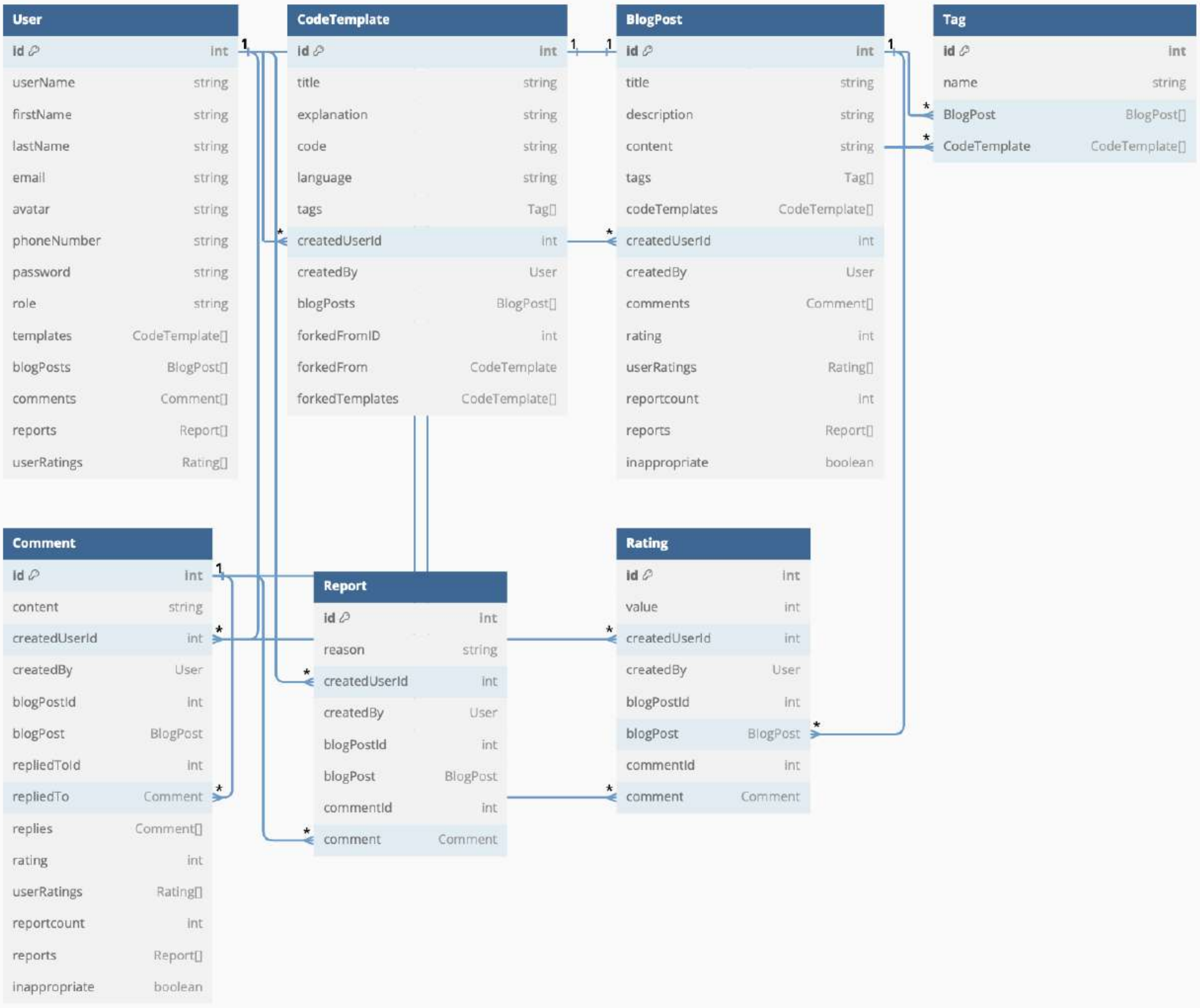

User	
id	int
userName	string
firstName	string
lastName	string
email	string
avatar	string
phoneNumber	string
password	string
role	string
templates	CodeTemplate[]
blogPosts	BlogPost[]
comments	Comment[]
reports	Report[]
userRatings	Rating[]

CodeTemplate	
id	int
title	string
explanation	string
code	string
language	string
tags	Tag[]
createdUserId	int
createdBy	User
blogPosts	BlogPost[]
forkedFromID	int
forkedFrom	CodeTemplate
forkedTemplates	CodeTemplate[]

BlogPost	
id	int
title	string
description	string
content	string
tags	Tag[]
codeTemplates	CodeTemplate[]
createdUserId	int
createdBy	User
comments	Comment[]
rating	int
userRatings	Rating[]
reportcount	int
reports	Report[]
inappropriate	boolean

Tag	
id	int
name	string
BlogPost	BlogPost[]
CodeTemplate	CodeTemplate[]

Comment	
id	int
content	string
createdUserId	int
createdBy	User
blogPostId	int
blogPost	BlogPost
repliedToId	int
repliedTo	Comment
replies	Comment[]
rating	int
userRatings	Rating[]
reportcount	int
reports	Report[]
inappropriate	boolean

Report	
id	int
reason	string
createdUserId	int
createdBy	User
blogPostId	int
blogPost	BlogPost
commentId	int
comment	Comment

Rating	
id	int
value	int
createdUserId	int
createdBy	User
blogPostId	int
blogPost	BlogPost
commentId	int
comment	Comment

To access a cleaner version of the docs, please head over to this link:

https://documenter.getpostman.com/view/39140364/2sAY4x9ggL

Authentication

This folder contains example requests to sign up and login as an admin or normal user.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

POST Admin sign up

http://localhost:3000/api/auth/signup/admin

This endpoint is used to sign up a new admin user and create their profile. The request should be sent as an HTTP POST to the specified URL.

Request Body

- `email` (string): The email address of the user.
- `userName` (string): The username of the user.
- `firstName` (string): The first name of the user.
- `lastName` (string): The last name of the user.
- `phoneNumber` (string): The phone number of the user.
- `password` (string): The password for the user
- `adminKey` (string): The secret admin key

Response

The response to this request is a basic acceptance/rejection message.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "email": "alberrt@gmail.com",
    "userName": "albert",
    "firstName": "albert",
    "lastName": "hyunh",
    "phoneNumber": "+1234567890",
    "password": "ilike309",
    "adminKey": "123"
}
```

POST User sign up

http://localhost:3000/api/auth/signup/user

This endpoint is used to sign up a new user and create their profile. The request should be sent as an HTTP POST to the specified URL.

Request Body

- `email` (string): The email address of the user.
- `userName` (string): The username of the user.
- `firstName` (string): The first name of the user.
- `lastName` (string): The last name of the user.
- `phoneNumber` (string): The phone number of the user.
- `password` (string): The password for the user

Response

The response to this request is a basic acceptance/rejection message.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "email": "albert@gmail.com",
    "userName": "albyh420",
    "firstName": "albert",
    "lastName": "hyunh"
    "phoneNumber": "+1234567890",


    "password": "ilike309"
}
```

POST User login

http://localhost:3000/api/auth/login

This API endpoint is used to authenticate and login a user or admin through a HTTP POST request.

Request Body

- One of the following:
  - `email` (string): The email address of the user.
  - `userName` (string): The email address of the user.
- `password` (string) - The password for the user's account.

Response

The response is an object that includes both the access token and refresh token on success.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

```json
{
    "email": "albert@gmail.com",
    "password": "ilike309"
}
```

Code Templates

This folder contains example requests to create, fork, retrieve, update, or delete code templates.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

POST Create code template

http://localhost:3000/api/CodeTemplates

This endpoint allows users to create a new code template using a HTTP POST request.

Request Body

- `title` (string, required): The title of the code template.
- `explanation` (string, required): A brief explanation or description of the code template.
- `code` (string, required): The actual code content of the template.
- `language` (string, required): The programming language of the code template.
- `tags` (array of strings, optional): The list of tags associated with the code template.

Response

Upon successful creation, the response will include the newly created code template as a JSON object.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
  "title": "Introduction to Python Programming",
  "explanation": "This template provides a basic introduction to Python syntax, including varia
  "code": "a = 5\nb = 10\nsum = a + b\nprint('The sum of a and b is:', sum)",
  "language": "python",
  "tags": ["beginner", "variables", "arithmetic"]
}
```

POST Fork code template

http://localhost:3000/api/CodeTemplates/Fork

This endpoint allows users to fork a code template using a HTTP POST request.

Request Body

- `id` (number): The ID of the code template to fork.

Response

Upon success, the response will include the newly forked code template as a JSON object and a notification informing the user that they have forked a template.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

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

GET Retrieve code templates

http://localhost:3000/api/CodeTemplates?language=python&tags=python&page=1&pageSize=10

This endpoint allows visitors or users to retrieve code templates by specific parameters using a HTTP GET request.

- The title, explanation, and language are matched using a “contains” method, allowing partial matches within those fields. The createdUserId and tags parameters require exact matches.

Request Query

- `title` (string, optional): The title of the code template.
- `explanation` (string, optional): A brief explanation or description of the code template.
- `language` (string, optional): The programming language of the code template.
- `createdUserId` (number): The ID of the user who created the comment.
- `tags` (array of strings, optional): The list of tags associated with the code template.
- `page` (number, required): The current page number of the results being requested.
- `pageSize` (number, required): The number of code templates returned per page

Response

The response, if successful, will include an array of code template objects (with corresponding tags) matching the query parameters and the specific page.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

PARAMS

language	python
tags	python
page	1
pageSize	10

PUT Update code template

http://localhost:3000/api/CodeTemplates/1

This endpoint allows users to edit details of a specific code template (by providing its ID in the URL) using a HTTP PUT request.

Request Body

- `title` (string, optional): The title of the code template.
- `explanation` (string, optional): The explanation of the template.
- `code` (string, optional): The code content of the template.
- `language` (string, optional): The programming language of the code.
- `tags` (array, optional): The list of tags associated with the code template.

Response

The response will confirm the successful update of the code template and return the updated template as a JSON object.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
  "title": "Example code on Python Programming",
  "code": "a = 5\nb = 10\nsum = a + b\nprint('The sum of a and b is:', sum)",
  "language": "python",
  "tags": []
}
```

DELETE Delete code template

http://localhost:3000/api/CodeTemplates/5

This endpoint allows users to delete a specific code template (by providing its ID in the URL) using an HTTP DELETE request.

Response

The response for this request is a message indicating successful or unsucessful template deletion.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Blog Posts

This folder contains example requests to create, retrieve, update, or delete blog posts.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

POST Create blog post

http://localhost:3000/api/BlogPosts

This endpoint allows a user to create a new blog post using an HTTP POST request.

Request Body

- `title` (string, required): The title of the blog post.
- `description` (string, required): A brief description of the blog post.
- `content` (string, required): The main content of the blog post.
- `tags` (array of strings, optional): Tags associated with the blog post.
- `codeTemplates` (array of numbers, optional): IDs of code templates related to the blog post.

Response

Upon successful creation, the response will include the newly created blog post as a JSON object.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "title": "Python and Java",
    "description": "Today I will teach Python and Java.",
    "content": "Python and Java are two of the most popular programming languages, widely used
    "tags": ["intro", "hello"],
    "codeTemplates": [1, 2]
}
```

GET Retrieve blog post

http://localhost:3000/api/BlogPosts?content=python&codeTemplates=1,2&page=2&pageSize=1&order=asc

This endpoint allows visitors or users to retrieve blog posts by specific parameters using an HTTP GET request.

- The title, description, and content are matched using a “contains” method, allowing partial matches within those fields. The createdUserId, tags and codeTemplates parameters require exact matches.

Request Query

- `title` (string, optional): The title of the blog post.
- `description` (string, optional): A brief description of the blog post.
- `content` (string, optional): The main content of the blog post.
- `createdUserId` (number): The ID of the user who created the blog post.
- `tags` (array of strings, optional): The list of tags associated with the blog post.
- `codeTemplates` (array of numbers, optional): The list of code template IDs associated with the blog post.
- `page` (number, required): The current page number of the results being requested.
- `pageSize` (number, required): The number of posts returned per page
- `order` (string, required): The order by rating value in which the blog posts should be sorted (`asc/desc`).
  - `asc` means that the most contrversial posts will appear first
  - `desc` means that the most valued posts will appear first

Response

The response, if successful, will include an array of blog post objects (with corresponding tags and code templates) matching the query parameters and the specific page.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

PARAMS

content	python
codeTemplates	1,2
page	2
pageSize	1
order	asc

PUT Update blog post

```
http://localhost:3000/api/BlogPosts/1
```

This endpoint allows users to edit details of a specific blog post (by providing its ID in the URL) using an HTTP PUT request.

Request Body

- `title` (string, optional): The title of the blog post.
- `description` (string, optional): A brief description of the blog post.
- `content` (string, optional): The main content of the blog post.
- `tags` (array of strings, optional): The list of tags associated with the blog post.
- `codeTemplates` (array of numbers, optional): The list of code template IDs associated with the blog post.

Response

The response will confirm the successful update of the blog post and return the updated post as a JSON object.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```
{
    "title": "More about Web Development",
    "content": "Advanced HTML and CSS techniques enable developers to create accessible, respon
    "tags": ["web", "frontend"],
    "codeTemplates": [4]
}
```

DELETE Delete blog post

http://localhost:3000/api/BlogPosts/5

This endpoint allows users to delete a specific blog post (by providing its ID in the URL) using an HTTP DELETE request.

Response

The response for this request is a message indicating successful or unsuccessful blog post deletion.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Comments

This folder contains example requests to create, retrieve, update, or delete comments.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

POST Create comment

http://localhost:3000/api/Comments

This endpoint allows a user to create a new comment using an HTTP POST request.

Request Body

- One of the following
    - `blogPostId` (string): The ID of the blog post to create the comment for
    - `commentId` (string): The ID of the comment to create the comment (reply) for
- `content` (string): The content of the comment.

Response

Upon successful creation, the response will include the newly created comment as a JSON object.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "content": "Amazing blog post! I learned a lot about coding today!",
    "blogPostId": 1
}
```

GET Retrieve comments

http://localhost:3000/api/Comments?createdUserId=1&blogPostId=1&page=2&pageSize=2&order=desc

This endpoint allows visitors or users to retrieve blog posts by specific parameters using an HTTP GET request.

- The content is matched using a “contains” method, allowing partial matches within those fields. The blogPostId, commentId, and createdUserId require an exact matches.

Request Query

- `blogPostId` (string): The ID of the blog post the comment is for
- `commentId` (string): The ID of the comment the comment (reply) is for
- `content` (string): The content of the comment.
- `createdUserId` (number): The ID of the user who created the comment
- `page` (number, required): The current page number of the results being requested.
- `pageSize` (number, required): The number of comments returned per page
- `order` (string, required): The order by rating value in which the comments should be sorted (`asc/desc`).
    - `asc` means that the most contrversial comments will appear first
    - `desc` means that the most valued comments will appear first

Response

The response, if successful, will include an array of comments objects (with corresponding tags and code templates) matching the query parameters and the specific page.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

PARAMS

createdUserId	1
blogPostId	1
page	2
pageSize	2
order	desc

PUT Update comment

http://localhost:3000/api/Comments/1

This endpoint allows users to edit the content of a specific comment (by providing its ID in the URL) using an HTTP PUT request.

Request Body

- `content` (string): The content of the comment.

Response

The response will confirm the successful update of the comment and return the updated comment as a JSON object.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "content": "What's up bro!"
}
```

DELETE Delete comment

http://localhost:3000/api/Comments/5

This endpoint allows users to delete a specific comment (by providing its ID in the URL) using an HTTP DELETE request.

Response

The response for this request is a message indicating successful or unsuccessful comment deletion.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Ratings

This folder contains example requests to create ratings for blog posts and comments.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

POST Create rating to blog post

http://localhost:3000/api/Ratings/Blog

This endpoint allows users to create a rating for (downvote or upvote) a blog post using a HTTP POST request.

- Upvoting or downvoting the same post multiple times for a single user is not allowed.
- If the user upvotes a post then downvotes it (or vice versa), the original rating gets removed from the post (and they can upvote or downvote again to change their rating).

Request Body

- `value` (number): Either 1 or -1
  - `1` representing an upvote for the blog post
  - `-1` representing a downvote for the blog post
- `blogPostId` (number): The ID of the blog post being rated

Response

The response for this request is a message indicating successful or unsuccessful rating creation or removal (as explained previously).

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```
{
    "value": 1,
    "blogPostId": 1
}
```

POST Create rating to comment

http://localhost:3000/api/Ratings/Comment

This endpoint allows users to create a rating for (downvote or upvote) a comment using a HTTP POST request.

- Upvoting or downvoting the same comment multiple times for a single user is not allowed.
- If the user upvotes a comment then downvotes it (or vice versa), the original rating gets removed from the post (and they can upvote or downvote again to change their rating).

Request Body

- `value` (number): Either 1 or -1
    - `1` representing an upvote for the comment
    - `-1` representing a downvote for the comment
- `commentId` (number): The ID of the comment being rated

Response

The response for this request is a message indicating successful or unsuccessful rating creation or removal (as explained previously).

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```
{
    "value": 1,


    "commentId": 1
}
```

Reports

This folder contains example requests to create reports for blog posts and comments.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

POST Create report to blog post

http://localhost:3000/api/Report/Blog

This endpoint is used for a user to report a blog post for a specific reason using a HTTP POST request.

- To prevent spam, a user can only report a specific blog post once.

Request Body

- `blogPostId` (Number): The ID of the blog post being reported.
- `reason` (String): The reason for reporting the blog post.

Response

The response for this request is a message indicating successful or unsuccessful report creation.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```
{
    "reason": "this post has bad words",
    "blogPostId": 1
}
```

POST Create report to comment

http://localhost:3000/api/Report/Comment

This endpoint is used for a user to report a comment for a specific reason using a HTTP POST request.

- To prevent spam, a user can only report a specific comment once.

Request Body

- `commentId` (Number): The ID of the comment being reported.
- `reason` (String): The reason for reporting the comment.

Response

The response for this request is a message indicating successful or unsuccessful report creation.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "reason": "this post has bad words",
    "commentId": 1
}
```

Admin

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

GET Get top reported blogs

http://localhost:3000/api/Admin/ReportedBlogs?blogAmount=5

This endpoint allows an admin user to retrieve the top reported blogs (by specifying the amount of blogs to be fetched), using a HTTP GET request.

Request Query

- `blogAmount` (number): The number of reported blogs to be fetched.

Response

The response, if successful, will include an array of the top `blogAmount` most reported blog posts objects.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

PARAMS

blogAmount	5

GET Get top reported comments

```
http://localhost:3000/api/Admin/ReportedComments?commentAmount=5
```

This endpoint allows an admin user to retrieve the top reported comments (by specifying the amount of comments to be fetched), using a HTTP GET request.

Request Query

- `commentAmount` (number): The number of reported comments to be fetched.

Response

The response, if successful, will include an array of the top `commentAmount` most reported comments objects.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

PARAMS

commentAmount	5

POST Hide content

http://localhost:3000/api/Admin/HideContent

This endpoint allows an admin user to hide a specific blog post or comment (identified by a blog post or comment ID) using a HTTP POST request.

Request Body

- `blogPostId` (number) - The ID of the blog post to be hidden.

OR

- `commentId` (number) - The ID of the comment to be hidden.

Response

The response for this request is a message indicating success or failure to hide content.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "blogPostId": 2
}
```

Code Execution

This API endpoint allows a vistitor or user to execute code in one of the following programming languages: **Python, C, Java, Javascript, C++**.

Request Body

- `language` (string): The programming language in which the code should be executed.
- `code` (string): The code to be executed.
- `input` (string, optional): The input to be provided to the code.

- `className` (string, optional): Creates and runs the file on the server:
    - className.[fileExt] (ie. .py, .c ...)

Response

The response for the request is either an error message that represents any compile errors, runtime errors, and warnings, or, if the code executes successfully, the output of the code.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

POST Valid Python

http://localhost:3000/api/CodeRunner

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "language": "python3",
    "code": "name = input(\"Please enter your name: \")\nprint(name)\nage = input(\"Please ente
    "input": "Albert\n25\n1998-04-10"
}
```

POST Valid C

http://localhost:3000/api/CodeRunner

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "language": "c",
    "code": "#include <stdio.h>\n#include <stdlib.h>\n\nint main() {\n    char name[100];\n
    "input": "Albert\n25\n2000-04-10"
}
```

POST Valid Java

http://localhost:3000/api/CodeRunner

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "language": "java",
    "code": "import java.util.Scanner;\n\npublic class UserInfo {\n    public static void main
    "input": "Albert\n25\n2000-04-10",
    "className": "UserInfo"
}
```

POST Valid Javascript

http://localhost:3000/api/CodeRunner

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
```

```
{
    "language": "javascript",
    "code": "const readline = require('readline');\n\nconst rl = readline.createInterface({\n

    "input": "Albert\n25\n1998-04-10"
}
```

POST Valid CPP

http://localhost:3000/api/CodeRunner

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```
{
    "language": "cpp",
    "code": "#include <iostream>\n#include <string>\n\nint main() {\n    std::string name, birt
    "input": "Albert\n25\n1998-04-10"
}
```

POST Compile error

http://localhost:3000/api/CodeRunner

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```
{
    "language": "python3",
    "code": "name = inpt(\"Please enter your name: \")\nprint(name)\nage = input(\"Please ente
    "input": "Albert\n25\n1998-04-10"
}
```

POST Syntax error

http://localhost:3000/api/CodeRunner

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "language": "python3",
    "code": "name = 10\nname.sort()\n",
    "input": "Albert\n25\n1998-04-10"
}
```

POST Code warning

http://localhost:3000/api/CodeRunner

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
  "language": "c",
  "code": "#include <stdio.h>\n\nint main() {\n    int x;\n    printf(\"Value: %d\\n\", x); //
}
```

Profile

This folder includes all requests related to retrieving and updating user profiles.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Scriptorium API

PATCH Profile: Patch

http://localhost:3000/api/Profile/Update

This endpoint allows a user to edit their profile using an HTTP PATCH request. Of the optional fields of the request body listed below, only the ones included will be updated.

Request Body

- `firstName` (string, optional): The new first name of the user.
- `lastName` (string, optional): The new last name of the user.
- `email` (string, optional): The new email of the user.
- `username` (string, optional): The new username of the user.
- `phoneNumber` (string, optional): The new phone number of the user.

Response

Upon successful creation, the response will include the full user row from the DB as a JSON object.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

Body raw (json)

json

```json
{
    "email": "oskip@gmail.com",
    "firstName": "Shenglong"
}
```

GET Profile: Get

http://localhost:3000/api/Profile/1

This endpoint returns a user's profile information using a HTTP GET request.

Request Query

- `id` (string, required): The id of the profile to fetch.

Response

The response, if successful, will be a JSON object with the fields `id, firstName`, `lastName`, `email`, `username`, `phoneNumber`, and `avatar`, which correspond to the user's information with the given id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

POST Avatar: Post

http://localhost:3000/api/Profile/Avatar

Profile Avatar Upload

This endpoint allows the user to upload a new avatar for their profile.

Request Body

- form-data
  - `file` (file): The image file to be uploaded as the avatar.

Response

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

json

```
{
    "success": boolean,
    "data": {
        "id": number,
        "firstName": string,
        "lastName": string,
        "email": string,
        "avatar": string,
        "phoneNumber": string,
        "password": string,
        "role": string
```

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API

PARAMS

Body formdata

file

GET Avatar: Get

http://localhost:3000/api/Profile/Avatar/1-9djeqaam94.png

This endpoint returns a profile image, given the image's ID using a HTTP GET request.

Request Query

- `id` (string): The id of the avatar to fetch.

Response

The response, if successful, will be a JSON object with field `imageBuffer`, and the value will be an object of type Buffer. Converting and displaying the image from the Buffer object will be handled on the UI.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Scriptorium API