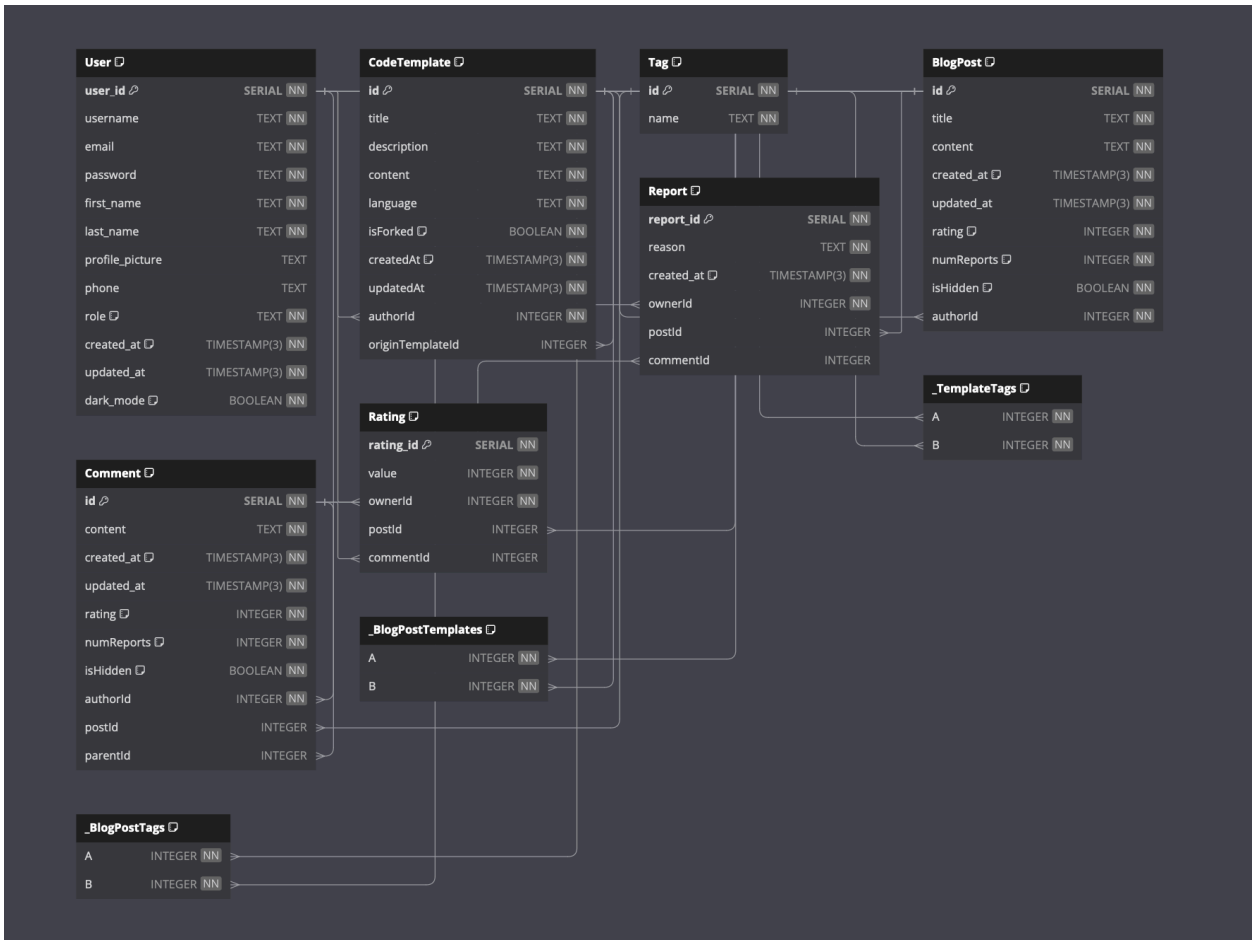
Model Design:



POST /api/auth/register

The `POST /api/auth/register` endpoint is used to register a new user with the provided information. The request should include the user's username, email, password, first name, last name, and phone number.

Request Body

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

Response

The response of this request is a JSON object representing the user's registration status.

Req :

```
{
"username": "exampleUser1",
"email": "example1@example.com",
"password": "examplePassword",
"firstName": "John",
"lastName": "Doe",
"phone": "1234567890"
}
```

Res:

```
{
"user_id": 3,
"username": "exampleUser1",
"email": "example1@example.com",
"password": "$2b$10$mkJeQnHJMA7mYsTjmr8BYeBKvthUl3p1GMEvX/mEWguHBxg/fbNZm",
"first_name": "John",
"last_name": "Doe",
"profile_picture": "/public/default_profile_pic.png",
"phone": "1234567890",
"role": "USER",
"created_at": "2024-11-04T00:41:22.806Z",
"updated_at": "2024-11-04T00:41:22.806Z",
"dark_mode": false
}
```

POST /api/auth/login

The `POST /api/auth/login` endpoint is used to authenticate a user by providing their username and password.

Request Body

The request should be sent with a JSON payload in the raw request body type, containing the `username` and `password` fields. The expected format for the request body is as follows:

- username(required): string
- password(required): string

Example:

Req:

```
"username": "exampleUser1",
"password": "examplePassword"
}
```

Res:

```
{
"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxLCJyb2xlIjoiVVNFUiIsImV4cGlyZXNB
dCI6MTczMDY3OTEyNjk3NywiaWF0IjoxNzMwNjc1NTI2LCJleHAiOjE3MzA2NzkxMjZ9.QSCrzZiWBmxhV0sPM
HilXQ0jI4TcOz7hf9XXE1-SI-I",
"refreshToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxLCJyb2xlIjoiVVNFUiIsImV4cGlyZXNB
dCI6MTczMDg0ODMyNjk3OCwiaWF0IjoxNzMwNjc1NTI2LCJleHAiOjE3MzA4NDgzMjZ9.-UJiZ4iuVITX-SIK6
C_j5Q0s4P-76_OsLHWb2YrX90c"
}
```

Put http://localhost:3000/api/auth/profile

This endpoint allows the user to update their profile information.

Request Body

- email (optional, string): The new email address of the user.
- firstName (optional, string): The new first name of the user.
- lastName (optional, string): The new last name of the user.
- profilePicture (optional, string): The new profile picture URL of the user.
- phone (optional, string): The new phone number of the user.
- darkMode (optional, boolean): The new dark mode preference of the user.
- new_username (optional, string): The new username of the user.

Example:

Req:


```
{
"email": "newEmail@example.com",
"firstName": "Andrew",
"darkMode": true,
"new_username": "tasdf"
}
```


Res:


```
{
"message": "Profile updated successfully",
"user": {
"user_id": 2,
"username": "tasdf",
"email": "newEmail@example.com",
"password": "$2b$10$R3bEionTZRNAhE4wB3NUHexypK1SrzDYt6uzlduazVJ/iNgqWfYWi",
"first_name": "Andrew",
"last_name": "Doe",
"profile_picture": "/public/default_profile_pic.png",
"phone": "1243567890",
"role": "USER",
"created_at": "2024-11-03T23:12:04.913Z",
"updated_at": "2024-11-03T23:12:15.254Z",
"dark_mode": true
}
}
```

Post http://localhost:3000/api/auth/refresh

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

Request Body

- `refresh_token` (string): The refresh token used to obtain a new access token.

Example:

Req:

```
{
"refresh_token": "Bearer {{refreshToken1}}"
}
```

Res:

```
{

"accessToken": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxLCJyb2xlIjoiVVNFUiIsImV4cGlyZXNBd
CI6MTczMDY3OTEyNjk3NywiaWF0IjoxNzMwNjc1NTI2LCJleHAiOjE3MzA2NzkxMjZ9.QSCrzZiWBmxhV0sPMH
ilXQ0jI4TcOz7hf9XXE1-SI-I"

}
```

Post /api/code/execute

This endpoint allows you to execute code in the specified programming language.

Request Body

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

Example:

Req:

```
{
"code": "#include <stdio.h>\n\nint add_numbers(int a, int b, int c) {\n return a * b * c;\n}\n\nint main() {\n int a, b, c;\n printf(\"Enter three numbers: \");\n scanf(\"%d %d %d\", &a, &b, &c);\n printf(\"Sum: %d\\n\", add_numbers(a, b, c));\n return 0;\n}",
"language": "c",
"input": "20 30 40"
}
```

Res:

```
{
"output": "Enter three numbers: Sum: 24000\n"
}
```

Post /api/templates/create

This endpoint allows the user to create/fork and save a new code template.

Request Body

- isForked (optional, boolean): Indicates if the template is a fork of another template.
- title (required if not forked, string): The title of the code template.
- description (required if not forked, string): The description of the code template.
- content (required if not forked, string): The content of the code template.
- language (required if not forked, string): The programming language of the code template.
- originTemplateId (required if forked, integer): The ID of the original template being forked.
- tags (optional, array of strings): The tags associated with the code template.

Example:
Req:


```
{
"title": "Add Numbers",
"description": "A simple Python script to add two numbers",
"content": "def add_numbers(a, b):\n return a + b\nif __name__ == \"__main__\":\n
try:\n numbers = input().strip().split()\n if len(numbers) != 2:\n print(\"Please
provide exactly two numbers\")\n exit(1)\n \n a = int(numbers[0])\n b =
int(numbers[1])\n print(add_numbers(a, b))\n except ValueError:\n print(\"Error:
Please provide valid integer numbers\")\n exit(1)",
"language": "python"
}
```


Res:


```
{
"id": 1,
"title": "Add Numbers",
"description": "A simple Python script to add two numbers",
"content": "def add_numbers(a, b):\n return a + b\nif __name__ == \"__main__\":\n
try:\n numbers = input().strip().split()\n if len(numbers) != 2:\n print(\"Please
provide exactly two numbers\")\n exit(1)\n \n a = int(numbers[0])\n b =
int(numbers[1])\n print(add_numbers(a, b))\n except ValueError:\n print(\"Error:
Please provide valid integer numbers\")\n exit(1)",
"language": "python",
"isForked": false,
"createdAt": "2024-11-04T00:55:39.971Z",
"updatedAt": "2024-11-04T00:55:39.971Z",
"authorId": 1,
"originTemplateId": null,
"author": {
```

```
"username": "GOAT",
"profile_picture": "/public/default_profile_pic.png"
},
"tags": []
}
```

Put or Delete /api/templates/[id]

This endpoint allows the user to update a specific template by making an HTTP PUT request to the specified URL or delete a specific template by providing the template ID in the URL.

Request Body

- title (optional, string): The new title of the code template.
- description (optional, string): The new description of the code template.
- content (optional, string): The new content of the code template.
- tags (optional, array of strings): The new tags associated with the code template.
- language (optional, string): The new programming language of the code template.

Example:
Req:


```
{
"title": "Comprehensive Update",
"description": "This is a comprehensive update of the code template.",
"content": "console.log('hello world!')",
"tags": ["javascript", "frontend"],
"language": "javascript"
}
```


Res:


```
{
"id": 14,
"title": "Comprehensive Update",
"description": "This is a comprehensive update of the code template.",
"content": "console.log('hello world!')",
"language": "javascript",
"isForked": true,
"createdAt": "2024-11-03T05:34:34.491Z",
"updatedAt": "2024-11-03T21:27:49.041Z",
"authorId": 2,
"originTemplateId": 7,
"tags": [
{
"id": 3,
"name": "javascript"
},
{
"id": 4,
"name": "frontend"
}
]
}
```

GET /api/templates

This endpoint retrieves a list of templates.

Request

No request body is required for this endpoint.

Example:
Req: http://localhost:3000/api/templates?query=numbers in c
Res:


```
{
"templates": [
{
"id": 2,
"title": "Add Numbers in c",
"description": "A simple Python script to add two numbers",
"content": "#include <stdio.h>\n\nint add_numbers(int a, int b) {\n return a +
b;\n}\n\nint main() {\n int a, b;\n printf(\"Enter two numbers: \");\n scanf(\"%d
%d\", &a, &b);\n printf(\"Sum: %d\\n\", add_numbers(a, b));\n return 0;\n}",
"language": "c",
"isForked": false,
"createdAt": "2024-11-04T01:13:46.165Z",
"updatedAt": "2024-11-04T01:13:46.165Z",
"authorId": 2,
"originTemplateId": null,
"author": {
"username": "tasdf",
"profile_picture": "/public/default_profile_pic.png"
},
"tags": []
}
],
"pagination": {
"currentPage": 1,
"totalPages": 1,
"totalItems": 1,
"itemsPerPage": 10,
"hasMore": false,
"hasLess": false
}
}
```

POST api/posts/create

This endpoint allows you to create a new post by submitting the title and content of the post.

Request Body

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

Example:
Req:

```
{
"title": "Introduction to JavaScript",
"content": "JavaScript is a versatile programming language used in web development, hihellolmao."
}
```

Res:

```
{
"id": 1,
"title": "Introduction to JavaScript",
"content": "JavaScript is a versatile programming language used in web development, hihellolmao.",
"created_at": "2024-11-04T01:45:53.341Z",
"updated_at": "2024-11-04T01:45:53.341Z",
"rating": 0,
"numReports": 0,
"isHidden": false,
"authorId": 1
}
```

GET api/posts/

This endpoint is used to retrieve posts based on the provided query parameters. The request is sent via an HTTP GET method to the specified URL. The query parameters include sortBy and query, which can be used to sort and filter the posts.

Request Body

This request does not require a request body.

Example:

Req: http://localhost:3000/api/posts/

Res:


```
{
"posts": [
{
"id": 1,
"title": "Introduction to JavaScript",
"content": "JavaScript is a versatile programming language used in web development,
hihellolmao.",
"created_at": "2024-11-04T01:45:53.341Z",
"updated_at": "2024-11-04T01:45:53.341Z",
"rating": 0,
"numReports": 0,
"isHidden": false,
"authorId": 1,
"tags": [],
"templates": [],
"author": {
"user_id": 1,
"username": "exampleUser1",
"email": "example1@example.com",
"password": "$2b$10$05YZUZ9hHE/41C4rkJSzD.s13qgnmME.5xmE1fJ9tKcnXUjVsSONm",
"first_name": "John",
"last_name": "Doe",
"profile_picture": "/public/default_profile_pic.png",
"phone": "1234567890",
"role": "USER",
"created_at": "2024-11-04T01:45:40.220Z",
"updated_at": "2024-11-04T01:45:40.220Z",
"dark_mode": false
```

```
}
}
],
"pagination": {
"totalPosts": 1,
"currentPage": 1,
"pageSize": 10,
"totalPages": 1
}
}
```

PATCH /api/admin/comments/:id

This endpoint allows administrators to hide a specific comment by its ID using an HTTP PATCH request.

Request Parameters

- `id` (path parameter): The ID of the comment to be updated.

Example:

Req: http://localhost:3000/api/admin/comments/:id
Res:

{ "message": "Success", "updatedComment": { "id": 4, "content": "ur post bad L aura", "created_at": "2024-11-04T02:42:54.226Z", "updated_at": "2024-11-04T02:44:10.021Z", "rating": 0, "numReports": 0, "isHidden": true, "authorId": 2, "postId": 1, "parentId": null } }

GET `/api/admin/comments`

This endpoint retrieves the comments based on the number of reports.

Request

- Method: GET
- URL: http://localhost:3000/api/admin/comments

`Example:`

`Req:` http://localhost:3000/api/admin/comments

`Res:`

{ "comments": [{ "id": 5, "content": "ur post bad L aura", "created_at": "2024-11-04T02:42:54.874Z", "updated_at": "2024-11-04T02:42:54.874Z", "rating": 0, "numReports": 0, "isHidden": false, "authorId": 2, "postId": 1, "parentId": null, "reports": [], "ratings": [] }], "pagination": { "totalComments": 1, "currentPage": 1, "pageSize": 10, "totalPages": 1 } }

PATCH api/admin/posts/:id

This endpoint is used to update a specific post by its ID.

Request

- Method: `PATCH`
- URL: `http://localhost:3000/api/admin/posts/:id`
- Body (x-www-form-urlencoded):
    - No parameters provided

Example:

Req: http://localhost:3000/api/admin/posts/1

`Res:`

{ "message": "Success", "updatedPost": { "id": 1, "title": "Introduction to JavaScript", "content": "JavaScript is a versatile programming language used in web development, hihellolmao.", "created_at": "2024-11-04T02:42:38.466Z", "updated_at": "2024-11-04T02:49:59.727Z", "rating": 0, "numReports": 0, "isHidden": true, "authorId": 2 } }

GET api/admin/posts

This endpoint retrieves a list of posts that has been reported for the admin

Example:
Req: http://localhost:3000/api/admin/posts
Res:

GET PUT DEL api/posts/[postId]/comments/[id]

POST DEL api/posts/[id]/comments/rate

POST api/posts/[id]/comments/create

GET api/posts/[id]/comments/

POST api/posts/[id]/comments/report

GET PUT DEL api/posts/[postId]

POST api/tags/create

GET api/tags/