Notely

notes

POST create Note

http://localhost:4000/api/v1/notes/create

Create a Note

This endpoint allows you to create a new note.

Request Body

- `title` (string, required): The title of the note.
- `content` (string, required): The content of the note.

Response

- `status` (string): The status of the request.
- `note`
  - `title` (string): The title of the created note.
  - `content` (string): The content of the created note.
  - `createdAt` (string): The timestamp when the note was created.
  - `modifiedAt` (string): The timestamp when the note was last modified.

- `_id` (string): The unique identifier of the note.
- `__v` (number): Version key for the note.

Upon successful creation, the response status will be 201 (Created).

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
    "title": "docter strange",
    "content": "my favourite superhero"
}
```

GET get Note by Id

http://localhost:4000/api/v1/notes/65932e555f809c5f5ed4865c

This API endpoint makes an HTTP GET request to retrieve a specific note using the unique identifier. The request does not contain a request body. The response will have a status code of 200, and it will include the details of the note such as its unique identifier, title, content, creation timestamp, modification timestamp, and version.

GET get all notes

http://localhost:4000/api/v1/notes/user

This endpoint makes an HTTP GET request to retrieve a list of notes from the API.
The request does not include a request body.

Response

- Status: 200
- Body:

```json
{
    "status": "",
    "notes": [
      {
        "_id": "",
        "title": "",
        "content": "",
        "createdAt": "",
        "modifiedAt": "",
        "__v": 0
      }
```

```
The response includes a status field and an array of notes, wher
```

PATCH update note

http://localhost:4000/api/v1/notes/65930365ba87f7f45ddc74ba

This endpoint allows you to update a specific note using the note ID.

Request

The HTTP PATCH request should be made to `http://localhost:4000/api/v1/notes/6592938ac572e28f3a2a0272`.

The request body should be of raw type and include the following payload:

```
 json

{
    "title": ""
}
### Response

Upon a successful execution (Status: 200), the response will inc

``` json
{
    "status": "",
    "note": {
```

Body raw (json)

```
 json

{
    "content" : "Man of steel"
}
```

DELETE delete note by id

http://localhost:4000/api/v1/notes/65930365ba87f7f45ddc74ba
```

This endpoint sends an HTTP DELETE request to delete a specific note identified by its ID.

No request body is required for this endpoint.

The response will have a status code of 200, indicating a successful deletion. The response body will include the status message and the details of the deleted note, including its ID, title, content, creation timestamp, modification timestamp, and version.

Auth

POST register a user

http://localhost:4000/api/v1/users/register

This API endpoint is used to register new users. Upon successful execution, it returns a status code of 201. The request should include the user's name, email, and password in the request body.

Request Body

- name (string, required): The name of the user.
- email (string, required): The email address of the user.
- password (string, required): The password for the user's account.
- role (string): Role for the user's account

Response

- status (string): Indicates the status of the request.
- user (object): Contains the details of the registered user, including their name, email, password, role, creation timestamp, modification timestamp, unique ID, and version.

Note: The actual values for the user's details will be returned in the response, but they are not provided here for privacy reasons.

Body raw (json)

json

```
{
    "name": "gopal",
    "email": "gopal@gmail.com",
    "password": "madangopal"
}
```

POST login

http://localhost:4000/api/v1/users/login

API Request Description

This endpoint is used to log in a user. The request should be sent as an HTTP POST to `http://localhost:4000/api/v1/users/login`, with the request body containing the user's email and password in JSON format.

Request Body

- `email` (string): The email of the user.
- `password` (string): The password of the user.

Response

Upon successful execution, the server responds with a status code of 201 and a JSON object containing the status and user details.

Response Body

- `status` (string): The status of the response.
- `user` (object): An object containing the user details including user ID, name, email, role, creation timestamp, modification timestamp, and version.

Body raw (json)

```json
{
    "email": "madan@gmail.com",
    "password": "madangopal"
}
```

GET get user details

http://localhost:4000/api/v1/users/

This API endpoint is a GET request to retrieve a specific user's details by providing the user's ID in the URL path.

Request

- Endpoint: `http://localhost:4000/api/v1/users/`
- Method: `GET`

Response

- Status: 200 OK
- Body:

```json
{
  "status": "",
  "user": {
    "_id": "",
    "name": "",
    "email": "",
    "role": "",
    "createdAt": "",
    "modifiedAt": "",
    "__v": 0
  }
```

Example

```
curl -X GET "http://localhost:4000/api/v1/users/123456"
```

This endpoint returns the details of a specific user identified by the provided user ID.

GET logout user

http://localhost:4000/api/v1/users/logout

This API endpoint is used to log out a user. When this request is made, the user's session will be terminated.

The request does not require any payload in the request body.

Response

- Status: 200
- Body:

```
Plain Text

{
      "status": "",
      "description": ""
  }
```

PATCH update user

http://localhost:4000/api/v1/users/

This endpoint makes an HTTP PATCH request to update the role of a specific user. The `userId` path parameter should be replaced with the unique identifier of the user. The request should include a JSON payload in the raw request body type with the `role` field to specify the new role for the user.

Request Body

- `role` : (string) The new role for the user.

Response

- Status: 200
- `status` : (string) A message indicating the status of the update.
- `description` : (string) Additional details about the update.
- `updatedUser` : An object containing the updated user details including `_id` , `name` , `email` , `role` , `createdAt` , `modifiedAt` , and `__v` .

Body raw (json)

```json
{
    "role": "ADMIN"
}
```

DELETE delete user

http://localhost:4000/api/v1/users/

Delete User

This endpoint is used to delete a specific user by their ID.

Request

- Method: DELETE
- URL: `http://localhost:4000/api/v1/users/`

Response

- Status: 200
- Body:

```json
{
    "success": true,
    "description": "",
    "deletedUser": {
```

```
"_id": "",
"name": "",
"email": "",
"role": "",
"createdAt": "",
"modifiedAt": "",
"__v": 0
```


admin

GET get All users

http://localhost:4000/api/v1/admin/users

This endpoint makes an HTTP GET request to retrieve a list of users. The request does not include a request body. The response will have a status code of 200, and will include an array of user objects, each containing the user's ID, name, email, role, creation date, modification date, and version.

GET get all notes

http://localhost:4000/api/v1/admin/notes

This endpoint makes an HTTP GET request to retrieve a list of notes . The request does not include a request body. The response will have a status code of 200, and it will contain an array of "notes" with each note object including properties such as "_id", "title", "content", "author", "createdAt", "modifiedAt", and "__v". The values for these properties will be specific to each note.

GET get note detail

http://localhost:4000/api/v1/admin/notes/

This endpoint makes an HTTP GET request to retrieve a specific note for the admin. The note ID is included in the URL as a path parameter.

The request does not include a request body.

Response

- Status: 200
- Body:

```json
{
    "status": "",
    "note": {
        "_id": "",
        "title": "",
        "content": "",
        "author": "",
        "createdAt": "",
        "modifiedAt": "",
        "__v": 0
    }
```

The response returns the status of the request and the details of the specific note, including its ID, title, content, author, creation date, modification date, and version.

GET get user detail

http://localhost:4000/api/v1/admin/users/

This API endpoint sends an HTTP GET request to retrieve the details of a specific user with the provided user ID.

The request does not contain a request body.

Response

- Status: 200
- Body:

```json
{
    "status": "",
    "user": {
        "_id": "",
        "name": "",
        "email": "",
        "role": "",
        "createdAt": "",
        "modifiedAt": "",
        "__v": 0
    }
```

PUT admin update user

http://localhost:4000/api/v1/admin/users/

This endpoint allows administrators to update a specific user by their ID. The HTTP PUT request should be made to the following URL: `http://localhost:4000/api/v1/admin/users/6593f081190b1705cc828aad`.

The request should include a raw JSON payload in the request body with the updated user details, including the `name` and `email` fields.

Upon a successful execution, the API will return a status code of 200 along with a JSON response containing the updated user's information, including their `_id`, `name`, `email`, `role`, `createdAt`, `modifiedAt`, and `__v` fields.

Please note that the `status` and `description` fields may also be included in the response, providing additional information about the update process.

Body raw (json)

```json
{
    "name" : "madan",
    "email" : "madan@gmail.com"
}
```

PUT admin update note

http://localhost:4000/api/v1/admin/notes/

Update Note

This endpoint allows the admin to update a specific note.

Request

- Method: PUT
- URL: `http://localhost:4000/api/v1/admin/notes/6593f0cb190b1705cc828ab0`
- Body (raw, JSON):

```json
{
    "title": ""
}
```

Response

- Status: 200
- Body:

```json
{
    "status": "",
    "description": "",
    "note": {
        "_id": "",
        "title": "",
        "content": "",
        "author": "",
        "createdAt": "",
        "modifiedAt": "",
        "__v": 0
```

Body raw (json)

```json
{
    "content" : "one of the most loved superhero of all time"
}
```

DELETE admin delete note

http://localhost:4000/api/v1/admin/notes/

This HTTP DELETE request is used to delete a specific note identified by the noteId in the URL. The request does not contain a request body.

The response to the request has a status code of 200, indicating a successful deletion. The response body includes a JSON object with the status and details of the deleted note, including its ID, title, content, author, creation and modification timestamps, and a version field.

DELETE admin delete user

http://localhost:4000/api/v1/admin/users/6592c578d952aa9847604773

This endpoint sends an HTTP DELETE request to delete a specific user with the provided user ID.

No request body is required for this request.

Response

- Status: 200
- Body:

```json
{
      "success": true,
      "description": "",
      "deletedUser": {
          "_id": "",
```

```
            "name": "",
            "email": "",
            "role": "",
            "createdAt": "",
            "modifiedAt": "",
```