User Registration Endpoint

API Route

POST /api/users/register

HTTP Method

POST

Description

This endpoint registers a new user by creating a user record in the database. It accepts a JSON payload with the user's first name, last name, email, and password. The password is securely hashed before being stored in the database. If a user with the provided email already exists, the endpoint will return an error.

Response Codes

- **201 Created**: User registered successfully.
- **400 Bad Request**: User with the provided email already exists.
- **500 Internal Server Error**: An error occurred while creating the user.
- **405 Method Not Allowed**: HTTP method not supported.

Request Body

```
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "password": "securepassword123"
```

Example Call

```
curl -X POST https://yourdomain.com/api/register \
-H "Content-Type: application/json" \
-d '{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
```

  "password": "securepassword123"

}'

Example Response

Success Response (201 Created)

  "message": "User registered successfully",

  "user": {

    "id": "12345",

    "firstName": "John",

    "lastName": "Doe",

    "email": "john.doe@example.com"

  }

Error Response (400 Bad Request)

  "error": "User already exists"

Error Response (500 Internal Server Error)

  "error": "Error creating user"

Error Response (405 Method Not Allowed)

  "error": "Method not allowed"

User Login Endpoint

API Route

POST /api/users/login

HTTP Method

POST

Description

This endpoint allows users to log in by providing their email and password. It verifies the provided password against the stored hashed password and, if valid, returns an access token and a refresh token for authentication. If the credentials are incorrect or the user is not found, an appropriate error message is returned.

Response Codes

- **200 OK**: Login successful, tokens and user information returned.
- **401 Unauthorized**: Invalid credentials.
- **404 Not Found**: User with the provided email does not exist.
- **500 Internal Server Error**: An error occurred during login.
- **405 Method Not Allowed**: HTTP method not supported.

Request Body

```
  "email": "john.doe@example.com",
  "password": "securepassword123"
```

Example Call

```
curl -X POST https://yourdomain.com/api/login \
-H "Content-Type: application/json" \
-d '{
  "email": "john.doe@example.com",
  "password": "securepassword123"
}'
```

Example Response

Success Response (200 OK)

```
  "message": "Login successful",
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "12345",
    "firstName": "John",
    "lastName": "Doe",
    "email": "john.doe@example.com"
  }
```

Error Response (401 Unauthorized)

```
 "error": "Invalid credentials"
```

Error Response (404 Not Found)

```
 "error": "User not found"
```

Error Response (500 Internal Server Error)

```
 "error": "Error logging in"
```

Error Response (405 Method Not Allowed)

```
 "error": "Method not allowed"
```

Refresh Access Token Endpoint

API Route
POST /api/users/refresh

HTTP Method
POST

Description
This endpoint refreshes the user's access token using a valid refresh token. The refresh token is verified, and if valid, a new access token is returned. This endpoint helps maintain session continuity without requiring the user to log in again.

Response Codes

- **200 OK**: New access token generated successfully.
- **401 Unauthorized**: Invalid or expired refresh token.
- **500 Internal Server Error**: An error occurred while refreshing the token.

- **405 Method Not Allowed**: HTTP method not supported.

Request Body

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

Example Response

Success Response (200 OK)

"accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

Error Response (401 Unauthorized)

"error": "Invalid refresh token"

Error Response (500 Internal Server Error)

"error": "Error refreshing token"

Error Response (405 Method Not Allowed)

"error": "Method not allowed"

User Profile Endpoint

API Route
GET /api/users/profile
PUT /api/users/profile
PATCH /api/users/profile

HTTP Methods
GET (Fetch user profile)
PUT or PATCH (Update user profile)

Description
This endpoint allows authenticated users to retrieve or update their profile information. The GET method retrieves the user's profile, while PUT or PATCH updates the user's profile details such as first name, last name, email, avatar, and phone number.

Response Codes

- **200 OK**: Profile fetched or updated successfully.

- **404 Not Found**: User profile not found.
- **500 Internal Server Error**: An error occurred while fetching or updating the profile.
- **405 Method Not Allowed**: HTTP method not supported.

Request Body for PUT/PATCH (Profile Update)

```
  "firstName": "Jane",
  "lastName": "Doe",
  "email": "jane.doe@example.com",
  "avatar": "https://example.com/avatar.jpg",
  "phoneNumber": "+1234567890"
```

Example Response

Success Response (200 OK) - GET

```
  "id": "12345",
  "firstName": "Jane",
  "lastName": "Doe",
  "email": "jane.doe@example.com",
  "avatar": "https://example.com/avatar.jpg",
  "phoneNumber": "+1234567890",
  "role": "user",
  "createTime": "2023-01-15T10:30:00Z",
  "lastLogin": "2023-04-10T14:45:00Z"
```

Success Response (200 OK) - PUT/PATCH

```
  "message": "Profile updated successfully",
  "user": {
    "id": "12345",
    "firstName": "Jane",
    "lastName": "Doe",
    "email": "jane.doe@example.com",

    "avatar": "https://example.com/avatar.jpg",

    "phoneNumber": "+1234567890"
```

Error Response (404 Not Found)

```
 "error": "User profile not found"
```

Error Response (500 Internal Server Error)

```
 "error": "Failed to fetch user profile"
```

Error Response (405 Method Not Allowed)

```
 "error": "Method not allowed"
```

Template Management Endpoint

API Route
POST /api/templates
GET /api/templates

HTTP Methods
POST (Create a new template)
GET (Fetch templates with optional search and pagination)

Description
This endpoint allows users to create and fetch templates. The POST method, which is protected by authentication, creates a new template associated with the logged-in user. The GET method retrieves templates, with optional filters like title, tags, content, and userId, as well as pagination options.

Response Codes

- **201 Created**: Template created successfully.
- **200 OK**: Templates fetched successfully.
- **400 Bad Request**: Required fields are missing.
- **500 Internal Server Error**: An error occurred during template creation or retrieval.

- **405 Method Not Allowed**: HTTP method not supported.

POST /api/templates (Create Template)

Request Body

"title": "My Template",

"explaination": "This is a sample template",

"content": "Template content here...",

"tags": "tag1,tag2,tag3"

Example Response
Success Response (201 Created)

"message": "Template created",

"templateId": "12345"

Error Response (400 Bad Request)

"error": "Title, explaination, content and tags are required"

Error Response (500 Internal Server Error)

"error": "Failed to create template"

GET /api/templates (Fetch Templates)

Query Parameters

- title: Filter by title (partial match).
- tags: Filter by comma-separated list of tags.
- content: Filter by content (partial match).
- userId: Filter by user ID.
- page: Page number for pagination (defaults to 1).
- itemPerPage: Number of items per page (defaults to 10).

Example Request

GET /api/templates?title=example&tags=tag1,tag2&page=2&itemPerPage=5

Example Response

Success Response (200 OK)

```
[
 {
   "id": "12345",
   "title": "Example Template",
   "explaination": "This is an example template",
   "fileContent": "Template content here...",
   "tags": [
    { "name": "tag1" },
    { "name": "tag2" }
   ],
   "userId": "54321"
 },
 {
   "id": "67890",
   "title": "Another Template",
   "explaination": "Another example",
   "fileContent": "Another content...",
   "tags": [
    { "name": "tag2" },
    { "name": "tag3" }
   ],
   "userId": "54321"
 }
```

]

Error Response (500 Internal Server Error)

  "error": "Failed to fetch templates"

Error Response (405 Method Not Allowed)

  "error": "Method not allowed"

Template Details Endpoint

API Route
GET /api/templates/:id
PUT /api/templates/:id
DELETE /api/templates/:id

HTTP Methods
GET (Fetch a template by ID)
PUT (Update a template by ID)
DELETE (Delete a template by ID)

Description
This endpoint allows users to fetch, update, or delete a specific template by its ID. The GET method retrieves the template's details, while PUT updates the template (authenticated users only), and DELETE removes the template (authenticated users only). Only the template owner can update or delete the template.

Response Codes

- **200 OK**: Template fetched, updated, or deleted successfully.
- **400 Bad Request**: Template ID is missing or invalid.
- **403 Forbidden**: Unauthorized to modify or delete the template (not the owner).
- **404 Not Found**: Template not found.
- **500 Internal Server Error**: An error occurred during the operation.
- **405 Method Not Allowed**: HTTP method not supported.

GET /api/templates/

(Fetch Template)

Example Response

Success Response (200 OK)

```
  "id": "12345",
  "title": "Example Template",
  "explaination": "This is an example template",
  "fileContent": "Template content here...",
  "tags": [
   { "name": "tag1" },
   { "name": "tag2" }
  ],
  "userId": "54321",
  "lastEditTime": "2023-04-10T14:45:00Z"
```

Error Response (400 Bad Request)

```
  "error": "Template ID is required"
```

Error Response (404 Not Found)

```
  "error": "Template not found"
```

PUT /api/templates/

(Update Template)

Request Body

```
  "title": "Updated Template",
  "explaination": "Updated explanation",
  "content": "Updated content here...",
  "tags": "updatedTag1,updatedTag2"
```

Example Response

Success Response (200 OK)

  "message": "Template updated"

Error Response (400 Bad Request)

  "error": "Template ID is required"

Error Response (403 Forbidden)

  "error": "Unauthorized to update this template (You are not the owner.)"

Error Response (404 Not Found)

  "error": "Template id not found"

Error Response (500 Internal Server Error)

  "error": "Failed to update template"

DELETE /api/templates/

(Delete Template)

Example Response

Success Response (200 OK)

  "message": "Template deleted"

Error Response (400 Bad Request)

  "error": "Template ID is required"

Error Response (403 Forbidden)

  "error": "Unauthorized to delete this template (You are not the owner.)"

Error Response (404 Not Found)

  "error": "Template id not found"

Error Response (500 Internal Server Error)

  "error": "Failed to delete template"

Template Forking Endpoint

API Route
POST /api/templates/fork

HTTP Method
POST

Description
This endpoint allows authenticated users to fork an existing template by creating a new template based on the original one. The new template will include the same title, explanation, content, and tags as the original, and will be associated with the user who created the fork. The forked template will also indicate the ID of the original template it was forked from.

Response Codes

- **201 Created**: Template successfully forked.
- **400 Bad Request**: Template ID is missing.
- **404 Not Found**: Template with the provided ID not found.
- **500 Internal Server Error**: Template found, but an error occurred while forking.
- **405 Method Not Allowed**: HTTP method not supported.

POST /api/templates/fork (Fork Template)

Request Body

"templateId": "12345"

Example Response

Success Response (201 Created)

"message": "Template forked",

"templateId": "67890"

Error Response (400 Bad Request)

"error": "Template ID is required"

Error Response (404 Not Found)

"error": "Failed to find template"

Error Response (500 Internal Server Error)

"error": "Template found, but fork failed"

Code Execution Endpoint

API Route
POST /api/execute

HTTP Method
POST

Description
This endpoint allows users to execute code in specified programming languages (JavaScript, Python, Java, C, C++). It accepts the code, language, and optional input (stdin) and returns the output of the code execution. The endpoint will validate the input to ensure required fields are provided and the language is supported.

Response Codes

- **200 OK**: Code executed successfully, output returned.
- **400 Bad Request**: Required fields are missing, or an unsupported language was specified.
- **500 Internal Server Error**: Code execution failed.
- **405 Method Not Allowed**: HTTP method not supported.

POST /api/execute (Execute Code)

Request Body

"code": "print('Hello, World!')",

"language": "python",

  "stdin": ""

Example Response

Success Response (200 OK)

  "output": "Hello, World!\n"

Error Response (400 Bad Request - Missing Code)

  "error": "Code is required"

Error Response (400 Bad Request - Missing Language)

  "error": "Language is required"

Error Response (400 Bad Request - Unsupported Language)

  "error": "Language not supported"

Error Response (500 Internal Server Error)

  "error": "Failed to execute code"

Blog Post Management Endpoint

API Route
GET /api/blog-post
POST /api/blog-post

HTTP Methods
GET (Fetch paginated list of blog posts)
POST (Create a new blog post - requires authentication)

Description
This endpoint allows users to retrieve a paginated list of non-hidden blog posts (GET) and create a new blog post (POST). The GET method is public and includes pagination metadata, while the POST method is protected and allows authenticated users to create blog posts with optional tags.

Response Codes

- **200 OK**: Blog posts fetched or new post created successfully.

- **400 Bad Request**: Invalid data (e.g., tags not an array of strings).
- **500 Internal Server Error**: An error occurred during the operation.
- **405 Method Not Allowed**: HTTP method not supported.

GET /api/blog-post (Fetch Blog Posts)

Query Parameters

- page: Page number for pagination (defaults to 1).
- limit: Number of posts per page (defaults to 10).

Example Response

Success Response (200 OK)

```
 "posts": [
  {
   "id": "1",
   "title": "Sample Blog Post",
   "description": "Description of the post",
   "content": "Content of the post...",
   "createdAt": "2023-01-15T10:30:00Z",
   "user": {
    "firstName": "John",
    "lastName": "Doe"
   },
   "tags": [
    { "name": "tag1" },
    { "name": "tag2" }
   ],
   "comments": [
      {
        "content": "Great post!",
        "user": {
          "firstName": "Jane",
          "lastName": "Smith"
        }
      }
    ]
  }
 ],
 "pagination": {
  "totalItems": 50,
  "totalPages": 5,
  "currentPage": 1,
  "pageSize": 10
 }
```

Error Response (500 Internal Server Error)

```
 "error": "Failed to fetch blog posts"
```

POST /api/blog-post (Create Blog Post)

Request Body

```
 "title": "New Blog Post",
 "description": "Description of the new post",
 "content": "Content of the new post...",
 "tags": ["tag1", "tag2"]
```

Example Response

Success Response (201 Created)

```
 "id": "2",

 "title": "New Blog Post",

 "description": "Description of the new post",

 "content": "Content of the new post...",

 "userId": "54321",

 "tags": [

  { "name": "tag1" },

  { "name": "tag2" }

 ]
```

Error Response (400 Bad Request - Invalid Tags)

```
 "error": "Tags should be an array of strings"
```

Error Response (500 Internal Server Error)

```
 "error": "Failed to create blog post"
```

Single Blog Post Management Endpoint

API Route
GET /api/blog-post/:id
PUT /api/blog-post/:id
DELETE /api/blog-post/:id

HTTP Methods
GET (Fetch a single blog post)
PUT (Update a blog post - requires authentication)
DELETE (Delete a blog post - requires authentication)

Description
This endpoint allows users to retrieve, update, or delete a single blog post by its ID. The GET method fetches a blog post's details, including tags, author information, and non-hidden

comments. The PUT and DELETE methods are protected, allowing authenticated users to update or delete their own blog posts.

Response Codes

- **200 OK**: Blog post fetched or updated successfully.
- **204 No Content**: Blog post deleted successfully.
- **400 Bad Request**: Missing required fields.
- **403 Forbidden**: Unauthorized to modify or delete the post.
- **404 Not Found**: Blog post not found.
- **500 Internal Server Error**: An error occurred during the operation.
- **405 Method Not Allowed**: HTTP method not supported.

GET /api/blog-post/

(Fetch Single Blog Post)

Example Response

Success Response (200 OK)

```
 "id": "1",
 "title": "Sample Blog Post",
 "description": "Description of the post",
 "content": "Content of the post...",
 "user": {
  "firstName": "John",
  "lastName": "Doe"
 },
 "tags": [
  { "name": "tag1" },
  { "name": "tag2" }
 ],
 "comments": [
  {
    "content": "Great post!",
    "user": {
      "firstName": "Jane",
      "lastName": "Smith"
    }
  }
 ]
```

Error Response (404 Not Found)

```
 "error": "Blog post not found"
```

PUT /api/blog-post/

(Update Blog Post)

Request Body

```
 "title": "Updated Blog Post",
 "description": "Updated description",
 "content": "Updated content...",
 "tags": ["updatedTag1", "updatedTag2"]
```

Example Response

Success Response (200 OK)

```
 "id": "1",
 "title": "Updated Blog Post",
 "description": "Updated description",
 "content": "Updated content...",
```

"tags": [

  { "name": "updatedTag1" },

  { "name": "updatedTag2" }

]

Error Response (403 Forbidden)

"error": "Unauthorized to update this post"

Error Response (404 Not Found)

"error": "Blog post not found"

Error Response (500 Internal Server Error)

"error": "Failed to update blog post"

DELETE /api/blog-post/

(Delete Blog Post)

Example Response

Success Response (204 No Content)

Error Response (403 Forbidden)

"error": "Unauthorized to delete this post"

Error Response (404 Not Found)

"error": "Blog post not found"

Error Response (500 Internal Server Error)

"error": "Failed to delete blog post"

Blog Post Search Endpoint

API Route
GET /api/blog-post/search

HTTP Method
GET

Description
This endpoint allows users to search for blog posts based on various criteria such as title, content, tags, and templates. The results can be sorted by specified fields (default is rating in descending order) and include pagination metadata. The response contains post details, tags, associated user information, templates, and non-hidden comments sorted by rating.

Response Codes

- **200 OK**: Blog posts retrieved successfully.
- **500 Internal Server Error**: An error occurred while searching for blog posts.
- **405 Method Not Allowed**: HTTP method not supported.

Query Parameters

- title (string, optional): Filter posts by title (partial match).
- content (string, optional): Filter posts by content (partial match).
- tag (string, optional): Filter posts by tag name.
- template (string, optional): Filter posts by template title.
- sortBy (string, optional): Field to sort by (default is rating).
- order (string, optional): Sort order (asc or desc, default is desc).
- page (number, optional): Page number for pagination (default is 1).
- limit (number, optional): Number of posts per page (default is 10).

Example Request
GET /api/blog-post/search?title=example&tag=technology&page=2&limit=5&sortBy=createdAt&order=asc

Example Response

Success Response (200 OK)

```
  "posts": [
{
 "id": "1",
 "title": "Sample Blog Post",
 "content": "Content of the post...",
 "user": {
  "id": "123",
  "firstName": "John",
  "lastName": "Doe"
 },
 "tags": [
  { "name": "technology" },
  { "name": "programming" }
 ],
 "templates": [
  {
   "id": "456",
   "title": "Sample Template",
   "fileContent": "Template content..."
  }
 ],
 "comments": [
  {
   "content": "Interesting post!",
   "user": {
    "firstName": "Jane",
    "lastName": "Smith"
```

```
      },
      "rating": 5
     }
    ]
   }
  ],
  "pagination": {
   "totalItems": 25,
   "totalPages": 5,
   "currentPage": 2,
   "pageSize": 5
  }
```

Error Response (500 Internal Server Error)

```
  "error": "Failed to search blog posts"
```

Link Templates to Blog Post Endpoint

API Route
POST /api/blog-post/:id/link-templates

HTTP Method
POST

Description
This endpoint allows authenticated users to link multiple templates to a specific blog post. The blog post must belong to the authenticated user. The templateIds array in the request body specifies the IDs of the templates to be linked.

Response Codes

- **200 OK**: Templates linked to the blog post successfully.

- **400 Bad Request**: templateIds array is missing or empty.
- **403 Forbidden**: Unauthorized to modify the blog post.
- **404 Not Found**: Blog post not found.
- **500 Internal Server Error**: An error occurred while linking templates.
- **405 Method Not Allowed**: HTTP method not supported.

POST /api/blog-post/:id/link-templates (Link Templates to Blog Post)

Request Body

```
 "templateIds": ["1", "2", "3"]
```

Example Response

Success Response (200 OK)

```
 "message": "Templates linked to blog post successfully",
 "blogPost": {
  "id": "1",
  "templates": [
   { "id": "1", "title": "Template One" },
   { "id": "2", "title": "Template Two" },
   { "id": "3", "title": "Template Three" }
  ]
 }
```

Error Response (400 Bad Request)

```
 "error": "templateIds array is required and should not be empty"
```

Error Response (403 Forbidden)

```
 "error": "Unauthorized to modify this blog post"
```

Error Response (404 Not Found)

```
 "error": "Blog post not found"
```

Error Response (500 Internal Server Error)

"error": "Failed to link templates to blog post"

Blog Post Comments Endpoint

API Route
GET /api/blog-comment
POST /api/blog-comment

HTTP Methods
GET (Fetch paginated comments for a blog post)
POST (Add a new comment to a blog post - requires authentication)

Description
This endpoint allows users to retrieve paginated comments for a specific blog post (GET) and to add a new comment (POST). Only non-hidden comments are included in the response. The POST method is protected, allowing authenticated users to add comments to a blog post.

Response Codes

- **200 OK**: Comments fetched or new comment created successfully.
- **201 Created**: Comment created successfully.
- **400 Bad Request**: Missing postId or invalid data.
- **500 Internal Server Error**: An error occurred during the operation.
- **405 Method Not Allowed**: HTTP method not supported.

GET /api/blog-comment (Fetch Comments)

Query Parameters

- page: Page number for pagination (defaults to 1).
- limit: Number of comments per page (defaults to 10).

Example Request
GET /api/blog-comment?page=2&limit=5

Example Response

Success Response (200 OK)

```
 "comments": [
  {
   "id": "123",
   "content": "This is a great post!",
   "createdAt": "2023-04-10T14:45:00Z",
   "user": {
    "firstName": "Jane",
    "lastName": "Smith"
   }
  }
 ],
 "pagination": {
  "totalItems": 20,
  "totalPages": 4,
  "currentPage": 2,
  "pageSize": 5
 }
```

Error Response (400 Bad Request)

```
 "error": "postId is required"
```

Error Response (500 Internal Server Error)

```
 "error": "Failed to fetch comments"
```

POST /api/blog-comment (Add Comment)

Request Body

"content": "This is a new comment",

"postId": "1"

Example Response

Success Response (201 Created)

"id": "124",

"content": "This is a new comment",

"postId": "1",

"userId": "456",

"createdAt": "2023-04-10T14:50:00Z"

Error Response (500 Internal Server Error)

"error": "Failed to create comment"

Comment Management Endpoint

API Route
PUT /api/blog-comment/:id
DELETE /api/blog-comment/:id

HTTP Methods
PUT (Update a comment - requires authentication)
DELETE (Delete a comment - requires authentication)

Description
This endpoint allows authenticated users to update or delete their own comments on blog posts. Users can modify the content of their comments (PUT) or delete a comment (DELETE) if they are the comment's owner.

Response Codes

- **200 OK**: Comment updated successfully.
- **204 No Content**: Comment deleted successfully.
- **403 Forbidden**: Unauthorized to modify or delete the comment.

- **404 Not Found**: Comment not found.
- **500 Internal Server Error**: An error occurred during the operation.
- **405 Method Not Allowed**: HTTP method not supported.

PUT /api/comments/:id

(Update Comment)

Request Body

  "content": "Updated comment content"

Example Response

Success Response (200 OK)

  "id": "123",

  "content": "Updated comment content",

  "userId": "456",

  "postId": "1",

  "createdAt": "2023-04-10T14:45:00Z"

Error Response (403 Forbidden)

  "error": "Unauthorized to update this comment"

Error Response (404 Not Found)

  "error": "Comment not found"

Error Response (500 Internal Server Error)

  "error": "Failed to update comment"

DELETE /api/blog-comment/:id

(Delete Comment)

Example Response

Success Response (204 No Content)

Error Response (403 Forbidden)

"error": "Unauthorized to delete this comment"

Error Response (404 Not Found)

"error": "Comment not found"

Error Response (500 Internal Server Error)

"error": "Failed to delete comment"

Vote Status Endpoint

API Route
GET /api/votes/:id

HTTP Method
GET

Description
This endpoint allows authenticated users to check if they have voted on a specific blog post or comment. Users must specify the type as either "post" or "comment" in the query parameters.

Response Codes

- **200 OK**: Vote status retrieved successfully.
- **400 Bad Request**: Missing or invalid type parameter.
- **500 Internal Server Error**: An error occurred while retrieving the vote.
- **405 Method Not Allowed**: HTTP method not supported.

Query Parameters

- id (required): The ID of the blog post or comment to check for a vote.
- type (required): Specify as "post" or "comment" to indicate the target of the vote check.

Example Request
GET /api/votes/1?type=post

Example Response

Success Response (200 OK - No Vote)

"hasVoted": false,

"voteType": null

Success Response (200 OK - Vote Found)

"hasVoted": true,

"voteType": "upvote"

Error Response (400 Bad Request)

"error": "Type must be specified as 'post' or 'comment'"

Error Response (500 Internal Server Error)

"error": "Failed to retrieve vote"

Voting Endpoint

API Route
POST /api/votes

HTTP Method
POST

Description
This endpoint allows authenticated users to cast an upvote or downvote on a specific blog post or comment. Users can only vote once per post or comment. If they attempt to vote on the same post or comment again, they will receive an error message.

Response Codes

- **200 OK**: Vote registered successfully.
- **400 Bad Request**: Missing postId or commentId, invalid vote type, or duplicate vote.
- **500 Internal Server Error**: An error occurred while processing the vote.

- **405 Method Not Allowed**: HTTP method not supported.

Request Body

- postId (optional): The ID of the blog post to vote on.
- commentId (optional): The ID of the comment to vote on.
- type (required): The type of vote, either "upvote" or "downvote".

Example Request

"postId": "1",

"type": "upvote"

Example Response

Success Response (200 OK)

"message": "Vote registered successfully"

Error Response (400 Bad Request - Missing ID)

"error": "postId or commentId is required"

Error Response (400 Bad Request - Invalid Vote Type)

"error": "Invalid vote type"

Error Response (400 Bad Request - Duplicate Vote)

"error": "You have already voted on this post"

Error Response (500 Internal Server Error)

"error": "Failed to process vote"

Content Moderation Endpoint

API Route
GET /api/moderation/reports
PATCH /api/moderation/reports

HTTP Methods
GET (Fetch reported posts and comments - admin only)
PATCH (Update visibility of a post or comment - admin only)

Description
This endpoint allows admin users to review reported blog posts and comments and update their visibility status. The GET method retrieves all reported posts and comments, ordered by report count. The PATCH method allows admins to hide or unhide specific posts or comments based on provided IDs.

Response Codes

- **200 OK**: Reports fetched or content visibility updated successfully.
- **400 Bad Request**: Missing postId or commentId for update.
- **403 Forbidden**: Unauthorized access (non-admin user).
- **500 Internal Server Error**: An error occurred during the operation.
- **405 Method Not Allowed**: HTTP method not supported.

GET /api/admin/reports (Fetch Reported Content)

Example Response

Success Response (200 OK)

```
 "posts": [
  {
   "id": "1",
   "title": "Reported Post Title",
   "user": {
    "id": "123",
    "firstName": "John",
    "lastName": "Doe"
   },
   "reports": [
      { "id": "1", "reason": "Inappropriate content" },
      { "id": "2", "reason": "Spam" }
    ]
   }
 ],
 "comments": [
  {
   "id": "2",
   "content": "Reported comment content",
   "user": {
    "id": "456",
    "firstName": "Jane",
    "lastName": "Smith"
   },
   "reports": [
    { "id": "3", "reason": "Harassment" }
   ]
  }
 ]
```

Error Response (403 Forbidden)

```
 "error": "Unauthorized access"
```

Error Response (500 Internal Server Error)

```
 "error": "Failed to fetch reports"
```

PATCH /api/admin/reports (Update Content Visibility)

Request Body

  "postId": "1",

  "hidden": true

or

  "commentId": "2",

  "hidden": true

Example Response

Success Response (200 OK)

  "id": "1",

  "title": "Updated Post Title",

  "hidden": true

Error Response (400 Bad Request - Missing ID)

  "error": "Specify either postId or commentId to update"

Error Response (500 Internal Server Error)

  "error": "Failed to update content visibility"

Report Content Endpoint

API Route
POST /api/report

HTTP Method
POST

Description
This endpoint allows authenticated users to report a blog post or comment. The user must specify a reason for the report. Either postId or commentId should be provided to identify the content being reported.

Response Codes

- **201 Created**: Report submitted successfully.

- **400 Bad Request**: Missing reason for report.
- **500 Internal Server Error**: An error occurred while creating the report.
- **405 Method Not Allowed**: HTTP method not supported.

Request Body

- postId (optional): The ID of the blog post to report.
- commentId (optional): The ID of the comment to report.
- reason (required): The reason for reporting the content.

Example Request

```
  "postId": "1",
  "reason": "Inappropriate content"
```

Example Response

Success Response (201 Created)

```
  "id": "123",
  "reason": "Inappropriate content",
  "postId": "1",
  "userId": "456",
  "createdAt": "2023-04-10T14:45:00Z"
```

Error Response (400 Bad Request)

```
  "error": "Reason for report is required"
```

Error Response (500 Internal Server Error)

```
  "error": "Failed to report content"
```

AI Disclosure: This file may partially contain contents generated by ChatGPT.