DATABASE SCHEMA:

```
generator client {
provider = "prisma-client-js"
}

datasource db {
 provider = "sqlite"
 url      = env("DATABASE_URL")
}

model User {
 id              String           @id @default(cuid())
 email           String           @unique
 password        String
 firstName       String?
 lastName        String?
 phoneNumber     String?
 avatar          String
@default("
QA/wD/AP+gvaeTAAACGUlEQVR4nO3TsQ2AMADAsNL+/xC3IdEHqqww2BdkyfXc6x3A0fw6AP7MIBAMAsEgEAwC
wSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEA
wCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgE
AwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEg
EAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEg
EAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEg
EAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDALBAMAsEgEAwCwSAQDAJhAyfSBEga26+EAAAAAElFTkSuQmCC")
 isAdmin         Boolean         @default(false)
 createdAt       DateTime        @default(now())
 updatedAt       DateTime        @updatedAt
 templates       CodeTemplate[]
 blogPosts       BlogPost[]
 comments        Comment[]
 blogPostReports BlogPostReport[]
 commentReports  CommentReport[]
 refreshToken    RefreshToken[]
 votes           UserVote[]
}

model RefreshToken {
 id        String   @id @default(uuid())
 token     String   @unique
 userId    String
 user      User     @relation(fields: [userId], references: [id])
 createdAt DateTime @default(now())
 updatedAt DateTime @updatedAt
}

model CodeTemplate {
 id          String     @id @default(cuid())
```

```
 title       String
 description String?
 code        String
 language    String
 tags        String
 authorId    String
 author      User       @relation(fields: [authorId], references: [id])
 createdAt   DateTime   @default(now())
 updatedAt   DateTime   @updatedAt
 blogPosts   BlogPost[]
 forked      Boolean    @default(false)
}

model BlogPost {
 id        String           @id @default(cuid())
 title     String
 content   String
 authorId  String
 author    User             @relation(fields: [authorId], references: [id])
 createdAt DateTime         @default(now())
 updatedAt DateTime         @updatedAt
 comments  Comment[]
 templates CodeTemplate[]
 tags      String
 report    BlogPostReport[]
 votes     UserVote[]
 hidden    Boolean          @default(false)
}

model UserVote {
 id         String    @id @default(cuid())
 userId     String
 user       User      @relation(fields: [userId], references: [id])
 blogPostId String?
 blogPost   BlogPost? @relation(fields: [blogPostId], references: [id])
 commentId  String?
 comment    Comment?  @relation(fields: [commentId], references: [id])
 type       Int // 1 for UPVOTE, -1 for DOWNVOTE
}

model Comment {
 id         String          @id @default(cuid())
 content    String
 authorId   String
 author     User            @relation(fields: [authorId], references: [id])
 blogPostId String
 blogPost   BlogPost        @relation(fields: [blogPostId], references: [id])
 createdAt  DateTime        @default(now())
 updatedAt  DateTime        @updatedAt
 reports    CommentReport[]
 parent     Comment?        @relation("commentToComment", fields: [parentId],
references: [id])
 parentId   String?
 children   Comment[]       @relation("commentToComment")
 votes      UserVote[]
```

```
 hidden     Boolean         @default(false)
}

model BlogPostReport {
 id                    String    @id @default(cuid())
 reason                String
 additionalExplanation String
 reporterId            String
 reporter              User      @relation(fields: [reporterId], references: [id])
 blogPostId            String?
 blogPost              BlogPost? @relation(fields: [blogPostId], references: [id])
 createdAt             DateTime  @default(now())
}

model CommentReport {
 id                    String   @id @default(cuid())
 reason                String
 additionalExplanation String
 reporterId            String
 reporter              User     @relation(fields: [reporterId], references: [id])
 commentId             String?
 comment               Comment? @relation(fields: [commentId], references: [id])
 createdAt             DateTime @default(now())
}
// used v0.dev to generate part of schema
```

ADMIN EMAIL AND PASSWORD:

email: **admin@example.com**
password: **adminPassword123**

API DOCUMENTATION

AUTHENTICATION:

POST /api/auth/signup

- **Description:** Creates a new user account. Returns an access token and sets a refresh token as a cookie upon successful registration.
- **Request Payload**

```
{
	"email": "newuser@example.com",
	"password": "password123",
	"firstName": "New",
	"lastName": "User",
	"phoneNumber": "1234567890"
}
```

- **Example Request**

```
POST /api/auth/signup HTTP/1.1
Host: localhost:3000
Content-Type: application/json
{
		"email": "newuser@example.com",
		"password": "password123",
		"firstName": "New",
		"lastName": "User",
		"phoneNumber": "1234567890"
}
```

- **Example Response**

  Success (201 Created)

```
{
		"message": "User created successfully",
		"user": {
			"id": "unique_user_id",
			"email": "newuser@example.com",
			"firstName": "New",
			"lastName": "User"
	},
	"accessToken": "access_token_value"
}
```

- **Error Responses**
  - **400 Bad Request:** Missing required fields
  - **400 Bad Request:** User already exists

- **405 Method Not Allowed:** Method not allowed.
- **500 Internal Server Error:** Internal Server Error

POST /api/auth/login

- **Description:** Authenticates a user and returns an access token and refresh token.
- **Payload:**

```
{
            "email": "string (required)",
            "password": "string (required)"
}
```

- **Example Request:**

```
POST http://localhost:3000/api/auth/login
Content-Type: application/json
{
            "email": "final@example.com",
            "password": "yourpassword123"
}
```

- **Example Response:**

```
{
             "message": "Logged in successfully",
            "user": {
                        "id": "user-id",
                        "email": "final@example.com",
                        "firstName": "John",
                        "lastName": "Doe"
            },
            "accessToken": "jwt-access-token"
}
```

- **Error Responses:**
  - **400 Bad Request**: Missing email or password.
  - **401 Unauthorized**: Invalid email or password.
  - **405 Method Not Allowed**: Method not allowed.
  - **500 Internal Server Error**: An error occurred on the server.

POST /api/auth/refresh

- **Description:** This endpoint allows clients to refresh their access token using a valid refresh token stored in cookies.
- **Request**
  - **Headers:** None required.
  - **Body:** No payload required.
- **Example Request:**

```
POST http://localhost:3000/api/auth/refresh
```

- **Responses:**
  - **200:** Returns a new access token.
  - **401 Unauthorized:** No refresh token provided *or* Invalid refresh token *or* Refresh token not found
  - **Status Code 500 Internal Server Error:** An error occurred on the server.

POST /api/auth/logout

- **Description:** Logs the user out by deleting the refresh token from the database and clearing the associated cookie.
- **Payload:** None (the refresh token is expected to be provided in cookies)
- **Example Request:**

```
POST http://localhost:3000/api/auth/logout
Cookie: refreshToken=<your_refresh_token>
```

- **Responses:**
  - **200:** Logged out successfully
  - **400 Bad Request:** No refresh token provided
  - **405 Method Not Allowed:** Method not allowed
  - **500 Internal Server Error:** Internal server error

ADMIN:

GET /api/admin/reports

- **Description:** Retrieves paginated blog posts and comments that have been reported, including their respective report counts. This endpoint is restricted to admin users.
- **Query Parameters:**
  - `page` (optional, default: 1): The page number of the results to retrieve.
  - `limit` (optional, default: 10): The number of items to return per page.
- **Authorization:** Bearer token required. Only accessible by users with admin privileges.
- **Example Request:**

```
GET http://localhost:3000/api/admin/reports?page=1&limit=2
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjbTMyNmM4NW4wMDBydGtlbGI0NmYwOWxhIiwiaWF0IjoxNzMwNjczNTM3LCJleHAiOjE3MzA2NzQ0Mzd9.c0MI_O6W6Ft0Z5vusjLncloO93AYy-ck9eO1nQml4GE
```

- **Example Response:**

```
{
    "blogPosts": {
        "items": [ /* Array of blog post objects */ ],
        "pagination": {
```

```
            "totalItems": 5,
            "totalPages": 3,
            "currentPage": 1,
            "pageSize": 2
        }
    },
    "comments": {
        "items": [ /* Array of comment objects */ ],
        "pagination": {
            "totalItems": 10,
            "totalPages": 5,
            "currentPage": 1,
            "pageSize": 2
        }
    }
}
```

- **Responses:**
    - **200 OK:** Successfully retrieved reports.
    - **403 Forbidden:** Unauthorized access (user is not an admin).
    - **500 Internal Server Error:** Error retrieving reported content.
    - **405 Method Not Allowed:** The method is not supported for this endpoint

---

POST /api/admin/hide-content

- **Description**: This endpoint allows an admin to hide or unhide a specific piece of content (either a blog post or a comment).
- **Payload**:

```
{
            "contentId": "string",
            "contentType": "string", // either "blogPost" or "comment"
            "hide": true // or false
}
```

- **Example Request**:

```
POST /api/admin/hide-content HTTP/1.1
Host: localhost:3000
Authorization: Bearer <your_token>
Content-Type: application/json
{
            "contentId": "cm30hsafw000612i90e1bwoib",
            "contentType": "comment",
            "hide": true
}
```

- **Example Response**:

```
{
            "message": "Content hidden successfully",
            "updatedContent": { /* updated content object */ }
}
```

- **Responses**:
  - **200 OK:** Content successfully updated.
  - **403 Forbidden:** Unauthorized access (not an admin).
  - **400 Bad Request:** Invalid content type.
  - **500 Internal Server Error:** Error updating content visibility.

CODE EXECUTION:

POST /api/execute

- **Description:** This endpoint executes a provided code snippet in the specified programming language and streams the output back to the client in real-time using Server-Sent Events (SSE).
- **Request Payload**

```
Content-Type: application/json
Accept: text/event-stream
Body:
{
                        "code": "import sys\\nimport time\\ndef fibonacci(n):\\n
if n <= 1:\\n        return n\\n    else:\\n        return fibonacci(n-1) +
fibonacci(n-2)\\nprint(\\"Fibonacci sequence up to 10th
term:\\")\\nsys.stdout.flush()\\nfor i in range(10):\\n    print(fibonacci(i), end=\\"
\\")\\n    sys.stdout.flush()\\n    time.sleep(0.5)\\nprint()\\nsys.stdout.flush()",
                       "language": "python",
                       "input": ""
}
```

- **Example Request**

```
POST http://localhost:3000/api/execute
Content-Type: application/json
Accept: text/event-stream
{
                        "code": "import sys\\nimport time\\ndef fibonacci(n):\\n
if n <= 1:\\n        return n\\n    else:\\n        return fibonacci(n-1) +
fibonacci(n-2)\\nprint(\\"Fibonacci sequence up to 10th
```

```
term:\\")\\nsys.stdout.flush()\\nfor i in range(10):\\n    print(fibonacci(i), end=\\"
\\")\\n    sys.stdout.flush()\\n    time.sleep(0.5)\\nprint()\\nsys.stdout.flush()",
                        "language": "python",
                        "input": ""
      }
```

- **Example Response**

  The response is streamed as a series of events. Each event contains a session ID, output type, and the data produced by the executed code.

```
HTTP/1.1 200 OK
Content-Type: text/event-stream
Cache-Control: no-cache
Connection: keep-alive
data:
{"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"Fibonacci
sequence up to 10th term:\\n0 "}
data: {"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"1
"}
data: {"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"1
"}
data: {"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"2
"}
data: {"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"3
"}
data: {"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"5
"}
data: {"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"8
"}
data:
{"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"13 "}
data:
{"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"21 "}
data:
{"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"stdout","data":"34 "}
data:
{"sessionId":"efba4fd0f2949dd6fa5a6a7d7cc5bb0c","type":"status","data":"completed","co
de":0}
```

VOTING:

`POST /api/blogs/:id/vote`

- **Description:** Allows users to vote on a specific blog post. Users can upvote or downvote, and can also toggle their vote.

- **Request Payload:**

```
{
  "userId": "string",
  "type": "number" // 1 for upvote, -1 for downvote
}
```

- **Example Request:**

```
POST http://localhost:3000/api/blogs/cm325mz7h0004tkelecfgsweh/vote
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjbTMyNWR1ZzgwMDAwdGtlbGx2bmN1ODJzI
iwiaWF0IjoxNzMwNjczNDI4LCJleHAiOjE3MzA2NzQzMjh9.jzPbD8uH6Jo9_hkzYcv29qpt-TzIpbK6Jt_wtr
-gxpU
Content-Type: application/json
{
 "userId": "cm325dug80000tkellvncu82s",
 "type": 1
}
```

- **Example Response:**
  - **On Success (Vote Created/Updated):**

```
{
 "id": "string", // The ID of the vote
 "userId": "string", // ID of the user who voted
 "blogPostId": "string", // ID of the blog post voted on
 "type": 1 // 1 for upvote, -1 for downvote
}
```

  - **On Success (Vote Removed):**

```
{
 "message": "Vote removed"
}
```

  - **On Error (Unauthorized):**

```
{
  "error": "string" // Error message for unauthorized access
}
```

  - **On Error (Method Not Allowed):**

```
{
 "error": "Method not allowed"
}
```

  - **On Error (Internal Server Error):**

```
{
  "error": "Failed to process vote"
}
```

POST /api/blogs/:id/comments/:commentId/vote

- **Description:** Allows a user to vote on a specific comment by upvoting or downvoting it. If a vote already exists, it toggles or updates the vote type.
- Request Payload

```
{
 "userId": "string (required)",
 "type": "integer (required, 1 for upvote, -1 for downvote)"
}
```

- Example Request

```
POST /api/blogs/cm325mz7h0004tkelecfgsweh/comments/cm2zizkgl0000o8hwf8pgy69z/vote HTTP/1.1
Host: localhost:3000
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjbTMyNWR1ZzgwMDAwdGtlbGx2bmN1ODJzIiwiaWF0IjoxNzMwNjczNDI4LCJleHAiOjE3MzA2NzQzMjh9.jzPbD8uH6Jo9_hkzYcv29qpt-TzIpbK6Jt_wtr-gxpU
Content-Type: application/json
{
	"userId": "cm325dug80000tkellvncu82s",
	"type": 1
}
```

- Example Response
  - **Success (Vote Created or Updated):**
    - **Status Code:** 201 (Created) or 200 (OK)
    - **Body:**

```
{
  "id": "string (vote ID)",
  "userId": "string (user ID)",
  "commentId": "string (comment ID)",
  "type": "integer (1 or -1)"
}
```

  - **403 (Forbidden):** Authorization error message
  - **405 (Method Not Allowed):** Method not allowed
  - **500 (Internal Server Error):** Failed to process vote

REPORTING SECTION:

/api/blogs/:id/report

- **Description**: Reports a blog post for inappropriate content.
- **Allowed Methods**:
  - POST: Submit a report for a blog post.
  - GET: Retrieve all reports for a specific blog post.
- **Request**
  - **Authorization**: Bearer token required.
  - **Payload** (for POST):

```
{
  "userId": "string",             // ID of the user reporting the blog
  "reason": "string",             // Reason for the report
  “additionalExplanation": "string" // Additional details regarding the report
}
```

- Example Request
  - **POST Request**:

```
POST http://localhost:3000/api/blogs/cm325mz7h0004tkelecfgsweh/report
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjbTMyNWR1d…
Content-Type: application/json
{
      "userId": "cm325dug80000tkellvncu82s",
      “reason": "Inappropriate content",
      "additionalExplanation": "The blog post contains offensive language and
      should be reviewed."
}
```

- Responses:
  - **201 Created**:

```
{
      "id": "string",                   // ID of the created report
      "reason": "Inappropriate content",
      "additionalExplanation": "The blog post contains offensive
      language and should be reviewed.",
      "reporter": {
            "id": "cm325dug80000tkellvncu82s", // ID of the user who
reported
            "name": "string"                  // Name of the user
      },
      "blogPost": {
      "id": "cm325mz7h0004tkelecfgsweh"  // ID of the reported blog post
      }
}
```

- **403 Forbidden**: Unauthorized access
- **400 Bad Request**: Invalid input
- **500 Internal Server Error**: Error creating report

/api/blogs/:id/comments/:commentId/report

- **Methods**:
  - POST: Create a report for a specific comment.
  - GET: Retrieve all reports for a specific comment.

- Request Headers:

- **Authorization**: Bearer token required.

- Request Body (for POST method):

```
{
  "userId": "cm325dug80000tkellvncu82s",
  "reason": "Inappropriate content",
  "additionalExplanation": "The comment contains offensive language and should be reviewed."
}
```

- Example Request (for POST method):

```
POST /api/blogs/cm325mz7h0004tkelecfgsweh/comments/cm325qn3h0006tkelhvmlpa3h/report HTTP/1.1
Host: localhost:3000
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjbTMyNWR1ZzgwMDAwdGtlbGx2bmN1ODJzIiwiaWF0IjoxNzMwNjcyMjU2LCJleHAiOjE3MzA2NzMxNTZ9.2CK_xt7heky-wuFmSusoD8jNWXc6NGAq7KQKMao6cA0
Content-Type: application/json
{
  "userId": "cm325dug80000tkellvncu82s",
  "reason": "Inappropriate content",
  "additionalExplanation": "The comment contains offensive language and should be reviewed."
}
```

Example Response (for POST method):

```
{
  "id": "reportId",
  "reason": "Inappropriate content",
```

```
        "reporterId": "cm325dug80000tkellvncu82s",
        "commentId": "cm325qn3h0006tkelhvmlpa3h",
        "additionalExplanation": "The comment contains offensive language and should
be reviewed.",
        "createdAt": "2024-11-03T12:00:00.000Z"
      }
```

Example Response (for GET method):

```
      [
        {
          "id": "reportId1",
          "reason": "Inappropriate content",
          "reporterId": "cm325dug80000tkellvncu82s",
          "commentId": "cm325qn3h0006tkelhvmlpa3h",
          "additionalExplanation": "The comment contains offensive language and
should be reviewed.",
          "createdAt": "2024-11-03T12:00:00.000Z"
        },
        {
          "id": "reportId2",
          "reason": "Spam",
          "reporterId": "cm325dug80000tkellvncu82s",
          "commentId": "cm325qn3h0006tkelhvmlpa3h",
          "additionalExplanation": "This comment is irrelevant to the discussion.",
          "createdAt": "2024-11-03T12:01:00.000Z"
        }
      ]
```

BLOGS ENDPOINTS:

POST /api/blogs

- **Description:** Creates a new blog post. The request must include the title, content, author ID, template IDs (optional), and tags.
- Request Payload

```
      {
          "title": "Final Test Blog",
          "content": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.",
          "authorId": "cm325dug80000tkellvncu82s",
```

```
        "tags": "Javascript",
        "templateIds": ["cm2zdl3yf000f1sbfrx4eeir8"]
    }
```

- Example Request

```
POST /api/blogs HTTP/1.1
Host: localhost:3000
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json
{
    "title": "Final Test Blog",
    "content": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.",
    "authorId": "cm325dug80000tkellvncu82s",
    "tags": "Javascript",
    "templateIds": ["cm2zdl3yf000f1sbfrx4eeir8"]
}
```

Example Response

- **Status Code:** `201 Created`
- **Response Body:**

```
{
    "id": "new-blog-post-id",
    "title": "Final Test Blog",
    "content": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.",
    "authorId": "cm325dug80000tkellvncu82s",
    "tags": "Javascript",
    "templateIds": ["cm2zdl3yf000f1sbfrx4eeir8"]
}
```

Error Responses

- **403 Forbidden:** User not authorized.
- **500 Internal Server Error**

`/api/blogs/:id`

This endpoint allows you to retrieve, update, or delete a blog post by its ID.

Allowed Methods

- **GET**: Retrieve a blog post by ID.
- **PUT**: Update a blog post by ID.
- **DELETE**: Delete a blog post by ID.

Request Payloads

- **GET**: No payload required.
- **PUT**:

```
{
    "title": "Updated Blog Post Title",
    "content": "This is the updated content of the blog post.",
    "templateIds": [],
    "tags": "Javascript,C#"
}
```

- **DELETE**: No payload required.

Example Requests

GET Request

```
GET /api/blogs/:id HTTP/1.1
Authorization: Bearer <your_token_here> // optional
```

Example Response

```
{
    "id": "cm2zbkw5f00041sbfa04yio17",
    "title": "Blog Post Title",
    "content": "This is the content of the blog post.",
    "hidden": false,
    "author": {
        "id": "author_id",
        "firstName": "John",
        "lastName": "Doe"
    },
    "comments": [
        // Filtered comments based on user authorization
    ],
    "templates": [
        // Associated templates
    ],
    "tags": "tag1,tag2"
}
```

PUT Request

```
PUT /api/blogs/:id HTTP/1.1
Authorization: Bearer <your_token_here>
Content-Type: application/json
{
    "title": "Updated Blog Post Title",
    "content": "This is the updated content of the blog post.",
    "templateIds": [],
    "tags": "Javascript,C#"
}
```

Example Response

```
{
    "id": "cm325hnhw0003tkelzkyjscv5",
    "title": "Updated Blog Post Title",
    "content": "This is the updated content of the blog post.",
    "hidden": false,
    "author": {
        "id": "author_id",
        "firstName": "John",
        "lastName": "Doe"
    },
    "comments": [],
    "templates": [],
    "tags": "Javascript,C#"
}
```

DELETE Request

DELETE /api/blogs/:id HTTP/1.1
Authorization: Bearer <your_token_here>

Example Response

```
{
    "id": "cm325hnhw0003tkelzkyjscv5",
    "title": "Deleted Blog Post Title",
    "content": "This blog post has been deleted."
}
```

POST /api/blogs/:id/comments

- **Description:** This endpoint allows users to create a new comment on a specified blog post. The user must be authorized to post comments.

- **Request Payload:**

```
{
    "content": "This is a new reply on the final blog post that will be
reported. 2",
    "authorId": "cm325dug80000tkellvncu82s",
    "blogPostId": "cm325mz7h0004tkelecfgsweh",
    "parentId": "optional_parent_id"  // optional
}
```

- **Example Request:**

```
POST http://localhost:3000/api/blogs/:id/comments
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjbTMyNWR1ZzgwMDAwdGtlbGx2bmN1ODJzI
iwiaWF0IjoxNzMwNjcyMjU2LCJleHAiOjE3MzA2NzMxNTZ9.2CK_xt7heky-wuFmSusoD8jNWXc6NGAq7KQKMa
o6cA0
Content-Type: application/json
{
    "content": "This is a new reply on the final blog post that will be
reported. 2",
    "authorId": "cm325dug80000tkellvncu82s",
    "blogPostId": "cm325mz7h0004tkelecfgsweh"
}
```

- **Example Response:**
    - **Success (201 Created):**

```
{
    "id": "new_comment_id",
    "content": "This is a new reply on the final blog post that will be
reported. 2",
    "author": {
        "id": "cm325dug80000tkellvncu82s",
        "name": "Author Name"
    },
    "blogPost": {
        "id": "cm325mz7h0004tkelecfgsweh",
        "title": "Blog Post Title"
    },
    "parent": null,  // if no parent
    "children": []   // array of child comments
}
```

- **Error (400 Bad Request):** content field empty
- **Error (403 Forbidden):** Authorization failed
- **Error (405 Method Not Allowed):** Method not allowed

GET /api/templates/search

- **Description:**
  Search for templates based on a specified search term, with pagination support.
- **Query Parameters:**
  - `searchTerm` (string): The term to search for in templates.
  - `page` (number): The page number for pagination (default: 1).
  - `limit` (number): The number of results per page (default: 10).
- **Example Request:**

```
GET http://localhost:3000/api/templates/search?searchTerm=example()&page=1&limit=2
```

- **Example Response:**

```
{
              "templates": [
                    {
"id": 1,
                    "name": "Template 1",
                    "content": "This is an example template."
                    },
                     {
                    "id": 2,
                           "name": "Template 2",
                    "content": "Another example template."
                    }
               ],
               "total": 2,
"page": 1,
             "limit": 2
}
```

POST /api/templates

- **Method**:
- **Description**: Creates a new code template with the provided details.
- Request Payload

```
{
    "title": "Example Code Template",
    "description": "This is a sample template for demonstrating code structure.",
    "code": "function example() { return 'Hello, world!'; }",
    "language": "JavaScript",
```

```
    "tags": "example,template,code",
    "authorId": "cm2zbkarm00011sbfbnh35j5d"
}
```

Example Request

```
POST http://localhost:3000/api/templates
Content-Type: application/json
{
    "title": "Example Code Template",
    "description": "This is a sample template for demonstrating code
structure.",
    "code": "function example() { return 'Hello, world!'; }",
    "language": "JavaScript",
    "tags": "example,template,code",
    "authorId": "cm2zbkarm00011sbfbnh35j5d"
}
```

- Example Response
  - **Status**: `201 Created`
  - **Body**:

```
{
    "id": "new_template_id",
    "title": "Example Code Template",
    "description": "This is a sample template for demonstrating code
structure.",
    "code": "function example() { return 'Hello, world!'; }",
    "language": "JavaScript",
    "tags": "example,template,code",
    "author": {
        "id": "cm2zbkarm00011sbfbnh35j5d",
        "name": "Author Name"
    },
    "blogPosts": []
}
```

Response Status Codes

- `201`: Template created successfully.
- `403`: Forbidden - User is not authorized.
- `500`: Internal Server Error - Failed to create template.
- `405`: Method Not Allowed - Invalid request method.

`/api/templates/:id`

Description:

This endpoint allows you to retrieve, update, or delete a code template by its ID.

Allowed Methods:

GET: Retrieve the specified code template.
- **PUT**: Update the specified code template.
- **DELETE**: Delete the specified code template.

Payloads:

- **GET**: No payload required.
- **PUT**: Requires a JSON payload with the following fields:
  - `title` (string): The title of the code template.
  - `description` (string): A description of the code template.
  - `code` (string): The code content of the template.
  - `language` (string): The programming language of the code.
  - `tags` (string): A comma-separated list of tags.
- **DELETE**: No payload required.

Example Requests and Responses:

1. GET Request:

```
GET http://localhost:3000/api/templates/cm2zd4bpf000c1sbfnvc2u8by
```

Response:

```
{
  "id": "cm2zd4bpf000c1sbfnvc2u8by",
  "title": "Template Title",
  "description": "Description of the code template.",
  "code": "function example() { return 'Hello, world!'; }",
  "language": "JavaScript",
  "tags": "example,template",
  "author": {
    "id": "authorId",
    "name": "Author Name"
  },
  "blogPosts": [
    // Array of related blog posts
  ]
}
```

2. PUT Request:

```
PUT http://localhost:3000/api/templates/cm2zd4bpf000c1sbfnvc2u8by
Content-Type: application/json
{
    "title": "Updated Template Title",
    "description": "Updated description of the code template.",
    "code": "function updatedExample() { return 'Updated code'; }",
    "language": "JavaScript",
    "tags": "updated,example,code"
}
```

Response:

```
{
  "id": "cm2zd4bpf000c1sbfnvc2u8by",
  "title": "Updated Template Title",
  "description": "Updated description of the code template.",
  "code": "function updatedExample() { return 'Updated code'; }",
  "language": "JavaScript",
  "tags": "updated,example,code",
  "author": {
    "id": "authorId",
    "name": "Author Name"
  },
  "blogPosts": [
    // Array of related blog posts
  ]
}
```

3. DELETE Request:

```
DELETE http://localhost:3000/api/templates/cm2zd4bpf000c1sbfnvc2u8by
```

Response:

```
{
  "id": "cm2zd4bpf000c1sbfnvc2u8by",
  "title": "Template Title",
  "description": "Description of the code template.",
  "code": "function example() { return 'Hello, world!'; }"
  "language": "JavaScript",
  "tags": "example,template"
}
```

POST /api/templates/:id/fork

Description: This endpoint allows users to create a fork of an existing template by providing new details while retaining the original code and tags.

- **Method**:

- **Authentication**: Bearer token required.

- **Request Payload**:

```
{
    "userId": "cm30fevkh00005m10r9j8rm4j",
    "newTitle": "Forked Template Title 2",
    "newDescription": "This is an updated description.",
    "newCode": null,  // This will use the original code
    "newLanguage": "JavaScript",
    "newTags": null  // This will use the original tags
}
```

- **Example Request**:

```
POST http://localhost:3000/api/templates/:id/fork

Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjbTMwZmV2a2gwMDAwNW0xMHI5ajhybTRqIiwiaWF0IjoxNzMwNTc2NjM4LCJleHAiOjE3MzA1Nzc1Mzh9.LzO97-wUUUDr9vPsi8diuOihVjyTuc51bR3D1vytEJk

Content-Type: application/json

{
    "userId": "cm30fevkh00005m10r9j8rm4j",
    "newTitle": "Forked Template Title 2",
    "newDescription": "This is an updated description.",
    "newCode": null,
    "newLanguage": "JavaScript",
    "newTags": null
}
```

- **Example Response**:

```
{
    "success": true,
    "message": "Template forked successfully",
    "templateId": "newTemplateId"
```

```
}
```

/api/templates/search

Description

Search for code templates based on a given search term. This endpoint supports both authenticated (POST) and unauthenticated (GET) requests. Authentication allows a user to view their own templates easily.

Allowed Methods

POST: Requires authentication.
- **GET**: No authentication required.

Request Payload (for POST)

```
{
    "userId": "cm30fevkh00005m10r9j8rm4j"
}
```

Query Parameters

- **searchTerm**: The term to search for in the templates (e.g., "function").
- **page**: The page number for pagination (default is 1).
- **limit**: The number of results to return per page (default is 10).

Example Request (POST)

```
POST
http://localhost:3000/api/templates/search?searchTerm=function&page=1&limit=2

Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjbTMwZmV2a2gwMDAwNW0xMHI5ajhybTRqI
iwiaWF0IjoxNzMwNTgwNjc2LCJleHAiOjE3MzA1ODE1NzZ9.rbDuoaZZ3c_wYsxx1ItIE3kdYsErX2UW9wJ8tm
KbA_I

Content-Type: application/json

{
    "userId": "cm30fevkh00005m10r9j8rm4j"
}
```

Example Request (GET)

```
GET
http://localhost:3000/api/templates/search?searchTerm=example()&page=1&limit=2
```

Example Response (Success)


```
{
    "totalTemplates": 10,
    "currentPage": 1,
    "totalPages": 5,
    "templates": [
        {
            "id": "templateId1",
            "title": "Example Template 1",
            "description": "Description for template 1",
            "tags": "tag1, tag2",
            "code": "console.log('Hello World')"
        },
        {

            "id": "templateId2",
            "title": "Example Template 2",
            "description": "Description for template 2",
            "tags": "tag3, tag4",
            "code": "function example() {}"
        }
    ]
}
```


Error Responses

- **403 Forbidden**: Unauthorized request.
- **500 Internal Server Error**: Error retrieving templates.

`/api/profile`

Description:
This endpoint allows you to retrieve and update a user profile based on the provided user ID.

Allowed Methods:

- GET: Retrieve user profile
- POST: Update user profile

GET Request

Request URL:

```
GET http://localhost:3000/api/profile?userId=<USER_ID>
```

Query Parameters:

- `userId` (required): The ID of the user whose profile you want to retrieve.

Example Request:

```
GET http://localhost:3000/api/profile?userId=cm2tqgfep0000fqy5p5g8dgqi
```

Example Response:

```
{
    "id": "cm2tqgfep0000fqy5p5g8dgqi",
    "firstName": "John",
    "lastName": "Doe",
    "phoneNumber": "+1234567890",
    "email": "john.doe@example.com",
    "avatar": "..."
}
```

POST Request

Request URL:

```
POST http://localhost:3000/api/profile?userId=<USER_ID>
```

Query Parameters:

- `userId` (required): The ID of the user whose profile you want to update.

Payload (Body):

```
{
    "firstName": "Jane Admin",
    "lastName": "Doe",
    "phone": "+1234567890",
    "color": "green",
    "email": "owner@e.com"
}
```

Example Request:

```
POST http://localhost:3000/api/profile?userId=cm2p0sm630000xougw6totvns
```

Example Response:

```
{
    "id": "cm2p0sm630000xougw6totvns",
    "firstName": "Jane Admin",
    "lastName": "Doe",
    "phoneNumber": "+1234567890",
    "email": "owner@e.com",
    "avatar": "..."
}
```


Error Responses:

- **`400 Bad Request`**: Missing or invalid parameters.
- **`403 Forbidden`:** User is not authorized to update the profile.
- **`404 Not Found`**: User profile not found.
- **`500 Internal Server Error`:** An error occurred while processing the request.
