



# Project Title: Blog Platform API



## Project Overview

A blogging backend that allows users to:

- Register/login,
- Create blog posts,
- Comment on others' posts,
- Like or unlike posts,
- View public blog posts with optional filters (e.g., by author, tags, or date).

It simulates a real-world content platform like **Medium**, **Hashnode**, or [Dev.to](#), but simplified.



## Core Entities & Relationships

### 1. User

- `id` (Primary Key)
- `username`
- `email`
- `password` (hashed)
- `bio` (optional)
- `created_at`



*Each user is the owner of their posts and comments. Users must be authenticated to post or comment.*

### 2. Post

- `id` (PK)
- `author_id` (FK to User)

- title
- slug (auto-generated from title)
- content
- is\_published (boolean)
- tags (array or comma-separated string)
- created\_at , updated\_at

 *Users can create, update, delete their own posts.*


### 3. Comment

- id (PK)
- post\_id (FK to Post)
- author\_id (FK to User)
- content
- created\_at

 *Comments are tied to both a post and a user.*

### 4. Like

- id (PK)
- post\_id (FK to Post)
- user\_id (FK to User)

 *Each user can like a post only once (unique constraint).*



## Core API Endpoints



## Authentication

- POST /register → Register new user
- POST /login → Issue JWT token
- GET /me → Get profile of current user



## Posts

- GET /posts → List published posts (public)
  - Supports pagination, search, and filtering:  
?author=John&tag=tech&page=2
- GET /posts/:slug → View a specific post (public)
- POST /posts → Create a new post (auth required)
- PUT /posts/:slug → Edit your post
- DELETE /posts/:slug → Delete your post
- GET /me/posts → View your own draft and published posts



## Comments

- GET /posts/:post\_id/comments → List comments on a post
- POST /posts/:post\_id/comments → Add comment (auth required)
- PUT /comments/:id → Edit your comment
- DELETE /comments/:id → Delete your comment



## Likes

- POST /posts/:post\_id/like → Like a post
- DELETE /posts/:post\_id/unlike → Unlike a post
- GET /posts/:post\_id/likes → Count or list of likes



## Optional Enhancements (Advanced)

Feature	Description
Soft Deletes	Don't delete posts/comments permanently – mark as deleted
Public Profiles	/users/:username – show user bio and their published posts
Bookmark Posts	Users can save posts to a personal reading list

# Learning Outcomes

Skill	How It's Applied
REST API Design	Clean, nested routes and HTTP methods
Authentication & Authz	JWT-based login, route protection
Pagination & Filtering	Efficient data listing
Relationships in DB	One-to-many (posts/comments), many-to-one
Input Validation	Using Pydantic or Joi for input schemas
Ownership-based Access	Only post/comment owners can update/delete