Todo

This API provides functionality for managing user accounts, categories, and todos. To interact with the API, you must first create a user account and log in to receive an access token. Include this token as a Bearer token in the Authorization header for all secured endpoints.

Signup/Login

This section contains API endpoints related to creating new user accounts and authenticating existing users. Upon successful login, you will be given an access token that must be included as a Bearer token in the Authorization header for all protected routes.

POST Signup

http://localhost:3000/signup

Register a new user by supplying the name, email, and password in the request body. After successful registration, the new user will be added to the database.

Body raw (json)

json

```
{
  "name": "Test User",
  "email": "testuser@example.com",
  "password": "testpassword"
}
```

POST Login

http://localhost:3000/login

Authenticate an existing user by providing their email address and password in the request body. If the login is successful, you will receive an access token. This token must be used as a Bearer token in the Authorization header for all protected routes.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
  "name": "testuser",
  "email": "test@gmail.com",
  "password": "0000"
}
```

Categories

This section includes API endpoints for managing categories. You can perform actions like retrieving all categories, getting a specific category, creating a new category, updating an existing category, and deleting a category. All category endpoints require a valid access token in the Authorization header.

GET Get All

http://localhost:3000/categories

Fetch all categories associated with the authenticated user. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

GET Get One

http://localhost:3000/categories/1

Obtain a single category associated with the authenticated user by supplying the category ID in the URL. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token <token>

POST Create New Category

http://localhost:3000/categories

Add a new category by submitting the category name in the request body. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token <token>

Body raw (json)

```json
{
  "name": "New Category"
}
```

DELETE Delete Category

http://localhost:3000/categories/3

Remove a category by providing the category ID in the URL. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token <token>

PUT Update Category

http://localhost:3000/categories/2

Modify an existing category by supplying the category ID in the URL and the updated category name in the request body. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
  "name": "Updated Category Name"
}
```

Todos

This section comprises API endpoints for managing todos. You can perform actions like retrieving all todos, getting a specific todo, creating a new todo, updating an existing todo, and deleting a todo. All todo endpoints require a valid access token in the Authorization header and at least one existing category.

GET Get All

http://localhost:3000/todos

Fetch all todos related to the authenticated user. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

GET Get One

http://localhost:3000/todos/1

Obtain a single todo related to the authenticated user by supplying the todo ID in the URL. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

POST Create New Todo

http://localhost:3000/todos

Add a new todo by submitting the todo name and a valid category ID in the request body. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
  "name": "New Todo",
  "categoryId": 1
}
```

PUT Update Todo by ID

http://localhost:3000/todos/1

Modify an existing todo by providing the todo ID in the URL and the updated todo name in the request body. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
```

```
{
  "name": "Updated Todo"

}
```

PUT Update Todo by Name

http://localhost:3000/todos/

Modify an existing todo by providing the old and new todo names in the request body. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```
json

{
  "oldName": "Old Todo",
  "newName": "Updated Todo"
}
```

DELETE Delete Todo by ID

http://localhost:3000/todos/1

Remove a todo by supplying the todo ID in the URL. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

DELETE Delete Todo by Name

http://localhost:3000/todos/

Remove a todo by providing the todo name in the request body. A valid access token must be present in the Authorization header.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
  "name": "Todo Item"
}
```