CSC309 SCRIPTORIUM DOCUMENTATION

Homayoun Elyasi

Yash Sethi

Christopher Allen Garcia Flores

ADMIN INFO:

username	admin
email	admin@example.com
password	adminPassword

USER MODEL

Field	Type	Params	Description
id	Int	@id @default(autoincrement())	Unique id for each user
email	String	@unique	Unique email associated with user
password	String		User’s password
username	String	@unique	Unique username for user
firstName	String		User’s first name
lastName	String		User;s last name
phoneNumber	BigInt		User’s phone number
avatar	String?	//Optional field	Optional avatar (male/female)
createdAt	DateTime	@default(now())	Time the user created their account
Blog	Blog[]		A list of blogs the user has created
Comment	Comment[]		A list of comments the user has created
Template	Template[]		A list of templated the user has created
role	String?	default(“USER”)	The role of the user (could be admin, or a simple user)
BlogReport	BlogReport[]		A user’s list of reports made on

			different blog posts
CommentReport	CommentReport[]		A user's list of reports made on different comments on blog posts
BlogVote	BlogVote[]		A user's list of upvotes and downvotes on a blog post
CommentVote	CommentVote[]		A user's list of upvotes and downvotes on a comment on a blog post

USER API ENDPOINTS

Signup API Handler

Description: Creates a new user based on the JSON payload.

Method: POST

URL: http://localhost:3000/api/users/signup

Headers:

Content-Type: application/json

Authorization: None

Payload:

- **email (string)**: email associated with the user account
- **username (string):** username associated with user account
- **password (string):** password associated with the user account
- **firstName (string):** user's first name
- **lastName (string)**: user's last name
- **phoneNumber (BigInt):** phone number associated with user account
- **avatar (String?):** avatar path which leads to the user's selected avatar

- **role (String?):** optional role (ADMIN or USER), if user does not specify they are admin, automatically are granted USER role

Example Request:

```
{
   "email": "test123@gmail.com",
   "username": "test1234",
   "password": "tester@123",
   "firstName": "Test",
   "lastName": "User",
   "phoneNumber": 6476476477,
   "avatar": "@/scriptorium/public/uploads/avatars/man.png",
   "role": "USER"
}
```

Example Response:


```
{
   "message": "User created successfully",
   "user": {
       "id": 23,
       "email": "test123@gmail.com",
       "password": "$2b$10$7sUbPqkCjLwwQ0cs/tls8ueRf1PnaU1VzO4Xg/K7XNZy1WnoCrSVi",
       "username": "test1234",
       "firstName": "Test",
       "lastName": "User",
       "phoneNumber": "6476476477",
       "avatar": "@/scriptorium/public/uploads/avatars/man.png",
       "createdAt": "2024-11-03T18:05:25.135Z",
       "role": "USER"
   }
}
```


Login API Handler

Description: Logs in the user and creates a new access token and refresh token for the user.

Method: POST

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

Headers:

Content-Type: application/json

Authorization: None

Payload:

- **email (string)**: email associated with the user account
- **password (string):** password associated with the user account

Example Request:

```
{
   "email": "test123@gmail.com",
   "password": "tester@123"
}
```

Example Response:


```
{
   "message": "Login successful",
   "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjIzLCJ1c2VybmFtZSI6InRlc3QxMjM0Iiwi
aWF0IjoxNzMwNjU3MTg1LCJleHAiOjE3MzA2NTgwODV9.5KLaJ9OsBD3poHY9jp4idGxzxmO0nvSNmd_Pugzhk
L8"
}
```


Logout API Handler

Description: Logs out the currently logged in user and clears all cookies in the header.

Method: POST

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

Headers:

Content-Type: application/json

Authorization: None

Payload: None

Example Request: POST Request to api/auth/logout

Example Response:


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


getAllUsers API Handler

Description: Gets all the user details stored in the user database

Method: GET

URL: http://localhost:3000/api/users/getallusers

Headers:

Content-Type: application/json

Authorization: None

Payload: None

Example Request: GET Request to api/users/getallusers

Example Response:


```
{
    "users": [
        {
            "id": 23,
            "email": "test123@gmail.com",
            "username": "test1234",
            "firstName": "Test",
            "lastName": "User",
            "phoneNumber": "6476476477",
```

```
            "avatar": "@/scriptorium/public/uploads/avatars/man.png",
            "role": "USER"
        }
    ]
}
```


getWithID API Handler

Description: Gets the user details of a specific user in the user database.

Method: GET

URL: http://localhost:3000/api/users/[id].js

Headers:

Content-Type: application/json

Authorization: None

Payload: None

Example Request: GET Request to api/users/23

Example Response:


```
{
    "user": {
        "id": 23,
        "email": "test123@gmail.com",
        "password": "$2b$10$7sUbPqkCjLwwQ0cs/tls8ueRf1PnaU1VzO4Xg/K7XNZy1WnoCrSVi",
        "username": "test1234",
        "firstName": "Test",
        "lastName": "User",
        "phoneNumber": "6476476477",
        "avatar": "@/scriptorium/public/uploads/avatars/man.png",
        "createdAt": "2024-11-03T18:05:25.135Z",
        "role": "USER"
    }
}
```

updateWithID API Handler

Description: Updates the user details of a specific user in the user database.

Method: PUT

URL: http://localhost:3000/api/users/[id].js

Headers:

Content-Type: application/json

Authorization: None

Payload:

- **email (String?)**: email associated with the user account
- **username (String?):** username associated with user account
- **password (String?):** password associated with the user account
- **firstName (String?):** user's first name
- **lastName (String?)**: user's last name
- **phoneNumber (BigInt?):** phone number associated with user account
- **avatar (String?):** avatar path which leads to the user's selected avatar
- **role (String?):** optional role (ADMIN or USER), if user does not specify they are admin, automatically are granted USER role

These are all optional because the user may not want to update all their information in one request.

Example Request: PUT Request to api/users/23

```
{
   "email": "uptatedtester123@gmail.com"
}
```

Example Response:

```
{
   "message": "User updated successfully",
   "user": {
       "id": 23,
       "email": "uptatedtester123@gmail.com",
       "password": "$2b$10$7sUbPqkCjLwwQ0cs/tls8ueRf1PnaU1VzO4Xg/K7XNZy1WnoCrSVi",
       "username": "test1234",
```

```
        "firstName": "Test",
        "lastName": "User",
        "phoneNumber": "6476476477",
        "avatar": "@/scriptorium/public/uploads/avatars/man.png",
        "createdAt": "2024-11-03T18:05:25.135Z",
        "role": "USER"
    }
}
```


deleteWithID API Handler

Description: Deletes the user details of a specific user in the user database.

Method: DELETE

URL: http://localhost:3000/api/users/[id].js

Headers:

Content-Type: application/json

Authorization: None

Payload: None

Example Request: DELETE Request to api/users/23

Example Response:


```
{
    "message": "User deleted successfully"
}
```

BLOG POSTS

BLOG MODEL:

Field	Type	Description
id	Int	The ID of the blog post
userId	Int	The ID of the user that made the blog post
user	User	The user associated with userId
title	String	The title of the blog post
description	String	The description of the blog post
tags	String?	The tags of the blog post, a comma separates each individual tag in the string (i.e., “tag1,tag2,tag3”)
upvotes	Int	The number of upvotes this blog post has received
downvotes	Int	The number of downvotes this blog post has received
reports	Int	The number of reports this blog post has received
createdAt	DateTime	When the blog post was created
comments	Comment[]	The comments that are on this blog post
templates	Template[]	The templates linked to this blog post
hidden	Boolean	Indicates whether this blog post is hidden or not
BlogVote	BlogVote[]	List of upvotes/downvotes associated with the blog

BLOG ENDPOINTS

Endpoint	Payload	Example
api/blogs/ Allows a GET request to be sent to retrieve blogs with certain characteristics. All payload parameters	**id (Int):** Only return the blog post with this ID **title (String):** Return blog posts with titles containing this query	

are optional; if none are provided, all blogs are returned.	**description (String):** Return blog posts with descriptions containing this query **tags (String):** Return blog posts that contain the specified tags (tags is a comma-separated string of tags) **templateQuery (String):** Return blog posts that link templates that have titles or explanations containing this query **templateTags (String):** Return blog posts that link templates that have tags contained in this query **order (String):** Orders the response in descending order of “upvotes”, “downvotes”, or “reports”. Ordering by reports can only be done if the currently logged-in user has the role “ADMIN”.	
api/blogs/create Allows a POST request to create a new blog post. Title and description are required, while tags and templateIds are optional.	**title (String):** Title of the blog post **description (String):** Description of the blog post **tags (String):** A comma-separated string representing the tags to put in the blog post **templateIds (String):** A comma-separated string of integers representing the templateIds to link in the blog post	

api/blogs/edit Allows a PUT request to edit an existing blog post. Blog ID must be valid and the currently logged-in user must own the blog post. Title, description, tags, and templateIds are all optional.	**id (Int):** Edit blog post with this ID **title (String)**: New title for blog post **description (String):** New description for blog post **tags (String):** Comma separated string, new tags for blog post **templateIds (String):** Comma separated string of integers, new template IDs for blog post	
api/blogs/delete Allows a DELETE request to delete an existing blog post. Blog ID must be valid and the currently logged-in user must own the blog post.	**id (Int):** Delete the blog post with this ID	
api/blogs/upvote Allows a POST request to upvote or unupvote a blog post. A user must be logged-in to use this endpoint.	**id (Int):** Upvote the blog post with this ID **diff (Int):** If diff == 1, then add an upvote to this blog post. If diff == -1, then unupvote this blog post. If a user had previously downvoted this post but decides to upvote it, the downvote will be removed.	
api/blogs/downvote Allows a POST request to downvote or undownvote a blog post. A user must be logged-in to use this endpoint.	**id (Int):** Downvote the blog post with this ID **diff (Int):** If diff == 1, then add a downvote to this blog post. If diff == -1, then undownvote this blog post. If a user had previously upvoted this post but decides to downote it, the	

	upvote will be removed.	
api/blogs/report Allows a POST request to report a blog post. A user must be logged-in to use this endpoint.	**id (Int):** Report the blog post with this ID **explanation (String):** The reason why this blog post was reported.	
api/blogs/hide Allows a PUT request to hide a blog post. A user must have the role “ADMIN” to use this endpoint.	**id (Int):** Hide the blog post with this ID	
api/blogs/unhide Allows a PUT request to unhide a blog post. A user must have the role “ADMIN” to use this endpoint.	**id (Int):** Unhide the blog post with this ID	

COMMENT MODEL

Field	Type	Description
id	Int	The ID of the comment
userId	Int	The ID of the user that made the comment
user	User	The user associated with userId
content	String	The content of the comment

upvotes	Int	The number of upvotes this comment has received
downvotes	Int	The number of downvotes this comment has received
Comment Vote	CommentVote[]	The upvotes and downvotes this comment has received
reports	Int	The number of reports this comment has received
Comment Report	CommentReport[]	The reports this comment has received
createdAt	DateTime	When the comment was created
parentId	Int?	If the comment is a reply, what is the ID of the comment it is a reply to?
parent	Comment	The parent comment
replies	Comment[]	The replies to this comment
blogId	Int	The ID of the blog post this comment is on
blog	Blog	The blog associated with blogId
hidden	Boolean	Indicates whether this comment is hidden or not

COMMENT ENDPOINTS

Endpoint	**Payload**	**Example**
api/blogs/comments Allows a GET request to be sent to retrieve blogs with certain characteristics. All payload parameters are optional; if none are provided, all blogs are returned.	**blogId (Int):** Get the comments of the blog post with this id **content (String):** Get t **order (String):** Orders the response in descending order of “upvotes”, “downvotes”, or “reports”. Ordering by reports can only be done if the currently logged-in user has the role “ADMIN”.	
api/blogs/comments/	**blogId (Int):** The blog ID of	

create Allows a POST request to create a new comment.	the blog that will be commented on **content (String):** The content of this comment **parentId (Int):** An optional parameter, the ID of the comment you want to reply to	
api/blogs/comments/edit Allows a PUT request to edit an existing comment. Comment ID must be valid and the currently logged-in user must own the comment.	**id (Int):** Edit blog post with this ID **content (String)**: The new content of the comment	
api/blogs/comments/upvote Allows a POST request to upvote or unupvote a comment. A user must be logged-in to use this endpoint.	**id (Int):** Upvote the comment with this ID **diff (Int):** If diff == 1, then add an upvote to this comment. If diff == -1, then unupvote this comment. If a user had previously downvoted this post but decided to upvote it, the downvote will be removed.	
api/blogs/comments/downvote Allows a POST request to downvote or undownvote a comment. A user must be logged-in to use this endpoint.	**id (Int):** Downvote the comment with this ID **diff (Int):** If diff == 1, then add a downvote to this comment. If diff == -1, then undownvote this commentt. If a user had already previously upvoted this post but decides to downote it, the upvote will be removed.	
api/blogs/comments/report	**id (Int):** Report the comment with this ID	

Allows a POST request to report a comment. A user must be logged-in to use this endpoint.	**explanation (String):** The reason why this comment was reported.	
api/blogs/comments/hide Allows a PUT request to hide a comment. A user must have the role “ADMIN” to use this endpoint.	**id (Int):** Hide the comment with this ID	
api/blogs/unhide Allows a PUT request to unhide a comment. A user must have the role “ADMIN” to use this endpoint.	**id (Int):** Unhide the comment with this ID	

BLOGREPORT MODEL

Field	**Type**	**Description**
id	Int	The ID of the report
userId	Int	The ID of the user that made the report
blogId	Int	The ID of the blog reported
explanation	String?	The explanation for the report
user	User	The user associated with userId
blog	Blog	The blog associated with blogId

createdAt	DateTime	When the report was created

COMMENTREPORT MODEL

Field	Type	Description
id	Int	The ID of the report
userId	Int	The ID of the user that made the report
commentId	Int	The ID of the comment reported
explanation	String?	The explanation for the report
user	User	The user associated with userId
comment	Comment	The comment associated with commentId
createdAt	DateTime	When the report was created

BLOGVOTE MODEL

Field	Type	Description
id	Int	The ID of the vote
userId	Int	The ID of the user that made the vote
blogId	Int	The ID of the blog voted for
type	String	The type of vote, either “UPVOTE” or “DOWNVOTE”
user	User	The user associated with userId
blog	Blog	The blog associated with blogId
createdAt	DateTime	When the vote was created

COMMENTVOTE MODEL

Field	Type	Description
id	Int	The ID of the vote

userId	Int	The ID of the user that made the vote
commentId	Int	The ID of the comment voted for
type	String	The type of vote, either “UPVOTE” or “DOWNVOTE”
user	User	The user associated with userId
blog	Blog	The comment associated with commentId
createdAt	DateTime	When the vote was created

CODE EXECUTION (For visitors)

/api/execute

Description: Executes code provided in the request payload.

Method: POST

Payload:

UserId (Integer): The id of the user.

language (string): Programming language for the code (Python, Java, Javascript,C,C++).

code (string): Code to execute.

stdin (array of strings): Standard input values to feed into the code execution.

saveAsTemplate (boolean): Indicates whether to save the executed code as a template.

title (string, required if saveAsTemplate is true): Title of the template if saved.

explanation (string, required if saveAsTemplate is true): Description of the template if saved**.**

tags (string, required if saveAsTemplate is true): Tags for the template if saved.

Example Request:(Just runs the code)

URL: http://localhost:3000/api/execute

Headers:

Content-Type: application/json

{

```
  "language": "python",
  "code": "name = input()\nprint('Hello, ' + name)",
  "stdin": ["Alice"],
  "saveAsTemplate": false
}
```

Example Response:

```
{
  "output": "Hello, Alice\n",
  "exitCode": 0
}
```

Example Request: **(Also saves it as a template)**

URL: http://localhost:3000/api/execute

Headers:
Content-Type: application/json

Authorization: Bearer {{access_token}} (In the **Authorization** Section)

```
{
  "language": "python",
  "code": "name = input()\nprint('Hello, ' + name)",
  "stdin": ["Alice"],
  "saveAsTemplate": true,
  "title": "Greeting Code Template",
  "explanation": "This template takes a name as input and prints a greeting.",
  "tags": "greeting, example"
}
```

Example Response:

```
{
  "output": "Hello, Alice\n",
  "exitCode": 0,
  "templateId": 15,
  "message": "Code executed and template saved successfully"
}
```

CODE TEMPLATES

Template Model

Field	Type	Description
id	Integer	Primary key, unique identifier for the template
userId	Integer	The ID of the user that made the template
title	String	The title of the template
explanation	String	Description explaining what the template does
tags	String?	Comma-separated tags for the templates
code	String	The actual code for the template
stdin	String?	JSON string of standard inputs for the code
language	String	Programming language of the code
forkedFrom	Integer	ID of the original template, if forked
createdAt	DateTime	Timestamp when the template was created
updatedAt	DateTime	Timestamp when the template was last updated

List of all API endpoints (All inside **/api/templates/)**

/api/templates/run-template

Description: Runs code from an existing template using its id. (only for Authorized Users)

Method: POST

URL: http://localhost:3000/api/templates/run-template

Headers:

Content-Type: application/json

Authorization: Bearer {{access_token}} (In the **Authorization** Section)

Payload:

templateId (integer): id of the template to run.

Example Request:

```
{
  "templateId": 15
}
```

Example Response:

```
{
  "output": "Hello, Alice\n",
  "exitCode": 0
}
```

/api/templates/create

Description: Creates a new code template.

Method: POST

URL: http://localhost:3000/api/templates/create

Headers:

Content-Type: application/json

Authorization: Bearer {{access_token}} (In the **Authorization** Section)

Payload:

- **userId**: Id of the user
- **title (string):** Title of the template.
- **explanation (string):** Explanation of the template.
- **tags (string):** Tags for the template.

- **code (string)**: The actual code for the template.
- **stdin (array of strings):** Standard inputs for the code.
- **language (string):** Language in which the code is written.

Example Request:


```
{
  “userId”:1,
  "title": "Java HelloWorld Template",
  "explanation": "A simple template for Java Hello World.",
  "tags": "java, example, hello world",
  "code": "public class Main { public static void main(String[] args) {
System.out.println(\"Hello, World!\"); } }",
  "stdin": [],
  "language": "java"
}
```


Example Response:


```
{
  "template": {
    "id": 16,
    "userId": 9,
    "title": "Java HelloWorld Template",
    "explanation": "A simple template for Java Hello World.",
    "tags": "java, example, hello world",
    "code": "public class Main { public static void main(String[] args) {
System.out.println(\"Hello, World!\"); } }",
    "stdin": "[]",
    "language": "java",
    "createdAt": "2024-11-01T00:00:00.000Z",
    "updatedAt": "2024-11-01T00:00:00.000Z"
  }
}
```


/api/templates/search

Description: Searches for templates by title, tags, or explanation.

Method: GET

URL: http://localhost:3000/api/templates/search

Headers:

Content-Type: application/json

Authorization: Bearer {{access_token}} (In the **Authorization** Section)

Query Parameters:

query (string): Search term to match across title, tags, or explanation.

tags (string): Comma-separated list of tags to filter by.

Example Request: http://localhost:3000/api/templates/search?query=hello

Example Request: http://localhost:3000/api/templates/search?tags=example

Example Response:


```
{
  "templates": [
    {
      "id": 15,
      "title": "Greeting Code Template",
      "explanation": "This template takes a name as input and prints a greeting.",
      "tags": "greeting, example",
      "code": "name = input()\nprint('Hello, ' + name)",
      "language": "python",
      "createdAt": "2024-11-01T00:00:00.000Z",
      "updatedAt": "2024-11-01T00:00:00.000Z"
    }
  ]
}
```


/api/templates/[id]

Description: Retrieves, updates, or deletes a specific template by id.

Headers:

Content-Type: application/json

Authorization: Bearer {{access_token}} (In the **Authorization** Section)

Allowed Methods:

GET: Retrieves a specific template.

PUT: Updates a specific template's tags, explanation, or code.

DELETE: Deletes a specific template.

Payload for PUT:

tags (string): Updated tags for the template.

explanation (string): Updated explanation.

code (string): Updated code.

Example Request (GET):

GET /api/templates/15
URL: http://localhost:3000/api/templates/15
Example Response (GET):


```
{
  "template": {
    "id": 15,
    "title": "Greeting Code Template",
    "explanation": "This template takes a name as input and prints a greeting.",
    "tags": "greeting, example",
    "code": "name = input()\nprint('Hello, ' + name)",
    "language": "python",
    "createdAt": "2024-11-01T00:00:00.000Z",
    "updatedAt": "2024-11-01T00:00:00.000Z"
  }
}
```


PUT /api/templates/15
URL: http://localhost:3000/api/templates/15
Example Request (PUT):


```
{
  "tags": "greeting, updated",
  "explanation": "This is an updated explanation.",
  "code": "name = input()\nprint('Hello, ' + name + '!')"
}
```


Example Response (PUT):


```
{
```

```
  "message": "Template updated successfully",
  "template": {
    "id": 15,
    "title": "Greeting Code Template",
    "explanation": "This is an updated explanation.",
    "tags": "greeting, updated",
    "code": "name = input()\nprint('Hello, ' + name + '!')",
    "language": "python",
    "createdAt": "2024-11-01T00:00:00.000Z",
    "updatedAt": "2024-11-01T01:00:00.000Z"
  }
}
```


DELETE /api/templates/15
URL: http://localhost:3000/api/templates/15

Example Response:


```
{
  "message": "Template deleted successfully"
}
```