Blog Platform with AI Content Moderation

Developed by Swaan Maharjan

A Django-based web application with AI-driven content moderation

December 2025

Project Overview

A blog platform enabling users to create, view, and manage posts

Features AI-powered content moderation to flag inappropriate content

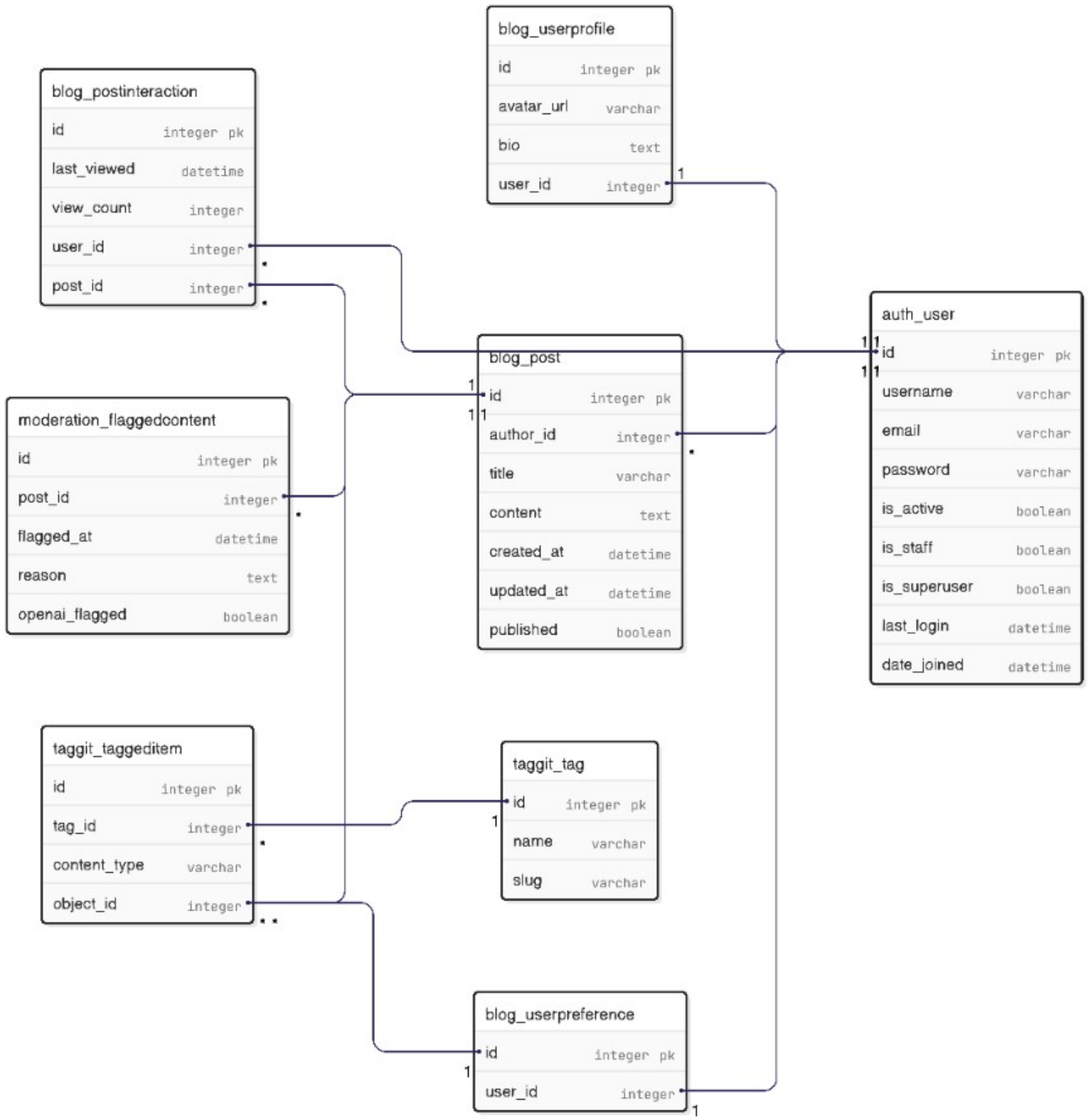
Built with Django, Bootstrap, and Python

Focus on user experience, security, and scalability

Problem Statement

Implementing ai content moderation system for autonomous moderation.
Implementing collaborative recommendation system using cosine similarity algorithm.
Implementing slug uniqueness handling to avoid repeating titles.

Class Diagram



Content Moderation

OpenAI based content moderation

Flagging system with confidence scores (e.g., 0.9)

Custom moderation dashboard (/admin/moderation/)

Stores flagged content with reasons and user details

Collaborative Filtering using Cosine Similarity

User Similarity Calculation (using Cosine Similarity): Each user's ratings are
treated as a vector in an n-dimensional space (where n is the number of items).
Cosine similarity then measures the cosine of the angle between two users' vectors.
The formula for cosine similarity between two users, u and v
, across common items Iuv,

$$sim(u, v) = \frac{\sum_{i \in Iuv} r_{u,i} \cdot r_{v,i}}{\sqrt{\sum_{i \in I_{uv}(} r_{ui})^2} \cdot \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi})^2}}$$

Slug uniqueness

The base slug is generated by the folowing code :

```
base_slug = normalize_and_slugify(title)
max_length = 60
if len(base_slug) > max_length:
    base_slug = truncate_to_last_word_boundary(base_slug, max_length)
```

then the following code is used for uniqueness verification loop:

```
candidate_slug = base_slug
counter = 1
while slug_exists_in_database(candidate_slug) do:
    if updating_existing_post and candidate_slug == post.current_slug:
        return candidate_slug # Allow post to keep its own slug
    suffix = "-" + str(counter)
    if length(base_slug) + length(suffix) > max_length:
        # Truncate base to fit suffix
        truncated = base_slug[0:(max_length - length(suffix))]
        truncated = remove_trailing_hyphen(truncated)
        candidate_slug = truncated + suffix
    else:
        candidate_slug = base_slug + suffix
    counter = counter + 1
return candidate_slug
```

Lessons Learned

Importance of precise implementation of OpenAI moderation API for moderation

Need for thorough URL pattern validation

Effective use of Django forms and widgets

Value of responsive design for user accessibility

Git best practices for managing project size

Conclusion

A robust blog platform with AI moderation

Enhanced user experience and admin functionality

Scalable foundation for future features

Ready for further development and deployment

Thank you for your attention!