CSC309 Scriptorium Project Part 1

Yehyun, Rosalie, and Sadra

README

Hello, this is a starting point to grade our part 1.
Please read this README section first to find the bare minimum instructions and important files.

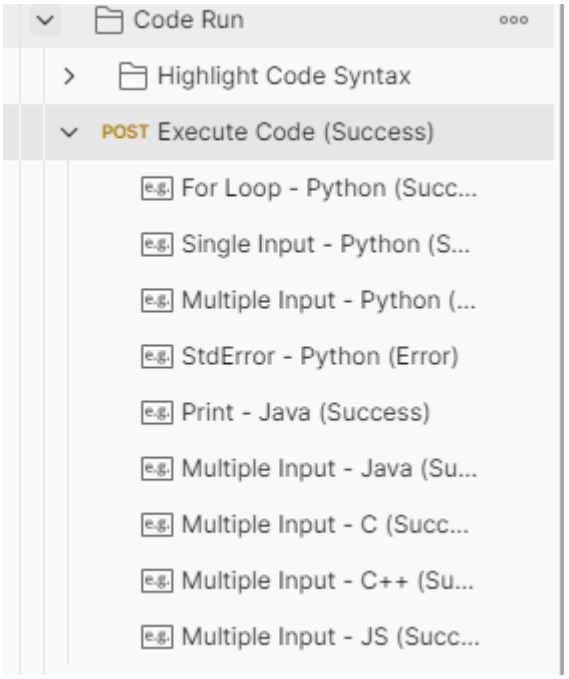
Important Files
startup.sh
run.sh
docs.pdf
postman.pdf

Instruction

```
hippo:~/Desktop/CSC309/group_7322/PP1/scriptorium$ ./startup.sh
bash: ./startup.sh: Permission denied
hippo:~/Desktop/CSC309/group_7322/PP1/scriptorium$ bash startup.sh
Installing npm packages:
```

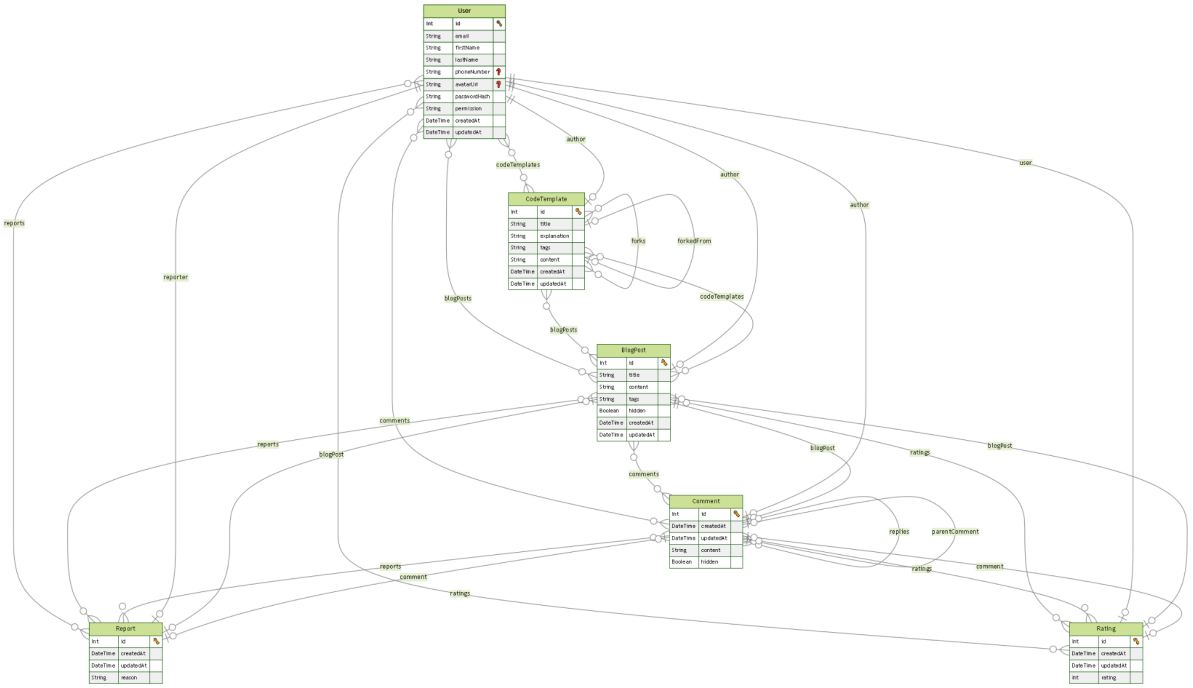
Running via ./filename may cause a permission issue.
You can fix it via chmod or we suggest you to run via:

bash startup.sh
bash run.sh



You can expand our *Execute Code (Success)* to try different languages!

Model Design



Tables:

1. User:

   Stores user data such as email, name, phone number, avatar URL, password hash, permission level, and timestamps. Tracks relationships to code templates, blog posts, comments, ratings, and reports made by the user.

2. CodeTemplate:

   Represents code templates, including title, explanation, tags (as a comma-separated string), content, and timestamps. Allows for tracking forks of templates and has an author relationship to the User who created it. It is associated with related blog posts.

3. BlogPost:

   Contains blog post data such as title, content, tags (comma-separated), visibility (hidden or public), and timestamps. It has an author relationship to the User who created it and links to comments, ratings, reports, and associated code templates.

4. Comment:

   Represents comments made on blog posts, containing the comment content, visibility, and timestamps. Tracks relationships to the author (User), the blog post it's associated with, and supports threaded replies. Also links to related reports and ratings.

5. Rating:

   Stores user ratings on blog posts or comments, with fields for rating value, timestamps, and relationships to the User who gave the rating, the blog post, or the comment it applies to.

6. Report:

   Represents reports made by users on blog posts or comments, including a reason, timestamps, and relationships to the User (reporter), the blog post, or the comment being reported.

Team Emails
yehyun.lee@mail.utoronto.ca
rosalie.pampolina@mail.utoronto.ca
s.setarehdan@mail.utoronto.ca

Branch Naming Convention
features/developer name/feature name
No dot .; just use dash -
Pull Req. One person approves & merges.

File and Folder Naming Convention
We use _ underscore instead of capital letters
Function names can be anything, even with capital letters though.

Communication Method
Discord Server and Git

Scriptorium

Auth

POST Signup

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

The signup endpoint allows a new user to be signed up to the platform (ie. create an account for them)

The endpoint will respond with a JSON with an object containing:

- "message": an message indicating if the signup was successful
- "user": a user object with all of the newly signed up user's details

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "email": "test@gmail.com",
    "firstName": "Test",
    "lastName": "Account",
    "password": "StrongPassword123!",
    "phone": "1234567890",
    "avatar": "https://example.com/avatar.jpg"
}
```

POST Login

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

The login endpoint authenticates and logs in a user for the current session on the platform if they exist.

`Auth token` in the global variable will be automatically updated when this endpoint is ran via by Postman.

The endpoint will respond with a JSON with an object containing:

- "message": an message indicating if the login was successful
- "token": an authorization token

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "email": "test@gmail.com",
    "password": "StrongPassword123!"
}
```

POST Refresh token

http://localhost:3000/api/auth/refresh_token

The refresh token endpoint checks the user token and extends its expiration.

The endpoint will respond with a JSON with an object containing:

- "message": a message indicating if the operation was successful
- "token": an authorization token

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

POST Logout

http://localhost:3000/api/auth/logout

The logout endpoint logs the user out of the session.

The endpoint will respond with a JSON with an object containing:

- "message": a message indicating if the logout was successful

HEADERS

Content-Type	application/json

GET Get Profile

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

The get profile endpoint gets the profile information of the corresponding user that matches the user ID of the current session's user.

The endpoint will respond with a JSON with an object containing:

- "email": the email of the user
- "firstName": the first name of the user
- "lastName": the last name of the user
- "phoneNumber": the phone number of the user
- "avatarUrl": the url to the users avatar
- "permission": the permission level of the user

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

Body raw (json)

json


```
{
    "email": "test@gmail.com",
    "firstName": "Test",
    "lastName": "Account"
```

```
    "lastName": "Account",
    "password": "StrongPassword123!",
    "phone": "1234567890",
    "avatar": "https://example.com/avatar.jpg"
}
```


PUT Update Profile

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

The put profile endpoint updates the profile information of the corresponding user that matches the user ID of the current session's user to match what is passed in the body of the request.

The endpoint will respond with a JSON with an object containing:

- "id": the id of the user
- "email": the email of the user
- "firstName": the first name of the user
- "lastName": the last name of the user
- "phoneNumber": the phone number of the user
- "avatarUrl": the url to the users avatar
- "passwordHash": the hash of the users password
- "permission": the permission level of the user
- "createdAt": the time the user was created at
- "updatedAt": the time the user was last updated

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

Body raw (json)

json


```
{
    "email": "test@gmail.com",
    "firstName": "Test",
    "lastName": "Account",
    "password": "StrongPassword123!",
    "phone": "1234567890",
    "avatar": "https://example.com/avatar.jpg"
}
```

Blogs

GET List of Blogs (all)

```
http://localhost:3000/api/blog/
```

The blog endpoint provides a list of all blog posts

The endpoint will respond with a JSON with an object containing:

- "blogs": an array of blog objects
- "total": the total amount of blogs in the response
- "page": the page of the paginated blogs
- "limit": the limit of the amount of blogs in a page of the paginated blogs

HEADERS

Content-Type	application/json

GET List of Blogs (search)

```
http://localhost:3000/api/blog/?search=darknet diaries
```

The search blog endpoint allows the user to search for blog posts that match the value provided for the search query parameter

The endpoint will respond with a JSON with an object containing:

- "blogs": an array of blog objects
- "total": the total amount of blogs in the response
- "page": the page of the paginated blogs
- "limit": the limit of the amount of blogs in a page of the paginated blogs

HEADERS

Content-Type	application/json

PARAMS

search	darknet diaries

GET List of Blogs (by template)

```
http://localhost:3000/api/blog/?template_id=1
```

The template blog endpoint allows the user to search for blog posts that contain the code template with the id matching the value provided for the template_id query parameter

The endpoint will respond with a JSON with an object containing:

- "blog_by_template": an array of blog objects
- "template_total": the total amount of blogs in the response
- "page": the page of the paginated blogs
- "limit": the limit of the amount of blogs in a page of the paginated blogs

HEADERS

Content-Type	application/json

PARAMS

template_id	1

GET List of Blogs (by rating)

```
http://localhost:3000/api/blog/?sort_by_rating=1
```

The rating blog endpoint allows the user to search for blog posts filtering by blog posts that have the same rating as the value provided for the sort_by_rating query parameter

The endpoint will respond with a JSON with an object containing:

- "blog_by_rating": an array of blog objects
- "rating_total": the total amount of blogs in the response
- "page": the page of the paginated blogs
- "limit": the limit of the amount of blogs in a page of the paginated blogs

HEADERS

Content-Type	application/json

PARAMS

sort_by_rating	1

POST Add New Blog

http://localhost:3000/api/blog

The new blog endpoint allows users to make a new blog post with the given information in the body of the request

The endpoint will respond with a JSON with an object containing:

- "id": the id of the new blog
- "title": the title of the new blog
- "content": the content of the new blog
- "tags": the tags of the new blog
- "hidden": a boolean indicating if the new blog is hidden
- "createdAt": the time the new blog was created at
- "updatedAt": the time the new blog was last updated
- "authorId": the ID of the author of the new blog

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

Body raw (json)

json

```
{
    "title": "Nope!!!!",
    "content": "JUst content!",
    "tags": "javascript,web development"
}
```

POST Add New Blog with Template

http://localhost:3000/api/blog?Content-Type=application/json

The new blog endpoint allows users to make a new blog post with the given information in the body of the request

The endpoint will respond with a JSON with an object containing:

- "id": the id of the new blog

- "title": the title of the new blog
- "content": the content of the new blog
- "tags": the tags of the new blog
- "hidden": a boolean indicating if the new blog is hidden
- "createdAt": the time the new blog was created at
- "updatedAt": the time the new blog was last updated
- "authorId": the ID of the author of the new blog

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

PARAMS

Content-Type	application/json

Body raw (json)

json

```
{
    "title": "new blog with code",
    "content": "JUst content!",
    "tags": "javascript,web development",
    "codeTemplateIds": [1]
}
```

PUT Edit Blog

http://localhost:3000/api/blog/edit/1?Content-Type=application/json

The edit blog endpoint allows a user to edit an existing blog post by updating its information to match the information provided in the body of the request

The endpoint will respond with a JSON with an object containing:

- "id": the id of the edited blog
- "title": the title of the edited blog
- "content": the content of the edited blog
- "tags": the tags of the edited blog
- "hidden": a boolean indicating if the edited blog is hidden
- "createdAt": the time the edited blog was created at
- "updatedAt": the time the edited blog was last updated
- "authorId": the ID of the author of the edited blog

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

PARAMS

Content-Type	application/json

Body raw (json)

```
json
```

```
{
    "title": "My idk how many Blog Post",
    "content": "This is a content of my first blog post.",
    "tags": "javascript,cybersecurity"
}
```

DELETE Delete Blog

http://localhost:3000/api/blog/delete/1?Content-Type=application/json

The delete blog endpoint allows users to delete an existing blog post

The endpoint will respond with a JSON with an object containing:

- "message": a message indicating if the deletion was successful

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

PARAMS

Content-Type	application/json

Body raw (json)

```
json
```

```
{
    "title": "My First Blog Post"
```

```
    "title": "My First Blog Post",
    "content": "This is a content of my first blog post. I like the harry potter book ser
    "tags": "javascript,web development"
}
```


POST Post Blog Comments

http://localhost:3000/api/blog/1/add_comment?Content-Type=application/json

The blog comments endpoint allows users to post comments to a specific blog post

The endpoint will respond with a JSON with an object containing:

- "id": the id of the new comment
- "createdAt": the time the new comment was created at
- "updatedAt": the time the new comment was last updated
- "content": the content of the new comment
- "hidden": a boolean indicating if the new comment is hidden
- "authorId": the ID of the author of the new comment
- "blogPostId": the ID of the associated blog post
- "parentCommentId": the id of the parent comment, if applicable

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

PARAMS

Content-Type	application/json

Body raw (json)

json


```
{
    "content": "This is a content of my first blog post comment. I like the harry potter
}
```


POST Post Blog Rating

http://localhost:3000/api/blog/1/add_rate

The blog rating endpoint allows users to leave a rating of a specific blog post

The endpoint will respond with a JSON with an object containing:

- "id": the id of the new rating
- "createdAt": the time the new rating was created at
- "updatedAt": the time the new rating was last updated
- "rating": the rating value
- "userId": the ID of the user leaving the rating
- "blogPostId": the ID of the associated blog post
- "commentId": the id of the corresponding comment, if applicable

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

Body raw (json)

```json
{
    "ratingValue": 1
}
```

GET List Blog Comments

```
http://localhost:3000/api/blog/1/get_comments
```

The blog comments endpoint provides a list of all comments on a blog.

The endpoint will respond with a JSON with an object containing:

- "comments": an array of comment objects
- "totalPages": the total amount of pages of comments in the response
- "page": the page of the paginated comments
- "totalComments": the limit of the amount of comments in a page of the paginated comments

HEADERS

Content-Type	application/json

GET Average rating of blog

```
http://localhost:3000/api/blog/1/get_rates
```

The blog rating endpoint provides the average and total ratings.

The endpoint will respond with a JSON with an object containing:

- "totalRates": the total number of ratings
- "averageRate": the average rating of all the ratings

HEADERS

Content-Type	application/json

Code Run

Highlight Code Syntax

The highlight syntax endpoint manages the highlighting of the code in the proper syntax of the language provided

POST Highlight Code Syntax (Success)

```
http://localhost:3000/api/code_run/highlight
```

The highlight code syntax endpoint highlights the code when it is successful

The endpoint will respond with a JSON with an object containing:

- "highlightedCode": a string of the code with the text formatted to be highlighted correctly

HEADERS

Content-Type	application/json

Body raw (json)

```
{
```

json

```
{
    "code": "print('hello world')",
    "language": "Python"
}
```

POST Highlight Code Syntax Error

http://localhost:3000/api/code_run/highlight

The highlight code syntax endpoint returns a 500 error if it fails to highlight the code

The endpoint will respond with a JSON with an object containing:

- "error": the error message
- "details": the details about the error

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "code": "print('hello world')",
    "language": "nonexistentlanguage"
}
```

POST Highlight Code Syntax without Code

http://localhost:3000/api/code_run/highlight

The highlight code syntax endpoint should return a 400 error if no code is provided

The endpoint will respond with a JSON with an object containing:

- "message": a message containing the error

HEADERS

Content-Type	application/json

Content-Type application/json

Body raw (json)

```
{
    "language": "Python"
}
```

GET Highlight Code Syntax Method Not Allowed

http://localhost:3000/api/code_run/highlight

The highlight code syntax endpoint should return a 405 error if an incorrect method (ie POST) is used.

The endpoint will respond with a JSON with an object containing:

- "message": a message containing the error

HEADERS

Content-Type	application/json

Body raw (json)

```
{
    "code": "print('hello world')",
    "language": "Python"
}
```

POST Highlight Code Syntax without Language

http://localhost:3000/api/code_run/highlight

The highlight code syntax endpoint should return a 400 error if no language is provided

The endpoint will respond with a JSON with an object containing:

- "message": a message containing the error

HEADERS

Content-Type	application/json

Body raw (json)

```
json
```

```
{
    "code": "print('hello world')"
}
```

POST Execute Code (Success)

```
http://localhost:3000/api/code_run/execute_code
```

The execute code endpoint executes the code in the language provided with any given input based on the information provided in the request body and returns the results of the code execution in real time.

The endpoint will respond with a JSON with an object containing:

- "executionResult": the result of if the execution was successful
- "output": the output of the code

HEADERS

Content-Type	application/json

PARAMS

code

language

Body raw (json)

```
json
```

```
{
    "code": "print('hello world')",
    "language": "Python",
```

```
    "input": ""
}
```

POST Execute Code without Code

http://localhost:3000/api/code_run/execute_code

The execute_code endpoint should return an error response if no code is provided

The endpoint will respond with a JSON with an object containing:

- "message": a message containing the error

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "language": "Python",
    "input": ""
}
```

POST Execute Code without Language

http://localhost:3000/api/code_run/execute_code

The execute_code endpoint should return an error response if no language is provided

The endpoint will respond with a JSON with an object containing:

- "message": a message containing the error

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "code": "print('hello world')",
    "input": ""
}
```

POST Execute Code without Input

http://localhost:3000/api/code_run/execute_code

The execute_code endpoint should still run properly if no input is provided (assuming the code and language provided work correctly).

The endpoint will respond with a JSON with an object containing:

- "message": a message containing the error

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "language": "Python",
    "code": "print('hello world')"
}
```

Code Template

POST Create Template

http://localhost:3000/api/code_template/user/create_template

The create template endpoint allows users to create a new code template with the information provided in the body of the request

The endpoint will respond with a JSON with an object containing:

- "message": message stating if the template was created successfully
- "template": a template object with various information about the template

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

Body raw (json)

```json
{
    "title": "My Code Template",
    "explanation": "This is a sample template.",
    "tags": ["sample", "code"],
    "content": "def hello_world():\n    print('Hello, World!')"
}
```

GET Search User Templates

http://localhost:3000/api/code_template/user/search_template?search=sample&page=1&limit=5

The search user templates endpoint allows users to search for a specific template

The endpoint will respond with a JSON with an object containing:

- "templates": an array of template objects (each of which contain various information about a template such as id and title)

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

PARAMS

search	sample
page	1
limit	5

PUT Edit Template

http://localhost:3000/api/code_template/user/edit_or_delete_template?id=1

The edit template endpoint allows users to edit a given template by modifying the information to match what is provided in the request body

The endpoint will respond with a JSON with an object containing:

- "message": message stating if the template was edited successfully
- "template": a template object with various information about the edited template

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

PARAMS

id	1

Body raw (json)

json

```
{
    "title": "Updated Template Title",
    "explanation": "Updated explanation.",
    "tags": ["updated", "template"],
    "content": "def updated_function():\n    print('Updated content')"
}
```

DELETE Delete Template

http://localhost:3000/api/code_template/user/edit_or_delete_template?id=1

The delete template endpoint allows users to delete a specific endpoint

The endpoint will respond with a JSON with an object containing:

- "message": message stating if the template was deleted successfully

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

PARAMS

id	1

POST Fork Template

http://localhost:3000/api/code_template/user/create_fork_template?id=1

The fork template endpoint allows users to fork an existing template

The endpoint will respond with a JSON with an object containing:

- "message": message stating if the template was forked successfully
- "template": a template object with various information about the forked template

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

PARAMS

id	1

Body raw (json)

json

```
{
    "title": "Forked Template",
    "explanation": "This is a forked version.",
    "tags": ["forked"],
    "content": "def forked_function():\n    print('Forked content')"
}
```

GET Visitor Search Templates

http://localhost:3000/api/code_template/visitor/search_template?search=sample&page=1&limit=5

The visitor search templates endpoint allows visitors to view blog posts that mention a code template

The endpoint will respond with a JSON with an object containing:

- "templates": an array of template objects

HEADERS

Content-Type	application/json

PARAMS

search	sample
page	1
limit	5

Comments

GET Average rating of comment

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

The average rating of comment endpoint calculates the average rating given to a comment based on the total

ratings it has received.

The endpoint will respond with a JSON with an object containing:

- "totalRates": the total number of ratings given to the comment
- "averageRate": the average rating given to the comment

HEADERS

Content-Type	application/json

POST Post Comment Rating

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

The blog rating endpoint allows users to leave a rating of a specific blog post

The endpoint will respond with a JSON with an object containing:

- "id": the id of the new blog rating
- "createdAt": the time the new blog rating was created at
- "updatedAt": the time the new blog rating was last updated
- "rating": the rating value of the new blog rating
- "userId": a boolean indicating if the new blog rating is hidden
- "blogPostId": the ID of the associated blog post
- "commentId": the id of the corresponding comment

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

Body raw (json)

```json
{
    "ratingValue": 1
}
```

Inappropriate Content

POST Report blog post

http://localhost:3000/api/report

The report endpoint allows users to report an inappropriate blog post.

The endpoint will respond with a JSON with an object containing:

- "id": the id of the new report
- "createdAt": the time the new report was created at
- "updatedAt": the time the new report was last updated
- "reason": the reason of the new report
- "reporterId": the ID of the user reporting
- "blogPostId": the ID of the associated blog post
- "commentId": the id of the corresponding comment

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
    "reason": "Inappropriate content",
    "blogPostId": "1"
}
```

POST Report comment

http://localhost:3000/api/report

The report endpoint allows users to report an inappropriate comment.

The endpoint will respond with a JSON with an object containing:

- "id": the id of the new report
- "createdAt": the time the new report was created at
- "updatedAt": the time the new report was last updated
- "reason": the reason of the new report
- "reporterId": the ID of the user reporting
- "blogPostId": the ID of the associated blog post
- "commentId": the id of the corresponding comment

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

Body raw (json)

```
json
```

```
{
    "reason": "Inappropriate content",
    "commentId": "1"
}
```

GET Admin - get sorted content based on reports

http://localhost:3000/api/admin/get_sorted_reports

The get sorted content based on reports endpoint should allow only ADMIN users to get an array of content that has reports. If the user does not have ADMIN permission, they should get a 403 status and a message saying forbidden.

The endpoint will respond with a JSON with an object containing:

- "blogPosts": an array of blog objects that have been reported
- "comments": an array of comment objects that have been reported
- "page": the page of the paginated blog/comment objects
- "limit": the limit of the amount of blog/comment objects in a page of the paginated blogs/comments

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

POST Admin - hide blog post

http://localhost:3000/api/admin/hide_content

The hide blog post endpoint allows admins to hide a blog post. Non-admins will get a 403 error and message stating that it is forbidden.

The endpoint will respond with a JSON with an object containing:

- "message": a message stating if the blog post has been hidden
- "hiddenContent": the blog post object that has been hidden

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

PARAMS

Body raw (json)

json

```
{
  "contentType": "blogPost",
  "contentId": "2"
}
```

POST Admin - hide comment

http://localhost:3000/api/admin/hide_content

The hide comment endpoint allows admins to hide a comment. Non-admins will get a 403 error and message stating that it is forbidden.

The endpoint will respond with a JSON with an object containing:

- "message": a message stating if the comment has been hidden
- "hiddenContent": the comment object that has been hidden

AUTHORIZATION Bearer Token

Token	{{Auth Token}}

HEADERS

Content-Type	application/json

Body raw (json)

json

```
{
  "contentType": "comment",
  "contentId": "2"
}
```

Contribution and Progress Update within the Doc

Colour Coding for Contribution

Yehyun
Sadra
Rosalie
Worked Together
N/A, Not Implemented, or Part 2 Frontend

<u>Underline:</u> In progress
~~Strokeline~~: Complete
Italic: Unit test complete

User Stories Contribution and Related Files, Postman, etc.

Schema

~~User, CodeTemplate, BlogPost, Comment, Rating, Report, Etc. Schema~~

Related info:
prisma/ERD.pdf

Accounts

~~As a user, I want to sign up, log in, log out, and edit my profile. Profile information should include first and last name, email, avatar, and phone number. Authentication should be handled with a proper JWT setup.~~

Related Code: auth/login.js, auth/logout.js, auth/profile.js, auth/signup.js

Code writing and execution

Rosalie and Yehyun worked together for this.
Rosalie did the majority of work. Yehyun improved later on to support other languages with bug fixes.

~~As a visitor (unauthenticated user), I want to write code in various programming languages (e.g., C, C++, Java, Python, JavaScript) on Scriptorium.~~

Related Code: code_run/execute_code.js, utils**code_executer**.js

~~As a visitor, I want my code to be highlighted based on the syntax rules of the programming language so that I get a good understanding of my code.~~

Related Code: code_run/highlight.js

~~As a visitor, I want to execute my code on Scriptorium and see the output in real-time so that I can quickly verify its correctness.~~

Related Code: code_run/execute_code.js, utils**code_executer**.js

- ~~As a visitor, I want to provide input to my code via the standard input (stdin) before execution so that I can test programs that require user input.~~

Related Code: code_run/execute_code.js, utils**code_executer**.js

- ~~As a visitor, I want to see error messages if my code fails to compile or run so that I can debug my code effectively. This includes compile errors, runtime errors, timeouts, or any warnings generated in the meantime.~~

Related Code: code_run/execute_code.js, utils**code_executer**.js

Isolation

- ~~As a visitor, I want my code to run in a secure, isolated environment so that it does not affect other users or the system.~~
- ~~As a visitor, I want the system to enforce a time and memory limit on code execution so that infinite loops or long-running processes are automatically terminated.~~

Code templates

- ~~As a user (authenticated), I want to save my code as a template with a title, explanation, and tags so that I can organize and share my work effectively.~~

Related Code: code_template/user/**create_template**.js

- ~~As a user, I want to view and search through my list of my saved templates, including their titles, explanations, and tags, so that I can easily find and reuse them.~~

Related Code: code_template/user/**search_template**.js

- ~~As a user, I want to edit an existing code template's title, explanation, tags, and code, or delete it entirely.~~

Related Code: code_template/user/**edit_or_delete_template**.js

- ~~As a visitor, I want to use an existing code template, run or modify it, and if desired, save it as a new template with a notification that it's a forked version, so I can build on others' work. Saving a template is only available to authenticated users.~~

Related Code: code_template/user/**create_fork_template**.js

- ~~As a visitor, I want to search through all available templates by title, tags, or content so that I can quickly find relevant code for my needs.~~

Related Code: code_template/visitor/**search_template**.js

Blog posts

- ~~As a user, I want to create/edit/delete blog posts. A blog post has title, description, and tag. It might also include links to code templates (either mine or someone else's).~~

Related Code: blog/**index**.js with POST, blog/**edit/[id]**.js, blog/**delete/[id]**.js

- ~~As a visitor, I want to browse and read blog posts so that I can learn from others' experiences and code examples. I want to search through blog posts by their title, content, tags, and also the code templates they contain.~~

Related Code: blog/**index**.js with GET, search, else

- As a visitor, I want to follow links from a blog post directly to the relevant code template so that I can view, run, or fork the code discussed.
- ~~As a visitor, I want to see the list of blog posts that mention a code template on the template page.~~

Related Code: blog/**index**.js with GET, template_id

- ~~As a user, I want to comment, or reply to existing comments on a blog post so that I can engage with the author and other readers by sharing feedback, asking questions, or starting discussions.~~

Related Code: blog/**[id]/add_comment**.js

- ~~As a user, I want to rate blog posts and comments with upvotes or downvotes so that I can express my agreement or disagreement with the content.~~

Related Code: blog/**[id]/add_rate**.js

- ~~As a visitor, I want to see the list of blog posts and comments sorted by their ratings so that I get exposed to the most valued or controversial discussions first.~~

Related Code: blog/**index**.js with GET, sort_by_rating

Inappropriate content reports

- ~~As a user, I want to report an inappropriate blog post or comment so that the website is purged of abusive content. I want to add additional explanation when submitting my report.~~

Related Code: **report**/index.js

- ~~As the system administrator, I want to sort blog posts and comments based on the total number of reports they received so that I can find the inappropriate content easier.~~

Related Code: admin/**get_sorted_reports**.js

- ~~As the system administrator, I want to hide a content that I deem inappropriate so that Scriptorium remains safe for everyone. This content would then be hidden from everyone, except for the author. The author can still see the content (with a flag that indicates the reports), but cannot edit it.~~

Related Code: admin/**hide_content**.js

User experience

- As a visitor, I want a clean and intuitive user interface so that I can easily write, execute, and manage my code templates without confusion.
- As a visitor, I want the website to be rendered well in different screen sizes (e.g., monitors, laptop, tablet, and mobile) so I can effectively work on my codes everywhere.
- As a visitor, I want to be able to toggle between dark and light themes so that I can choose the most comfortable working environment.