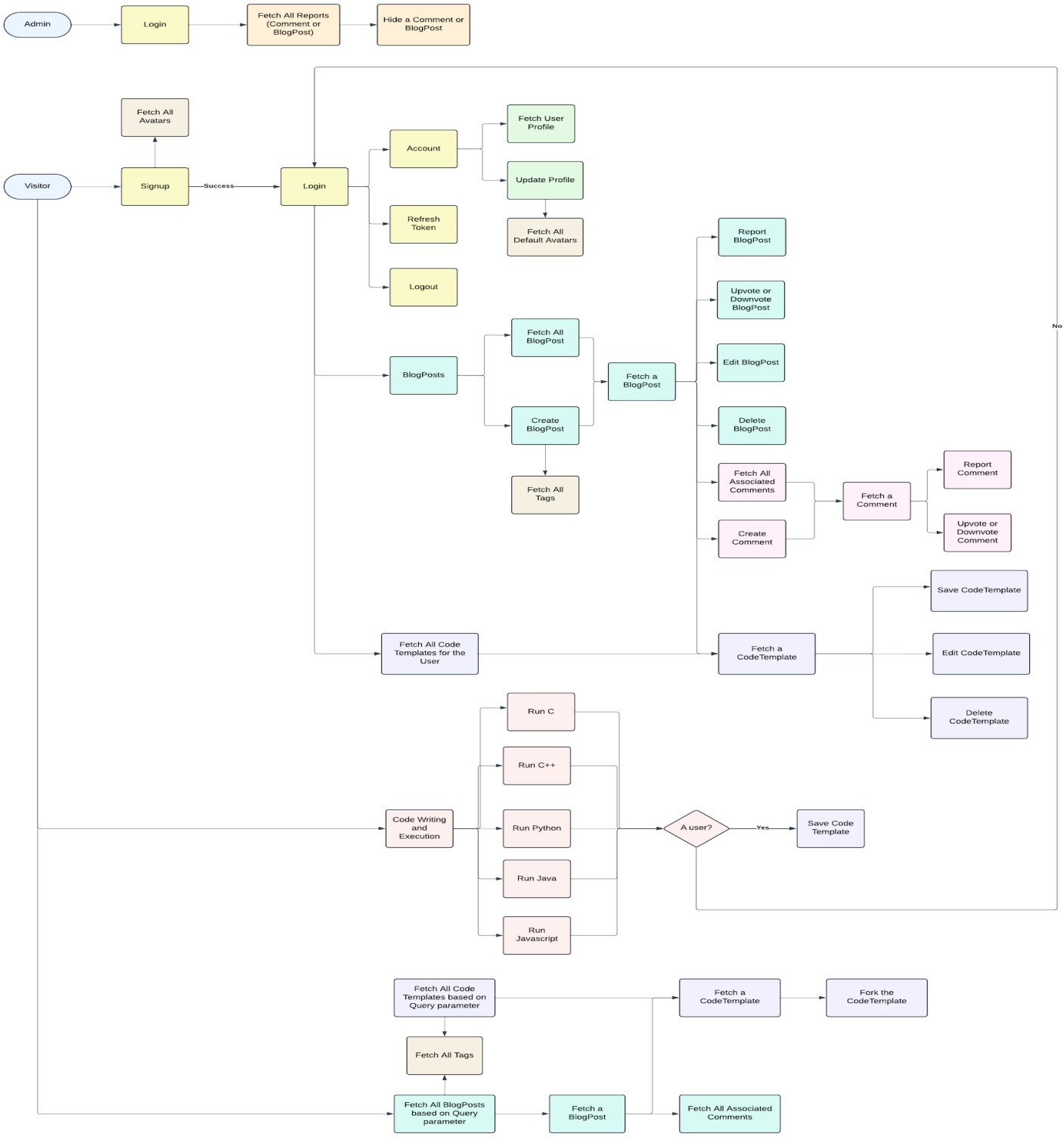
Scriptorium API Documentation

Model Design



Base URL

http://localhost:30

Account Management

Sign Up

Creates a new user account in the system.

Endpoint: `/api/auth/signup`
Method: POST
Authentication: None

Request Body:

```
{

    "email": "testavatar@gmail.com",

    "password": "Test123*",

    "firstName": "test",

    "lastName": "test",

    "phone": "0399394909",

    "role": "ADMIN"  // Optional, defaults to "USER"

}
```

Response:

```
{

    "id": 12,

    "firstName": "test",

    "lastName": "test",

    "phone": "0399394909",
```

"role": "ADMIN",

"avatar": "/avatars/avatar1.jpg"

}

Status Code: 201 Created

Validation:

- Email must be in valid format (e.g., user@example.com)
- Password must meet complexity requirements
- Phone number must be in valid format
- Email and phone number must be unique in the system

Login

Authenticates a user and returns access and refresh tokens.

Endpoint: `/api/auth/login`
Method: POST
Authentication: None

Request Body:

{

"email": "test5@gmail.com",

"password": "Test123*"

}

Response:

{

"accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

}

Status Code: 200 OK

Logout

Invalidates the current user session.

Endpoint: `/api/auth/logout`
Method: POST
Authentication: Required

Response:

{

"message": "Logged out successfully"

}

Status Code: 200 OK

Refresh Token

Generates a new access token using a refresh token.

Endpoint: `/api/auth/refresh`
Method: POST
Authentication: Required (via refresh token)

Response:

{

"accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

}

Status Code: 200 OK

Get Profile

Retrieves the current user's profile information.

Endpoint: `/api/user/profile`
Method: GET
Authentication: Required

Response:


```
{
    "id": 12,
    "email": "test5@gmail.com",
    "firstName": "test",
    "lastName": "test",
    "avatar": "/avatars/avatar1.jpg",
    "phone": "0399394909"
}
```


Status Code: 200 OK

Update Profile

Updates the current user's profile information.

Endpoint: `/api/user/update`
Method: PUT
Authentication: Required

Request Body:


```
{
    "firstName": "john",
    "lastName": "doe",
    "avatar": "/avatars/avatar2.png"
}
```

Response:


```
{
    "id": 12,
    "email": "test5@gmail.com",
    "firstName": "john",
    "lastName": "doe",
    "avatar": "/avatars/avatar2.png",
    "phone": "0399394909"
}
```


Status Code: 200 OK

Note: The update profile endpoint allows partial updates - you can update one or more fields without having to provide all fields. Only the provided fields will be updated.

Code Execution Endpoints

Run JavaScript Code

Executes JavaScript code and returns the output.

Endpoint: `/api/run-javascript`
Method: POST
Authentication: None

Request Body:


```
{
    "code": "console.log(\"Hello, World!\");"
}
```


Response:

```
{
    "output": "Hello, World!\n"
}
```


Run Python Code

Executes Python code and returns the output.

Endpoint: `/api/run-python`
Method: POST
Authentication: None

Request Body:


```
{
    "code": "print(\"Hello, World!\")"
}
```


Response:


```
{
    "output": "Hello, World!\n"
}
```


Run C Code

Executes C code and returns the output.

Endpoint: `/api/run-c`
Method: POST
Authentication: None

Request Body:


```
{
    "code": "#include <stdio.h>\n\nint main() {\n    printf(\"Hello, World!\\n\");\n    return 0;\n}"
```

}

Response:

{

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

}

Run C++ Code

Executes C++ code and returns the output.

Endpoint: `/api/run-cpp`
Method: POST
Authentication: None

Request Body:

{

"code": "#include <iostream>\n\nint main() {\n std::cout << \"Hello, World!\" << std::endl;\n return 0;\n}"

}

Response:

{

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

}

Run Java Code

Executes Java code and returns the output.

Endpoint: `/api/run-java`
Method: POST
Authentication: None

Request Body:

{

"code": "public class HelloWorld {\n public static void main(String[] args) {\n System.out.println(\"Hello, World!\");\n }\n}"

}

Response:

{

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

}

Note: All code execution endpoints accept code as a string in the request body and return the program output as a string. Each endpoint is specific to a programming language and handles compilation (if needed) and execution of the code. Any errors during compilation or execution will be included in the output.

Code Templates

Save Template

Creates a new code template.

Endpoint: `/api/code-template/save`
Method: POST
Authentication: Required

Request Body:

{

"title": "Test",

"explanation": "Test Test",

"code": "int x",

"language": "Python",

```
    "tag_ids": [1]

}
```


Response:


```
{

    "id": 8,

    "title": "Test",

    "explanation": "Test Test",

    "code": "int x"

}
```


Status Code: 200 OK

Fetch All Code Templates

Retrieves all code templates with optional filtering.

Endpoint: `/api/code-template/fetch-all`
Method: GET
Authentication: None

Query Parameters:

- page (number): Page number for pagination
- limit (number): Number of items per page
- title (string): Filter by title
- tag_id (number): Filter by tag ID
- explanation (string): Filter by explanation content

Response:


```
{

    "data": [

        {
```

```
        "id": 3,
        "title": "Basic JavaScript Loop",
        "explanation": "A simple loop in JavaScript that prints numbers from 0 to 9.",
        "code": "for(let i = 0; i < 10; i++) { console.log(i); }",
        "language": "JavaScript",
        "authorId": 6,
        "createdAt": "2024-11-01T20:31:00.731Z",
        "updatedAt": "2024-11-01T20:31:00.731Z",
        "tags": [
          {
            "id": 2,
            "name": "JavaScript"
          }
        ],
        "author": {
          "id": 6,
          "firstName": "Admin",
          "lastName": "Admin"
        }
      }
    ],
    "totalPages": 1,
```

```
  "totalCount": 1

}
```


Fetch User's Code Templates

Retrieves code templates for the authenticated user.

Endpoint: `/api/code-template/fetch-user`
Method: GET
Authentication: Required

Query Parameters:

- page (number): Page number for pagination
- limit (number): Number of items per page
- tag_id (number): Filter by tag ID
- title (string, optional): Filter by title
- explanation (string, optional): Filter by explanation content

Response:


```
{

  "data": [

    {

      "id": 8,

      "title": "Test",

      "explanation": "Test Test",

      "code": "int x",

      "language": "Python",

      "authorId": 12,

      "createdAt": "2024-11-02T22:16:23.247Z",

      "updatedAt": "2024-11-02T22:16:23.247Z",
```

```
            "tags": [
                {
                    "id": 2,
                    "name": "JavaScript"
                }
            ],
            "author": {
                "id": 12,
                "firstName": "john",
                "lastName": "doe"
            }
        }
    ],
    "totalPages": 1,
    "totalCount": 1
}
```


Edit Code Template

Updates an existing code template.

Endpoint: `/api/code-template/edit`
Method: PUT
Authentication: Required

Request Body:


```
{
```

```
    "codeTemplateId": 3,

    "code": "int y",

    "title": "Optional title update",

    "explanation": "Optional explanation update",

    "tag_ids": [2]

}
```


Note: All fields except `codeTemplateId` are optional in the update request.

Delete Code Template

Deletes a code template.

Endpoint: `/api/code-template/delete`
Method: DELETE
Authentication: Required

Request Body:


```
{

    "codeTemplateId": 8

}
```


Response:


```
{

    "message": "Code template deleted successfully",

    "codeTemplateId": 8

}
```


Get Specific Code Template

Retrieves a specific code template by ID.

Endpoint: `/api/code-template/fetch`
Method: GET
Authentication: None

Query Parameters:

- id (number): The ID of the code template to retrieve

Response:


```
{
  "data": {
    "id": 7,
    "title": "Test",
    "explanation": "Test Test",
    "code": "int x",
    "language": "Python",
    "authorId": 10,
    "createdAt": "2024-11-02T21:56:56.124Z",
    "updatedAt": "2024-11-02T21:56:56.124Z",
    "tags": [
      {
        "id": 2,
        "name": "JavaScript",
        "createdAt": "2024-11-01T20:31:00.728Z",
        "updatedAt": "2024-11-01T20:31:00.728Z"
      }
```

```
    ]
  }
}
```

Note: Code templates can be associated with multiple tags for better organization and searchability. When creating or editing a template, you can specify multiple tag IDs. The template's author information is automatically included in the response when fetching templates.

Blog Posts

Create Blog Post

Creates a new blog post.

Endpoint: `/api/blog-post/create`
Method: POST
Authentication: Required

Request Body:

```
{
  "title": "Test Blogpost Title",
  "description": "This is a test description for the blog post",
  "tag_ids": [39]
}
```

Response:

```
{
  "id": 17,
  "title": "Test Blogpost Title",
  "description": "This is a test description for the blog post",
  "authorId": 32,
```

```
    "tags": [
        {
            "id": 39,
            "name": "JavaScript",
            "createdAt": "2024-11-02T22:56:12.059Z",
            "updatedAt": "2024-11-02T22:56:12.059Z"
        }
    ],
    "upvotes": 0,
    "downvotes": 0,
    "createdAt": "2024-11-02T22:58:08.695Z",
    "updatedAt": "2024-11-02T22:58:08.695Z"
}
```


Edit Blog Post

Updates an existing blog post.

Endpoint: `/api/blog-post/edit`
Method: PUT
Authentication: Required

Request Body:


```
{
    "blogPostId": 17,
    "title": "UPDATED title",
    "description": "This is the UPDATED main content of the blog post.",
```

```
    "tag_ids": [40]
}
```


Response:


```
{
    "id": 17,
    "title": "UPDATED title",
    "description": "This is the UPDATED main content of the blog post.",
    "tags": [
        {
            "id": 40,
            "name": "Python",
            "createdAt": "2024-11-02T22:56:12.068Z",
            "updatedAt": "2024-11-02T22:56:12.068Z"
        }
    ],
    "upvotes": 0,
    "downvotes": 0
}
```


Delete Blog Post

Deletes a blog post.

Endpoint: `/api/blog-post/delete`
Method: DELETE
Authentication: Required

Request Body:


```
{
   "blogPostId": 18
}
```


Response:


```
{
   "message": "Blog post deleted successfully",
   "blogPostId": 18
}
```


Fetch All Blog Posts

Retrieves all blog posts with optional filtering and sorting.

Endpoint: `/api/blog-post/fetch-all`
Method: GET
Authentication: None

Query Parameters:

- page (number): Page number for pagination
- limit (number): Number of items per page
- title (string): Filter by title
- description (string): Filter by description
- tag_ids (array): Filter by tag IDs
- code_template_ids (array): Filter by referenced code template IDs
- sorting (string): Sort options: "Most valued", "Most controversial", or null

Response:


```
{
   "sorting": "Most controversial",
   "totalPages": 1,
```

```
"totalCount": 1,
"data": [
  {
    "id": 15,
    "title": "Introduction to JavaScript",
    "description": "An introductory post on JavaScript basics.",
    "upvotes": 0,
    "downvotes": 0,
    "tags": [
      {
        "id": 39,
        "name": "JavaScript"
      }
    ],
    "codeTemplates": [
      {
        "id": 21,
        "title": "Basic JavaScript Loop"
      }
    ],
    "authorId": 32
  }
```

```
  ]
}
```

Change Blog Post Vote

Updates the vote count for a blog post.

Endpoint: `/api/blog-post/change-vote`
Method: PATCH
Authentication: Required

Request Body:

```
{
  "type": "UPVOTE",  // or "DOWNVOTE"
  "blogPostId": 15,
  "change": 1
}
```

Response:

```
{
  "id": 15,
  "title": "Introduction to JavaScript",
  "description": "An introductory post on JavaScript basics.",
  "upvotes": 1,
  "downvotes": 0
}
```

Note: Blog posts can be tagged and linked to code templates for better organization and reference. The voting system allows users to express their opinion on posts, and posts can be sorted by their vote counts. All operations except fetching require authentication.

Inappropriate Content Reports

Admin Sort Reports

Retrieves and sorts reported content for admin review.

Endpoint: `/api/admin/sort-reports`
Method: GET
Authentication: Required (Admin only)
Query Parameters:

- type (string, optional): Filter by "BLOG_POST" or "COMMENT"
- status (string, optional): Filter by "PENDING" or "RESOLVED"
- page (number, optional): Page number for pagination
- limit (number, optional): Number of items per page

Response:


```
{
  "results": [
    {
      "type": "COMMENT",
      "id": 1,
      "content": {
        "id": 1,
        "text": "Great post! Learned a lot.",
        "author": {
          "id": 9,
          "email": "user@example.com",
          "firstName": "User",
          "lastName": "User"
```

```
    },
    "blogPost": {
        "id": 1,
        "title": "Introduction to JavaScript"
    }
},
"reportCount": 1,
"reports": [
    {
        "id": 5,
        "reason": "Inappropriate content",
        "explanation": "This comment is irrelevant and spammy.",
        "status": "PENDING",
        "createdAt": "2024-11-02T20:16:12.315Z",
        "reporter": {
            "id": 8,
            "email": "admin@example.com",
            "firstName": "Admin",
            "lastName": "Admin"
        }
    }
],
```

```
            "isHidden": false
        }
    ],
    "pagination": {
        "total": 1,
        "pages": 1,
        "currentPage": 1,
        "limit": 10
    }
}
```


Admin Hide Content

Allows admins to hide or unhide reported content.

Endpoint: `/api/admin/hide-content`
Method: POST
Authentication: Required (Admin only)

Request Body:


```
{
    "type": "COMMENT",  // or "BLOG_POST"
    "contentId": 1,
    "hide": true
}
```


Response:


```
{
```

```
"success": true,
"message": "Content has been hidden",
"content": {
    "type": "COMMENT",
    "id": 1,
    "isHidden": true,
    "hiddenAt": "2024-11-02T20:25:51.024Z",
    "author": {
        "id": 9,
        "email": "user@example.com",
        "firstName": "User",
        "lastName": "User"
    },
    "blogPost": {
        "id": 1,
        "title": "Introduction to JavaScript"
    },
    "reportCount": 1,
    "updatedReports": {
        "count": 0
    }
}
```

}

Create New Report for Comment

Reports inappropriate comments.

Endpoint: `/api/report`
Method: POST
Authentication: Required

Request Body:


```
{
    "type": "COMMENT",
    "reason": "inappropriate",
    "explanation": "This comment is inappropriate",
    "commentId": 1
}
```


Response:


```
{
    "id": 6,
    "type": "COMMENT",
    "reason": "inappropriate",
    "explanation": "This comment is inappropriate",
    "status": "PENDING",
    "createdAt": "2024-11-02T20:28:16.471Z",
    "updatedAt": "2024-11-02T20:28:16.471Z"
}
```

Create New Report for Blog Post

Reports inappropriate blog posts.

Endpoint: `/api/report`
Method: POST
Authentication: Required

Request Body:


```
{
    "type": "BLOG_POST",
    "reason": "spam",
    "explanation": "This content is spam",
    "blogPostId": 1
}
```


Response:


```
{
    "id": 7,
    "type": "BLOG_POST",
    "reason": "spam",
    "explanation": "This content is spam",
    "status": "PENDING",
    "createdAt": "2024-11-02T20:29:21.993Z",
    "updatedAt": "2024-11-02T20:29:21.993Z"
}
```

Note: The reporting system allows users to flag inappropriate content for admin review. Admins can then review reported content and take appropriate action by hiding or unhiding the content. All reports maintain a status of either "PENDING" or "RESOLVED" to track their progress.

Comments

Create Comment

Creates a new comment on either a blog post or another comment.

Endpoint: `/api/comment/create`
Method: POST
Authentication: Required

Request Body:

```
{
    "toBlogPost": false,  // true for blog-post, false for comment
    "content": "I'm replying to comment with id 16",
    "blogPostId": 3,      // required if toBlogPost is true
    "parentCommentId": 16 // required if toBlogPost is false
}
```

Response:

```
{
    "id": 18,
    "content": "I'm replying to comment with id 16",
    "parentId": 16,
    "createdAt": "2024-11-02T22:41:39.104Z",
    "updatedAt": "2024-11-02T22:41:39.104Z"
}
```

Change Comment Vote

Updates the vote count for a comment.

Endpoint: `/api/comment/change-vote`
Method: PATCH
Authentication: Required

Request Body:

```
{
    "type": "DOWNVOTE",  // or "UPVOTE"
    "commentId": 17,
    "change": 1
}
```

Response:

```
{
    "id": 17,
    "content": "Thanks for sharing!",
    "upvotes": 0,
    "downvotes": 1
}
```

Fetch All Comments

Retrieves comments with optional filtering and sorting.

Endpoint: `/api/comment/fetch-all`
Method: GET
Authentication: None

Query Parameters:

- page (number): Page number for pagination
- limit (number): Number of items per page
- content (string): Filter by comment content
- authorId (number): Filter by author ID
- blogPostId (number): Filter by blog post ID
- parentId (number): Filter by parent comment ID
- sorting (string): Sort options: "Most valued", "Most controversial", or null

Response:


```
{
    "sorting": "Most controversial",
    "totalPages": 1,
    "totalCount": 1,
    "data": [
        {
            "id": 16,
            "content": "Great post! Learned a lot.",
            "upvotes": 0,
            "downvotes": 0,
            "authorId": 31,
            "blogPostId": 13,
            "parentId": null
        }
    ]
}
```

Tags

Create New Tag

Creates a new tag.

Endpoint: `/api/tag/create`
Method: POST
Authentication: Required

Request Body:


```
{

    "name": "algo"

}
```


Response:


```
{

    "id": 4,

    "name": "algo"

}
```


Fetch All Tags

Retrieves all available tags.

Endpoint: `/api/tag/fetch`
Method: GET
Authentication: Required

Response:


```
{

    "tags": [

        {
```

```
            "id": 2,
            "name": "JavaScript"
        },
        {
            "id": 3,
            "name": "Python"
        },
        {
            "id": 4,
            "name": "algo"
        },
        {
            "id": 5,
            "name": "ds"
        }
    ]
}
```


Avatars

Get Default Avatars

Retrieves the list of available default avatars.

Endpoint: `/api/avatars`
Method: GET
Authentication: None

Response:


```
{
   "data": [
      {
         "path": "/avatars/avatar1.jpg",
         "label": "Default Avatar 1",
         "alt": "A man avatar"
      },
      {
         "path": "/avatars/avatar2.png",
         "label": "Default Avatar 2",
         "alt": "A woman avatar"
      }
   ],
   "message": "Avatars retrieved successfully"
}
```


Note:

- Comments can be nested (replies to other comments) and can be voted on similar to blog posts
- Tags are used to categorize both blog posts and code templates
- Avatars are available for users to customize their profiles
- Most endpoints that modify data (create, update, vote) require authentication
- Pagination is available for comment listing to manage large datasets