Scriptorium

🚀 Meldy Asili Bile

This documentation describes CRUD operations and tests for the CSC309 Scriptorium Backend.

Introduction

Step 1: Set Up Environement

To set up the environment, please run the `run.sh` and `startup.sh` files.

For running tests which require authorisation, please use the registered admin user: 'admin' with password 'admin123'.

Step 2: Send requests

This project allows for a range of POST, GET, PUT, and DELETE HTTP methods.

Open each request and click "Send" to see what happens.

Step 2: View responses

Observe the response tab for status code (200 OK), response time, and size.

Step 3: Send new Body data

Update or add new data in "Body" in the POST request. Typically, Body data is also used in PUT request.

Plain Text

```
{
    "name": "Add your name in the body"
}
```

References:

Yash - AI & Growth. (2022, March 23). *How to create API documentation in Postman [under 5 Minutes] 2023* [Video]. YouTube. https://www.youtube.com/watch?v=6_bYRsbeyCc

Replit. (n.d.). *CSC309 W6*. Replit. https://replit.com/@kianoosh76/CSC309-W6?v=1#README.md

Replit. (n.d.-a). *CSC309 W5*. Replit. https://replit.com/@kianoosh76/CSC309-W5?v=1#README.md

POST RegisterUser

https://postman-rest-api-learner.glitch.me/api/users/register

This is a POST request, submitting data to an API via the request body. This request submits JSON data, and the data is reflected in the response. The test here registers a user by providing JSON data and outputs a response code given the scenario.

A successful POST request returns a `201 Created` response code.

Body raw (json)

```json
{
  "email": "newuser@example.com",
  "password": "SecurePassword123",
  "firstName": "John",
  "lastName": "Doe",
  "avatar": "https://example.com/avatar.png",
  "phone": "1234567890"
}
```

POST Login

https://postman-rest-api-learner.glitch.me/api/users/login

This POST request, submits data to the API to allow a user to login. If the user data is found in the database, the request returns a `200 OK` response. The request is dependent upon utilising a hashed password from the database, hence, the token generated at registration must be used here.

Body raw (json)

```json
{
  "email": "newuser4@example.com",
  "password": "SecurePassword123"
}
```

PUT editProfile

https://postman-rest-api-learner.glitch.me/api/users/edit

This PUT request is used to overwrite existing data in a user's profile. The user is identified by their id number and authorisation is implemented to ensure a user can only update their own profile. This uses the generated token to allow for use of plain text for the JSON data request.

A successful PUT request returns a `200 OK` response code. If an unauthorised user attempts to edit a profile, a `403 Forbidden` response is output.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```json
{
  "userId": 5,
  "firstName": "UpdatedFirstName",
  "lastName": "UpdatedLastName",
  "avatar": "https://example.com/avatar.png",
  "phone": "1234567890"
}
```

POST createBlog

https://postman-rest-api-learner.glitch.me/api/blogs/create

This is an endpoint allowing registered users to create blog posts. The POST request is used to input data into the prisma database.

A successful response will have a `201 CREATED` status.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```json
{
  "title": "My First Blog Post",
  "description": "This is a description of my first blog p
  "content": "This is the full content of the blog post.",
  "tags": "blog, first, introduction"
}
```

PUT editBlog

https://postman-rest-api-learner.glitch.me/api/blogs/edit

This PUT request allows users to edit existing blog posts with updated data. For security and functionality, the user must be authorised i.e. logged into their account in order to make edits and will only be able to edit blogs which they have written.

A successful response will have a `200 OK` status.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```
{
  "id": 1,
  "title": "My Updated Blog Post",
  "description": "Updated description.",
  "content": "Updated content.",
  "tags": "updated, blog"
}
```

DELETE deleteBlog

https://postman-rest-api-learner.glitch.me/api/blogs/delete

This is DELETE request allowing users to delete their own blog posts. The request deletes blogs given the blogPost id in the prisma database.

A successful response will have a `204 Success - No Content` status.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```json
{
  "id": 1
}
```

POST blogComment

https://postman-rest-api-learner.glitch.me/api/blogs/:id/comments

This is an endpoint allowing registered users to comment on existsing blog posts on the system. The POST request is used to input data into the prisma database and increment the comments model.

The system ensures that the request body is not null, returning `400 Bad Request` if it is and successful response will have a `201 CREATED` status.

AUTHORIZATION Bearer Token

Token	<token>

PATH VARIABLES

id

Body raw (json)

json

```json
{
    "content": "This is a comment on the blog post."
}
```

POST blogVote

https://postman-rest-api-learner.glitch.me/api/blogs/:id/vote

This is an endpoint allowing registered users to vote 'up' or 'down' on existsing blog posts on the system. The POST request is used to input data into the prisma database and increment the vote count for blogs.

A successful response will have a `200 OK` status.

AUTHORIZATION Bearer Token

Token	<token>

PATH VARIABLES

id

Body raw (json)

json

```json
{
    "voteType": "up"
}
```

POST codeExecution

https://postman-rest-api-learner.glitch.me/api/code/execute

This POST request allows users and visitors to execute code.

A successful POST response will have a `200 OK` status and return a response body of the compiled code.

Body raw (json)

json

```json
{
  "language": "PYTHON",
  "code": "print('Hello World')",
  "input": ""
}
```

POST createTemplate

https://postman-rest-api-learner.glitch.me/api/templates/create

This is an endpoint allowing registered users to create templates. The POST request is used to input data into the prisma database into the

Templates table. In order to create a new template, the user must be registered, which is authorised using their generated token.

A successful response will have a `201 CREATED` status and a new template is generated.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```json
{
  "title": "Second Template",
  "explanation": "This template is an example for testing.
  "tags": "example, testing",
  "code": "console.log('Hello World');",
  "language": "JavaScript",
  "userId": 1
}
```

GET getTemplate

https://postman-rest-api-learner.glitch.me/api/templates/:id

This is a GET request and it is used to "get" template data from the database. This can be used by either a registered user or a site visitor, hence authorrisation is not needed. As a GET request, there is no request body, but we use query parameters to specify the resource to get data for.

A successful GET response will have a `200 OK` status.

PATH VARIABLES

id

PUT editTemplateUser

https://postman-rest-api-learner.glitch.me/api/templates/edit/:id

This PUT request allows users to edit existing templates with updated data given the id of the template. A user will be able to make edits to their template.

A successful response will have a `200 OK` status.

AUTHORIZATION Bearer Token

Token	<token>

PATH VARIABLES

id

Body raw (json)

json

```
{
    "title": "Updated Template Title",
    "explanation": "Updated explanation for the template.'
    "tags": "updated, tags",
    "code": "console.log('Updated Hello World');",
    "language": "JavaScript"
}
```

PUT editTemplateVisitor

https://postman-rest-api-learner.glitch.me/api/templates/edit/:id

This PUT request allows visitors to edit existing templates with updated data by creating a fork of the original template.

A successful response will have a `200 OK` status.

AUTHORIZATION Bearer Token

Token	<token>

PATH VARIABLES

id

Body raw (json)

json

```
{
    "title": "Attempt to Update",
    "explanation": "Should not be allowed.",
    "tags": "attempt, update",
    "code": "console.log('Hello World');",
    "language": "JavaScript"
}
```

DELETE deleteTemplate

https://postman-rest-api-learner.glitch.me/api/templates/delete/:id

This DELETE request allows users to delete existing templates with updated data given the id of the template.

A successful response will have a `200 OK` status and output a message.

AUTHORIZATION Bearer Token

Token	<token>

PATH VARIABLES

id

Body raw (json)

json

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

POST reportContent

https://postman-rest-api-learner.glitch.me/api/report

This is a POST request, submitting data to an API via the request body. This request allows a registered user to report inappropriate content including blog posts and/or blog comments.

A successful POST request returns a `201 Created` response code.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```json
{
  "explanation": "This post contains abusive language.",
  "userId": 1,
  "blogPostId": 1
}
```

POST hideContent

https://postman-rest-api-learner.glitch.me/api/admin/hideContent

This is a POST request, submitting data to an API via the request body. This request allows a registered admin user to hide content, making it inaccessible to other users.

A successful POST request returns a `201 Created` response code.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```json
{
  "contentId": 1,
  "contentType": "BlogPost"
}
```

GET sortPosts

https://postman-rest-api-learner.glitch.me/api/admin/sort?contentType=BlogPost

This is a GET request, returning a sorted list of blog posts. In order to sort the post, the user must be registered and categorised as an admin user.

A successful GET request returns a `200 OK` response code.

AUTHORIZATION Bearer Token

Token	<token>

PARAMS

contentType	BlogPost

GET authorViewHidden

https://postman-rest-api-learner.glitch.me/api/posts/:postId

This is a GET request, returning a hidden blog post to the author of said post. In order to sort the post, the user must be logged into their account and the author of the hidden post.

A successful GET request returns a `200 OK` response code.

AUTHORIZATION Bearer Token

Token	<token>

PATH VARIABLES

postId