pp1_backend

collection for testing the backend of scriptorium

AUTHORIZATION Bearer Token

user

user endpoints

- login
- signup
- edit profile of a user
- get the profile of a specific user
- refresh access token

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection pp1_backend

POST signup

http://localhost:3000/api/users/signup

API Request Description

This endpoint allows users to sign up by providing their user details including username, first name, last name, avatar URL, email, password, role, and phone number.

- `userName` (string): The username of the user.
- `firstName` (string): The first name of the user.
- `lastName` (string): The last name of the user.
- `avatar` (string): The URL of the user's avatar.
- `email` (string): The email address of the user.
- `password` (string): The password for the user account.
- `role` (string): The role of the user (e.g., USER, ADMIN).
- `phoneNumber` (string): The phone number of the user.

API Response

The response of this request is in JSON format and includes the following schema:

```json
{
    "user": {
        "id": 0,
        "firstName": "",
        "lastName": "",
        "userName": "",
        "email": "",
        "phoneNumber": "",
        "role": ""
    }
}
```

The `user` object in the response contains the user details with the following fields:

- `id` (number): The unique identifier of the user.
- `firstName` (string): The first name of the user.
- `lastName` (string): The last name of the user.
- `userName` (string): The username of the user.
- `email` (string): The email address of the user.
- `phoneNumber` (string): The phone number of the user.
- `role` (string): The role of the user.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection pp1_backend

HEADERS

Content-Type	application/json

Body raw (json)

```json
{
    "userName": "user2",
    "firstName": "fname",
    "lastName": "lastname",
    "avatar": "www.icon.com",
    "email": "user2@example.com",
```

```
    "password": "johndoe123",
    "role": "USER", // USER or ADMIN

    "phoneNumber": "1111111112"
}
```

POST login

http://localhost:3000/api/users/login

The `POST /api/users/login` endpoint is used to authenticate a user and obtain access and refresh tokens.

Login is needed to perform USER and ADMIN level actions in the application.

Request Body

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

Response

The response for this request is a JSON object with the following schema:

```
json

{
    "accessToken": {
      "type": "string"
    },
    "refreshToken": {
      "type": "string"
}
```

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection pp1_backend

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "userName": "user2",
    "password": "johndoe123"
}
```

GET get-profile-of-user

http://localhost:3000/api/users/3

The endpoint retrieves user data for a single user. Please pass in the id of the user in the url (api/users/{id}).

Users cannot see hidden posts here, even if the user who the user is serching is the same as the user logged in.

The response returned is in JSON format and has the following schema:

json

```
{
    "type": "object",
    "properties": {
        "id": {
            "type": "number"
        },
        "firstName": {
            "type": "string"
        },
        "lastName": {
            "type": "string"
```

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

POST edit-profile-of-user

http://localhost:3000/api/users/3

The endpoint allows users to perform editing of a user's profile who owns this account by passing in a user id in the request quary parameter, and the data to send in the request body. The allowed methods are POST and GET. The payload

should include the user's details such as name, email, and role. An example request and response will be added to illustrate the usage of this endpoint.

Example Request

```json
{
  "firstName": "whatever",
  "lastName": "whatevr",
  "userName": null,
  "phoneNumber": "0000000110",
  "avatar": "www.newicon.com",
  "role": null,
  "password": null,
  "email": null
}
```

Example Response

- Status: 200
- {"message": "","updatedProfile": { "id": 0, "firstName": "", "lastName": "", "userName": "", "avatar": "", "email": "", "phoneNumber": "", "password": "", "role": ""}}

Payload

The request should have a raw body with the following parameters:

- firstName (text): The first name of the user.
- lastName (text): The last name of the user.
- phoneNumber (text): The phone number of the user.
- avatar (text): The URL of the user's avatar image.
- role (text, optional):

Example

```json
{
    "firstName": "whatever",
    "lastName": "whatevr",
    "phoneNumber": "0000000110",
    "avatar": "www.newicon.com"
}
```

Response

Upon successful execution, the endpoint returns a JSON object with a status code of 200 and the following structure:

```json
{
    "message": "",
    "updatedProfile": {
        "id": 0,
        "firstName": "",
        "lastName": "",
        "userName": "",
        "avatar": "",
        "email": "",
        "phoneNumber": "",
        "password": "",
```

The request body should contain the user's details such as first name, last name, phone number, avatar URL, role, username, password, and email.

The parameters for this endpoint include the ID of the user in the URL path and the user details in the request body. The user details are optional, and only the fields that need to be updated or created should be included in the request body.

This endpoint allows you to update your user information.

Request Body

- `firstName` (text, optional): The first name of the user.
- `lastName` (text, optional): The last name of the user.
- `phoneNumber` (text, optional): The phone number of the user.
- `avatar` (text, optional): The URL of the user's avatar image.
- `role` (text, optional): The role of the user.
- `userName` (text, optional): The username of the user.
- `password` (text, optional): The password of the user.
- `email` (text, optional): The email address of the user.

Response

The response is in JSON format with the following schema:

```json
{
  "firstName": ""
}
```

AUTHORIZATION Bearer Token

Token

Body raw (json)

```json
{
    "firstName": "whatever",
    "lastName": "whatevr",
    "userName": null,
    "phoneNumber": "0000000110",
    "avatar": "www.newicon.com",
    "role": "ADMIN",
    "password": null,
    "email": null
}
```

POST refresh

http://localhost:3000/api/users/refresh

Refresh User Access Token

This endpoint is used to refresh the user's access token by providing the refresh token.

Request Body

- refreshToken (string, required): The refresh token used to generate a new access token.

Example:

```json
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6Ik ..."
}
```

Response

The response is in JSON format and includes the new access token.

Example:

```json
{
  "accessToken": ""
}
```

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

Body raw (json)

```
json

{
        "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NCwidXNlck5hbWUiOiJ1c2VyN
}
```

blog

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection pp1_backend

POST create a blog

http://localhost:3000/api/blogs

This endpoint allows you to create a new blog by sending an HTTP POST request to the specified URL. The request should include a JSON payload in the raw request body with the following parameters:

- `description` (string): The description of the blog.
- `tags` (string): Tags associated with the blog.
- `title` (string): The title of the blog.
- `templateIds` (array): IDs of templates related to the blog.

Upon successful creation, the response will include the details of the newly created blog.

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

Body raw (json)

```json
{
    "description": "blog 6 description",
    "tags": "arrays, hashmaps",
    "title": "blog 6",
    "templateIds": [3, 2] // add these template with given id's to this blog, could be null als
}
```

GET get list of blogs

http://localhost:3000/api/blogs?tags&description&title&templateTitle

Get Blogs

This endpoint retrieves a list of blogs based on the provided query parameters. Its going to return blogs based on filtering by the query parameters, ordered by upvotes.

Request

Query Parameters

- `tags` (optional): Filter blogs by tags.
- `description` (optional): Filter blogs by description.
- `title` (optional): Filter blogs by title.
- `templateTitle` (optional): Filter blogs by template title.

Return ordered by upvotes.

Response

The response for this request is a JSON object representing the list of blogs ordered by highest upvotes. The schema for the response can be described as follows:

json

```
{
  "type": "object",
  "properties": {
    "blogs": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {
            "type": "string"
          }
```

AUTHORIZATION Bearer Token

Token

PARAMS

tags

description

title

templateTitle

POST rate a blog

http://localhost:3000/api/blogs/1/rate

Rate Blog

This endpoint allows the user to rate a specific blog by providing an upvote or downvote.

Request Body

- `voteType` (string, required): The type of vote, which can be either "upvote" or "downvote".

Response

The response will be in JSON format and includes the following schema:

json

```
{
    "voteType": "" //upvote or downvote
}
```

The last execution returned a status of 200 with the following response body:

json

```
{
    "message": "string",
    "updatedBlogPost": {
        "id": "number",
        "authorId": "number",
        "title": "string",
        "description": "string",
        "upvote": "number",
        "downvote": "number",
        "createdAt": "string",
        "updatedAt": "string",
```

This endpoint allows the user to rate a specific blog by providing an upvote or downvote.

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "voteType": "" // "upvote" or "downvote"
}
```

POST hide a blog

http://localhost:3000/api/blogs/7/hide

Hide Blog Endpoint

This endpoint allows an admin to hide a specific blog with the given ID. Hiding a blog will mark its attribute "hidden" to true, and now users won't be able to see it in get requests anymore (you can try this my providing the same blog id in the

next request and seeing it will not show).

This url accepts a blog id to hide. (api/blogs/{id}/hide)

Request

- Method: POST
- URL: `http://localhost:3000/api/blogs/{id}/hide`

Response

The response for this request will be a JSON schema with the following properties:

- `message` (string) - Provides a message regarding the operation.

Example Response:

```
json

{
  "message": "Blog with ID 1 has been successfully hidden."
   "hiddenPost": {
        "id": 1,
        "authorId": 1,
        "title": "blog 1",
        "description": "blog 1 description",
        "upvote": 1,
        "downvote": 1,
        "createdAt": "2024-11-03T06:07:00.281Z",
        "updatedAt": "2024-11-03T06:18:55.832Z",
```

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

GET **get a blog**

http://localhost:3000/api/blogs/2

Blog Details

This endpoint retrieves the details of a specific blog.

(api/blogs/{id}).

Method

GET

Payload

This endpoint does not require a request payload.

Example

Request

Plain Text

```
GET http://localhost:3000/api/blogs/2
```

Response

json

```
{
    "id": 0,
    "authorId": 0,
    "title": "",
    "description": "",
    "upvote": 0,
    "downvote": 0,
    "createdAt": "",
    "updatedAt": "",
    "tags": "",
    "hidden": true,
```

This endpoint retrieves a specific blog with the ID passed into the request parameter.

Response

The response will be in JSON format with an array of blog data. Here is a sample JSON schema for the response:

json

```
[]
```

AUTHORIZATION Bearer Token

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

POST edit a blog

http://localhost:3000/api/blogs/6

Request Description

This endpoint is used to update a specific blog with the given ID in the query parameter. The request should be sent as an HTTP POST to the specified URL with the blog ID in the endpoint. The request body should contain the updated title, description, tags, templatesIdsToAdd, and templatesIdsToRemove.

If the blog is flagged (marked as hidden: true), then no one can edit this blog.

Request Body (all fields are optional)

- title (string): The updated title of the blog.
- description (string): The updated description of the blog.
- tags (string): The updated tags for the blog.
- templatesIdsToAdd (array): IDs of templates to be added to the blog.
- templatesIdsToRemove (array): IDs of templates to be removed from the blog.

Response

The response will be in JSON format with the following schema:

```json
{
  "id": 0,
  "authorId": 0,
  "title": "",
  "description": "",
  "upvote": 0,
  "downvote": 0,
  "createdAt": "",
  "updatedAt": "",
  "tags": "",
  "hidden": true,
```

The response will include the updated blog details, including the ID, author information, title, description, upvote count,

downvote count, creation and update timestamps, tags, and hidden status.

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

Body raw (json)

```json
{
    "title": "blog 10 title change",
    "description": "blog 10 title CHANGE",
    "tags": "arrays, hashmaps",
    "templateIdsToAdd": [6], // example template id to add to this blog
    "templateIdsToRemove": [4]// example template id to remove from thsi blog
}
```

DELETE delete a blog

http://localhost:3000/api/blogs/5

DELETE /api/blogs/2

This endpoint is used to delete a specific blog with the ID of {id} at the end of the endpoint (api/blogs/{id}) inly if the user is the author of that blog.

Request

No request body is required for this request.

Response

The response for this request is a JSON object with the following schema:

```json
{
  "type": "object"
```

```
  "type": "object",
  "properties": {
    "message": {

      "type": "string"
    }
  }
}
```

The response contains a status code of 200 and a JSON object with a "message" key, which provides information about the success of the deletion operation.

AUTHORIZATION Bearer Token

Token

GET view hidden blogs of logged in user

http://localhost:3000/api/blogs/view-hidden

This endpoint makes an HTTP GET request to retrieve hidden blog posts of this user that are flagged. The response will be in JSON format and will include an array of blog posts with their respective details. Below is the JSON schema for the response.

Only posts hidden for the current logged in user will be seen here, thus fulfilling the fact that other users won't be able to see hidden posts, but only the authors will, and they wont be able to edit them.

json

```
[
    {
        "id": "number",
        "authorId": "number",
        "title": "string",
        "description": "string",
        "upvote": "number",
        "downvote": "number",
        "createdAt": "string",
        "updatedAt": "string",
        "tags": "string",
```

The response will include the blog post details such as ID, author ID, title, description, upvotes, downvotes, creation and update timestamps, tags, and a boolean flag indicating if the post is hidden.

No request body is required for this GET request.

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

POST report a blog

http://localhost:3000/api/report

POST /api/report

This endpoint is used to report a blog.

Request Body

- `description` (string) - The description of the report.
- `blogId` (number) - The ID of the blog being reported.

Response

The response will be in JSON format with the following schema:

```json
{
    "report": {
        "id": 0,
        "status": "OPEN",
        "authorId": 0,
        "description": "",
        "createdAt": "",
        "blogId": 0,
        "commentId": null
    }
}
```

The `report` object contains the details of the reported blog, including its ID, status, author ID, description, creation timestamp, blog ID, and associated comment ID (if any).

AUTHORIZATION Bearer Token

Token

Body raw (json)

json

```
{
    "description": "this blog is mean to me",
    "blogId": 7

}
```

templates

note that only authenticated users/admin can have access to creating, editing, deleting (their own) templates. Visitors can only view and use an already exisitng code template

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection pp1_backend

GET get list of templates

http://localhost:3000/api/template

GET /api/template

This endpoint retrieves a list of templates.

Request

This is a GET request to fetch the list of templates.

Response

The response will be in JSON format and will have the following schema:

json

```
{
  "templates": [
    {
      "id": "number",
      "ownerId": "number",
      "title": "string",
      "explanation": "string",
      "codeId": "number",
      "isForked": "boolean",
      "parentTemplateId": "number or null",
```

"createdAt": "string",

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection pp1_backend

HEADERS

PARAMS

POST create a template

http://localhost:3000/api/template/create

Create Template

This endpoint allows the user to create a new template by providing the title, explanation, tags, code, and language.

Request Body

- title (string, required): The title of the new template.
- explanation (string, required): A brief explanation of the new template.
- tags (string, required): Tags associated with the new template.
- code (string, required): The code content of the new template.
- language (string, required): The programming language of the code.

Response

The response will be in JSON format with the following schema:

```json
{
  "type": "object",
  "properties": {
    "template": {
      "type": "object",
      "properties": {
        "id": { "type": "integer" },
        "ownerId": { "type": "integer" },
        "title": { "type": "string" },
```

```
        "explanation": { "type": "string" },
        "codeId": { "type": "null" },
```


AUTHORIZATION Bearer Token

Token

Body raw (json)

json


```
{
    "title": "Java Calculator Template",
    "explanation": "A simple calculator that performs basic arithmetic operations.",
    "tags": "Java, Calculator",
    "code": "import java.util.Scanner;\n\npublic class Calculator {\n    public static void ma
    "language": "java",
    "input": "5.0\n3.0\n+\n"
}
```


POST fork a template as a user

http://localhost:3000/api/template/create

Create Template

This endpoint allows the user to create a new template by providing the necessary details.

Request Body

- title (text, required): The title of the template.
- explanation (text, required): A brief explanation of the template.
- tags (text, required): Tags associated with the template.
- code (text, required): The code content of the template.
- language (text, required): The programming language of the template.
- parentTemplateId (integer, required): The ID of the parent template if this template is a forked version.

Response

- Status: 201
- Content-Type: application/json
- template (object): Details of the created template including ID, owner ID, title, explanation, code ID, forked status, parent template ID, creation and update timestamps, and tags.
- message (string): Any additional message related to the response.

AUTHORIZATION Bearer Token

Token

Body raw (json)

```
json

{
    "title": "this is my forked template",
    "explanation": "trying out new stuff from forked",
    "tags": "coooool",
    "code": "print('hi')",
    "language": "python",
    "parentTemplateId": 1
}
```

DELETE delete a template

http://localhost:3000/api/template/delete/4

The API endpoint sends an HTTP DELETE request to http://localhost:3000/api/template/delete/4 to delete a specific template. Upon successful execution, the response will have a status code of 200 and a content type of application/json. The response body will be a JSON object with a "msg" key, which may contain a message or be empty.

Query Parameters

- [id] → refers to the id of the template of which the user/admin wants to delete (template/delete/[id])

```
json

{
    "type": "object",
    "properties": {
        "msg": {
            "type": "string"
        }
    }
}
```

AUTHORIZATION Bearer Token

Token

PUT edit template

http://localhost:3000/api/template/edit/7

Update Template

This endpoint is used to update a specific template with the provided ID.

Request

- Method: PUT
- Endpoint: `http://localhost:3000/api/template/edit/7`
- { "title": "Lribbonacci"}

Query Parameters

- [id] → refers to the id of the template of which the user/admin wants to make some edits to (template/edit/[id])

Response

The response is in JSON format and has the following schema:

```json
{
  "type": "object",
  "properties": {
    "id": {
      "type": "integer"
    },
    "ownerId": {
      "type": "integer"
    },
    "title": {
      "type": "string"
```

AUTHORIZATION Bearer Token

Token

Body raw (json)

```json
{
```

```
    "title": "new title2",
    "explanation": "new explanation",


    "input": "5.0 4.0 -"
}
```

code

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection pp1_backend

POST execute code

http://localhost:3000/api/code/execute

The endpoint allows you to execute the provided code in the specified programming language. The request should include the code to be executed, the programming language, and the standard input if required.

Request Body

- `code` (text): The code to be executed.
- `language` (text): The programming language of the code.
- `stdin` (text): The standard input for the code execution.

Response

The response is a JSON object with the following schema:

json

```
{
    "type": "object",
    "properties": {
        "status": {
            "type": "string"
        },
        "output": {
            "type": "string"
        }
    }
}
```

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

Body raw (json)

```json
{
  "code": "def factorial(n): return 1 if n == 0 else n * factorial(n - 1)\nnum = int(input())\r
  "language": "python",
  "stdin": ""
}
```

comments

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection pp1_backend

POST create a comment

http://localhost:3000/api/blogs/1/comments

Create a Comment on a Blog

This endpoint allows the user to create a new comment on a specific blog. Note that Blog ID is a query parameter as we are passing in the blog id of the blog of which we want to write a comment for.

Query Parameters

- blogs[id] → refers to the specific blog of which we want to access the comments of (blogs/[id]/comments)

Request Body

- content (text, required): The content of the comment.

Response

The response is a JSON object with the following properties:

- message (string): A message from the server.
- newComment (object): An object representing the newly created comment with the following properties:
  - id (number): The unique identifier of the comment.
  - authorId (number): The unique identifier of the author of the comment.
  - content (string): The content of the comment.
  - blogId (number): The unique identifier of the blog on which the comment was made.
  - upvote (number): The number of upvotes received by the comment.
  - downvote (number): The number of downvotes received by the comment.
  - createdAt (string): The timestamp indicating when the comment was created.
  - hidden (boolean): Indicates whether the comment is hidden or not.
  - parentId (number): The unique identifier of the parent comment, if the comment is a reply to another comment.

Response Schema

```json
{
  "type": "object",
  "properties": {
    "message": {
      "type": "string"
    },
    "newComment": {
      "type": "object",
      "properties": {
        "id": {
          "type": "number"
```

AUTHORIZATION Bearer Token

Token

Body raw (json)

```json
{
    "content": "gross!!!!"
}
```

POST rate a comment

http://localhost:3000/api/blogs/1/comments/1/rate

Rate Blog Comment

This endpoint allows the user to rate a specific comment on a blog.

Request Body

- voteAction (string, required): The action to be performed on the comment, e.g., "upvote" or "downvote".

Query Parameters

- blogs[id] → refers to the id of the blog of which we want to access the comments too (blogs/[id])
- comments[id] → refers to the specific comment associated with the blog (blogs/[id]/comments[id])

Response

The response will be in JSON format with the following schema:

```json
{
    "type": "object",
    "properties": {
        "message": {
            "type": "string"
        },
        "votedComment": {
            "type": "object",
            "properties": {
                "id": {
                    "type": "integer"
```

AUTHORIZATION Bearer Token

Token

Body raw (json)

```json
{
    "voteAction": "upvote"
}
```

POST Create an admin account

POST Create an admin account

http://localhost:3000/api/users/signup

The endpoint `POST /api/users/signup` is used to sign up a new user with the provided details. The request should include the user's first name, last name, username, password, role, email, avatar, and phone number in the request body.

Request Body

- `firstName` (string): The first name of the user.
- `lastName` (string): The last name of the user.
- `userName` (string): The username chosen by the user.
- `password` (string): The password for the user account.
- `role` (string): The role of the user (e.g., ADMIN, USER, etc.).
- `email` (string): The email address of the user.
- `avatar` (string): The avatar URL or image for the user.
- `phoneNumber` (string): The phone number of the user.

Response (JSON Schema)

The response of this request follows the JSON schema below:

```json
{
  "type": "object",
  "properties": {
    "userId": {
      "type": "string"
    },
    "message": {
      "type": "string"
    }
  }
}
```

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection pp1_backend

Body raw (json)

```json
{
    "firstName": "ADMINNNNNNNNNN",
```

```
    "lastName": "ADMINNNNNN",
    "userName": "IM DA ADMIN",
    "password": "admin123",
    "role": "ADMIN",
    "email": "www.hello@wuwu.com",
    "avatar": "shshshshs",
    "phoneNumber": "6475555555"
}
```

POST JWT Token for Admin

http://localhost:3000/api/users/login

Login User

This endpoint allows users to log in by providing their username and password.

Request Body

- `userName` (string) - The username of the user.
- `password` (string) - The password of the user.

Response

The response is a JSON object with the following properties:

- `userId` (string) - The unique identifier of the user.
- `token` (string) - The authentication token for the user session.

json

```
{
    "type": "object",
    "properties": {
        "userId": {
            "type": "string"
        },
        "token": {
            "type": "string"
        }
    }
}
```

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection pp1_backend

HEADERS

Body raw (json)

```json
{
    "userName": "IM DA ADMIN",
    "password": "admin123"
}
```

POST Flag a comment as an admin

http://localhost:3000/api/report

This endpoint allows you to create a report for a comment by making an HTTP POST request to http://localhost:3001/api/report. The request should include a JSON payload in the raw request body type with the keys "description" and "commentId".

Request Body

- `description` : (string) A brief description of the report.
- `commentId` : (number) The ID of the comment being reported.

Response

Upon a successful execution, the endpoint returns a status code of 200 and a JSON response with the following structure:

```json
{
    "report": {
        "id": 0,
        "status": "",
        "authorId": 0,
        "description": "",
        "createdAt": "",
        "blogId": null,
        "commentId": 0
    }
}
```

The response contains the details of the created report, including the report ID, status, author ID, description, creation timestamp, blog ID (if applicable), and the ID of the reported comment.

AUTHORIZATION Bearer Token

Token

Body raw (json)

```
json

{
    "description": "comment mean",
    "commentId": 4
}
```

PUT Hiding comment as an Admin

http://localhost:3000/api/blogs/1/comments/1/hide

The endpoint allows the user to hide a comment on a specific blog post.

Query Parameters

- blogs[id] → refers to the specific blog of which we want to access the comments of (blogs/[id])
- comments[id] → refers to the specific comment of which we want to hide (blogs/[id]/comments/[id]/hide)

Response

The response of the request is a JSON object with the following schema:

```
json

{
    "type": "object",
    "properties": {
        "message": {
            "type": "string"
        },
        "flaggedComment": {
            "type": "object",
            "properties": {
                "id": {
                    "type": "integer"
```

AUTHORIZATION Bearer Token

Token

GET Viewing flagged comment as the author only

http://localhost:3000/api/blogs/1/comments/4/view-hidden

Get Hidden Comment

This endpoint retrieves a hidden comment associated with a specific blog.

Request Body

This request does not require a request body.

Query Parameters

- blogs[id] → refers to the specific blog of which we want to access the comments of (blogs/[id])
- comments[id] → refers to the specific comment of which we want to show the author that its hidden (not the comment is only visible to the author) (blogs/[id]/comments/[id]/view-hidden)

Response

- `message` (string) - A message related to the comment retrieval.
- `comment` (object)
  - `id` (number) - The unique identifier of the comment.
  - `authorId` (number) - The unique identifier of the comment author.
  - `content` (string) - The content of the comment.
  - `blogId` (number) - The unique identifier of the associated blog.
  - `upvote` (number) - The number of upvotes for the comment.
  - `downvote` (number) - The number of downvotes for the comment.
  - `createdAt` (string) - The timestamp of the comment creation.
  - `hidden` (boolean) - Indicates whether the comment is hidden.
  - `parentId` (number or null) - The unique identifier of the parent comment, if applicable.
  - `abuseReports` (array) - An array of abuse reports associated with the comment.
    - `id` (number) - The unique identifier of the abuse report.
    - `status` (string) - The status of the abuse report.
    - `authorId` (number) - The unique identifier of the author of the abuse report.
    - `description` (string) - The description of the abuse report.
    - `createdAt` (string) - The timestamp of the abuse report creation.
    - `blogId` (number or null) - The unique identifier of the associated blog, if applicable.
    - `commentId` (number) - The unique identifier of the associated comment.

AUTHORIZATION Bearer Token

Token

Abuse Report

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection pp1_backend

POST Create Abuse Report

http://localhost:3000/api/report

Report API

This API endpoint allows users to report offensive content in a blog or a comment.

Request Body

- `description` (string, required): The description of the offensive content.
- `blogId` (number, optional): The ID of the blog being reported.
- commentId (number, optional): The ID of the comment being reported

Atleast one of blogId or commentId have to be passed

Response

The response is in JSON format and follows the schema below:

json

```
{
  "report": {
    "id": 0,
    "status": "",
    "authorId": 0,
    "description": "",
    "createdAt": "",
    "blogId": 0,
    "commentId": null
  }
}
```

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

PARAMS

Body raw (json)

```
json

{
    "description": "This blog contains offensive language",
    "blogId": 3
}
```

DELETE Delete an abuse report (ADMIN only)

http://localhost:3000/api/report/delete

Delete Report

This endpoint is used to delete an abuse report. Available to ADMIN users only.

Request

- Method: DELETE
- URL: `http://localhost:3000/api/report/delete`
- Body:
  - id (number, required): The ID of the report to be deleted.

Response

The response is a JSON object with the following schema:

```
json
```

```
{
  "type": "object",
  "properties": {
    "message": {
      "type": "string"
    }
  }
}
```

The response contains a status code of 200 and a JSON object with a `message` key.

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "id": 1
}
```

GET **view (admin only)**

http://localhost:3000/api/report/view_admin?page=1&limit=1

Report View Admin

This endpoint makes an HTTP GET request to retrieve a specific page of admin view for blog and comment reports.

Request

Query Parameters

- `page` (integer) - The page number to retrieve.
- `limit` (integer) - The maximum number of items to retrieve per page.

Response

Upon a successful request, the server responds with a status code of 200 and a JSON object containing the following fields:

Blogs (array)

- `id` (integer) - The unique identifier of the blog.
- `authorId` (integer) - The ID of the blog author.
- `title` (string) - The title of the blog.
- `description` (string) - The description of the blog.
- `upvote` (integer) - The number of upvotes for the blog.
- `downvote` (integer) - The number of downvotes for the blog.
- `createdAt` (string) - The date and time of blog creation.
- `updatedAt` (string) - The date and time of blog update.
- `tags` (string) - The tags associated with the blog.
- `hidden` (boolean) - Indicates if the blog is hidden.
- `abuseReports` (array) - An array of abuse reports for the blog, each containing:
  - `id` (integer) - The unique identifier of the abuse report.
  - `status` (string) - The status of the abuse report.
  - `authorId` (integer) - The ID of the author of the abuse report.
  - `description` (string) - The description of the abuse report.
  - `createdAt` (string) - The date and time of the abuse report creation.
  - `blogId` (integer) - The ID of the blog associated with the abuse report.
  - `commentId` (null) - The ID of the comment associated with the abuse report (null for blog reports).

Comments (array)

- `id` (integer) - The unique identifier of the comment.
- `authorId` (integer) - The ID of the comment author.
- `content` (string) - The content of the comment.
- `blogId` (integer) - The ID of the blog associated with the comment.
- `upvote` (integer) - The number of upvotes for the comment.
- `downvote` (integer) - The number of downvotes for the comment.
- `createdAt` (string) - The date and time of comment creation.
- `hidden` (boolean) - Indicates if the comment is hidden.
- `parentId` (null) - The ID of the parent comment (null for top-level comments).
- `abuseReports` (array) - An array of abuse reports for the comment, each containing:
  - `id` (integer) - The unique identifier of the abuse report.
  - `status` (string) - The status of the abuse report.
  - `authorId` (integer) - The ID of the author of the abuse report.
  - `description` (string) - The description of the abuse report.
  - `createdAt` (string) - The date and time of the abuse report creation.
  - `blogId` (null) - The ID of the blog associated with the abuse report (null for comment reports).

- `blogId` (null) - The ID of the blog associated with the abuse report (null for comment reports).
- `commentId` (integer) - The ID of the comment associated with the abuse report.

This endpoint retrieves a paginated view of the blog posts and comments sorted by number of abuse reports. This feature is only available to admin users.

Request

- Method: GET
- URL: `http://localhost:3000/api/report/view_admin`
- Query Parameters:
  - `page` (number) - The page number for paginated results.
  - `limit` (number) - The maximum number of items to return per page.

Response

The response is in JSON format and follows the schema below:

```json
{
  "blogs": [
    {
      "id": "number",
      "authorId": "number",
      "title": "string",
      "description": "string",
      "upvote": "number",
      "downvote": "number",
      "createdAt": "string",
      "updatedAt": "string",
```

AUTHORIZATION Bearer Token

Token

HEADERS

Content-Type	application/json

PARAMS

page	1
limit	1