Altona Gators Basketball Club Admin System

API Design Document (ADD)

Team AL-bluering

Version: 3.0
Last Modified: October 4 2023

"AgeGroup" Collection

This API documentation outlines the endpoints and operations available for interacting with the "ageGroups" collection in the MongoDB database. The collection stores information about ageGroup.

Base URL

The base URL for accessing the " ageGroups " collection API is: https://api.example.com/ageGroup

Authentication

The request heater contains the correct JWT token.

Endpoints

Get All AgeGroups

Retrieve a list of all ageGroups available in the "ageGroups" collection.
Endpoint: GET /
Response:

- Status Code: 200 OK
- Response Body: A JSON array containing objects representing classes. Each class object includes the following fields:
    - class_id (ObjectId): The internal unique identifier for the class.
    - name (String): The name of the class.

Sample request:

http://localhost:8080/ageGroup/

sample response:


```
[
    {
        "_id": "64e822e15828ed2a8bfccf00",
        "name": "age2",
        "coaches": [
            "64e822b75828ed2a8bfccef9",
            "64e822c45828ed2a8bfccefc"
        ],
        "modules": null,
        "createdAt": "2023-08-25T03:41:21.245Z",
        "updatedAt": "2023-08-25T03:41:21.245Z",
        "__v": 0
    },
    {
        "_id": "64e822f85828ed2a8bfccf03",
        "name": "age3",
        "coaches": [
            "64e822b75828ed2a8bfccef9",
            "64e822c45828ed2a8bfccefc"
        ],
        "modules": [
```

```
            "64e8223a8b133e132dab039f"
        ],
        "createdAt": "2023-08-25T03:41:44.123Z",
        "updatedAt": "2023-08-25T03:41:44.123Z",
        "__v": 0
    }
]
```

Get AgeGroups by ID

Retrieve details of a specific ageGroup by providing its ID.
Endpoint: GET /{ ageGroup_id}
Path Parameters:

- ageGroup_id (String): The unique identifier of the ageGroup.

Response:

- Status Code: 200 OK
- Response Body: An object representing the ageGroup with the following fields:
  - ageGroup_id (ObjectId): The internal unique identifier for the ageGroup.
  - name (String): The name of the ageGroup.
  - coach_id (Array of ObjectId): A list of references to coaches associated with the ageGroup.
  - module_id (Array of ObjectId): A list of references to modules associated with the ageGroup.

Sample Request:
http://localhost:8080/ageGroup/64e6e96a40c9d5e9af657ed9
Sample Response:

```
{
    "_id": "64e822f85828ed2a8bfccf03",
    "name": "age3",
    "coaches": [
        "64e822b75828ed2a8bfccef9",
        "64e822c45828ed2a8bfccefc"
    ],
    "modules": [
        "64e8223a8b133e132dab039f"
    ],
    "createdAt": "2023-08-25T03:41:44.123Z",
    "updatedAt": "2023-08-25T03:41:44.123Z",
    "__v": 0
}
```

Create a New AgeGroup

Create a new ageGroup by providing the necessary details.
Endpoint: POST /
Request Body:

- {
- "name": "String",
- "coaches": ["ObjectId"],
- "modules": ["ObjectId"]
- }

Response:

- Status Code: 201 Created
- Response Body: An object representing the newly created class with the same fields as mentioned in the "Get AgeGroup by ID" response.

Sample request:

POST http://localhost:8080/ageGroup/

Update AgeGroup

Update details of a specific agegroup by providing its ID.
Endpoint: PUT /{class_id}
Path Parameters:

- class_id (String): The unique identifier of the class.

Request Body:

- {
- "name": "String",
- "coach_id": ["ObjectId"],
- "module_id": ["ObjectId"]
- }

Response:

- Status Code: 200 OK
- Response Body: An object representing the updated class with the same fields as mentioned in the "Get AgeGroup by ID" response.

Delete AgeGroup

Delete a specific agegroup by providing its ID.
Endpoint: DELETE /{ ageGroup _id}
Path Parameters:

- ageGroup _id (String): The unique identifier of the class.

Response:

- Status Code: 204 No Content

Find AgeGroup

Find a specific agegroup by providing keywords
Endpoint: GET /{ keyword}
Path Parameters:

- keyword (String): The keyword used for searching the agegroup name.

Response:

- Status Code: 200 OK
- Response Body: A list of agegroup objects with coaches and modules populated.

Sample request:
http://localhost:8080/ageGroup/find/ag1

Sample response:


```
[
    {
        "_id": "651d3eb45b29b79ef6007ed9",
        "name": "age1",
        "coaches": [
            {
                "_id": "651d3e5e5b29b79ef6007ed3",
                "firstName": "sefseefsef",
                "middleName": "afwfvvsdvsev",
                "lastName": "ADWajcJOHN",
                "loginID": "awfdafda",
                "age": 33,
                "description": "",
                "createdAt": "2023-10-04T10:28:46.773Z",
                "updatedAt": "2023-10-04T10:28:46.773Z",
                "__v": 0
            }
        ],
        "modules": [
            {
                "_id": "651d3e905b29b79ef6007ed6",
                "name": "TestModule1",
                "level": 1,
                "materials": [],
                "createdAt": "2023-10-04T10:29:36.601Z",
                "updatedAt": "2023-10-04T10:29:36.601Z",
                "__v": 0
            }
        ],
        "createdAt": "2023-10-04T10:30:12.354Z",
        "updatedAt": "2023-10-04T10:30:12.354Z",
        "__v": 0
    }
]
```


Error Responses

- Status Code: 400 Bad Request Response Body: An error message indicating the invalid request.
- Status Code: 404 Not Found Response Body: An error message indicating the resource was not found.

- Status Code: 500 Internal Server Error Response Body: An error message indicating a server-side issue.

"Materials" Collection

This API documentation outlines the endpoints and operations available for interacting with the "Materials" collection in the database. The collection stores information about materials, including their names, content, and last modification timestamps.

Base URL

The base URL for accessing the "Materials" collection API is:
https://api.example.com/material

Authentication

The request heater contains the correct JWT token.

Endpoints

Get Material by ID

Retrieve details of a specific texture material by providing its ID.
Endpoint: GET /{material_id}
Path Parameters:

- material_id (String): The unique identifier of the texture material.

Response:

- Status Code: 200 OK
- Response Body: An object representing the material with the following fields:
  - id (ObjectId): The internal unique identifier for the texture material (collection primary key).
  - Material: The object of the material (fileMaterial/textureMaterial/Assessment).

Sample Request:
http://localhost:8080/material/
Sampel Response:


```
[
    {
        "_id": "64e844e4c1b57c03ae62a2bb",
        "type": "file",
        "fileMaterials": "64e844e4c1b57c03ae62a2b9",
        "createdAt": "2023-08-25T06:06:28.677Z",
        "updatedAt": "2023-08-25T06:06:28.677Z",
        "__v": 0
    },
    {
        "_id": "64e844e6c1b57c03ae62a2bf",
        "type": "texture",
        "textureMaterials": "64e844e6c1b57c03ae62a2bd",
        "createdAt": "2023-08-25T06:06:30.678Z",
        "updatedAt": "2023-08-25T06:06:30.678Z",
        "__v": 0
    },
    {
        "_id": "64e844e8c1b57c03ae62a2c7",
        "type": "assessment",
```

```
        "assessment": "64e844e8c1b57c03ae62a2c1",
        "createdAt": "2023-08-25T06:06:32.461Z",
        "updatedAt": "2023-08-25T06:06:32.461Z",
        "__v": 0
    }
]
```


Create a New Material

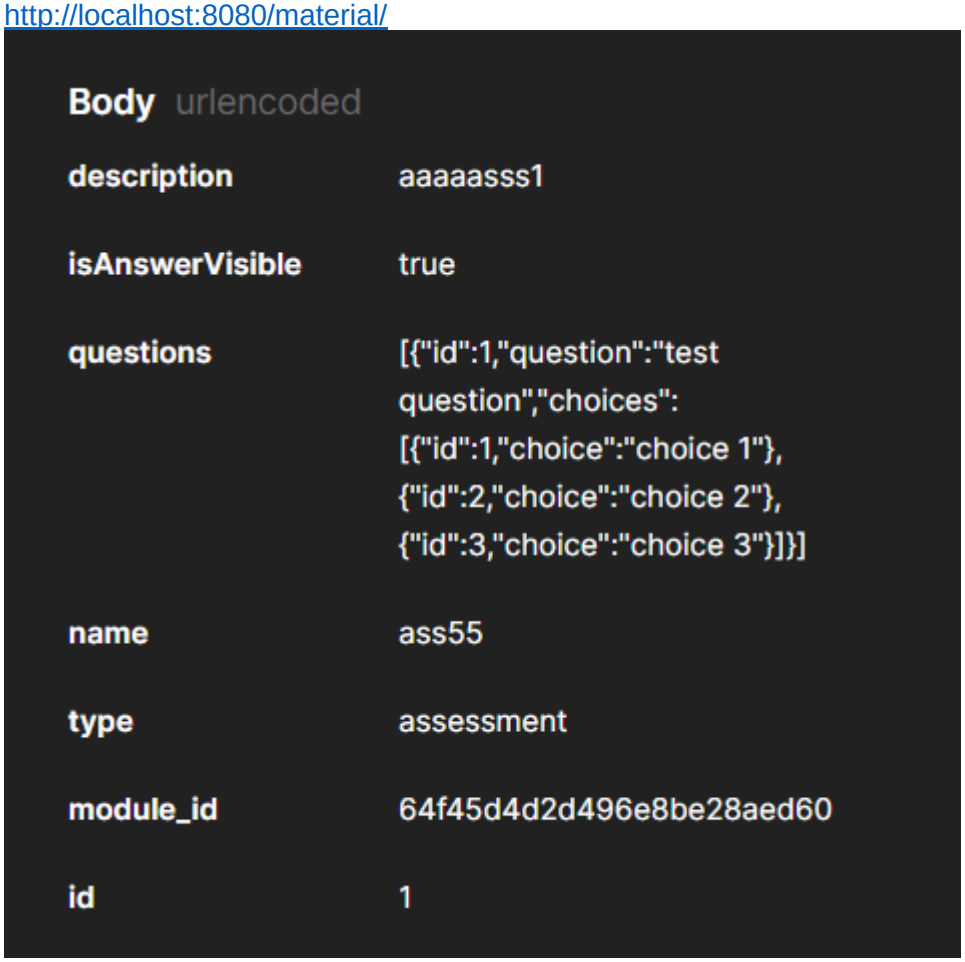
Create a new texture material by providing the necessary details. This operation will insert the reference of the new material to the module.materials list for the given module_id. Reorder the module's materials list, insert the new material into the index id.
Endpoint: POST /
Request Body:
{
type: string (file/texture/assessment),
name: string,
description: string,
url: string,
content: string,
questions: object,
isAnswerVisible: Boolean,
module_id: OID,
id: number
}

Response:

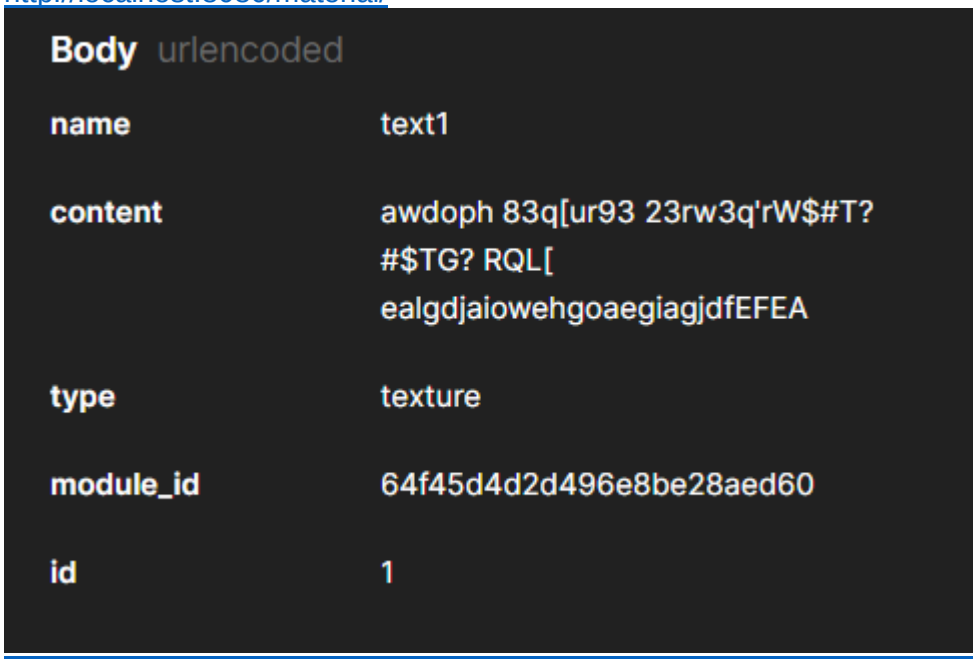
- Status Code: 201 Created
- Response Body: An object representing the newly created texture material with the same fields as mentioned in the "Get Material by ID" response.

Sampel Request for creating assessment:

http://localhost:8080/material/



Sampel Request for creating text:
http://localhost:8080/material/



Sampel Request for creating file:
http://localhost:8080/material/

Update Material

Update details of a specific material by providing its ID.
Endpoint: PUT /{material_id}
Path Parameters:

- material_id (String): The unique identifier of the texture material.

Request Body:
{
type: string (file/texture/assessment),
name: string,
description: string,
url: string,
content: string,
questions: object,
isAnswerVisible: boolean
}

Response:

- Status Code: 200 OK
- Response Body: An object representing the updated texture material with the same fields as mentioned in the "Get Texture Material by ID" response.

Delete Material

Delete a specific texture material by providing its ID. This operation will also delete all the references to the material within the modules.
Endpoint: DELETE /{material_id}
Path Parameters:

- material_id (String): The unique identifier of the texture material.

Response:

- Status Code: 204 No Content

Error Responses

- Status Code: 400 Bad Request Response Body: An error message indicating the invalid request.
- Status Code: 404 Not Found Response Body: An error message indicating the resource was not found.
- Status Code: 500 Internal Server Error Response Body: An error message indicating a server-side issue.

"Coaches" Collection

This API documentation provides details on how to interact with the "Coaches" collection in the database. The collection stores information about coaches, including their IDs, names, contact details, and login credentials.

Base URL

The base URL for accessing the "Coaches" collection API is:
https://api.example.com/coacher

Authentication

The request heater contains the correct JWT token.

Endpoints

Get All Coaches

Retrieve a list of all coaches available in the "Coaches" collection.
Endpoint: GET /
Response:

- Status Code: 200 OK
- Response Body: A JSON array containing objects representing coaches and ageGroups. Each coach object includes the following fields:
    - id (ObjectId): The internal unique identifier for the coach (collection primary key).
    - firstName (String): The first name of the coach.
    - middleName (String): The middle name of the coach.
    - lastName (String): The last name of the coach.
    - age (String): The age of the coach.

Sample Request:
http://localhost:8080/coacher/
Sample Response:


```
[
    {
        "coach": {
            "_id": "651d3e5e5b29b79ef6007ed3",
            "firstName": "sefseefsef",
            "middleName": "afwfvvsdvsev",
            "lastName": "ADWajcJOHN",
            "loginID": "awfdafda",
            "age": 33,
            "description": "",
            "createdAt": "2023-10-04T10:28:46.773Z",
            "updatedAt": "2023-10-04T10:28:46.773Z",
            "__v": 0
        },
        "ageGroup": [
            {
                "_id": "651d3eb45b29b79ef6007ed9",
                "name": "age1",
```

```
                "coaches": [
                    "651d3e5e5b29b79ef6007ed3"
                ],
                "modules": [
                    "651d3e905b29b79ef6007ed6"
                ],
                "createdAt": "2023-10-04T10:30:12.354Z",
                "updatedAt": "2023-10-04T10:30:12.354Z",
                "__v": 0
            }
        ]
    }
]
```


Get Coach by ID

Retrieve details of a specific coach by providing their ID.
Endpoint: GET /{coach_id}
Path Parameters:

- coach_id (String): The unique identifier of the coach.

Response:

- Status Code: 200 OK
- Response Body: An object representing the coach with the following fields:
    - id (ObjectId): The internal unique identifier for the coach (collection primary key).
    - firstName (String): The first name of the coach.
    - middleName (String): The middle name of the coach.
    - lastName (String): The last name of the coach.
    - age (int32): The age of the coach.
    - address (String): The address of the coach.
    - phone (String): The phone number of the coach (e.g., "+61 041234567").
    - loginID (String): The login ID of the coach.

Sample Response:


```
{
    "_id": "64e822c45828ed2a8bfccefc",
    "firstName": "3r13 r",
    "middleName": "middleName",
    "lastName": "lastName",
    "loginID": "awdawdawd",
    "password": "q3fqf",
    "description": "",
    "createdAt": "2023-08-25T03:40:52.067Z",
    "updatedAt": "2023-08-25T03:40:52.067Z",
    "__v": 0
}
```


Create a New Coach

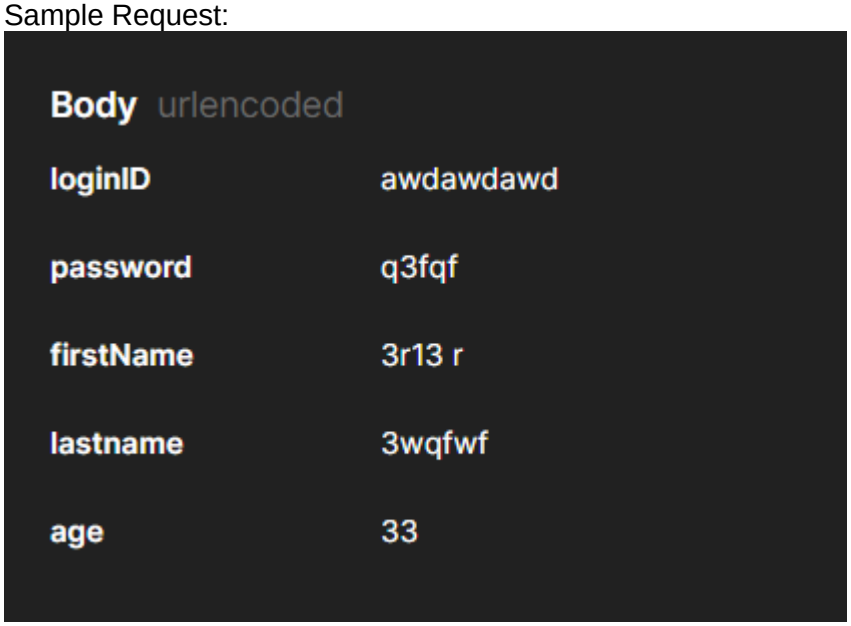
Create a new coach by providing the necessary details.
Endpoint: POST /
Request Body:

```
{
  "firstName": "String",
  "middleName": "String",
  "lastName": "String",
  "age": int,
  "address": "String",
  "phone": "String",
  "loginID": "String",
  "password": "String"
}
```

Response:

- Status Code: 201 Created
- Response Body: An object representing the newly created coach with the same fields as mentioned in the "Get Coach by ID" response.

Sample Request:



Update Coach

Update details of a specific coach by providing their ID.
Endpoint: PUT /{coach_id}
Path Parameters:

- coach_id (String): The unique identifier of the coach.

Request Body:

```
{
  "firstName": "String",
  "middleName": "String",
  "lastName": "String",
  "age": int,
  "address": "String",
  "phone": "String",
  "loginID": "String",
  "password": "String"
}
```

Response:

- Status Code: 200 OK

- Response Body: An object representing the updated coach with the same fields as mentioned in the "Get Coach by ID" response.

Delete Coach

Delete a specific coach by providing their ID.
Endpoint: DELETE /{coach_id}
Path Parameters:

- coach_id (String): The unique identifier of the coach.

Response:

- Status Code: 204 No Content

Find Coach

Find coach by their firstName/middleName/lastName
Endpoint: GET /find/{keyword}
Path Parameters:

- keyword (String): The keyword used for searching.

Response:

- Status Code: 200 OK
- Response Body: A list of objects, each object has coach and agegroup fields containing related data.

Sample request:

```
GET http://localhost:8080/coacher/find/john
```

Sample response:


```
[
    {
        "coach": {
            "_id": "651d3e5e5b29b79ef6007ed3",
            "firstName": "sefseefsef",
            "middleName": "afwfvvsdvsev",
            "lastName": "ADWajcJOHN",
            "loginID": "awfdafda",
            "age": 33,
            "description": "",
            "createdAt": "2023-10-04T10:28:46.773Z",
            "updatedAt": "2023-10-04T10:28:46.773Z",
            "__v": 0
        },
        "ageGroup": [
            {
                "_id": "651d3eb45b29b79ef6007ed9",
                "name": "age1",
                "coaches": [
                    "651d3e5e5b29b79ef6007ed3"
                ],
                "modules": [
                    "651d3e905b29b79ef6007ed6"
                ],
                "createdAt": "2023-10-04T10:30:12.354Z",
                "updatedAt": "2023-10-04T10:30:12.354Z",
                "__v": 0
```

```
                }
            ]
        }
]
```

Error Responses

- Status Code: 400 Bad Request Response Body: An error message indicating the invalid request.
- Status Code: 404 Not Found Response Body: An error message indicating the resource was not found.
- Status Code: 500 Internal Server Error Response Body: An error message indicating a server-side issue.

"Administrators" Collection

This API documentation outlines the endpoints and operations available for interacting with the "administrators" collection in the database. The collection stores information about administrators, including their IDs, names, login credentials, and other details.

Base URL

The base URL for accessing the "administrators" collection API is:
https://api.example.com/administrators

Authentication

To access the API endpoints, you must include proper authentication credentials in the request headers. Refer to your authentication mechanism documentation for details.

Endpoints

Get All Administrators

Retrieve a list of all administrators available in the "administrators" collection.
Endpoint: GET /
Response:

- Status Code: 200 OK
- Response Body: A JSON array containing objects representing administrators. Each administrator object includes the following fields:
    - admin_id (ObjectId): The internal unique identifier for the administrator (collection primary key).
    - firstName (String): The first name of the administrator.
    - middleName (String): The middle name of the administrator.
    - lastName (String): The last name of the administrator.
    - loginID (String): The login ID of the administrator.

Get Administrator by ID

Retrieve details of a specific administrator by providing their ID.
Endpoint: GET /{admin_id}
Path Parameters:

- admin_id (String): The unique identifier of the administrator.

Response:

- Status Code: 200 OK
- Response Body: An object representing the administrator with the following fields:
    - admin_id (ObjectId): The internal unique identifier for the administrator (collection primary key).
    - firstName (String): The first name of the administrator.
    - middleName (String): The middle name of the administrator.
    - lastName (String): The last name of the administrator.
    - loginID (String): The login ID of the administrator.
    - password (String): The encrypted login password of the administrator.

Create a New Administrator

Create a new administrator by providing the necessary details.

Endpoint: POST /
Request Body:

```
{
  "firstName": "String",
  "middleName": "String",
  "lastName": "String",
  "loginID": "String",
  "password": "String"
}
```

Response:

- Status Code: 201 Created
- Response Body: An object representing the newly created administrator with the same fields as mentioned in the "Get Administrator by ID" response.

Update Administrator

Update details of a specific administrator by providing their ID.
Endpoint: PUT /{admin_id}
Path Parameters:

- admin_id (String): The unique identifier of the administrator.

Request Body:

```
{
  "firstName": "String",
  "middleName": "String",
  "lastName": "String",
  "loginID": "String",
  "password": "String"
}
```

Response:

- Status Code: 200 OK
- Response Body: An object representing the updated administrator with the same fields as mentioned in the "Get Administrator by ID" response.

Delete Administrator

Delete a specific administrator by providing their ID.
Endpoint: DELETE /{admin_id}
Path Parameters:

- admin_id (String): The unique identifier of the administrator.

Response:

- Status Code: 204 No Content

Error Responses

- Status Code: 400 Bad Request Response Body: An error message indicating the invalid request.
- Status Code: 404 Not Found Response Body: An error message indicating the resource was not found.
- Status Code: 500 Internal Server Error Response Body: An error message indicating a server-side issue.

User Login

This API documentation outlines the endpoints and operations for user login and authentication.

Base URL

The base URL for the login API is: https://api.example.com/login

Endpoints

User Login

Authenticate a user by providing valid login credentials.
Endpoint: POST /
Request Body:
json
Copy code
```
{
  "username": "String",
  "password": "String"
}
```

Response:

- Status Code: 200 OK
- Response Body:

json
Copy code
```
{
  "token": "JWT_TOKEN"
}
```

- Status Code: 401 Unauthorized Response Body: An error message indicating invalid credentials.

Error Responses

- Status Code: 400 Bad Request Response Body: An error message indicating the invalid request.
- Status Code: 500 Internal Server Error Response Body: An error message indicating a server-side issue.

Logout

This API documentation outlines the "Logout" endpoint, which allows users to securely log out of the application.

Base URL

The base URL for accessing the "Logout" API is: https://api.example.com/logout

Authentication

To access the logout endpoint, the user must be authenticated and provide proper authentication credentials in the request headers. Refer to your authentication mechanism documentation for details.

Endpoint

Logout

Log out the currently authenticated user.
Endpoint: POST /
Response:

- Status Code: 204 No Content

Error Responses

- Status Code: 401 Unauthorized Response Body: An error message indicating that the user is not authenticated.
- Status Code: 500 Internal Server Error Response Body: An error message indicating a server-side issue.

Code Examples

```
const baseUrl = "https://api.example.com/logout";
// Perform logout
const headers = {
    "Authorization": "Bearer <access_token>"
};
fetch(baseUrl, {
    method: "POST",
    headers: headers
})
.then(response => {
    if (response.status === 204) {
        console.log("Logout successful");
    } else {
        console.log("Logout failed");
    }
});
```

"Modules" Collection

This API documentation outlines the endpoints and operations available for interacting with the "ageGroups" collection in the MongoDB database. The collection stores information about ageGroup.

Base URL

The base URL for accessing the " modules " collection API is: https://api.example.com/module

Authentication

The request heater contains the correct JWT token.

Endpoints

Get All modules

Retrieve a list of all modules available in the " modules " collection.
Endpoint: GET /
Response:

- Status Code: 200 OK
- Response Body: A JSON array containing objects representing classes. Each class object includes the following fields:
  - class_id (ObjectId): The internal unique identifier for the class.
  - name (String): The name of the class.

Sample request:

GET http://localhost:8080/module/

Sample response:


```
[
    {
        "_id": "64e8222a8b133e132dab039c",
        "name": "TestModule1",
        "father_id": "64e6e2d529fcb9a264a6eea3",
        "level": 3,
        "materials": null,
        "createdAt": "2023-08-25T03:38:18.503Z",
        "updatedAt": "2023-08-25T03:38:18.503Z",
        "__v": 0
    },
    {
        "_id": "64e8223a8b133e132dab039f",
        "name": "TestModule2",
        "father_id": "64e6e2d529fcb9a264a6eea3",
        "level": 3,
        "materials": [
            {
                "materialType": "fileMaterials",
                "id": 1,
                "material": "64e6e3b6618f1e63422b6779",
```

```
                "_id": "64e8223a8b133e132dab03a0"
            }
        ],
        "createdAt": "2023-08-25T03:38:34.180Z",
        "updatedAt": "2023-08-25T03:38:34.180Z",
        "__v": 0
    },
    {
        "_id": "64eab0c372b4342d977cc433",
        "name": "TestModule4",
        "level": 3,
        "materials": [
            {
                "materialType": "fileMaterials",
                "id": 1,
                "material": "64e6e3b6618f1e63422b6779",
                "_id": "64eab0c372b4342d977cc434"
            }
        ],
        "createdAt": "2023-08-27T02:11:15.091Z",
        "updatedAt": "2023-08-27T02:11:15.091Z",
        "__v": 0
    }
]
```


Get modules by ID

Retrieve details of a specific module by providing its ID.
Endpoint: GET /{ module_id}
Path Parameters:

- module_id (String): The unique identifier of the module.

Response:

- Status Code: 200 OK
- Response Body: An object representing the module with the following fields:
    - module_id (ObjectId): The internal unique identifier for the module.
    - name (String): The name of the module.
    - Level (Number): The level of the module.
    - father_id (ObjectId): A reference to the father module.
    - materials (Object): A list of objects containing the material type and reference id.

Sample Request:

http://localhost:8080/module/64e8222a8b133e132dab039c

Sample Response:


```
{
    "_id": "64e8222a8b133e132dab039c",
    "name": "TestModule1",
    "father_id": "64e6e2d529fcb9a264a6eea3",
    "level": 3,
    "materials": null,
    "createdAt": "2023-08-25T03:38:18.503Z",
    "updatedAt": "2023-08-25T03:38:18.503Z",
```

```
    "__v": 0
}
```


Create a New Module

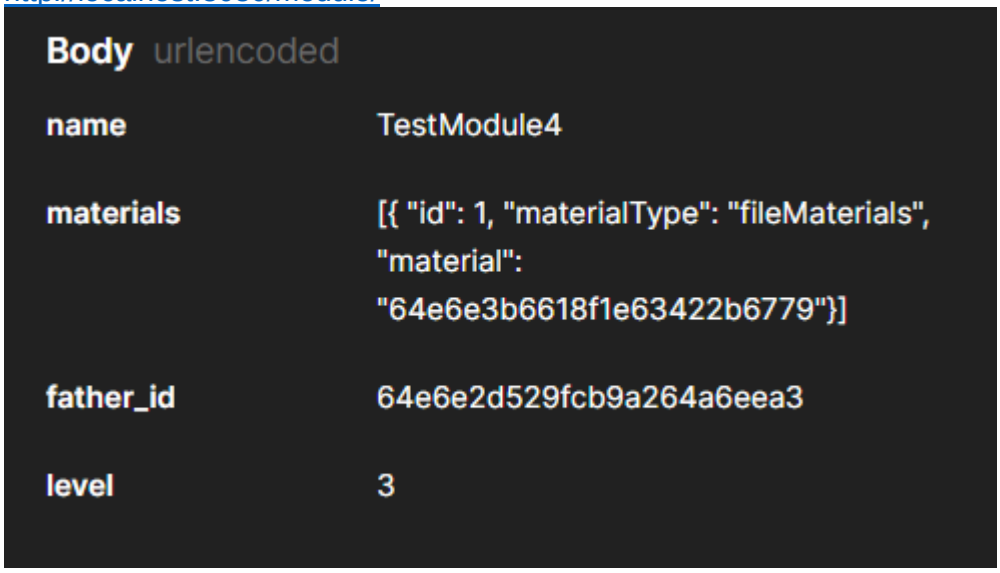
Create a new module by providing the necessary details.
Endpoint: POST /
Request Body:

- {
- "name": "String",
- "materials": ["Object"],
- "father_id": ObjectId,
- “level”: Number
- }

Response:

- Status Code: 201 Created
- Response Body: An object representing the newly created class with the same fields as mentioned in the "Get AgeGroup by ID" response.

Sample request:
http://localhost:8080/module/



Update Module

Update details of a specific class by providing its ID.
Endpoint: PUT /{module_id}
Path Parameters:

- module_id (String): The unique identifier of the module.

Request Body:

- {
- "name": "String",
- "materials": ["Object"],
- "father_id": ObjectId,
- “level”: Number

- }

Response:
- Status Code: 200 OK
- Response Body: An object representing the updated class with the same fields as mentioned in the "Get Module by ID" response.

Delete Module

Delete a specific module by providing its ID. This will also delete all related materials if the module have no submodules.
Endpoint: DELETE /{ module_id}
Path Parameters:
- module _id (String): The unique identifier of the module.

Response:
- Status Code: 204 No Content

Get sub modules by ID

Retrieve details of a specific module by providing its ID.
Endpoint: GET /subModule/{module_id}
Path Parameters:
- module_id (String): The unique identifier of the module.

Response:
- Status Code: 200 OK
- Response Body: A list of object representing the module with the following fields:
  - module_id (ObjectId): The internal unique identifier for the module.
  - name (String): The name of the module.
  - Level (Number): The level of the module.
  - father_id (ObjectId): A reference to the father module.
  - materials (Object): A list of objects containing the material type and reference id.

Sample Request:
http://localhost:8080/module/subModule/64e6e2d529fcb9a264a6eea3

Sample Response:


```
[
    {
        "_id": "64e8222a8b133e132dab039c",
        "name": "TestModule1",
        "father_id": "64e6e2d529fcb9a264a6eea3",
        "level": 3,
        "materials": null,
        "createdAt": "2023-08-25T03:38:18.503Z",
        "updatedAt": "2023-08-25T03:38:18.503Z",
        "__v": 0
    },
    {
        "_id": "64e8223a8b133e132dab039f",
        "name": "TestModule2",
        "father_id": "64e6e2d529fcb9a264a6eea3",
        "level": 3,
        "materials": [
```

```
            {
                "materialType": "fileMaterials",
                "id": 1,
                "material": "64e6e3b6618f1e63422b6779",
                "_id": "64e8223a8b133e132dab03a0"
            }
        ],
        "createdAt": "2023-08-25T03:38:34.180Z",
        "updatedAt": "2023-08-25T03:38:34.180Z",
        "__v": 0
    }
]
```


-

Get materials by module ID

Retrieve details of a specific module by providing its ID.
Endpoint: GET /materials/{module_id}
Path Parameters:

- module_id (String): The unique identifier of the module.

Response:

- Status Code: 200 OK
- Response Body: A list of object representing the materials with the following fields:
  - material (Object): the material object.
  - Id (Number): the order of the material.
  - Type (String): the type of the material.

Sample Request:
http://localhost:8080/module/materials/64eb04db8249dd2d01125d3d

Sample Response:


```
[
    {
        "type": "file",
        "id": 1,
        "material": {
            "_id": "64eb04548249dd2d01125d28",
            "url": "http://aaa.com",
            "description": "testfile",
            "name": "file1",
            "lastModified": "2023-08-27T08:07:37.361Z",
            "createdAt": "2023-08-27T08:07:48.295Z",
            "updatedAt": "2023-08-27T08:07:48.295Z",
            "__v": 0
        }
    },
    {
        "type": "file",
        "id": 2,
        "material": {
            "_id": "64eb04658249dd2d01125d38",
            "url": "http://bbb.com",
            "description": "testfile",
```

```
            "name": "file2",
            "lastModified": "2023-08-27T08:07:37.361Z",
            "createdAt": "2023-08-27T08:08:05.726Z",
            "updatedAt": "2023-08-27T08:08:05.726Z",
            "__v": 0
        }
    },
    {
        "type": "texture",
        "id": 3,
        "material": {
            "_id": "64eb04588249dd2d01125d2c",
            "content": "awdoph 83q[ur93 23rw3q'rW$#T?#$TG?\n RQL[\nealgdjaiowehgoae
giagjdfEFEA",
            "name": "text1",
            "lastModified": "2023-08-27T08:07:37.363Z",
            "createdAt": "2023-08-27T08:07:52.378Z",
            "updatedAt": "2023-08-27T08:07:52.378Z",
            "__v": 0
        }
    },
    {
        "type": "assessment",
        "id": 4,
        "material": {
            "_id": "64eb045b8249dd2d01125d30",
            "description": "aaaaasss1",
            "isAnswerVisible": true,
            "questions": [
                {
                    "id": 1,
                    "question": "test question",
                    "choices": [
                        {
                            "id": 1,
                            "choice": "choice 1",
                            "_id": "64eb045b8249dd2d01125d32"
                        },
                        {
                            "id": 2,
                            "choice": "choice 2",
                            "_id": "64eb045b8249dd2d01125d33"
                        },
                        {
                            "id": 3,
                            "choice": "choice 3",
                            "_id": "64eb045b8249dd2d01125d34"
                        }
                    ],
                    "_id": "64eb045b8249dd2d01125d31"
                }
```

```
            ],
            "name": "ass3",
            "lastModified": "2023-08-27T08:07:37.364Z",
            "createdAt": "2023-08-27T08:07:55.111Z",
            "updatedAt": "2023-08-27T08:07:55.111Z",
            "__v": 0
        }
    }
]
```


Error Responses

- Status Code: 400 Bad Request Response Body: An error message indicating the invalid request.
- Status Code: 404 Not Found Response Body: An error message indicating the resource was not found.

- Status Code: 500 Internal Server Error Response Body: An error message indicating a server-side issue.