

Scriptorium API Documentation

Omar El Malak & Allaith Alzoubi
CSC309: Programming on the Web
University of Toronto
Professor Kianoosh Abbasi
November 3, 2024

Users

Object Base Url: <http://localhost:3000/api/users>

Description

Scriptorium **users** register and undergo verification via JSON Web Token to access a comprehensive suite of features, allowing them to create, edit, and delete their own **blogs**, **comments**, and **code templates**.

Database Schema

Component	Type	Description & Example	Relational Query & Database
id	Int (<i>default</i>)	The Prisma database ID of the user <i>Example: 1</i>	N/A
username	String (<i>unique</i>)	The user's Scriptorium username <i>Example: omarelmalak28</i>	N/A
password	String	The user's Scriptorium password <i>Example: ilovecsc309!</i>	N/A
firstName	String	The user's first name <i>Example: Omar</i>	N/A
lastName	String	The user's last name <i>Example: El Malak</i>	N/A
email	String	The user's email <i>Example: omar@scriptorium.com</i>	N/A
profilePicture	String?	The user's profile picture (relative file path) <i>Example: ../assets/images/hi.png</i>	N/A
phoneNumber	String	The user's associated phone number <i>Example: 123-456-7890</i>	N/A
createdAt	DateTime (<i>default</i>)	The date/time the account was created <i>Example: 2024-11-03T15:30:00.000Z</i>	N/A
updatedAt	DateTime (updatedAt)	The date/time the account was last updated <i>Example: 2024-11-04T13:30:00.000Z</i>	N/A
role	String	The permission level (user/admin) the user has <i>Example: USER</i>	N/A
blogs	Blog[]	The list containing all blogs the user has created	"UserBlogs" <i>DB: Blog</i>

codes	CodeTemplate[]	The list containing all code templates the user has created	“UserCodeTemplates” <i>DB: CodeTemplate</i>
comments	Comment[]	The list containing all comments the user has created	“UserComments” <i>DB: Comment</i>
upvotedComments	Comment[]	The list containing all comments the user has upvoted	“UpvotedComments” <i>DB: Comment</i>
downvotedComments	Comment[]	The list containing all comments the user has downvoted	“DownvotedComments” <i>DB: Comment</i>
upvotedBlogs	Blog[]	The list containing all blogs the user has upvoted	“UpvotedBlogs” <i>DB: Blog</i>
downvotedBlogs	Blog[]	The list containing all blogs the user has downvoted	“DownvotedBlogs” <i>DB: Blog</i>

Endpoints

Registration

Method(s): **POST**

Relative Path Endpoint: /register

Description

This endpoint allows a user to register to Scriptorium, adding them to the **users** database.

Query Parameters

N/A

Sample Request

```
POST http://localhost:3000/api/users/register
Content-Type: application/json
{
  "username": "johnDoe",
  "password": "securePassword123",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "phoneNumber": "123-456-7890"
}
```

201 Sample Response

```
{
  "message": "Successful Registration",
  "user": {
    "id": 1,
    "username": "johnDoe",
    "firstName": "John",
    "lastName": "Doe",
    "email": "john.doe@example.com",
    "phoneNumber": "+1234567890",
    "profilePicture":
    "../../../assets/images/default_avatar.png",
    "role": "USER",
    "createdAt": "2024-11-03T15:30:00.000Z",
    "updatedAt": "2024-11-03T15:30:00.000Z"
  }
}
```

Login

Method(s): **POST**

Relative Path Endpoint: /login

Description

This endpoint allows a user to login to Scriptorium and generate their unique JSON Web Tokens (login and refresh).

Query Parameters

N/A

Sample Request

```
POST http://localhost:3000/api/users/login
Content-Type: application/json
{
  "username": "johnDoe",
  "password": "securePassword123"
}
```

201 Sample Response

```
{
  "accessToken":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImpvaG5Eb2UiLCJyb2x1IjoivVNFUiIsImV4cGlyZXNBdCI6MTYwOTI0MDA4Mn0.2G8fMKk_G4HaxNljNY5b3YApdm2YVsoYex08nIqG9ps",
  "refreshToken":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImpvaG5Eb2UiLCJyb2x1IjoivVNFUiIsImV4cGlyZXNBdCI6MTYwOTI0MDA4Mn0.2g7SKKkG2kP5X3do39s9SjL4Ih2k68_EaKrkGfDDcbM"
}
```

Refresh

Method(s): **POST**

Relative Path Endpoint: /refresh

Description

This endpoint generates a new access token when a valid refresh token is provided. This is crucial for maintaining **user** sessions without requiring them to log in again after the access token expires.

Query Parameters

N/A

Sample Request

```
POST http://localhost:3000/api/users/refresh
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json
{
  "refreshToken":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImpvaG5Eb2UiLCJyb2x1IjoivVNFUiIsImV4cGlyZXNBdCI6MTYwOTI0MDA4Mn0.xxxx"
}
```

201 Sample Response

```
{
  "accessToken":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImpvaG5Eb2UiLCJyb2x1IjoivVNFUiIsImV4cGlyZXNBdCI6MTYwOTI0MDA4Mn0.xxxx"
```

```
yb2x1IjoiVVNFUiIsImV4cGlyZXNBdCI6MTYwOTI0MDA4Mn0.yyyy"  
}
```

Blogs

Object Base Url: <http://localhost:3000/api/blogs>

Description

Blogs are where Scriptorium **users** share insights, experiences, and knowledge on various code-related topics. They facilitate engagement through **comments** and ratings, allowing other **users** to interact with content. Additionally, blogs often include links to **code templates**, enhancing the learning experience and fostering community discussions.

Database Schema

Component	Type	Description & Example	Relational Query & Database
id	Int (<i>default</i>)	The Prisma database ID of the blog <i>Example: 1</i>	N/A
authorId	Int	The author's Prisma database ID <i>Example: 1</i>	N/A
title	String (<i>unique</i>)	The title of the blog <i>Example: How to Code</i>	N/A
description	String	The description (content) of the blog <i>Example: First, print("Hello world")</i>	N/A
createdAt	DateTime (<i>default</i>)	The date/time the blog was created <i>Example: 2024-11-03T15:30:00.000Z</i>	N/A
updatedAt	DateTime (<i>updatedAt</i>)	The date/time the blog was last updated <i>Example: 2024-11-04T13:30:00.000Z</i>	N/A
upvotes	Int (<i>default(0)</i>)	The number of upvotes the comment has <i>Example: 1</i>	N/A
downvotes	Int (<i>default(0)</i>)	The number of downvotes the comment has <i>Example: 1</i>	N/A
totalVotes	Int (<i>default(0)</i>)	The total number of downvotes and upvotes the comment has <i>Example: 2</i>	N/A
hidden	Boolean (<i>default(false)</i>)	A flag indicating if the blog has been hidden (true) by an administrator <i>Example: False</i>	N/A
comments	Comment[]	The list containing all comments under the blog	"BlogComments" <i>DB: Comment</i>

codes	CodeTemplate[]	The list containing all code templates the blog references	“BlogCodeTemplates” <i>DB: CodeTemplate</i>
tags	Tag[]	The list containing all tags the blog references	“BlogTags” <i>DB: Tag</i>
author	User	The user who is the author of the blog	“UserBlogs” <i>DB: User</i>
upvoters	User[]	The list containing all users that have upvoted the blog	“UpvotedBlogs” <i>DB: User</i>
downvoters	User[]	The list containing all users that have upvoted the blog	“DownvotedBlogs” <i>DB: User</i>
reports	Report[]	The list containing all reports filed against the blog	“BlogReports” <i>DB: Report</i>

Endpoints

Create

Method(s): **POST**

Relative Path Endpoint: /create

Description

This endpoint allows a **user** to create a Scriptorium **blog** post, adding it to the **blogs** database.

Query Parameters

N/A

Sample Request

```
POST http://localhost:3000/api/blogs/create
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json
{
  "title": "Understanding Async Programming",
  "description": "This blog post explains the basics of
asynchronous programming in JavaScript.",
  "tags": ["JavaScript", "Async", "Programming"],
  "codeIds": [1, 2, 3]
}
```

201 Sample Response


```
{
  "blog": {
    "id": 123,
    "title": "Understanding Async Programming",
    "description": "This blog post explains the basics of
asynchronous programming in JavaScript.",
    "authorId": 45,
    "tags": [
      { "id": 1, "name": "JavaScript" },
      { "id": 2, "name": "Async" },
      { "id": 3, "name": "Programming" }
    ],
    "codes": [
      { "id": 1 },
      { "id": 2 },
      { "id": 3 }
    ]
  }
}
```

Blog Management

Method(s): **GET**, **PUT**, **DELETE**

Relative Path Endpoint: */:id*

Description

This endpoint allows visitors to obtain viewing access to blogs and allows **users** to obtain viewing access, modify, and delete blogs.

Query Parameters

Parameter	Request Method	Type	Description & Example	In <i>Path/Query</i>
id	GET , PUT , DELETE	Int	Requested Prisma database blog ID <i>Example: 1</i>	Path
page	GET	Int	Requested page number of associated comments, tags, and codes	Query

			<i>Example: 1</i>	
limit	GET	Int	Limit to number of entries per page of associated comments, tags, and codes <i>Example: 10</i>	Query

Sample Request (**GET**)

```
GET http://localhost:3000/api/blogs/1?page=1&limit=10
```

200 Sample Response (**GET**)

```
{
  "blog": {
    "id": 1,
    "title": "Sample Blog Post",
    "description": "This is a sample blog post.",
    "authorId": 2,
    "hidden": false,
    "comments": [
      {
        "id": 101,
        "text": "Great post!",
        "authorId": 3,
        "createdAt": "2024-11-01T12:34:56Z"
      },
      // More comments (up to `limit`)
    ],
    "tags": [
      { "id": 201, "name": "JavaScript" },
      { "id": 202, "name": "Web Development" }
    ],
    "codes": [
      { "id": 301, "content": "console.log('Hello World');" }
    ]
  }
}
```

Sample Request (**PUT**)

```
PUT http://localhost:3000/api/blogs/1
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json

{
  "title": "Updated Blog Title",
  "description": "Updated description of the blog post.",
  "tagIds": [201, 203],
  "codeIds": [301]
}
```

200 Sample Response (PUT)

```
{
  "blog": {
    "id": 1,
    "title": "Updated Blog Title",
    "description": "Updated description of the blog post.",
    "authorId": 2,
    "tags": [
      { "id": 201, "name": "JavaScript" },
      { "id": 203, "name": "Node.js" }
    ],
    "codes": [
      { "id": 301, "content": "console.log('Updated Content');" }
    ]
  }
}
```

Sample Request (DELETE)

```
DELETE http://localhost:3000/api/blogs/1
Authorization: Bearer <JWT_TOKEN>
```

200 Sample Response (DELETE)

```
{
  "message": "Successfully deleted blog"
}
```

Search

Method(s): **GET**

Relative Path Endpoint: `/search`

Description

This endpoint allows visitors and **users** to search for blogs by any combination of **code templates**, **content**, **tags**, or **title** using a search term. Alternatively, visitors and **users** can perform a general search for all blogs.

Query Parameters

Parameter	Request Method	Type	Description & Example	In Path/Query
category	GET	String?	A comma-separated string of categories to search by (title/codes/content/tags) <i>Example: title,content</i>	Query
page	GET	Int	Requested page number <i>Example: 1</i>	Query
limit	GET	Int	Limit to number of blog entries per page <i>Example: 10</i>	Query

Sample Request (GENERAL SEARCH)

```
GET http://localhost:3000/api/blogs/search?page=1&limit=10
```

200 Sample Response (GENERAL SEARCH)

```
{
  "results": [
    { "id": 1, "title": "First Blog Post", "content": "Introduction to programming", "tags": ["coding", "tutorial"], "codes": ["JS101"] },
    { "id": 2, "title": "JavaScript Basics", "content": "JavaScript essentials", "tags": ["JavaScript", "basics"], "codes": ["JS201"] },
    ...
  ],
}
```

```
"total": 20
}
```

Sample Request (SINGLE-CATEGORY SEARCH)

```
GET
http://localhost:3000/api/blogs/search?category=title&searchTerm=Java
Script&page=1&limit=10
```

200 Sample Response (GENERAL SEARCH)

```
{
  "results": [
    { "id": 1, "title": "First Blog Post", "content": "Introduction
to programming", "tags": ["coding", "tutorial"], "codes": ["JS101"]
  },
    { "id": 2, "title": "JavaScript Basics", "content": "JavaScript
essentials", "tags": ["JavaScript", "basics"], "codes": ["JS201"] },
    ...
  ],
  "total": 20
}
```

Sample Request (MULTI-CATEGORY SEARCH)

```
GET
http://localhost:3000/api/blogs/search?category=title,content&searchT
erm=programming&page=1&limit=10
```

200 Sample Response (MULTI-CATEGORY SEARCH)

```
{
  "results": [
    { "id": 1, "title": "First Programming Post", "content":
"Introduction to Java", "tags": ["coding", "tutorial"], "codes":
["JS101"] },
    { "id": 2, "title": "JavaScript Basics", "content": "JavaScript
programming essentials", "tags": ["JavaScript", "basics"], "codes":
["JS201"] },
    ...
  ]
}
```

```
],  
  "total": 20  
}
```

Interact

Method(s): **POST**

Relative Path Endpoint: /interact

Description

This endpoint allows **users** to interact with blogs by upvoting and downvoting them.

Query Parameters

Parameter	Request Method	Type	Description & Example	In Path/Query
upvote/downvote	POST	String	Requested interaction <i>Example: upvote</i>	Path
blogId	POST	Int	Requested Prisma database blog ID <i>Example: 1</i>	Query

Sample Request (UPVOTE)

```
POST http://localhost:3000/api/blogs/upvote?blogId=1  
Authorization: Bearer <JWT_TOKEN>
```

200 Sample Response (UPVOTE)

```
{  
  "message": "Upvoted successfully"  
}
```

Sample Request (DOWNVOTE)

```
POST http://localhost:3000/api/blogs/downvote?blogId=1  
Authorization: Bearer <JWT_TOKEN>
```

200 Sample Response (DOWNVOTE)

```
{  
  "message": "Downvoted successfully"  
}
```

Sort

Method(s): **GET**

Relative Path Endpoint: /sort

Description

This endpoint allows visitors and **users** to receive sorted blogs based on blog ratings from most controversial to least controversial, or most valued to least valued. As well, the endpoint allows visitors and **users** to sort **comments** within a blog via the same metrics.

Query Parameters

Parameter	Request Method	Type	Description & Example	In Path/Query
blogRatings/commentRatings	GET	Int	Item to sort by filter type metric (blogs/comments) <i>Example: blogRatings</i>	Path
filterType	GET	String	Requested filter type metric (most valued/most controversial) <i>Example: value</i>	Query
page	GET	Int	Requested page number <i>Example: 1</i>	Query
limit	GET	Int	Limit to number of entries per page <i>Example: 10</i>	Query

Sample Request (MOST VALUED BLOGS)

GET

http://localhost:3000/api/blogs/sort/blogRatings?filterType=value&page=1&limit=10

200 Sample Response (MOST VALUED BLOGS)

```

{
  "blogs": [
    {
      "id": 1,
      "title": "Understanding JavaScript Closures",
      "upvotes": 25,
      "downvotes": 5,
      "hidden": false,
      "authorId": 3,
      "tags": [
        { "id": 1, "name": "JavaScript" },
        { "id": 2, "name": "Programming" }
      ]
    },
    {
      "id": 2,
      "title": "The Importance of Clean Code",
      "upvotes": 20,
      "downvotes": 2,
      "hidden": false,
      "authorId": 4,
      "tags": [
        { "id": 3, "name": "Coding" }
      ]
    }
  ],
  "page": 1,
  "limit": 10,
  "totalBlogs": 50,
  "totalPages": 5
}

```

Sample Request (MOST CONTROVERSIAL BLOGS)

GET

http://localhost:3000/api/blogs/sort/blogRatings?filterType=controversial&page=2&limit=10

200 Sample Response (MOST CONTROVERSIAL BLOGS)


```

{
  "blogs": [
    {
      "id": 11,
      "title": "Why Testing is Essential",
      "upvotes": 5,
      "downvotes": 30,
      "hidden": false,
      "authorId": 5,
      "tags": [
        { "id": 4, "name": "Testing" }
      ]
    },
    {
      "id": 12,
      "title": "The Downside of Over-Engineering",
      "upvotes": 3,
      "downvotes": 25,
      "hidden": false,
      "authorId": 6,
      "tags": [
        { "id": 5, "name": "Architecture" }
      ]
    }
  ],
  "page": 2,
  "limit": 10,
  "totalBlogs": 50,
  "totalPages": 5
}

```

Sample Request (MOST VALUED COMMENTS)

GET

http://localhost:3000/api/blogs/sort/commentRatings?blogId=1&filterType=value&page=1&limit=10

200 Sample Response (MOST VALUED COMMENTS)

```

{

```

```

"comments": [
  {
    "id": 1,
    "content": "Great explanation of closures!",
    "upvotes": 10,
    "downvotes": 2,
    "hidden": false,
    "authorId": 3,
    "blogId": 1
  },
  {
    "id": 2,
    "content": "I completely disagree with this point.",
    "upvotes": 5,
    "downvotes": 15,
    "hidden": false,
    "authorId": 4,
    "blogId": 1
  }
],
"page": 1,
"limit": 10,
"totalComments": 20,
"totalPages": 2
}

```

Sample Request (MOST CONTROVERSIAL COMMENTS)

GET

http://localhost:3000/api/blogs/api/comments?blogId=1&filterType=controversial&page=2&limit=10

200 Sample Response (MOST CONTROVERSIAL COMMENTS)

```

{
  "comments": [
    {
      "id": 11,
      "content": "The examples could be improved.",
      "upvotes": 2,

```

```
        "downvotes": 10,  
        "hidden": false,  
        "authorId": 5,  
        "blogId": 1  
    },  
    {  
        "id": 12,  
        "content": "Interesting perspective!",  
        "upvotes": 1,  
        "downvotes": 3,  
        "hidden": false,  
        "authorId": 6,  
        "blogId": 1  
    }  
],  
"page": 2,  
"limit": 10,  
"totalComments": 20,  
"totalPages": 2  
}
```

Comments

Object Base Url: <http://localhost:3000/api/comments>

Description

Comments allow Scriptorium users to engage in discussions and share feedback on **blogs**. They facilitate interaction by enabling users to ask questions and provide insights, creating a collaborative environment. Additionally, comments can be rated, promoting valuable contributions and enriching the community experience.

Database Schema

Component	Type	Description & Example	Relational Query & Database
id	Int (<i>default</i>)	The Prisma database ID of the comment <i>Example: 2</i>	N/A
parentId	Int?	The Prisma database ID of the parent comment (if it is a reply) <i>Example: 1</i>	N/A
createdAt	DateTime (<i>default</i>)	The date/time the comment was created <i>Example: 2024-11-03T15:30:00.000Z</i>	N/A
updatedAt	DateTime (<i>updatedAt</i>)	The date/time the comment was last updated <i>Example: 2024-11-04T13:30:00.000Z</i>	N/A
upvotes	Int	The number of upvotes the comment has <i>Example: 1</i>	N/A
downvotes	Int	The number of downvotes the comment has <i>Example: 1</i>	N/A
content	String	The description (content) of the comment <i>Example: So interesting!</i>	N/A
authorId	Int	The Prisma database ID of the author of the comment <i>Example: 1</i>	N/A
blogId	Int	The Prisma database ID of the blog the comment is under <i>Example: 1</i>	N/A

hidden	Boolean (<i>default(false)</i>)	A flag indicating if the comment has been hidden (true) by an administrator <i>Example: False</i>	N/A
parent	Comment?	The parent comment of the blog	“CommentReplies” <i>DB: Comment</i>
replies	Comment[]	The list containing all comment replies the blog references	“CommentReplies” <i>DB: CodeTemplate</i>
author	User	The user who is the author of the comment	“UserComments” <i>DB: Tag</i>
blog	Blog	The blog the comment is under	“BlogComments” <i>DB: User</i>
upvoters	User[]	The list containing all users that have upvoted the comment	“UpvotedComments” <i>DB: User</i>
downvoters	User[]	The list containing all users that have upvoted the comment	“DownvotedComments” <i>DB: User</i>
reports	Report[]	The list containing all reports filed against the comment	“CommentReports” <i>DB: Report</i>

Endpoints

Create

Method(s): **POST**

Relative Path Endpoint: /create

Description/

This endpoint allows a **user** to create a Scriptorium **comment** under a **blog**, adding it to the **comment** database.

Query Parameters

N/A

Request Details

Field	Type	Description & Example
-------	------	-----------------------

Sample Request

```
POST http://localhost:3000/api/comments/create
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json
```

```
{
  "blogId": "1",
  "content": "This is a great blog post! Thanks for sharing.",
  "parentId": null
}
```

201 Sample Response

```
{
  "comment": {
    "id": 1,
    "content": "This is a great blog post! Thanks for sharing.",
    "parentId": null,
    "blogId": 1,
    "authorId": "1",
    "createdAt": "2024-11-03T12:00:00.000Z",
    "updatedAt": "2024-11-03T12:00:00.000Z"
  }
}
```

Comment Management

Method(s): **PUT, DELETE**
Relative Path Endpoint: */:id*

Description

This endpoint allows a **user** to modify and delete **comments** (NOTE: The GET method is not included since **blogs** are able to fetch their respective **comments**).

Query Parameters

Parameter	Request Method	Type	Description & Example	In Path/Query
id	PUT, DELETE	Int	Requested Prisma database comment ID <i>Example: 1</i>	Path

Sample Request (**PUT**)

```
PUT http://localhost:3000/api/comments/5
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json
{
  "content": "Updated comment content."
}
```

201 Sample Response (PUT)

```
{
  "blog": {
    "id": 123,
    "title": "Understanding Async Programming",
    "description": "This blog post explains the basics of asynchronous programming in JavaScript.",
    "authorId": 45,
    "tags": [
      { "id": 1, "name": "JavaScript" },
      { "id": 2, "name": "Async" },
      { "id": 3, "name": "Programming" }
    ],
    "codes": [
      { "id": 1 },
      { "id": 2 },
      { "id": 3 }
    ]
  }
}
```

Sample Request (DELETE)

```
DELETE /1
Authorization: Bearer <JWT_TOKEN>
```

200 Sample Response (DELETE)

```
{
  "message": "Comment deleted"
}
```

Interact

Method(s): **POST**

Relative Path Endpoint: /**interact**

Description

This endpoint allows **users** to interact with comments by upvoting and downvoting them.

Query Parameters

Parameter	Request Method	Type	Description & Example	In <i>Path/Query</i>
upvote/downvote	POST	String	Requested interaction <i>Example: upvote</i>	Path
commentId	POST	Int	Requested Prisma database comment ID <i>Example: 1</i>	Query

Request Details

Field	Type	Description & Example
-------	------	-----------------------

Sample Request (UPVOTE)

```
POST http://localhost:3000/api/comments/upvote?commentId=1
Authorization: Bearer <JWT_TOKEN>
```

200 Sample Response (UPVOTE)

```
{
  "message": "Upvoted successfully"
}
```

Sample Request (DOWNVOTE)

```
POST http://localhost:3000/api/comments/downvote?commentId=1
Authorization: Bearer <JWT_TOKEN>
```

200 Sample Response (DOWNVOTE)


```
{  
  "message": "Downvoted successfully"  
}
```

Code Templates

Object Base Url: <http://localhost:3000/api/codes>

Description

Code templates (codes) empower Scriptorium **users** to create their own coding solutions, modify existing ones they developed, and easily fork templates from others. This flexibility encourages collaboration and innovation, allowing users to adapt and enhance their projects while sharing their improvements with the community.

Database Schema

Component	Type	Description & Example	Relational Query & Database
id	Int (<i>default</i>)	The Prisma database ID of the code template <i>Example: 2</i>	N/A
authorId	Int	The author's Prisma database ID <i>Example: 1</i>	N/A
title	String (<i>unique</i>)	The title of the code template <i>Example: Hello World Program</i>	N/A
explanation	String	The explanation of the template <i>Example: Seamlessly prints Hello World</i>	N/A
content	String	The content of the template <i>Example: print("Hello World")</i>	N/A
createdAt	DateTime (<i>default</i>)	The date/time the code template was created <i>Example: 2024-11-03T15:30:00.000Z</i>	N/A
updatedAt	DateTime (<i>updatedAt</i>)	The date/time the code template was last updated <i>Example: 2024-11-04T13:30:00.000Z</i>	N/A
language	String	The programming language selected <i>Example: Javascript</i>	N/A
input	String	The input from stdin <i>Example: "Laith"</i>	N/A
parentId	Int?	The Prisma database ID of the parent code template that the code template is forked from <i>Example: 1</i>	N/A

blogs	Blog[]	The list containing all blogs that the code template is associated with	“BlogCodeTemplates” <i>DB: Blog</i>
author	User	The user that is the author of the code template	“UserCodeTemplates” <i>DB: User</i>
tags	Tag[]	The tags that is the code template is associated with	“CodeTemplateTags” <i>DB: Tags</i>
parent	CodeTemplate?	The code template that this code template is forked from	“CodeTemplateForks” <i>DB: CodeTemplate</i>
forkedChildren	CodeTemplate[]	The list containing all code templates forked from this code template	“CodeTemplateForks” <i>DB: CodeTemplate</i>

Endpoints

Create

Method(s): **POST**

Relative Path Endpoint: /create

Description

This endpoint allows a **user** to create a Scriptorium **code template**, adding it to the **code templates (codes)** database.

Query Parameters

N/A

Sample Request

```
POST http://localhost:3000/api/codes/create
Content-Type: application/json
Authorization: Bearer <JWT_TOKEN>

{
  "title": "Sample Code Template",
  "explanation": "This template demonstrates how to implement a
simple algorithm.",
  "content": "function sampleFunction() {\n    // Your code
here\n}",
  "tagIds": [1, 2, 3]
}
```

201 Sample Response

```
{
  "codeTemplate": {
    "id": 1,
    "title": "Sample Code Template",
    "explanation": "This template demonstrates how to implement a simple algorithm.",
    "content": "function sampleFunction() {\n    // Your code here\n}",
    "authorId": 123,
    "parentId": null,
    "tags": [
      { "id": 1, "name": "JavaScript" },
      { "id": 2, "name": "Algorithm" },
      { "id": 3, "name": "Template" }
    ],
    "createdAt": "2024-11-03T12:00:00Z",
    "updatedAt": "2024-11-03T12:00:00Z"
  }
}
```

Code Template Management
Method(s): **GET, PUT, DELETE**
Relative Path Endpoint: */:id*

Description

This endpoint allows visitors to obtain viewing access to **code templates** and allows **users** to obtain viewing access, modify, and delete **code templates**.

Query Parameters

Parameter	Request Method	Type	Description & Example	In Path/Query
id	GET, PUT, DELETE	Int	Requested Prisma database code template ID <i>Example: 1</i>	Path
page	GET	Int	Requested page number of associated blogs and	Query

			tags <i>Example: 1</i>	
limit	GET	Int	Limit to number of entries per page of blogs and tags <i>Example: 10</i>	Query

Sample Request (**GET**)

```
GET http://localhost:3000/api/codes/1?page=1&limit=5
Authorization: Bearer <JWT_TOKEN>
```

200 Sample Response (**GET**)

```
{
  "codeTemplate": {
    "id": 1,
    "title": "Sample Code Template",
    "explanation": "This template demonstrates how to implement a simple algorithm.",
    "content": "function sampleFunction() {\n    // Your code here\n}",
    "authorId": 123,
    "parentId": null,
    "tags": [
      { "id": 1, "name": "JavaScript" },
      { "id": 2, "name": "Algorithm" }
    ],
    "blogs": [
      { "id": 10, "title": "Blog Post 1" },
      { "id": 11, "title": "Blog Post 2" }
    ],
    "createdAt": "2024-11-03T12:00:00Z",
    "updatedAt": "2024-11-03T12:00:00Z"
  }
}
```

Sample Request (**PUT**)

```
PUT http://localhost:3000/api/codes/1
Content-Type: application/json
```

Authorization: Bearer <JWT_TOKEN>

```
{
  "title": "Updated Sample Code Template",
  "explanation": "This template now includes more details.",
  "content": "function updatedFunction() {\n    // Updated code here\n}",
  "tagIds": [1, 3]
}
```

200 Sample Response (PUT)

```
{
  "codeTemplate": {
    "id": 1,
    "title": "Updated Sample Code Template",
    "explanation": "This template now includes more details.",
    "content": "function updatedFunction() {\n    // Updated code here\n}",
    "authorId": 123,
    "parentId": null,
    "tags": [
      { "id": 1, "name": "JavaScript" },
      { "id": 3, "name": "New Tag" }
    ],
    "createdAt": "2024-11-03T12:00:00Z",
    "updatedAt": "2024-11-03T12:15:00Z"
  }
}
```

Sample Request (DELETE)

DELETE http://localhost:3000/api/codes/1
Authorization: Bearer <JWT_TOKEN>

200 Sample Response (DELETE)

```
{
  "message": "Successfully deleted code template"
}
```

Search

Method(s): **GET**

Relative Path Endpoint: **/search**

Description

This endpoint allows visitors and **users** to search for code templates by any combination of **explanation**, **tags**, or **title** using a search term. Alternatively, visitors and **users** can perform a general search for all code templates. The client should specify whether the scope of the search is “local” (pertaining to a single user’s code templates) or “global” (general search across all code templates in the database). The handler defaults to “global” scope.

Query Parameters

Parameter	Request Method	Type	Description & Example	In Path/Query
category	GET	String?	A comma-separated string of categories to search by (title/codes/content/tags) <i>Example: title,content</i>	Query
scope	GET	String	The scope of search (local, user-centric /global, all users) <i>Example: local</i>	Query
page	GET	Int	Requested page number <i>Example: 1</i>	Query
limit	GET	Int	Limit to number of code template entries per page <i>Example: 10</i>	Query

Sample Request (GLOBAL, GENERAL SEARCH)

GET

`http://localhost:3000/api/codes/search?page=1&limit=10&scope=global`

200 Sample Response (GLOBAL, GENERAL SEARCH)

```
{
```

```

    "results": [
      {
        "id": 1,
        "title": "Example Code 1",
        "explanation": "This code example demonstrates...",
        "content": "function example1() {...}",
        "tags": ["JavaScript", "Example"],
        "authorId": 123
      },
      {
        "id": 2,
        "title": "Example Code 2",
        "explanation": "Another code example...",
        "content": "function example2() {...}",
        "tags": ["JavaScript", "Tutorial"],
        "authorId": 124
      }
    ],
    "total": 20
  }
}

```

Sample Request (GLOBAL, SINGLE-CATEGORY SEARCH)

```

GET
http://localhost:3000/api/codes/api/codes/search?category=title&searchTerm=example&page=1&limit=10&scope=global

```

200 Sample Response (GLOBAL, SINGLE-CATEGORY SEARCH)

```

{
  "results": [
    {
      "id": 1,
      "title": "Example Code 1",
      "explanation": "This code example demonstrates...",
      "content": "function example1() {...}",
      "tags": ["JavaScript", "Example"],
      "authorId": 123
    },
    {

```



```
        "id": 2,
        "title": "Example Code 2",
        "explanation": "Another code example...",
        "content": "function example2() {...}",
        "tags": ["JavaScript", "Tutorial"],
        "authorId": 124
      }
    ],
    "total": 20
  }
```

Sample Request (GLOBAL, MULTI-CATEGORY SEARCH)

```
GET
http://localhost:3000/api/codes/search?category=title,explanation&searchTerm=example&page=1&limit=10&scope=global
```

200 Sample Response (GLOBAL, MULTI-CATEGORY SEARCH)

```
{
  "results": [
    {
      "id": 1,
      "title": "Example Code 1",
      "explanation": "This code demonstrates...",
      "content": "function example1() {...}",
      "tags": ["JavaScript", "Example"],
      "authorId": 123
    },
    {
      "id": 2,
      "title": "Code 2",
      "explanation": "Another code example...",
      "content": "function example2() {...}",
      "tags": ["JavaScript", "Tutorial"],
      "authorId": 124
    }
  ],
  "total": 20
}
```

Sample Request (LOCAL, GENERAL SEARCH)

```
GET
http://localhost:3000/api/codes/search?page=1&limit=10&scope=local
```

200 Sample Response (LOCAL, GENERAL SEARCH)

```
{
  "results": [
    {
      "id": 1,
      "title": "Example Code 1",
      "explanation": "This code example demonstrates...",
      "content": "function example1() {...}",
      "tags": ["JavaScript", "Example"],
      "authorId": 123
    },
    {
      "id": 2,
      "title": "Example Code 2",
      "explanation": "Another code example...",
      "content": "function example2() {...}",
      "tags": ["JavaScript", "Tutorial"],
      "authorId": 123
    }
  ],
  "total": 20
}
```

Sample Request (LOCAL, SINGLE-CATEGORY SEARCH)

```
GET
http://localhost:3000/api/codes/api/codes/search?category=title&searchTerm=example&page=1&limit=10&scope=local
```

200 Sample Response (LOCAL, SINGLE-CATEGORY SEARCH)

```
{
  "results": [
    {
```

```

        "id": 1,
        "title": "Example Code 1",
        "explanation": "This code example demonstrates...",
        "content": "function example1() {...}",
        "tags": ["JavaScript", "Example"],
        "authorId": 123
    },
    {
        "id": 2,
        "title": "Example Code 2",
        "explanation": "Another code example...",
        "content": "function example2() {...}",
        "tags": ["JavaScript", "Tutorial"],
        "authorId": 123
    }
],
"total": 20
}

```

Sample Request (LOCAL, MULTI-CATEGORY SEARCH)

```

GET
http://localhost:3000/api/codes/search?category=title,explanation&searchTerm=example&page=1&limit=10&scope=local

```

200 Sample Response (LOCAL, MULTI-CATEGORY SEARCH)

```

{
  "results": [
    {
      "id": 1,
      "title": "Example Code 1",
      "explanation": "This code demonstrates...",
      "content": "function example1() {...}",
      "tags": ["JavaScript", "Example"],
      "authorId": 123
    },
    {
      "id": 2,
      "title": "Code 2",

```

```
        "explanation": "Another code example...",
        "content": "function example2() {...}",
        "tags": ["JavaScript", "Tutorial"],
        "authorId": 123
    }
],
"total": 20
}
```

Reports

Object Base Url: <http://localhost:3000/api/reports>

Description

In Scriptorium, **users** can report **blogs** or **comments** they find inappropriate, helping to remove harmful content from the platform. When submitting a **report**, **users** have the option to include an explanation to clarify the reason for their **report**, enabling more effective moderation.

For system administrators, **reports** are prioritized by the total number of submissions each **blog** or **comment** has received, making it easier to identify and manage flagged content. Once deemed inappropriate, administrators can hide this content from public view. Hidden content is visible only to its author, marked with a flag indicating the report, but the author is restricted from editing it.

Database Schema

Component	Type	Description & Example	Relational Query & Database
id	Int (<i>default</i>)	The Prisma database ID of the code template <i>Example: 1</i>	N/A
explanation	String	The explanation of the template <i>Example: Seamlessly prints Hello World</i>	N/A
parentType	String	The type of content <i>Example: Comment or Blog</i>	N/A
parentId	Int	The Prisma database ID of the reported content	N/A
blog	Blog?	The Prisma database ID of the blog that is reported	“BlogReports” <i>DB: Blog</i>
comment	Comment?	The Prisma database ID of the comment that is reported	“CommentReports” <i>DB: Comment</i>

Endpoints

Report Content

Method(s): **POST**

Relative Path Endpoint: /reportContent

Description

This endpoint allows a **user** to report a content (comment or blog).

Sample Request

```
POST http://localhost:3000/api/reports/reportContent
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json
{
  "explanation": "This is an example explanation for the report.",
  "parentId": 2,
  "parentType": "blog"
}
```

201 Sample Response

```
{
  "message": "Report submitted successfully",
  "report": {
    "id": 5,
    "explanation": "This is an example explanation for the report.",
    "parentId": 2,
    "parentType": "blog"
  }
}
```

Report Management

Method(s): **GET**

Relative Path Endpoint: **/:id**

Description

This endpoint allows a **user** or **admin** to view a report.

Query Parameters

Parameter	Request Method	Type	Description & Example	In Path/Query
id	GET	Int	Requested Prisma database report ID <i>Example: 1</i>	Path

Sample Request

```
GET http://localhost:3000/api/reports/5
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json
```

201 Sample Response

```
{
  "report": {
    "id": 5,
    "explanation": "This is an example explanation for the
report.",
    "parentId": 2,
    "parentType": "blog"
  }
}
```

Hide

Method(s): **PATCH**

Relative Path Endpoint: /hideReportedContent

Description

This endpoint allows an **admin** to hide reported content.

Query Parameters

N/A

Sample Request

```
PATCH http://localhost:3000/api/reports/hideReportedContent
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json
{
  "contentId": "5",
  "contentType": "blog"
}
```

201 Sample Response

```
{
```

```
"message": "Content has been hidden successfully"
}
```

Sort

Method(s): **GET**

Relative Path Endpoint: /sortReportedContent

Description

This endpoint allows an **admin** to sort content by most reports.

Query Parameters

Parameter	Request Method	Type	Description & Example	In Path/Query
page	GET	Int	Requested page number <i>Example: 1</i>	Query
limit	GET	Int	Limit to number of report entries per page <i>Example: 10</i>	Query

Request Details

N/A

Sample Request

```
GET http://localhost:3000/api/reports/api/reports?page=1&limit=10
Authorization: Bearer <JWT_Token>
```

201 Sample Response

```
{
  "page": 1,
  "limit": 10,
  "total": 8,
  "data": [
    {
      "id": 2,
      "authorId": 5,
      "title": "Healthy Living",

```



```
    "description": "Practical tips for a healthier
lifestyle.",
    "createdAt": "2024-11-03T07:10:07.927Z",
    "updatedAt": "2024-11-03T07:10:07.927Z",
    "upvotes": 20,
    "downvotes": 2,
    "totalVotes": 0,
    "hidden": false,
    "reports": [
      {
        "id": 2,
        "explanation": "This is an example explanation
for the report.",
        "parentId": 2,
        "parentType": "comment"
      },
      {
        "id": 3,
        "explanation": "This is an example explanation
for the report.",
        "parentId": 2,
        "parentType": "blog"
      },
      {
        "id": 5,
        "explanation": "This is an example explanation
for the report.",
        "parentId": 2,
        "parentType": "blog"
      }
    ],
    "_count": {
      "reports": 3
    },
    "type": "blog",
    "reportCount": 3
  },
  {
    "id": 2,
```

```
    "parentId": null,
    "createdAt": "2024-11-03T07:10:07.930Z",
    "updatedAt": "2024-11-03T07:59:13.533Z",
    "upvotes": 8,
    "downvotes": 1,
    "content": "Great health tips!",
    "authorId": 5,
    "blogId": 2,
    "hidden": true,
    "reports": [
      {
        "id": 2,
        "explanation": "This is an example explanation
for the report.",
        "parentId": 2,
        "parentType": "comment"
      },
      {
        "id": 3,
        "explanation": "This is an example explanation
for the report.",
        "parentId": 2,
        "parentType": "blog"
      },
      {
        "id": 5,
        "explanation": "This is an example explanation
for the report.",
        "parentId": 2,
        "parentType": "blog"
      }
    ],
    "_count": {
      "reports": 3
    },
    "type": "comment",
    "reportCount": 3
  }
}
```

Tags

*NO ENDPOINT (REFERENCED THROUGH **Blogs** and **CodeTemplates**)*

Description

Tags allow Scriptorium users to categorize and easily find **blogs** and **code templates** based on relevant topics. **Users** can create new **tags** to enhance content discoverability, modify existing ones to better reflect themes, and utilize **tags** to filter searches effectively. This system fosters a more organized and collaborative environment, making it simpler for users to share and access knowledge within the community.

Database Schema

Component	Type	Description & Example	Relational Query & Database
id	Int (<i>default</i>)	The Prisma database ID of the blog <i>Example: 1</i>	N/A
name	String (<i>unique</i>)	The name of the tag <i>Example: Java</i>	N/A
blogs	Blog[]	The list containing all the blogs associated with the tag	“BlogTags” <i>DB: Blog</i>
codeTemplates	CodeTemplate[]	The list containing all the code templates associated with the tag	“CodeTemplateTags” <i>DB: CodeTemplate</i>

Execute

Method(s): **POST**

Absolute Url: `http://localhost:3000/api/execution/execute`

Description

An additional endpoint enabling **users** to execute code written in a **code template** locally in the specified language. If the **code template** ID is not provided, the API takes language, code and input as a request.

Query Parameters

N/A

Sample Request

```
POST http://localhost:3000/api/execution/execute
Content-Type: application/json
{
  "codeTemplate": "1"
}
```

200 Sample Response

```
{
  "output": "Hello, JavaScript!\n",
  "error": ""
}
```