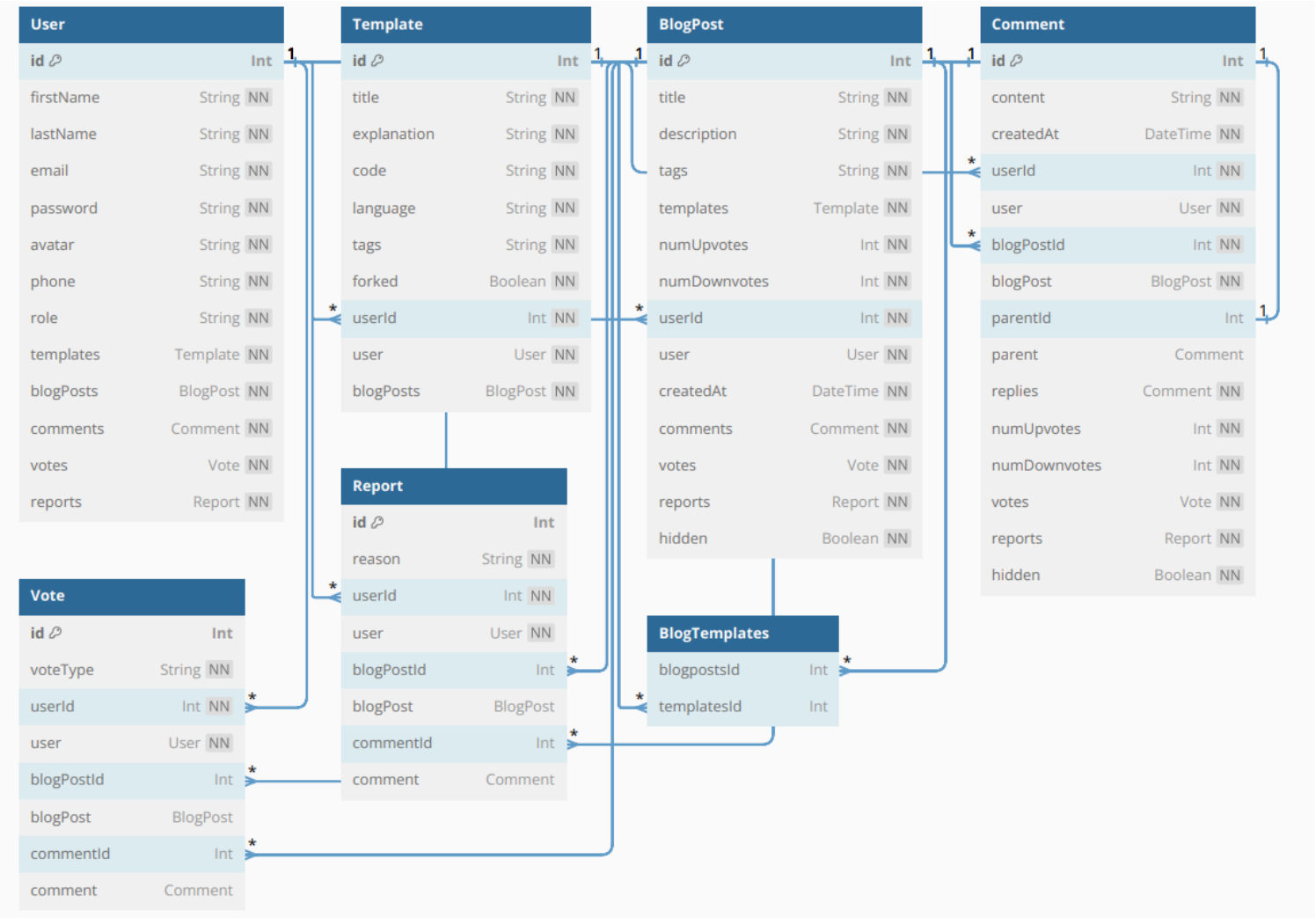
1. Model Design: (Attached pdf for better readability)



Explanations:

- **NN**: Not Null, meaning they cannot be empty and are required.
- Lines connecting tables represent a relationship
    - 1-1 - one-to-one relationship
    - 1:* - one-to-many
        - For example, 1 next to User and * next to Template means a User can have many Templates
            - A User can also have many BlogPosts, Comments, Votes, and Reports
        - A BlogPost can have many Comments, Votes, and Reports
        - A Comment can have many replies
            - parentId being not null means this Comment is a reply. The value of the parentId is the id of the comment it's replying to.

- Many-to-Many:
  - For example, BlogPost and Template have a many-to-many relationship, represented by the BlogTemplates join table. In Prisma Schema, it looks like this:

```
model Template {
  id          Int        @id @default(autoincrement())
  title       String
  explanation String
  code        String
  language    String
  tags        String     // Comma separated
  forked      Boolean    @default(false)
  userId      Int
  user        User       @relation(fields: [userId], references: [id])
  blogPosts   BlogPost[] @relation("BlogTemplates")
}

model BlogPost {
  id           Int        @id @default(autoincrement())
  title        String
  description  String
  tags         String
  templates    Template[] @relation("BlogTemplates")
  numUpvotes   Int        @default(0)
```

  - A BlogPost can be linked to many templates, and a Template can be linked to many blog posts.

2. API Endpoints list (shown in a table grouped by folders for readability and to save space):

blogs	code	comments	reporting	templates	users
/api/blogs/create	/api/code/execute	/api/comments/create	/api/reporting/create	/api/templates/[id]	/api/users/login
/api/blogs/delete		/api/comments/sort	/api/reporting/hideContent	/api/templates/delete	/api/users/logout
/api/blogs/edit		/api/comments/vote	/api/reporting/sortBlogReports	/api/templates/edit	/api/users/profile
/api/blogs/sort			/api/reporting/sortCommentReports	/api/templates/fork	/api/users/refresh
/api/blogs/viewAll				/api/templates/save	/api/users/register
/api/blogs/vote				/api/templates/viewAll	
				/api/templates/viewOwn	

Endpoint Details

Admin Account

When you run the startup.sh file, an admin account will automatically be created for you with the following information:

```
firstName: "Test"
lastName: "Admin"
email: "adminUser@gmail.com"
password: "123"
avatar: "Test Avatar"
phone: "1234567890"
role: "ADMIN"
```

NOTE:

You will need to import BOTH the postman collection called “Scriptorium.postman_collection” AND the postman environment called “New Environment.postman_environment”.

/api/users/register

- Short description:
  - This API endpoint allows the creation of a new user account with fields for personal information, a hashed password, and a default avatar.
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
firstName	String	Y	The user's first name.
lastName	String	Y	The user's last name.
email	String	Y	The user's email address. Must be unique.
password	String	Y	The user's password. It will be hashed before storage.
phone	String	Y	The user's phone number.
role	String	Y	The user's role in the system.

- An example request
  - http://localhost:3000/api/users/register

```
{
    "firstName": "John",
    "lastName": "Doe",
    "email": "JohnDoe@gmail.com",
    "password": "123",
    "phone": "123",
    "role": "USER"
}
```


- An example response


```
{
    "message": "User created successfully",
    "newUser": {
        "id": 1,
        "firstName": "John",
        "lastName": "Doe",
        "email": "JohnDoe@gmail.com",
        "password": "$2b$10$KpxU1XdCiVgVQxNSyQPF9.9/EWL2PZwgBu0MtZ5AFfFkwtU47r/1W",
        "avatar": "/images/default-avatar.png",
        "phone": "123",
        "role": "USER"
    }
}
```


/api/users/login

- Short description:
  - This API endpoint allows users to log in by verifying their email and password. On successful authentication, it returns an access token and a refresh token.
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
email	String	Y	The user's email address. Must be unique.
password	String	Y	The user's password.

- An example request
    - http://localhost:3000/api/users/login

```
{
    "email": "JohnDoe@gmail.com",
    "password": "123"
}
```

- An example response

```
{
    "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6IkpvaG5Eb2VAZ21haWwuY29tIiwicm9sZSI6IlVTRVIiLCJpYXQiOjE3MzA0Mjg5MDYsImV4cCI6MTczMDQzMDEwNn0.kQ_tIjG4tkBuuhNiaDiMqqpnpM-13no-ZZl4wygvdOw",
    "refreshToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6IkpvaG5Eb2VAZ21haWwuY29tIiwicm9sZSI6IlVTRVIiLCJpYXQiOjE3MzA0Mjg5MDYsImV4cCI6MTczMDUxNTMwNn0.cQ_6UkV84MNUBIBhwIAGui8zBAWp_PGHVrdmAA_aBiM"
}
```

/api/users/profile

- Short description:
    - This API endpoint allows an authenticated user to update their profile information, including their password, phone number, and avatar image.
- Allowed methods:
    - PUT
- Payload:
    - Request Body:

Parameter	Type	Required	Description
phone	String	N (then password or avatar)	The user's email address. Must be unique.
password	String	N (then phone or avatar)	The user's password.
avatar	File	N (then phone or password)	Any avatar in the public/avatars

- An example request
    - http://localhost:3000/api/users/profile

form-data

phone	999-999-9999
password	123
avatar	

- An example response


```
{
    "message": "Profile updated",
    "updatedUser": {
        "id": 1,
        "firstName": "John",
        "lastName": "Doe",
        "email": "JohnDoe@gmail.com",
        "password": "$2b$10$VOFenNwPqXm9uDdLvpV/Vu.AdxsN/Y2psYg9pL99g54IYlzLnOSue",
        "avatar": "/images/default-avatar.png",
        "phone": "999-999-9999",
        "role": "USER"
    }
}
```


/api/users/refresh

- Short description:
  - This API endpoint generates a new access token using a valid refresh token.
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
refreshToken	String	Y	The user's refresh token.

- An example request
  - http://localhost:3000/api/users/refresh

```
{
    "refreshToken": "{{refreshToken}}"
}
```

- An example response

```
{
    "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiVVNFUiIsImlhdCI6MTczMDUwMjE2MywiZXhwIjoxNzMwNTAzMzYzfQ.9mVv-u8bVZLtYqMp3q88kD62OOED7penj
PmwbAhz_KI"
}
```


/api/users/logout

- Short description:
  - This API endpoint logs out the user by clearing the refresh token cookie.
- Allowed methods:
  - POST
- Payload:
  - None
- An example request
  - http://localhost:3000/api/users/logout
- An example response


```
{
    "message": "Logged out successfully"
}
```


/api/blogs/create:

- Short description:
  - Allows authenticated users to create a new blog post with a title, description, tag(s), and (optional) links to code templates.
- Allowed methods:
  - POST
- Payload:
  - Expects a JSON object in the request body
  - Request Body:

Parameter	**Type**	**Required (Y/N)**
title	string	Y
description	string	Y
tags	string	Y
templateIds	array	N

userEmail	string	Y

- An example request:

  `POST /api/blogs/create`

  http://localhost:3000/api/blogs/create

  `Content-Type: application/json`

```
{
    "title": "Introduction to JavaScript",
    "description": "A blog post about JavaScript basics and applications.",
    "tags": "javascript,programming",
    "templateIds": [],
    "userEmail": "JohnDoe@gmail.com"
}
```

- An example response:

```
{
    "id": 3,
    "title": "Introduction to JavaScript",
    "description": "A comprehensive blog post about JavaScript basics and applications.",
    "tags": "javascript,web development,programming",
    "numUpvotes": 0,
    "numDownvotes": 0,
    "userId": 1,
    "createdAt": "2024-11-01T01:02:06.921Z",
    "hidden": false
}
```

/api/blogs/delete

- Short description:
  - Allows an authenticated user to delete a blog post they created.
- Allowed methods:
  - DELETE
- Payload:

- Request Query:

Parameter	Type	Required	Description
blogID	int	Y	The ID of the blog user wants to delete.

- Request Body:

Parameter	Type	Required	Description
userEmail	string	Y	Email of the user who wants to delete their blog post.

- An example request: `DELETE` http://localhost:3000/api/blogs/delete?blogID=1

```
{
    "userEmail": "JohnDoe@gmail.com"
}
```

- An example response:

```
{
    "message": "Blog post deleted successfully"
}
```

/api/blogs/edit

- Short description:
  - Allows an authenticated user to edit the details of a blog post they created (title, description, tags, and template IDs). Hidden blog posts cannot be edited.
- Allowed methods:
  - PUT
- Payloads:
  - Request Query:

Parameter	Type	Required	Description
blogID	int	Y	The ID of the blog user wants to edit.

  - Request Body:

Parameter	Type	Required
title	string	Y

description	string	Y
tags	string	Y
templateIDs	array	N
userEmail	string	Y

- An example request: PUT http://localhost:3000/api/blogs/edit?blogID=3

```
{
    "title": "Updated Blog Title",
    "description": "Updated description of the blog post.",
    "tags": "updated,blog,post",
    "templateIds": [1],
    "userEmail": "JohnDoe@gmail.com"
}
```

- An example response

```
{
    "id": 3,
    "title": "Updated Blog Title",
    "description": "Updated description of the blog post.",
    "tags": "updated,blog,post",
    "numUpvotes": 0,
    "numDownvotes": 0,
    "userId": 1,
    "createdAt": "2024-11-01T01:02:06.921Z",
    "hidden": false
}
```

/api/blogs/sort

- Short description:
    - Allows visitors (and users) to view and sort a list of blog posts by either most liked (number of upvotes), most disliked (number of downvotes), or most recent.
- Allowed methods:
    - GET
- Payload:
    - Request Query:

Parameter	Type	Required	Description
sortBy	string	Y	Sorting criteria; can be either "mostLiked", "mostDisliked", or "mostRecent".
page	int	N	For pagination, specify page number. Default = 1.
limit	int	N	For pagination, specify the number of blog posts per page. Default = 10.

- An example request: `GET` http://localhost:3000/api/blogs/sort?sortBy=mostLiked&page=1&limit=3

- An example response


```
[
    {
        "id": 3,
        "title": "Updated Blog Title",
        "description": "Updated description of the blog post.",
        "tags": "updated,blog,post",
        "numUpvotes": 1,
        "numDownvotes": 0,
        "userId": 1,
        "createdAt": "2024-11-01T01:02:06.921Z",
        "hidden": false,
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "templates": [
            {
                "id": 1,
                "title": "Code Template 2",
                "explanation": "This is a test code template!",
                "code": "console.log('Hello World!')",
                "language": "javascript",
                "tags": "javascript,coding",
                "forked": false,
                "userId": 1
            }
        ],
```

```
        "comments": []
    },
    {
        "id": 2,
        "title": "Introduction to JavaScript",
        "description": "A blog post about JavaScript basics and applications.",
        "tags": "javascript,programming",
        "numUpvotes": 0,
        "numDownvotes": 0,
        "userId": 1,
        "createdAt": "2024-11-01T01:01:22.164Z",
        "hidden": false,
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "templates": [],
        "comments": []
    },
    {
        "id": 4,
        "title": "Introduction to JavaScript",
        "description": "A blog post about JavaScript basics and applications.",
        "tags": "javascript,programming",
        "numUpvotes": 0,
        "numDownvotes": 1,
        "userId": 1,
        "createdAt": "2024-11-01T22:24:52.166Z",
        "hidden": false,
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "templates": [],
        "comments": []
    }
]
```

/api/blogs/viewAll

- Short description:
  - Allows visitors (and other users) to view and search for blog posts.
- Allowed methods:
  - GET
- Payload:
  - Request Query:

Parameter	Type	Required	Description
page	int	N	For pagination, specify page number. Default = 1.
limit	int	N	For pagination, specify the number of blog posts per page. Default = 10.
search	string	N	Searches blog posts by title, description, tags, or templates.

  - Request Body:

Parameter	Type	Required
userEmail	string	N

- An example request: `GET` http://localhost:3000/api/blogs/viewAll
  (empty body - to show what a visitor would see)
- An example response


```
[
    {
        "id": 4,
        "title": "Introduction to JavaScript",
        "description": "A blog post about JavaScript basics and applications.",
        "tags": "javascript,programming",
        "numUpvotes": 0,
        "numDownvotes": 1,
        "userId": 1,
        "createdAt": "2024-11-01T22:24:52.166Z",
        "hidden": false,
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
```

```
        "templates": [],
        "comments": []
    },
    {
        "id": 3,
        "title": "Updated Blog Title",
        "description": "Updated description of the blog post.",
        "tags": "updated,blog,post",
        "numUpvotes": 1,
        "numDownvotes": 0,
        "userId": 1,
        "createdAt": "2024-11-01T01:02:06.921Z",
        "hidden": false,
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "templates": [
            {
                "id": 1,
                "title": "Code Template 2",
                "explanation": "This is a test code template!",
                "code": "console.log('Hello World!')",
                "language": "javascript",
                "tags": "javascript,coding",
                "forked": false,
                "userId": 1
            }
        ],
        "comments": []
    },
    {
        "id": 2,
        "title": "Introduction to JavaScript",
        "description": "A blog post about JavaScript basics and applications.",
        "tags": "javascript,programming",
        "numUpvotes": 0,
```

```
        "numDownvotes": 0,
        "userId": 1,
        "createdAt": "2024-11-01T01:01:22.164Z",
        "hidden": false,
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "templates": [],
        "comments": []
    }
]
```

/api/blogs/vote

- Short description:
  - Allows authenticated users to rate a blog post by giving it an upvote or downvote. Users can only vote once per post and have the option to change their vote type.
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
blogPostId	int	Y	The ID of the blog post to vote on.
voteType	string	Y	Type of vote ("upvote" or "downvote").
userEmail	string	Y	Email of the user who wants to delete their blog post.

- An example request: `POST` http://localhost:3000/api/blogs/vote

```
{
    "blogPostId": 3,
    "voteType": "upvote",
    "userEmail": "JohnDoe@gmail.com"
}
```

- An example response

```
{
```

```
    "id": 3,
    "title": "Updated Blog Title",
    "description": "Updated description of the blog post.",
    "tags": "updated,blog,post",
    "numUpvotes": 1,
    "numDownvotes": 0,
    "userId": 1,
    "createdAt": "2024-11-01T01:02:06.921Z",
    "hidden": false
}
```


If user tries to upvote again (ie, send the same request as above again):


```
}
 "message": "You already upvoted this post."
}
```


/api/code/execute

- Short description:
  - Allows visitors to write, execute, and test code in multiple programming languages (C, C++, Java, Python, JavaScript) with optional input through standard input (stdin). Supports syntax highlighting, error handling for runtime errors, compile errors, and various signals (e.g., seg faults).
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
code	string	Y	The code to execute.
language	string	Y	The programming language (c, cpp, java, python, or javascript).
input	string	N	Stdin for program execution.
templateId	int	N	Template ID. If provided, the code and language from the template will be used.

- An example request
  - (A bunch of other test cases + expected results are in the execute_test.txt in the __tests__ folder of the repo)

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

```
{
    "code": "public class Main { public static int add(int a, int b) { return a + b; } public static void main(String[] args) { int result = add(4, 5); System.out.println(\"Result: \" + result); } }",
    "language": "java",
    "input": ""
}
```

- An example response

```
{
   "output":"Result: 9\n"}
}
```

/api/comments/create

- Short description:
  - Allows authenticated users to write comments on a blog post or reply to an existing comment.
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
content	string	Y	The content of the comment/reply.
userEmail	string	Y	The email of the user making the comment.
blogPostId	int	Y	The blog post under which the comment is being made.
parentId	int	N	For replies, the ID of the parent comment. If not provided, the comment is not a reply.

- An example request: `POST` http://localhost:3000/api/comments/create

```
{
    "content": "This is a comment on a blog post.",
    "userEmail": "JohnDoe@gmail.com",
    "blogPostId": 2
}
```

- An example response

```
{
    "id": 1,
```

```
    "content": "This is a comment on a blog post.",
    "createdAt": "2024-11-01T23:24:35.811Z",
    "userId": 1,
    "blogPostId": 2,
    "parentId": null,
    "numUpvotes": 0,
    "numDownvotes": 0,
    "hidden": false
}
```


- Reply (request):


```
{
    "content": "This is a reply to the original comment on blog post 2.",
    "userEmail": "JohnDoe@gmail.com",
    "blogPostId": 2,
    "parentId": 1
}
```


- Reply (response):


```
{
    "id": 2,
    "content": "This is a reply to the original comment on blog post 2.",
    "createdAt": "2024-11-01T23:29:37.562Z",
    "userId": 1,
    "blogPostId": 2,
    "parentId": 1,
    "numUpvotes": 0,
    "numDownvotes": 0,
    "hidden": false
}
```


/api/comments/vote

- Short description:
  - Allows authenticated users to rate a comment by giving it an upvote or a downvote. If the user has already voted, they can only change their vote type, can't vote several times.
- Allowed methods:
  - POST

- Payload:
  - Request Body:

Parameter	Type	Required	Description
commentId	int	Y	The ID of the comment being voted on.
voteType	string	Y	The type of vote ("upvote" or "downvote").
userEmail	string	Y	Email of the authenticated user submitting the vote.

- An example request: `POST` http://localhost:3000/api/comments/vote

```
{
    "commentId": 1,
    "voteType": "upvote",
    "userEmail": "JohnDoe@gmail.com"
}
```

- An example response

```
{
    "id": 1,
    "content": "This is a comment on a blog post.",
    "createdAt": "2024-11-01T23:24:35.811Z",
    "userId": 1,
    "blogPostId": 2,
    "parentId": null,
    "numUpvotes": 1,
    "numDownvotes": 0,
    "hidden": false
}
```

/api/comments/sort

- Short description:
  - Allows visitors to sort comments on a blog post by either most liked, most disliked, or most recent. Sorting applies separately to main comments and their replies.
- Allowed methods:
  - GET
- Payload:
  - Request Query:

Parameter	Type	Required	Description
blogPostId	int	Y	The ID of the blog post for which to retrieve and sort comments.
sortByMain	string	N	Sorting method for main comments ("mostLiked," "mostDisliked," or "mostRecent"). Default = "mostLiked".
sortByReplies	string	N	Sorting method for replies within each main comment ("mostLiked," "mostDisliked," or "mostRecent"). Default = "mostRecent".
page	int	N	For pagination, specify page number. Default = 1.
limit	int	N	For pagination, specify the number of comments per page. Default = 10.

- An example request:
  `GET http://localhost:3000/api/comments/sort?blogPostId=2&sortByMain=mostLiked&sortByReplies=mostLiked`
- An example response:


```
[
    {
        "id": 1,
        "content": "This is a comment on a blog post.",
        "createdAt": "2024-11-01T23:24:35.811Z",
        "userId": 1,
        "blogPostId": 2,
        "parentId": null,
        "numUpvotes": 1,
        "numDownvotes": 0,
        "hidden": false,
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "replies": [
            {
                "id": 2,
                "content": "This is a reply to the original comment on blog post 2.",
                "createdAt": "2024-11-01T23:29:37.562Z",
                "userId": 1,
                "blogPostId": 2,
```

```
            "parentId": 1,
            "numUpvotes": 1,
            "numDownvotes": 0,
            "hidden": false,
            "user": {
                "firstName": "John",
                "lastName": "Doe"
            }
        },
        {
            "id": 5,
            "content": "This is reply2 to the original comment on blog post 2.",
            "createdAt": "2024-11-01T23:39:57.128Z",
            "userId": 1,
            "blogPostId": 2,
            "parentId": 1,
            "numUpvotes": 0,
            "numDownvotes": 1,
            "hidden": false,
            "user": {
                "firstName": "John",
                "lastName": "Doe"
            }
        },
        {
            "id": 6,
            "content": "This is reply3 to the original comment on blog post 2.",
            "createdAt": "2024-11-01T23:40:24.415Z",
            "userId": 1,
            "blogPostId": 2,
            "parentId": 1,
            "numUpvotes": 0,
            "numDownvotes": 0,
            "hidden": false,
            "user": {
                "firstName": "John",
                "lastName": "Doe"
```

```
                }
            }
        ]
    },
    {
        "id": 3,
        "content": "This is comment2 on blog post 2.",
        "createdAt": "2024-11-01T23:33:26.951Z",
        "userId": 1,
        "blogPostId": 2,
        "parentId": null,
        "numUpvotes": 0,
        "numDownvotes": 1,
        "hidden": false,
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "replies": []
    }
]
```


/api/reporting/create

- Short description:
  -
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
blogPostId	int	N (then commentID is required)	ID of the blog post being reported.
commentId	int	N (then blogPostId is required)	ID of the comment being reported.
reason	string	Y	The reason for the report, must be at least 3 characters.

userEmail	string	Y	Email of the authenticated user submitting the report.

- An example request: `POST` http://localhost:3000/api/reporting/create

```
{
    "blogPostId": 1,
    "reason": "Bad",
    "userEmail": "JohnDoe@gmail.com"
}
```

- An example response:

```
{
    "id": 1,
    "reason": "Bad",
    "userId": 1,
    "blogPostId": 1,
    "commentId": null
}
```

/api/reporting/hideContent

- Short description:
  - This API endpoint allows an admin to hide a blog post or comment by setting its `hidden` attribute to `true`.
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
blogPostId	int	N (then commentID is required)	ID of the blog post being reported.
commentId	int	N (then blogPostId is required)	ID of the comment being reported.

- An example request
  - http://localhost:3000/api/reporting/hideContent

```
{
    "blogPostId": 1
}
```

- An example response


```
{
    "message": "Blog post hidden successfully",
    "blogPost": {
        "id": 1,
        "title": "test",
        "description": "tester des",
        "tags": "tag1",
        "numUpvotes": 0,
        "numDownvotes": 0,
        "userId": 1,
        "createdAt": "2024-10-31T00:40:31.170Z",
        "hidden": true
    }
}
```


/api/reporting/sortBlogReports

- Short description:
  - This API endpoint allows an admin to fetch a paginated list of blog posts, ordered by the number of reports in descending order.
- Allowed methods:
  - GET
- Payload:
  - Request Query:

Parameter	Type	Required	Description
page	int	N	For pagination, specify page number. Default = 1.
limit	int	N	For pagination, specify the number of blog posts per page. Default = 10.

- An example request
  - http://localhost:3000/api/reporting/sortBlogReports?page=1&limit=10
- An example response


```
[
    {
        "id": 1,
        "title": "test",
        "description": "tester des",
        "tags": "tag1",
```

```
        "numUpvotes": 0,
        "numDownvotes": 0,
        "userId": 1,
        "createdAt": "2024-10-31T00:40:31.170Z",
        "hidden": true,
        "_count": {
            "reports": 11
        },
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "templates": [
            {
                "id": 2,
                "title": "Code Template 2",
                "explanation": "This is a test code template!",
                "code": "console.log('Hello World!')",
                "language": "javascript",
                "tags": "javascript,coding",
                "forked": false,
                "userId": 1
            }
        ],
        "comments": []
    }
]
```


/api/reporting/sortCommentReports

- Short description:
  - This API endpoint allows an admin to fetch a paginated list of comments, ordered by the number of reports in descending order.
- Allowed methods:
  - GET
- Payload:
  - Request Query:

Parameter	Type	Required	Description

page	int	N	For pagination, specify page number. Default = 1.
limit	int	N	For pagination, specify the number of blog posts per page. Default = 10.

- An example request: `GET` http://localhost:3000/api/reporting/sortCommentReports
- An example response


```
[
    {
        "id": 5,
        "content": "This is comment2 on blog post 2.",
        "createdAt": "2024-11-02T01:20:23.235Z",
        "userId": 2,
        "blogPostId": 2,
        "parentId": null,
        "numUpvotes": 0,
        "numDownvotes": 0,
        "hidden": false,
        "_count": {
            "reports": 2
        },
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "blogPost": {
            "id": 2,
            "title": "Introduction to JavaScript",
            "description": "A blog post about JavaScript basics and applications.",
            "tags": "javascript,programming",
            "numUpvotes": 0,
            "numDownvotes": 0,
            "userId": 2,
            "createdAt": "2024-11-02T01:20:21.830Z",
            "hidden": false
        },
        "replies": []
    },
```

```
{
    "id": 1,
    "content": "This is a comment on a blog post.",
    "createdAt": "2024-11-02T01:20:22.893Z",
    "userId": 2,
    "blogPostId": 2,
    "parentId": null,
    "numUpvotes": 1,
    "numDownvotes": 0,
    "hidden": false,
    "_count": {
        "reports": 0
    },
    "user": {
        "firstName": "John",
        "lastName": "Doe"
    },
    "blogPost": {
        "id": 2,
        "title": "Introduction to JavaScript",
        "description": "A blog post about JavaScript basics and applications.",
        "tags": "javascript,programming",
        "numUpvotes": 0,
        "numDownvotes": 0,
        "userId": 2,
        "createdAt": "2024-11-02T01:20:21.830Z",
        "hidden": false
    },
    "replies": [
        {
            "id": 3,
            "content": "This is a reply to the original comment on blog post 2.",
            "createdAt": "2024-11-02T01:20:23.046Z",
            "userId": 2,
            "blogPostId": 2,
            "parentId": 1,
            "numUpvotes": 0,
```

```
            "numDownvotes": 1,
            "hidden": false
        },
        {
            "id": 4,
            "content": "This is reply2 to the original comment on blog post 2.",
            "createdAt": "2024-11-02T01:20:23.141Z",
            "userId": 2,
            "blogPostId": 2,
            "parentId": 1,
            "numUpvotes": 0,
            "numDownvotes": 1,
            "hidden": false
        }
    ]
},
{
    "id": 2,
    "content": "This is a bad comment and will get reported.",
    "createdAt": "2024-11-02T01:20:22.985Z",
    "userId": 2,
    "blogPostId": 3,
    "parentId": null,
    "numUpvotes": 1,
    "numDownvotes": 0,
    "hidden": false,
    "_count": {
        "reports": 0
    },
    "user": {
        "firstName": "John",
        "lastName": "Doe"
    },
    "blogPost": {
        "id": 3,
        "title": "Updated Blog Title",
        "description": "Updated description of the blog post.",
```

```
            "tags": "updated,blog,post",
            "numUpvotes": 1,
            "numDownvotes": 0,
            "userId": 2,
            "createdAt": "2024-11-02T01:20:21.909Z",
            "hidden": false
        },
        "replies": []
    },
    {
        "id": 3,
        "content": "This is a reply to the original comment on blog post 2.",
        "createdAt": "2024-11-02T01:20:23.046Z",
        "userId": 2,
        "blogPostId": 2,
        "parentId": 1,
        "numUpvotes": 0,
        "numDownvotes": 1,
        "hidden": false,
        "_count": {
            "reports": 0
        },
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "blogPost": {
            "id": 2,
            "title": "Introduction to JavaScript",
            "description": "A blog post about JavaScript basics and applications.",
            "tags": "javascript,programming",
            "numUpvotes": 0,
            "numDownvotes": 0,
            "userId": 2,
            "createdAt": "2024-11-02T01:20:21.830Z",
            "hidden": false
        },
```

```
        "replies": []
    },
    {
        "id": 4,
        "content": "This is reply2 to the original comment on blog post 2.",
        "createdAt": "2024-11-02T01:20:23.141Z",
        "userId": 2,
        "blogPostId": 2,
        "parentId": 1,
        "numUpvotes": 0,
        "numDownvotes": 1,
        "hidden": false,
        "_count": {
            "reports": 0
        },
        "user": {
            "firstName": "John",
            "lastName": "Doe"
        },
        "blogPost": {
            "id": 2,
            "title": "Introduction to JavaScript",
            "description": "A blog post about JavaScript basics and applications.",
            "tags": "javascript,programming",
            "numUpvotes": 0,
            "numDownvotes": 0,
            "userId": 2,
            "createdAt": "2024-11-02T01:20:21.830Z",
            "hidden": false
        },
        "replies": []
    }
]
```


/api/templates/save

- Short description:
    - This API endpoint allows an authenticated user to create and save a new code template.

- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
title	String	Y	The title of the template.
explanation	String	Y	A description or explanation for the template.
language	String	Y	The programming language of the template.
code	String	Y	The code content of the template.
tags	Array of strings	Y	An array of tags for categorizing the template.

- An example request
  - http://localhost:3000/api/templates/save

```
{
    "title": "Code Template",
    "explanation": "This is a test code template!",
    "language": "javascript",
    "code": "console.log('Hello World!')",
    "tags": ["javascript", "coding"]
}
```

- An example response

```
{
    "message": "Template saved",
    "newTemplate": {
        "id": 1,
        "title": "Code Template",
        "explanation": "This is a test code template!",
        "code": "console.log('Hello World!')",
        "language": "javascript",
        "tags": "javascript,coding",
        "forked": false,
        "userId": 1
    }
```

```
}
```

/api/templates/fork

- Short description:
  - This API endpoint allows an authenticated user to fork a specific template. The forked template will be created under the user's account with the option to modify the code.
- Allowed methods:
  - POST
- Payload:
  - Request Body:

Parameter	Type	Required	Description
templateId	int	Y	The ID of the template to fork.
modifiedCode	String	N	The modified code to be included in the forked template. If not provided, the original template's code is used.

- An example request
  - http://localhost:3000/api/templates/fork

```
{
    "templateId": 1,
    "modifiedCode": "console.log('Modified Code')"
}
```

- An example response

```
{
    "message": "Template forked",
    "forkedTemplate": {
        "id": 2,
        "title": "Forked: Code Template",
        "explanation": "This is a test code template!",
        "code": "console.log('Modified Code')",
        "language": "javascript",
        "tags": "javascript,coding",
        "forked": true,
        "userId": 1
    }
}
```

/api/templates/edit

- Short description:
  - This API endpoint allows an authenticated user to update the details of their template, including title, explanation, code, and tags.
- Allowed methods:
  - PUT
- Payload:
  - Request Body:

Parameter	Type	Required	Description
templateId	int	Y	The ID of the template to retrieve.
title	String	N	The new title.
explanation	String	N	The new explanation.
code	String	N	The new code.
tags	Array of strings	N	The new tags.

- An example request
  - http://localhost:3000/api/templates/edit

```
{
    "templateId": 1,
    "title": "Modified Template",
    "explanation": "This template was modified.",
    "code": "console.log('This was modified!')",
    "tags": ["Modified", "code", "javascript"]
}
```

- An example response

```
{
    "message": "Template updated",
    "updatedTemplate": {
        "id": 1,
        "title": "Modified Template",
        "explanation": "This template was modified.",
        "code": "console.log('This was modified!')",
        "language": "javascript",
```

```
        "tags": "Modified,code,javascript",
        "forked": false,
        "userId": 1
    }
}
```


/api/templates/viewOwn

- Short description:
  - This API endpoint allows an authenticated user to retrieve their templates in a paginated format.
- Allowed methods:
  - GET
- Payload:
  - Request Query:

Parameter	Type	Required	Description
page	int	N	For pagination, specify page number. Default = 1.
limit	int	N	For pagination, specify the number of blog posts per page. Default = 10.

- An example request
  - http://localhost:3000/api/templates/viewOwn
- An example response


```
{
    "templates": [
        {
            "id": 1,
            "title": "Modified Template",
            "explanation": "This template was modified.",
            "code": "console.log('This was modified!')",
            "language": "javascript",
            "tags": [
                "Modified",
                "code",
                "javascript"
            ],
            "forked": false,
            "userId": 1
        },
```

```
        {
            "id": 2,
            "title": "Forked: Code Template",
            "explanation": "This is a test code template!",
            "code": "console.log('Modified Code')",
            "language": "javascript",
            "tags": [
                "javascript",
                "coding"
            ],
            "forked": true,
            "userId": 1
        }
    ],
    "page": 1,
    "limit": 10,
    "totalTemplates": 2
}
```


/api/templates/delete

- Short description:
  - This API endpoint allows an authenticated user to delete a template they own, based on the template ID.
- Allowed methods:
  - DELETE
- Payload:
  - Request Body:

Parameter	Type	Required	Description
id	int	Y	The ID of the template to retrieve.

- An example request
  - http://localhost:3000/api/templates/delete


```
{
    "id": 2
}
```


- An example response


```
{
```

```
    "message": "Template deleted"
}
```

/api/templates/viewAll

- Short description:
  - This API endpoint allows users to search for templates based on title, tags, or explanation and retrieve them in a paginated format. Or if nothing is specified it displays all the templates.
- Allowed methods:
  - GET
- Payload:
  - Request Query:

Parameter	Type	Required	Description
page	int	N	For pagination, specify page number. Default = 1.
limit	int	N	For pagination, specify the number of blog posts per page. Default = 10.
search	string	N	A search term to filter templates by title, tags, or explanation.

- An example request
  - http://localhost:3000/api/templates/viewAll
- An example response (I created another template with a different user)


```
[
    {
        "id": 1,
        "title": "Modified Template",
        "explanation": "This template was modified.",
        "code": "console.log('This was modified!')",
        "language": "javascript",
        "tags": [
            "Modified",
            "code",
            "javascript"
        ],
        "forked": false,
        "userId": 1,
        "blogPosts": []
    },
```

```
    {
        "id": 3,
        "title": "Code Template Number Two",
        "explanation": "This is another test code template!",
        "code": "console.log('Hi!')",
        "language": "c",
        "tags": [
            "c",
            "coding"
        ],
        "forked": false,
        "userId": 2,
        "blogPosts": []
    }
]
```


http://localhost:3000/api/templates/viewAll?search=javascript


```
[
    {
        "id": 1,
        "title": "Modified Template",
        "explanation": "This template was modified.",
        "code": "console.log('This was modified!')",
        "language": "javascript",
        "tags": [
            "Modified",
            "code",
            "javascript"
        ],
        "forked": false,
        "userId": 1,
        "blogPosts": []
    }
]
```


/api/templates/[id]

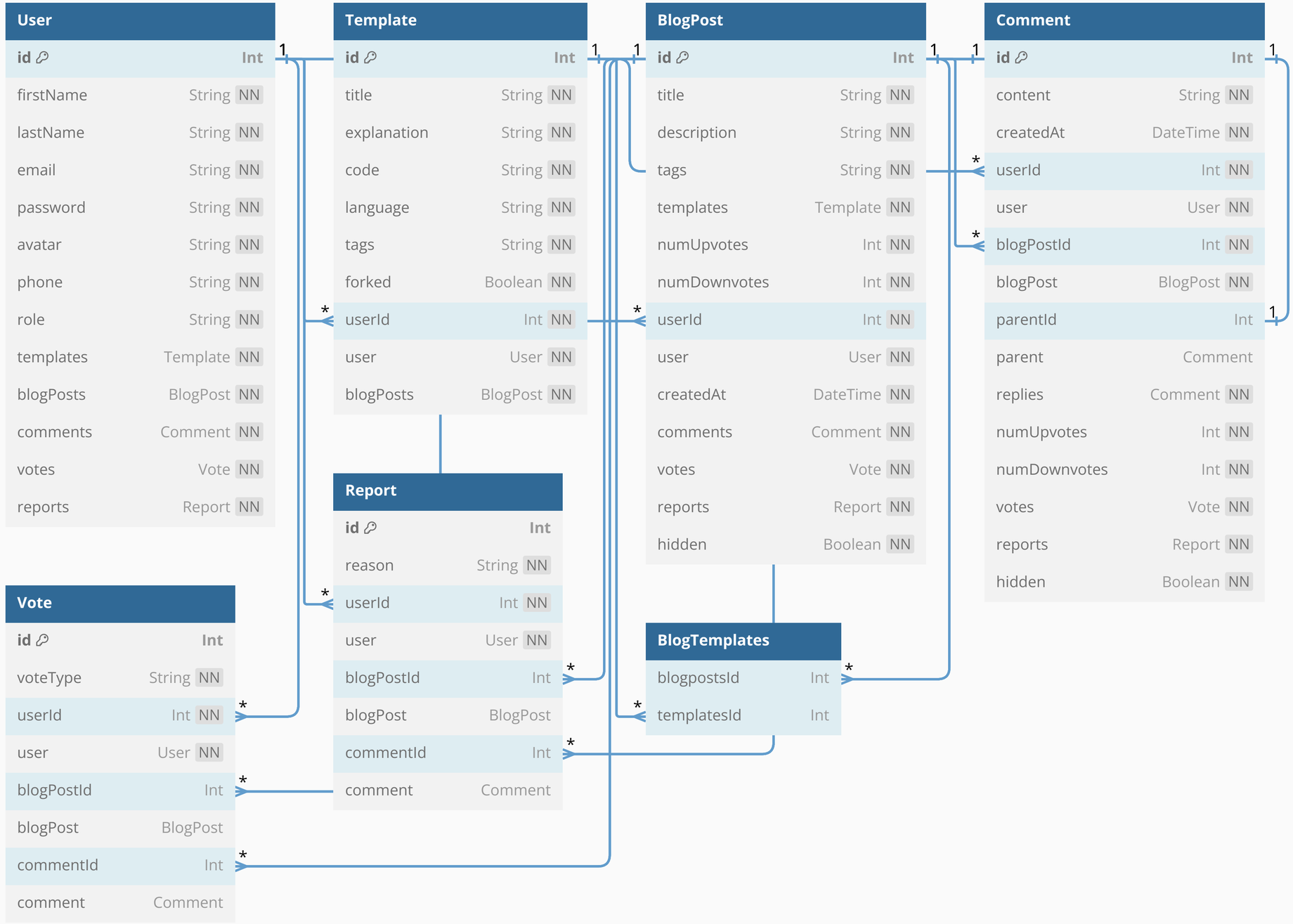
- Short description:

    - This API endpoint retrieves a specific template by its ID.
- Allowed methods:
    - GET
- Payload:
    - Request Query:

Parameter	Type	Required	Description
id	int	Y	The ID of the template to retrieve.

- An example request
    - http://localhost:3000/api/templates/1
- An example response


```
{
    "id": 1,
    "title": "Modified Template",
    "explanation": "This template was modified.",
    "code": "console.log('This was modified!')",
    "language": "javascript",
    "tags": "Modified,code,javascript",
    "forked": false,
    "userId": 1
}
```



dbdiagram.io