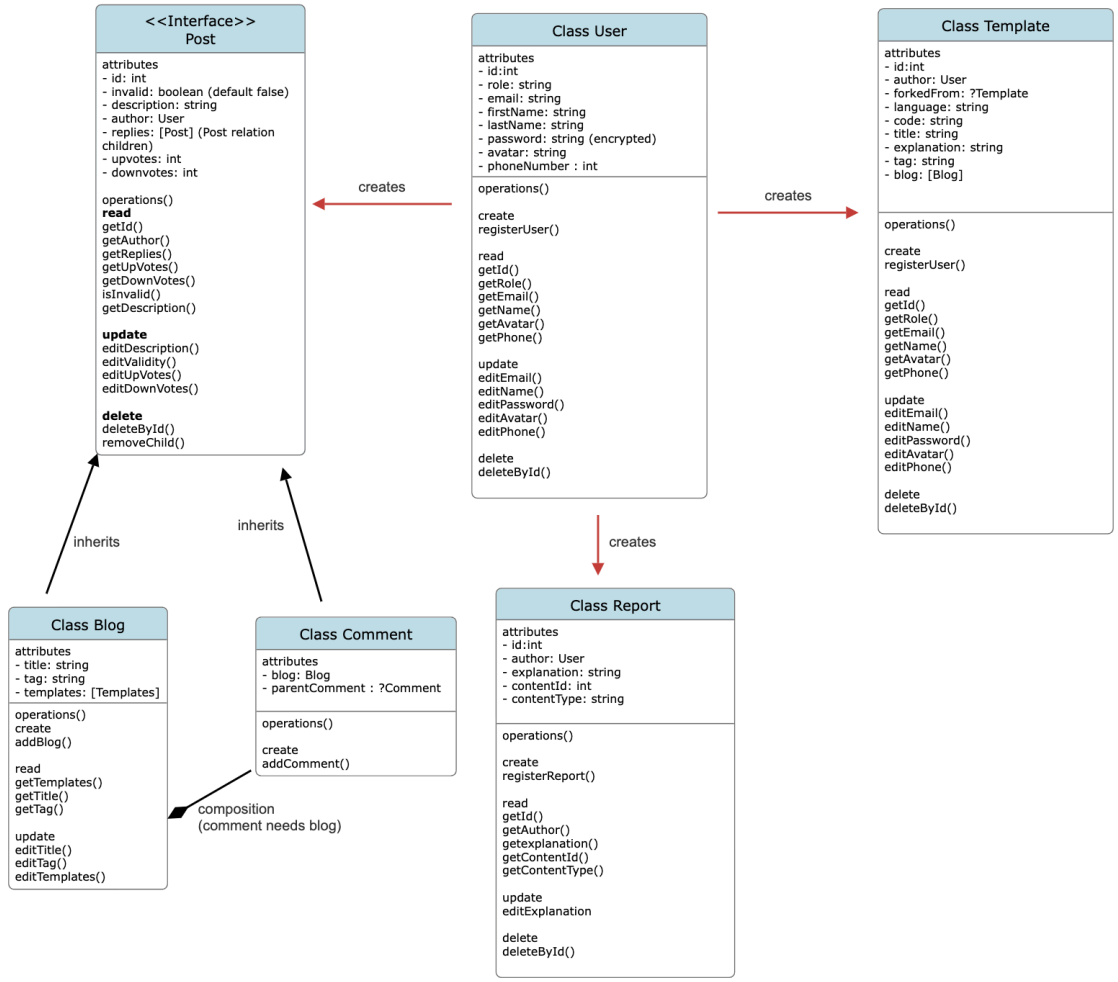
Group: Hannah Omeiza, Gregory Gismondi, Karyna Lim

Model design:



Admin information

email	password	firstName	lastName	phoneNumber	role
'admin@example.com'	'SecurePassword123!'	'Sys'	'Admin'	'0000000000'	'SYS_ADMIN'

API endpoints

<u>Karyna's endpoints</u>

1) Endpoint: /api/users/signup
Description: Registers a new user with the provided details.
Method: POST
Headers:

- Content-Type: application/json

Req payload: Raw JSON
Body Parameters:

- firstName (String): User's first name.
- lastName (String): User's last name.
- password (String): User's password.
- confirmPassword (String): Confirmation of the user's password.
- email (String): User's email address.

- phoneNumber (String): User's phone number.

Responses:

- @apiSuccess (201) Created user successfully
- @apiError (400) ValidationErrors/User error Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only POST requests are allowed.
- @apiError (422) UnprocessableEntity Errors in user creation.

Example request


```
{
 "firstName": "John",
 "lastName": "Doe",
 "email": "johndoe@example.com",
 "password": "SecurePassword1!",
 "confirmPassword": "SecurePassword1!",
 "phoneNumber": "1234567890"
}
```


Example response


```
{"user": {
    "id": 7,
    "firstName": "John",
    "lastName": "Doe",
    "email": "johnsdfdoe@example.com",
    "phoneNumber": "1234567890",
    "role": "user",
    "createdAt": "2024-11-03T07:25:21.842Z",
    "avatar": "uploads/default-avatar.png"
  }}
```


2) Endpoint: /api/users/login

Method: POST

Description: Authenticates a user and returns a token upon successful login.

Headers:

- Content-Type: application/json

Req payload: Raw JSON

Body Parameters:

- email (String): User's email address.
- password (String): User's password.

Responses:

- Success (200): Login successful
- * @apiError (400) Additional validation issues
- * @apiError (404) UserNotFound User with the specified email/password was not found.
- * @apiError (405) MethodNotAllowed Only POST requests are allowed.
- * @apiError (422) LoginFailed unexpected issues during login.

Example request:


```
{
 "email": "johnsdfdoe@example.com",
 "password": "SecurePassword1!"
}
```


Example response:

```
{
  "message": "Login successful",
  "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjcsInJvbGUiOiJ1c2VyIiwiaWF0IjoxNzMwNjE4OTc3LCJleHAiOjE3MzA
2MTk4Nzd9.InJYrWUyjcS4ABGwmtHUw0-KK4ie1nDjIhOaebXA8IM",
  "refreshToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjcsImlhdCI6MTczMDYxODk3NywiZXhwIjoxNzMxMjIzNzc3fQ.Aof8zI
BFRwN0pblZti3naRYtAzr3cqBMzuEXUR7c1Vw",
  "user": {
    "id": 7,
    "firstName": "John",
    "lastName": "Doe",
    "email": "johnsdfdoe@example.com",
    "role": "user"
  }
}
```


3) Endpoint: /api/users/update
Method: PUT
Description: Updates at least 1 attribute of a registered User. Role attribute is not modifiable. Avatar attribute must be an existing photo from /public/avatar_images.
Headers:

- Content-Type: application/json
- Authorization: User's access token

Req payload: Raw JSON
Body Parameters:

- firstName (String): User's first name.
- lastName (String): User's last name.
- password (String): User's password.
- confirmPassword (String): Confirmation of the user's password.
- email (String): User's email address.
- phoneNumber (String): User's phone number.
- avatar (String): User's avatar image, maintained as a file path to the image.
- At Least 1 parameter must be provided

Responses:

- * @apiSuccess (200) Successful update
- * @apiError (400) ValidationError Validation errors if fields are invalid or at least 1 field wasn't provided.
- * @apiError (401) Unauthorised Invalid access token or user not found, must be logged in..
- * @apiError (405) MethodNotAllowed Only PUT requests are allowed.
- * @apiError (409) Caught when duplicate data has been found.
- * @apiError (422) UpdateFailed unable to update user.

Example request:


```
{
 "avatar": "/avatar_images/pfp2.png"
}
```


Example response:


```
{
  "message": "User updated successfully",
  "user": {
```

```
    "id": 7,
    "firstName": "John",
    "lastName": "Doe",
    "email": "johnsdfdoe@example.com",
    "phoneNumber": "1234567890",
    "avatar": "/avatar_images/pfp2.png",
    "updatedAt": "2024-11-03T07:30:28.601Z"
  }
}
```


4) Endpoint: /api/reports/report
Method: POST
Description: Submits a report for specified content (Blog or Comment).
Headers:

- Authorization: User's access token.
- Content-Type: application/json

Req payload: Raw JSON
Body Parameters:

- contentId (Number): ID of the content being reported.
- contentType (String): Type of content being reported, either BLOG or COMMENT.
- explanation (String): Explanation for the report.

Responses:

- Success (201)
- Error (400): Invalid content type.
- Error(401): Unauthorised, invalid token or expired, not logged in
- Error (405): Method not allowed .
- Error (404): Content not found.
- Error (422): Unable to process the request.
-

Example request:


```
{
 "contentId": 123,       // ID of the content (blog or comment)
 "contentType": "BLOG",   // Either "BLOG" or "COMMENT"
 "explanation": "Inappropriate bla-bla-bla." // Explanation for the report
}
```


Example response:


```
{
  "error": "Content not found." // case when blog post doesn't exist
}
```


5) Endpoint: /api/reports/report
Method: GET
Description: Retrieves a paginated list of reported content for review (Admin-only access).
Headers:

- Content-Type: application/json
- Authorization: User's access token

Req payload: Raw JSON
Query Parameters:

- page (Number): Page number for pagination. Default is 1.
- pageSize (Number): Number of items per page. Default is 10.
- contentType (String): type of report to retrieve, Options are BLOG, COMMENT, and BOTH.

Responses:

- Success (200):
- Error (400): Invalid content type.
- Error(401): Unauthorised, invalid token or expired, not logged in
- Error (403): Forbidden for non-admin users.
- Error(404): Resource not found.
- Error (405): Method not allowed .
- Error (422): Unable to process the request.

Example response (Case when User logged in, not sysadmin):

```
{
  "error": "Forbidden: Admin access required to view reports."
}
```

6) Endpoint: /api/auth/refresh
Method: POST
Description: Refreshes Access Token using Refresh token and JWT secrets.
Headers:

- Authorization: User's access token
- x-refresh-token : User's refresh token
- No req body , no params

Responses:

- Success 200: refresh token succesfully
- Error (401): Refresh token not provided
- Error (403): Expired refresh token
- Error (405) : Method unallowed

Example response:

```
{
  "accessToken": "abcde.12345.hello!"
}
```

End of Karyna's endpoint

<u>Hannah's endpoints</u>

1) /api/blog/[Id]/comment/[commentId]/reply

   Method: POST
   Description: add a reply to a comment that exists in a blog
   Restricted to USERS only

   Headers:

   - Content-Type: application/json
   - Authorization: Users access token

   Req payload: Raw JSON
   Body Parameters:

   - Description (String): content of reply

   Responses:

   - @apiSuccess(201): comment created successfully (returns the comment made)
   - @apiError (401): Unauthorised user
   - @apiError (405): Invalid request type
   - @apiError (400): Invalid type of parameters in request

- @apiError (422): failed to create comment

Example request:

```
{
   "description": "This is my second reply."
}
```

Example response:

```
{
   "id": 3,
   "blogId": 2,
   "flagged": false,
   "description": "This is my second reply.",
   "authorId": 1,
   "createdAt": "2024-11-04T21:56:41.903Z",
   "parentId": 1,
   "upvotes": 0,
   "downvotes": 0
}
```

2) /api/blog/[Id]/comment/sorted-rating

Method: GET
Description: get the topmost comments within a blog sorted in descending order by upvotes (paginated default page 1, 10 per page)
If you are a logged in user, flagged comments which you own have an additional report section to tell you the reason for being flagged.

Headers:

- Content-Type: application/json
- Authorization: Users access token (optional)

Responses:

- @apiSuccess (200): successfully fetched sorted blogs
- @apiError (422): Unable to fetch sorted blogs
- @apiError (400): Invalid Blog Id
- @apiError (405): Invalid request type

Example request:

```
GET {{baseUrl}}/blog/sorted-rating
```

Example response:

```
{
   "data": [
       {
           "id": 1,
           "blogId": 2,
           "flagged": true,
           "description": "changed comment cos i'm the admin ahahahha.",
           "authorId": 1,
           "createdAt": "2024-11-04T06:35:17.161Z",
```

```
            "parentId": null,
            "upvotes": 5,
            "downvotes": 2,
            "reports": [
                "Inappropriate blablabla."
            ]
        },
        {
            "id": 4,
            "blogId": 2,
            "flagged": false,
            "description": "This is my second comment.",
            "authorId": 1,
            "createdAt": "2024-11-04T21:59:01.530Z",
            "parentId": null,
            "upvotes": 0,
            "downvotes": 0
        }
    ],
    "pagination": {
        "total": 2,
        "page": 1,
        "limit": 10,
        "totalPages": 1
    }
}
```


3) api/blog/[Id]/comment

Method: POST
Description: create a top level comment within a blog
Restricted to USERS only
Headers:

- Content-Type: application/json
- Authorization: Users access token

Req payload: Raw JSON
Body Parameters:

- Description : content of comment

Responses:

- @apiSuccess(201): comment created successfully (returns the comment made)
- @apiError (401): Unauthorised user
- @apiError (405): Invalid request type
- @apiError (400): Invalid type of parameters in request
- @apiError (422): failed to create comment

Example request


```
{
    "description": "This is my second comment."
}
```


Example response:


```
{
    "id": 4,
    "blogId": 2,
    "flagged": false,
    "description": "This is my second comment.",
```

```
    "authorId": 1,
    "createdAt": "2024-11-04T21:59:01.530Z",
    "parentId": null,
    "upvotes": 0,
    "downvotes": 0
}
```


4) api/blog/[id]/template/[templateId]

Method: PUT
Description: add a template to the blogs list of templates
Restricted to USERS only
Headers:

- Content-Type: application/json
- Authorization: user's access token

Responses:

- @apiSuccess (200): template successfully added to blog
- @apiError (400): Invalid type for request parameters
- @apiError (404): Template not found.
- @apiError (422): Unable to add template to blog
- @apiError (405): invalid request type

Example request:

```
PUT {{baseUrl}}/blog/1/template/1
```

Example response:


```
{
    "id": 1,
    "title": "My Blog Title",
    "description": "This is the description of the blog.",
    "flagged": false,
    "authorId": 1,
    "createdAt": "2024-11-03T20:32:31.379Z",
    "upvotes": 0,
    "downvotes": 0,
    "templates": [
        {
            "id": 1,
            "ownerId": 1,
            "forkedFromId": null,
            "code": "print(\"Python File\")",
            "language": "python",
            "title": "Simple Print",
            "explanation": "A simple python file that only prints one line",
            "createdAt": "2024-11-04T22:00:48.678Z",
            "updatedAt": "2024-11-04T22:00:48.678Z"
        }
    ]
}
```


5) api/blog/[id]/template/[templateId]

Method: DELETE

Description: remove a template from a blogs list of templates
Restricted to USERS only

Headers:

- Content-Type: application/json
- Authorization: user's access token

Responses:

- @apiSuccess (200): template successfully added to blog
- @apiError (400): Invalid type for request parameters
- @apiError (404): Template not found.
- @apiError (422): Unable to add template to blog
- @apiError (405): invalid request type

Example request:

```
DELETE {{baseUrl}}/blog/1/template/1
```

Example response:

```
{
   "id": 1,
   "title": "My Blog Title",
   "description": "This is the description of the blog.",
   "flagged": false,
   "authorId": 1,
   "createdAt": "2024-11-03T20:32:31.379Z",
   "upvotes": 0,
   "downvotes": 0,
   "templates": []
}
```

6) /api/blog/[Id]

Method: PUT
Description: update a blog post, by passing in any of the following valid optional parameters: Title, description, tags, upvotes, downvotes, flagged.

Title, description, tags can only be changed by the owner of the blog
Those who do not own the blog are restricted to updating upvotes and downvotes only
The admin can change anything, and only the admin can change flagged

Restricted to USERS only

Headers:

- Content-Type: application/json
- Authorization: access token

Req payload: Raw JSON
Body Parameters:

- Title (String): Title of the blog
- Description (String): Content of blog
- Tags (List of Strings): Tags associated with the blog (there must be at least 1 tag)
- Upvotes (Integer): New number of upvotes
- Downvotes (Integer): New number of downvotes
- Flagged (Boolean): Blog is reported

Responses:

- @apiSuccess (200): Successfully updated the blog
- @apiError (400): Invalid blog Id or nothing is provided to update.
- @apiError (403): Permission denied.
- @apiError (404): Blog not found.
- @apiError (422): Unable to update blog.
- @apiError (405): invalid request type

Example request

```
{
 "title": "Updated Blog Title",
 "description": "This is the updated description of the blog.",
 "tags": ["javascript", "webdev"],
 "upvotes": 5,
 "downvotes": 0
}
```

Example response:

```
{
   "message": "Blog updated successfully.",
   "blog": {
       "id": 1,
       "title": "Updated Blog Title",
       "description": "This is the updated description of the blog.",
       "flagged": false,
       "authorId": 1,
       "createdAt": "2024-11-03T20:32:31.379Z",
       "upvotes": 5,
       "downvotes": 0,
       "tags": [
           {
               "id": 4,
               "name": "webdev"
           },
           {
               "id": 5,
               "name": "javascript"
           }
       ],
       "templates": []
   }
}
```

7) /api/blog/[Id]

Method: DELETE
Description: Delete everything about a blog post
Restricted to USERS only

Headers:

- Content-Type: application/json
- Authorization: User access token

Responses:

- @apiSuccess (200): Blog was deleted successfully
- @apiError (400): Invalid blog ID
- @apiError (404): Blog not found.
- @apiError (403): Permission denied.
- @apiError (422): Unable to delete blog.
- @apiError (405): invalid request type

Example request: `api/blog/2`
Example response:

```
{
   "message": "Blog was deleted successfully."
}
```

8) /api/blog

Method: POST
Description: create a new blog
Restricted to USERS only

Headers:

- Content-Type: application/json
- Authorization: user access token

Req payload: Raw JSON
Body Parameters:

- Title (String): Title of the blog
- Description (String): Content of blog
- Tags (List of Strings): Tags associated with the blog (there must be at least 1 tag)

Responses:

- @apiSuccess (200): blog created successfully
- @apiError (401): Unauthorised user
- @apiError (422): Unable to create blog
- @apiError (405): Invalid request type

Example request:

```
{
 "title": "My Blog Title",
 "description": "This is the description of the blog.",
 "tags": ["javayo javamama", "python"] // Tags array must be a list of
strings
}
```

Example response:

```
{
```

```
    "id": 4,
    "title": "My Blog Title",
    "description": "This is the description of the blog.",
    "flagged": false,
    "authorId": 1,
    "createdAt": "2024-11-03T20:43:39.203Z",
    "upvotes": 0,
    "downvotes": 0,
    "tags": [
        {
            "id": 1,
            "name": "c++"
        },
        {
            "id": 2,
            "name": "python"
        }
    ],
    "templates": []
}
```


9) Endpoint: /api/blog
Method: GET
Description: get list of blogs by different search criteria. Only 1 title, 1 content and 1 template id at a time, only expect strings. All search criteria are optional. The result is Paginated (default page 1, 10 things per page).

A query param usage (if all fields were filled):

```
GET
{{baseUrl}}/blog?page=1&limit=5&tags=javascript,python&title=myblog&content=
description
```

Example request:

- GET {{baseUrl}}/blog?page=1&limit=2

Example response:


```
{
    "blogs": [
        {
            "id": 1,
            "title": "My Blog Title",
            "description": "This is the description of the blog.",
            "flagged": false,
            "authorId": 1,
            "createdAt": "2024-11-03T20:32:31.379Z",
            "upvotes": 0,
            "downvotes": 0,
```

```
            "author": {
                "id": 1,
                "firstName": "John",
                "lastName": "Doe",
                "email": "johndoe@example.com",
                "role": "user",
            },
            "templates": [],
            "comments": [],
            "tags": [
                {
                    "id": 1,
                    "name": "javascript"
                },
                {
                    "id": 2,
                    "name": "python"
                }
            ]
        },
}
```

10) /api/blog/sorted-rating

Method: GET
Description: get the topmost blogs in descending order by upvotes, paginated (default page 1, 10 things per page)
If you are a logged in user, flagged blogs which you own have an additional report section to tell you the reason for being flagged.

Headers:

- Content-Type: application/json
- Authorization: user access token (optional)

Responses:

- @apiSuccess (200): successfully fetched sorted blogs
- @apiError (422): Unable to fetch sorted blogs
- @apiError (405): Invalid request type

Example request:

```
GET {{baseUrl}}/blog/sorted-rating
```

Example response :

```
{
   "data": [
       {
           "id": 2,
           "title": "My Blog Title",
```

```
            "description": "This is the description of the blog.",
            "flagged": false,
            "authorId": 1,
            "createdAt": "2024-11-03T20:32:52.340Z",
            "upvotes": 1000,
            "downvotes": 0,
            "reports": []
        },
        {
            "id": 1,
            "title": "My Blog Title",
            "description": "This is the description of the blog.",
            "flagged": false,
            "authorId": 1,
            "createdAt": "2024-11-03T20:32:31.379Z",
            "upvotes": 0,
            "downvotes": 0,
            "reports": []
        },
],
   "pagination": {
        "total": 2,
        "page": 1,
        "limit": 10,
        "totalPages": 1
   }
}
```


11) /api/comment/[commentId]

Method: PUT
Description: update any comment (top level/reply) by any of the following criteria: description,flagged, upvotes, downvotes
Restricted to USERS

We can only update comment descriptions that belong to us
If we are not the admin, the flagged value does not get changed
If we are not the comment author or not the admin, we are restricted to only upvotes and downvotes

Headers:

- Content-Type: application/json
- Authorization: User's access token

Req payload: Raw JSON
Body Parameters:

- description (String): Content of comment

- flagged (Boolean): Comment is reported
- Upvotes (Integer): New number of upvotes
- Downvotes (Integer): New number of downvotes

.

Responses:

- @apiSuccess (200): comment was updated successfully
- @apiError (400): Invalid update criteria type or nothing provided to update
- @apiError (404): comment not found.
- @apiError(403): Permission Denied
- @apiError (422): Unable to update comment
- @apiError(405): Invalid request type

Example request:

```
{
 "description": "This is the updated description of the comment.",
 "upvotes": 5,
 "downvotes": 0
}
```

Example response:

```
{
   "id": 1,
   "blogId": 2,
   "flagged": false,
   "description": "This is the updated description of the comment.",
   "authorId": 1,
   "createdAt": "2024-11-04T06:35:17.161Z",
   "parentId": null,
   "upvotes": 5,
   "downvotes": 0
}
```

12) /api/comment/sorted-rating

Method: GET

Description: get the topmost comments in the entire app in descending order by upvotes. Paginated (default page 1, 10 things per page).

If you are a logged in user, flagged comments which you own have an additional report section to tell you the reason for being flagged.

Headers:

- Content-Type: application/json
- Authorization: user access token (optional)

Responses:

- @apiSuccess (200): successfully fetched sorted blogs
- @apiError (422): Unable to fetch sorted blogs
- @apiError (405): Invalid request type

Example request:

```
GET {{baseUrl}}/comment/sorted-rating
```

Example response:


```
{
   "data": [
       {
           "id": 1,
           "blogId": 2,
           "flagged": true,
           "description": "changed comment cos i'm the admin ahahahha.",
           "authorId": 1,
           "createdAt": "2024-11-04T06:35:17.161Z",
           "parentId": null,
           "upvotes": 5,
           "downvotes": 2,
           "reports": [
               "Inappropriate blablabla."
           ]
       },
       {
           "id": 4,
           "blogId": 2,
           "flagged": false,
           "description": "This is my second comment.",
           "authorId": 1,
           "createdAt": "2024-11-04T21:59:01.530Z",
           "parentId": null,
           "upvotes": 0,
           "downvotes": 0
       },
       {
           "id": 5,
           "blogId": 2,
           "flagged": false,
           "description": "This is my second reply.",
           "authorId": 1,
           "createdAt": "2024-11-04T23:41:09.885Z",
           "parentId": 1,
           "upvotes": 0,
           "downvotes": 0
       },
  ],
   "pagination": {
       "total": 3,
       "page": 1,
       "limit": 10,
       "totalPages": 1
   }
}
```


End of Hannah's endpoints

Gregory's Endpoints

1) Endpoint: /api/template
Description: Creates a code template based on given parameters. User must be logged in
Method: POST
Headers:

- Authenticator: Bearer {{accessToken}}
- Content-Type: application/json

Req payload: Raw JSON
Body Parameters:

- forkedFromId (Optional Number): Id of template that the current template is being forked from
- code (string): The code that exists within the template
- language (string): Programming language of code template
- title (string): Title/name of code template
- explanation (string): Explanation of what the code template does
- tags (string[]): Pre-existing or new tags that groups code
- blogs (Optional Number[]): Blogs that reference this code template

Responses:

- @apiSuccess (201) Created template successfully.
- @apiError (400) ValidationErrors Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only POST requests are allowed.
- @apiError (401) Unauthorized Must be logged in.
- @apiError (422) UnprocessableEntity Errors in template creation.

Example request

```
{
  "forkedFromId": 1,
  "code": "print(\"Python File\")",
  "language": "python",
  "title": "Simple Print",
  "explanation": "A simple python file that only prints one line",
  "tags": ["strin ghjgjh g1", "python", "printing", "simple"],
  "blogs": [1, 2]
}
```

Example response

```
{
    "id": 7,
    "ownerId": 1,
    "forkedFromId": 1,
    "code": "print(\"Python File\")",
    "language": "python",
    "title": "Simple Print",
    "explanation": "A simple python file that only prints one line",
    "createdAt": "2024-11-04T17:42:16.750Z",
    "updatedAt": "2024-11-04T17:42:16.750Z",
    "blogs": [
        {
            "id": 1,
            "title": "My Blog Title",
```

```
            "description": "This is the description of the blog.",
            "flagged": false,
            "authorId": 1,
            "createdAt": "2024-11-03T20:32:31.379Z",
            "upvotes": 0,
            "downvotes": 0
        },
        {
            "id": 2,
            "title": "My Blog Title",
            "description": "This is the description of the blog.",
            "flagged": false,
            "authorId": 1,
            "createdAt": "2024-11-03T20:32:52.340Z",
            "upvotes": 0,
            "downvotes": 0
        }
    ],
    "tags": [
        {
            "id": 1,
            "name": "strin ghjgjh g1"
        },
        {
            "id": 2,
            "name": "python"
        },
        {
            "id": 4,
            "name": "printing"
        },
        {
            "id": 5,
            "name": "simple"
        }
    ]
}
```


2) Endpoint: /api/template
Description: Gets a code template based on given search queries
Method: GET
Headers:

- Content-Type: application/json

Query Parameters:

- forkedFromId (Optional Number): Id of forked template to search templates for
- code (Optional string): The code to search templates for
- language (Optional string): Programming language to search templates for
- title (Optional string): Title/name to search templates for

- explanation (Optional string): Explanation to search templates for
- tags (Optional string): Tags separated by commas to search templates for
- blogs (Optional Number): Blogs separated by commas that reference templates to search templates for
- page (int, default = 1): Page of return
- limit (int, default = 10): Size of each page

Responses:

- @apiSuccess (200) Searched template successfully.
- @apiError (400) ValidationErrors Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only GET requests are allowed.
- @apiError (422) UnprocessableEntity Errors in template creation.

Example parameters

- /template?page=1&limit=120&tags=printing,strin ghjgjh g1&language=python&blogs=1,3

Example response


```
{
    "templates": [
        {
            "id": 1,
            "ownerId": 1,
            "forkedFromId": 1,
            "code": "print(\"Python File\")",
            "language": "python",
            "title": "Simple Print",
            "explanation": "A simple python file that only prints one line",
            "createdAt": "2024-11-04T20:55:17.895Z",
            "updatedAt": "2024-11-04T20:55:17.895Z",
            "blogs": [
                {
                    "id": 1,
                    "title": "My Blog Title",
                    "description": "This is the description of the blog.",
                    "flagged": false,
                    "authorId": 1,
                    "createdAt": "2024-11-03T20:32:31.379Z",
                    "upvotes": 0,
                    "downvotes": 0
                },
                {
                    "id": 2,
                    "title": "My Blog Title",
                    "description": "This is the description of the blog.",
                    "flagged": false,
                    "authorId": 1,
                    "createdAt": "2024-11-03T20:32:52.340Z",
                    "upvotes": 0,
                    "downvotes": 0
                },
                {
```

```
                    "id": 3,
                    "title": "My Blog Title",
                    "description": "This is the description of the blog.",
                    "flagged": false,
                    "authorId": 1,
                    "createdAt": "2024-11-03T20:41:49.120Z",
                    "upvotes": 0,
                    "downvotes": 0
                }
            ],
            "tags": [
                {
                    "id": 1,
                    "name": "strin ghjgjh g1"
                },
                {
                    "id": 2,
                    "name": "python"
                },
                {
                    "id": 4,
                    "name": "printing"
                },
                {
                    "id": 5,
                    "name": "simple"
                }
            ]
        }
    ],
    "numPage": 1,
    "numLimit": 120
}
```


3) Endpoint: /api/template/[id]
Description: Updates code template for given id based on given parameters. User must be owner of template or admin
Method: PUT
Headers:

- Authenticator: Bearer {{accessToken}}
- Content-Type: application/json

Req payload: Raw JSON
Body Parameters:

- code (Optional string): The code that exists within the template to be updated
- language (Opiontal string): Programming language of code template to be updated
- title (Optional string): Title/name of code template to be updated
- explanation (Optional string): Explanation of what the code template does to be updated
- tags (Optional string[]): Pre-existing or new tags that groups code to be updated
- blogs (Optional Number[]): Blogs that reference this code template to be updated

Responses:

- @apiSuccess (200) Updated template successfully.
- @apiError (400) ValidationErrors Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only PUT requests are allowed.
- @apiError (401) Unauthorized Must be logged in.
- @apiError (404) NotFound Given ID not found.
- @apiError (403) AccessDenied Do not have correct permission to update.
- @apiError (422) UnprocessableEntity Errors in template creation.

Example request


```
{
  "explanation": "This explanation has been updated",
  "blogs": [1, 4],
  "tags": ["strin ghjgjh g1", "python"]
}
```


Example response


```
{
  "id": 1,
  "ownerId": 1,
  "forkedFromId": 1,
  "code": "print(\"Python File\")",
  "language": "python",
  "title": "Simple Print",
  "explanation": "This explanation has been updated",
  "createdAt": "2024-11-04T20:55:17.895Z",
  "updatedAt": "2024-11-04T23:04:25.126Z",
  "tags": [
    {
      "id": 1,
      "name": "strin ghjgjh g1"
    },
    {
      "id": 2,
      "name": "python"
    }
  ],
  "blogs": [
    {
      "id": 1,
      "title": "My Blog Title",
      "description": "This is the description of the blog.",
      "flagged": false,
      "authorId": 1,
      "createdAt": "2024-11-03T20:32:31.379Z",
      "upvotes": 0,
```

```
      "downvotes": 0
    },
    {
      "id": 4,
      "title": "My Blog Title",
      "description": "This is the description of the blog.",
      "flagged": false,
      "authorId": 1,
      "createdAt": "2024-11-03T20:43:39.203Z",
      "upvotes": 0,
      "downvotes": 0
    }
  ]
}
```


4) Endpoint: /api/template/[id]
Description: Deletes code template for given id. User must be owner of template or admin.
Method: DELETE
Headers:

- Authenticator: Bearer {{accessToken}}
- Content-Type: application/json

Responses:

- @apiSuccess (200) Deleted template successfully.
- @apiError (400) ValidationErrors Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only DELETE requests are allowed.
- @apiError (401) Unauthorized Must be logged in.
- @apiError (404) NotFound Given ID not found.
- @apiError (403) AccessDenied Do not have correct permission to update.
- @apiError (422) UnprocessableEntity Errors in template creation.

Example response


```
{
  "message": "Successfully deleted template"
}
```


5) Endpoint: /api/template/[id]
Description: Gets code template for given id.
Method: GET
Headers:

- Content-Type: application/json

Responses:

- @apiSuccess (200) Updated template successfully.
- @apiError (400) ValidationErrors Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only GET requests are allowed.
- @apiError (404) NotFound Given ID not found.
- @apiError (422) UnprocessableEntity Errors in template creation.

Example response


```
{
  "id": 2,
```

```
  "ownerId": 1,
  "forkedFromId": null,
  "code": "print(\"Python File\")",
  "language": "python",
  "title": "Simple Print",
  "explanation": "A simple python file that only prints one line",
  "createdAt": "2024-11-04T23:10:41.927Z",
  "updatedAt": "2024-11-04T23:10:41.927Z",
  "blogs": [],
  "tags": [
    {
      "id": 2,
      "name": "python"
    }
  ]
}
```


6) Endpoint: /api/template/run
Description: Executes code sent and returns output
Method: GET
Headers:

- Content-Type: application/json

Req payload: Raw JSON
Body Parameters:

- code (string): The code to execute
- language (string): Programming language the code is written in
- inputs (Number[], default = []): Inputs for the code

Responses:

- @apiSuccess (201) Returned template successfully
- @apiError (400) ValidationErrors Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only GET requests are allowed.
- @apiError (422) UnprocessableEntity Errors in template creation.

Example request


```
{
  "language": "python",
  "code": "user_input = input()\nprint(user_input)",
  "inputs": ["Python Input"]
}
```


Example response


```
{
  "status": "pass",
  "output": "Python Input\r\n",
  "error": ""
}
```

7) Endpoint: /api/blog/[id]/template/[templateId]
Description: Links blog and template given by id an templateId respectively. User must be the author of the blog post
Method: PUT
Headers:

- Authorization: Bearer {{accessToken}}
- Content-Type: application/json

Responses:

- @apiSuccess (200) Updated blog and template successfully.
- @apiError (400) ValidationErrors Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only PUT requests are allowed.
- @apiError (401) Unauthorized Must be logged in.
- @apiError (404) NotFound Given ID not found.
- @apiError (403) AccessDenied Do not have correct permission to update.
- @apiError (422) UnprocessableEntity Errors in template creation.

Example response


```
{
  "id": 1,
  "title": "My Blog Title",
  "description": "This is the description of the blog.",
  "flagged": false,
  "authorId": 1,
  "createdAt": "2024-11-03T20:32:31.379Z",
  "upvotes": 0,
  "downvotes": 0,
  "templates": [
    {
      "id": 2,
      "ownerId": 1,
      "forkedFromId": null,
      "code": "print(\"Python File\")",
      "language": "python",
      "title": "Simple Print",
      "explanation": "A simple python file that only prints one line",
      "createdAt": "2024-11-04T23:10:41.927Z",
      "updatedAt": "2024-11-04T23:10:41.927Z"
    }
  ]
}
```


8) Endpoint: /api/blog/[id]/template/[templateId]
Description: Unlinks blog and template given by id an templateId respectively
Method: DELETE
Headers:

- Authorization: Bearer {{accessToken}}
- Content-Type: application/json

Responses:

- @apiSuccess (200) Updated blog and template successfully.

- @apiError (400) ValidationErrors Validation errors if fields are missing or invalid.
- @apiError (405) MethodNotAllowed Only DELETE requests are allowed.
- @apiError (401) Unauthorized Must be logged in.
- @apiError (404) NotFound Given ID not found.
- @apiError (403) AccessDenied Do not have correct permission to update.
- @apiError (422) UnprocessableEntity Errors in template creation.

Example response


```
{
  "id": 1,
  "title": "My Blog Title",
  "description": "This is the description of the blog.",
  "flagged": false,
  "authorId": 1,
  "createdAt": "2024-11-03T20:32:31.379Z",
  "upvotes": 0,
  "downvotes": 0,
  "templates": []
}
```