responses**



- 55** Type "mobile"

**56** Click "Execute"

(query)  
todoitem \* required  
string  
(query)

mobile

method  
string  
(query)

method

Responses

| Code | Description         |
|------|---------------------|
| 200  | Successful Response |

Media type

**57** Double-click the "todoitem" field.

| Name                                     | Description   |
|------------------------------------------|---------------|
| user_id * required<br>integer<br>(query) | 4             |
| title * required<br>string<br>(query)    | shopping list |
| todoitem * required<br>string<br>(query) | mobile        |
| method<br>string<br>(query)              | method        |

Execute

58 Type "car"

59 Click "Execute"

todoitem \* required  
string  
(query)  
car

method  
string  
(query)  
method

Responses

Curl

```
curl -X 'POST' \
 'https://usease.in:8181/add_remove_todo/?user_id=4&title=shopping%20list&todoitem=mobile'
 -H 'accept: application/json'
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2
 -d ''
```

- 60** Double-click the "todoitem" field.

| Name                                     | Description   |
|------------------------------------------|---------------|
| user_id * required<br>integer<br>(query) | 4             |
| title * required<br>string<br>(query)    | shopping list |
| todoitem * required<br>string<br>(query) | car           |
| method<br>string<br>(query)              | method        |

**Execute**

- 61** Type "dress"

## 62 Click "Execute"

The screenshot shows a REST API execution interface. On the left, there are two input fields: 'todoitem' (string, required) containing 'dress' and 'method' (string) containing 'method'. Below these is a large blue 'Execute' button with a yellow circle highlighting the center. To the right of the execute button is a section titled 'Responses' which contains a 'Curl' command:

```
curl -X 'POST' \
 'https://ulease.in:8181/add_remove_todo/?user_id=4&title=shopping%20list&todoitem=car' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VuZD... \
 -d ''
```

## 63 Click "Users"

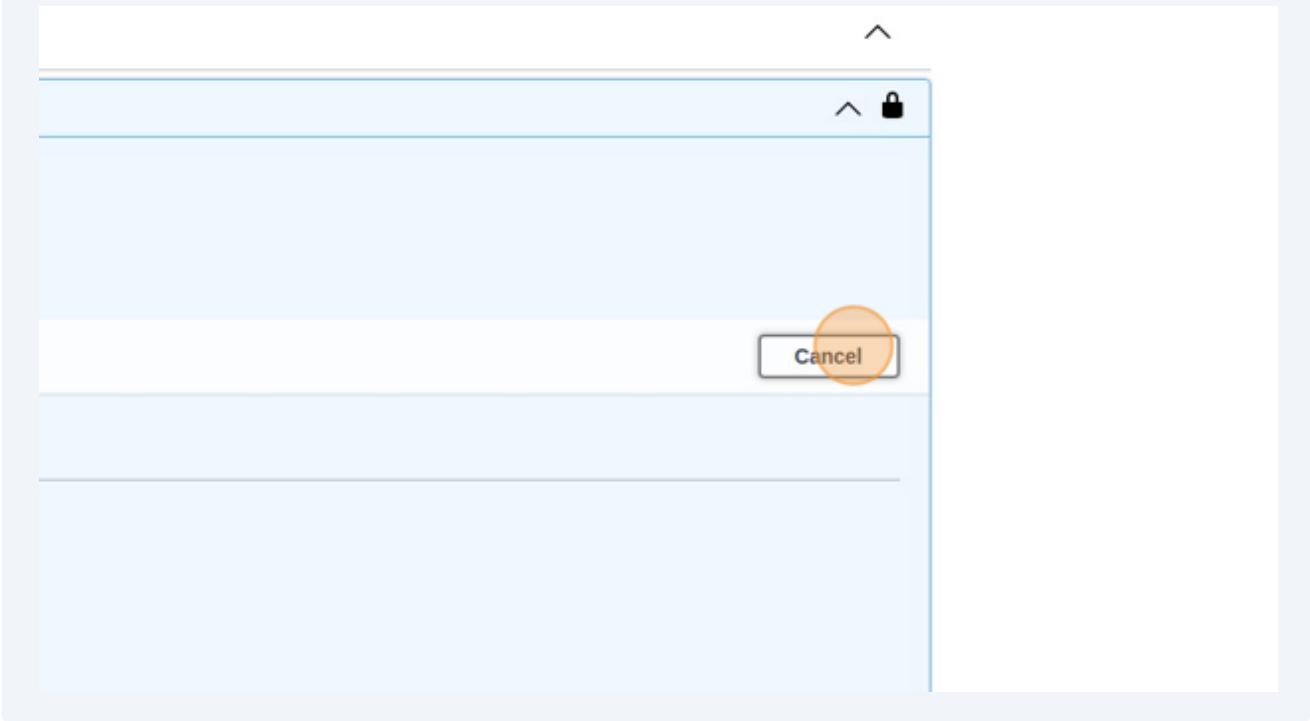
The screenshot shows a REST API interface with several sections:

- Users:** A list of operations:
  - GET /users/{record\_id} Users
  - DELETE /users/{record\_id} Users
  - PATCH /users/{record\_id} Users
- todo:** A list of operations:
  - GET /todo/ Users
  - GET /todo/{record\_id} Users
  - DELETE /todo/{record\_id} Users
  - POST /add\_remove\_todo/ Users

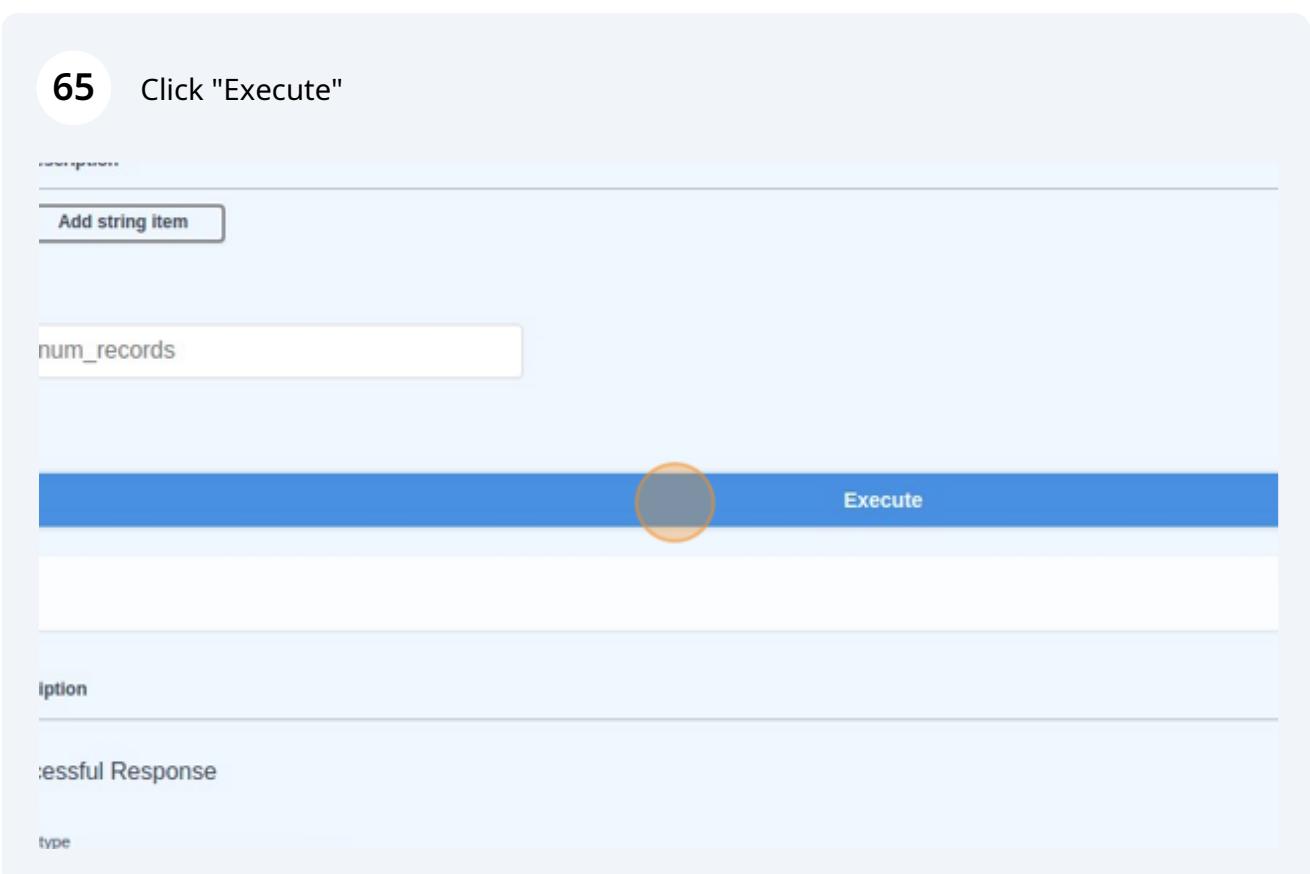
At the bottom, there is a note:

- Data is expected in json like
- Copy JSON and Execute to post records to Database

**64** Click "Try it out"



**65** Click "Execute"



66 Click the "method" field.

The screenshot shows a user interface for an API endpoint. On the left, there is a vertical list of input fields with their types and descriptions:

- integer (query) - A dropdown menu with an upward arrow icon.
- title \* required string (query) - A text input field containing "shopping list".
- todoitem \* required string (query) - A text input field containing "dress".
- method string (query) - A text input field containing "method", which is highlighted with a yellow circle.

Below these fields is a large blue button labeled "Execute". Underneath the "Execute" button is a section titled "Responses" which is currently empty.

67 Type "**CAPSLOCK CAPSLOCK DELETE**"

## 68 Click "Execute"

The screenshot shows a REST API execution interface. On the left, there are two input fields: 'todoitem' (string, required) with value 'dress' and 'method' (string) with value 'DELETE'. Below these is a large blue 'Execute' button with a yellow circle highlighting the center. To the right of the button is a section titled 'Responses' which contains a 'Curl' command:

```
curl -X 'POST' \
 'https://ulease.in:8181/add_remove_todo/?user_id=4&title=shopping%20list&todoitem=dress'
 -H 'accept: application/json'
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VlIiwibmJmIjoxNjEwOTMxODQyfQ.eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VlIiwibmJmIjoxNjEwOTMxODQyfQ'
 -d ''
```

## 69 Click "Execute"

The screenshot shows a REST API execution interface. On the left, there are two input fields: 'array[string]' (query) with an empty value and 'num\_records' (integer) with value 'num\_records'. Below these is a large blue 'Execute' button with a yellow circle highlighting the center. To the right of the button is a section titled 'Responses' which contains a 'Curl' command:

```
curl -X 'GET' \
 'https://ulease.in:8181/todo/' \
 -H 'accept: application/json'
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VlIiwibmJmIjoxNjEwOTMxODQyfQ.eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VlIiwibmJmIjoxNjEwOTMxODQyfQ'
```

Below the 'Responses' section is a 'Request URL' field containing the value 'https://ulease.in:8181/todo/'.

**70** Double-click the "todoitem" field.

| Name                                     | Description   |
|------------------------------------------|---------------|
| user_id * required<br>integer<br>(query) | 4             |
| title * required<br>string<br>(query)    | shopping list |
| todoitem * required<br>string<br>(query) | dress         |
| method<br>string<br>(query)              | DELETE        |

**Execute**

**71** Type "car"

72

Click "Execute"

todoitem \* required  
string  
(query)  
value: car

method  
string  
(query)  
value: DELETE

Execute

Responses

Curl

```
curl -X 'POST' \
 'https://ulease.in:8181/add_remove_todo/?user_id=4&title=shopping%20list&todoitem=dress&method=DELETE' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VuZXJhdGVkIGZv \
 -d ''
```

73

Click "Execute"

fields  
array[string]  
(query)  
value: Add string item

num\_records  
integer  
(query)  
value: num\_records

Execute

Responses

Curl

```
curl -X 'GET' \
 'https://ulease.in:8181/todo/' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VuZXJhdGVkIGZv \
 -d ''
```

**74** Double-click the "todoitem" field.

| Name                                     | Description   |
|------------------------------------------|---------------|
| user_id * required<br>integer<br>(query) | 4             |
| title * required<br>string<br>(query)    | shopping list |
| todoitem * required<br>string<br>(query) | car           |
| method<br>string<br>(query)              | DELETE        |

**Execute**

**75** Type "mobile"

## 76 Click "Execute"

(query)

todoitem \* required  
string  
(query)

mobile

method  
string  
(query)

DELETE

Execute

Responses

Curl

```
curl -X 'POST' \
 'https://ulease.in:8181/add_remove_todo/?user_id=4&title=shopping%20list&todoitem=car&method=DELETE' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VlIiwidG9rZW4gZ2VlIjoiMjAxMCIsInNjb3BlcyI6WyJhcGlfcmVhZCJdfQ.' \
 -d ''
```

## 77 Click "Execute"

fields  
array[string]  
(query)

Add string item

num\_records  
integer  
(query)

num\_records

Execute

Responses

Curl

```
curl -X 'GET' \
 'https://ulease.in:8181/todo/' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VuZXJhdGUiLCJpc3MiOiJodHRwOi8vYXVuseS5pby84MTgxL2Rvd24vIiwic3ViIjoiNjQyNjEwOTkifQ.'
```

**78** Click "/edit\_todo\_title/{record\_id}"

The screenshot shows a REST API endpoint for editing a todo title. The URL is `PATCH /edit_todo_title/{record_id}`. The response schema is displayed as JSON:

```
{ "detail": [{ "loc": ["string"], "msg": "string", "type": "string" }] }
```

Below the URL, there are two buttons: `code` (highlighted with an orange circle) and `GET /download_code/ Files`. There is also a section labeled `Schemas`.

**79** Click "Try it out"

The screenshot shows the "Try it out" interface for the API endpoint. At the bottom, there is a dropdown menu set to `application/json`. In the center, there is a button labeled `Try it out`, which is highlighted with an orange circle.

- 80** Click the "record\_id" field.

The screenshot shows a user interface for defining API endpoints. At the top, there is a code editor containing the following JSON:

```
{ "title": "fruit" }
```

Below the code editor is a table titled "Parameters". The table has two columns: "Name" and "Description". There is one row in the table:

| Name                                           | Description |
|------------------------------------------------|-------------|
| record_id <small>* required<br/>(path)</small> | record_id   |

The "record\_id" field in the "Description" column is highlighted with a yellow circle, indicating it is the target for the click action.

At the bottom of the interface, there is a section titled "Request body" with the value "{}".

- 81** Type "4"

82

Click "{  
"title": "fruit"  
}"

The screenshot shows a REST API endpoint for editing a todo item. The method is PATCH, the URL is /edit\_todo\_title/{record\_id}, and the resource is Users. The description states: "Edit an item by ID. This endpoint gets an item with id passed the database `id`." Below the description, there is a code snippet: `{ "title": "fruit" }`. A yellow circle highlights the opening brace '{'. The parameters section shows a table with one row:

| Name                                                        | Description                    |
|-------------------------------------------------------------|--------------------------------|
| <code>record_id</code> <small>* required<br/>(path)</small> | <input type="text" value="4"/> |

83

Press **CTRL + C**

**84** Double-click this field.

| Name                           | Description                    |
|--------------------------------|--------------------------------|
| record_id * required<br>(path) | <input type="text" value="4"/> |
| Request body required          |                                |
| {}                             |                                |

**85** Press **CTRL + V**

## 86 Click "Execute"

The screenshot shows a user interface with a large, empty input field at the top. Below it is a blue horizontal button labeled "Execute". A circular highlight is placed over the center of the "Execute" button.

**Responses**

| Code | Description         |
|------|---------------------|
| 200  | Successful Response |

Media type

## 87 Double-click "9"

The screenshot shows a browser window displaying a JSON response from the URL <https://usease.in:8181/todo/>. The response is labeled "Server response".

**Code**      **Details**

| Code | Details       |
|------|---------------|
| 200  | Response body |

```
{ "success": true, "total_count": 1, "count": 1, "records": [{ "id": 9, "user_id": 4, "my_list": [], "title": "shopping list", "created_at": "2023-06-23T05:55:44.727103", "updated_at": "2023-06-23T05:55:44.727103" }] }
```

**Response headers**

```
content-length: 196
content-type: application/json
date: Fri, 23 Jun 2023 05:52:25 GMT
server: uvicorn
```

**88** Click the "record\_id" field.

The screenshot shows a configuration interface for a REST API endpoint. At the top, there is a code block containing a JSON object:

```
{
 "title": "fruit"
}
```

Below this is a section titled "Parameters". It contains a table with two columns: "Name" and "Description". There is one row for the parameter "record\_id", which is marked as required (\* required). The "Name" column shows "(path)" and the "Description" column shows a text input field containing the value "4". This input field is highlighted with a yellow circle.

Below the parameters is a section titled "Request body" (required), which also contains a JSON object:

```
{
 "title": "fruit"
}
```

**89** Press **CTRL + V**

90 Click "Execute"

The screenshot shows a user interface for executing API requests. At the bottom center is a blue button labeled "Execute". An orange circle highlights the left side of this button. Below the button, the word "Responses" is visible. To the left of the responses section, there is a "Curl" section containing a command-line script:

```
curl -X 'PATCH' \
 'https://ulease.in:8181/edit_todo_title/4' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2Vu
 -H 'Content-Type: application/json' \
 -d '{'
```

91 Click the "title" field.

The screenshot shows a "Parameters" section of a REST API configuration tool. It lists four parameters with their values:

| Name                                     | Description   |
|------------------------------------------|---------------|
| user_id * required<br>integer<br>(query) | 4             |
| title * required<br>string<br>(query)    | shopping list |
| todoitem * required<br>string<br>(query) | mobile        |
| method<br>string<br>(query)              | DELETE        |

An orange circle highlights the text "shopping list" in the "title" input field.

**92** Type "fruit"

**93** Double-click the "todoitem" field.

| Name                                     | Description |
|------------------------------------------|-------------|
| user_id * required<br>integer<br>(query) | 4           |
| title * required<br>string<br>(query)    | fruit       |
| todoitem * required<br>string<br>(query) | mobile      |
| method<br>string<br>(query)              | DELETE      |

**Execute**

**94** Type "apple"

**95** Double-click the "method" field.

The screenshot shows a user interface for a REST API endpoint. On the left, there is a vertical list of fields with their types and descriptions:

- integer (query)
- title \* required string (query) value: fruit
- todoitem \* required string (query) value: apple
- method string (query) value: DELETE

Below the fields is a large blue button labeled "Execute". Underneath the "Execute" button is a section labeled "Responses" which is currently empty.

**96** Type "**BACKSPACE**"

97

Click "Execute"

The screenshot shows a REST API execution interface. On the left, there are two input fields: 'todoitem \* required' with the value 'apple' and 'method' with the value 'method'. Below these fields is a large blue 'Execute' button, which is highlighted with a yellow circle. To the right of the button is a section labeled 'Responses'.

curl -X 'POST' \  
'https://usease.in:8181/add\_remove\_todo/?user\_id=4&title=shopping%20list&todoitem=mobile&method=DE' \  
-H 'accept: application/json' \  
-H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VuZXJhdGVd' \  
-d ''

98

Double-click the "todoitem" field.

The screenshot shows a REST API execution interface. On the left, there are four input fields: 'user\_id \* required' with the value '4', 'title \* required' with the value 'fruit', 'todoitem \* required' with the value 'apple', and 'method' with the value 'method'. The 'todoitem' field is highlighted with a yellow circle. Below these fields is a large blue 'Execute' button. To the right of the button is a section labeled 'Responses'.

99 Type "mango"

100 Click "Execute"

todoitem \* required  
string  
(query)

mango

method  
string  
(query)

method

Execute

Responses

Curl

```
curl -X 'POST' \
 'https://useless.in:8181/add_remove_todo/?user_id=4&title=fruit&todoitem=apple' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VuZXJhdGUiLCJpYXQiOjE2MjQwOTkxNjAsImV4cCI6MTYyNDA5OTE2MF0.' \
 -d ''
```

## 101 Click "Execute"

The screenshot shows a REST API interface with a POST request. The 'fields' parameter is set to 'array[string] (query)' and has an 'Add string item' button. The 'num\_records' parameter is set to 'integer (query)' and has a 'num\_records' value. A large blue 'Execute' button is at the bottom, with an orange circle highlighting it. Below the request, there's a 'Responses' section and a 'Curl' command:

```
curl -X 'GET' \
 'https://usease.in:8181/todo/' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoidG9rZW4gZ2VuZXIiLCJpYXQiOjE2MjQwOTUyNTAsImV4cCI6MTYyNDA5NTI1MCwiaWF0IjoxNjI0MDk1MjUwLCJzY29wZWQiOiJsb2dpbiIsInNjb3BlcyI6WyJhcGlfcmVhZCJdfQ.'
```

## 102 Click "Users"

The screenshot shows a list of endpoints. The first three are for 'Users': GET /users/{record\_id} (blue), DELETE /users/{record\_id} (red), and PATCH /users/{record\_id} (green). Below them is a section for 'todo': GET /todo/{record\_id} (blue). An orange circle highlights the /todo/{record\_id} endpoint. Below it is a 'Parameters' section with two items: 'fields' and 'num\_records'. The 'fields' item has a note: 'This retrieves only the field name given from all record' followed by an example('fields:user\_name'). The 'num\_records' item has a note: 'This retrieves only the number of records mentioned' followed by an example('num\_records').

103 Click "/add\_remove\_todo/"

**todo**

GET /todo/ Users

GET /todo/{record\_id} Users

DELETE /todo/{record\_id} Users

POST /add\_remove\_todo/ Users

- Data is expected in json like
- Copy JSON and Execute to post records to Database

```
{
 "user_id": 1,
 "title": "shopping list",
 "todoitem": "apple",
 "method" : "ADD/DELETE" - default ADD
}
```

104 Click "/edit\_todo\_title/{record\_id}"

GET /todo/ Users

GET /todo/{record\_id} Users

DELETE /todo/{record\_id} Users

POST /add\_remove\_todo/ Users

PATCH /edit\_todo\_title/{record\_id} Users

Edit an item by ID.

This endpoint get an item with id passed the database `id`.

```
{
 "title": "fruit"
}
```

Parameters

105 Click "Users"

GET /users/{record\_id} Users

DELETE /users/{record\_id} Users

PATCH /users/{record\_id} Users

### todo

GET /todo/ Users

GET /todo/{record\_id} Users

DELETE /todo/{record\_id} Users

POST /add\_remove\_todo/ Users

PATCH /edit\_todo\_title/{record\_id} Users

106 Click "GET  
/todo/  
Users"

GET /users/{record\_id} Users

DELETE /users/{record\_id} Users

PATCH /users/{record\_id} Users

### todo

GET /todo/ Users

#### Parameters:

- fields: This retrieves only the field name given from all record `example(fields:user_name)`
- num\_records: This retrieves only the number of records mentioned `example(num_records:1)`

#### Parameters

**107** Click "/todo/{record\_id}"

The screenshot shows a REST API documentation interface. At the top, there is a green button labeled "PATCH /users/{record\_id} Users". Below this, the word "todo" is displayed in bold. A list of methods is shown in boxes:

- GET /todo/** Users
- DELETE /todo/{record\_id}** Users (This method is highlighted with a red background and has a red circle around its button)
- POST /add\_remove\_todo/** Users
- PATCH /edit\_todo\_title/{record\_id}** Users

Below the "todo" section, the word "code" is displayed in bold. A single method is listed:

- GET /download\_code/** Files

**108** Click "Try it out"

The screenshot shows a "Try it out" interface. At the top right, there are icons for a lock and a file. The main area is divided into several horizontal sections. In the middle section, there is a button labeled "Try it out" which is highlighted with an orange circle.

109 Click the "record\_id" field.

Delete an item by ID.  
This endpoint get an item with id passed the database `id`.

**Parameters**

| Name                                                                    | Description            |
|-------------------------------------------------------------------------|------------------------|
| <code>record_id</code> <small>* required<br/>integer<br/>(path)</small> | <code>record_id</code> |

**Responses**

| Code | Description |
|------|-------------|
|------|-------------|

110 Type "9"

## 111 Click "Execute"

The screenshot shows a REST API endpoint for a todo item. The URL is `/todo/9`. The method is `GET`. The response code is `200` and the description is `Successful Response`. The media type is not specified.

**Parameters**

| Name                                                   | Description |
|--------------------------------------------------------|-------------|
| <code>record_id</code> * required<br>integer<br>(path) | 9           |

**Responses**

| Code | Description         |
|------|---------------------|
| 200  | Successful Response |

Media type:

## 112 Click "/todo/{record\_id}"

The screenshot shows a REST API endpoint for deleting a todo item by ID. The URL is `/todo/{record_id}`. The method is `DELETE`. The description is `Delete an item by ID.` It states that the endpoint gets an item with id passed the database `id`.

**Parameters**

| Name                                                   | Description |
|--------------------------------------------------------|-------------|
| <code>record_id</code> * required<br>integer<br>(path) | 9           |

### 113 Click "Users"

113 Click "Users"

**Users**

GET /users/{record\_id} Users

DELETE /users/{record\_id} Users

PATCH /users/{record\_id} Users

**todo**

GET /todo/ Users

GET /todo/{record\_id} Users

DELETE /todo/{record\_id} Users

POST /add\_remove\_todo/ Users

PATCH /edit\_todo\_title/{record\_id} Users

### 114 Click "Execute"

114 Click "Execute"

fields

array[string]  
(query)

num\_records

integer  
(query)

Execute

Responses

Curl

```
curl -X 'GET' \
 'https://usease.in:8181/todo/' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoi
```

## 115 Click "Users"

The screenshot shows a REST API documentation page for the 'todo' endpoint. At the top, there is a green button labeled 'PATCH /users/{record\_id}' with the word 'Users' next to it. Below this, the word 'todo' is displayed in bold. A blue button labeled 'GET /todo/' is shown, with the word 'Users' highlighted by a yellow circle. Below the button, the word 'Parameters:' is followed by a list of items:

- **fields**: This retrieves only the field name given from all record `example(fields:user_name)`
- **num\_records**: This retrieves only the number of records mentioned `example(num_records: 10)`

Below this, there is a section titled 'Parameters' with a table:

| Name                               | Description     |
|------------------------------------|-----------------|
| fields<br>array[string]<br>(query) | Add string item |

## 116 Click "Users"

The screenshot shows a REST API documentation page for the 'users' endpoint. At the top, there is a blue button labeled 'GET /users/' with the word 'Users' next to it. Below the button, the word 'Parameters:' is followed by a list of items:

- **fields**: This retrieves only the field name given from all record `example(fields:user_name)`
- **num\_records**: This retrieves only the number of records mentioned `example(num_records: 10)`

Below this, there is a section titled 'Parameters' with two tables:

| Name                               | Description     |
|------------------------------------|-----------------|
| fields<br>array[string]<br>(query) | Add string item |

| Name                              | Description |
|-----------------------------------|-------------|
| num_records<br>integer<br>(query) | num_records |