SoftDesk - API

API-SoftDesk is a RESTful API that allows tracking and managing technical issues for BACK-END, FRONT-END, IOS, ANDROID projects. The application enables users to:

1. Create multiple projects.
2. Add users (contributors) to specific projects.
3. Create issues within projects.
4. Assign labels to these issues based on their priorities, tags, and more.

The API facilitates issue management and collaboration across different projects, making it easier to address technical challenges and improve software development and maintenance.

Users

This folder contains all the necessary endpoints required to efficiently manage user accounts. These endpoints facilitate various operations related to user accounts, such as user registration, authentication, profile updates, password changes, and more, making it a comprehensive
and essential resource for user management within the system.

POST Register

http://127.0.0.1:8000/api/register/

User Registration

Request

- **Method**: POST
- **Endpoint**: `http://127.0.0.1:8000/api/register/`
- **Headers**: (Add any required headers here)

Request Body

json

```
{
    "username": "Coddnrad",
    "email": "conrad@softdesk.com",
    "password": "P@ssword123",
    "date_of_birth": "1985-09-08",
    "can_be_contacted": true,
    "can_share_data": true
}
```

```
}
```

Response

- **Status Code**: 200 OK

Response Body

json

```
{
    "message": "User registration successful",
    "data": {
        "id": 23,
        "username": "Coddnrad",
        "email": "conrad@softdesk.com",
        "date_of_birth": "1985-09-08",
        "can_be_contacted": true,
        "can_share_data": true,
        "date_joined": "2023-10-10T12:26:22.650732Z"
    }
```

Description

This endpoint allows users to register by providing their registration information in the request body. The provided information must adhere to certain constraints, including uniqueness of `username` and `email`, password complexity, and accepted values for `can_be_contacted` and `can_share_data`.

Example

Here's an example request to register a new user:

http

```
POST http://127.0.0.1:8000/api/register/
Content-Type: application/json
{
    "username": "Coddnrad",
    "email": "conrad@softdesk.com",
    "password": "P@ssword123",
    "date_of_birth": "1985-09-08",
    "can_be_contacted": true,
    "can_share_data": true
}
```

Response

Upon successful registration, you will receive a response with a status code of 200 OK. The response will include a message confirming the registration and user profile details in the `data` field, including the user's `id`, `username`, `email`, `date_of_birth`, `can_be_contacted`, `can_share_data`, and `date_joined`.

Body raw (json)

json

```
{
    "username": "User1", // must be unique
    "email":"user1@softdesk.com", // must be unique
    "password": "P@ssword123", // password accept Numbers, special caracters
    "date_of_birth": "1985-09-08", // date of birth "YYY-MM-DD"
    "can_be_contacted": true, // accepted value : true, false
    "can_share_data": true  // accepted value : true, false
}
```

POST Login

http://127.0.0.1:8000/api/login/

User Login

Request

- **Method**: POST
- **Endpoint**: `http://127.0.0.1:8000/api/login/`
- **Headers**: (Add any required headers here)

Request Body

json

```
{
    "username": "Conrad",
    "password": "P@ssword123"
}
```

Response

- **Status Code**: 200 OK

Response Body

json

```
{
    "message": "Login successfully",
    "data": {
        "id": 21,
        "username": "Conrad",
        "tokens": {
            "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...", // Refresh token
            "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."   // Access token
        }
    }
}
```

Description

This endpoint allows users to log in by providing their username and password in the request body. Upon successful login, the server responds with access tokens and user information.

Example

Here's an example request to log in:

http

```
POST http://127.0.0.1:8000/api/login/
Content-Type: application/json
{
    "username": "Conrad",
    "password": "P@ssword123"
}
```

Response

Upon successful login, you will receive a response with a status code of 200 OK. The response will include a message confirming the login and provide access tokens (`access` and `refresh`) in the `data` field. You can use these tokens for authentication in subsequent requests.

Plain Text

```
javascript
```

Plain Text

```
// Parse the JSON response
var jsonData = pm.response.json();
// Set the access token as an environment variable
pm.environment.set("access_token", jsonData.data.tokens.access);
// Set the refresh token as an environment variable
pm.environment.set("refresh_token", jsonData.data.tokens.refresh);
pm.environment.set("authenticated_user", jsonData.data.id);
```

This script will set the `access_token`, `refresh_token`, and `authenticated_user` environment variables with the values obtained from the login response.

Body raw (json)

json

```
{
    "username": "Conrad",
    "password": "P@ssword123"
}
```

GET Profile

http://127.0.0.1:8000/api/user/profile/

Get User Profile

Request

- **Method**: GET
- **Endpoint**: `http://127.0.0.1:8000/api/user/profile/`
- **Headers**: (Add any required headers here)

Response

- **Status Code**: 200 OK

Response Body

json


```
{
    "id": 21,
    "username": "Conrad1",
    "email": "conrad@softdesk.com",
    "date_of_birth": "2001-04-04",
    "can_be_contacted": false,
    "can_share_data": false,
    "date_joined": "2023-10-06T13:37:30.754865Z"
}
```


Description

This endpoint allows you to retrieve the user's profile information.

Example

Here's an example request to get the user's profile:

http

```
GET http://127.0.0.1:8000/api/user/profile/
```

Response

Upon successfully making the GET request, you will receive a response with a status code of 200 OK. The response will include the user's profile details in the body, including the `id`, `username`, `email`, `date_of_birth`, `can_be_contacted`, `can_share_data`, and `date_joined`.

AUTHORIZATION Bearer Token

Token

DELETE Delete account

http://127.0.0.1:8000/api/user/profile/delete/

Delete User Profile

Request

- **Method**: DELETE
- **Endpoint**: `http://127.0.0.1:8000/api/user/profile/delete/`
- **Headers**: (Add any required headers here)

Response

- **Status Code**: 204 No Content

Response Body

No response body is expected for a successful deletion.

Description

This endpoint allows users to delete their own user profiles. Upon successful deletion, no content (204 No Content) is returned, and there is no response body.

Example

Here's an example request to delete a user's profile:

http

```
DELETE http://127.0.0.1:8000/api/user/profile/delete/
```

Response

Upon successfully deleting the user profile, you will receive a response with a status code of 204 No Content. There is no response body, as indicated by the status code.

AUTHORIZATION Bearer Token

Token

PATCH Update profile

http://127.0.0.1:8000/api/user/profile/update/

Update User Profile

Request

- **Method**: PATCH
- **Endpoint**: `http://127.0.0.1:8000/api/user/profile/update/`
- **Headers**: (Add any required headers here)

Request Body

```json
{
    "username": "Conrad1",
    "password": "P@ssword1234",
    "email": "conrad@softdesk.com",
    "date_of_birth": "2001-04-04",
    "can_be_contacted": false,
    "can_share_data": false
}
```

Response

- **Status Code**: 200 OK

Response Body

```json
{
    "message": "Profile updated successfully",
    "data": {
        "id": 21,
        "username": "Conrad1",
        "email": "conrad@softdesk.com",
        "date_of_birth": "2001-04-04",
        "can_be_contacted": false,
        "can_share_data": false,
        "date_joined": "2023-10-06T13:37:30.754865Z"
    }
```

Description

This endpoint allows you to update the user's profile information. You can provide the following information in the request body:

- `username`: The user's new username.
- `password`: The user's new password.
- `email`: The user's new email address.
- `date_of_birth`: The user's date of birth in the format "YYYY-MM-DD."
- `can_be_contacted`: A boolean value indicating whether the user can be contacted (true or false).
- `can_share_data`: A boolean value indicating whether the user can share data (true or false).

Example

Here's an example request to update the user's profile:

Here's an example request to update the user's profile:

http

```
PATCH http://127.0.0.1:8000/api/user/profile/update/
Content-Type: application/json
{
    "username": "Conrad1",
    "password": "P@ssword1234",
    "email": "conrad@softdesk.com",
    "date_of_birth": "2001-04-04",
    "can_be_contacted": false,
    "can_share_data": false
}
```

Response

Upon successfully updating the user's profile, you will receive a response with a status code of 200 OK. The response will include a message confirming the update and the updated user profile details in the `data` field, including the user's `id`, `username`, `email`, `date_of_birth`, `can_be_contacted`, `can_share_data`, and `date_joined`.

AUTHORIZATION Bearer Token

Token

Body raw (json)

json

```
{
    "date_of_birth":"2001-04-04"
}
```

POST Logout

http://127.0.0.1:8000/api/logout/

AUTHORIZATION Bearer Token

Token

Body raw (json)

json

```
{
    "token":""
}
```

Projects

These endpoints are designed to manage the projects associated with the connected user. In this system, a user can engage with projects in two primary roles: either as an author or as a contributor. These endpoints offer a comprehensive suite of functionalities for effectively managing these user-project interactions.

AUTHORIZATION Bearer Token

Token	<token>

POST Create a project

http://127.0.0.1:8000/api/projects/

Create a New Project

Request

- **Method**: POST
- **Endpoint**: `http://127.0.0.1:8000/api/projects/`
- **Headers**: (Add any required headers here)

Request Body

json

```
{
    "type": "BACK-END",
    "title": "My New Project",
    "description": "This is a description of my new project."
}
```

Response

- **Status Code**: 200 OK

Response Body

json

```json
{
    "message": "Project created.",
    "data": {
        "id": 41,
        "type": "BACK-END",
        "title": "My New Project",
        "description": "This is a description of my new project.",
        "created_time": "2023-10-10T15:07:31.654765Z",
        "contributors": [
```

Description

This endpoint allows you to create a new project. Provide the following information in the request body:

- `type`: Choose from available project types (BACK-END, FRONT-END, IOS, ANDROID).
- `title`: Provide a title for your project (maximum 50 characters).
- `description`: Provide a project description (maximum 500 characters).

Example

Here's an example request to create a new project:

```http
POST http://127.0.0.1:8000/api/projects/
Content-Type: application/json
{
    "type": "BACK-END",
    "title": "My New Project",
    "description": "This is a description of my new project."
}
```

Response

Upon successfully creating the project, you will receive a response with a status code of 200 OK. The response will include a message confirming the creation of the project, and the project details in the `data` field, including its unique `id`, `type`, `title`, `description`, `created_time`, `contributors`, and an empty list of `issues`.

This documentation outlines how to use the endpoint to create a new project and what to expect in the response. You can further enhance it by adding more details such as authentication requirements or error handling information, depending on your specific API implementation.

AUTHORIZATION Bearer Token

Token

Body raw (json)

```json
```

```
{
    "type": "BACK-END", // Choose from available project types: BACK-END, FRONT-END, IOS, A
    "title": "My New Project", // Provide a title for your project (maximum 50 characters)
    "description": "This is a description of my new project." // Provide a project descript
}
```

GET Projects List

http://127.0.0.1:8000/api/projects/

Retrieve List of Projects Associated with a User

Retrieve a list of projects that include (ID:21 - Conrad) as either the Author or Contributor.

Endpoint

- Base URL: http://127.0.0.1:8000/api
- Endpoint: `/projects/`

1. Search Projects

- **Description**: Search for projects by title and type.
- **Endpoint**: `/projects/search/`
- **Method**: GET

Query Parameters:

- `title` (String, optional): Filter projects by title.
- `type` (String, optional): Filter projects by type.

Example Request:

Plain Text

```
http
```

Plain Text

```
GET /projects/search/?title=MyProject&type=OpenSource
```

Response:

Plain Text

```
json
```

Plain Text

```
[
    {
        "id": 1,
        "title": "MyProject",
        "type": "OpenSource",
        "description": "A great open-source project."
    },
    {
        "id": 2,
        "title": "AnotherProject",
```


2. Get Projects list

Request

- HTTP Method: GET

Response

The response will be a JSON object representing a list of projects associated with the user:

json


```
{
    "message": "Listing projects that include (ID:21 - Conrad) as either the Author or Cont
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 40,
            "type": "BACK-END",
            "title": "Android App 1 Updated",
            "description": "This is a description of my new project.",
```


Usage

To retrieve a list of projects associated with a user (ID:21 - Conrad), send a GET request to the `/projects/` endpoint. The response will provide a list of projects, including their details such as ID, type, title, description, contributors, and associated issues.

AUTHORIZATION Bearer Token

Token

GET Project details

http://127.0.0.1:8000/api/projects/1/

Retrieve Project Details

Retrieve details of a project, including its title, description, contributors, and associated issues.

Endpoint

- Base URL: http://127.0.0.1:8000/api
- Endpoint: `/projects/31/`

Request

- HTTP Method: GET

Response

The response will be a JSON object representing the project details:

json


```
{
    "id": 31,
    "type": "BACK-END",
    "title": "My New Project",
    "description": "This is a description of my new project.",
    "created_time": "2023-10-09T12:10:30.625218Z",
    "contributors": [
        {
            "id": 88,
            "role": "AUTHOR",
            "user_details": {
```


Usage

To retrieve details of a project, send a GET request to the `/projects/31/` endpoint. The response will provide information about the project, including its title, description, contributors, and associated issues.

AUTHORIZATION Bearer Token

Token

PATCH Update project

http://127.0.0.1:8000/api/projects/1/

Update Project Details

Update the details of a project, such as its title and description. Only the project's author can perform this action.

Endpoint

- Base URL: http://127.0.0.1:8000/api

- Endpoint: `/projects/40/`

Request

- HTTP Method: PATCH

Response

The response will be a JSON object representing the updated project details:

json


```
{
    "id": 40,
    "type": "BACK-END",
    "title": "Android App 1 Updated",
    "description": "This is a description of my new project.",
    "created_time": "2023-10-09T19:57:44.340599Z",
    "contributors": [
        {
            "id": 128,
            "role": "AUTHOR",
            "user_details": {
```


Usage

To update the details of a project, send a PATCH request to the `/projects/40/` endpoint. Only the project's author can perform this action. The response will provide the updated project details, including its title, description, and contributors.

AUTHORIZATION Bearer Token

Token

Body raw (json)

json


```
{
        "title": "Android App 1 Updated"
}
```

DELETE Delete project

http://127.0.0.1:8000/api/projects/38/

Delete Project

Description

This endpoint allows the author of a specific project to delete it by its ID. Deleting a project will also remove all associated data, including issues, comments, and contributors.

Endpoint

Base URL: http://127.0.0.1:8000/api
Endpoint: /projects/38/

Request

- HTTP Method: DELETE

Response

- HTTP Status Code: 204 No Content (indicating successful deletion with no response body)

Permissions

Only the author of the project is allowed to delete it.

Example

Request

plaintext

```
DELETE http://127.0.0.1:8000/api/projects/38/
```

Response

json

```
HTTP/1.1 204 No Content
```

Usage

To delete a project, send a DELETE request to the endpoint with the project's ID in the URL. Only the author of the project is authorized to perform this action, and it will remove the project and all associated data from the system.

AUTHORIZATION Bearer Token

Token

Contributors

GET Contributors List

http://127.0.0.1:8000/api/projects/1/contributors/

Contributor Search and Filter

Description

This feature allows you to search and filter contributors for a specific project.

Request

- **Endpoint:** `http://127.0.0.1:8000/api/projects/2/contributors/`
- **Method:** GET

Query Parameters

1. **user (Search by User)**
   a. **Type:** String
   b. **Description:** Search for contributors by the username of the user associated with the contributor.
   c. **Example:** `user=JohnDoe`
2. **project (Search by Project)**
   a. **Type:** String
   b. **Description:** Search for contributors by the title of the project they are associated with.
   c. **Example:** `project=Sample Project`

Example Request

Plain Text

```
http
```

Plain Text

```
GET http://127.0.0.1:8000/api/projects/2/contributors/?user=JohnDoe&project=Sample Project
```

Response

- **Status Code:** 200 OK

Response Body

Plain Text

```
json
```

Plain Text

```
{
    "count": 2,
    "next": null,
    "previous": null
```

```
    "previous": null,
    "results": [
        {
            "id": 1,
            "user": "JohnDoe",
            "project": "Sample Project",
            "role": "Contributor"
        },
```

Notes

- Use the `user` parameter to search for contributors by the username of the associated user.
- Use the `project` parameter to search for contributors by the title of the associated project.

You can use both parameters in the same request to narrow down the results.List Contributors for Project

And without using search and filter you, this API endpoint allows you to retrieve a list of contributors for a specific project.

Request:

- Request Method: GET

Response:

- Response Status: 200 OK
- Response Body:

{
"message": "Listing contributors for the project (ID: {project_id} - {project_title}).",
"count": {number_of_contributors},
"next": null,
"previous": null,
"results": [
{
"id": {contributor_id},
"role": "{contributor_role}",
"user": {
"id": {user_id},
"username": "{username}"
}
},
{
"id": {contributor_id},
"role": "{contributor_role}",
"user": {
"id": {user_id},
"username": "{username}"
}
},
...
]
}

- The `message` field indicates that contributors for the specific project (ID: {project_id} - {project_title}) are being

listed.

- The `count` field specifies the total number of contributors for the project.
- The `results` field contains an array of contributor details, where each contributor has an `id`, `role`, and `user` object with `id` and `username`.

Replace `{project_id}`, `{project_title}`, `{number_of_contributors}` `{contributor_id}`, `{contributor_role}` `{user_id}`,and `{username}` with actual values when making requests.

AUTHORIZATION Bearer Token

Token

POST Add Contributors to a project

http://127.0.0.1:8000/api/projects/1/contributors/2/

Add Contributor to Project

Endpoint: `/projects/{project_id}/contributors/{user_id}`

Description:

This API endpoint allows you to add a contributor to the project with the specified `project_id` using the HTTP POST method. You can provide the necessary data to create a new contributor for the project.

Request:

- Request Method: POST

Request Body:

```
{
"message": "Contributor created.",
"data": {
"id": {contributor_id},
"role": "CONTRIBUTOR",
"user": {
"id": {user_id},
"username": "{username}"
}
}
}
```

- The `message` field confirms the successful addition of a contributor to the project with ID `{project_id}`.
- The `data` field contains contributor information, including their `id` `role`,and `user` object with `id` and `sername`.
- The role is automatically set to "CONTRIBUTOR" for the new contributor.

This response provides details about the newly added contributor, including their ID,username, and role.

AUTHORIZATION Bearer Token

Token

DELETE Delete a contriibutor from a project

http://127.0.0.1:8000/api/projects/40/contributors/129/

Delete Contributor from Project

Endpoint: `/projects/{project_id}/contributors/{contributor_id}/`

Description:

This API endpoint allows you to remove a contributor with the specified `contributor_id` from the project with the specified `project_id`. This action will permanently remove all data associated with the contributor as a project contributor.

Request:

- Request Method: DELETE

Response:

Response Status: 204 No Content

- Response Body:

{
"message": "Contributor deleted."
}

The `message` field confirms the successful removal of the contributor from the project.

This response indicates that the contributor has been successfully removed from the project, and all their data as a project contributor has been deleted.

Response 403 Forbiden

{ "detail": "UPDATE and DELETE actions are restricted to the author only"}

AUTHORIZATION Bearer Token

Token

Issues

GET Issues project List

http://127.0.0.1:8000/api/projects/1/issues/

Issue Search and Filter

Description

This feature allows you to search and filter issues for a specific project.

Request

- **Endpoint:** `http://127.0.0.1:8000/api/projects/2/issues/`
- **Method:** GET

Query Parameters

1. **tag (Search by Tag)**
   a. **Type:** String
   b. **Description:** Search for issues with a specific tag.
   c. **Example:** `tag=bug`
2. **status (Search by Status)**
   a. **Type:** String
   b. **Description:** Search for issues with a specific status.
   c. **Example:** `status=in progress`
3. **priority (Search by Priority)**
   a. **Type:** String
   b. **Description:** Search for issues with a specific priority.
   c. **Example:** `priority=high`
4. **title (Search by Title)**
   a. **Type:** String
   b. **Description:** Search for issues with a specific title.
   c. **Example:** `title=authentication`
5. **project (Search by Project)**
   a. **Type:** String
   b. **Description:** Search for issues within a specific project by the project's title.
   c. **Example:** `project=Sample Project`
6. **author (Search by Author)**
   a. **Type:** String
   b. **Description:** Search for issues created by a specific author (user's username).
   c. **Example:** `author=JohnDoe`
7. **assigned (Search by Assigned User)**
   a. **Type:** String
   b. **Description:** Search for issues assigned to a specific user (user's username).
   c. **Example:** `assigned=JaneSmith`

Example Request

Plain Text

```
http
```

Plain Text

```
GET http://127.0.0.1:8000/api/projects/2/issues/?tag=bug&status=in progress&project=Sample
```

Response

- **Status Code:** 200 OK

Response Body

Plain Text

```
json
```

Plain Text


```
{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "tag": "bug",
            "status": "in progress",
            "priority": "high",
            "title": "Authentication Issue",
```


Notes

- Use the query parameters to search and filter issues based on different criteria.
- You can use multiple query parameters to refine your search.

Retrieve List of Projects Associated with a User

Description

This endpoint allows you to retrieve a list of all projects associated with a user. A user can have two roles in a project, either as an "author" or as a "contributor." The response will consist of a list of dictionaries with detailed information about each project, including its ID, title, description, the author, and contributors in that project.

Endpoint

- **Base URL:** http://127.0.0.1:8000/api

- **Base URL:** http://127.0.0.1:8000/api
- **Endpoint:** `/projects/40/issues/`

Request

- **HTTP Method:** GET

Response

The response will be a list of dictionaries, where each dictionary represents a project associated with the user. The structure of each dictionary is as follows:

```json
[
    {
        "id": 1,                        // Project ID
        "type": "BACK-END",             // Project Type
        "title": "Project 1",          // Project Title
        "description": "Description", // Project Description
        "author": {
            "id": 1001,                 // Author ID
            "username": "Author1"       // Author Username
        },
        "contributors": [
```

Example

Request

```http
GET http://127.0.0.1:8000/api/projects/40/issues/
```

Response

```json
HTTP/1.1 200 OK
[
    {
        "id": 1,
        "type": "BACK-END",
        "title": "Project 1",
        "description": "Description",
        "author": {
            "id": 1001,
            "username": "Author1"
        },
```

Usage

To retrieve a list of all projects associated with a user, send a GET request to the `/projects/40/issues/` endpoint.

The response will provide detailed information about each project, including its ID, title, description, the author, and contributors in that project.

AUTHORIZATION Bearer Token

Token

GET Retrieve issue details

http://127.0.0.1:8000/api/projects/5/issues/4/

Retrieve Specific Project Issue

Description

This endpoint allows you to retrieve a specific issue associated with a project. The response will provide detailed information about the selected issue, including its ID, tag, status, priority, title, description, project details, author details, assigned user details, and comments associated with the issue.

Endpoint

- **Base URL:** http://127.0.0.1:8000/api
- **Endpoint:** `/projects/5/issues/4/`

Request

- **HTTP Method:** GET

Response

The response will be a dictionary representing the selected issue with the following structure:

json

```
{
    "id": 1,                              // Issue ID
    "tag": "BUG",                        // Issue Tag
    "status": "TODO",                    // Issue Status
    "priority": "HIGH",                  // Issue Priority
    "title": "Issue Title",              // Issue Title
    "description": "Issue Description",  // Issue Description
    "project": {
        "id": 5,                          // Project ID
        "type": "BACK-END",               // Project Type
        "title": "Project 1",            // Project Title
```

Example

Request

```http
GET http://127.0.0.1:8000/api/projects/5/issues/4/
```

Response

```json
HTTP/1.1 200 OK
{
    "id": 1,
    "tag": "BUG",
    "status": "TODO",
    "priority": "HIGH",
    "title": "Issue Title",
    "description": "Issue Description",
    "project": {
        "id": 5,
        "type": "BACK-END",
```

Usage

To retrieve a specific issue associated with a project, send a GET request to the `/projects/5/issues/4/` endpoint. The response will provide detailed information about the selected issue, including its ID, tag, status, priority, title, description, project details, author details, assigned user details, and comments associated with the issue.

AUTHORIZATION Bearer Token

Token

POST Add an issue

http://127.0.0.1:8000/api/projects/40/issues/

Create a New Issue in a Project

Description

This endpoint allows you to create a new issue within a specific project. You can provide the necessary details such as the issue tag, status, priority, title, description, and the assigned user. Upon successful creation, the response will confirm the new issue's details.

Endpoint

- **Base URL:** http://127.0.0.1:8000/api
- **Endpoint:** `/projects/40/issues/`

- **Endpoint:** `/projects/40/issues/`

Request

- **HTTP Method:** POST

Request Body

The request body should be a JSON object containing the following parameters:

- `tag` (string, required): The tag associated with the issue (e.g., "BUG", "TASK", "FEATURE").
- `status` (string, required): The status of the issue (e.g., "TODO", "INPROGRESS", "FINISHED").
- `priority` (string, required): The priority of the issue (e.g., "LOW", "MEDIUM", "HIGH").
- `title` (string, required): The title of the issue.
- `description` (string, required): The description of the issue.
- `assigned` (integer, optional): The ID of the user assigned to the issue.

Example Request Body:

json

```json
{
    "tag": "BUG",
    "status": "TODO",
    "priority": "HIGH",
    "title": "New Issue Title",
    "description": "Description of the new issue",
    "assigned": 123
}
```

Response

Upon successful creation, the response will include the details of the newly created issue:

json

```json
{
    "id": 1,                              // Issue ID
    "tag": "BUG",                        // Issue Tag
    "status": "TODO",                    // Issue Status
    "priority": "HIGH",                  // Issue Priority
    "title": "New Issue Title",          // Issue Title
    "description": "Description of the new issue",  // Issue Description
    "project": {
        "id": 40,                         // Project ID
        "type": "BACK-END",               // Project Type
        "title": "Project 1",            // Project Title
```

Example

Request

http

```
POST http://127.0.0.1:8000/api/projects/40/issues/
Content-Type: application/json
{
    "tag": "BUG",
    "status": "TODO",
    "priority": "HIGH",
    "title": "New Issue Title",
    "description": "Description of the new issue",
    "assigned": 123
}
```

Response

json

```
HTTP/1.1 201 Created
{
    "id": 1,
    "tag": "BUG",
    "status": "TODO",
    "priority": "HIGH",
    "title": "New Issue Title",
    "description": "Description of the new issue",
    "project": {
        "id": 40,
        "type": "BACK-END",
```

Usage

To create a new issue within a specific project, send a POST request to the `/projects/40/issues/` endpoint with the required parameters in the request body. The response will confirm the details of the newly created issue, including its ID, tag, status, priority, title, description, project details, author details, assigned user details, and an empty list of comments (initially).

AUTHORIZATION Bearer Token

Token

Body raw (json)

json

```
{
        "tag": "BUG",       //  BUG, TASK, FEATURE
        "status": "TODO",   // TODO, INPROGRESS, FINISHED
        "priority": "LOW", // LOW , MEDIUM, HIGH
        "title": "fisrt issue",
        "description":"description of the first issue",
        "assigned":12   // User ID
    }
```

PATCH Update an issue

http://127.0.0.1:8000/api/projects/40/issues/7/

Update an Issue in a Project

Description

This endpoint allows you to update an existing issue within a specific project. You can modify the issue's tag, status, priority, title, description, and the assigned user. Upon successful update, the response will confirm the updated issue's details.

Endpoint

- **Base URL:** http://127.0.0.1:8000/api
- **Endpoint:** `/projects/40/issues/7/`

Request

- **HTTP Method:** PATCH

Request Body

The request body should be a JSON object containing the following parameters:

- `tag` (string, optional): The updated tag associated with the issue (e.g., "BUG", "TASK", "FEATURE").
- `status` (string, optional): The updated status of the issue (e.g., "TODO", "INPROGRESS", "FINISHED").
- `priority` (string, optional): The updated priority of the issue (e.g., "LOW", "MEDIUM", "HIGH").
- `title` (string, optional): The updated title of the issue.
- `description` (string, optional): The updated description of the issue.
- `assigned` (integer, optional): The ID of the updated user assigned to the issue.

Example Request Body (To Update Title and Description):

json

```
{
    "title": "Updated Issue Title",
    "description": "Updated description of the issue"
}
```

Response

Upon successful update, the response will include the details of the updated issue:

json

```
{
```

```
    "id": 7,                              // Issue ID
    "tag": "BUG",                         // Updated Issue Tag
    "status": "TODO",                     // Updated Issue Status

    "priority": "HIGH",                   // Updated Issue Priority
    "title": "Updated Issue Title",       // Updated Issue Title
    "description": "Updated description of the issue",  // Updated Issue Description
    "project": {
        "id": 40,                         // Project ID
        "type": "BACK-END",               // Project Type
```

Example

Request

http

```
PATCH http://127.0.0.1:8000/api/projects/40/issues/7/
Content-Type: application/json
{
    "title": "Updated Issue Title",
    "description": "Updated description of the issue"
}
```

Response

json

```
HTTP/1.1 200 OK
{
    "id": 7,
    "tag": "BUG",
    "status": "TODO",
    "priority": "HIGH",
    "title": "Updated Issue Title",
    "description": "Updated description of the issue",
    "project": {
        "id": 40,
        "type": "BACK-END",
```

Usage

To update an existing issue within a specific project, send a PATCH request to the `/projects/40/issues/7/` endpoint with the desired changes in the request body. The response will confirm the details of the updated issue, including its ID, updated tag, status, priority, title, description, project details, author details, assigned user details, and an unchanged list of comments.

AUTHORIZATION Bearer Token

Token

Body raw (json)

json

```
{
        "status": "INPROGRESS"}
```

DELETE Delete an issue from a project

http://127.0.0.1:8000/api/projects/40/issues/6/

Delete an Issue in a Project

Description

This endpoint allows you to delete an existing issue within a specific project. Upon successful deletion, the response will return a status code of 204 (No Content) to indicate the removal of the issue.

Endpoint

- **Base URL:** http://127.0.0.1:8000/api
- **Endpoint:** `/projects/40/issues/6/`

Request

- **HTTP Method:** DELETE

Response

Upon successful deletion, the response will return a status code of 204 (No Content) to indicate the removal of the issue.

Example

Request

http

```
DELETE http://127.0.0.1:8000/api/projects/40/issues/6/
```

Response

http

```
HTTP/1.1 204 No Content
```

Usage

To delete an existing issue within a specific project, send a DELETE request to the `/projects/40/issues/6/` endpoint. Upon successful deletion, the response will return a status code of 204 (No Content) to indicate the removal of the issue.

AUTHORIZATION Bearer Token

Token

Comments

GET Comments Issue List

http://127.0.0.1:8000/api/projects/1/issues/1/comments/

Comment Search and Filter

Description

This feature allows you to search and filter comments for a specific project's issues.

Request

- **Endpoint:** `http://127.0.0.1:8000/api/projects/2/comments/`
- **Method:** GET

Query Parameters

1. **unique_id (Search by Comment ID)**
   a. **Type:** String
   b. **Description:** Search for comments with a specific unique ID.
   c. **Example:** `unique_id=12345`
2. **description (Search by Description)**
   a. **Type:** String
   b. **Description:** Search for comments with a specific description.
   c. **Example:** `description=feedback`
3. **issue (Search by Issue Title)**
   a. **Type:** String
   b. **Description:** Search for comments related to issues with a specific title.
   c. **Example:** `issue=Authentication Issue`
4. **author (Search by Author)**
   a. **Type:** String
   b. **Description:** Search for comments authored by a specific user (user's username).
   c. **Example:** `author=JohnDoe`

Example Request

Plain Text

```
http
```

Plain Text

```
GET http://127.0.0.1:8000/api/projects/2/comments/?unique_id=12345&description=feedback
```

Response

- **Status Code:** 200 OK

Response Body

Plain Text

```
json
```

Plain Text

```
{
    "message": "Listing comments included in the Authentication Project",
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "unique_id": "12345",
            "description": "User feedback on authentication",
            "issue": "Authentication Issue",
            "author": "JohnDoe"
```

Notes

- Use the query parameters to search and filter comments based on different criteria.
- You can use multiple query parameters to refine your search.

Get Comments for an Issue

Description

This endpoint allows you to retrieve a list of comments associated with a specific issue in a project. The response will provide details about the comments, including their unique IDs, descriptions, and author details.

Endpoint

- **Base URL:** http://127.0.0.1:8000/api

- **Endpoint:** `/projects/40/issues/7/comments/`
- **HTTP Method:** GET

Response

The response will include a message indicating the issue and project related to the comments, the total count of comments, and the list of comments themselves.

Response Body

json


```
{
  "message": "Listing comments included in the (ID:40 - BACK) project",
  "count": 6,
  "next": null,
  "previous": null,
  "results": [
    {
      "unique_id": "8c66b8cf-06c7-4968-986c-cc19a9811723",
      "description": "Testing comment",
      "author_details": {
        "id": 21,
```


The response message will include the issue's unique ID, the issue's title, the project's unique ID, and the project's type. It will also include the total count of comments related to the issue and a list of comment objects with their unique IDs, descriptions, and author details.

This format provides detailed information about each comment, allowing for efficient management and understanding of the comments associated with the issue.

AUTHORIZATION Bearer Token

Token

POST Add a comment

http://127.0.0.1:8000/api/projects/1/issues/2/comments/

Create Comment

Description

This endpoint allows you to create a new comment associated with a specific issue in a project. Provide the required data in the request body to create the comment.

Endpoint

Base URL: http://127.0.0.1:8000/api
Endpoint: /projects/40/issues/8/comments/

Request

- HTTP Method: POST

Request Body

Provide the following data in the request body in JSON format:

- `description` (string, required): The description of the comment.

Example

Request

```plaintext
POST http://127.0.0.1:8000/api/projects/40/issues/8/comments/
Content-Type: application/json
{
    "description": "This is a test comment."
}
```

Response

```json
HTTP/1.1 201 Created
{
    "message": "Comment created.",
    "data": {
        "unique_id": "fa365314-e028-4889-a501-e8f2a2c453b9",
        "description": "This is a test comment.",
        "author_details": {
            "id": 21,
            "username": "Conrad"
        }
    }
```

Response

- HTTP Status Code: 201 Created (indicating successful creation)
- JSON response containing the unique identifier (UUID), description, and author details of the newly created comment.

Usage

To create a new comment associated with an issue in a project, send a POST request to the endpoint with the required data in the request body. The response will provide information about the newly created comment, including its unique identifier, description, and author details.

AUTHORIZATION Bearer Token

Token

Body raw (json)

```json
{
    "description": "Testing comment "
}
```

PATCH Update a comment

http://127.0.0.1:8000/api/projects/1/issues/2/comments/e58cd61f-6247-42b6-8e99-6fb439afcf8a/

Endpoint: `http://127.0.0.1:8000/api/projects/40/issues/8/comments/fa365314-e028-4889-a501-e8f2a2c453b9/`

Description: Update a comment associated with a specific issue.

HTTP Method: PATCH

Request Headers:

- Content-Type: application/json
- Authorization: Bearer {{your_access_token}}

Request Body (example JSON data):

```json
{
    "description": "Updated comment text"
}
```

Response:

- **Status Code**: 200 OK - The comment was successfully updated.
- **Status Code**: 400 Bad Request - If the request body is invalid or missing required fields.
- **Status Code**: 404 Not Found - If the comment or issue is not found.
- **Status Code**: 403 Forbidden - If the user does not have permission to update the comment.
- **Status Code**: 401 Unauthorized - If the user is not authenticated.

Response Body (example JSON data):

```json
{
    "message": "Comment updated successfully",
    "data": {
        "unique_id": "fa365314-e028-4889-a501-e8f2a2c453b9",
        "description": "Updated comment text",
```

```
        "author_details": {
            "id": 21,
            "username": "Conrad"
        }
    }
```

Usage: This endpoint allows you to update the description of a specific comment associated with an issue. Provide the updated comment text in the request body as JSON data. The `unique_id` in the URL identifies the specific comment to be updated. The response will confirm the successful update with the updated comment data.

AUTHORIZATION Bearer Token

Token

Body raw (json)

json

```
{
    "description": "2002-02-01"
}
```

DELETE Retrieve comment

http://127.0.0.1:8000/api/projects/1/issues/2/comments/e58cd61f-6247-42b6-8e99-6fb439afcf8a/

Delete Comment

Description

This endpoint allows you to delete a specific comment associated with an issue in a project. Provide the unique identifier (UUID) of the comment to be deleted in the URL.

Endpoint

Base URL: http://127.0.0.1:8000/api
Endpoint: /projects/40/issues/8/comments/fa365314-e028-4889-a501-e8f2a2c453b9/

Request

- HTTP Method: DELETE

Response

- HTTP Status Code: 204 No Content (indicating successful deletion)

Example

Request

plaintext

```
DELETE http://127.0.0.1:8000/api/projects/40/issues/8/comments/fa365314-e028-4889-a501-e8f2
```

Response

json

```
HTTP/1.1 204 No Content
```

Usage

To delete a specific comment associated with an issue in a project, send a DELETE request to the endpoint with the unique identifier (UUID) of the comment you want to delete. The response will indicate the successful deletion of the comment with a 204 No Content status code.

AUTHORIZATION Bearer Token

Token

GET Comments Project List

http://127.0.0.1:8000/api/projects/1/comments/

Comment Search and Filter

Description

This feature allows you to search and filter comments for a specific project's issues.

Request

- **Endpoint:** `http://127.0.0.1:8000/api/projects/2/comments/`
- **Method:** GET

Query Parameters

1. **unique_id (Search by Comment ID)**
    a. **Type:** String
    b. **Description:** Search for comments with a specific unique ID.
    c. **Example:** `unique_id=12345`
2. **description (Search by Description)**
    a. **Type:** String
    b. **Description:** Search for comments with a specific description.
    c. **Example:** `description=feedback`
3. **issue (Search by Issue Title)**

a. **Type:** String
b. **Description:** Search for comments related to issues with a specific title.
c. **Example:** `issue=Authentication Issue`

4. **author (Search by Author)**
   a. **Type:** String
   b. **Description:** Search for comments authored by a specific user (user's username).
   c. **Example:** `author=JohnDoe`

Example Request

Plain Text

```
http
```

Plain Text

```
GET http://127.0.0.1:8000/api/projects/2/comments/?unique_id=12345&description=feedback
```

Response

- **Status Code:** 200 OK

Response Body

Plain Text

```
json
```

Plain Text

```
{
    "message": "Listing comments included in the Authentication Project",
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "unique_id": "12345",
            "description": "User feedback on authentication",
            "issue": "Authentication Issue",
            "author": "JohnDoe"
```

Notes

- Use the query parameters to search and filter comments based on different criteria.
- You can use multiple query parameters to refine your search.

Get Comments for an Issue

Description

This endpoint allows you to retrieve a list of comments associated with a specific issue in a project. The response will provide details about the comments, including their unique IDs, descriptions, and author details.

Endpoint

- **Base URL:** http://127.0.0.1:8000/api
- **Endpoint:** `/projects/40/issues/7/comments/`
- **HTTP Method:** GET

Response

The response will include a message indicating the issue and project related to the comments, the total count of comments, and the list of comments themselves.

Response Body

json


```
{
  "message": "Listing comments included in the (ID:40 - BACK) project",
  "count": 6,
  "next": null,
  "previous": null,
  "results": [
    {
      "unique_id": "8c66b8cf-06c7-4968-986c-cc19a9811723",
      "description": "Testing comment",
      "author_details": {
        "id": 21,
```


The response message will include the issue's unique ID, the issue's title, the project's unique ID, and the project's type. It will also include the total count of comments related to the issue and a list of comment objects with their unique IDs, descriptions, and author details.

This format provides detailed information about each comment, allowing for efficient management and understanding of the comments associated with the issue.

AUTHORIZATION Bearer Token

Token