

Social Media API – Posts & Comments Documentation

This document describes how to interact with the **Posts** and **Comments** endpoints of the Social Media API. It includes authentication requirements, available endpoints, request/response examples, pagination, and filtering.

Authentication

All endpoints require **Token Authentication**.

Required Header

```
Authorization: Token <your_token_here>
```

If the token is missing or invalid, the API will return:

```
{  
  "detail": "Authentication credentials were not provided."  
}
```

POSTS ENDPOINTS

Base URL:

```
/api/posts/
```

List All Posts

Request

```
GET /api/posts/
```

Optional Query Parameters

- `page` – page number (default: 1)
- `page_size` – number of posts per page (max: 50)
- `search` – search by title or content

Example:

```
GET /api/posts/?page=1&page_size=5&search=django
```

Response (200 OK)

```
{
  "count": 12,
  "next": "http://localhost:8000/api/posts/?page=2",
  "previous": null,
  "results": [
    {
      "id": 1,
      "author": "john",
      "title": "Learning Django",
      "content": "Django is a powerful framework",
      "created_at": "2025-01-01T10:00:00Z",
      "updated_at": "2025-01-01T10:00:00Z"
    }
  ]
}
```



Retrieve a Single Post

Request

```
GET /api/posts/{id}/
```

Example:

```
GET /api/posts/1/
```

Response (200 OK)

```
{
  "id": 1,
  "author": "john",
  "title": "Learning Django",
  "content": "Django is a powerful framework",
  "created_at": "2025-01-01T10:00:00Z",
  "updated_at": "2025-01-01T10:00:00Z"
}
```

Create a New Post

Request

```
POST /api/posts/
```

Request Body

```
{
  "title": "My First Post",
  "content": "This is my first post content"
}
```

Response (201 Created)

```
{
  "id": 2,
  "author": "john",
  "title": "My First Post",
  "content": "This is my first post content",
  "created_at": "2025-01-02T09:30:00Z",
  "updated_at": "2025-01-02T09:30:00Z"
}
```



Update a Post

Request

```
PUT /api/posts/{id}/
```

Request Body

```
{
  "title": "Updated Title",
  "content": "Updated content"
}
```

Response (200 OK)

```
{
  "id": 2,
  "author": "john",
  "title": "Updated Title",
  "content": "Updated content",
  "created_at": "2025-01-02T09:30:00Z",
  "updated_at": "2025-01-02T10:00:00Z"
}
```



Delete a Post

Request

```
DELETE /api/posts/{id}/
```

Response (204 No Content)

No response body



COMMENTS ENDPOINTS

Base URL:

```
/api/comments/
```

List All Comments

Request

```
GET /api/comments/
```

Response (200 OK)

```
{
  "count": 5,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "post": 2,
      "author": "alice",
      "content": "Great post!",
      "created_at": "2025-01-02T11:00:00Z",
      "updated_at": "2025-01-02T11:00:00Z"
    }
  ]
}
```

Retrieve a Single Comment

Request

```
GET /api/comments/{id}/
```

Response (200 OK)

```
{
  "id": 1,
  "post": 2,
  "author": "alice",
}
```

```
"content": "Great post!",
"created_at": "2025-01-02T11:00:00Z",
"updated_at": "2025-01-02T11:00:00Z"
}
```

Create a Comment

Request

```
POST /api/comments/
```

Request Body

```
{
  "content": "This is a comment"
}
```

The post is automatically assigned by the backend logic (not provided by the client).

Response (201 Created)

```
{
  "id": 2,
  "post": 2,
  "author": "john",
  "content": "This is a comment",
  "created_at": "2025-01-02T12:00:00Z",
  "updated_at": "2025-01-02T12:00:00Z"
}
```

Update a Comment

Request

```
PUT /api/comments/{id}/
```

Request Body

```
{
  "content": "Updated comment text"
}
```

Response (200 OK)

```
{
  "id": 2,
  "post": 2,
  "author": "john",
  "content": "Updated comment text",
  "created_at": "2025-01-02T12:00:00Z",
  "updated_at": "2025-01-02T12:30:00Z"
}
```



Delete a Comment

Request

```
DELETE /api/comments/{id}/
```

Response (204 No Content)

No response body



Notes

- All list endpoints are **paginated**
- Posts support **search filtering** by `title` and `content`
- Authentication is required for all operations
- Authors are automatically set from the authenticated user

✓ This documentation fulfills the requirement to describe all Posts and Comments endpoints with request and response examples.