Scriptorium

General notes

In CSC309, we're building "Scriptorium," an innovative web app where users can write, execute, and manage code in multiple languages. Part 1 of the project focuses on creating the backend with Next.js, Prisma, and REST APIs to support features like code writing, syntax highlighting, and template saving. We'll be working in groups, and each team member is expected to contribute fairly. The deliverable is a fully functional backend that supports essential user stories and will be evaluated through interviews with TAs.

Test Admin (for demonstration):
Email: admin@gmail.com (https://mailto:admin@gmail.com)

Password: admin123@

API detail

Health Check

Checks if the API server is online and functioning. Expected Responses:

200 OK

```
{
  "status": "Online",
  "message": "API is running smoothly."
}
```

500 Internal Server Error

```
{
  "status": "Offline",
  "message": "API is experiencing issues."
}
```

Curl HTTP

```
curl -X GET "http://localhost:3000/api/health"
```

Users

Database Models and Relationships

1. User Model

Fields:

- **id: Int - Primary key, auto-incremented.**
- **firstName: String - User's first name.**
- **lastName: String - User's last name.**
- **email: String - Unique email address.**
- **password: String - Hashed password.**
- **profilePicture: String? - Optional URL or file path.**
- **phoneNumber: String? - Optional contact number.**
- **isAdmin: Boolean - Indicates admin privileges, defaults to false.**
- **templates: CodeTemplate[] - Code templates created by the user.**
- **blogPosts: BlogPost[] - Blog posts authored by the user.**
- **comments: Comment[] - Comments made by the user.**
- **reports: Report[] - Reports submitted by the user.**
- **votes: Vote[] - Votes cast by the user.**

Relationships:

- **One-to-Many with CodeTemplate, BlogPost, Comment, Report, and Vote.**

Usage:

- **Central to user management, authentication, and authorization.**
- **Enables users to create content (templates, posts), interact (comments, votes), and report inappropriate content.**

Create User

Curl HTTP

```
curl -X POST -H "Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4Y
WxkTrZu0gW" -F "firstName=Latch" -F "lastName=Kapoor" -F "email=latch.kapoor@gmail.com"
-F "password=password" -F "avatar=postman-cloud:///1ef9b089-5021-4500-9cdc-9c76f860c3d7"
"http://localhost:3000/api/users"
```

Retrieve User by ID

Retrieves details of a specific user by their ID. This is dependant on whatever post is present with ID 7, it would return that. Below is an example of a successful and an error request.

200 OK

```
{
  "data": {
    "id": 7,
    "firstName": "Adam",
    "lastName": "John",
    "email": "adamjohn@gmail.com",
    "profilePicture": null,
    "phoneNumber": null,
    "isAdmin": false,
    "templates": [...],
    "blogPosts": [...],
    "comments": [...],
    "reports": [...],
    "votes": [...]
  }
}
```

404 Not Found

```
{
  "error": "User not found"
}
```

Curl HTTP

```
curl -X GET "http://localhost:3000/api/users/3"
```

Update User by ID

Updates details of a specific user by their ID. This depens on if the user exist with ID 7. If so it will sucessfully update the User fields specified and will not change the ones not specified. Here is an example of a successful request and 2 error ones.

200 OK

```
{
  "data": {
    "id": 7,
    "firstName": "James",
    "lastName": "Don",
    "email": "jamesdon@gmail.com",
    "profilePicture": "http://example.com/pic.jpg",
    "phoneNumber": "123-456-7890",
    "isAdmin": false,
    "templates": [...],
    "blogPosts": [...],
    "comments": [...],
    "reports": [...],
    "votes": [...]
  }
}
```

404 Not Found

```
{
  "error": "User not found"
}
```

Curl HTTP

```
curl -X PUT -d '{
  "firstName": "James",
  "lastName": "Don",
  "profilePicture": "http://example.com/pic.jpg",
  "phoneNumber": "123-456-7890"
}' "http://localhost:3000/api/users/3"
```

Sessions

This folder is concerned with the creation and management of user sessions using JWT access and refresh tokens.

Create Session

Description: Authenticates a user and creates a new session (login). This can result in either a sucessful creation or an error request. Examples below:

201 Created

```
{
  "data": {
    "sessionId": "unique-session-id",
    "userId": 7,
    "token": "jwt-token-string",
    "createdAt": "2024-05-06T10:00:00.000Z"
  }
}
```

401 Unauthorized

```
{
  "error": "Invalid email or password."
}
```

Curl HTTP

```
curl -X POST -d '{
  "email": "admin1@scriptorium.com",
  "password": "admin123@"
}
' "http://localhost:3000/api/sessions"
```

Refresh Session

Description: Refreshes an existing session token. This would refresh the User Token (session), and generate a new token for the user to be able to login with. Here are examples of each of the responses below:

200 OK

```
{
  "data": {
    "sessionId": "unique-session-id",
    "userId": 7,
    "token": "new-jwt-token-string",
    "createdAt": "2024-05-06T11:00:00.000Z"
  }
}
```

401 Unauthorized

```
{
  "error": "Invalid refresh token."
}
```

Curl HTTP

```
curl -X PUT "http://localhost:3000/api/sessions"
```

Delete Session

Deletes an existing session (logout). This call to delete a session will either return with success or an error. Examples are below:

200 OK

```
{
  "message": "Session deleted successfully."
}
```

Curl HTTP

```
curl -X DELETE "http://localhost:3000/api/sessions"
```

Templates

2. CodeTemplate Model

Fields:

- **id: Int - Primary key, auto-incremented.**
- **title: String - Title of the code template.**
- **explanation: String - Description of the template's purpose.**
- **code: String - The code snippet.**
- **language: String - Programming language (e.g., JavaScript, Python).**
- **createdAt: DateTime - Creation timestamp, defaults to now.**
- **author: User - Author of the template.**
- **authorId: Int - Foreign key to User.**
- **forkedFrom: CodeTemplate? - Reference to original template if forked.**
- **forkedFromId: Int? - Foreign key to parent CodeTemplate.**
- **forks: CodeTemplate[] - Templates forked from this template.**
- **blogPosts: BlogPost[] - Blog posts linking this template.**
- **tags: Tag[] - Tags associated with the template.**

Relationships:

- **Many-to-One with User.**
- **Self-Referential Many-to-One for forking.**
- **Many-to-Many with BlogPost and Tag.**

Usage:

- **Facilitates code sharing and collaboration.**
- **Supports versioning through forking.**
- **Enhances blog posts with relevant code examples.**
- **Organized via tags for easy discovery.**

Create Template

Creates a new code template. This call to create a tempelate can result in 3 different return codes.

201 Created


```
{
  "data": {
    "id": 10,
    "title": "Neural Network for Image Classification",
    "explanation": "A neural network implemented in Python for classifying images.",
    "code": "import tensorflow as tf\n# Define and train neural network...",
    "language": "Python",
    "createdAt": "2024-05-06T16:00:00.000Z",
    "authorId": 7,
    "forkedFromId": null,
    "forks": [],
    "blogPosts": [],
    "tags": [
      { "id": 3, "name": "machine learning" },
      { "id": 4, "name": "neural network" },
      { "id": 5, "name": "python" },
      { "id": 6, "name": "image classification" }
    ]
  }
}
```


401 Unauthorized


```
{
  "error": "Unauthorized"
}
```


Curl HTTP

```
curl -X POST -d '{
  "title": "Neural Network for Image Classification",
  "explanation": "A neural network implemented in Python for classifying images.",
  "code": "import tensorflow as tf\n# Define and train neural network...",
  "language": "Python",
  "tags": ["machine learning", "neural network", "python", "image classification"]
}

' "http://localhost:3000/api/templates"
```

Search Templates

Searches for code templates based on query parameters.

Query Parameters:

- `title` (String, Optional): Keyword to search in the template title.
- `tags` (String, Optional): Comma-separated list of tags to filter templates.
- `content` (String, Optional): Keyword to search within the code content.
- `page` (Integer, Optional): Page number for pagination (default: 1).
- `limit` (Integer, Optional): Number of templates per page (default: 5).
- `userOnly` (Boolean, Optional): If `true` , filters templates created by the authenticated user.

We can expect either a Sucessful responce here, 400 Bad Request (error in body) or 401 Unauthorized. Examples are below:

200 Ok

```
{
  "data": [
    {
      "id": 12,
      "title": "Quick Sort Implementation",
      "explanation": "An efficient sorting algorithm implemented in JavaScript.",
      "code": "function quickSort(arr) { /* implementation */ }",
      "language": "JavaScript",
      "createdAt": "2024-05-06T17:00:00.000Z",
      "authorId": 7,
      "forkedFromId": null,
      "forks": [],
      "blogPosts": [...],
      "tags": [
        { "id": 2, "name": "sorting" }
      ]
    }
    // ...up to 5 templates
  ],
  "meta": {
    "total": 10,
    "totalPages": 2,
    "page": 1
  }
}
```


401 Unauthorized


```
{
  "error": "Unauthorized"
}
```


Curl HTTP

```
curl -X GET "http://localhost:3000/api/templates?title=Sorting&tags=sorting&content=func
tion&page=1&limit=5&userOnly=true"
```

Retrieve Template by ID

This is simply supposed to get get the tempelated based on the ID (decided on creation time auto-incremented). In the case above, it would return the template with ID 1. We can expect the below responses: (example only)

200 OK


```
{
  "data": {
    "id": 1,
    "title": "Bubble Sort Implementation",
    "explanation": "A simple bubble sort algorithm implemented in Python.",
    "code": "def bubble_sort(arr):\n    n = len(arr)\n    for i in range(n):\n        fo
r j in range(0, n-i-1):\n            if arr[j] > arr[j+1]:\n                arr[j], arr[
j+1] = arr[j+1], arr[j]",
    "language": "Python",
    "createdAt": "2024-05-05T09:00:00.000Z",
    "authorId": 7,
    "forkedFromId": null,
    "forks": [...],
    "blogPosts": [...],
    "tags": [
      { "id": 1, "name": "sorting" }
    ]
  }
}
```


404 Not Found


```
{
  "error": "Template not found"
}
```


Curl HTTP

```
curl -X GET "http://localhost:3000/api/templates/1"
```

Update Template by ID

Updates details of a specific code template by its ID. Expected Responses below:

200 OK


```
{
  "data": {
    "id": 2,
    "title": "Updated Sorting Algorithms",
    "explanation": "This template now includes quicksort and mergesort algorithms.",
    "code": "function quickSort(arr) { /*...*/ }",
    "language": "JavaScript",
    "createdAt": "2024-05-05T10:00:00.000Z",
    "authorId": 7,
    "forkedFromId": null,
    "forks": [...],
    "blogPosts": [...],
    "tags": [
      { "id": 1, "name": "sorting" },
      { "id": 7, "name": "algorithms" },
      { "id": 8, "name": "advanced" }
    ]
  }
}
```


404 Not Found


```
{
  "error": "Template not found"
}
```


Curl HTTP

```
curl -X PUT -d '{
  "title": "Updated Sorting Algorithms",
  "explanation": "This template now includes quicksort and mergesort algorithms.",
  "code": "function quickSort(arr) { /*...*/ }",
  "language": "JavaScript",
  "tags": ["sorting", "algorithms", "advanced"]
}
' "http://localhost:3000/api/templates/1"
```

Delete Template by ID

Deletes a specific code template by its ID. Expected responses below:

200 OK


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


404 Not found


```
{
  "error": "Template not found."
}
```


Curl HTTP

```
curl -X DELETE "http://localhost:3000/api/templates/1"
```

Fork Template by ID

Forks an existing code template, creating a new template based on the original. Expected responses below:

201 Created

```
{
  "data": {
    "id": 13,
    "title": "Bubble Sort Implementation - Forked",
    "explanation": "Forked version with added optimizations.",
    "code": "def bubble_sort(arr):\n    n = len(arr)\n    for i in range(n):\n        already_sorted = True\n        for j in range(n - i - 1):\n            if arr[j] > arr[j + 1]:\n                arr[j], arr[j + 1] = arr[j + 1], arr[j]\n                already_sorted = False\n        if already_sorted:\n            break",
    "language": "Python",
    "createdAt": "2024-05-06T18:00:00.000Z",
    "authorId": 7,
    "forkedFromId": 2,
    "forks": [],
    "blogPosts": [],
    "tags": [
      { "id": 1, "name": "sorting" },
      { "id": 9, "name": "optimized" }
    ]
  }
}
```


404 Not Found


```
{
  "error": "Template not found."
}
```


Curl HTTP

```
curl -X POST "http://localhost:3000/api/templates/2"
```

Posts

3. BlogPost Model

Fields:

- **id: Int - Primary key, auto-incremented.**
- **title: String - Title of the blog post.**
- **description: String - Summary of the post.**
- **content: String - Full content.**
- **hidden: Boolean - Visibility flag, defaults to false.**
- **createdAt: DateTime - Creation timestamp, defaults to now.**
- **author: User - Author of the post.**
- **authorId: Int - Foreign key to User.**
- **templates: CodeTemplate[] - Linked code templates.**
- **comments: Comment[] - Comments on the post.**
- **reports: Report[] - Reports against the post.**
- **tags: Tag[] - Tags associated with the post.**
- **upvotes: Int - Number of upvotes, defaults to 0.**
- **downvotes: Int - Number of downvotes, defaults to 0.**
- **votes: Vote[] - Votes cast on the post.**

Relationships:

- **Many-to-One with User.**
- **Many-to-Many with CodeTemplate and Tag.**
- **One-to-Many with Comment, Report, and Vote.**

Usage:

- **Core content publishing mechanism.**
- **Enables interaction through comments and votes.**

- **Allows content categorization via tags.**
- **Integrates code templates for enriched content.**

Create Blog Post

Creates a new blog post. Expected responses below:

201 Created


```
{
  "data": {
    "id": 3,
    "title": "Mastering Next.js",
    "description": "A comprehensive guide to Next.js.",
    "content": "In this blog post, we delve deep into Next.js features like server-side
rendering, static site generation, and API routes...",
    "hidden": false,
    "createdAt": "2024-05-05T10:00:00.000Z",
    "author": {
      "id": 1,
      "firstName": "Alice",
      "lastName": "Johnson",
      "email": "alice.johnson@example.com"
    },
    "tags": [
      { "id": 5, "name": "Next.js" },
      { "id": 6, "name": "React" },
      { "id": 7, "name": "Advanced" }
    ],
    "upvotes": 0,
    "downvotes": 0,
    "rating": 0
  }
}
```


401 Unauthorized


```
{
  "error": "Unauthorized"
}
```


Curl | HTTP

```
curl -X POST -d '{
  "title": "Introduction to Sorting Algorithms",
  "description": "An overview of popular sorting algorithms with examples.",
  "content": "Sorting is a fundamental concept in computer science. This post explores p
opular algorithms like Bubble Sort, Merge Sort, and Quick Sort.",
  "tags": ["algorithms", "sorting"]
}' "http://localhost:3000/api/posts"
```

Search Blog Posts

Retrieves a specific blog post by parameters. Responses:

200 OK

```
{
  "data": [
    {
      "id": 3,
      "title": "Mastering Next.js",
      "description": "A comprehensive guide to Next.js.",
      "content": "In this blog post, we delve deep into Next.js features like server-sid
e rendering, static site generation, and API routes...",
      "hidden": false,
      "createdAt": "2024-05-05T10:00:00.000Z",
      "author": {
        "id": 1,
        "firstName": "Alice",
        "lastName": "Johnson",
        "email": "alice.johnson@example.com"
      },
      "tags": [
        { "id": 5, "name": "Next.js" },
        { "id": 6, "name": "React" },
        { "id": 7, "name": "Advanced" }
      ],
      "upvotes": 5,
      "downvotes": 1,
      "rating": 4,
      "comments": [...],
      "reports": [...],
      "votes": [...]
    }
    // ... up to 5 blog posts
  ],
  "meta": {
    "total": 1,
    "totalPages": 1,
    "page": 1
  }
}
```


401 Unauthorized


```
{
  "error": "Unauthorized"
}
```


Curl HTTP

```
curl -X GET "http://localhost:3000/api/posts?page=1&limit=5&sort=controversial"
```

Retrieve Blog Post by ID

ID of the blog post to retrieve. Exptected Responses:

200 OK

```
{
  "data": {
    "id": 3,
    "title": "Mastering Next.js",
    "description": "A comprehensive guide to Next.js.",
    "content": "In this blog post, we delve deep into Next.js features like server-side
rendering, static site generation, and API routes...",
    "hidden": false,
    "createdAt": "2024-05-05T10:00:00.000Z",
    "author": {
      "id": 1,
      "firstName": "Alice",
      "lastName": "Johnson",
      "email": "alice.johnson@example.com"
    },
    "tags": [
      { "id": 5, "name": "Next.js" },
      { "id": 6, "name": "React" },
      { "id": 7, "name": "Advanced" }
    ],
    "upvotes": 5,
    "downvotes": 1,
    "rating": 4,
    "comments": [...],
    "reports": [...],
    "votes": [...]
  }
}
```


404 Not Found


```
{
  "error": "Template not found."
}
```


Curl HTTP

```
curl -X GET "http://localhost:3000/api/posts/1"
```

Update Blog Post by ID

Updates details of a specific blog post by its ID. Expected Responses below:

200 OK - Successfully Updated Post


```
{
  "id": 1,
  "title": "Updated Introduction to Sorting Algorithms",
  "description": "An updated overview of sorting algorithms with examples.",
  "content": "Sorting is essential in computer science. This updated post explores algor
ithms like Bubble Sort, Merge Sort, and Quick Sort in more depth.",
  "tags": [
    { "id": 5, "name": "algorithms" },
    { "id": 6, "name": "sorting" },
    { "id": 7, "name": "updated" }
  ],
  "templates": [
    { "id": 2, "title": "Merge Sort Template" },
    { "id": 3, "title": "Quick Sort Template" }
  ],
  "upvotes": 3,
  "downvotes": 2,
  "author": {
    "id": 123,
    "firstName": "Jane",
    "lastName": "Doe"
  },
  "createdAt": "2024-01-01T12:00:00Z",
  "updatedAt": "2024-01-02T15:00:00Z"
}
```


401 Unauthorized

```
{
  "error": "Unauthorized: Valid authentication token required."
}
```

403 Forbidden

```
{
  "error": "Forbidden: You do not have permission to update this post."
}
```

404 Not Found

```
{
  "error": "Blog post not found."
}
```

Curl HTTP

```
curl -X PUT -d '{
  "title": "Updated Introduction to Sorting Algorithms",
  "description": "An updated overview of sorting algorithms with examples.",
  "content": "Sorting is essential in computer science. This updated post explores algor
ithms like Bubble Sort, Merge Sort, and Quick Sort in more depth.",
  "tags": ["algorithms", "sorting", "updated"],
  "upvotes": 3,
  "downvotes": 2
}' "http://localhost:3000/api/posts/1"
```

Delete Blog Post by ID

Deletes a specific blog post by its ID. Expected Responses

200 OK

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

404 Not Found

```
{
  "error": "Template not found."
}
```

Curl HTTP

```
curl -X DELETE "http://localhost:3000/api/posts/2"
```

Hide Post by ID

Curl HTTP

```
curl -X PATCH "http://localhost:3000/api/posts/2"
```

Ratings

7. Rating Model

Fields:

- **id: Int - Primary key, auto-incremented.**
- **type: String - Vote type: “upvote” or “downvote”.**
- **user: User - User who cast the vote.**
- **userId: Int - Foreign key to User.**
- **blogPost: BlogPost? - Voted blog post (Optional).**

- **blogPostId: Int? - Foreign key to BlogPost (Optional).**
- **comment: Comment? - Voted comment (Optional).**
- **commentId: Int? - Foreign key to Comment (Optional).**
- **createdAt: DateTime - Vote timestamp, defaults to now.**

Unique Constraints:

- **@@unique([userId, blogPostId]) - Ensures a user can vote only once per blog post.**
- **@@unique([userId, commentId]) - Ensures a user can vote only once per comment.**

Relationships:

- **Many-to-One with User.**
- **Many-to-One with BlogPost and Comment.**

Usage:

- **Allows users to express approval or disapproval of content.**
- **Influences content ranking and visibility based on community feedback.**
- **Prevents vote manipulation through unique constraints.**

Upvote or Downvote Post by ID

Allows users to upvote or downvote a blog post. Expected Response below:

200 OK:

```
{
  "data": {
    "id": 1,
    "type": "downvote",
    "userId": 2,
    "blogPostId": 2,
    "commentId": null,
    "createdAt": "2024-05-06T14:00:00.000Z"
  }
}
```

400 Bad Request

```
{
  "error": "Provide either blogPostId or commentId, but not both."
}
```

401 Unauthorized

```
{
  "error": "Unauthorized"
}
```

404 Not Found

```
{
  "error": "User not found."
}
```

Curl HTTP

```
curl -X POST -d '{
    "action": "upvote"
}' "http://localhost:3000/api/ratings/posts/3"
```

Upvote or Downvote Comment by ID

Allows users to upvote or downvote a comment. Expected below

200 OK

```
{
  "data": {
    "id": 2,
    "type": "downvote",
    "userId": 1,
    "blogPostId": null,
    "commentId": 1,
    "createdAt": "2024-05-06T15:00:00.000Z"
  }
}
```


400 Bad Request


```
{
  "error": "Provide either blogPostId or commentId, but not both."
}
```


401 Unauthorized


```
{
  "error": "Unauthorized"
}
```


404 Not Found


```
{
  "error": "Comment/User not found"
}
```


Curl HTTP

```
curl -X POST -d '{
    "action": "downvote"
}' "http://localhost:3000/api/ratings/comments/1"
```

Reports

6. Report Model

Fields:

- **id: Int - Primary key, auto-incremented.**
- **reason: String - Reason for reporting the content.**
- **createdAt: DateTime - Creation timestamp, defaults to now.**
- **reporter: User - User who submitted the report.**
- **reporterId: Int - Foreign key to User.**
- **blogPost: BlogPost? - Reported blog post (Optional).**
- **blogPostId: Int? - Foreign key to BlogPost (Optional).**
- **comment: Comment? - Reported comment (Optional).**
- **commentId: Int? - Foreign key to Comment (Optional).**

Relationships:

- **Many-to-One with User (Reporter).**
- **Many-to-One with BlogPost and Comment.**

Usage:

- **Enables users to report inappropriate or violating content.**
- **Supports platform moderation and content quality control.**

Retrieve & Sort Reports

Curl HTTP

```
curl -X GET "http://localhost:3000/api/reports?include=posts"
```

Create Report

Allows a user to report a blog post or comment by specifying the target type and ID. The reason field provides context for the report. Expected Responses:

201 Created


```
{
  "id": 10,
  "reason": "This post contains even more inappropriate language.",
  "userId": 123,
  "commentId": 4,
  "createdAt": "2024-01-01T12:00:00Z"
}
```


400 Bad Request


```
{
  "error": "Invalid data: 'targetType' and 'targetId' are required."
}
```


401 Unauthorized


```
{
  "error": "Unauthorized: Valid authentication token required."
}
```


Curl HTTP

```
curl -X POST -d '{
  "targetType": "post",
  "targetId": 3,
  "reason": "This post contains even more inappropriate language."
}' "http://localhost:3000/api/reports"
```

Comments

Fields:

- **id: Int - Primary key, auto-incremented.**
- **content: String - Comment text.**
- **createdAt: DateTime - Creation timestamp, defaults to now.**
- **author: User - Author of the comment.**
- **authorId: Int - Foreign key to User.**
- **blogPost: BlogPost - Blog post the comment is associated with.**
- **blogPostId: Int - Foreign key to BlogPost.**
- **parent: Comment? - Parent comment if it's a reply.**
- **parentId: Int? - Foreign key to parent Comment.**
- **replies: Comment[] - Replies to the comment.**
- **reports: Report[] - Reports against the comment.**
- **upvotes: Int - Number of upvotes, defaults to 0.**
- **downvotes: Int - Number of downvotes, defaults to 0.**
- **votes: Vote[] - Votes cast on the comment.**

Relationships:

- **Many-to-One with User and BlogPost.**
- **Self-Referential One-to-Many for nested replies.**
- **One-to-Many with Report and Vote.**

Usage:

- **Facilitates user discussions and feedback on blog posts.**
- **Supports threaded conversations through replies.**

- **Allows community moderation via reports and votes.**

Hide Comment by ID

Allows an admin to hide a comment by its ID. Expected Responses:

200 OK - Content Hidden

```
{
  "id": 4,
  "content": "This is the original comment content.",
  "isHidden": true,
  "authorId": 123,
  "blogPostId": 1,
  "upvotes": 5,
  "downvotes": 2,
  "createdAt": "2024-01-01T10:00:00Z",
  "updatedAt": "2024-01-01T12:00:00Z"
}
```

401 Unauthorized

```
{
  "error": "Unauthorized: Valid authentication token required."
}
```

403 Forbidden

```
{
  "error": "Forbidden: Admin access required to hide comments."
}
```

404 Not Found

```
{
  "error": "Comment not found."
}
```

Curl HTTP

```
curl -X PATCH "http://localhost:3000/api/comments/4"
```

Retrieve Comment by ID

Curl HTTP

```
curl -X GET "http://localhost:3000/api/comments/1"
```

Runners

This folder manages the server-side code execution.

Create Runner

This endpoint allows users to submit code in various programming languages, which is then executed on the server. The user can provide code, specify the language, and include any input required for execution via standard input (stdin). The API returns the output of the code execution, execution time, and any errors if the code fails to compile or run.

200 OK - Execution Successful

```
{
    "output": "Hello, World!\n",
    "timeTaken": 55,
    "success": true
}
```

Curl HTTP

```
curl -X POST -d '{
  "language": "JavaScript",
  "code": "console.log('Hello, World!');",
  "stdin": ""
}
' "http://localhost:3000/api/runners"
```